

Liviu Ciortuz, Alina Munteanu, Elena Bădărău

Exerciții de învățare automată

2020

Liviu Ciortuz este actualmente conferențiar la Facultatea de Informatică a Universității „Alexandru Ioan Cuza“ din Iași. A absolvit Facultatea de Matematică-Informatică (1985) și a obținut doctoratul în informatică la Universitatea din Lille, Franța (1996). A lucrat în cercetare la Institutul de Informatică Teoretică al Academiei Române (filiala din Iași) între anii 1987 și 1990, la Institutul German de Cercetare în Inteligență Artificială (1997-2001), la Universitățile din York și Aberystwyth din Marea Britanie (2001-2003), precum și la Institutul INRIA de la Rennes, Franța (2012-2013).

Alina Munteanu a absolvit mai întâi Facultatea de Matematică-Informatică (licență, 2005, și master, 2008) și apoi Facultatea de Informatică (master, 2011) de la Universitatea „Alexandru Ioan Cuza“ din Iași. Din 2013 lucrează la Max Delbrück Center for Molecular Medicine din Berlin, iar în 2017 a obținut doctoratul în informatică la Universitatea Humboldt din Berlin, Germania.

Elena Bădărău a absolvit Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza“ din Iași (licență, 2008, și master, 2010).

Redactor: Eduard Dulman și Cerasela Cirimpei (pentru edițiile I și II)
Coperta: Manuela Oboroceanu

Pe copertă: ilustrație de Mihaela Romanică
ISBN:

© Autorii
<http://www.info.uaic.ro/~ciortuz/ML.ex-book/>
e-mail: ciortuz@info.uaic.ro

Liviu Ciortuz, Alina Munteanu, Elena Bădărău

Exerciții de învățare automată

Editia a treia revizuită și adăugită

Cuvânt înainte de Liviu Ciortuz

2020

Motto:

*„Tot ce găsește mâna ta să facă, fă cu toată puterea ta,
căci în locuința morților, în care mergi, nu mai este nici lucrare,
nici chibzuială, nici știință, nici înțelepciune.“*

Cartea Eclesiastului 9:10

Cuprins

Mulțumiri	9
Cuvânt înainte	10
1 Metode de regresie	19
1.1 Probleme rezolvate	21
1.1.1 Regresia liniară	21
1.1.2 Regresia logistică	49
1.2 Probleme propuse	69
1.2.1 Regresia liniară	69
1.2.2 Regresia logistică	80
2 Clasificare bayesiană	93
2.1 Probleme rezolvate	95
2.1.1 Ipoteze de probabilitate maximă a posteriori (MAP)	95
2.1.2 Algoritmii Bayes Naiv și Bayes Optimal	105
2.1.3 Clasificare bayesiană gaussiană	131
2.2 Probleme propuse	149
2.2.1 Ipoteze de probabilitate maximă a posteriori (MAP)	149
2.2.2 Algoritmii Bayes Naiv și Bayes Optimal	150
2.2.3 Clasificare bayesiană gaussiană	157
3 Învățare bazată pe memorare — Algoritmul k-NN	167
3.1 Probleme rezolvate	168
3.2 Probleme propuse	194
4 Arbori de decizie	205
4.1 Probleme rezolvate	209
4.1.1 Algoritmul ID3	209
4.1.2 Algoritmul AdaBoost	257
4.2 Probleme propuse	298
4.2.1 Algoritmul ID3	298
4.2.2 Algoritmul AdaBoost	315
5 Mașini cu vectori-suport	329
5.1 Probleme rezolvate	332
5.1.1 SVM cu margine “hard”	332
5.1.2 SVM cu margine “soft”	357
5.1.3 Alte probleme de optimizare de tip SVM	391
5.2 Probleme propuse	405
5.2.1 SVM cu margine “hard”	405
5.2.2 SVM cu margine “soft”	412
5.2.3 Alte probleme de optimizare de tip SVM	420

6 Rețele neuronale artificiale	431
6.1 Probleme rezolvate	433
6.1.1 Chestiuni introductive	433
6.1.2 Unități neuronale — algoritmi de antrenare	450
6.1.3 Rețele “feed-forward” — algoritmul de retropropagare	472
6.2 Probleme propuse	489
6.2.1 Chestiuni introductive	489
6.2.2 Unități neuronale — algoritmi de antrenare	491
6.2.3 Rețele “feed-forward” — algoritmul de retropropagare	497
7 Clusterizare	511
7.1 Probleme rezolvate	517
7.1.1 Clusterizare ierarhică	517
7.1.2 Algoritmul K -means	532
7.1.3 Algoritmul EM pentru modele de mixturi gaussiene	551
7.2 Probleme propuse	590
7.2.1 Clusterizare ierarhică	590
7.2.2 Algoritmul K -means	598
7.2.3 Algoritmul EM pentru modele de mixturi gaussiene	609
8 Algoritmul EM	629
8.1 Probleme rezolvate	631
8.2 Probleme propuse	695
9 Modele Markov ascunse	707
9.1 Probleme rezolvate	708
9.2 Probleme propuse	731

Mulțumiri

O mare parte dintre exercițiile și problemele din această culegere au fost date la examenele sau temele pentru acasă de la cursurile de învățare automată ținute în anii 1994-2019 la Carnegie Mellon University (CMU) din Pittsburgh, Statele Unite ale Americii, de către profesorii Tom Mitchell, Andrew Moore, Carlos Guestrin, Geoff Gordon, Eric Xing, William Cohen, Ziv Bar-Joseph, Aarti Singh, Roni Rosenfeld, Alex Smola, Barnabás Póczos, Seyoung Kim, Nina Balcan și Matt Gormley. Alte exerciții și probleme provin de la cursul de învățare automată al profesorului Andrew Ng de la Universitatea Stanford, precum și cel al profesorilor Tommi Jaakkola, Regina Barzilai și Leslie Kaelbling de la MIT, S.U.A.

Facem mențiunea că, în general, rezolvările care apar în prezenta culegere au un nivel de detaliere semnificativ mai elaborat decât rezolvările originale, prezentate adeseori doar în mod esențializat de către profesorii mai sus menționați și/sau asistenții dânsilor.

Tuturor acestora le aducem pe această cale cele mai sincere mulțumiri pentru generozitatea cu care au pus la dispoziție pe internet aceste materiale didactice.

Mai mulți studenți de la Facultatea de Informatică a Universității „Alexandru Ioan Cuza“ din Iași ne-au ajutat de-a lungul ultimilor ani la redactarea textelor sau a imaginilor pentru unele probleme. Îi menționăm în ordine alfabetică: Gheorghe Balan, Ciprian Băetu, Cristian Budăianu, Giorgiana Caltais, Petru Cehan, Ivona Chili, Sebastian Ciobanu, Lidia Corciova, Oana Cotoman, Denis Crusos, Sergiu Dinu, Ahmad Cezar El-Nazli, Oana Floorean, Eugen Goriaic, Irina Grosu, Andrei-Constantin Iacob, Anca Luca, Diana Lucaci, Ștefan Matcovici, Dorin Miron, Alexandru Mitan, Andi Munteanu, Oriana Oniciuc, Mihaela Potlog, Andreea Prodan, Mădălina Racoviță, Elena-Irina Radu, Ciprian Recianu, Cristian Rotaru, Irina Roznovăț, Alexandra Sbiera, Marius Spănu, Andreea Stanciu, Cristina Șerban, Octavian Tamaș, Călin-Rareș Turliuc și Liana-Ștefania Țucăr. (Ne cerem sincer scuze pentru eventualele omisiuni.)

De asemenea, alături de Ștefan Panțiru, Anca Ignat, Mihaela Breabă și Adrian Zălinescu, care au ținut seminarii, mai mulți studenți — dintre care îi menționăm pe Vlad Aioanei, Mihai Baboi, Sebastian Ciobanu, Andrei Cioromilă, Cristian Cojocaru, Rareș Dima, Alexandru Dobranici, Marius Georgică, Mihai Grigoriță, Sorin Grițco, Mircea Irimescu, Diana Lucaci, Ștefan Matcovici, Alexandru Mărtinaș, Alexandra Minghel, Ionuț-Vlad Modoranu, Alexandru Oloieri, Petru Munteanu, Lucian Neștian, Lucian Nevoe, Axenia Niță, Iulian Pichiu, Codrina Prisecaru, Emanuel Pușcașu, Mihaela Radu, Carol Rameder, Roxana Sălăvăstru, Raluca Scorțanu, Cristian Simionescu, Nicolae Șoitu, Ionel Ștefanucă, Silviu Șutea-Drăgan, Iulian Vâscu și Silviu Vițel —, precum și domnul Liviu Olaru și conf. dr. Elena Nenciu, au identificat anumite erori în versiuni preliminare ale prezentei culegeri.

Mulțumim domnilor Ștefan Panțiru și Daniel Munteanu pentru instalarea programelor necesare pentru tehnoredactare. Mulțumim redactorilor de carte Eduard Dulman și Cerasela Cirimeș, precum și doamnelor Dana Lungu și Luminița Răducanu de la Editura Universității „Alexandru Ioan Cuza“ din Iași pentru asistența acordată în vederea publicării edițiilor precedente ale acestei cărți. Mulțumiri speciale se cuvin Mihaiei Romanică din Antwerpen, Belgia, care ne-a pus la dispoziție desenele care apar în deschiderea fiecărui capitol, precum și desenul de pe copertă.

Liviu Ciortuz îi mulțumește pe această cale profesorului Cristian Preda de la Universitatea din Lille și INRIA – Nord Europe pentru posibilitatea de a lucra acolo la finalizarea primei ediții a acestei culegeri, în luna iulie 2015.

Cuvânt înainte — de Liviu Ciortuz

Așa cum se cuvine, vom preciza aici următoarele chestiuni: de ce anume a fost scrisă această culegere, cum a fost ea alcătuită în timp, care au fost criteriile de selecție atât la nivel tematic — adică, de ce au fost alese respectivele capitole de învățare automată și nu altele — cât și la nivelul tipului / tipurilor de exerciții incluse, cum se situează această culegere în raport cu alte culegeri din aria învățării automate, care este suportul bibliografic,¹ documentele on-line care o însoțesc — site-ul asociat, slide-uri, un index tematic și un „companion“ de tip probleme de implementare —, precum și ideile pe care le avem acum cu privire la o posibilă extindere ulterioară a culegerii.

Cartea aceasta s-a născut adunând mai întâi exercițiile care au fost date la examenele și teste „pe parcurs“ din cadrul cursului de *Învățare automată* de la Facultatea de Informatică a Universității „Alexandru Ioan Cuza“ din Iași. Cursul acesta s-a ținut începând din anul 2003.² Atunci când s-au acumulat un număr semnificativ de exerciții traduse în limba română, a fost limpede că ar fi de mare ajutor ca rezolvările acestor exerciții să fie puse la dispoziția studenților din seriile noi, sub forma unei culegeri. Pentru aceasta, am socotit că este benefic să apelez la ajutorul unora dintre cei mai buni studenți care au absolvit deja cursul. Astfel, mai întâi Elena Bădărău (studentă la master, seria 2008–2010) și apoi, într-o măsură considerabil sporită, Alina Munteanu (studentă la master, seria 2009–2011, iar după aceea doctorand) au redactat soluții (în general, semnificativ mai extinse față de original), la multe dintre problemele pe care le culesesem anterior.

De atunci și până acum am cules, am tradus și chiar am formulat multe alte exerciții, cu scopul acoperirii unei arii suficient de largi pentru fiecare dintre capitolele predate la curs. Fiecare capitol din culegere a fost împărțit într-o secțiune de probleme în probleme rezolvate și o secțiune de probleme propuse. În ultimii ani, după ce cele două co-autoare au plecat din facultate,³ am redactat multe soluții și am realizat șlefuirea întregului material. De-a lungul acestor ani, am prezentat la cursurile de învățare automată următoarele capitole (selecția și ordinea lor de succesiune variind ușor de la an la an): fundamente matematice — și anume, probabilități și statistică, estimarea parametrilor unor distribuții probabiliste, teoria informației, funcții-nucleu și metode de optimizare (toate aceste

¹ Este vorba despre cartea sau cărțile care sunt mai potrivite pentru a fi folosite în tandem cu această culegere și, foarte posibil, pentru cursul aferent.

² În perioada 2003–2008 el a fost curs opțional la ciclul de licență, anul IV. Apoi, odată cu trecerea la sistemul Bologna, el a devenit curs obligatoriu la ciclul de master. Ulterior, două dintre cele cinci specializări / programe de master de la facultatea noastră l-au pus la dispoziția studenților drept curs obligatoriu, iar celelalte trei l-au păstrat drept curs opțional.

Începând din anul 2015, cursul acesta s-a scindat într-un curs de bază (obligatoriu) în cadrul ciclului de licență, la anul III, și un curs avansat (sub denumirea de *Capitole speciale de Învățare automată*) la master.

În anii universitari 2018–2019 și 2019–2020 cursul de bază de *Învățare automată* a fost ținut sub formă de curs opțional și la masterul de *Matematici aplicate* de la Facultatea de Matematică a Universității „Alexandru Ioan Cuza“ din Iași. De asemenea, în semestrul întâi al anului universitar 2012–2013 am ținut cursul de *Învățare automată* în regim modular (timp de 4 săptămâni) și la studenții de la master, specializarea informatică, de la Facultatea de Matematică-Informatică a Universității de Vest din Timișoara.

³ După absolvirea masterului, Elenă Bădărău a lucrat pentru un timp la compania Amazon din Iași. În perioada 2015–2017, dânsa a avut amabilitatea de a ține seminarii la cursul de *Învățare automată* de la ciclul de licență din facultatea noastră. Alina Munteanu a început doctoratul la Iași și apoi l-a continuat la *Max Delbrück Center for Molecular Medicine* din Berlin, Germania, folosind tehnici de învățare automată în rezolvarea unor probleme de cercetare în bioinformatică. Dânsa și-a susținut teza de doctorat la *Universitatea Humboldt* din Berlin în octombrie 2017.

secțiuni având rol de suport pentru capitolele următoare) —, modele de regresie, clasificare bayesiană, învățare bazată pe memorare, arbori de decizie, mașini cu vectori-suport, rețele neuronale artificiale, clusterizare, schema algoritmică EM, modele Markov ascunse, și, în sfârșit, teoria învățării computaționale.

În actuala versiune a culegerii apar exerciții pentru primele zece capitole dintre cele unsprezece enumerate mai sus. Dintre acestea, capitolul 0 (anumite secțiuni) și capitolele 2-4 și 7 corespund acum programei cursului de *Învățare automată* de la licență, iar restul capitolelor corespund programei cursului de *Capitole speciale de Învățare automată* de la programul de master de la facultatea noastră.⁴ Pe lângă adăugarea de noi capitole, ne propunem ca pe viitor să extindem culegerea prin introducerea de noi exerciții care să acopere „segmente” care nu sunt prezente în capitolele deja elaborate. Ne gândim, de exemplu, la unele exerciții legate de analiza componentelor principale (engl., *Principal Component Analysis*), clusterizare spectrală (engl., spectral clustering)), precum și analiza prin factori (engl., factor analysis).

Majoritatea exercițiilor din această culegere sunt destinate a fi rezolvate „cu creionul / pe hârtie”. Ele solicită în general fie *aplicarea* principaliilor algoritmi din capitolele / domeniile de învățare automată care au fost menționate mai sus, fie *demonstrarea* unor proprietăți (teoretice) ale acestor algoritmi sau ale conceptelor utilizate de ei.

Remarcăm modul în care au fost și sunt în general concepute exercițiile cu caracter teoretic de la cursurile de *Machine Learning* de la CMU. și anume, ele călăuzesc pas cu pas studentul în aşa fel încât să ajungă să demonstreze el însuși (cu încredere crescândă în forțele proprii) rezultate importante. În acest fel se micșorează sau chiar se elimină efortul necesar pentru memorarea integrală a acestor demonstrații, care devin astfel mai degrabă obiect de lucru acasă sau în grup decât de prezentare abstractă și aridă la curs. Pe această cale se realizează o conlucrare foarte bine mediată între profesor (și / sau asistent) pe de o parte și student pe de altă parte, pe baza acestor texte de tip problemă / exercițiu, bine concepute și elaborate având în minte acest scop didactic.

În alcătuirea acestei culegeri am lăsat în mod intenționat de o parte problemele cu specific de implementare. Am procedat aşa din două motive. În primul rând, există suficient de multe cărți de specialitate care se ocupă de astfel de chestiuni. În al doilea rând, am început deja să creăm pentru studenții noștri un repertoriu (engl., *repository*) de seturi de date, implementări și scripturi pentru cele mai instructive probleme practice date la lucrările pentru acasă (engl., *homeworks*) de la cursurile de *Machine Learning* de la CMU și eventual de la alte universități. Corespondent, este în curs avansat de elaborare (în limba engleză) o fasciculă care constituie un *Companion practic* pentru prezenta culegere.

Culegerea are un *site* asociat: <http://profs.info.uaic.ro/~ciortuz/ML.ex-book/>. Acolo se găsesc — pe lângă unele *seturi de date* folosite la exerciții pentru care au fost create grafice ce pun în evidență comportamentul unor algoritmi de învățare automată pe care îi studiem — *slide-uri* pentru cele mai utile exerciții din culegere. Menționăm că multe dintre aceste slide-uri au fost folosite nu doar la seminarii ci și la curs, pentru exemplificarea aplicării algoritmilor predate, demonstrarea unor proprietăți importante ale lor etc. Sperăm că acest set de slide-uri se va mări în timp și că ele vor fi folosite și de cadre didactice de la alte facultăți. Majoritatea slide-urilor au fost redactate *în limba engleză* (spre deosebire de culegere!), pentru ca de ele să beneficieze un număr cât mai mare de studenți.⁵

⁴Capitolul de *Teoria învățării computaționale* (engl., Computational learning theory) se află acum într-un stadiu incipient de elaborare. Sperăm că vom avea răstimpul necesar pentru finalizarea lui și atunci acest capitol vor fi adăugat la culegere într-o ediție viitoare.

⁵La fiecare dintre exercițiile pentru care am făcut deja slide-uri, am pus în antet — mai precis,

Cuvânt înainte

Ca *suport bibliografic* la curs am folosit cu preponderență în primii ani de predare carteia *Machine Learning* a profesorului Tom Mitchell de la CMU (ed. McGraw-Hill, 1997). Deși nu mai este de dată recentă, această carte are un stil ușor accesibil studenților noștri. Menționăm că, dintre cele unsprezece capituloare enumerate mai înainte, carteia profesorului Tom Mitchell nu conține capituloare de fundamente, clusterizare, mașini cu vectori-suport și modele Markov ascunse (dar conține alte capituloare, precum învățare bazată pe ranforsare și algoritmi genetici). Pentru trei dintre cele patru capituloare pe care tocmai le-am menționat,⁶ am folosit carteia *Foundations of Statistical Natural Language Processing* (Christopher Manning, Hinrich Schütze, MIT Press, 2002).⁷ Pe parcursul acestor ani de predare, am făcut referire din ce în ce mai mult și la notițele de curs (engl., *Lecture notes*) ale renomitului profesor Andrew Ng de la Stanford University, în care se folosesc adeseori formalizări matematice mai sofisticate.⁸

La începutul fiecărui capitol din carte am alcătuit un *sumar*, sau o *privire de ansamblu*, un conspect (engl., *overview*) asupra capitolului respectiv. El poate fi văzut și ca un *index de probleme*. Ansamblul acestor conspective constituie un rezumat al cursului / cursurilor de *Învățare automată* de la facultatea noastră. Pentru aproape fiecare chețiune care se pretează să fie predată la curs, acest *sumar* / *index* indică unul sau mai multe exerciții care se referă la (sau folosesc, exemplifică) chețiunea respectivă.

Sintetizând, putem spune că această culegere este o încercare de a pune împreună în mod unitar exerciții cu caracter preponderent conceptual și teoretic pentru principalele capituloare de învățare automată. Ea folosește un aparat matematic care este bazat în special pe notiunile de matematică din liceu și de la cursurile de matematică și de probabilități din primul an de facultate.

Remarcăm că, pe de o parte, există cărți conținând exerciții specializate pe [câte] un capitol de învățare automată, de exemplu *Regression Analysis by Example* (Samprit Chatterjee, Ali S. Hadi, Wiley-Interscience, 2006) și *Reinforcement Learning: An Introduction* (Richard S. Sutton, Andrew G. Barto, MIT Press, 1998). Însă, pe de altă parte, nu știm să existe o culegere similară culegerii noastre, având o arie suficient de vastă, cu exerciții situate pe două pale, unul exemplificativ iar celălalt conceptual și teoretic, în așa fel încât (foarte important!) să fie adecvate pentru nivelul studenților de la facultatea noastră. Trebuie spus că exercițiile din cărțile consacrate în domeniul învățării automate, care acoperă un număr mare de capituloare — de exemplu *Pattern Classification* (Richard O. Duda, Peter E. Hart, David G. Stork, Wiley-Interscience, 2003), *Pattern Recognition and Machine Learning* (Christopher M. Bishop, Springer, 2006), *The Elements of Statistical Learning. Data Mining, Inference and Prediction* (Trevor Hastie, Robert Tibshirani, Jerome Friedman, Springer, 2009), precum și *Pattern Recognition* (Sergios Theodoridis, Konstantinos Koutroumbas, Academic Press, 2009) —, folosesc adeseori un aparat matematic destul de complex. Rolul culegerii noastre este de a fi o „punte“ înspre nivelul

acolo unde se indică *subiectul* și *sursa* respectivului exercițiu — semnul ■.

⁶Excepție face capitolul despre mașini cu vectori-suport (SVM). O foarte bună prezentare a acestui domeniu este făcută în carteia *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* de Nello Cristianini și John Shawe-Taylor, publicată la editura Cambridge University Press, în anul 2000.

⁷Alegerea acestei cărți se datorează experienței pe care am acumulat-o mai înainte în acest domeniu (NLP), asupra căruia învățarea automată a avut un foarte puternic impact în ultimele două decenii, iar acest fapt este bine reflectat în carteia menționată.

⁸Pentru studenții cei mai sărguincoși, care vor să se inițieze în cercetarea din domeniul învățării automate de tip „shallow“, recomandăm acum carteia *Machine Learning — A Probabilistic Perspective* de Kevin P. Murphy, publicată la editura MIT Press în anul 2012.

cărților pe care tocmai le-am menționat,⁹ astfel încât studenții noștri cei mai buni să poată să acceadă la acel nivel.¹⁰

Experiența pe care am acumulat-o în ultimii ani, în care am folosit deja culegerea, dotând studenții de la cursurile noastre de *Învățare automată* cu drafturi preliminare ale ei, ne arată că exercițiile din această carte sunt suficiente nu doar pentru două cursuri de câte un semestru (având săptămânal două ore de predare propriu-zisă și două ore de seminar), ci și pentru seminarii suplimentare / avansate, cu un grup de studenți de “top”.

Înainte de a încheia, trebuie precizate câteva *detalii tehnice*:

1. La folosirea ghilimelelor, am utilizat “ ” pentru expresii în limba engleză, respectiv „ “ pentru expresii românești, conform regulilor de ortografie corespunzătoare.
2. Pentru numere zecimale, din pură conveniență, am folosit în general notația de tip anglo-saxon (adică, de exemplu, 1.5 în loc de 1,5, cum ar fi trebuit), datorită faptului că am tehnoredactat în paralel două versiuni (engleză și română) ale culegerii, iar formulele matematice sunt (deocamdată) comune pentru cele două versiuni.
3. Menționăm că, tot din motive de conveniență la tehnoredactare, figurile (dar și tabelele) din carte nu au fost numerotate și, în consecință, fiecare figură a fost plasată în imediata vecinătate a textului care se referă la ea.
4. Precizăm că atunci când, în cadrul unei probleme (fie al enunțului, fie al soluției), ne vom referi la un anumit punct al problemei respective, îl vom desemna printr-unul din caracterele italice *a*, *b*, *c* etc, deși la începutul punctului respectiv apare litera scrisă sub forma uzualei: *a*, *b*, *c* etc. De asemenea, atunci când, într-o anumită problemă, ne vom referi la un punct al unei alte probleme, vom scrie pe scurt, de exemplu, „vedeți problema 14.b“; se va înțelege că facem trimitere la punctul *b* al problemei 14.

Atât eu cât și ceilalți doi coautori ai acestei culegeri ne dorim ca vastul efort depus pentru alcătuirea ei — în primul rând, de către autorii exercițiilor originale dar, cu modestia de rigoare, și de către noi — să permită generațiilor actuale și viitoare de studenți și doctoranți să-și însușească cunoștințele de bază din domeniul învățării automate, care a ajuns azi la un nivel impresionant de aplicabilitate, atât în industria IT cât și în cercetarea științifică.

Mențiune privind ediția a III-a (septembrie 2020)

La prezenta ediție, în raport cu ediția a doua, care a fost publicată online pe 25 septembrie 2019, am adăugat un număr semnificativ de probleme noi (printre care menționăm

⁹Pentru soluții la exercițiile din cărțile enumerate, vedeți: *Solution Manual to accompany Pattern Classification (2nd ed.) by R. O. Duda, P. E. Hart and D. G. Stork* (David G. Stork, 2001), *Pattern Recognition and Machine Learning Solutions to the Exercises: Web-Edition* (Markus Svensén and Christopher M. Bishop, 2009), *A Solution Manual and Notes for: The Elements of Statistical Learning by Jerome Friedman, Trevor Hastie, and Robert Tibshirani* (John L. Weatherwax, David Epstein, http://waxworksmath.com/Authors/G_M/Hastie/hastie.html, 21 June 2013), *Notes and Solutions for: Pattern Recognition by Sergios Theodoridis and Konstantinos Koutroumbas* (John L. Weatherwax, http://waxworksmath.com/Authors/N_Z/Theodoridis/theodoridis.html, October 17, 2015).

¹⁰O culegere de exerciții de tip implementare în domeniul învățării automate este următoarea: *Introduction to Pattern Recognition: A Matlab Approach* (Sergios Theodoridis, Aggelos Pikrakis, Konstantinos Koutroumbas, Dionisis Cavouras, Academic Press, 2010). Pentru exemplificări pe aceeași „linie“, vedeți și *Computational Statistics Handbook with Matlab* (Wendy Martinez, Angel Martinez, CRC Press, 2007, 2015).

Cuvânt înainte

În special pe cele referitoare la clasificare bayesiană gaussiană, precum și la modelele liniare generalizate), am introdus capitolul de modele Markov ascunse și am transferat secțiunea de probleme pentru estimarea parametrilor unor distribuții probabiliste uzuale de la capitolul al doilea (care acum este destinat exclusiv metodelor de regresie) la primul capitol, cel de *Fundamente matematice pentru învățarea automată*. Apoi, întrucât acest capitol de *Fundamente* a ajuns să numere circa 200 de pagini, am decis să-l transformăm într-un volum distinct, cu titlul de *Fundamente matematice pentru învățarea automată: exerciții*.¹¹

Ori de câte ori în acest volum se face referire la „capitolul“ de *Fundamente*, se va înțelege că este vorba despre volumul de Fundamente matematice ale învățării automate.

Liviu Ciortuz

Iași, noiembrie 2020

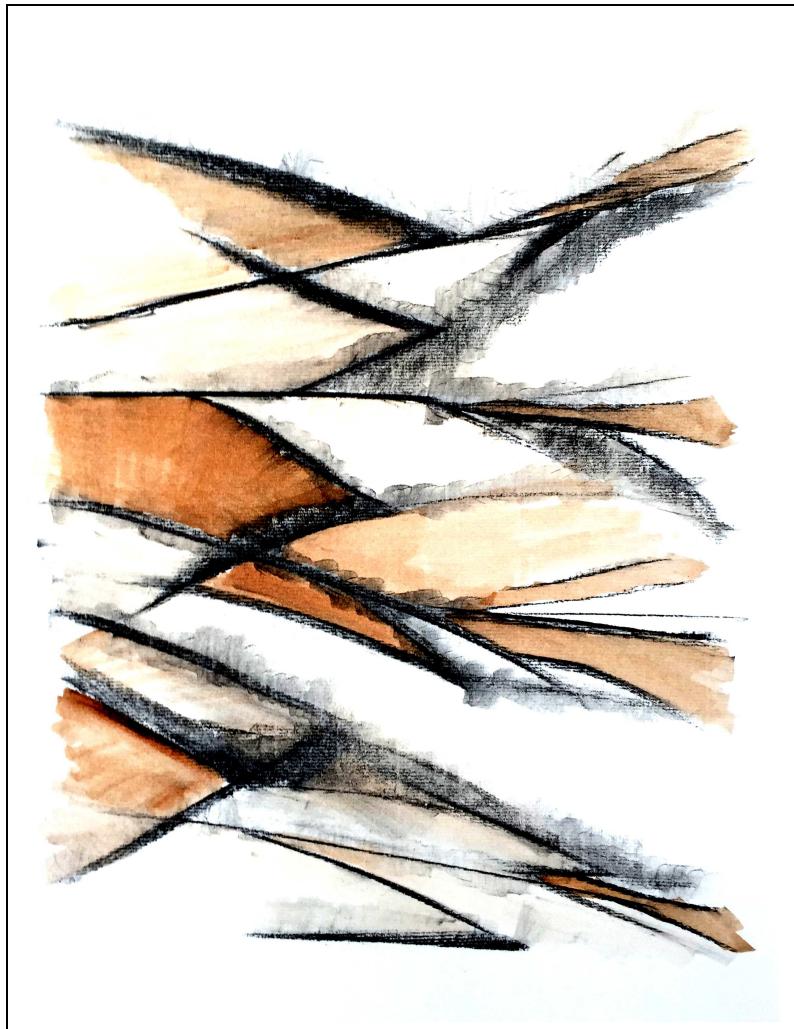
¹¹ Acest volum poate fi folosit în tandem — măcar în parte! — cu recenta carte *Mathematics for Machine Learning*, de M. Deisenroth, A. Faisal și C. Ong, publicată la Cambridge University, <https://mml-book.com>.

“Puerile as such an exercise may seem, it sharpens the faculties of observation, and teaches one where to look and what to look for.”

Sherlock Holmes in *Study in Scarlet*,

cited by Wolfgang Karl Härdle and Zdeněk Hlávka
in *Multivariate Statistics — Exercises and Solutions*,
second edition, Springer, 2015

Această pagină a fost lăsată liberă în mod intenționat.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

1 Metode de regresie

Sumar

Noțiuni preliminare

- estimarea parametrilor unor distribuții probabiliste uzuale (în special distribuția Bernoulli, distribuția gaussiană, distribuția Laplace); vedeți secțiunea corespunzătoare de la cap. *Fundamente*;
- elemente de calcul vectorial (în particular, produsul scalar) și de calcul matriceal: ex. 33 de la cap. *Fundamente*; norma L_2 (euclidiană) și norma L_1 : ex. 3, ex. 25, ex. 26; calculul derivatelor parțiale [de ordinul întâi și al doilea]: ex. 7; reguli de derivare cu argumente vectoriale: ex. 1;
- metode de optimizare (în spătă pentru aflarea maximului / minimului unei funcții reale, derivabile): metoda analitică, metoda gradientului, metoda lui Newton; exemplificare: ex. 72 de la capitolul de *Fundamente*.

Regresia liniară

- prezentarea *generală* a metodei regresiei liniare:¹²
 - MLE și corespondența cu estimarea în sens LSE (least squared errors): ex. 3.A; particularizare pentru cazul unidimensional: ex. 1.ab, ex. 19; exemplificare pentru cazul unidimensional (ex. 2, ex. 20) și pentru cazul bidimensional (ex. 22.a, ex. 30);
 - (P1) *scalarea atributelor* nu schimbă predicțiile obținute (pentru instanțele de test) cu ajutorul *formulelor analitice*: ex. 4, 37.a;
 - (P2) adăugarea de noi trăsături / atrbute nu mărește suma pătratelor erorilor: ex. 24;
 - o proprietate surprinzătoare a regresiei liniare: adăugarea câtorva „observații“ suplimentare poate conduce la modificarea radicală a valorilor optime ale parametrilor de regresie: CMU, 2014 fall, Z. Bar-Joseph, W. Cohen, HW2, pr. 4;
 - [rezolvarea problemei de] regresie liniară folosind *metoda lui Newton*: ex. 7;
 - MAP și corespondența cu *regularizarea* de normă L_2 (regresia *ridge*): ex. 3.C; particularizare pentru cazul unidimensional: ex. 1.c;
 - *regularizarea* de normă L_1 (regresia *Lasso*): ex. 25.a;
 - (P3) efectul de diminuare a ponderilor (engl., weight decay) în cazul regularizării de normă L_2 (respectiv L_1) a regresiei liniare, în comparație cu cazul neregularizat: ex 25.b;
- bias-ul și [co]varianța estimatorului regresiei liniare; bias-ul regresiei *ridge*: ex. 5;
- regresia polinomială [LC: mai general: folosirea așa-numitelor *funcții de bază*]: ex. 3.B; exemplificare pentru cazul bidimensional: CMU, 2015 spring, T. Mitchell, N. Balcan, HW4, pr. 1;

¹²În mod implicit, în această secțiune se va considera că termenul-zgomot este modelat cu distribuția gaussiană (dacă nu se specifică altfel, în mod explicit).

- cazul regresiei liniare cu termen de regularizare L_2 (regresia *ridge*):
deducerea regulilor de actualizare pentru *metoda gradientului ascendent*: varianta “batch” / “steepest descent”: ex. 6.a;
și varianta stochastică / secvențială / “online”: ex. 6.b; exemplu de aplicare: ex. 23;
- cazul regresiei liniare cu termen de regularizare L_1 (regresia Lasso):
rezolvare cu *metoda descreșterii pe coordonate* (engl., “coordinate descent”): ex. 26;
rezolvare cu *metoda sub-gradientului* (aplicare la selecția de trăsături): CMU, 2009 fall, C. Guestrin, HW2, pr. 2;
- regresia liniară în cazul zgomotului modelat cu distribuția *Laplace* (în locul zgomotului gaussian): ex. 8.B;
exemplificare pentru cazul bidimensional: ex. 22.c;
rezolvare în cazul unidimensional [chiar particularizat] cu ajutorul derivatei, acolo unde aceasta există: ex. 27;
 - regresia liniară și *overfitting-ul*: ex. 11;
 - regresie liniară folosită pentru *clasificare*: exemplificare: ex. 30;
- cazul *multivaluat* al regresiei liniare, reducerea la cazul uninomial: ex. 29;
- regresia liniară cu regularizare L_2 (regresia *ridge*), *kernel-izarea* ecuațiilor „normale“: ex. 9; (P4) folosind nucleu RBF, eroarea la antrenare devine 0 atunci când parametrul de regularizare λ tinde la 0: ex. 10;
- *regresia liniară ponderată*: ex. 8.A;
particularizare / exemplificare pentru cazul bidimensional: ex. 22.b;
 - proprietate a regresiei liniare local-ponderate [demonstrată în cazul unidimensional]: „netezirea“ liniară: ex. 28; cazul multivaluat, cu regularizare L_2 : Stanford, 2015 fall, Andrew Ng, midterm, pr. 2;
 - *regresia liniară (kernelizată) local-ponderată*, neparametrică:
particularizare / exemplificare pentru cazul unidimensional, cu nucleu gaussian: CMU, 2010 fall, Aarti Singh, midterm, pr. 4.

Regresia logistică

- prezentare generală,
- (•) calculul funcției de log-verosimilitate, estimarea parametrilor în sens MLE, folosind *metoda gradientului* (i.e., deducerea regulilor de actualizare a parametrilor): ex. 12, 37.b; particularizare pentru cazul datelor din \mathbb{R}^2 : ex. 31 (inclusiv *regularizare L_1* / estimarea parametrilor în sens MAP, folosind o distribuție a priori Laplace);
- (P0) *granița de decizie* pentru regresia logistică: ex. 31.d;
- (P1) funcția de log-verosimilitate în cazul regresiei logistice este concavă (deci are un maxim global), fiindcă matricea hessiană este pozitiv definită: ex. 13;
Observație: Demonstrația furnizează tot ce este necesar pentru obținerea [ulterioră a] relației de actualizare a parametrilor la aplicarea *metodei lui Newton* în cazul regresiei logistice;
- (P2) analiza efectului duplicării atributelor: ex. 32;
- (P3) efectul de diminuare a ponderilor (engl., weight decay) în cazul regularizării de normă L_2 a regresiei logistice — adică la estimarea parametrilor în sens MAP, folosind ca distribuție a priori distribuția gaussiană multidimensională sferică —, în comparație cu cazul estimării parametrilor în sensul MLE: ex. 14;

- *Variante / extensii* ale regresiei logistice:
 - regresia logistică local-ponderată, cu regularizare L_2 : calcularea vectorului gradient și a matricei hessiene (necesare pentru aplicarea metodei lui Newton în acest caz): ex. 15;
 - regresia logistică kernel-izată: adaptarea metodei gradientului: ex. 16;
 - regresia logistică n -ară (așa-numita regresie *softmax*): calculul funcției de log-verosimilitate, cu regularizare L_2 , deducerea regulilor de actualizare a ponderilor, folosind metoda gradientului: ex. 17; (P4) echivalența cu un anumit tip de mixtură de distribuții gaussiene multidimensionale: ex. 34;
- (P5) o [interesantă] proprietate comună pentru regresia liniară și regresia logistică: ex. 33;
- întrebări (cu răspuns A/F) cu privire la aplicarea *metodei lui Newton* comparativ cu *metoda gradientului* (în contextul rezolvării problemelor de regresie liniară și / sau regresie logistică): ex. 37.c;
- comparații între regresia logistică și alți clasificatori (Bayes Naiv, ID3): ex. 31.c, ex. 35.ab;
- (P6) teorema de reprezentare: ex. 18, ex. 36.

Modele liniare generalizate (GLM)

- condiții suficiente pentru *concavitatea* funcției de log-verosimilitate: ex. 40;
- particularizare pentru cazul distribuției geometrice: ex. 38;
- particularizare pentru cazul distribuției gaussiene unidimensionale: ex. 39.

1.1 Probleme rezolvate

1.1.1 Regresia liniară

1. (Estimare [în sens MLE și MAP] pentru parametrul unui model de regresie liniară unidimensională fără termen liber, în prezența unei componente-zgomot modelate de o distribuție gaussiană)
 - CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, midterm, pr. 3
 - CMU, 2011 spring, Tom Mitchell, midterm, pr. 4

Introducere: Obiectivul acestei probleme este acela de a face cunoștință cu metoda / problema de regresie liniară într-una dintre cele mai simple variante ale sale: cazul unidimensional. Aceasta înseamnă că variabila de ieșire (Y) depinde de o singură variabilă de intrare (X),¹³ dar și de o componentă „zgomot“ (ε). Modelarea acestei componente va fi făcută în manieră probabilistă, și anume (aici) cu ajutorul unei distribuții gaussiene.

¹³Evident, dependența dintre Y și X va fi considerată de tip liniar.

Fie variabilele aleatoare X și Y . Presupunem că valorile variabilei Y se generează în funcție de valorile variabilei X conform relației:

$$Y = aX + \varepsilon,$$

unde fiecare ε reprezintă valoarea unei variabile aleatoare (care este independentă de variabilele precedente), numită „zgomot“, și care urmează o distribuție gaussiană de medie 0 și deviație standard $\sigma > 0$. Acest fapt se notează îndeobște astfel: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Acesta este un model probabilist (de tip „regresie liniară“, cu un singur atribut, X), în care a este singurul parametru (o pondere; engl., weight).

Probabilitatea condiționată a lui Y în raport cu X urmează distribuția $\mathcal{N}(aX, \sigma^2)$, deci poate fi descrisă ca

$$p(Y|X, a) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y - aX)^2\right).$$

Următoarele întrebări se referă la acest model de regresie liniară.

Estimare în sens MLE:

a. Presupunem că avem un set de date de antrenament constând din n perechi (X_i, Y_i) , cu $i = 1, \dots, n$, și că varianța σ este cunoscută. Care dintre următoarele expresii reprezintă în mod corect problema de optimizare pentru estimarea lui a în sensul verosimilității maxime? Răspundetă cu *da* sau *nu* la fiecare dintre ele. Nu neapărat una singură dintre aceste expresii trebuie să fie însotită de răspunsul *da*.

- i. $\arg \max_a \sum_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$
- ii. $\arg \max_a \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$
- iii. $\arg \max_a \sum_i \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$
- iv. $\arg \max_a \prod_i \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$
- v. $\arg \max_a \sum_i (Y_i - aX_i)^2$
- vi. $\operatorname{argmin}_a \sum_i (Y_i - aX_i)^2$

b. Obțineți estimarea de verosimilitate maximă pentru parametrul a în funcție de exemplele de antrenament X_i și Y_i ($i = 1, \dots, n$). Vă recomandăm / cerem să porniți de la forma cea mai simplă a problemei de optimizare pe care ati identificat-o la punctul precedent.

Estimare în sens MAP:

Să presupunem că $a \sim \mathcal{N}(0, \lambda^2)$, deci

$$p(a|\lambda) = \frac{1}{\sqrt{2\pi}\lambda} \exp\left(-\frac{1}{2\lambda^2}a^2\right) \text{ cu } \lambda > 0.$$

Probabilitatea (de fapt, p.d.f.) a posteriori a parametrului a este

$$\begin{aligned} p(a|Y_1, \dots, Y_n, X_1, \dots, X_n, \lambda) &\stackrel{T.B.}{=} \frac{p(Y_1, \dots, Y_n|X_1, \dots, X_n, a) p(X_1, \dots, X_n, a|\lambda)}{p(Y_1, \dots, Y_n|X_1, \dots, X_n, \lambda)} \\ &\stackrel{F.P.T.}{=} \frac{p(Y_1, \dots, Y_n|X_1, \dots, X_n, a) p(a|\lambda)}{\int_{a'} p(Y_1, \dots, Y_n|X_1, \dots, X_n, a') p(a'|\lambda) da'}. \end{aligned}$$

Putem să ignorăm numitorul acestei fracții atunci când facem estimare în sens MAP, fiindcă el nu depinde de parametrul a .

c. Presupunem că $\sigma = 1$ și că parametrul a priori λ a fost fixat. Găsiți estimarea de probabilitate maximă a posteriori (engl., maximum a posteriori probability, MAP) pentru parametrul a :

$$\arg \max_a [\ln p(Y_1, \dots, Y_n | X_1, \dots, X_n, a) + \ln p(a|\lambda)].$$

Soluția pe care o veți da trebuie să fie în funcție de X_i , Y_i ($i = 1, \dots, n$) și λ .

Răspuns:

a. Funcția de *verosimilitate condițională* a datelor de ieșire $Y = (Y_1, \dots, Y_n)$ în raport cu datele de intrare $X = (X_1, \dots, X_n)$ și cu parametrul a este:

$$\begin{aligned} L(a) &\stackrel{\text{def.}}{=} p(Y_1, \dots, Y_n | X_1, \dots, X_n, a) \\ &\stackrel{i.i.d.}{=} \prod_{i=1}^n p(Y_i | X_i, a) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right). \end{aligned}$$

Prin urmare,

$$a_{MLE} \stackrel{\text{def.}}{=} \arg \max_a L(a) = \arg \max_a \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right).$$

Așadar, soluția *ii* propusă în enunț este corectă.

Mai departe, se poate scrie imediat o formă mai simplă pentru expresia funcției de verosimilitate:

$$\begin{aligned} L(a) &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \prod_{i=1}^n \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right) \\ &= \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\sum_{i=1}^n \frac{1}{2\sigma^2}(Y_i - aX_i)^2\right). \end{aligned}$$

Factorul $\frac{1}{(\sqrt{2\pi}\sigma)^n}$ este constant în raport cu parametrul a și este de asemenea pozitiv, pentru că deviația standard σ este prin definiție un număr pozitiv. În consecință,

$$a_{MLE} \stackrel{\text{def.}}{=} \arg \max_a L(a) = \arg \max_a \prod_{i=1}^n \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right),$$

ceea ce înseamnă că și soluția *iv* propusă în enunț este corectă.

Funcția de log-verosimilitate a datelor are expresia:

$$\begin{aligned} \ell(a) &\stackrel{\text{def.}}{=} \ln L(a) = \ln \left(\frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - aX_i)^2\right) \right) \\ &= -n \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - aX_i)^2. \end{aligned}$$

Datorită faptului că funcția \ln (ca și orice funcție \log_b cu $b > 1$) este strict crescătoare, rezultă că la compunere de funcții conservă / păstrează monotonia. În cazul nostru, această proprietate implică

$$\begin{aligned} a_{MLE} &\stackrel{\text{def.}}{=} \arg \max_a L(a) = \arg \max_a \ell(a) \\ &= \arg \max_a \left(-n \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - aX_i)^2 \right). \end{aligned}$$

Termenul $-n \ln(\sqrt{2\pi}\sigma)$ fiind constant în raport cu a , rezultă că

$$\begin{aligned} a_{MLE} &= \arg \max_a \left(-\frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - aX_i)^2 \right) = \arg \max_a \left(-\sum_{i=1}^n (Y_i - aX_i)^2 \right) \\ &= \arg \min_a \sum_{i=1}^n (Y_i - aX_i)^2. \end{aligned}$$

Penultima egalitate de mai sus are loc fiindcă factorul $\frac{1}{2\sigma^2}$ este pozitiv și constant în raport cu a . Ultima egalitate se explică prin faptul că la schimbarea de semn (adică, prin renunțarea la semnul — din fața simbolului \sum), operatorul $\arg \max$ se transformă în $\arg \min$. Așadar, soluția *vi* din enunț este, de asemenea, corectă. Celelalte soluții propuse în enunț (*i*, *iii* și *v*) sunt incorecte.

b. Vom face calculele pornind de la expresia *vi* de la punctul precedent.

$$a_{MLE} = \arg \min_a \sum_{i=1}^n (Y_i - aX_i)^2 = \arg \min_a \left(a^2 \sum_{i=1}^n X_i^2 - 2a \sum_{i=1}^n X_i Y_i + \sum_{i=1}^n Y_i^2 \right).$$

Putem observa că expresia $a^2 \sum_{i=1}^n X_i^2 - 2a \sum_{i=1}^n X_i Y_i + \sum_{i=1}^n Y_i^2$ reprezintă o funcție de gradul 2 în raport cu parametrul / variabila a , care are — în ipoteza că există măcar un $X_i \neq 0$ — coeficientul dominant pozitiv, deci va avea un (singur) minim. Abscisa acestui punct de minim este chiar a_{MLE} . Prin urmare,¹⁴

$$a_{MLE} = -\frac{-2 \sum_{i=1}^n X_i Y_i}{2 \sum_{i=1}^n X_i^2} = \frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2}.$$

c. Înținând cont și de faptul că

$$\arg \max_a \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp \left(-\frac{1}{2\sigma^2} (Y_i - aX_i)^2 \right) = \arg \max_a \left(-\frac{1}{2} \sum_i (Y_i - aX_i)^2 \right),$$

problema de optimizare în sens MAP devine:

$$\begin{aligned} &\arg \max_a \left(-\frac{1}{2} \sum_{i=1}^n (Y_i - aX_i)^2 + \ln \frac{1}{\sqrt{2\pi}\lambda} - \frac{1}{2\lambda^2} a^2 \right) \\ &= \arg \max_a \left(-\frac{1}{2} \sum_{i=1}^n (Y_i - aX_i)^2 - \frac{1}{2\lambda^2} a^2 \right) \\ &= \arg \min_a \left(\sum_{i=1}^n (Y_i - aX_i)^2 + \frac{a^2}{\lambda^2} \right) \end{aligned} \tag{1}$$

¹⁴Se știe că pentru funcția de gradul al doilea $ax^2 + bx + c$, abscisa punctului de optim este $-\frac{b}{2a}$.

$$= \arg \min_a \left(a^2 \left(\sum_{i=1}^n X_i^2 + \frac{1}{\lambda^2} \right) - 2a \sum_{i=1}^n X_i Y_i + \sum_{i=1}^n Y_i^2 \right).$$

Ca și la punctul b , observăm că funcția de gradul al doilea căreia i se aplică operatorul $\arg \max$ are coeficientul dominant pozitiv, deci va avea un (singur) minim. Prin urmare, obținem pentru a_{MAP} următorul rezultat:

$$a_{MAP} = \frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\lambda^2}}.$$

2. (Regresie liniară [cu anumite restricții impuse]: exemplu de aplicare)

CMU, 2005 fall, T. Mitchell, A. Moore, midterm exam, pr. 3

Presupunem că vrem să „antrenăm“ (engl., fit) un model de regresie liniară pe următoarele date:

x	-1	0	2
y	1	-1	1

- a. Antrenați modelul $Y_i = \beta_1 X_i + \varepsilon_i$ (reprezentând o regresie liniară unidimensională, dar fără termenul constant (β_0)); găsiți valoarea optimă pentru parametrul β_1 .¹⁵
- b. Antrenați modelul $Y_i = \beta_0 + \varepsilon_i$ (reprezentând o regresie liniară „degenerată“, fiind că nu avem nicio variabilă de intrare); găsiți valoarea optimă pentru parametrul β_0 .

Răspuns:

- a. Adaptând definiția regresiei liniare pentru acest caz particular, vom scrie: $Y_i | X_i \sim \mathcal{N}(\beta_1 X_i, \sigma^2)$ pentru $i = 1, 2, 3$. Apoi, pornind de la expresia funcției de verosimilitate a datelor — în scrierea căreia vom folosi notația $Y = (Y_1, Y_2, Y_3)$ și $X = (X_1, X_2, X_3)$ —, vom putea calcula $\hat{\beta}_1$, valoarea optimă a lui β_1 , astfel:

$$\begin{aligned} \hat{\beta}_1 &= \underset{\beta_1}{\operatorname{argmax}} P(Y|X; \beta_1) \stackrel{i.i.d.}{=} \underset{\beta_1}{\operatorname{argmax}} \prod_{i=1}^3 P(Y_i|X_i; \beta_1) \\ &= \underset{\beta_1}{\operatorname{argmax}} \prod_{i=1}^3 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \beta_1 X_i)^2}{2\sigma^2}} = \underset{\beta_1}{\operatorname{argmax}} \prod_{i=1}^3 e^{-\frac{(Y_i - \beta_1 X_i)^2}{2\sigma^2}} \\ &= \underset{\beta_1}{\operatorname{argmax}} \ln \prod_{i=1}^3 e^{-\frac{(Y_i - \beta_1 X_i)^2}{2\sigma^2}} = \underset{\beta_1}{\operatorname{argmax}} \sum_{i=1}^3 -\frac{(Y_i - \beta_1 X_i)^2}{2\sigma^2} \\ &= \underset{\beta_1}{\operatorname{argmin}} \sum_{i=1}^3 (Y_i - \beta_1 X_i)^2 = \underset{\beta_1}{\operatorname{argmin}} \left(\beta_1^2 \sum_{i=1}^3 X_i^2 - 2\beta_1 \sum_{i=1}^3 X_i Y_i + \sum_{i=1}^3 Y_i^2 \right) \\ &= \frac{\sum_{i=1}^3 X_i Y_i}{\sum_{i=1}^3 X_i^2} = \frac{(-1) \cdot 1 + 0 \cdot (-1) + 2 \cdot 1}{(-1)^2 + 0^2 + 2^2} = \frac{1}{5}. \end{aligned}$$

¹⁵Acest punct poate fi văzut ca o aplicație / exemplificare a modelului de regresie liniară de la problema 1.ab.

b. În acest caz, $Y_i|X_i \sim \mathcal{N}(\beta_0, \sigma^2)$ pentru $i = 1, 2, 3$. Similar raționamentului de la punctul precedent, vom avea:

$$\begin{aligned}
\hat{\beta}_0 &= \underset{\beta_0}{\operatorname{argmax}} P(Y|X; \beta_0) \stackrel{i.i.d.}{=} \underset{\beta_0}{\operatorname{argmax}} \prod_{i=1}^3 P(Y_i|X_i; \beta_0) \\
&= \underset{\beta_0}{\operatorname{argmax}} \prod_{i=1}^3 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \beta_0)^2}{2\sigma^2}} = \underset{\beta_0}{\operatorname{argmax}} \prod_{i=1}^3 e^{-\frac{(Y_i - \beta_0)^2}{2\sigma^2}} \\
&= \underset{\beta_0}{\operatorname{argmax}} \ln \prod_{i=1}^3 e^{-\frac{(Y_i - \beta_0)^2}{2\sigma^2}} = \underset{\beta_0}{\operatorname{argmax}} \sum_{i=1}^3 -\frac{(Y_i - \beta_0)^2}{2\sigma^2} \\
&= \underset{\beta_0}{\operatorname{argmin}} \sum_{i=1}^3 (Y_i - \beta_0)^2 = \underset{\beta_0}{\operatorname{argmin}} \left(3\beta_0^2 - 2\beta_0 \sum_{i=1}^3 Y_i + \sum_{i=1}^3 Y_i^2 \right) \\
&= \frac{\sum_{i=1}^3 Y_i}{3} = \frac{1 + (-1) + 1}{3} = \frac{1}{3}.
\end{aligned}$$

3. (Regresia liniară: prezentare generală, rezolvare folosind MLE;
 regresie neliniară [LC: polinomială];
 regularizare de normă L_2 (regresia ridge),
 corespondență cu estimarea MAP)
 ■ *prelucrare de Liviu Ciortuz, după CMU, 2015 spring, Alex Smola, HW1, pr. 2*

Introducere:

Obiectivul acestei probleme este acela de a ne furniza o privire mai generală¹⁶ asupra regresiei liniare, mai întâi din perspectiva estimării de verosimilitate maximă (engl., Maximum Likelihood Estimation, MLE) a parametrilor, iar apoi din perspectiva estimării de probabilitate maximă a posteriori (engl., Maximum-a-Posteriori Estimation probability, MAP). Vom vedea că în cel de-al doilea caz în expresia funcției obiectiv apar și niște termeni de *regularizare*.¹⁷

În fiecare dintre cele două cazuri vom pune în evidență *i.* cum anume problema de *estimare* a parametrilor pentru respectiva problemă de regresie liniară conduce la [și, chiar mai mult, este echivalentă cu] a rezolva o anumită problemă de *optimizare* scrisă sub formă matriceală și *ii.* cum se calculează *soluția analitică* a acelei probleme de optimizare.

Vom prezenta succint și o variantă de regresie neliniară, și anume regresia polinomială.

A. Regresia liniară: prezentare generală, bazată pe estimarea în sens MLE și, echivalent, folosind *criteriul LSE*

Considerăm un *model liniar* care conține o componentă de tip „zgomot“ (engl., noise) care urmează o distribuție gaussiană:

$$Y_i = X_i \cdot w + b + \varepsilon_i \text{ cu } \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \text{ pentru } i = 1, \dots, n, \quad (2)$$

¹⁶La precedentele două probleme (1 și 2) am lucrat cu o singură variabilă de intrare; aici vom lucra cu un număr arbitrar (d) de variabile de intrare (X_i , cu $i = 1, \dots, d$).

¹⁷La problema 1, termenul de regularizare apare sub forma $\frac{a^2}{\lambda^2}$ în relația (1).

unde $Y_i \in \mathbb{R}$ este „răspunsul“ pentru intrarea $X_i \in \mathbb{R}^d$ (acesta din urmă fiind văzut ca vector-linie), $b \in \mathbb{R}$ este un număr real (care în terminologia de limbă engleză se numește *bias*, iar în limba română *termen liber*), $w \in \mathbb{R}^d$ este tot un vector-linie, format din *ponderi* (engl., weights) care „acționează“ asupra instanțelor X_i , iar ε_i sunt „zgomote“ i.i.d.¹⁸ de tip gaussian, cu varianță σ^2 .

Având instanțele $\{X_i | i = 1, \dots, n\}$, scopul nostru este să estimăm valorile parametrilor w și b , care specifică / determină modelul de regresie.

Vom arăta că a rezolva modelul liniar (2) prin metoda MLE revine în mod echivalent la a rezolva următoarea problemă de tip *Least [Sum of] Squared Errors* (rom., *suma celor mai mici pătrate*):

$$\arg \min_{\beta} \underbrace{(Y - X' \beta)^T (Y - X' \beta)}_{\|Y - X' \beta\|^2}, \quad (3)$$

unde notația $\| \cdot \|$ indică norma euclidiană (L_2),¹⁹ $Y = (Y_1, \dots, Y_n)^\top$, $X'_i = (1, X_i)^\top$, $X' = (X'_1, \dots, X'_n)^\top$, iar $\beta = (b, w)^\top$.²⁰

- a. Pornind de la modelul (2), calculați funcția densitate de probabilitate (p.d.f.) condițională a „răspunsului“ Y_i văzut ca variabilă aleatoare, în raport cu intrarea X_i și cu parametrii w și b . (Notație: $\Pr(Y_i | X_i, w, b)$.) Remarcați faptul că putem să-l privim pe X_i pur și simplu ca pe o instanță (punct fixat) din \mathbb{R}^d , nu ca pe o variabilă aleatoare.
- b. Calculați în mod explicit expresia funcției de log-verosimilitate a datelor $\ell(\beta) \stackrel{\text{def.}}{=} \Pr(Y | X, \beta)$, unde $X \stackrel{\text{not.}}{=} (X_1, \dots, X_n)$.
- c. Demonstrați că a găsi estimarea de tip MLE a parametrului β — adică, valoarea lui β pentru care funcția de log-verosimilitate condițională $\ell(\beta)$ își atinge maximul — este echivalent cu a rezolva problema de tip *Least Squared Errors* (3).
- d. Deducreți valoarea lui β care maximizează funcția de log-verosimilitate condițională. Puteți presupune că *matricea de design* X' , care a fost definită mai sus, are rangul maxim posibil (engl., full rank) în spațiul determinat de vectorii-colonă din care este formată această matrice.

Sugestie (1): Puteti, eventual, să folosiți următoarele formule de calcul cu derivate vectoriale:²¹

$$(5a) \quad \frac{\partial}{\partial X} a^\top X = \frac{\partial}{\partial X} X^\top a = a$$

$$(5b) \quad \frac{\partial}{\partial X} X^\top A X = (A + A^\top) X$$

B. Regresie neliniară [LC: polinomială]

¹⁸Adică independente și identic distribuite.

¹⁹Vă readucem aminte că dind vectorul $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, norma sa de ordin k , cu $k \in \mathbb{N}$, se definește astfel: $\|x\|_k = (x_1^k + \dots + x_d^k)^{1/k} = \sqrt[k]{x_1^k + \dots + x_d^k}$. În general, atunci când nu se precizează indicele din scrierea lui $\| \cdot \|_k$, se consideră că este vorba despre valoarea $k = 2$, care corespunde normei euclidiene.

²⁰Matricea X' se numește *matrice de design*. Linia i a acestei matrice este instanța X_i extinsă cu elementul 1 pe prima poziție, adică $(X'_i)^\top$.

²¹Cf. *Matrix Identities*, Sam Roweis, 1999, <http://www.cs.nyu.edu/~roweis/notes/matrixid.pdf>.

e. Considerăm vectorul $\phi(X_i) = (1, X_i, X_i^2, \dots, X_i^k)^\top$, cu $X_i \in \mathbb{R}$. În noul model de regresie pe care-l construim acum (și pe care-l putem numi / considera un *model de ordin k*), vom exprima vectorul Y astfel:

$$Y_i = \phi(X_i) \cdot \beta' + \varepsilon_i \text{ cu } \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \text{ pentru } i = 1, \dots, n, \quad (4)$$

unde $\beta' \in \mathbb{R}^{k+1}$.

Arătați că atunci când facem estimare în sens MLE și presupunem că matricea $\phi(X) \stackrel{\text{not.}}{=} (\phi(X_1), \phi(X_2), \dots, \phi(X_n))^\top$ are rangul maxim în spațiul determinat de coloanele acestei matrice, valoarea optimă a parametrului β' din relația (4) este

$$(\phi(X)^\top \phi(X))^{-1} \phi(X)^\top Y.$$

Sugestie (2): Nu este nevoie să elaborați toți pașii demonstrației, cum ați procedat în secțiunea A. Concentrați-vă pe schimbările care trebuie operate asupra expresiei funcției de log-verosimilitate, și deduceți noua formă a problemei de optimizare.

C. Regresia liniară cu „zgomot“ gaussian și regularizare de normă L_2 (*regresia ridge*); estimarea ponderilor în sensul MAP

*Comentariu:*²² Multe probleme de regresie pot avea sute sau mii de atrbute / variabile de intrare (numite și *variabile de predicție*). Dacă aceste atrbute sunt corelate, tehnici standard de regresie (precum cele de la punctele A și B) vor conduce în faza de antrenament la modele foarte complexe, precum și la erori de generalizare mari în faza de testare / generalizare. O altă chestiune care poate să apară este faptul că unele probleme pot avea mai multe atrbute decât numărul de instanțe din setul de date de antrenament. Atunci când se întâmplă așa ceva, metodele regresie standard vor *eșua*. (Vom vedea mai jos de ce.) O metodă care ne permite să rezolvăm ambele chestiuni de mai sus este *regresia ridge*. Ideea pe care se bazează această metodă de regresie este una simplă: este bine să modificăm *funcția de cost / pierdere* (engl., loss function), adăugând un *termen de penalizare* aplicat ponderilor (engl., weights), pentru a le determina să ia valori mici. Acest termen de penalizare este cunoscut ca *regularizator*; el controlează *complexitatea modelului* prin faptul că permite ponderi mari (în valoare absolută) doar pentru *cele mai importante atrbute* din model. Regresia *ridge* penalizează pătratul lungimii / mărimii vectorului de ponderi (β). Aceasta este numită uneori *penalizare L_2* , fiindcă este pătratul normei L_2 (adică, norma euclidiană) aplicate vectorului de ponderi (β).

f. În situația în care matricea $X'^\top X'$ din secțiunea A (respectiv matricea $\phi(X)^\top \phi(X)$ din secțiunea B) nu este inversabilă,²³ se poate arăta că există $\lambda > 0$, astfel încât matricea $X'^\top X' + \lambda I$ — unde I este matricea identitate de dimensiune $d+1$ — să fie inversabilă.²⁴

²²Cf. CMU, 2008 fall, Eric Xing, HW1, pr. 4.2.

²³Aceasta este situația când, spre exemplu, numărul trăsăturilor, $d+1$ din secțiunea A — respectiv $k+1$ din secțiunea B —, este mai mare decât numărul de instanțe, n (adică, $d+1 > n$). În acest caz, vom avea $\text{rang}(X'^\top X') = \text{rang}(X')$ (vedeți documentul *Matrix identities*, de Sam Roweis, formula (2f)). Așadar, $\text{rang}(X'^\top X')$ este mai mic sau egal cu $\min(n, d+1) = n$, care este mai mic decât $d+1$. Prin urmare, matricea $X'^\top X'$, care are dimensiunea $(d+1) \times (d+1)$, nu are rangul maxim și în consecință nu poate fi inversată.

²⁴Pentru a înțelege în mod intuitiv această chestiune, observați că dacă două coloane din matricea $X^\top X$ ar fi liniar dependente, atunci λI ar adăuga exact aceeași valoare (λ) la doar două dintre componente (diferite) ale acestor coloane. Așadar, aceste coloane ar deveni liniar independente în matricea $X^\top X + \lambda I$.

Arătați că

$$\hat{\beta}'' \stackrel{not.}{=} (X'^\top X' + \lambda I)^{-1} X'^\top Y$$

este soluția problemei de optimizare

$$\arg \min_{\beta''} (\|Y - X'\beta''\|^2 + \lambda \|\beta''\|^2). \quad (5)$$

g. Considerăm cazul în care β'' urmează o distribuție a priori $\beta'' \sim \mathcal{N}(0, \eta^2 I)$.

Scrieți distribuția a posteriori a parametrului β'' în raport cu Y_i și instanța X_i , precum și distribuția a posteriori a lui β'' în raport cu Y și întreg setul de date de antrenament, X . Veți considera că β'' și „zgomottele” $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, pentru $i = 1, \dots, n$ sunt independente.

Sugestie (3): Folosiți regula lui Bayes:

$$\Pr(\beta''|Y_i, X_i) = \frac{\Pr(Y_i|X_i, \beta'') \Pr(\beta''|X_i)}{\Pr(Y_i|X_i)} = \frac{\Pr(Y_i|X_i, \beta'') \Pr(\beta'')}{\Pr(Y_i|X_i)}.$$

Apoi urmați aceeași pași ca în secțiunea A.

Sugestie (4): Folosiți faptul că prin corelarea a două sau mai multe variabile gaussiene unidimensionale independente se obține o distribuție (multidimensională) tot de tip gaussian. (Vedeți problema 31 de la capitolul de *Fundamente*.)

h. $\hat{\beta}''$, estimarea în sens MAP a parametrului β'' , este definită ca fiind acea valoare a lui β'' pentru care se obține *modul* [adică, valoarea maximă a] probabilității a posteriori $\Pr(\beta''|Y, X)$. Arătați că identificarea acestei estimări MAP conduce la problema (5) în cazul în care λ poate fi exprimat în funcție de σ și η , adică $\lambda = h(\sigma, \eta)$. Găsiți expresia funcției $h(\sigma, \eta)$.

i. Indicați o problemă care poate eventual să apară dacă am elimina termenul de regularizare din relația (5). Precizați cum anume poate termenul de regularizare să rezolve această potențială problemă.

Răspuns:

a. Observați că $Y_i|X_i; w, b \sim \mathcal{N}(X_i \cdot w + b, \sigma^2)$, aşadar putem scrie funcția densitate de probabilitate (p.d.f.) a lui $Y_i|X_i, w, b$ în forma următoare:

$$p(Y_i = y_i|X_i; w, b) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - X_i \cdot w - b)^2}{2\sigma^2}\right).$$

b. Notând $y = (y_1, \dots, y_n)^\top$, întrucât instanțele X_i sunt date, iar variabilele reprezentând „zgomottele” ε_i sunt i.i.d., funcția de verosimilitate a lui Y în raport cu X și β se scrie astfel:

$$\begin{aligned} L(\beta) &\stackrel{def.}{=} p(Y = y|X, \beta) = \prod_{i=1}^n p(y_i|X_i, w, b) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - X_i \cdot w - b)^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^n (y_i - X_i \cdot w - b)^2}{2\sigma^2}\right). \end{aligned}$$

Apoi, aplicând logaritmul natural (\ln), rezultă că funcția de log-verosimilitate a lui $Y|\beta$ este următoarea:

$$\ell(\beta) \stackrel{\text{def.}}{=} \ln L(\beta) = -n \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - X_i \cdot w - b)^2. \quad (6)$$

c. Pentru a calcula maximul funcției de log-verosimilitate $\ell(\beta)$, ne vom concentra asupra celui de-al doilea termen din expresia (6), fiindcă primul termen este constant în raport cu β . Observăm că pentru a maximiza cel de-al doilea termen din expresia (6), trebuie să minimizăm $\sum_{i=1}^n (y_i - X_i \cdot w - b)^2$. Scriind această sumă în notație matriceală / vectorială, obținem:

$$\max_{\beta} \ell(\beta) = \min_{\beta} \sum_{i=1}^n (y_i - X_i \cdot w - b)^2 = \min_{\beta} (y - X' \beta)^\top (y - X' \beta), \quad (7)$$

unde, din nou, $Y = (Y_1, \dots, Y_n)^\top$, $X'_i = (1, X_i)^\top$, $X' = (X'_1, \dots, X'_n)^\top$, iar $\beta = (b, w)^\top$.

d. Definind funcția obiectiv

$$J(\beta) \stackrel{\text{def.}}{=} (y - X' \beta)^\top (y - X' \beta), \quad (8)$$

și aplicând regulile (5a) și (5b) menționate în *Sugestia* (1) din enunț, rezultă:²⁵

$$\nabla_{\beta} J(\beta) = 2X'^\top (X' \beta - y).$$

Se poate demonstra relativ ușor că matricea hessiană corespunzătoare funcției $J(\beta)$ este pozitiv definită (vedeți problema 7.a), deci această funcție este convexă.

Valoarea $\hat{\beta}$ pentru care se obține maximul funcției de log-verosimilitate poate fi găsită rezolvând următoarea ecuație:

$$\begin{aligned} \nabla_{\beta} J(\hat{\beta}) = 0 &\Leftrightarrow X'^\top (X' \hat{\beta} - y) = 0 \Leftrightarrow X'^\top X' \hat{\beta} = X'^\top y \Leftrightarrow \\ &\hat{\beta} = (X'^\top X')^{-1} X'^\top y. \end{aligned} \quad (9)$$

Remarcați faptul că matricea $(X'^\top X')^{-1}$ există fiindcă s-a presupus că matricea X' are rangul maxim în spațiul determinat de coloanele sale.²⁶ (Acet fapt, corroborat cu observația de mai sus referitoare la convexitatea funcției $J(\beta)$, implică faptul că punctul de maxim al lui J este unic.)

²⁵Într-adevăr,

$$\begin{aligned} J(\beta) &= (y - X' \beta)^\top (y - X' \beta) = (y^\top - (X' \beta)^\top)(y - X' \beta) \\ &= (y^\top - \beta^\top X'^\top)(y - X' \beta) = y^\top y - y^\top X' \beta - \beta^\top X'^\top y + \beta^\top X'^\top X' \beta \\ \stackrel{(5a)}{\Rightarrow} \stackrel{(5b)}{\Rightarrow} \nabla_{\beta} J(\beta) &= 0 - (y^\top X')^\top - X'^\top y + 2X'^\top X' \beta \\ &= -2X'^\top y + 2X'^\top X' \beta = 2X'^\top (X' \beta - y). \end{aligned}$$

²⁶LC: Afirmația „matricea X' are rangul maxim în spațiul determinat de coloanele sale“ trebuie interpretată astfel: numărul coloanelor din matricea X — matrice ale cărei linii sunt chiar instanțele X'_i — care sunt liniar independente este maxim, deci $d+1$ (dar și $n \geq d+1$, unde n este numărul de instanțe de antrenament, deci și numărul de linii din matricea X'). Folosind următoarea proprietate din documentul *Matrix Identities* de Sam Rowens

$$(2f) \quad \text{rang}(A^\top A) = \text{rang}(AA^\top) = \text{rang}(A) \quad \text{pentru orice matrice } A,$$

rezultă că rangul matricei $X'^\top X'$ este $d+1$. Prin urmare, matricea $X'^\top X'$ este inversabilă.

e. Întrucât s-a modificat relația dintre Y_i și X_i , se vor modifica și distribuția condițională (adică, p.d.f.-ul variabilei $Y_i|X_i, \beta'$) și funcția de verosimilitate. Noile lor expresii pot fi obținute pur și simplu înlocuind X_i cu $\phi(X_i)$, după cum urmează:

$$\begin{aligned} p(Y_i = y_i | X_i; w, b) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(y_i - \phi(X_i) \cdot \beta')^2}{2\sigma^2}\right) \\ \ell(\beta') &= -n \ln(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \phi(X_i) \cdot \beta')^2. \end{aligned}$$

La fel cum am procedat mai înainte, putem aplica și acum metoda estimării de verosimilitate maximă:

$$\max_{\beta'} \ell(\beta') = \min_{\beta'} \sum_{i=1}^n (y_i - \phi(X_i) \cdot \beta')^2 = \min_{\beta'} (y - \phi(X)\beta')^\top (y - \phi(X)\beta'),$$

iar estimatorul de verosimilitate maximă este

$$\hat{\beta}' = (\phi(X)^\top \phi(X))^{-1} \phi(X)^\top y. \quad (10)$$

f. Demonstrația urmează aproape același curs ca la punctul d. Mai întâi, vom defini funcția obiectiv J'' astfel:

$$\begin{aligned} J''(\hat{\beta}'') &= (y - X'\beta'')^\top (y - X'\beta'') + \lambda \underbrace{\|\beta''\|^2}_{\beta^\top \beta} \\ \stackrel{(5a),(5b)}{\implies} \nabla_{\beta''} J''(\beta') &= 2X'^\top (X'\beta'' - y) + 2\lambda\beta''. \end{aligned}$$

Similar cu observația făcută la punctul d, se poate arăta că și în cazul funcției J'' matricea hessiană este pozitiv definită, deci J'' este funcție convexă. Valoarea $\hat{\beta}''$ pentru care se obține maximul funcției de log-verosimilitate poate fi găsită rezolvând următoarea ecuație:

$$\begin{aligned} \nabla_{\beta''} J''(\hat{\beta}'') &= 0 \\ \Leftrightarrow X'^\top (X'\hat{\beta}'' - y) + \lambda\hat{\beta}'' &= 0 \\ \Leftrightarrow (X'^\top X' + \lambda I)\hat{\beta}'' &= X'^\top y \\ \Leftrightarrow \hat{\beta}'' &= (X'^\top X' + \lambda I)^{-1} X'^\top y. \end{aligned} \quad (11)$$

g. Stim că $Y_i|X_i, \beta'' \sim \mathcal{N}(X_i \cdot \beta'', \sigma^2)$, iar $\beta'' \sim \mathcal{N}(0, \eta^2 I)$. Folosind regula lui Bayes, rezultă că

$$f(\beta''|Y_i, X_i) \propto p(Y_i|X_i, \beta'') g(\beta'') \propto \exp\left(-\frac{1}{2\sigma^2}(Y - X_i \cdot \beta'')^2\right) \cdot \exp\left(-\frac{1}{2\eta^2}\beta''^\top \beta''\right),$$

unde semnul \propto înseamnă ‘proporțional cu’, $g(\cdot)$ este funcția de densitate de probabilitate (p.d.f.) pentru β'' , iar $f(\cdot)$ este funcția de densitate de probabilitate pentru $\beta''|Y_i, X_i$. Factorul de normalizare pentru p.d.f.-ul f este definit ca $Z_i = \int_{-\infty}^{\infty} f(Y_i|X_i, \beta'') g(\beta'') d\beta''$, aşadar putem scrie p.d.f.-ul $f(\beta''|Y_i, X_i)$ astfel:

$$f(\beta''|Y_i, X_i) = \frac{1}{Z_i} \exp\left(-\frac{1}{2\sigma^2}(Y - X_i \cdot \beta'')^2 - \frac{1}{2\eta^2}\beta''^\top \beta''\right).$$

În mod similar, ținând cont de *Sugestia* (4) din enunț, rezultă că p.d.f.-ul lui $\beta''|Y, X$ este

$$\begin{aligned} f(\beta''|Y) &= \frac{1}{Z} \exp \left(-\frac{1}{2\sigma^2} (Y - X'\beta'')^\top (Y - X'\beta'') - \frac{1}{2\eta^2} \beta''^\top \beta'' \right) \\ &= \frac{1}{Z} \exp \left(-\frac{1}{2\sigma^2} \|Y - X'\beta''\|^2 - \frac{1}{2\eta^2} \|\beta''\|^2 \right), \end{aligned}$$

unde Z este factorul de normalizare, definit ca $Z = \int_{-\infty}^{\infty} f(Y|X, \beta'') g(\beta'') d\beta''$.

h. De la punctul g este clar că

$$\max_{\beta''} f(\beta''|Y_i) = \min_{\beta''} \left(\frac{1}{2\sigma^2} \|Y - X\beta''\|^2 + \frac{1}{2\eta^2} \|\beta''\|^2 \right). \quad (12)$$

Dacă η^2 devine $\frac{\sigma^2}{\lambda}$, atunci problema de minimizare devine echivalentă cu problema (5),

$$\arg \min_{\beta''} (\|Y - X\beta''\|^2 + \lambda \|\beta''\|^2).$$

Altfel spus, prin stabilirea corespondenței termenilor, suntem conduși la a lua $\lambda = h(\sigma, \eta) = \frac{\sigma^2}{\eta^2}$, ceea ce forțează relația (12) să devină (5).

i. Termenul de regularizare penalizează componentele din β care au valori mari; în consecință, β va avea valori mici (în normă). Prin urmare, termenul de regularizare „încurajează“ modelul să evite *overfitting*-ul, împiedicându-l astfel să se aducă la instanțele care sunt *outlier*-e (altfel, acestea ar influența în mod drastic valoarea lui β).

4.

(O proprietate a regresiei liniare, varianta LSE:
la rezolvarea cu ajutorul *formulelor analitice*,
scalarea atributelor nu schimbă predicțiile obținute
pentru instanțele de test)

■ *MIT, 2001 fall, Tommi Jaakkola, HW1, pr. 1.6*

Considerăm un set de exemple de antrenament $D = \{(x_i, y_i) | i = 1, \dots, n\}$ cu $x_i \in \mathbb{R}^d$ și $y_i \in \mathbb{R}$, precum și o instanță de test $x_{test} \in \mathbb{R}^d$ și, în plus, factorii de scalare $\alpha_1, \dots, \alpha_d \in \mathbb{R}^*$.²⁷ Demonstrați următoarea egalitate:

$$predicted y_{test} = predicted \tilde{y}_{test},$$

unde

predicted y_{test} este predicția pentru outputul sau *valoarea de răspuns*²⁸ pentru instanța x_{test} după ce am „antrenat“ / produs un model de regresie liniară de tip LSE pe setul D folosind *soluția analitică*;²⁹

predicted \tilde{y}_{test} este predicția obținută în mod similar, antrenând un model de regresie LSE pe setul de date $D' = \{(\tilde{x}_i, y_i) | i = 1, \dots, n\}$, cu $\tilde{x}_{i,j} = \alpha_j x_{i,j}$ și facând apoi predicție pentru \tilde{x}_{test} , unde $\tilde{x}_{test,j} = \alpha_j x_{test,j}$, pentru $j \in \{1, \dots, d\}$.

²⁷Pentru attributele pe care nu le scalăm vom considera în mod implicit $\alpha_j = 1$.

²⁸Vedeți problema 5, unde pentru această notiune am folosit notația \hat{y} .

²⁹Adică, „ecuația normală“ reprezentată de relația (9) de la problema 3.d.

Observație: O proprietate similară este valabilă și în cazul metodei lui Newton; vedeti problema 152 de la capitolul de *Fundamente*. Rezultatul respectiv este însă mai general; el nu este limitat la folosirea metodei lui Newton pentru rezolvarea problemelor de regresie [liniară].

Răspuns:

Putem să formalizăm relația dintre noua *matrice de design* (\tilde{X}) și cea veche (X) folosind o *matrice de transformare*, $A \in \mathbb{R}^{d \times d}$, pe a cărei diagonală sunt plasate coeficienții de scalare α_j (și care are celelalte elemente 0):

$$\tilde{X} = XA.$$

Utilizând această relație în „ecuația normală“ care ne dă valoarea optimă a vectorului de ponderi β , vom obține:³⁰

$$\begin{aligned}\tilde{\beta} &= (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top y \\ &= ((XA)^\top XA)^{-1} (XA)^\top y \\ &= (A^\top X^\top XA)^{-1} A^\top X^\top y \\ &= A^{-1} (X^\top X)^{-1} \underbrace{(A^\top)^{-1} A^\top}_{I} X^\top y \\ &= A^{-1} (X^\top X)^{-1} X^\top y \\ &= A^{-1} \beta.\end{aligned}$$

Așadar, outputul care va fi prezis este următorul:

$$\begin{aligned}\text{predicted } \tilde{y}_{test} &= \tilde{X}_{test} \tilde{\beta} \\ &= (X_{test} A)(A^{-1} \beta) = X_{test} (AA^{-1}) \beta \\ &= X_{test} \beta \\ &= \text{predicted } y_{test}.\end{aligned}$$

Observație:

La același rezultat se poate ajunge făcând *o altă demonstrație*, pe care doar o schițăm: Întrucât instanțele \tilde{x}_i sunt obținute prin scalarea atributelor instanțelor x_i , multimea de transformări liniare asupra [atributelor] unei instanțe oarecare $x \in \mathbb{R}^d$ este exact aceeași cu multimea de transformări liniare asupra [atributelor] instanței scalate \tilde{x} . *Predictor*-ul, în ambele cazuri, este [vectorul de ponderi β care determină] funcția liniară din această multime care minimizează eroarea la antrenare.

[LC:] Se poate demonstra ușor că valorile optime ale funcțiilor obiectiv în cele două cazuri sunt identice. Aceasta implică faptul că pătratele erorilor (de forma $\|y_i - x_i \beta\|^2$ și respectiv $\|y_i - \tilde{x}_i \tilde{\beta}'\|^2$) sunt identice pentru orice $i = 1, \dots, n$. Ca să arătăm că avem și $x_i \beta = \tilde{x}_i \tilde{\beta}'$ pentru $i = 1, \dots, n$ (ceea ce este adevarat dacă $\tilde{\beta}' = A^{-1} \beta$), observăm mai întâi că cele două funcții obiectiv sunt convexe, fiindcă matricele lor hessiene sunt pozitiv definite; vedeti problema 7.a), iar apoi că cele două matrice hessiene sunt chiar inversabile, deoarece conform enunțului putem scrie ecuațiile „normale“. Așadar, punctele de minim pentru cele două funcții obiectiv sunt unice. Rezultă cu necesitate că relația între cei doi predictori

³⁰În continuare, X și y se referă la datele de antrenament.

este cea de mai sus, $\tilde{\beta}' = A^{-1} \beta$.

În concluzie, cei doi predictori vor produce rezultate identice și atunci când sunt aplicăți pe instanțele de test X_{test} și respectiv \tilde{X}_{test} .

5.

(Regresia liniară, varianta LSE:
bias-ul și matricea de covarianță a estimatorului;
Regresia ridge: bias-ul)

■ CMU, (?) spring, (10-701 course), HW1, pr. 3ab+4a

După cum știți, modelul regresiei liniare are forma

$$y = x\beta + \varepsilon,$$

unde $x = (x_1, \dots, x_d)^\top$, $\beta = (\beta_1, \dots, \beta_d)^\top$, $y \in \mathbb{R}$, iar ε este un „zgomot“, pe care aici îl vom considera de tip gaussian, având $E(\varepsilon) = 0$ și $Var(\varepsilon) = \sigma^2$. Dat fiind setul de date de antrenament $(x_1, y_1), \dots, (x_n, y_n)$, în care fiecare instanță x_i este un vector de forma $(x_{i1}, \dots, x_{id})^\top \in \mathbb{R}^d$ iar $y_i \in \mathbb{R}$, vrem să estimăm parametrii β . Notăm cu X matricea de dimensiune $n \times d$ în care linia i este vectorul x_i^\top ; în mod similar, vectorul-coloană Y , cu n -componente, este format din ieșirile / „răspunsurile“ corespunzătoare instanțelor din setul de antrenament.

Remember: Am arătat la problema 3.A că a minimiza suma pătratelor erorilor — vă reamintim că în notația matricială această sumă se scrie $\|Y - X\beta\|^2$ — revine la a calcula următoarea *estimare* a parametrului β :

$$\hat{\beta} = (X^\top X)^{-1} X^\top Y. \quad (13)$$

Rezultă imediat că vectorul de *predicții* corespunzător lui Y este calculat astfel:

$$\hat{y} = X\hat{\beta} = X(X^\top X)^{-1} X^\top Y.$$

Remarcați faptul că vectorul Y este o variabilă aleatoare (*n-ară*), fiindcă pentru orice instanță x_i , valoarea outputului y_i este afectată de „zgomotul“ aleatoriu ε_i . Într-adevăr, vectorul $\hat{\beta}$ se calculează în funcție de vectorul Y deci și el ($\hat{\beta}$) este o cantitate aleatoare.

- a. Demonstrați că $\hat{\beta}$, estimatorul regresiei liniare în varianta LSE, este *nedeplasat* (engl., unbiased), adică $E(\hat{\beta}) = \beta$.
- b. Demostrați că matricea de covarianță a lui $\hat{\beta}$ este egală cu $\sigma^2(X^\top X)^{-1}$.
- c. *Remember:* La problema 3.C am arătat că putem obține estimatorul *ridge*, o variantă a estimatorului de tip LSE pentru regresia liniară, impunând un factor de penalizare asupra mărimii vectorului format din coeficienții de regresie ($\hat{\beta}$):

$$\|X\beta - Y\|^2 + \lambda\beta^\top\beta,$$

unde parametrul λ controlează contribuția / efectul termenului de regularizare. Minimizarea acestei funcții de cost ne conduce la următorul estimator al vectorului de coeficienți ai regresiei liniare regularizate:

$$\hat{\beta} = (X^\top X + \lambda I)^{-1} X^\top Y, \quad (14)$$

care în statistică este cunoscut sub numele de *estimator ridge*.³¹ Predicția de tip *ridge* a lui Y este [dată de formula]

$$\hat{y} = X\hat{\beta} = X(X^\top X + \lambda I)^{-1}X^\top Y.$$

Demonstrați că estimatorul *ridge* este deplasat (engl., biased).

Răspuns:

a. Pornind de la relația (13), putem scrie:

$$\begin{aligned}\hat{\beta} &= (X^\top X)^{-1}X^\top Y \\ &= (X^\top X)^{-1}X^\top(X\beta + \varepsilon) \\ &= \beta + (X^\top X)^{-1}X^\top\varepsilon\end{aligned}\tag{15}$$

Apoi, aplicând proprietatea de liniaritate a mediei, vom obține:

$$E[\hat{\beta}] = E[\beta + (X^\top X)^{-1}X^\top\varepsilon] = \beta + (X^\top X)^{-1}X^\top\underbrace{E[\varepsilon]}_0 = \beta.$$

Așadar, estimatorul $\hat{\beta}$ este nedeplasat.

b. Vom demonstra mai întâi că pentru orice vector de variabile aleatoare $V = (V_1, \dots, V_d)$, matricea sa de covarianță, care este notată cu $Cov[V]$ și are ca element generic $Cov(V_i, V_j) \stackrel{\text{def}}{=} E[(V_i - E[V_i])(V_j - E[V_j])]$, satisfac egalitatea $Cov[V] = E[VV^\top] - E[V](E[V]^\top)$:

$$\begin{aligned}Cov[V] &\stackrel{\text{def.}}{=} [Cov(V_i, V_j)]_{i,j \in \{1, \dots, d\}} \\ &\stackrel{\text{def.}}{=} [E[(V_i - E[\beta_i])(V_j - E[V_j])]]_{i,j \in \{1, \dots, d\}} \\ &= [E[V_i V_j] - E[\beta_i]E[V_j]]_{i,j \in \{1, \dots, d\}} \\ &= E[VV^\top] - E[V](E[V]^\top).\end{aligned}$$

Observație: Această egalitate este o generalizare a egalității $Cov(X, Y) = E[XY] - E[X]E[Y]$, unde X și Y sunt variabile aleatoare oarecare; vedeti problema 9.c de la capitolul de *Fundamente*.

Înlocuind V cu $\hat{\beta}$ în relația pe care am demonstrat-o mai sus, obținem

$$Cov[\hat{\beta}] = E[\hat{\beta}\hat{\beta}^\top] - E[\hat{\beta}](E[\hat{\beta}]^\top).\tag{16}$$

În continuare, făcând uz de relația (15), în membrul drept al egalității (16) îl vom înlocui pe $\hat{\beta}$ cu $\beta + (X^\top X)^{-1}X^\top\varepsilon$ și apoi vom folosi rezultatul $E[\hat{\beta}] = \beta$ demonstrat la punctul a. Așadar,

³¹Remarcați faptul că atât estimatorul LSE cât și estimatorul *ridge* sunt *estimatori liniari*. Prin definiție, un estimator $\tilde{\beta}$ este liniar dacă el poate fi scris ca o combinație liniară de vectorul de ieșirile Y , adică $\tilde{\beta} = KY$, unde K este o matrice oarecare de dimensiune $d \times n$.

$$\begin{aligned}
Cov[\hat{\beta}] &= E[(\beta + (X^\top X)^{-1}X^\top \varepsilon)(\beta + (X^\top X)^{-1}X^\top \varepsilon)^\top] - \beta\beta^\top \\
&= E[(\beta + (X^\top X)^{-1}X^\top \varepsilon)(\beta^\top + \varepsilon^\top X((X^\top X)^{-1})^\top)] - \beta\beta^\top \\
&= E[(\beta + (X^\top X)^{-1}X^\top \varepsilon)(\beta^\top + \underbrace{\varepsilon^\top X((X^\top X)^\top)^{-1}}_{X^\top X})] - \beta\beta^\top \\
&\stackrel{lin. med.}{=} \underbrace{E[\beta\beta^\top]}_{\beta\beta^\top} + E[\beta\varepsilon^\top X(X^\top X)^{-1} + (X^\top X)^{-1}X^\top \varepsilon\beta^\top] + \\
&\quad E[(X^\top X)^{-1}X^\top \varepsilon \varepsilon^\top X(X^\top X)^{-1}] - \beta\beta^\top \\
&\stackrel{lin. med.}{=} \underbrace{\beta E[\varepsilon]^\top X(X^\top X)^{-1}}_0 + (X^\top X)^{-1}X^\top \underbrace{E[\varepsilon]\beta^\top}_0 + \\
&\quad E[(X^\top X)^{-1}X^\top \varepsilon \varepsilon^\top X(X^\top X)^{-1}] \\
&= E[(X^\top X)^{-1}X^\top \varepsilon \varepsilon^\top X(X^\top X)^{-1}] \\
&\stackrel{lin. med.}{=} (X^\top X)^{-1}X^\top \underbrace{E[\varepsilon \varepsilon^\top]}_{\sigma^2 I_{n \times n}} X(X^\top X)^{-1} \\
&= \sigma^2 (X^\top X)^{-1}(X^\top X)(X^\top X)^{-1} \\
&= \sigma^2 (X^\top X)^{-1}.
\end{aligned}$$

c. Pentru a arăta că estimatorul $\hat{\beta}$ al regresiei *ridge* este deplasat, va trebui să demonstrăm că $E[\hat{\beta}] \neq \beta$ sau, alternativ, că $E[\hat{\beta}] - \beta$ este diferit de vectorul nul. Într-adevăr, folosind relația (14), putem scrie:

$$\begin{aligned}
E[\hat{\beta}] - \beta &= E[(X^\top X + \lambda I)^{-1}X^\top y] - \beta \\
&= E[(X^\top X + \lambda I)^{-1}X^\top (X\beta + \varepsilon)] - \beta \\
&\stackrel{lin. med.}{=} (X^\top X + \lambda I)^{-1}X^\top X\beta + (X^\top X + \lambda I)^{-1}X^\top \underbrace{E[\varepsilon]}_0 - \beta \\
&= (X^\top X + \lambda I)^{-1}X^\top X\beta - \beta \\
&= (X^\top X + \lambda I)^{-1}(X^\top X + \lambda I - \lambda I)\beta - \beta \\
&\stackrel{distrib.}{=} (X^\top X + \lambda I)^{-1}(X^\top X + \lambda I)\beta - (X^\top X + \lambda I)^{-1}\lambda I\beta - \beta \\
&= \beta - \lambda(X^\top X + \lambda I)^{-1}\beta - \beta \\
&= -\lambda(X^\top X + \lambda I)^{-1}\beta \\
&\neq 0.
\end{aligned}$$

6.

(Regresia liniară cu regularizare L_2 (Regresia *ridge*): rezolvare cu metoda gradientului, varianta “batch” și varianta stochastică)

■ CMU, 2008 fall, Eric Xing, HW1, pr. 4.2.2

Fie $D = \{(x_i, y_i) | i = 1, \dots, n\}$ un set de n exemple de antrenament pentru o problemă de regresie liniară. Fie $y_i \in \mathbb{R}$ eticheta (sau, variabila de răspuns) pentru exemplul i , iar $y \in \mathbb{R}^n$ vectorul-coloană alcătuit din toate variabilele de răspuns. Fiecare exemplu de

antrenament are d atribute (sau, variabile de intrare / predicție). Fie $x_i \in \mathbb{R}^{d \times 1}$ vectorul coloană format din toate variabilele de predicție pentru exemplul i , iar $X \in \mathbb{R}^{n \times d}$ matricea de design, formată din toate variabilele de predicție, linia i a matricei X fiind x_i^\top . Fie $\beta \in \mathbb{R}^{d \times 1}$ un vector-coloană care conține *ponderile* / *parameterii* pe care vrem să le / îi învățăm“.

Comentariu: La problema 3.C am arătat (folosind calculul matriceal și derivatele vectoriale) că *soluția analitică* a problemei

$$\arg \min_{\beta} [\|y - X\beta\|^2 + \lambda\|\beta\|^2]$$

este

$$\hat{\beta}_{ridge} = (X^\top X + \lambda I)^{-1} X^\top y.$$

Deși această *soluție analitică* este ușor de implementat, ea poate deveni total nepractică atunci când numărul atributelor este foarte mare, fiindcă trebuie să calculăm atât produsul $X^\top X$ cât și inversa acestei matrice. O tehnică / soluție pe care o putem folosi pentru a depăși acest neajuns este metoda *gradientului descendente* (engl., gradient descent), care este o metodă de optimizare, iterativă.

a. La acest punct vom lucra cu varianta “batch” (sau “steepest descent”) a metodei gradientului descendente. La fiecare iterație se îmbunătățesc ponderile β , aplicând următoarea *regulă de actualizare*:

$$\beta^{(t)} = \beta^{(t-1)} - \eta \nabla_{\beta} \ell(\beta^{(t-1)}),$$

unde constanta η este aşa-numita *rată de învățare* (engl., learning rate), iar $\nabla_{\beta} \ell(\beta)$ este transpusul vectorului gradient pentru funcția de cost / pierdere $\ell(\beta)$.

Deducreți regula de actualizare pentru metoda gradientului, varianta “batch”, știind că expresia funcției de log-verosimilitate $\ell(\beta)$ este:

$$\ell(\beta) = \frac{1}{2} \sum_{i=1}^n (y_i - x_i^\top \beta)^2 + \frac{\lambda}{2} \beta^\top \beta.$$

Sugestie: Puteti folosi următoarele formule de calcul pentru derivare vectorială (din documentul *Matrix Identities*, de Sam Roweis, 1999):³²

$$(5a) \frac{\partial}{\partial X} a^\top X = \frac{\partial}{\partial X} X^\top a = a \quad (5b) \frac{\partial}{\partial X} X^\top A X = (A + A^\top) X.$$

b. Varianta secvențială / stochastică a metodei gradientului descendente actualizează parametrii β la procesarea fiecărui exemplu. Acest fapt este extrem de folositor în cazul seturilor de date de antrenament foarte mari, precum și atunci când exemplele sunt furnizate în manieră secvențială, printr-un “stream” de date. În acest caz, *regula de actualizare* a ponderilor β este următoarea:

$$\beta^{(t)} = \beta^{(t-1)} - \eta \nabla_{\beta} \ell_i(\beta^{(t-1)}; x_i, y_i),$$

unde η este rata de învățare (o constantă), iar $\ell_i(\beta; x_i, y_i)$ este transpusul vectorului gradient pentru funcția de cost / pierdere calculată pentru exemplul particular i .

³²<http://www.cs.nyu.edu/~roweis/notes/matrixid.pdf>.

Calculați regula de actualizare pentru varianta stochastică a metodei gradientului descentant, știind că $\ell_i(\beta; x_i, y_i)$ este

$$\ell_i(\beta; x_i, y_i) = (y_i - x_i^\top \beta)^2 + \frac{\lambda}{2} \beta^\top \beta.$$

Răspuns:

a. Mai întâi vom aplica unele transformări elementare asupra funcției de log-verosimilitate ℓ (în aşa fel încât să putem aplica ulterior regulile de derivare vectorială):

$$\begin{aligned}\ell(\beta) &= \frac{1}{2} \sum_{i=1}^n (y_i - x_i^\top \beta)^2 + \frac{\lambda}{2} \beta^\top \beta \\ &= \frac{1}{2} \sum_{i=1}^n (y_i^2 - 2y_i x_i^\top \beta + \underbrace{(x_i^\top \beta)^2}_{(x_i^\top \beta)^\top (x_i^\top \beta)}) + \frac{\lambda}{2} \beta^\top \beta \\ &= \frac{1}{2} \sum_{i=1}^n (y_i^2 - 2y_i x_i^\top \beta + \beta^\top x_i x_i^\top \beta) + \frac{\lambda}{2} \beta^\top \beta.\end{aligned}$$

Acum vom calcula gradientul funcției ℓ , aplicând regulile de derivare vectorială:

$$\nabla_\beta \frac{\partial \ell(\beta)}{\partial \beta} \stackrel{(5a)}{=} \frac{1}{2} \sum_{i=1}^n (-2y_i x_i + 2x_i x_i^\top \beta) + \lambda \beta = \sum_{i=1}^n [-y_i x_i + x_i (x_i^\top \beta)] + \lambda \beta. \quad (17)$$

Așadar, pentru regresia *ridge*, metoda gradientului folosește în varianta “batch” următoarea regulă de actualizare:

$$\beta^{(t)} = \beta^{(t-1)} - \eta \left[\sum_{i=1}^n [-y_i x_i + x_i (x_i^\top \beta^{(t-1)})] + \lambda \beta^{(t-1)} \right].$$

b. Procedând similar cu rezolvarea de la punctul precedent, vom ajunge la următorul rezultat:

$$\beta^{(t)} = \beta^{(t-1)} - \eta [-y_i x_i + x_i (x_i^\top \beta^{(t-1)}) + \lambda \beta^{(t-1)}].$$

7.

(Regresia liniară, varianta LSE:
rezolvare cu metoda lui Newton)

Stanford, 2007 fall, Andrew Ng, HW1, pr. 1

Introducere: La problema 3 am rezolvat problemele de regresie prezentate acolo folosind *metoda analitică* (și obținând aşa-numitele *ecuații normale*, (9), (10), (11)). În schimb, aici (și la problema 6) vom folosi *metode de optimizare iterativă*: metoda lui Newton și respectiv metoda gradientului.

În această problemă vom demonstra că la folosirea *metodei lui Newton*³³ pentru rezolvarea problemei de regresie liniară în forma *sumei celor mai mici pătrate* (engl., least squared error, LSE), avem nevoie de o singură iterație pentru a ajunge la convergență (și, deci, pentru a obține soluția $\hat{\beta}$).

³³Pentru o introducere la metoda lui Newton, vedeți *Comentariul* din enunțul problemei 72 de la capitolul de *Fundamente*.

a. Calculați matricea hessiană³⁴ pentru funcția de cost / pierdere

$$J(\beta) = \frac{1}{2} \sum_{i=1}^n (\beta^\top x^{(i)} - y^{(i)})^2.$$

Observație: Funcția de cost $J(\beta)$ coincide (până la factorul 1/2) cu funcția obiectiv din problema de optimizare (3) de la problema 3.A.³⁵

b. Arătați că indiferent de valoarea $\beta^{(0)}$ pe care o atribuim inițial vectorului β , la prima iterată a metodei lui Newton se obține vectorul $\beta^{(1)} = (X^\top X)^{-1} X^\top y = \hat{\beta}$, adică exact soluția problemei de regresie liniară de tipul (sau, în varianta) *suma celor mai mici pătrate*.

Răspuns:

a. Calculăm mai întâi derivatele parțiale ale funcției $J(\beta)$ în raport cu β_j , pentru $j = 1, \dots, d$, unde d este numărul de atribute din instanțele $x^{(i)}$:

$$\frac{\partial J(\beta)}{\partial \beta_j} = \sum_{i=1}^n (\beta^\top x^{(i)} - y^{(i)}) x_j^{(i)}.$$

Rezultă că derivatele parțiale de ordin secund ale lui $J(\beta)$ sunt de forma următoare:

$$\frac{\partial^2 J(\beta)}{\partial \beta_j \partial \beta_k} = \sum_{i=1}^n \frac{\partial}{\partial \beta_k} (\beta^\top x^{(i)} - y^{(i)}) x_j^{(i)} = \sum_{i=1}^n x_j^{(i)} x_k^{(i)} = (X^\top X)_{jk}.$$

Așadar, matricea hessiană pentru funcția $J(\beta)$ este $X^\top X$.³⁶

Observație: Se poate demonstra imediat că matricea $X^\top X$ este pozitiv definită: pentru orice $z \in \mathbb{R}^d$ avem $z^\top (X^\top X) z = (Xz)^\top (Xz) = (Xz)^2 \geq 0$. În consecință, J este funcție convexă, deci admite [măcar un punct de] minim. În anumite condiții — de exemplu, atunci când $X^\top X$ este matrice inversabilă, aşa cum s-a presupus la problema 3.A —, punctul respectiv de minim este unic.

b. Pornind de la o valoarea arbitrară $\beta^{(0)}$ pentru vectorul de ponderi β , metoda lui Newton calculează $\beta^{(1)}$ conform formulei

$$\beta^{(1)} = \beta^{(0)} - H^{-1} \nabla_\beta J(\beta^{(0)}).$$

Se poate arăta relativ ușor că $\nabla_\beta J(\beta) = X^\top X \beta - X^\top y$.³⁷ Prin urmare,

$$\begin{aligned} \beta^{(1)} &= \beta^{(0)} - (X^\top X)^{-1} (X^\top X \beta^{(0)} - X^\top y) \\ &= \beta^{(0)} - \beta^{(0)} + (X^\top X)^{-1} X^\top y \\ &= (X^\top X)^{-1} X^\top y. \end{aligned}$$

³⁴Denumirea de matrice hessiană este derivată de la numele matematicianului german Ludwig Otto Hesse (1811 - 1874), care l-a avut ca profesor pe un alt matematician german vestit, Carl Gustav Jacob Jacobi (1804 - 1851). Prin definiție, matricea hessiană a unei funcții — pentru care există derivatele parțiale de ordinul întâi și al doilea — este o matrice pătratică formată din derivatele parțiale de ordin secund ale funcției respective.

³⁵În ce privește notățile din aceste două probleme, avem corespondențele următoare: $x^{(i)} \rightarrow X'_i$, $y^{(i)} \rightarrow Y_i$ și $X \rightarrow X'$.

³⁶Se poate ajunge la acest rezultat și aplicând pur și simplu regulile de derivare vectorială [care au fost învățate la cursul de algebră liniară].

³⁷Cunoscând $\frac{\partial J(\beta)}{\partial \beta_j}$ de la punctul a , se poate scrie vectorul gradient $\nabla_\beta J(\beta) \stackrel{\text{def.}}{=} \frac{\partial J(\beta)}{\partial \beta_j}$

În consecință, indiferent de valoarea $\beta^{(0)}$ pe care o atribuim inițial vectorului β , metoda lui Newton obține soluția $\hat{\beta}$ (adică, optimul funcției obiectiv $J(\beta)$) după o singură iterație.

8.

(Regresia liniară, [alte] două variante:
regresia liniară local-ponderată (cu „zgomot“ gaussian) și
modelarea „zgomotului“ cu distribuția Laplace)

■ CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW2, pr. 3.1-2
CMU, 2007 fall, Carlos Guestrin, HW1, pr. 2.2

Remember: Pentru regresia liniară, se dă un set de date de antrenament de forma $D = (X, y) = \{(x_i, y_i) | i = 1, 2, \dots, n\}$, unde $x_i \in \mathbb{R}^{d \times 1}$, adică $x_i = (x_{i,1}, \dots, x_{i,d})^\top$, $y_i \in \mathbb{R}$, $X \in \mathbb{R}^{n \times d}$, linia i a matricei X fiind x_i^\top și, în sfârșit, $y = (y_1, \dots, y_n)^\top$. Folosind un model [parametrizat] de forma $y_i = x_i \beta + \varepsilon_i$, unde ε_i sunt „zgomote“ generate de o anumită distribuție probabilistă, metoda regresiei liniare caută să afle acea valoare a vectorului de parametri β care asigură cea mai bună „potrivire“ / adecvare (engl., *fit*) a modelului de mai sus pe datele de antrenament D . O modalitate (sau, un *criteriu*) de a măsura / evalua această „potrivire“ / adecvare este să găsim acel β care minimizează o funcție de cost / pierdere (engl., *loss function*) dată, $J(\beta)$. La problema 3.A am introdus modelul regresiei lineare în varianta *sumei pătratelor erorilor* (engl., least squared errors, LSE), iar la secțiunile B și C din aceeași problemă am prezentat două extensii / variante ale acestui model: regresia polinomială și regresia *ridge*.

În această problemă vom explora alte două extensii / variante ale modelului de regresie LSE: *regresia local-ponderată*, în cazul căreia asociem fiecărei instanțe x_i un coeficient w_i care reprezintă ponderea / importanța respectivei instanțe, precum și un model de regresie în care „zgomotele“ ε_i sunt modelate cu distribuția Laplace.

A. Regresie liniară *local-ponderată*, cu „zgomot“ gaussian³⁸

Presupunem că „zgomotele“ $\varepsilon_1, \dots, \varepsilon_n$ sunt [tot de tip gaussian, tot] independente, dar acum varianțele nu mai sunt identice, deci $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, pentru $i = 1, \dots, n$.

a. Stabiliți formula de calcul pentru estimarea de verosimilitate maximă (MLE) a vectorului de parametri β în această variantă de regresie.

Sugestie: Următoarele formule (din documentul *Matrix Identities*, de Sam Roweis, 1999)³⁹ vă pot fi de folos:

$$(5a) \frac{\partial}{\partial X} a^\top X = \frac{\partial}{\partial X} X^\top a = a \quad (5b) \frac{\partial}{\partial X} X^\top AX = (A + A^\top)X.$$

$\left(\frac{\partial J(\beta)}{\partial \beta_1}, \dots, \frac{\partial J(\beta)}{\partial \beta_d} \right)$ astfel:

$$\nabla_\beta J(\beta) = \sum_{i=1}^n (\beta^\top x^{(i)} - y^{(i)}) x^{(i)} = \sum_{i=1}^n (\beta^\top x^{(i)}) x^{(i)} - \sum_{i=1}^n y^{(i)} x^{(i)} = X^\top X \beta - X^\top y.$$

Pentru a justifica ultima egalitate, este suficient să scrieți componentele generice ale celor două matrice, $X^\top X \beta$ și $X^\top y$.

³⁸Acest model de regresie corespunde sumei ponderate a celor mai mici pătrate (engl., locally-weighted least squares).

³⁹<http://www.cs.nyu.edu/~roweis/notes/matrixid.pdf>.

b. Arătați că estimarea în sens MLE pe care ați calculat-o la punctul a coincide cu valoarea parametrului β pentru care se obține minimul unei funcții de cost / pierdere de forma următoare⁴⁰

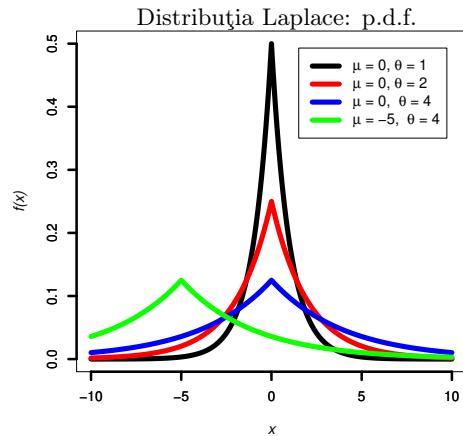
$$J_W(\beta) = \sum_i w_i (y_i - x_i^\top \beta)^2.$$

Exprimați ponderea w_i în funcție de σ_i^2 , varianța „zgomotului“ ε_i din exemplul de antrenament i .

c. Explicați de ce acest nou estimator — suma ponderată a celor mai mici pătrate — este [uneori] preferată în raport cu versiunea neponderată.

Sugestie: Analizați cazul / cazurile în care σ_i^2 are o valoare mare, comparativ cu cazul / cazurile în care valoarea sa este mică.

B. Regresie liniară [neponderată], cu „zgomot“ de tip *Laplace*



Presupunem că „zgomotele“ $\varepsilon_1, \dots, \varepsilon_n$ sunt distribuite în mod identic și independent, conform unei distribuții de tip Laplace de parametri $\mu = 0$ și $\theta > 0$. Vă readucem aminte că funcția de densitate de probabilitate (p.d.f.) pentru această distribuție este

$$\text{Laplace}(x; \mu, \theta) \stackrel{\text{def.}}{=} \frac{1}{2\theta} \exp\left(-\frac{|x - \mu|}{\theta}\right).$$

d. Găsiți funcția de cost / pierdere $J_{\text{Laplace}}(\beta)$ a cărei minimizare este echivalentă cu găsirea estimării în sens MLE pentru parametrul β în cazul acestui tip de modelare a „zgomotului“.

e. Care credeți că este avantajul acestui model, comparativ cu varianta standard (cea în care modelarea „zgomotului“ se face folosind o distribuție gaussiană)?

Sugestie: Gândiți-vă la excepții / anomalii (engl., outliers).

Răspuns:

a. Știm că $y_i = x_i^\top \beta + \varepsilon_i$, iar $p(y_i|x_i; \beta) = \mathcal{N}(x_i^\top \beta, \sigma_i^2)$. Așadar, formula care ne dă estimarea în sens MLE pentru parametrul β este:

$$\begin{aligned} \beta_{MLE} &= \underset{\beta}{\operatorname{argmax}} \ln \prod_i p(y_i|x_i; \beta) = \underset{\beta}{\operatorname{argmax}} \sum_i \ln p(y_i|x_i; \beta) \\ &= \underset{\beta}{\operatorname{argmax}} \sum_i \ln \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - x_i^\top \beta)^2}{2\sigma_i^2}\right) \right) \\ &= \underset{\beta}{\operatorname{argmax}} \sum_i \left(\ln \frac{1}{\sqrt{2\pi\sigma_i^2}} - \frac{(y_i - x_i^\top \beta)^2}{2\sigma_i^2} \right) = \underset{\beta}{\operatorname{argmax}} \sum_i -\frac{(y_i - x_i^\top \beta)^2}{2\sigma_i^2} \end{aligned}$$

⁴⁰Această funcție se numește *suma ponderată a celor mai mici pătrate*; engl., weighted least squares loss function.

$$\begin{aligned}
&= \arg \min_{\beta} \sum_i \frac{(y_i - x_i^\top \beta)^2}{2\sigma_i^2} \\
&= \arg \min_{\beta} \sum_i \frac{1}{\sigma_i^2} (y_i - x_i^\top \beta)^2.
\end{aligned} \tag{18}$$

Vom scrie acum expresia (18) în notație matriceală. Considerând W matricea diagonală de dimensiune $n \times n$, în care $w_{ii} = \frac{1}{\sigma_i^2}$ pentru $i = 1, \dots, n$, vom obține:

$$\beta_{MLE} = \arg \min_{\beta} (y - X\beta)^\top W(y - X\beta). \tag{19}$$

Ca și în cazul regresiei liniare neponderate în varianta LSE, se poate demonstrea relativ ușor că matricea hessiană corespunzătoare funcției de cost $J_W(\beta) = (y - X\beta)^\top W(y - X\beta)$ este *pozitiv definită*, deci această funcție este *convexă* și prin urmare admite punct de minim.

Valoarea efectivă a estimatorului β_{MLE} este soluția ecuației care se obține egalând gradientul expresiei $(y - X\beta)^\top W(y - X\beta)$ cu vectorul 0:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \beta} \left((y - X\beta)^\top W(y - X\beta) \right) \\
&\stackrel{distrib.}{=} \frac{\partial}{\partial \beta} \left(y^\top W y - \underbrace{y^\top W X \beta}_{\in \mathbb{R}} - \underbrace{\beta^\top X^\top W y}_{(y^\top W^\top X \beta)^\top} + \beta^\top X^\top W X \beta \right).
\end{aligned} \tag{20}$$

Pentru orice număr real z are loc egalitatea $z^\top = z$, astădat $((\beta^\top X^\top) W y)^\top = y^\top W^\top X \beta = y^\top W X \beta$, întrucât $W^\top = W$. (Vă reamintim că matricea W este diagonală.) Tinând cont de aceasta, ecuația (20) va fi rescrisă astfel:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \beta} \left(y^\top W y - 2\beta^\top X^\top W y + \beta^\top X^\top W X \beta \right) \stackrel{(5a)(5b)}{\Leftrightarrow} 0 = -2X^\top W y + 2X^\top W X \beta \\
&\Leftrightarrow X^\top W y = X^\top W X \beta.
\end{aligned}$$

În cazul în care matricea $X^\top W X$ este inversabilă, vom avea:

$$\beta_{MLE} = (X^\top W X)^{-1} X^\top W y. \tag{21}$$

Aceasta este ecuația „normală“ corespunzătoare regresiei LSE local-ponderate.

b. La relația (18) am demonstrat că estimatorul MLE al parametrului β este valoarea pentru care se atinge minimul pentru suma ponderată a celor mai mici pătrate, $\sum_i \frac{1}{\sigma_i^2} (y_i - x_i^\top \beta)^2$. Coroborând această expresie cu relația $J_W(\beta) \stackrel{def.}{=} \sum_i w_i (y_i - x_i^\top \beta)^2$ din enunț, obținem $w_i = \frac{1}{\sigma_i^2}$.

c. Atunci când varianța σ_i^2 este mare, „zgomotul“ ε_i poate fi oricât de mare, deci punctul (x_i, y_i) poate fi un outlier. Nu este de dorit ca în astfel de cazuri estimatorul β_{MLE} să fie deplasat (engl., biased) în aşa fel încât el să „explice“ aceste outlier-e (mai ales atunci când se folosește ca funcție de cost / pierdere suma pătratelor erorilor). Modelul regresiei liniare local-ponderate formulat în problema de față rezolvă această chestiune ponderând „importanța“ (sau, „contribuția“) fiecărui exemplu de antrenament la funcția

obiectiv (J_W) cu ajutorul unui factor care este tocmai inversul varianței corespunzătoare exemplului respectiv. În consecință, instanțele care au o varianță mare nu vor afecta mult funcția de cost / pierdere; ele pot fi ignorate, sau [cel puțin] li se poate acorda o [mai] mică importanță atunci când se caută valoarea optimă a lui β .

d. Știm că $y_i = x_i^\top \beta + \varepsilon_i$, iar $p(y_i|x_i; \beta) = \text{Laplace}(x_i^\top \beta; 0, \theta)$. Așadar, formula care ne dă estimarea în sens MLE pentru parametrul β în acest caz este:

$$\begin{aligned}\beta_{MLE} &= \underset{\beta}{\operatorname{argmax}} \ln \prod_i p(y_i|x_i; \beta) = \underset{\beta}{\operatorname{argmax}} \sum_i \ln p(y_i|x_i; \beta) \\ &= \underset{\beta}{\operatorname{argmax}} \sum_i \ln \left(\frac{1}{2\theta} \exp \left(-\frac{|y_i - x_i^\top \beta|}{\theta} \right) \right) \\ &= \underset{\beta}{\operatorname{argmax}} \sum_i \left(\ln \frac{1}{2\theta} - \frac{|y_i - x_i^\top \beta|}{\theta} \right) = \underset{\beta}{\operatorname{argmax}} \sum_i -\frac{|y_i - x_i^\top \beta|}{\theta} \\ &\stackrel{\theta \geq 0}{=} \arg \min_{\beta} \sum_i |y_i - x_i^\top \beta|.\end{aligned}$$

În concluzie,

$$J_{\text{Laplace}}(\beta) = \sum_i |y_i - x_i^\top \beta|. \quad (22)$$

e. Dacă o instanță de antrenament este “outlier”, eroarea produsă atunci când se face predicția corespunzătoare acestei instanțe — folosind valoarea corectă pentru parametrul β — este mult mai mare în cazul în care se lucrează cu „zgomot“ de tip gaussian (pentru că atunci se folosește pătratul diferenței dintre valoarea “target” și valoarea prezisă de model) decât în cazul zgomotului de tip laplacian (fiindcă în acest caz se lucrează cu modulul diferenței respective). Prin urmare, outlier-ele afectează estimarea lui β mult mai mult în cazul „zgomotului“ gaussian decât în cazul celui laplacian.

Din punctul de vedere al modelării, ținând cont că $y_i = x_i^\top \beta + \varepsilon_i$, atunci când y_i este outlier, modelul poate „explica“ acest fapt dând lui ε_i o valoare mare (în modul), pentru a se putea „accepta“ diferența dintre valoarea “target” și valoarea prezisă pentru outlier-ul respectiv. Acest fapt este posibil în cazul modelului Laplace, fiindcă funcția densitate de probabilitate (p.d.f.) a distribuției Laplace are brațele (engl., tails) mai ridicate decât p.d.f.-ul distribuției gaussiene.

Observație: Făcând legătura cu secțiunea A a problemei de față, putem spune că pentru a obține un efect similar [cu cazul „zgomotului“ Laplace], acolo am presupus că fiecarei instanțe îi asociem o varianță proprie, σ_i^2 . Remarcăți totuși că aceste varianțe trebuie estimate (folosind, de exemplu, un algoritm de tip EM — vedeți capitolul *Algoritmul EM*) fiindcă ele influențează problema de optimizare, în vreme ce în modelul Laplace nu trebuie să facem astfel de estimări. Pe de altă parte, este mai dificil să facem optimizare cu o funcție de cost / pierdere care folosește norma L_1 (cazul distribuției Laplace) decât atunci când se folosește norma L_2 (cazul distribuției gaussiene), fiindcă funcția modul nu este derivabilă pe tot domeniul ei de definiție.

9.

(Regresia liniară cu regularizare L_2 (Regresia *ridge*):
kernel-izare)*CMU, 2014 spring, B. Poczos, A. Singh, HW2, pr. 4.B*

În această problemă vom arăta cum anume poate fi obținută varianta kernel-izată a regresiei *ridge*.⁴¹ După cum se procedează în general la regresia liniară, vom lucra cu un set de instanțe / date $\{x_i\}_{i=1}^n$ din \mathbb{R}^d și cu valorile de „răspuns“ asociate lor, $\{y_i\}_{i=1}^n$ din \mathbb{R} .

Pentru a obține rezultate mai bune pentru problema noastră de regresie, putem folosi o funcție de „mapare“ a atributelor (engl., attribute mapping function) ϕ , care asociază fiecărui vector d -dimensional x câte un nou vector, $\bar{x} = \phi(x)$ din \mathbb{R}^m , numit vector de *trsături* (engl., feature vector), unde m este mai mare decât d .⁴² Considerăm matricea de design $\Phi \in \mathbb{R}^{n \times m}$, în care linia i este vectorul de trăsături \bar{x}_i^\top corespunzător instanței i . Notăm cu y vectorul-colonă format din valorile de răspuns; în acest vector, componenta de pe poziția i este valoarea de răspuns y_i , care a fost asociată instanței i .

Așa cum am arătat la problema 3.C, pentru regresia *ridge* funcția obiectiv este

$$J(\beta) = \|\mathbf{y} - \Phi\beta\|^2 + \lambda\|\beta\|^2, \quad (23)$$

unde $\lambda > 0$ este parametrul de *regularizare*. Vom nota cu $\hat{\beta}$ soluția / estimatorul acestei variante de regresie *ridge*. În cele ce urmează vom arăta cum anume se poate calcula această soluție.

a. Mai întâi, arătați că $\hat{\beta}$, care este un vector din \mathbb{R}^m , este în *spațiul* generat de liniile din matricea Φ . Aceasta înseamnă că $\hat{\beta}$ poate fi scris ca o combinație liniară de vectorii-linie care alcătuiesc matricea Φ . Altfel spus, există vectorul $\hat{\alpha} \in \mathbb{R}^{n \times 1}$ astfel încât $\hat{\beta}$ se poate scrie sub forma

$$\hat{\beta} = \Phi^\top \hat{\alpha}. \quad (24)$$

Arătați apoi că, în ipoteza că matricele $\Phi\Phi^\top$ și $\Phi\Phi^\top + \lambda I$ sunt inversabile, urmează că vectorul de coeficienți $\hat{\alpha}$ se poate exprima cu ajutorul matricei de design Φ astfel:⁴³

$$\hat{\alpha} = (\Phi\Phi^\top + \lambda I)^{-1}y. \quad (25)$$

Sugestie: Vectorul de ponderi β din expresia (23) — ca și orice alt vector din \mathbb{R}^m — poate fi descompus în două componente, și anume $\beta = \beta_{\parallel} + \beta_{\perp}$, unde componenta β_{\parallel} aparține spațiului generat de liniile din Φ , iar componenta β_{\perp} este ortogonală pe acest spațiu.⁴⁴ Această ultimă proprietate înseamnă că produsul scalar dintre orice vector din spațiul generat de liniile din Φ și β_{\perp} este 0.

b. În practică, este foarte dificil să construim funcții de „mapare“ ϕ . Dar chiar și atunci când putem să construim astfel de funcții, calcularea produselor $\phi(x)^\top \phi(x')$ — care sunt necesare pentru aflarea vectorului $\hat{\alpha}$, conform relației (25) — poate fi ineficientă dacă se

⁴¹ Pentru o introducere în subiectul funcțiilor-nucleu, vedeți secțiunea corespunzătoare din capitolul de *Fundamente*. Similar, pentru regresia *ridge*, vedeți problema 3.C.

⁴² În general, m este mult mai mare decât d ; vom nota acest fapt prin $m \gg d$.

⁴³ Pentru regresia liniară nekernel-izată, am abținut deja o astfel de relație la problema 3.f.

⁴⁴ Pentru explicații mai detaliate, puteți vedea documentul *Supplemental Lecture Notes* (pentru cursul de *Machine Learning*), de John Duchi, de la Universitatea Stanford, mai precis secțiunea *A more general representer theorem*, pag. 11-12.

folosește maniera clasică de calcul.⁴⁵ În schimb, putem calcula în mod *implicit* aceste produse, folosind o funcție-nucleu $k(x, x')$ care are proprietatea $k(x, x') = \phi(x)^\top \phi(x')$. În principiu, a *kernel-iza* înseamnă să folosim „apeluri“ la o funcție-nucleu pentru a înlocui produsele $\phi(x)^\top \phi(x')$.

La acest punct, vă cerem să kernel-izați regresia *ridge*, presupunând că vi se dă o funcție-nucleu $k(x, x')$, unde x și x' sunt vectorii originali, de dimensiune d , care corespund instanțelor de antrenament. Va trebui să lucrezi atât asupra antrenării cât și asupra testării. Mai exact, va trebui ca pentru faza de antrenare să înlocuiți cu „apeluri“ la funcția-nucleu k toate produsele scalare care apar în calculul necesar obținerii vectorului $\hat{\alpha}$ (vedeți relația (25)). De asemenea, pentru faza de testare, dată fiind o instanță de test x , va trebui să înlocuiți toate produsele scalare care apar în calculul lui $\phi(x)^\top \hat{\beta} = \phi(x)^\top (\Phi^\top \hat{\alpha}) = (\phi(x)^\top \Phi^\top) \hat{\alpha}$ cu „apeluri“ la funcția-nucleu k .⁴⁶

Răspuns:

a. Înținând cont de *Sugestia* din enunț, vom putea scrie astfel funcția obiectiv a regresiei *ridge* kernel-izate:

$$\begin{aligned} J(\beta) &= \|y - \Phi\hat{\beta}\|^2 + \lambda\|\hat{\beta}\|^2 = \\ J(\beta_{\parallel} + \beta_{\perp}) &= \|y - \underbrace{\Phi\beta_{\perp}}_0 - \Phi\beta_{\parallel}\|^2 + \lambda\|\beta_{\perp}\|^2 + \lambda\|\beta_{\parallel}\|^2 \\ &= \|y - \Phi\beta_{\parallel}\|^2 + \lambda\|\beta_{\perp}\|^2 + \lambda\|\beta_{\parallel}\|^2. \end{aligned}$$

Minimizarea acestei funcții obiectiv va trebui să fie făcută în raport cu cele două componente, β_{\perp} și β_{\parallel} , care sunt ortogonale una în raport cu cealaltă. Din expresia lui J rezultă că, pentru minimizare, componenta β_{\perp} trebuie să fie vectorul $0 \in \mathbb{R}^d$. În consecință, $\hat{\beta} = \beta_{\parallel}$. (Și $J(\beta) = J(\hat{\beta}_{\parallel})$.) Rezultă, deci, că vectorul β aparține spațiului generat de liniile din matricea Φ , ceea ce înseamnă că există într-adevăr vectorul $\hat{\alpha} \in \mathbb{R}^{n \times 1}$ astfel încât $\hat{\beta}$ să se poată scrie sub forma $\hat{\beta} = \Phi^\top \hat{\alpha}$.

Acum, pentru a calcula efectiv vectorul de coeficienți $\hat{\alpha}$, vom folosi relația $\hat{\beta} = \Phi^\top \hat{\alpha}$ și-l vom înlocui pe $\hat{\beta}$ cu $\Phi^\top \hat{\alpha}$ în expresia lui $J(\hat{\beta})$:

$$J(\hat{\beta}) = J(\hat{\beta}_{\parallel}) = \|y - \Phi\hat{\beta}_{\parallel}\|^2 + \lambda\|\hat{\beta}_{\parallel}\|^2 = \|y - \Phi\Phi^\top \hat{\alpha}\|^2 + \lambda\hat{\alpha}^\top \Phi\Phi^\top \hat{\alpha}. \quad (26)$$

Întrucât $\hat{\beta}$ minimizează funcția $J(\beta)$, putem calcula $\hat{\alpha}$ egalând cu vectorul $0 \in \mathbb{R}^n$ derivata în raport cu $\hat{\alpha}$ a expresiei rezultat din relația (26). Așadar, vom avea:⁴⁷

$$-2\Phi\Phi^\top(y - \Phi\Phi^\top \hat{\alpha}) + 2\lambda\Phi\Phi^\top \hat{\alpha} = 0 \Leftrightarrow \Phi\Phi^\top(\Phi\Phi^\top + \lambda I)\hat{\alpha} = \Phi\Phi^\top y,$$

ceea ce implică, în ipoteza că matricele $\Phi\Phi^\top$ și $\Phi\Phi^\top + \lambda I$ sunt inversabile, faptul că

⁴⁵Observați că putem scrie în mod echivalent produsul $\phi(x)^\top \phi(x')$ cu ajutorul produsului scalar al vectorilor: $\phi(x) \cdot \phi(x')$.

⁴⁶Veți observa că, din cauza acestei ultime expresii, la faza de antrenare este suficient să calculăm vectorul $\hat{\alpha}$, nu și vectorul $\hat{\beta} = \Phi^\top \hat{\alpha}$.

⁴⁷Pentru derivarea efectivă, putem folosi mai întâi descompunerea

$$\begin{aligned} \|y - \Phi\Phi^\top \hat{\alpha}\|^2 &\stackrel{\text{def.}}{=} (y - \Phi\Phi^\top \hat{\alpha})^\top (y - \Phi\Phi^\top \hat{\alpha}) = (y^\top - \hat{\alpha}^\top \Phi\Phi^\top)(y - \Phi\Phi^\top \hat{\alpha}) \\ &= y^\top y - \hat{\alpha}^\top \Phi\Phi^\top y - y^\top \Phi\Phi^\top \hat{\alpha} + \hat{\alpha}^\top (\Phi\Phi^\top)(\Phi\Phi^\top) \hat{\alpha} \\ &= y^\top y - 2\hat{\alpha}^\top \Phi\Phi^\top y + \hat{\alpha}^\top (\Phi\Phi^\top)(\Phi\Phi^\top) \hat{\alpha} \end{aligned}$$

și apoi putem aplica regulile de derivare vectorială (5a) și (5b) care au fost folosite și la problema 6.

$$\hat{\alpha} = (\Phi\Phi^\top + \lambda I)^{-1}y.$$

b. La faza de antrenare, trebuie să calculăm vectorul $\hat{\alpha}$. Dacă notăm cu K matricea ale cărei elemente sunt $K_{ij} = k(x_i, x_j)$ rezultă că aceasta este chiar matricea $\Phi\Phi^\top$. Așadar, conform relației (25), rezultă că $\hat{\alpha} = (K + \lambda I)^{-1}y$. (Matricea $K = \Phi\Phi^\top$ se numește *matrice-nucleu* sau *matrice Gram*.)

La faza de testare va trebui să calculăm

$$\phi(x)^\top \hat{\beta} = \phi(x)^\top (\Phi^\top \hat{\alpha}) = (\phi(x)^\top (\Phi^\top)) \hat{\alpha} = \sum_{i=1}^n \hat{\alpha}_i k(x_i, x).$$

10. (Regresia *ridge* kernel-izată, cu nucleu RBF;
un rezultat interesant:
atunci când parametrul de regularizare λ tinde la 0,
eroarea la antrenare devine 0)
*prelucrare de Liviu Ciortuz, după
MIT, 2006 fall, Tommi Jaakkola, HW2, pr. 2.a-c*

La problema 9 am prezentat versiunea kernel-izată a regresiei *ridge*. Vă reamintim că regresie *ridge* înseamnă regresia liniară cu „zgomot” gaussian și regularizare (L_2). Regula de predicție pentru acest tip de regresie kernel-izată este

$$y(x) = \sum_{i=1}^n \hat{\alpha}_i K(x_i, x), \quad (27)$$

unde $\hat{\alpha}_i$ este valoarea optimală a coeficientului α_i , pentru $i = 1, \dots, n$. Valorile $\hat{\alpha}_i$ au fost obținute astfel:

$$\hat{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1}y. \quad (28)$$

În această formulă, $\hat{\alpha} \stackrel{\text{not.}}{=} (\hat{\alpha}_1, \dots, \hat{\alpha}_n)^\top$, $y \stackrel{\text{not.}}{=} (y_1, \dots, y_n)^\top$, iar \mathbf{K} este așa-numita *matrice-nucleu* (sau, *matricea Gram*) corespunzătoare instanțelor x_1, \dots, x_n .⁴⁸

În acest exercițiu vrem să studiem evoluția erorii la antrenare pentru acest model de regresie liniară kernel-izată în condițiile următoare:

1. folosim ca nucleu așa-numita funcție cu baza radială (engl., radial basis function, RBF), care este definită astfel:

$$K(x, x') = \exp\left(-\frac{\gamma}{2}\|x - x'\|^2\right), \quad \gamma > 0 \text{ pentru orice } x, x' \in \mathbb{R}^d;$$

2. valorile parametrului de regularizare λ tind la 0.⁴⁹

- a. Arătați că, în ipoteza $x_i \neq x_j$ pentru orice $i \neq j$, matricea Gram corespunzătoare funcției-nucleu RBF este inversabilă (indiferent de valoarea parametrului γ).⁵⁰

⁴⁸Observație: Spre deosebire de problema 9, aici am notat matricea Gram (care acolo era desemnată cu K) cu simbolul \mathbf{K} , iar funcția-nucleu (care acolo era $k(\cdot, \cdot)$) cu $K(\cdot, \cdot)$.

⁴⁹Facem aceasta, deși a seta $\lambda = 0$ — ceea ce înseamnă a lucra fără regularizare — nu este deloc convenabil, din cauza overfitting-ului.

⁵⁰Justificare: Trecerea la limită pentru $\lambda \rightarrow 0$ în relația (27) necesită calcularea valorii pe care o are vectorul de coeficienți $\hat{\alpha}$ la limită. Din cauza relației (28), întrucât $\lim_{\lambda \rightarrow 0} (\mathbf{K} + \lambda \mathbf{I}) = \mathbf{K}$, trebuie ca [în prealabil] să ne asigurăm că matricea Gram \mathbf{K} este inversabilă.

Sugestie: Puteți folosi *teorema lui Michelli* (1986): dacă o funcție ρ de variabilă t este monotonă pentru $t \in [0, \infty)$, atunci pentru orice set de instanțe distințe x_1, \dots, x_n , matricea pătratică definită prin relația $\rho_{ij} = \rho(\|x_i - x_j\|)$ pentru $i, j \in \{1, \dots, n\}$ este inversabilă.

- Care este valoarea funcției $y(x)$ atunci când $\lambda \rightarrow 0$? Cu alte cuvinte, care este formula de calcul pentru predicțiile făcute de către modelul de regresie *ridge kernel-izată* atunci când se face această trecere la limită?
- Demonstrați că eroarea la antrenare produsă la trecerea la limită (pentru $\lambda \rightarrow 0$) de către regresia *ridge kernel-izată* cu nucleu RBF — presupunând, ca mai înainte, că $x_i \neq x_j$ pentru orice $i \neq j$ — este exact zero.

Răspuns:

- Funcția $f(t) = -\gamma t^2/2$ este monoton decrescătoare în raport cu t (pentru orice $\gamma > 0$ fixat). De asemenea, funcția $g(t) = e^t$ este monoton crescătoare în raport cu t . Prin compunere, funcția $(g \circ f)(t) \stackrel{\text{not.}}{=} h(t) = e^{-\gamma t^2/2}$ va fi monoton decrescătoare în raport cu t .

Funcția-nucleu RBF, care este definită prin relația

$$K(x, x') = e^{-\frac{\gamma}{2}\|x-x'\|^2} = (g \circ f)(\|x-x'\|), \text{ pentru orice } x, x' \in \mathbb{R}^d,$$

satisfac condițiile teoremei lui Michelli. Prin urmare, ea va da naștere, pentru orice set de instanțe $x_1, \dots, x_n \in \mathbb{R}^d$ care sunt diferite între ele, la o matrice Gram care este inversabilă.

- Calculăm mai întâi valoarea, la limită, a vectorului de parametri $\hat{\alpha}$:

$$\begin{aligned} \lim_{\lambda \rightarrow 0} \hat{\alpha} &= \lim_{\lambda \rightarrow 0} ((\mathbf{K} + \lambda \mathbf{I})^{-1} y) = \left(\lim_{\lambda \rightarrow 0} ((\mathbf{K} + \lambda \mathbf{I})^{-1}) \right) y \\ &= \left[\lim_{\lambda \rightarrow 0} (\mathbf{K} + \lambda \mathbf{I}) \right]^{-1} y = K^{-1} y. \end{aligned}$$

Această limită este [întotdeauna] bine-definită, fiindcă

- matricea \mathbf{K} este inversabilă (vedeți punctul a),
- matricea $\mathbf{K} + \lambda \mathbf{I}$, pentru valori mici $\lambda > 0$, este și ea inversabilă (LC: datorită continuității), iar
- pentru astfel de matrice, limita matricelor inversate este inversa matricei-limită.

Folosind acum relația (27), vom putea scrie:

$$\begin{aligned} \lim_{\lambda \rightarrow 0} y(x) &= \lim_{\lambda \rightarrow 0} \left(\sum_{i=1}^n \hat{\alpha}_i K(x_i, x) \right) = \sum_{i=1}^n \lim_{\lambda \rightarrow 0} \hat{\alpha}_i K(x_i, x) \\ &= \sum_{i=1}^n B_i K(x_i, x), \text{ unde } B \stackrel{\text{not.}}{=} [B_1, \dots, B_n]^\top = \mathbf{K}^{-1} y. \end{aligned}$$

În consecință, prin trecerea la limită specificată în enunț, vom avea $y(x) = \sum_{i=1}^n B_i K(x_i, x)$.

c. Pentru a demonstra că eroarea la antrenare este zero, trebuie să arătăm că $y(x_t) = y_t$ pentru orice $t \in \{1, \dots, n\}$. Plecând de la rezultatul obținut la punctul b, putem scrie:

$$\begin{aligned}
 y(x_t) &= \sum_{i=1}^n B_i K(x_i, x_t) \\
 &= \sum_{i=1}^n \left(\sum_{j=1}^n y_j \mathbf{K}^{-1}(i, j) \right) K(x_i, x_t) \stackrel{asoc.}{=} \sum_{j=1}^n y_j \sum_{i=1}^n \mathbf{K}^{-1}(i, j) K(x_i, x_t) \\
 &\stackrel{sim.}{=} \sum_{j=1}^n y_j \sum_{i=1}^n \mathbf{K}^{-1}(j, i) K(x_i, x_t) = \sum_{j=1}^n y_j \sum_{i=1}^n \mathbf{K}^{-1}(j, i) \mathbf{K}(i, t) \\
 &= \sum_{j=1}^n y_j \delta(j, t) \\
 &= y_t,
 \end{aligned}$$

unde $\mathbf{K}^{-1}(i, j)$ elementul de pe poziția (i, j) a matricei \mathbf{K}^{-1} , iar

$$\delta(i, j) = \begin{cases} 0 & \text{pentru } i \neq j \\ 1 & \text{pentru } i = j. \end{cases}$$

În cele de mai sus, am ținut cont în primul rând de faptul că funcția-nucleu K (și, în consecință și matricea \mathbf{K}) este simetrică, ceea ce implică faptul că matricea \mathbf{K}^{-1} este simetrică, deci $\mathbf{K}^{-1}(i, j) = \mathbf{K}^{-1}(j, i)$ pentru orice indici i și j . În al doilea rând, din relația $\mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$ rezultă că $\sum_i \mathbf{K}^{-1}(j, i) \mathbf{K}(i, t) = \delta(j, t)$ pentru orice indici j și t .

11.

(Regresia liniară și fenomenul de overfitting:

Adevărat sau Fals?)

Stanford, 2015 fall, Andrew Ng, midterm, pr. 1.a

Presupunem că, folosind un set [fix] format din m exemple, antrenați un model de *regresie liniară*, $h_\beta(x) = \beta^\top x$, unde β și $x \in \mathbb{R}^{d+1}$. După antrenare, vă dați seama că *varianța* modelului dumneavoastră este relativ mare, deci se produce *supra-specializare* (engl., *overfitting*). Pentru fiecare dintre metodele listate mai jos, asociați fie răspunsul *Adevărat* în caz că metoda respectivă poate contracara / reduce fenomenul de overfitting, fie răspunsul *Fals* în cazul contrar. Explicați de ce.

- a. Extindeți vectorul de trăsături prin adăugarea de noi atrbute / trăsături.
- b. Impuneti condiția ca parametrul (de fapt, vectorul de parametri) β să urmeze o distribuție a priori, și anume o distribuție de forma $\mathcal{N}(0, \tau^2 I)$, iar apoi deduceți valoarea lui β prin metoda estimării de probabilitate maximă a posteriori (engl., maximum a posteriori probability, MAP).

Răspuns:

a. Fals. Adăugarea de noi trăsături va face ca modelul dumneavoastră să fie și mai *complex*. El va reuși să se adapteze la și mai multe excepții (engl., *outliers*) din setul de antrenament și, în consecință să producă și mai mult overfitting.

b. Adevărat. Impunând / cerând ca parametrul β să urmeze o distribuție probabilistă a priori [precum cea indicată în enunț], se limitează în mod efectiv norma vectorului β , fiindcă vectorii mai mari [în normă] au o probabilitate mai mică.⁵¹ Astfel, acest procedeu

⁵¹Vedeți și problema 25.

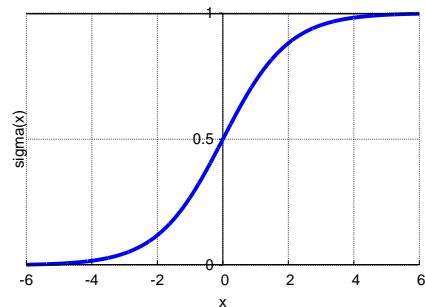
face ca modelul nostru să fie mai puțin predispus la overfitting.

1.1.2 Regresia logistică

12. (Regresia logistică, chestiuni introductive:
estimare MLE; deducerea regulilor de actualizare a parametrilor,
folosind metoda gradientului)
*formulare de Liviu Ciortuz, după
 ■ Stanford, 2016 spring, Chris Piech,
 Introduction to Probability for Computer Scientists course (CS109),
 Logistic Regression*

În învățarea automată, *algoritmii de clasificare* primesc ca date de intrare n instanțe de antrenament care sunt identic și independent distribuite, $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n)}, y^{(n)})$, fiecare vector $x^{(i)}$ având d atrbute. În cazul de față vom presupune că $y^{(i)} \in \{0, 1\}$ pentru $i = 1, \dots, n$.

Regresia logistică este un algoritm de clasificare, care lucrează urmărind să învețe o funcție care să aproximeze în mod convenabil distribuția $P(Y|X)$.⁵² Presupunerea ei de bază este că $P(Y|X)$ poate fi aproximată cu o *funcție sigmoidală* (numită și *funcție logistică*) aplicată unei combinații liniare de atrbutele de intrare.



Din punct de vedere matematic, dată fiind o (singură) instanță de antrenament (x, y) , regresia logistică va considera

$$\begin{aligned} P(Y = 1|X = x) &= \sigma(z) \text{ și, echivalent, } P(Y = 0|X = x) = 1 - \sigma(z), \text{ unde} \\ \sigma(z) &\stackrel{\text{def.}}{=} \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \text{ cu} \\ z &\stackrel{\text{not.}}{=} w_0 + \sum_{i=1}^d w_i x_i = w \cdot x \text{ și } w \stackrel{\text{not.}}{=} (w_0, w_1, \dots, w_d) \in \mathbb{R}^{d+1}, \text{ presupunând } x_0 = 1. \end{aligned} \quad (29)$$

Operatorul \cdot desemnează produsul scalar al vectorilor.⁵³ Pornind de la aceste formule pentru probabilitatea condiționată $Y|X$, putem crea un algoritm care selectează valori

⁵²La capitolul *Clasificare bayesiană* veți vedea că în acest tip de clasificare ideea de bază este că, dată fiind o instanță x , urmărим să alegem ca output acea etichetă y care maximizează probabilitatea condiționată $P(Y = y|X = x)$. Algoritmul Bayes Naiv este [tot] un algoritm de clasificare care modelează / aproximează probabilitatea $P(Y = y|X = x)$ folosind *presupozitia naivă* că toate trăsăturile / atrbutele sunt mutual independente în raport cu eticheta clasei. Regresia logistică nu folosește o astfel de presupozиcie.

⁵³În notație matriceală, dacă ne referim la w și x ca vectori-coloană din același spațiu vectorial, am putea scrie $w^\top x$ în loc de $w \cdot x$. Operatorul \top desemnează operația de transpunere a matricelor / vectorilor.

pentru vectorul w care maximizează această probabilitate pentru întreg setul de date de antrenament.

În acest exercițiu veți calcula mai întâi expresia funcției de log-verosimilitate [condițională] pentru datele $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$. Apoi vă vom arăta modul în care un algoritm iterativ bazat pe *metoda gradientului* poate să selecteze valorile optime pentru parametrii w . În final, veți descoperi cum anume se obțin *regulile de actualizare* a parametrilor din acest algoritm, folosind derivatele parțiale ale funcției de log-verosimilitate în raport cu componentele vectorului w .

- a. Demonstrați că log-verosimilitatea condițională a întregului set de date de antrenament — ținând cont de *presupoziția de bază* a regresiei logistice — este:

$$\ell(w) = \sum_{i=1}^n \left(y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)})) \right). \quad (30)$$

*Observație:*⁵⁴ De fapt, funcția de *log-verosimilitate completă* este următoarea:

$$\begin{aligned} \text{log-likelihood} &= \ln \prod_{i=1}^n P(x^{(i)}, y^{(i)}) = \ln \prod_{i=1}^n (P_{Y|X}(y^{(i)}|x^{(i)}) P_X(x^{(i)})) \\ &= \ln \left(\left(\prod_{i=1}^n P_{Y|X}(y^{(i)}|x^{(i)}) \right) \cdot \left(\prod_{i=1}^n P_X(x^{(i)}) \right) \right) \\ &= \ln \prod_{i=1}^n P_{Y|X}(y^{(i)}|x^{(i)}) + \ln \prod_{i=1}^n P_X(x^{(i)}) \stackrel{\text{not.}}{=} \ell(w) + \ell_x. \end{aligned}$$

Observăm că ℓ_x nu depinde de parametrul w . Întrucât facem estimarea în sensul verosimilității maxime (MLE) a parametrului w , putem să ne limităm la a maximiza $\ell(w)$.

Comentariu: Dispunând de expresia (30) pentru funcția de log-verosimilitate condițională, putem să trecem la identificarea celor valori ale lui w care maximizează această funcție. Spre deosebire de alte situații în care se pune(a) problema găsirii optimului funcției de log-verosimilitate, în acest caz nu există o *formulă analitică* (engl., closed form formula) care să ne permită să obținem în mod direct valoarea optimă a lui w . De aceea, vom identifica această valoare utilizând o *metodă de optimizare*. Mai precis, vom folosi un algoritm iterativ numit *gradientul ascendent*.⁵⁵ Ideea acestui algoritm este că atunci când facem în mod continuu pași mici în direcția gradientului funcției de optimizat — gradientul fiind, conform definiției, vectorul de derivate parțiale al funcției respective —, vom ajunge la un moment dat într-un punct de maxim local al acelei funcții. În cazul regresiei logistice, se poate demonstra că rezultatul aplicării metodei / algoritmului gradientului ascendent va fi întotdeauna punctul de *maxim global*.⁵⁶ Pașii mici pe care îi parcurgem în mod iterativ, dat fiind setul de date de antrenament, sunt calculați / determinați folosind următoarea *regulă de actualizare*:

$$w_j^{new} = w_j^{old} + \eta \frac{\partial}{\partial w_j^{old}} \ell(w^{old}), \quad (31)$$

⁵⁴Cf. pr. 33.

⁵⁵Pentru o ilustrare simplă a modului cum funcționează algoritmul gradientului *descendent*, vedeti problema 72.b de la capitolul de *Fundamente*.

⁵⁶Vedeti pr. 13.

unde η este un factor constant care determină mărimea pasului pe care îl facem la fiecare iterație, cunoscut sub numele de *rată de învățare*.

b. Arătați că forma derivatelor parțiale ale funcției de log-verosimilitate condițională în raport cu fiecare componentă w_j a vectorului w este următoarea:

$$\frac{\partial}{\partial w_j} \ell(w) = \sum_{i=1}^n [y^{(i)} - \underbrace{\sigma(w \cdot x^{(i)})}_{P(Y=1|X=x;w)}] x_j^{(i)} \text{ pentru } j = 0, 1, \dots, d. \quad (32)$$

Sugestie: Puteți folosi următoarea proprietate pentru derivata funcției logistice σ în raport cu argumentul ei:

$$\frac{\partial}{\partial z} \sigma(z) = \sigma(z)[1 - \sigma(z)] \text{ pentru } \forall z \in \mathbb{R}.$$

Observație: Din relația (32) urmează că vectorul gradient $\nabla_w \ell(w)$ se poate scrie astfel:

$$\nabla_w \ell(w) = \sum_{i=1}^n [y^{(i)} - \sigma(w \cdot x^{(i)})] x^{(i)}. \quad (33)$$

La problema 15 vom arăta că $\nabla_w \ell(w)$ se poate scrie sub o formă și mai compactă, folosind așa-numita *matrice de design* X .⁵⁷

c. [Legătura cu *funcția de cost logistică*]⁵⁸

Arătați că dacă în loc de $y^{(i)} \in \{0, 1\}$ vom considera $y'^{(i)} \in \{-1, 1\}$ pentru $i = 1, \dots, n$,⁵⁹ atunci se poate demonstra că următoarea *funcție de cost mediu* (sau de *risc empiric*; engl., *empirical risk*)

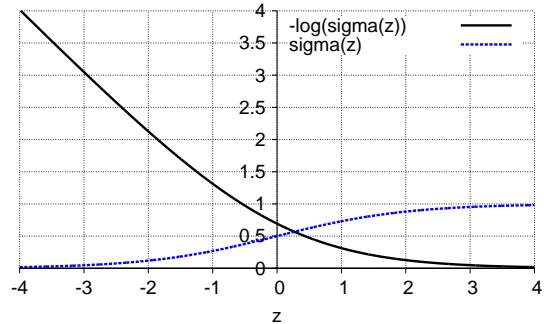
$$J(w) \stackrel{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^n \ln(1 + \exp(-y'^{(i)} w \cdot x^{(i)}))$$

este chiar $-\frac{1}{n} \ell(w)$, unde $\ell(w)$ este funcția de log-verosimilitate condițională. Ca o *consecință*, a maximiza funcția de log-verosimilitate $\ell(w)$ este echivalent cu a minimiza funcția de cost mediu $J(w)$.

Observație: Funcția $\phi(y'w \cdot x) \stackrel{\text{not.}}{=} \ln(1 + \exp(-y'w \cdot x))$ sau, mai simplu, $\phi(z) \stackrel{\text{not.}}{=} \ln(1 + e^{-z})$ este numită *funcția de cost* (sau, pierdere) *logistică*.

Legătura dintre $\phi(z) = \ln(1 + e^{-z})$ pe de o parte și (atenție!) *funcția logistică* (sau *funcția sigmoidală*) $\sigma(z) \stackrel{\text{def.}}{=} \frac{1}{1 + e^{-z}}$ pe de altă parte este următoarea:

$$\phi(z) = \ln(1 + e^{-z}) = -\ln \sigma(z).$$



⁵⁷Acolo de fapt se lucrează cu o formă mai generală de regresie logistică, și anume *regresia logistică local-ponderată*.

⁵⁸Cf. secțiunii 2 (Logistic Regression) din documentul *Supplemental lecture notes* (pentru cursul de *Machine Learning*), de John Duchi, de la Universitatea Stanford.

⁵⁹Definim $y'^{(i)} = -1$ dacă $y^{(i)} = 0$ și $y'^{(i)} = 1$ dacă $y^{(i)} = 1$.

Din această cauză, funcția de cost logistică este numită uneori și *funcția log-sigmoidală*.

Răspuns:

- a. Pentru început, iată o modalitate de a scrie în mod compact expresia (29) din enunț, care reprezintă probabilitatea condiționată a unei *singure* instanțe de antrenament:

$$P(Y = y|X = x) = \sigma(w \cdot x)^y [1 - \sigma(w \cdot x)]^{1-y}, \text{ presupunând că } y \in \{0, 1\}.$$

Întrucât toate instanțele de antrenament sunt independente, verosimilitatea condițională a întregului set de date este următoarea:

$$\prod_{i=1}^n P(Y = y^{(i)}|X = x^{(i)}) = \prod_{i=1}^n \left(\sigma(w \cdot x^{(i)})^{y^{(i)}} [1 - \sigma(w \cdot x^{(i)})]^{1-y^{(i)}} \right). \quad (34)$$

Dacă aplicăm logaritmul natural acestei funcții, obținem imediat expresia indicată în enunț (30) pentru log-verosimilitatea datelor în cazul regresiei logistice.

- b. Calculul derivatelor parțiale ale expresiei $\sigma(w \cdot x)$ în raport cu componentele vectorului w va fi făcut folosind formula de derivare pentru funcții compuse.

Concret, valoarea derivatei parțiale a lui $\sigma(w \cdot x)$ în raport cu componenta w_j a vectorului w , pentru o instanță (x, y) , se calculează astfel:

$$\begin{aligned} & \frac{\partial}{\partial w_j} \ln[\sigma(w \cdot x)^y [1 - \sigma(w \cdot x)]^{1-y}] \\ &= \frac{\partial}{\partial w_j} y \ln \sigma(w \cdot x) + \frac{\partial}{\partial w_j} (1 - y) \ln [1 - \sigma(w \cdot x)] \\ &= \left[\frac{y}{\sigma(w \cdot x)} - \frac{1 - y}{1 - \sigma(w \cdot x)} \right] \frac{\partial}{\partial w_j} \sigma(w \cdot x) \\ &= \left[\frac{y}{\sigma(w \cdot x)} - \frac{1 - y}{1 - \sigma(w \cdot x)} \right] \sigma(w \cdot x)[1 - \sigma(w \cdot x)]x_j \\ &= \frac{y - \sigma(w \cdot x)}{\sigma(w \cdot x)[1 - \sigma(w \cdot x)]} \sigma(w \cdot x)[1 - \sigma(w \cdot x)]x_j \\ &= [y - \sigma(w \cdot x)]x_j. \end{aligned} \quad (35)$$

Întrucât derivata sumei este suma derivatelor, rezultă că derivata parțială a funcției de log-verosimilitate condițională $\ell(w)$ în raport cu w_j este suma acestui tip de termeni, câte unul pentru fiecare instanță de antrenament. Mai exact, după ce aplicăm funcția \ln expresiei (34) și apoi calculăm derivele ei parțiale în raport cu w_j (pentru $j \in \{0, 1, \dots, d\}$), vom obține rezultatul (32), datorită relației (35), pe care am demonstrat-o mai sus.

- c. Demonstrația se bazează în esență pe proprietatea $1 - \sigma(z) = \sigma(-z)$, care este imediată.

Cazul $y'^{(i)} = 1$: $P(Y = 1|X = x^{(i)}) = \sigma(w \cdot x^{(i)}) = \sigma(y'^{(i)}w \cdot x^{(i)})$;

Cazul $y'^{(i)} = -1$: $P(Y' = -1|X = x^{(i)}) = 1 - P(Y = 1|X = x^{(i)}) = 1 - \sigma(w \cdot x^{(i)})$

$$= \sigma(-w \cdot x^{(i)}) = \sigma(y'^{(i)}w \cdot x^{(i)}).$$

Prin urmare,

$$\begin{aligned}
\ell(w) &= \ln \prod_{i=1}^n P(Y = y^{(y_i)} | X = x^{(i)}) = \sum_{i=1}^n \ln P(Y = y^{(y_i)} | X = x^{(i)}) \\
&= \sum_{i=1}^n \ln \sigma(y'^{(i)} w \cdot x^{(i)}) = \sum_{i=1}^n \ln \frac{1}{1 + \exp(-y'^{(i)} w \cdot x^{(i)})} \\
&= - \sum_{i=1}^n \ln(1 + \exp(-y'^{(i)} w \cdot x^{(i)}))).
\end{aligned}$$

Așadar, $-\frac{1}{n}\ell(w) = J(w)$.

13. (Regresia logistică, o proprietate importantă: funcția de log-verosimilitate este concavă, deci are un maxim global; calculul matricei hessiene pentru funcția de log-verosimilitate)
■ Stanford, 2008 fall, Andrew Ng, HW1, pr. 1.a

La problema 12.a am demonstrat că în cazul regresiei logistice funcția de log-verosimilitate condițională se scrie sub forma următoare:

$$\ell(w) = \sum_{i=1}^n \left(y^{(i)} \ln h(x^{(i)}) + (1 - y^{(i)}) \ln(1 - h(x^{(i)})) \right),$$

unde instanțele $x^{(i)}$ sunt vectori-colonă de forma $x = (x_1, \dots, x_d)^\top$, etichetele $y^{(i)} \in \{0, 1\}$, vectorul de ponderi $w \in \mathbb{R}^d$, iar $h(x) \stackrel{\text{def}}{=} \sigma(w \cdot x) = \frac{1}{1 + \exp(-w \cdot x)}$.

Calculați matricea hessiană (H) a acestei funcții, care este prin definiție matricea formată din derivatele parțiale de ordin secund ale funcției ℓ . Apoi arătați că pentru orice $z = (z_1, \dots, z_d)^\top$, are loc proprietatea

$$z^\top H z \leq 0.$$

Observație: Aceasta este una dintre modalitățile clasice de a arăta că matricea H este negativ semidefinită, ceea ce se notează sub forma $H \leq 0$. Aceasta implică faptul că ℓ este funcție concavă și, în consecință, ea nu are decât o singură valoare maximă (care este și maxim global).

Răspuns:

Vă reamintim o proprietate importantă a funcției logistice: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$. Prin urmare, $\frac{\partial h(x)}{\partial w_k} = h(x)(1 - h(x))x_k$, fiindcă $h(x) = \sigma(w \cdot x)$. Acest fapt este utilizat în cele ce urmează.

Calculăm mai întâi derivatele parțiale de ordinul întâi ale funcției de log-verosimilitate:

$$\begin{aligned}
\frac{\partial \ell(w)}{\partial w_k} &= \frac{\partial}{\partial w_k} \left(\sum_{i=1}^n y^{(i)} \ln h(x^{(i)}) + (1 - y^{(i)}) \ln(1 - h(x^{(i)})) \right) \\
&= \sum_{i=1}^n \left[y^{(i)} \frac{x_k^{(i)}}{h(x^{(i)})} - (1 - y^{(i)}) \frac{x_k^{(i)}}{1 - h(x^{(i)})} \right] h(x^{(i)})(1 - h(x^{(i)}))
\end{aligned}$$

$$= \sum_{i=1}^n x_k^{(i)} \left[y^{(i)} - \cancel{y^{(i)} h(x^{(i)})} - h(x^{(i)}) + \cancel{y^{(i)} h(x^{(i)})} \right] = \sum_{i=1}^n (y^{(i)} - h(x^{(i)})) x_k^{(i)}. \quad (36)$$

În consecință, elementul generic al matricei hessiene (H) este

$$H_{kl} = \frac{\partial^2 \ell(w)}{\partial w_k \partial w_l} = \sum_{i=1}^n -\frac{\partial h(x^{(i)})}{\partial w_l} x_k^{(i)} = \sum_{i=1}^n -h(x^{(i)})(1-h(x^{(i)})) x_l^{(i)} x_k^{(i)}. \quad (37)$$

Tinând cont de faptul că putem scrie xx^\top ca o matrice X [dacă și numai dacă $X_{ij} = x_i x_j$], rezultă că matricea hessiană H este:⁶⁰

$$H = - \underbrace{\sum_{i=1}^n h(x^{(i)})(1-h(x^{(i)}))}_{\in \mathbb{R}_+} x^{(i)} (x^{(i)})^\top.$$

Mai departe, pentru a demonstra că matricea H este negativ semidefinită, vom arăta că pentru orice vector-coloană z are loc inegalitatea $z^\top H z \leq 0$.

$$\begin{aligned} z^\top H z &= -z^\top \left(\sum_{i=1}^n h(x^{(i)})(1-h(x^{(i)})) x^{(i)} (x^{(i)})^\top \right) z \\ &= -\sum_{i=1}^n h(x^{(i)})(1-h(x^{(i)})) z^\top x^{(i)} \underbrace{(x^{(i)})^\top z}_{(z^\top x^{(i)})^\top} \\ &= -\sum_{i=1}^n h(x^{(i)})(1-h(x^{(i)}))(z^\top x^{(i)})^2 \leq 0. \end{aligned}$$

Ultima inegalitate de mai sus se justifică astfel: $(z^\top x^{(i)})^2 \geq 0$ pentru orice z și orice $x^{(i)}$ din \mathbb{R}^d , iar din definiția funcției logistice σ rezultă $0 < h(x^{(i)}) < 1$, ceea ce implică $h(x^{(i)})(1-h(x^{(i)})) > 0$.

Observație importantă: Demonstrația de mai sus furnizează tot ce este necesar pentru obținerea [ulterioră a] relației de actualizare a parametrilor la *aplicarea metodei lui Newton* în cazul regresiei logistice.⁶¹

14.

(Regresia logistică:
estimarea parametrilor în sens MAP — regularizare de normă L_2 ;
efectul de *diminuare a ponderilor* (engl., weight decay))

*prelucrare de Liviu Ciortuz, după
Stanford, 2008 fall, Andrew Ng, HW3, pr. 4*

Presupunem că folosim un model de *regresie logistică* $h_\theta(x) = \sigma(\theta \cdot x)$, unde σ este *funcția logistică* (sau *sigmoidală*), și că dispunem de un set de date de antrenament $D = \{(x^{(i)}, y^{(i)}); i = 1, \dots, n\}$ definit ca de obicei.

⁶⁰La problema 15 vom arăta că H se poate scrie și mai compact, folosind așa-numita *matrice de design* X . (Veți observa că coloanele de fapt se lucrează cu o formă mai generală de regresie logistică, și anume *regresia logistică local-ponderată*.)

⁶¹Pentru o descriere a metodei lui Newton, vedeți *Comentariul* din enunțul problemei 72 de la capitolul de *Fundamente*. Problema 7 de la prezentul capitol arată cum anume se aplică metoda lui Newton în cazul regresiei liniare.

Remember: Estimarea de verosimilitate maximă a parametrilor θ este definită astfel:

$$\theta_{ML} \stackrel{\text{def.}}{=} \operatorname{argmax}_{\theta} P(D|\theta) \stackrel{i.i.d.}{=} \operatorname{argmax}_{\theta} \prod_{i=1}^n P(y^{(i)}|x^{(i)}; \theta). \quad (38)$$

Dacă dorim să regularizăm regresia logistică, putem impune ca [vectorul de] parametri θ să urmeze o distribuție probabilistă a priori P . Estimarea de probabilitate maximă a posteriori pentru θ este definită astfel:

$$\begin{aligned} \theta_{MAP} &\stackrel{\text{def.}}{=} \operatorname{argmax}_{\theta} P(\theta|D) \stackrel{F.B.}{=} \operatorname{argmax}_{\theta} \frac{P(D|\theta)P(\theta)}{P(D)} = \operatorname{argmax}_{\theta} P(D|\theta)P(\theta) \quad (39) \\ &\stackrel{i.i.d.}{=} \operatorname{argmax}_{\theta} P(\theta) \prod_{i=1}^m P(y^{(i)}|x^{(i)}; \theta). \end{aligned} \quad (40)$$

Presupunem că alegem ca distribuție a priori pentru θ distribuția gaussiană multidimensională $\mathcal{N}(0, \tau^2 I)$, cu $\tau > 0$ și I matricea identitate de dimensiune $(d+1) \times (d+1)$.⁶²

a. Arătați că atunci când parametrul θ urmează o distribuție probabilistă de genul celei indicate mai sus, efectul practic este următorul: maximizarea funcției de probabilitate a posteriori $P(\theta|D)$ este echivalentă cu maximizarea sumei dintre funcția de log-verosimilitate $\ln P(D|\theta)$ și un termen de forma $-\lambda\|\theta\|^2$, unde $\lambda = \frac{1}{2\tau^2}$. (Astfel se justifică denumirea de *regularizare de normă L₂* sau, mai simplu, *regularizare L₂*.)

b. Demonstrați următoarea inegalitate:

$$\|\theta_{MAP}\|^2 \leq \|\theta_{ML}\|^2.$$

Răspuns:

a. Înținând cont de faptul că funcția \ln este strict creșătoare pe întreg domeniul ei de definiție, putem rescrie egalitatea reprezentată de primul și ultimul termen din relația (38) sub forma următoare:

$$\theta_{ML} = \operatorname{argmax}_{\theta} \ln \prod_{i=1}^n P(y^{(i)}|x^{(i)}; \theta) = \operatorname{argmax}_{\theta} \underbrace{\sum_{i=1}^n \ln P(y^{(i)}|x^{(i)}; \theta)}_{\text{not.: } \ell(\theta)}.$$

Similar, rescriem relația (40) sub forma

$$\begin{aligned} \theta_{MAP} &= \operatorname{argmax}_{\theta} \ln P(\theta) \prod_{i=1}^m P(y^{(i)}|x^{(i)}; \theta) = \operatorname{argmax}_{\theta} (\ln P(\theta) + \sum_{i=1}^m \ln P(y^{(i)}|x^{(i)}; \theta)) \\ &= \operatorname{argmax}_{\theta} (\ln P(\theta) + \ell(\theta)) \\ &= \operatorname{argmax}_{\theta} \left(\ln \left(\frac{1}{(2\pi)^{(d+1)/2} \tau^{d+1}} \exp \left(-\frac{1}{2} \theta^\top \frac{1}{\tau^2} I \theta \right) \right) + \ell(\theta) \right) \\ &= \operatorname{argmax}_{\theta} \left(-\ln \left((2\pi)^{(d+1)/2} \tau^{d+1} \right) - \frac{1}{2\tau^2} \theta^2 + \ell(\theta) \right). \end{aligned}$$

⁶²Am scris $d+1$ în loc de d fiindcă se consideră că se lucrează și cu termenul liber θ_0 , deci $\theta = (\theta_0, \theta_1, \dots, \theta_d)$.

Întrucât expresia $(2\pi)^{(d+1)/2} \tau^{d+1}$ nu depinde de θ , rezultă că

$$\theta_{MAP} = \operatorname{argmax}_{\theta} \left(- \underbrace{\frac{1}{2\tau^2}}_{not.: \lambda} \theta^2 + \ell(\theta) \right) = \operatorname{argmax}_{\theta} \left(\ell(\theta) - \lambda \|\theta\|^2 \right). \quad (41)$$

b. Presupunem prin reducere la absurd că $\|\theta_{MAP}\|^2 > \|\theta_{ML}\|^2$. Rezultă că

$$\begin{aligned} P(\theta_{MAP}) &= \frac{1}{(2\pi)^{\frac{d+1}{2}} |\tau^2 I|^{\frac{1}{2}}} \exp \left(- \frac{1}{2\tau^2} \|\theta_{MAP}\|^2 \right) \\ &< \frac{1}{(2\pi)^{\frac{d+1}{2}} |\tau^2 I|^{\frac{1}{2}}} \exp \left(- \frac{1}{2\tau^2} \|\theta_{ML}\|^2 \right) \\ &= P(\theta_{ML}). \end{aligned}$$

În consecință,

$$\begin{aligned} P(\theta_{MAP})P(D|\theta_{MAP}) &< P(\theta_{ML})P(D|\theta_{MAP}) \\ &\leq P(\theta_{ML})P(D|\theta_{ML}). \end{aligned}$$

Ultima inegalitate de mai sus are loc încrucișat prin definiție θ_{ML} maximizează verosimilitatea datelor, $P(D|\theta)$. Însă rezultatul $P(\theta_{MAP})P(D|\theta_{MAP}) < P(\theta_{ML})P(D|\theta_{ML})$ constituie o contradicție, fiindcă θ_{MAP} prin definiție maximizează produsul $P(\theta)P(D|\theta)$. Prin urmare, presupunerea $\|\theta_{MAP}\|^2 > \|\theta_{ML}\|^2$ este falsă.

Observații:

1. Remarcați faptul că în vreme ce la regresia liniară termenul de regularizare era precedat de semnul + (vedeți relația (5) de la problema 3.C), în relația (40) termenul de regularizare este precedat de semnul -. Explicația rezidă în faptul că la regresia liniară — varianta LSE — se caută *minimul* sumei pătratelor erorilor, iar la regresia logistică se caută *maximul* funcției de verosimilitate condițională.
2. Inegalitatea $\|\theta_{MAP}\|^2 \leq \|\theta_{ML}\|^2$ indușă de regularizarea de normă L_2 este numită uneori și proprietatea de *diminuare a ponderilor* (engl., *weight decay*), fiindcă regularizarea aceasta încurajează ponderile / parametrii să ia valori care în general sunt mai mici [LC: în valoare absolută] decât în cazul în care nu se folosește regularizare, ceea ce conduce la limitarea / prevenirea fenomenului de *overfitting*. Veți observa din demonstrație că această proprietate indușă de regularizarea L_2 este valabilă în general, nu doar pentru regresia logistică.⁶³

15.

(Regresia logistică local-ponderată regularizată (L_2): exprimarea vectorului gradient și a matricei hessiene cu ajutorul matricei de design, X)
prelucrare de Liviu Ciortuz, după Stanford, 2007 fall, Andrew Ng, HW1, pr. 2

Problema de regresie logistică local-ponderată cu regularizare L_2 constă în a maximiza o funcție de log-verosimilitate condițională de forma

⁶³Mentionăm că în contextul regresiei liniare, proprietatea de diminuarea a ponderilor apare la problema 25.

$$\ell(\theta) = \sum_{i=1}^n w_i [y^{(i)} \ln h_\theta(x^{(i)}) + (1 - y^{(i)}) \ln(1 - h_\theta(x^{(i)}))] - \frac{\lambda}{2} \|\theta\|^2,$$

unde $x^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \{0, 1\}$, $w_i \in \mathbb{R}_+$ pentru $i = 1, \dots, n$, $\theta \in \mathbb{R}^d$, iar h_θ desemnează o funcție de tip sigmoidal / logistic definită prin $h_\theta(x) = \frac{1}{1 + e^{-\theta \cdot x}}$ pentru orice $x \in \mathbb{R}^d$.⁶⁴ Remarcați faptul că $-\frac{\lambda}{2} \|\theta\|^2 = -\frac{\lambda}{2} \theta^2$ din expresia de mai sus este termenul de regularizare; el este necesar pentru ca antrenarea să nu fie afectată de fenomenul de *overfitting* (rom., supra-specializare).⁶⁵

a. Demonstrați că vectorul gradient pentru $\ell(\theta)$ este

$$\nabla_\theta \ell(\theta) = X^\top z - \lambda \theta, \quad (42)$$

unde X este *matricea de design* (care are dimensiunea $n \times d$, fiecare linie a acestei matrice fiind reprezentată de către o instanță de antrenament), iar vectorul-coloană $z \in \mathbb{R}^n$ este de forma $(z_1, \dots, z_n)^\top$, cu

$$z_i = w_i(y^{(i)} - h_\theta(x^{(i)})), \text{ pentru } i = 1, \dots, n.$$

b. Demonstrați că matricea hessiană pentru $\ell(\theta)$ este

$$H = X^\top D X - \lambda I,$$

unde $D \in \mathbb{R}^{n \times n}$ este o matrice diagonală, în care

$$D_{ii} = -w_i h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})).$$

Observație: Dacă se consideră $w_i = 1$ pentru $i = 1, \dots, m$ și se elimină termenul corespunzător regularizării — adică, se revine la modelul de regresie logistică introdus la problema 12 —, atunci expresia (42) va corespunde expresiei (32) de la problema 12 (și totodată expresiei (36) de la problema 13). De asemenea, elementul generic al matricei H de mai sus va corespunde expresiei (37) de la problema 13. În practică, folosirea matricei de design X este recomandată a fi folosită ori de către ori este posibil, din motive de eficiență computațională.

Răspuns:⁶⁶

a. Mai întâi vom calcula expresia termenului generic pentru vectorul gradient $\nabla_\theta l(\theta)$, generalizând ușor raționamentul de la problema 13, adică adăugând ponderile w_i , precum și termenul de regularizare $-\frac{\lambda}{2} \theta^2$. Vă reamintim că $\frac{\partial}{\partial \theta_k} h_\theta(x) = h_\theta(x)(1 - h_\theta(x))x_k$.

$$\frac{\partial}{\partial \theta_k} \ell(\theta) = \sum_{i=1}^m w_i \left[y^{(i)} \frac{\partial}{\partial \theta_k} \ln(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta_k} \ln(1 - h_\theta(x^{(i)})) \right] - \frac{\lambda}{2} \frac{\partial}{\partial \theta_k} \theta^2$$

⁶⁴Vectorului w de la problemele 12 și 13 îi corespunde aici vectorul θ . În schimb, în problema de față ponderea w_i reprezintă „importanță“ acordată instanței de antrenament $(x^{(i)}, y^{(i)})$ în determinarea separatorului decizional produs de regresia logistică pe aceste date. Separatorul decizional determinat de acest model, și anume $\theta \cdot x = 0$, va fi tot liniar.

⁶⁵Pentru regularizarea de normă L_2 a formei de bază pentru regresia logistică, vedeți problema 14.

⁶⁶Soluția care urmează a fost redactată într-o formă inițială de către Mădălina Racoviță, MSc student, UAIC, Iași, 2020.

$$\begin{aligned}
&= \sum_{i=1}^m w_i \left[y^{(i)} \cdot \frac{1}{h_\theta(x^{(i)})} \cdot h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \cdot x_k^{(i)} + \right. \\
&\quad \left. + (1 - y^{(i)}) \cdot \frac{1}{1 - h_\theta(x^{(i)})} \cdot (-1) \cdot h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) \cdot x_k^{(i)} \right] - \frac{\lambda}{2} \cdot \|\theta\|_2^2 \\
&= \sum_{i=1}^m w_i \left[y^{(i)} \cdot (1 - h_\theta(x^{(i)})) x_k^{(i)} - (1 - y^{(i)}) \cdot h_\theta(x^{(i)}) x_k^{(i)} \right] - \lambda \|\theta\|_2^2 \\
&= \sum_{i=1}^m w_i \left(y^{(i)} x_k^{(i)} - \cancel{y^{(i)} h_\theta(x^{(i)}) x_k^{(i)}} - h_\theta(x^{(i)}) x_k^{(i)} + \cancel{y^{(i)} h_\theta(x^{(i)}) x_k^{(i)}} \right) - \lambda \|\theta\|_2^2 \\
&= \sum_{i=1}^m w_i \underbrace{(y^{(i)} - h_\theta(x^{(i)}))}_{z_i} x_k^{(i)} - \lambda \|\theta\|_2^2 \\
&= \sum_{i=1}^m z_i x_k^{(i)} - \lambda \|\theta\|_2^2, \text{ pentru } k = 1, \dots, d.
\end{aligned}$$

În consecință, vectorul gradient, $\nabla_\theta \ell(\theta)$, se scrie astfel:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^m z_i x^{(i)} - \lambda \theta.$$

Acum vom arăta că $\nabla_\theta \ell(\theta) = X^\top z - \lambda \theta$, unde X este *matricea de design*. Conform rezultatului demonstrat mai sus, va fi suficient să arătăm că $X^\top z = \sum_{i=1}^m z_i x^{(i)}$. Într-adevăr,

$$\begin{aligned}
X^\top z &= (x^{(1)\top}, \dots, x^{(m)\top}) \times (z_1, \dots, z_m)^\top = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ \vdots & \vdots & & \vdots \\ x_d^{(1)} & x_d^{(2)} & \dots & x_d^{(m)} \end{pmatrix} \times \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix} \\
&= \left(\sum_{i=1}^m z_i x_1^{(i)}, \dots, \sum_{i=1}^m z_i x_d^{(i)} \right)^\top = \sum_{i=1}^m z_i x^{(i)}.
\end{aligned}$$

b. Mai întâi vom calcula expresia termenului generic al matricei hessiane pentru funcția de log-verosimilitate condițională regularizată, $\ell(\theta)$.

$$\begin{aligned}
H_{k,l} &= \frac{\partial^2}{\partial \theta_k \partial \theta_l} \ell(\theta) = \frac{\partial}{\partial \theta_l} \left[\sum_{i=1}^m w_i (y^{(i)} - h_\theta(x^{(i)})) x_k^{(i)} - \lambda \|\theta\|_2^2 \right] \\
&= \sum_{i=1}^m w_i \cdot (-1) \cdot \left(\frac{\partial}{\partial \theta_l} h_\theta(x^{(i)}) \right) \cdot x_k^{(i)} - \lambda \frac{\partial}{\partial \theta_l} \theta_k \\
&= - \sum_{i=1}^m w_i h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) x_l^{(i)} x_k^{(i)} - \lambda \frac{\partial}{\partial \theta_l} \theta_k,
\end{aligned}$$

pentru orice $k, l \in \{1, \dots, d\}$.

Așadar,

$$H_{k,l} = \begin{cases} - \sum_{i=1}^m w_i h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) (x_k^{(i)})^2 - \lambda & \text{pentru } k = l, \\ - \sum_{i=1}^m w_i h_\theta(x^{(i)}) (1 - h_\theta(x^{(i)})) x_l^{(i)} x_k^{(i)} & \text{pentru } k \neq l. \end{cases}$$

Folosind notația $D_{ii} = -w_i h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))$ dată în enunț, putem scrie:

$$H_{k,l} = \begin{cases} \sum_{i=1}^m D_{i,i}(x_k^{(i)})^2 - \lambda & \text{pentru } k = l, \\ \sum_{i=1}^m D_{i,i} x_l^{(i)} x_k^{(i)} & \text{pentru } k \neq l. \end{cases}$$

Similar cu modul cum am procedat la punctul a , vom arăta că matricea H este egală cu $X^\top DX - \lambda I$, ceea ce revine la a demonstra că $X^\top DX = H + \lambda I$.

$$\begin{aligned} X^\top DX &= (x^{(1)}, \dots, x^{(m)}) \times \begin{bmatrix} D_{1,1} & \dots & 0 \\ & \ddots & \\ 0 & \dots & D_{m,m} \end{bmatrix} \times (x^{(1)}, \dots, x^{(m)})^\top \\ &= (D_{11} x^{(1)}, \dots, D_{m,m} x^{(m)}) \times (x^{(1)}, \dots, x^{(m)})^\top = \sum_{i=1}^m D_{i,i} x^{(i)^\top} x^{(i)} \\ &= \begin{bmatrix} \sum_{i=1}^m D_{i,i} \cdot (x_1^{(i)})^2 & \sum_{i=1}^m D_{i,i} \cdot x_1^{(i)} x_2^{(i)} & \dots \\ \sum_{i=1}^m D_{i,i} \cdot x_2^{(i)} x_1^{(i)} & \sum_{i=1}^m D_{i,i} \cdot (x_2^{(i)})^2 & \dots \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^m D_{i,i} \cdot x_d^{(i)} x_1^{(i)} & \dots & \sum_{i=1}^m D_{i,i} \cdot (x_d^{(i)})^2 \end{bmatrix} \\ &= H + \lambda I. \end{aligned}$$

16.

(Regresia logistică: kernel-izare
în cazul folosirii metodei gradientului [ascendent])

■ prelucrare de Liviu Ciortuz, după
CMU, 2005 fall, Tom Mitchell, HW3, pr. 2.ab

Introducere: Ideea centrală din spatele metodei kernel-izării, aşa cum este folosită în clasificare,⁶⁷ este transformarea / „maparea“ vectorilor care reprezintă instanțe de antrenament, provenind dintr-un spațiu \mathcal{X} care [în mod normal] are un număr mic de dimensiuni, într-un alt spațiu \mathcal{Z} (numit îndeobște *spațiu de trăsături*), care are [de obicei] un număr mare de dimensiuni. O astfel de transformare poate da naștere [în anumite condiții] unui clasificator mai flexibil, care moștenește însă eficiența computațională din spațiul inițial \mathcal{X} . Mai concret, *kernel-izarea* implică găsirea unei transformări (sau, a unei mapări, de la engl. mapping) $\phi : \mathcal{X} \rightarrow \mathcal{Z}$, astfel încât

- i. spațiul \mathcal{Z} să aibă un număr de dimensiuni mai mare decât cel al spațiului \mathcal{X} ;
- ii. calculele pe care ar / va trebui să le facem cu vectori din spațiul \mathcal{Z} să se limiteze doar la folosirea produsului scalar;
- iii. să existe o funcție $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, astfel încât pentru orice x_i și x_j din \mathcal{X} produsul scalar dintre $\phi(x_i)$ și $\phi(x_j)$ să fie egal cu $K(x_i, x_j)$, însă valoarea $K(x_i, x_j)$ să poată fi calculată folosind în mod direct x_i și x_j (adică fără a face apel la $\phi(x_i)$ și $\phi(x_j)$). Vom

⁶⁷ Metoda kernel-izării a fost introdusă inițial pentru mașinile cu vectori-suport (engl., Support Vector Machines, SVM). Puteți consulta capitolul respectiv din prezența culegere și (eventual, în prealabil) secțiunea *Funcții-nucleu* de la capitolul de *Fundamente*.

numi K *funcție-nucleu* (engl., kernel function).⁶⁸

Veți vedea că un clasificator liniar din spațiul \mathcal{Z} va corespunde (în general) unui clasificator neliniar din spațiul \mathcal{X} .

Așa cum am arătat deja la problema 12, în varianta *standard* a regresiei logistice se consideră

$$\begin{aligned} P(Y = 1|X) &= \sigma(w_0 + \sum_{i=1}^d w_i X_i) \\ P(Y = 0|X) &= 1 - P(Y = 1|X), \end{aligned}$$

unde σ este funcția logistică, definită prin relația $\sigma(a) \stackrel{\text{def.}}{=} 1/(1+e^{-a})$ pentru orice $a \in \mathbb{R}$. Fie o funcție ϕ care transformă o instanță oarecare X din spațiul \mathcal{X} (de dimensiune d) într-un element din spațiul \mathcal{Z} (a cărui dimensiune este m , cu $m > d$). Folosind „maparea“ ϕ , vom scrie:

$$P(Y = 1|\phi(X)) = \sigma(w_0 + \sum_{i=1}^M w_i \phi(X)_i). \quad (43)$$

unde $\phi(X)$ este vectorul de „trsături“ (engl., feature vector) m -dimensional, care îi corespunde vectorului X din spațiul \mathcal{X} (d -dimensional). În cele ce urmează, vom nota cu $\phi(X)_i$ componenta i a vectorului $\phi(X)$.

a. Presupunem că vectorul de ponderi w este o combinație liniară de toți vectorii de trsături $\phi(X^{(i)})$ cu $i = 1, \dots, n$, unde n este numărul instanțelor de antrenament.⁶⁹ Din punct de vedere formal, putem exprima acest fapt astfel:

$$(w_1, \dots, w_m)^\top = \sum_{j=1}^n \alpha_j \phi(X^{(j)}), \quad (44)$$

unde $X^{(i)}$ este instanță cu numărul de ordine i , iar α_i sunt constante reale ($i = 1, \dots, n$). Presupunem de asemenea că $w_0 = \alpha_0$.

Exprimați probabilitatea condiționată $P(Y = 1|\phi(X))$ folosind procedeul kernel-izării (engl., kernel trick). (Astfel veți evita să faceți calculele în mod explicit în spațiul \mathcal{Z} .)

b. Scrieți *regula de actualizare* specifică metodei gradientului ascendent pentru varianta kernel-izată a regresiei logistice.

c. Scrieți *regula de decizie* pentru o instanță oarecare de test X din spațiul \mathcal{X} , presupunând că parametrii α_i (pentru $i = 1, \dots, m$) au fost deja învățați.

Răspuns:

⁶⁸Funcția K trebuie să fie și pozitivă definită (însă la acest exercițiu nu vom face uz de această proprietate). Spre exemplu, nucleul gaussian $K(x, x') \stackrel{\text{def.}}{=} \frac{1}{\sqrt{2\pi\tau}} \exp\left(\frac{\|x - x'\|^2}{2\tau^2}\right)$ este o astfel de funcție-nucleu.

⁶⁹Dacă aplicăm metoda gradientului [ascendent] pornind cu valoarea $w = 0$, conform regulii de actualizare (31) și expresiei (33) din cadrul problemei 12, rezultă că într-adevăr vom obține vectorul de ponderi w sub forma unei combinații liniare de instanțe $X^{(i)}$ (respectiv $\phi(X^{(i)})$ în cazul kernel-izării).

a. Pornind de la relația (43), putem exprima probabilitatea condiționată $P(Y = 1|\phi(X))$ astfel:

$$\begin{aligned} P(Y = 1|\phi(X)) &= \sigma(w_0 + \sum_{i=1}^m w_i \phi(X)_i) = \sigma(w_0 + w \cdot \phi(X)) \\ &\stackrel{(44)}{=} \sigma(w_0 + (\sum_{j=1}^n \alpha_j \phi(X^{(j)})) \cdot \phi(X)) = \sigma(w_0 + \sum_{j=1}^n \alpha_j (\phi(X^{(j)})) \cdot \phi(X)) \\ &= \sigma(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X)). \end{aligned}$$

b. În cele ce urmează, vom folosi deja-cunoscutul „artificiu“ necesar pentru a exprima în mod unitar probabilitățile $P(Y = 1|\phi(X))$ și $P(Y = 0|\phi(X))$:⁷⁰

$$P(Y = y|\phi(X)) = \sigma(z)^y (1 - \sigma(z))^{1-y} \text{ pentru orice } y \in \{0, 1\},$$

unde

$$z \stackrel{\text{not.}}{=} \alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X). \quad (45)$$

De asemenea, vom ține cont de următoarele proprietăți ale funcției logistică:

$$\begin{aligned} \sigma(z) &= \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z} \\ \Rightarrow 1 - \sigma(z) &= \frac{e^{-z}}{1 + e^{-z}} = \frac{1}{1 + e^z} \\ \Rightarrow \ln(1 - \sigma(z)) &= -\ln(1 + e^z), \end{aligned}$$

pentru orice $z \in \mathbb{R}$.

Prin urmare, vom putea exprima log-verosimilitatea unei instanțe de antrenament oarecare $(\phi(X), Y = y)$ astfel:

$$\begin{aligned} \ln P(Y = y|\phi(X)) &= y \ln \sigma(z) + (1 - y) \ln(1 - \sigma(z)) \\ &= y \ln \frac{e^z}{1 + e^z} + (1 - y) \ln \frac{1}{1 + e^z} \\ &= yz - y \ln(1 + e^z) - \ln(1 + e^z) + y \ln(1 + e^z) \\ &= yz - \ln(1 + e^z). \end{aligned} \quad (46)$$

Datele de antrenament fiind îndeobște considerate independente și identic distribuite (abrev., i.i.d.), vom putea scrie probabilitatea condiționată a setului de date $(\phi(X^{(1)}), Y^{(1)}), \dots, (\phi(X^{(n)}), Y^{(n)})$ astfel:

$$P(Y^{(1)}, \dots, Y^{(n)} | \phi(X^{(1)}), \dots, \phi(X^{(n)}); \alpha) \stackrel{i.i.d.}{=} \prod_{l=1}^n P(Y^{(l)} | \phi(X^{(l)}); \alpha),$$

unde $\alpha \stackrel{\text{not.}}{=} (\alpha_1, \dots, \alpha_n)$ este vectorul de coeficienți care au fost introdusi la relația (44).

⁷⁰Am folosit deja acest „artificiu“ la rezolvarea problemei 12.a.

Așadar, log-verosimilitatea condițională a setului de date $\{(\phi(X^{(1)}), Y^{(1)}), \dots, (\phi(X^{(n)}), Y^{(n)})\}$, exprimată ca funcție de α , este următoarea:

$$\begin{aligned} \ell(\alpha) &= \ln \prod_{l=1}^n P(Y^{(l)} | \phi(X^{(l)}); \alpha) = \sum_{l=1}^n \ln P(Y^{(l)} | \phi(X^{(l)}); \alpha) \\ &\stackrel{(45)(46)}{=} \sum_{l=1}^n \left[Y^{(l)} \left(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)}) \right) - \ln \left(1 + \exp(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)})) \right) \right]. \end{aligned}$$

De aici se obțin ușor derivatele parțiale ale funcției $\ell(\alpha)$ în raport cu componenta α_i , pentru $i = 1, \dots, n$:⁷¹

$$\begin{aligned} \frac{\partial \ell(\alpha)}{\partial \alpha_i} &= \sum_{l=1}^n \left(Y^{(l)} - \frac{\exp(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)}))}{1 + \exp(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)}))} \right) K(X^{(i)}, X^{(l)}) \\ &= \sum_{l=1}^n \left(Y^{(l)} - \sigma(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)})) \right) K(X^{(i)}, X^{(l)}). \end{aligned} \quad (47)$$

și respectiv derivata parțială a funcției $\ell(\alpha)$ în raport cu componenta α_0 :

$$\begin{aligned} \frac{\partial \ell(\alpha)}{\partial \alpha_0} &= \sum_{l=1}^n \left(Y^{(l)} - \frac{\exp(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)}))}{1 + \exp(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)}))} \right) \\ &= \sum_{l=1}^n \left(Y^{(l)} - \sigma(\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X^{(l)})) \right). \end{aligned}$$

Regula de actualizare specifică metodei gradientului ascendent aplicată la rezolvarea acestui problema de regresie logistică kernel-izată este:

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + \eta \frac{\partial \ell(\alpha)}{\partial \alpha_i} \text{ pentru } i = 1, \dots, n,$$

sau, scriind vectorial: $\alpha^{(t+1)} = \alpha^{(t)} + \eta \nabla_\alpha \ell(\alpha)$.

c. Conform relației (43), modelul învățat de regresia logistică asociază instanței X eticheta $Y = 1$ dacă și numai dacă $\sigma(w_0 + \sum_{i=1}^m w_i \phi(X)_i) \geq 1/2$, ceea ce este echivalent cu $w_0 + \sum_{i=1}^m w_i \phi(X)_i \geq 0$, adică $w_0 + w \cdot \phi(X) \geq 0$. Înținând cont de relația (44) și de faptul că $w_0 = \alpha_0$, această ultimă inegalitate revine la $\alpha_0 + \sum_{j=1}^n \alpha_j \phi(X^{(j)}) \cdot \phi(X) \geq 0$, adică

$$\alpha_0 + \sum_{j=1}^n \alpha_j K(X^{(j)}, X) \geq 0.$$

17. (Regresia logistică: clasificare n -ară [regresie softmax] cu regularizare L_2 , folosind metoda gradientului ascendent)

■ CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 2

⁷¹A se compara expresia (47) cu expresia (35) care a fost obținută la problema 12, pentru regresia logistică nekernel-izată.

În acest exercițiu vom arăta că putem extinde ușor modelul binar al regresiei logistice în aşa fel încât să realizăm clasificare n -ară. Să presupunem că avem K clase distincte și că probabilitatea a posteriori pentru clasa k este definită prin

$$\begin{aligned} P(Y = k|X = x) &= \frac{\exp(w_k \cdot x)}{1 + \sum_{l=1}^{K-1} \exp(w_l \cdot x)} \text{ pentru } k = 1, \dots, K-1 \\ P(Y = K|X = x) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp(w_l \cdot x)}, \end{aligned}$$

unde x și w_k cu $k = 1, \dots, K$ sunt vectori d -dimensionali. Remarcați că pentru a simplifica expresia acestor probabilități nu am folosit componentele w_{k0} .⁷² Remarcați de asemenea că pentru $K = 2$ se obține regresia logistică binară, aşa cum a fost prezentată la problema 12.

Observație: Putem simplifica expresia de mai sus pentru regresia logistică n -ară (engl., multiclass logistic regression) introducând un parametru fix, vectorul $w_K = 0$ (un vector d -dimensional format în întregime din zerouri).⁷³

Scopul nostru este să estimăm parametrii / ponderile w_t folosind ca metodă de optimizare gradientul ascendent. Vom stabili de asemenea distribuțiile *a priori* pe care le urmează parametrii, pentru a evita ca ei să [ajungă să] aibă valori foarte mari [în valoare absolută] și să se producă *overfitting* (rom., supra-specializare).

- a. Presupunem că se dă o matrice de antrenament de dimensiune $n \times d$, unde n este numărul de exemple de antrenament, iar d este numărul de atribute. Scrieți expresia funcției de log-verosimilitate condițională, $L(w_1, \dots, w_K)$, folosind *regularizare* de normă L_2 pentru ponderi.⁷⁴ Arătați pașii pe care i-ați urmat ca să deduceți această expresie.
- b. Remarcați că, asemenea cazului clasificării binare (vedeți problema 12), nu există o expresie pentru calculul direct (engl., closed form) al valorii parametrului w_k care corespunde maximului funcției de log-verosimilitate condițională, $L(w_1, \dots, w_K)$. Totuși, putem găsi soluția folosind *metoda gradientului ascendent*, cu ajutorul derivatelor parțiale. Calculați expresia componente de pe poziția k din vectorul gradient $L(w_1, \dots, w_K)$, și anume derivata parțială a lui $L(w_1, \dots, w_K)$ în raport cu w_k .
- c. Dând ponderilor valoarea inițială 0, scrieți *regula de actualizare* (engl., update rule) pentru w_k , folosind η ca rată a învățării (engl., step size). Va converge oare soluția la maximul global?

⁷² Alternativ, se poate considera că extindem „artificial“ vectorul x — și, la fel, fiecare vector x^l din setul de date de antrenament — cu componenta $x_0 = 1$ și, respectiv, $x_0^l = 1$.

⁷³ Funcția $f : \{w_1 \cdot x, \dots, w_K \cdot x\} \rightarrow [0, 1]$ definită prin $f(w_k \cdot x) = \frac{\exp(w_k \cdot x)}{1 + \sum_{l=1}^{K-1} \exp(w_l \cdot x)}$ și numită *funcția softmax*, are proprietatea că menține ordinea, adică $w_k \cdot x < w_j \cdot x \Rightarrow f(w_k \cdot x) < f(w_j \cdot x)$. Întrucât, de asemenea, f este derivabilă în raport cu w_k — ceea ce justifică particula *soft* în denumirea softmax —, această funcție este folosită adeseori ca funcție de activare pentru stratul de ieșire al rețelelor neuronale cu output multiplu. (Vedeți problema 52 de la capitolul *Rețele neuronale artificiale*.) În mecanica statistică funcția softmax este cunoscută sub numele *distribuția Boltzmann* (introdusă de fizicianul austriac Ludwig în anul 1868) sau *distribuția Gibbs*.

⁷⁴ Termenii de *regularizare* și *overfitting* au fost introdusi pentru regresia liniară la problema 3.C. La secțiunea de *Regresie logistică*, am folosit pentru prima dată regularizarea (de normă) L_2 la problema 14.

Răspuns:

a. Presupunem că dispunem de m instanțe de antrenament. Folosind funcția-indicator $1_{\{lk\}}$, care este definită prin $1_{\{lk\}} = 1$ dacă $Y^l = k$ și 0 în caz contrar, putem scrie funcția de verosimilitate condițională astfel:

$$L(w_1, \dots, w_K) = \prod_{l=1}^n \prod_{k=1}^K P(Y^l = k | X^l = x; w)^{1_{\{lk\}}} = \prod_{l=1}^n \prod_{k=1}^K \left(\frac{\exp(w_k \cdot x^l)}{\sum_r \exp(w_r \cdot x^l)} \right)^{1_{\{lk\}}}.$$

Aplicând funcția \ln , vom obține funcția de log-verosimilitate condițională:

$$\ell(w_1, \dots, w_K) = \sum_{l=1}^n \sum_{k=1}^K 1_{\{lk\}} \left((w_k \cdot x^l - \ln \sum_r \exp(w_r \cdot x^l)) \right).$$

Adăugând termenul de regularizare corespunzător normei L_2 , rezultă:⁷⁵

$$\ell(w_1, \dots, w_K) = \sum_{l=1}^n \sum_{k=1}^K 1_{\{lk\}} \left(w_k \cdot x^l - \ln \sum_r (\exp(w_r \cdot x^l)) \right) - \frac{\lambda}{2} \sum_{k=1}^K \|w_k\|^2.$$

b. Calculăm derivata parțială — de fapt, vectorul de derive parțiale — pentru funcția de log-verosimilitate condițională, în raport cu parametrul w_i .⁷⁶

$$\begin{aligned} \frac{\partial}{\partial w_i} \ell(w_1, \dots, w_K) &= \sum_{l=1}^n \left(1_{\{li\}} x^l - \frac{\exp(w_i \cdot x^l) x^l}{\sum_r \exp(w_r \cdot x^l)} \right) - \lambda w_i \\ &= \sum_{l=1}^n \left(1_{\{li\}} - P(Y^l = i | X^l) \right) x^l - \lambda w_i. \end{aligned}$$

c. Regula de actualizare pentru parametrul w_i , folosind gradientul ascendent, este următoarea:

$$w_i \leftarrow w_i + \eta \sum_{l=1}^n \left(1_{\{li\}} - P(Y^l = i | X^l) \right) x^l - \eta \lambda w_i.$$

Convergența la punctul de maxim global are loc întrucât funcția de log-verosimilitate condițională este concavă.⁷⁷

18. (Teorema de reprezentare: un rezultat foarte util pentru kernel-izarea mai multor algoritmi de învățare automată care produc separatori liniari)

prelucrare de Liviu Ciortuz după Stanford, John Duchi, Supplemental lecture notes CMU, 2015 spring, Alex Smola, midterm, pr. 5

Introducere:

⁷⁵Vedeți problema 14.

⁷⁶Vă reamintim că acest tip de derivată se numește *derivată vectorială* (sau, *direcțională*). Pentru expresia $w \cdot x = w^\top x$, derivata în raport cu vectorul w este vectorul x . (Vedeți documentul *Matrix Identities* de Sam Roweis.)

⁷⁷LC: Demonstrația acestei proprietăți extinde în mod natural demonstrația din cazul clasificării binare, care a fost prezentată la problema 13.

La problemele 9 și 16 de la secțiunea Regresia liniară și respectiv de la secțiunea Regresia logistică am arătat cum anume se pot kernel-iza aceste metode, în vederea obținerii unor separatori neliniari. Pentru aceasta, în ambele cazuri a fost esențială o *proprietate* — care a fost argumentată acolo —, și anume că ponderile w , care se combină liniar cu atributele de intrare, se pot scrie ca o combinație liniară de instanțele de antrenament x_i .

La această nouă problemă vom demonstra (în manieră unitară!) că această proprietate este valabilă pentru mai multe metode de învățare automată, care au în comun faptul că minimizează un anumit *cost* (sau, *risc empiric*).

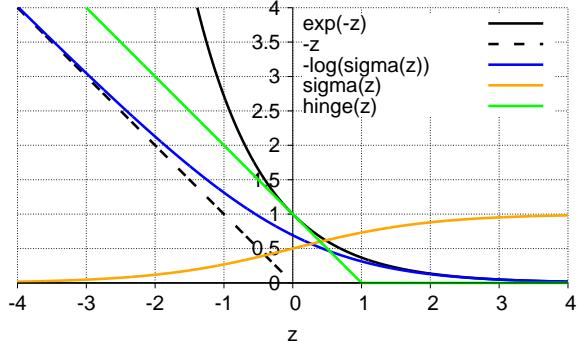
Pentru regresia logistică, echivalența dintre maximizarea funcției de log-verosimilitate condițională și minimizarea costului mediu folosind funcția de cost logistică a fost demonstrată la problema 12.c. Pentru regresia liniară, echivalența dintre maximizarea funcției de log-verosimilitate condițională și minimizarea sumei pătratelor erorilor a fost demonstrată pentru cazul unidimensional la problema 1.a, iar pentru cazul general la problema 3.d.

Fie $L : \mathbb{R}^2 \rightarrow \mathbb{R}$. Vom spune că L este *funcție de cost* sau *funcție de pierdere* (engl., loss function) dacă ea este *i*. nenegativă și *ii*. convexă în raport cu primul argument, pentru orice valoare fixată a celui de-al doilea argument.

Spre exemplu,

- pentru regresia liniară și pentru perceptronul liniar se folosește – cu rol de funcție de cost – pătratul erorii, adică $L(r, y) = (y - r)^2$,
- pentru regresia logistică se folosește *funcția de cost logistică* $L(r, y) = \ln(1 + e^{-yr})$,
- pentru mașini cu vectori-suport (SVM) se folosește *funcția de cost hinge* $L(r, y) = \max\{0, 1 - yr\}$, iar
- pentru AdaBoost, se folosește *funcția de cost [negativ] exponentială* $L(r, y) = e^{-yr}$.

(În figura alăturată, am folosit notația $z = yr$.)



Fie de asemenea instanțele $x_i \in \mathbb{R}^d$ (cu $d \in \mathbb{N}^*$), precum și $y_i \in \mathbb{R}$ (desemnând fie clase, fie — ca în cazul regresiei liniare și al perceptronului liniar — valori reale oarecare pentru funcția de învățat), pentru $i = 1, \dots, n$.

a. Demonstrați că orice soluție w^* a următoarei *probleme de optimizare* (cu termen de regularizare de normă L_2)

$$\min_{w \in \mathbb{R}^d} \left(\frac{1}{n} \sum_{i=1}^n L(w \cdot x_i, y_i) + \frac{\lambda}{2} \|w\|^2 \right), \quad (48)$$

unde $\lambda \geq 0$, iar $\|\cdot\|$ desemnează norma euclidiană, poate fi scrisă sub forma următoare:

$$w^* = \sum_{i=1}^n \alpha_i x_i \text{ pentru anumite valori } \alpha_i \in \mathbb{R}. \quad (49)$$

Sugestie: Puteți rezolva problema în *cazul particular* când $\lambda > 0$, iar funcția de cost $L : \mathbb{R}^2 \rightarrow \mathbb{R}$ este derivabilă în raport cu primul argument, pentru orice valoare fixată a celui de-al doilea argument.

Observație: Acest rezultat se numește *teorema de reprezentare* și se datorează statisticienilor Grace Wahba și George Kimeldorf.⁷⁸

b. Se dă o funcție $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$, unde $m > d$ (de obicei m este chiar mult mai mare decât d). În continuare vom considera „imaginile“ instanțelor x_i prin funcția ϕ , adică $\phi(x_i)$, pentru $i = 1, \dots, n$. Dintre toate operațiile care pot fi (sau, sunt deja) definite între aceste „imagini“, singura care va fi permisă (pentru a fi folosită) aici este produsul scalar: $\phi(x_i) \cdot \phi(x_j) \stackrel{\text{not.}}{=} K(x_i, x_j)$. Funcția K astfel definită — și, de fapt, extinsă în mod natural la întreg spațiul $\mathbb{R}^d \times \mathbb{R}^d$ — se numește *funcție-nucleu*.⁷⁹

Aplicați aşa-numitul “kernel-trick” problemei de optimizare

$$\tilde{w}^* = \arg \min_{\tilde{w} \in \mathbb{R}^m} \left(\underbrace{\frac{1}{n} \sum_{i=1}^n L(\tilde{w} \cdot \phi(x_i), y_i) + \frac{\lambda}{2} \|\tilde{w}\|^2}_{\text{not.}: J(\tilde{w})} \right) \quad (50)$$

adică:

i. Înlocuiți în expresia $J(\tilde{w})$ din relația (50) pe \tilde{w} cu expresia lui

$$\tilde{w}^* = \sum_{j=1}^n \alpha_j \phi(x_j) \in \mathbb{R}^m, \quad (51)$$

existența coeficienților $\alpha_j \in \mathbb{R}$ care satisfac această egalitate fiind garantată de rezultatul de la punctul a. Veți nota această nouă expresie cu $J(\alpha)$, unde $\alpha \stackrel{\text{not.}}{=} (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$.

ii. Transformați *forma* expresiei $J(\alpha)$ pe care ați obținut-o la punctul b.i într-o echivalentă, în care nu mai apare funcția $\phi()$ ci (doar) funcția-nucleu $K(,)$.

iii. Similar, scrieți ecuația *separatorului decizional* $\tilde{w}^* \cdot \phi(x) = 0$ — unde x este o instanță oarecare de test din \mathbb{R}^d —, înlocuind \tilde{w}^* conform relației (51) și, în final, eliminați $\phi()$, punând $K(x_i, x)$ în locul produselor scalare $\phi(x_i) \cdot \phi(x)$.

c. Fie problema de optimizare

$$\alpha^* = \arg \min_{\alpha \in \mathbb{R}^n} J(\alpha),$$

unde expresia $J(\alpha)$ a fost obținută la punctul b.ii. Presupunând că se dorește rezolvarea acestei probleme de optimizare folosind metoda gradientului descendente, deduceți *regula de actualizare* pentru vectorul de coeficienți α .

Sugestie: Pentru a vă facilita deducerea gradientului funcției $J(\alpha)$, vă sugerăm ca în prealabil să rescrieți expresia funcției $J(\alpha)$, pe care ați obținut-o la punctul b.ii, sub formă matricială. Concret, notând cu K_i vectorul-coloană $(K(x_i, x_1), \dots, K(x_n, x_1))^\top$, pentru $i = 1, \dots, n$, și apoi cu K matricea de dimensiune $n \times n$ obținută prin „alăturarea“ vectorilor K_i , adică $K = (K_1, \dots, K_n)$, veți putea demonstra egalitatea

$$J(\alpha) = \frac{1}{n} \sum_{i=1}^n L(K_i^\top \alpha, y_i) + \frac{\lambda}{2} \alpha^\top K \alpha.$$

Apoi, puteți, eventual, să folosiți următoarele formule de calcul cu derivate vectoriale:⁸⁰

⁷⁸Vedeți articolul *A correspondence between Bayesian estimation on stochastic processes and smoothing by splines* din Annals of Mathematical Statistics, 41 (2): 495–502, 1970.

⁷⁹Pentru mai multe informații despre acest subiect, vedeți secțiunea *Funcții-nucleu* de la capitolul de *Fundamente*.

⁸⁰Cf. *Matrix Identities*, Sam Roweis, 1999, <http://www.cs.nyu.edu/~roweis/notes/matrixid.pdf>.

$$(5a) \quad \frac{\partial}{\partial X} a^\top X = \frac{\partial}{\partial X} X^\top a = a$$

$$(5b) \quad \frac{\partial}{\partial X} X^\top AX = (A + A^\top)X$$

Răspuns:

a. Conform Sugestiei din enunț, vom presupune că $\lambda > 0$ și, de asemenea, că *funcția de cost* $L : \mathbb{R}^2 \rightarrow \mathbb{R}$ este derivabilă în raport cu primul argument, pentru orice valoare fixată a celui de-al doilea argument. Pentru convenientă, vom nota costul (sau riscul empiric) mediu cu regularizare L_2 astfel:

$$J(w) = \frac{1}{n} \sum_{i=1}^n L(w \cdot x_i, y_i) + \frac{\lambda}{2} \|w\|^2.$$

De asemenea, în cele ce urmează vom folosi notația $L'(z, y) \stackrel{\text{not.}}{=} \frac{\partial}{\partial z} L(z, y)$. Gradientul funcției $J(w)$ în raport cu vectorul w este

$$\begin{aligned} \nabla J(w) &= \frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial w} L(w \cdot x_i, y_i) + \frac{\lambda}{2} \frac{\partial}{\partial w} \|w\|^2 \\ &= \frac{1}{n} \sum_{i=1}^n L'(w \cdot x_i, y_i) \frac{\partial}{\partial w} (w \cdot x_i) + \frac{\lambda}{2} \frac{\partial}{\partial w} w^2 = \frac{1}{n} \sum_{i=1}^n L'(w \cdot x_i, y_i) x_i + \lambda w. \end{aligned}$$

La a doua egalitate de mai sus am folosit o generalizare a regulei de derivare pentru compunere de funcții, iar la ultima egalitate am folosit regulile de derivare vectorială (5a) și (5b), care au fost menționate la finalul enunțului acestei probleme.

Întrucât în punctul de minim al funcției J gradientul ei se anulează (adică, $\nabla J(w^*) = 0 \in \mathbb{R}^d$),⁸¹ rezultă

$$w^* = -\frac{1}{n\lambda} \underbrace{\sum_{i=1}^n \frac{\partial}{\partial w} L(w^* \cdot x_i, y_i)}_{\in \mathbb{R}} x_i.$$

Considerând $\alpha_i \stackrel{\text{not.}}{=} -\frac{1}{n\lambda} \frac{\partial}{\partial w} L(w^* \cdot x_i, y_i)$, relația precedentă va putea fi rescrisă imediat sub forma egalității (49) din enunț.

b. Înlocuind în expresia funcției $J(\tilde{w})$, care a fost definită în relația (50), argumentul \tilde{w} cu expresia lui \tilde{w}^* din relația (51), vom obține funcția J exprimată de astă dată în raport cu vectorul de coeficienți α :

$$\begin{aligned} J(\alpha) &= \frac{1}{n} \sum_{i=1}^n L\left(\left(\sum_{j=1}^n \alpha_j \phi(x_j)\right) \cdot \phi(x_i), y_i\right) + \frac{\lambda}{2} \left(\sum_{j=1}^n \alpha_j \phi(x_j)\right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n L\left(\sum_{j=1}^n \alpha_j (\phi(x_j) \cdot \phi(x_i)), y_i\right) + \frac{\lambda}{2} \left(\sum_{i=1}^n \alpha_i \phi(x_i)\right) \cdot \left(\sum_{j=1}^n \alpha_j \phi(x_j)\right) \\ &= \frac{1}{n} \sum_{i=1}^n L\left(\sum_{j=1}^n \alpha_j K(x_i, x_j), y_i\right) + \frac{\lambda}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j). \end{aligned} \tag{52}$$

⁸¹Proprietatea pe care am aplicat-o aici generalizează ceea ce știm deja din liceu despre funcțiile reale f de o singură variabilă, x , care sunt derivabile, și anume: $f'(x^*) = 0$, pentru orice punct de optim x^* .

Am ținut cont de faptul că funcția K este simetrică, întrucât produsul scalar al vectorilor este simetric: $K(x_i, x_j) \stackrel{\text{def.}}{=} \phi(x_i) \cdot \phi(x_j) = \phi(x_j) \cdot \phi(x_i) \stackrel{\text{def.}}{=} K(x_j, x_i)$.

Ecuatia separatorului decizional $\tilde{w}^* \cdot \phi(x) = 0$, unde x este o instanță oarecare de test din \mathbb{R}^d , se scrie astfel:

$$\begin{aligned} \tilde{w}^* \cdot \phi(x) = 0 &\Leftrightarrow \left(\sum_{i=1}^n \alpha_i \phi(x_i) \right) \cdot \phi(x) = 0 \Leftrightarrow \sum_{i=1}^n \alpha_i (\phi(x_i) \cdot \phi(x)) = 0 \Leftrightarrow \\ &\sum_{i=1}^n \alpha_i K(x_i, x) = 0. \end{aligned} \quad (53)$$

Observați că expresia separatorului decizional este liniară în spațiul de trăsături \mathbb{R}^m (și anume, $\tilde{w}^* \cdot \phi(x)$), însă în general ea este neliniară în spațiul inițial, \mathbb{R}^d . Pentru exemplificare — și pentru a înțelege mai bine afirmația aceasta —, vă sugerăm să-l înlocuiți în relația (53) pe $K(x_i, x)$ cu expresia nucleului gaussian, $\frac{\|x - x_i\|^2}{2\sigma^2}$, unde $\sigma \in \mathbb{R}_+$ este fixat.⁸²

c. Folosind notațiile matriceale introduse în Sugestia din enunț, se verifică imediat că relația (52) se rescrie într-adevăr sub forma

$$J(\alpha) = \frac{1}{n} \sum_{i=1}^n L(K_i^\top \alpha, y_i) + \frac{\lambda}{2} \alpha^\top K \alpha.$$

Gradientul funcției $J(\alpha)$ se obține ușor folosind regulile de derivare vectorială indicate de asemenea în Sugestia din enunț:

$$\begin{aligned} \nabla_\alpha J(\alpha) &= \frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial \alpha} L(K_i^\top \alpha, y_i) + \frac{\partial}{\partial \alpha} \frac{\lambda}{2} \alpha^\top K \alpha \\ &\stackrel{(5a)}{=} \frac{1}{n} \sum_{i=1}^n L'(K_i^\top \alpha, y_i) \frac{\partial}{\partial \alpha} (K_i^\top \alpha) + \frac{\lambda}{2} (K + K^\top) \alpha \\ &\stackrel{(5b)}{=} \frac{1}{n} \sum_{i=1}^n L'(K_i^\top \alpha, y_i) K_i + \lambda K \alpha. \end{aligned} \quad (54)$$

Am ținut cont de faptul că matricea K este simetrică (deci $K = K^\top$), întrucât funcția-nucleu K este simetrică, după cum am justificat mai sus.

Acum putem scrie regula gradientului descedent, astfel:

$$\alpha \leftarrow \alpha - \eta \left[\frac{1}{n} \sum_{i=1}^n L'(K_i^\top \alpha, y_i) K_i + \lambda K \alpha \right],$$

unde $\eta > 0$ este rata de învățare.

Observație: Dacă ne propunem să obținem regula de actualizare corespunzătoare gradientului descedent stochastic, atunci este util (și chiar necesar!) de observat că produsul $K\alpha$ se poate scrie în mod echivalent astfel:

$$K\alpha = \sum_{i=1}^n \alpha_i K_i.$$

⁸²Această funcție-nucleu mai este cunoscută și sub numele de “Radial Basis Function”, RBF. Vedeți problemele 66 și 67 de la capitolul de Fundamente.

Tinând cont de această relație, precum și de relația (54), rezultă că putem scrie regula gradientului descentent stochastic astfel:⁸³

$$\alpha \leftarrow \alpha - \eta \left[\frac{1}{n} L'(K_i^\top \alpha, y_i) K_i + \lambda \alpha_i K_i \right].$$

Sugestie: Vă recomandăm ca, după ce ați înțeles această problemă, să parcurgeți — pe lângă problemele 9 și 16 de la acest capitol — și problema 19 de la capitolul *Rețele neuronale artificiale*, problema 47 de la capitolul *Clusterizare* și problema 14 de la capitolul *Învățare bazată pe memorare*. Este foarte instructiv să analizați elementele pe care aceste probleme le au în comun cu problema de față. De asemenea, menționăm că această problemă poate constitui o foarte bună „poartă“ de intrare în capitolul de *Mașini cu vectori-suport*.

1.2 Probleme propuse

1.2.1 Regresia liniară

19.

(Regresia liniară unidimensională:
deducerea directă a soluției analitice MLE / LSE)
CMU, 2011 fall, Eric Xing, HW1, pr. 2.1

Considerăm un caz simplu de regresie liniară, în care modelul [parametric] este de forma $y = ax + b + \varepsilon$, unde $x, y, a, b \in \mathbb{R}$, iar ε este un termen „zgomot“ care urmează o distribuție gaussiană de medie zero, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.⁸⁴ Avem n instanțe de antrenament, (x_i, y_i) , $i = 1, 2, \dots, n$, cu $x_i, y_i \in \mathbb{R}$.

a. Scrieți expresia funcției de cost reprezentând suma pătratelor erorilor în acest model. Nu folosiți forma matricială. Funcția aceasta va fi exprimată folosind modelul parametric unidimensional (1-D).

b. Calculați valorile \hat{a} și \hat{b} care minimizează funcția de cost / pierdere și arătați că

$$\begin{aligned}\hat{a} &= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{n(\sum_i x_i^2) - (\sum_i x_i)^2} \\ \hat{b} &= \frac{(\sum_i x_i^2)(\sum_i y_i) - (\sum_i x_i)(\sum_i x_i y_i)}{n(\sum_i x_i^2) - (\sum_i x_i)^2},\end{aligned}$$

⁸³Observație: În documentul *Supplemental lecture notes*, John Duchi recomandă următoarea variantă a acestei reguli:

$$\alpha \leftarrow \alpha - \eta [L'(K_i^\top \alpha, y_i) K_i + n \lambda \alpha_i K_i],$$

cu scopul de a evita ca estimatorul α calculat prin aplicarea metodei gradientului să fie *deplasat* (engl., biased). Se recomandă de asemenea ca valoarea lui λ să fie mică, iar η să se modifice în raport cu iterația. De exemplu, pentru iterația t se poate lua $\eta_t = 1/\sqrt{t}$ sau un multiplu al acestei valori.

⁸⁴Așadar, acest model este ușor mai elaborat decât cel care a fost prezentat la problema 1, dar nu [atât de] general precum cel de la problema 3.A.

bineînțeles, în condițiile în care expresia de la numitorul celor două fracții este nenulă. Ca și mai sus, nu folosiți notația matriceală. Minimizați în manieră directă funcția de cost / pierdere de la punctul a .

- c. Deducreți formula pentru estimarea de verosimilitate maximă (MLE) pentru modelul 1-D. Arătați că estimarea aceasta (de tip MLE) coincide cu estimarea corespunzătoare minimizării criteriului sumei pătratelor erorilor, obținută la punctul a .
- d. Arătați că soluția optimă calculată la punctul b este aceeași cu forma matriceală obținută în cazul modelului [mai] general de regresie liniară,

$$\hat{\beta} = (X^\top X)^{-1} X^\top y,$$

unde, în cazul nostru $\hat{\beta} \stackrel{not.}{=} (\hat{a}, \hat{b})$, $X \in \mathbb{R}^{n \times 2}$, cu linia i din X fiind $(1, x_i)$ pentru $i = 1, \dots, n$, iar $y = (y_1, \dots, y_n)^\top$.⁸⁵

20. (Compararea a două modele de regresie unidimensională de tip LSE)
CMU, 2014 fall, Z. Bar-Joseph, W. Cohen, HW2, pr. 3

Fie X și Y două variabile aleatoare, iar $(x_1, y_1), \dots, (x_n, y_n)$ un set de n exemple de antrenament generate în mod independent.

Considerăm că modelele de regresie de mai jos sunt de tipul sumei celor mai mici pătrate (engl., least squared errors, LSE). Așadar, „zgomotul“ ε urmează o distribuție gaussiană $N(0, \sigma^2)$

- i. $Y = \beta X + \varepsilon$
- ii. $Y = \beta^2 X + \varepsilon$.

- a. Calculați soluțiile acestor două modele de regresie.
- b. Care dintre cele două modele de mai sus produce o eroare la antrenare mai mică?

Sugestie: Răspunsul poate să depindă de setul de date de antrenament. În consecință, explicați în ce situație un anumit model este mai bun decât celălalt.

21. (Regresia liniară: aplicație pentru „învățarea“ parametrului unei funcții neliniare, componenta „zgomot“ fiind modelată cu o distribuție gaussiană)
CMU, 2009 spring, Tom Mitchell, midterm, pr. 6

Se dă un set de date constituit din n instanțe, $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Atât atributul de intrare x cât și atributul de ieșire y au ca valori numere reale. Valoarea atributului de ieșire este generată de o distribuție gaussiană având media $\sin(wx_i)$ și varianță 1:

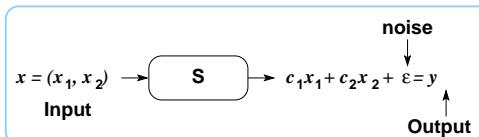
$$p(y_i | x_i, w) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i - \sin(wx_i))^2}{2}}$$

În modelul de mai sus avem un parametru necunoscut w și dorim să deducem / „învățăm“ din aceste date care este estimarea sa în sensul verosimilității maxime.

⁸⁵Vedeți relația (9) de la problema 3.d.

- a. Scrieți expresia verosimilității [condiționale a] datelor ca funcție de parametrul w , date fiind valorile observate x_i și y_i , cu $i = 1, \dots, n$.
- b. Precizați care dintre egalitățile de mai jos este adevărată atunci când se dorește / face estimarea parametrului w în sensul maximizării verosimilității datelor. În cazul în care niciuna dintre relațiile $i-v$ nu este adevărată, marcați opțiunea vi .
- i. $\sum_i \cos^2(wx_i) = \sum_i y_i \sin(x_i)$
 - ii. $\sum_i \cos^2(wx_i) = \sum_i y_i \sin(2wx_i)$
 - iii. $\sum_i \sin(wx_i) \cos(wx_i) = \sum_i y_i \sin(wx_i/2)$
 - iv. $\sum_i x_i \sin(wx_i) \cos(wx_i) = \sum_i x_i y_i \cos(wx_i)$
 - v. $w \cos(x_i) = \sum_i x_i y_i \cos(x_i)$
 - vi. Niciuna dintre variantele de mai sus.
22. (Estimarea parametrilor unui model de regresie liniară bidimensională, în care componenta-zgomot este modelată fie cu o distribuție gaussiană — inclusiv o variantă de regresie local-ponderată —, fie cu o distribuție Laplace)
- CMU, 2009 fall, Carlos Guestrin, HW1, pr. 1.5.2
CMU, 2012 fall, E. Xing, A. Singh, HW1, pr. 2

Figura următoare reprezintă un sistem S care ia ca intrări [valorile] x_1 și x_2 și produce la ieșire o combinație liniară a celor două intrări, $c_1x_1 + c_2x_2$, unde c_1 și c_2 sunt două numere reale necunoscute.



Dispozitivul folosit ca să măsoare outputul sistemului S introduce în plus o „eroare“ ε , care este [asimilată cu] o variabilă aleatoare ce urmează o anumită distribuție. Astfel, outputul observat (y) este dat de ecuația

$$y = c_1x_1 + c_2x_2 + \varepsilon.$$

Presupunem că avem $n > 2$ instanțe (x_{j1}, x_{j2}, y_j) , $j = 1, \dots, n$ sau, echivalent, (x_j, y_j) , $j = 1, \dots, n$, unde $x_j = [x_{j1}, x_{j2}]$. Prin urmare, a avea la dispoziție n măsurători revine la a avea n egalități de forma $y_j = c_1x_{j1} + c_2x_{j2} + \varepsilon_j$, cu $j = 1, \dots, n$. Scopul nostru este să estimăm parametrii c_1 și c_2 din aceste măsurători cu ajutorul criteriului verosimilității maxime, făcând [la fiecare dintre punctele de mai jos] diferențe presupunerii în legătură cu „zgomotul“.

- a. Presupunem că ε_i pentru $i = 1, \dots, n$ sunt variabile aleatoare gaussiene i.i.d. (independente și identic distribuite) cu media 0 și varianță σ^2 . Calculați funcția de log-verosimilitate a datelor. Apoi folosiți această funcție pentru a demonstra că estimarea de verosimilitate maximă $c^* = [c_1^*, c_2^*]$ este soluția unei probleme de aproximare prin așa-numita *metodă a celor mai mici pătrate* (engl., least squares approximation method). Nu este necesar să găsiți efectiv soluția problemei celor mai mici pătrate [pentru acest caz].

b. [Varianta regresiei local-ponderate] Presupunem că ε_i pentru $i = 1, \dots, n$ sunt variabile aleatoare independente de tip gaussian având media 0 și varianța σ_i^2 . Calculați funcția de log-verosimilitate și găsiți perechea de valori $c^* = [c_1^*, c_2^*]$ care o maximizează, adică estimarea de verosimilitate maximă (MLE) a parametrilor c_1 și c_2 .

c. Presupunem că pentru orice $i = 1, \dots, n$, variabila ε_i are funcția de densitate

$$f_{\varepsilon_i}(x) = f(x) = \frac{1}{2\theta} e^{-\frac{|x|}{\theta}}, \text{ cu } \theta > 0.$$

Cu alte cuvinte, zgometul este i.i.d. și urmează o distribuție Laplace având parametrul de locație $\mu = 0$ și parametrul de scalare θ . Calculați funcția de log-verosimilitate a datelor pentru acest model al zgometului și indicați motivul / motivele pentru care acest model conduce la *soluții mai robuste* [la „zgomote“ și outlier-e] decât în cazurile precedente.⁸⁶

23.

(Regresia liniară bidimensională: exemplu de aplicare a metodei gradientului, varianta stochastică)

CMU, 2014 spring, Seyoung Kim, HW2, pr. 1.2.B

În această problemă vom presupune că ori de câte ori se înmulțește o instanță [reprezentată printr-un vector] x cu un vector de ponderi, x a fost deja augmentat [în aşa fel încât să înceapă] cu o componentă 1. De exemplu, x poate fi $(1, x_1)$ sau $(1, x_1, x_2)$, iar constanta 1 va fi înmulțită cu ponderea β_0 .

Considerăm că folosim metoda gradientului pentru a antrena un model de regresie liniară pentru a „învăță“ funcția $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, ale cărei argumente (sau attribute) sunt notate cu x_1 și respectiv x_2 . Facem inițializarea vectorului de ponderi astfel: $\beta = (0.5, 0.5, 0.5)$. Primul exemplu considerat este $x = (2, 6)$ și, corespunzător, $y = 8.5$.

- a. Folosind aceste valori inițiale pentru ponderi, cât este eroarea $y - f(x) = y - x \cdot \beta$ produsă de modelul de regresie?
- b. Cât va fi β_1^{new} , noua valoare a ponderii β_1 (care corespunde atributului / trăsăturii x_1) la finalul primei iterații executate de algoritm gradientului descendente, varianta stochastică,⁸⁷ folosind exemplul indicat mai sus? Veți utiliza rata de învățare (engl., learning rate) $\eta = 0.05$.

În cele ce urmează vom presupune că am terminat de antrenat modelul de regresie liniară și că am obținut vectorul de ponderi $\hat{\beta} = (0.5, 1, 1)$. Acum vom considera instanța de test $x_q = (3, 4)$ și vom prezice $f(x_q)$. Presupunem că în prealabil am estimat în mod empiric varianța componentei-zgomot (ε) din modelul de regresie și am obținut $\sigma = 1$.

- c. Cât este valoarea „așteptată“ (engl., expected value) pentru $f(x_q)$?
 - d. Cât este probabilitatea ca valoarea adevărată pentru $f(x_q)$ să fie în intervalul $[9, 10]$?
- Sugestie:* Puteți folosi tabela de valori ale funcției cumulative de distribuție (Φ) pentru distribuția gaussiană standard care apare la pr. 30 de la capitolul de *Fundamente*.

⁸⁶Pentru reprezentarea grafică a mai multor distribuții Laplace, vedeti problema 8.B.

⁸⁷Pentru prezentarea metodei gradientului aplicată la rezolvarea regresiei liniare, vedeti pr. 6.

24. (Regresia liniară, varianta LSE: o proprietate interesantă: adăugarea de noi trăsături / atribute nu mărește suma pătratelor erorilor)
Stanford, 2014 fall, Andrew Ng, midterm, pr. 1

După cum am arătat la problema 3.A, la regresia liniară în varianta LSE (în care modelarea „zgomotului“ se face folosind distribuția gaussiană) se lucrează cu o funcție de cost de tipul sumei celor mai mici pătrate (engl., Least Squared Errors).⁸⁸

$$J(\beta) = \sum_{i=1}^n (\beta^\top x^{(i)} - y^{(i)})^2 = (X\beta - y)^\top (X\beta - y).$$

Obiectivul la aplicarea acestei metode de regresie este să găsim acea valoare a vectorului de parametri β pentru care se atinge minimul expresiei $J(\beta)$, pentru respectivul set de date de antrenament.

Presupunem că în acest set de date avem inițial d trăsături și, în consecință, inputul la antrenare este dat sub forma lui $X \in \mathbb{R}^{n \times (d+1)}$, aşa-numita *matrice de design*.

Să zicem că acum avem acces la încă o trăsătură pentru fiecare exemplu de antrenament. Așadar, avem un vector adițional de trăsături $v \in \mathbb{R}^{n \times 1}$ pentru setul nostru de antrenament și dorim să-l folosim în problema noastră de regresie. Putem face acest lucru creând o nouă matrice de design: $\tilde{X} \stackrel{\text{not.}}{=} [X \ v] \in \mathbb{R}^{n \times (d+2)}$. Prin urmare, noul nostru vector de parametri este $\beta_{new} \stackrel{\text{not.}}{=} [\beta, p]^\top$, unde $p \in \mathbb{R}$ este un parametru care corespunde noului vector de trăsături v .

Observație importantă: Pentru a asigura o anumită simplitate pentru demonstrația matematică, în cele ce urmează vom presupune că $X^\top X = I \in \mathbb{R}^{(d+1) \times (d+1)}$ și $\tilde{X}^\top \tilde{X} = I \in \mathbb{R}^{(d+2) \times (d+2)}$, $v^\top v = 1$. Aceasta este aşa-numita presupozitie de *ortonormalitate*. Altfel spus, coloanele matricei X (respectiv \tilde{X}) sunt ortonormale. Concluziile demonstrației din această problemă se mențin chiar și fără această presupozitie de ortonormalitate, însă cu ea demonstrația va fi mai ușoară.

- a. Fie $\hat{\beta} \stackrel{\text{not.}}{=} \arg \min_{\beta} J(\beta)$ valoarea vectorului de parametri β care minimizează funcția obiectiv originală (cea care folosește matricea de design X). Folosind presupozitia de ortonormalitate, demonstrați că

$$J(\hat{\beta}) = (XX^\top y - y)^\top (XX^\top y - y),$$

adică, arătați că aceasta este valoarea lui $\min_{\beta} J(\beta)$ (valoarea funcției obiectiv în punctul de minim).

- b. Vom nota cu $\tilde{\beta}_{new}$ valoarea vectorului de parametri β pentru care se atinge minimul funcției

$$\tilde{J}(\beta_{new}) \stackrel{\text{not.}}{=} (\tilde{X}\beta_{new} - y)^\top (\tilde{X}\beta_{new} - y).$$

Determinați $\tilde{J}(\tilde{\beta}_{new})$, valoarea minimă a noii funcții obiectiv, și scrieți această expresie sub forma

$$\tilde{J}(\beta_{new}) = J(\hat{\beta}) + f(X, v, y),$$

unde termenul $J(\hat{\beta})$ a fost obținut la punctul a , iar f este o anumită funcție de argumentele X , v și y .

⁸⁸Corespondența dintre notațiile de aici și cele din problema 3.A este următoarea: $x^{(i)} \rightarrow X'_i$ și $X \rightarrow X'$.

c. Demonstrați că valoarea optimă a funcției obiectiv nu se mărește dacă adăugăm o nouă trăsătură la matricea de design. Altfel spus, arătați că

$$\tilde{J}(\hat{\beta}_{new}) \leq J(\hat{\beta}).$$

d. Arată oare rezultatul de mai sus că adăugând [din ce în ce] mai multe trăsături vom obține întotdeauna (sau, cu necesitate) un model de regresie care generalizează mai bine decât un model cu mai puține trăsături? Explicați de ce *da*, ori, dimpotrivă de ce *nu*.

25.

(Regresia liniară cu regularizare L_2 (*ridge*) și respectiv regularizare L_1 (Lasso): efectul de *diminuare a ponderilor* (engl., weight decay))
prelucrare de Liviu Ciortuz, după CMU, 2011 fall, Eric Xing, HW1, pr. 2.2

În regresia liniară, ni se dă un set de date de antrenament de forma $D = \{(x_i, y_i) | i = 1, \dots, n\}$, cu $x_i = (x_{i,1}, \dots, x_{i,d})^\top$ și $y_i \in \mathbb{R}$. Fie matricea de design $X \in \mathbb{R}^{n \times d}$, unde linia i este x_i^\top și, de asemenea, vectorul $y = (y_1, \dots, y_n)^\top$.

Considerând că lucrăm cu un model parametrizat (engl., parametric model) de forma $y_i = x_i^\top \beta + \varepsilon_i$, unde $\beta = (\beta_1, \dots, \beta_d)^\top$, iar ε_i este un termen „zgomot“ (engl., noise term) care urmează o [anumită] distribuție dată, regresia liniară caută să găsească un vector de parametri β care furnizează cel mai bun “fit” pentru modelul de regresie de mai sus. Un exemplu (de *criteriu*) pentru a măsura fitness-ul, este găsirea aceluia β care minimizează o anumită *funcție de cost / pierdere* (engl., loss function) $J(\beta)$, dată.

a. Pentru regresia liniară cu termen de regularizare de normă L_1 (numită și regresie Lasso), minimizăm următoarea funcție de cost / pierdere:⁸⁹

$$J_L(\beta) = \sum_i (x_i^\top \beta - y_i)^2 + \lambda \sum_{j=1}^d |\beta_j| = \underbrace{(X\beta - y)^\top (X\beta - y)}_{\|X\beta - y\|^2} + \lambda \|\beta\|_1.$$

Presupunem că $X^\top X = I$ (este aşa-numita proprietate de *ortonormalitate* a matricei X). Calculați valoarea parametrului β care minimizează criteriul J_L . (Veți nota această valoare cu $\hat{\beta}_L$).

Indicație (1): Vă readucem aminte următoarea regulă de derivare:

$$\frac{\partial |\beta_a|}{\partial \beta_a} = \begin{cases} 1 & \text{dacă } \beta_a > 0 \\ -1 & \text{dacă } \beta_a < 0 \\ \text{nedefinit} & \text{dacă } \beta_a = 0. \end{cases}$$

Indicație (2): Pentru λ situat într-un anumit interval, nu există soluție analitică (ecuații „normale“) pentru această problemă de optimizare. Ce înseamnă acest fapt?

⁸⁹Atenție! A nu se confunda acest tip de regresie regularizată cu varianta de regresie liniară în care zgomotul ε este modelat cu distribuția Laplace. Pentru regresia aceasta din urmă, criteriul de optimizat este — conform problemei 8.B — o sumă a erorilor în modul / norma L_1 (vezi relația (22)). În problema de față, doar termenul de regularizare este în norma L_1 (este vorba despre $\lambda \|\beta\|_1$), celălalt termen din criteriul J_L este o sumă de pătrate (deci este în norma L_2).

b. *Remember:* La problema 3.A am arătat că atunci când luăm ca funcție de cost suma pătratelor erorilor (engl., the square-error), adică

$$J(\beta) = \sum_i (x_i^\top \beta - y_i)^2 = (X^\top \beta - y)^\top (X^\top \beta - y),$$

obținem (atunci când matricea $X^\top X$ este inversabilă) următoarea soluție pentru problema de optimizare $\arg \min_{\beta} J(\beta)$:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y.$$

În cazul regresiei liniare cu termen de regularizare de normă L_2 (regresia *ridge*), minimizăm următoarea funcție de cost / pierdere:

$$J_R(\beta) = \sum_i (x_i^\top \beta - y_i)^2 + \lambda \sum_{j=1}^d \beta_j^2 = (X\beta - y)^\top (X\beta - y) + \lambda \|\beta\|^2.$$

La problema 3.C am arătat că valoarea parametrului β care minimizează criteriul J_R este următoarea (atunci când matricea $X^\top X + \lambda I$ este inversabilă):

$$\hat{\beta}_R = (X^\top X + \lambda I)^{-1} X^\top y.$$

Presupunem că $X^\top X = I$. Scrieți valorile parametrului β care minimizează criteriile J și J_R . Comparați și explicați cum anume cele două metode de regularizare, *ridge* și Lasso, afectează (eventual, diminuează) valorile parametrilor $\hat{\beta}_j$, pentru $j \in \{1, \dots, d\}$. (Am folosit notația $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_d)^\top$.)

26.

(Regresia liniară cu zgromot gaussian și
cu regularizare L_1 (regresia *Lasso*):
rezolvare cu metoda “coordinate descent”)

Stanford, 2007 fall, Andrew Ng, HW3, pr. 3.a

La problema 3.C am prezentat cazul regresiei *ridge*, adică regresia liniară cu „zgromot“ gaussian la care funcția obiectiv este augmentată cu termenul de *regularizare* $\lambda \|\beta\|^2$. De astă dată, vom analiza cazul în care regularizarea se face folosind norma L_1 (numit și *regresia Lasso*). Așadar, acum dorim să minimizăm funcția obiectiv

$$J(\beta) = \frac{1}{2} \sum_{i=1}^n (\beta^\top x^{(i)} - y^{(i)})^2 + \lambda \|\beta\|_1,$$

unde $\|\beta\|_1$ este definit ca fiind $\sum_{i=1}^d |\beta_i|$.

Regularizarea de normă L_1 este foarte utilă pentru că, aşa cum vom vedea, ea oferă avantajul creării de soluții „rare“ (engl., sparse), ceea ce înseamnă că multe dintre componentele vectorului rezultat β vor fi egale cu 0.

Comentariu: Acest tip de regresie este mai dificil de rezolvat decât regresia neregularizată ori cea cu regularizare de normă L_2 , pentru că termenul L_1 nu este derivabil. Totuși, în acest scop au fost dezvoltăți mai mulți algoritmi eficienți, care lucrează foarte bine în practică. Între aceștia, o abordare foarte directă este reprezentată de metoda *descreșterii pe coordonate* (engl., *coordinate descent*).

În această problemă veți concepe un algoritm de tipul descreșterii pe coordonate pentru regresia liniară de tip LSE cu regularizare L_1 . Mai specific, întrucât metoda descreșterii pe coordonate este iterativă — ca de altfel și metoda gradientului, care se aplică în cazul regresiei *ridge*, aşa cum ati văzut la problema 6 — veți deduce *regula de actualizare* pentru o pondere oarecare β_i , cu $i \in \{1, \dots, d\}$.

Considerând matricea de design X și vectorul y aşa cum au fost definite la problemele 3 și 6, precum și vectorul de parametri β , cum am putea să „ajustăm“ valoarea lui β_i astfel încât să minimizăm funcția obiectiv?

Pentru a răspunde la această întrebare, vom scrie funcția obiectiv de mai sus astfel:

$$\begin{aligned} J(\beta) &= \frac{1}{2} \|X\beta - \vec{y}\|^2 + \lambda \|\beta\|_1 \\ &= \frac{1}{2} \|X\bar{\beta} + X_i\beta_i - \vec{y}\|^2 + \lambda \|\bar{\beta}\|_1 + \lambda |\beta_i|, \end{aligned}$$

unde $X_i \in \mathbb{R}^n$ reprezintă coloana i a matricei X , iar vectorul $\bar{\beta}$ este identic cu vectorul β , doar că $\bar{\beta}_i = 0$. Tot ce am făcut aici a fost să rescriem expresia precedentă, în aşa fel încât termenul β_i să apară în mod explicit în funcția obiectiv.

Totuși, ultima expresie încă îl are în componență sa pe $|\beta_i|$, care este nederivabil, deci este dificil de optimizat. Pentru a depăși acest impediment, *remarcăm* faptul că semnul parametrului β_i este fie pozitiv fie negativ (mai precis, fie negativ fie nenegativ). Dacă am ști semnul lui β_i , atunci $|\beta_i|$ ar deveni doar un termen liniar. Așadar, putem scrie funcția obiectiv astfel:

$$J(\beta) = \frac{1}{2} \|X\bar{\beta} + X_i\beta_i - \vec{y}\|^2 + \lambda \|\bar{\beta}\|_1 + \lambda s_i \beta_i, \quad (55)$$

unde prin s_i am notat semnul lui β_i , deci $s_i \in \{-1, 1\}$.

Pentru a actualiza valoarea parametrului β_i , putem să calculăm valoarea sa optimă pentru fiecare dintre cele două valori posibile ale lui s_i în parte — asigurându-ne apoi că restricționăm aceste valori optime ale lui β_i astfel încât ele să satisfacă restricția de semn pe care am fixat-o pentru cazul respectiv —, iar apoi să vedem care dintre cele două soluții corespunde valorii celei mai bune pentru funcția obiectiv la iterată respectivă.

Vă cerem ca pentru fiecare dintre valorile posibile pentru semnul s_i , să calculați valorile optime corespunzătoare pentru parametrul β_i .

Sugestie: În acest scop, puteți fixa valoarea lui s_i în expresia (55), pe care apoi urmează să o derivați în raport cu β_i pentru a-i determina valoarea optimă. În final, impuneți restricții asupra valorii optime obținute pentru β_i , astfel încât ea să aparțină domeniului corespunzător. (De exemplu, pentru $s_i = 1$, trebuie să impuneți asupra lui β_i restricția $\beta_i \geq 0$.)

27.

(Regresia liniară cu „zgomot“ modelat cu distribuția Laplace:
rezolvare în cazul unidimensional [chiar particularizat]
cu ajutorul derivelei, acolo unde există)
CMU, 2009 fall, Carlos Guestrin, HW2, pr. 1

Presupunem că dorim să prezicem o valoare necunoscută Y , după ce ne-au fost puse la dispoziție [ca date de antrenament] o secvență de „observații“ x_1, \dots, x_n pentru Y , care sunt afectate de prezența unor „zgomote“ i.i.d., mai precis $y_i = x_i + \epsilon_i$, pentru $i = 1, \dots, n$.

Comentariu: Se poate arăta⁹⁰ că atunci când presupunem că „zgomotele“ sunt independente și distribuite conform aceleiași distribuții gaussiene ($\epsilon_i \sim \mathcal{N}(0, \sigma^2)$), a găsi estimarea în sens MLE pentru Y echivalează cu a găsi acea valoare \hat{y} care minimizează suma pătratelor erorilor [adică, a diferențelor] în raport cu aceste valori date, x_i . Altfel spus,

$$\hat{y} = \arg \min_y \sum_{i=1}^n (y - x_i)^2,$$

iar această valoare se calculează cu o formulă analitică (engl., closed form solution) simplă:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^n x_i.$$

În acest exercițiu veți demonstra că atunci când presupunem că „zgomotele“ ε_i sunt independente și distribuite conform distribuției $Laplace(0, b)$, al cărei p.d.f., vă readucem aminte, este

$$f_{\varepsilon_i}(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right),$$

obținem o estimare mai robustă a soluției decât în cazul „zgomotului“ gaussian (vedeți problema 3.A).

La problema 8.B am demonstrat că a găsi estimarea în sensul MLE pentru Y , atunci când presupunem că „zgomotul“ este de tip Laplace, este echivalent cu a găsi valoarea \hat{y} care minimizează suma valoarei absolute (sau, în modul). Altfel spus,

$$L(y) = \sum_{i=1}^n |y - x_i| \text{ și } \hat{y} = \arg \min_y L(y).$$

O modalitate standard de a calcula minimul funcției de cost / pierdere este să calculăm derivata ei și să o egalăm cu 0. Problema este că funcția $L(y)$ nu este derivabilă pe tot domeniul său de definiție. Totuși, este ușor de observat că $L(y)$ este nederivabilă doar atunci când y ia aceeași valoare ca una dintre instanțele x_i .

a. Presupunem că instanțele x_i sunt diferite între ele și că sunt sortate în ordine crescătoare ($x_i < x_j$ pentru orice $i < j$).

Calculați $L'(y)$, derivata lui $L(y)$ în raport cu y , pentru cazul când y este situat între două valori consecutive ale lui x (adică, $x_i < y < x_{i+1}$).

Sugestie: Analizați situația x -ilor care sunt mai mici decât y separat de cea a x -ilor care sunt mai mari decât y .

b. Presupunând că n , numărul instanțelor de antrenament este par, care sunt valorile lui y pentru care $L'(y) = 0$?

c. Dacă n este impar, [se poate arăta că] nu există nicio valoare a lui y astfel încât $L'(y) = 0$. Totuși, există o valoare y_0 astfel încât $L'(y) < 0$ pentru orice $y < y_0$ și $L'(y) > 0$ pentru orice $y > y_0$. Cât este acest y_0 ?

d. Valoarea lui \hat{y} este determinată de răspunsurile pe care le-ați obținut la punctele b și c . Dați pe scurt o explicație pentru faptul că această soluție poate fi mai robustă la prezența outlier-elor din date în comparație cu soluția obținută în cazul regresiei de tip LSE (engl., least squared errors).

⁹⁰Observați că acesta este un caz particular în raport cu regresia liniară de tip LSE prezentat la problema 3.A.

28. (Regresia liniară local-ponderată, cazul unidimensional:
o proprietate: „netezirea“ liniară)
CMU, 2014 spring, B. Poczos, A. Singh, HW2, pr. 4.A

Presupunem că lucrăm cu un model de regresie liniară unidimensională local-ponderată, în care „răspunsul“ prezis pentru o valoare oarecare a variabilei $x \in \mathbb{R}$ este $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$, unde parametrii $\hat{\beta}_0$ și $\hat{\beta}_1$ sunt definiți printr-o relație de forma următoare:

$$\hat{\beta}_0, \hat{\beta}_1 = \arg \min_{\beta_0, \beta_1} \left(\sum_{i=1}^n w_i(x) (y_i - \beta_0 - \beta_1 x_i)^2 \right),$$

cu ponderile $w_i(x) \in \mathbb{R}$.⁹¹

a. Arătați că funcția obiectiv din definiția dată mai sus pentru ponderile $\hat{\beta}_0, \hat{\beta}_1$ poate fi rescrisă astfel:⁹²

$$(y - X\beta)^\top W(x) (y - X\beta),$$

unde $y \stackrel{\text{not.}}{=} (y_1, \dots, y_n)^\top$, $\beta \stackrel{\text{not.}}{=} (\beta_0, \beta_1)^\top$, $W(x)$ este o matrice diagonală, care are diagonala $(w_1(x), \dots, w_n(x))$, iar

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

b. Arătați că funcția $\hat{f}(x)$, care a fost definită mai sus prin relația $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$, este o combinație liniară de valorile $\{y_i\}_{i=1}^n$.⁹³ Mai precis, valoarea $\hat{f}(x)$ poate fi scrisă sub forma $\hat{f}(x) = \sum_{i=1}^n \ell_i(x) y_i = \ell(x)^\top y$, unde $\ell_i(x)$ desemnează o anumită cantitate / funcție

⁹¹Deși nu este neapărat necesar [pentru a demonstra rezultatele care fac obiectul acestui exercițiul], am putea considera că pentru fiecare $i \in \{1, \dots, n\}$, ponderea $w_i(x)$ are valoarea

$\frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)}$, unde funcția K poate fi, de exemplu, aşa-numitul *nucleu RBF*. Această func-

ție este definită pe \mathbb{R}^d prin relația $K\left(\frac{x}{h}\right) \stackrel{\text{def.}}{=} \exp\left(-\frac{\|x\|^2}{h^2}\right)$, unde $h > 0$ este [pentru problema de regresie] un hiper-parametru, numit *lățimea de bandă* (engl., *bandwidth*). De obicei, într-un astfel de caz, regresia liniară local-ponderată este numită regresie [locală] kernel-izată (engl., *kernelized [local] regression*).

Observație importantă: Vă rugăm să rețineți că termenul *kernel* (sau, *kernel-izat*) în acest context nu are nimic de a face cu metoda de *kernel-izare* (engl., *kernel trick*) care este folosită pentru mașinile cu vectori-suport (engl., *Support Vector Machines*, *SVMs*), regresia *ridge kernel-izată* (vedeți problema 9) și regresia logistică *kernel-izată* (problema 16).

⁹²Remarcați faptul că expresia care urmează este, ca formă, identică cu relația (19) din cazul regresiei liniare (în varianta LSE) local-ponderate cu n atrbute / variabile de intrare. Aici vă cerem însă să lucrați în cazul (particular) unidimensional.

⁹³Pentru cazul regresiei liniare neponderate în varianta LSE (care a fost prezentată la problema 3.A), această proprietate decurge imediat din relația (9). Pentru cazul regresiei liniare (în varianta LSE) local-ponderate în general (deci nu doar pentru cazul unidimensional despre care discutăm aici), proprietatea decurge imediat din relația (21). Totuși, ca și mai sus, în problema de față vă cerem să elaborați demonstrația pentru cazul (particular) al regresiei local-ponderate unidimensionale.

definită în raport cu x , iar $\ell(x) \stackrel{\text{not.}}{=} (\ell_1(x), \ell_2(x), \dots, \ell_n(x))^\top$. Cu alte cuvinte, regresia liniară [unidimensională] local-ponderată manifestă proprietatea de *netezire liniară* (cf. CMU, 2007 fall, Carlos Guestrin, HW3, pr. 1).

29. (O extensie / generalizare a regresiei liniare, varianta LSE:
cazul „răspunsului“ multivaluat)

Stanford, 2007 fall, Andrew Ng, HW1, pr. 3

La toate problemele de până acum, în ce privește regresia liniară, am considerat că variabila de „răspuns“ y are o valoare reală. Acum vom presupune că ni se cere să facem antrenarea pe un set de date având răspunsuri / outputuri multiple pentru toate exemplele de antrenament:

$$\{(x^{(i)}, y^{(i)}) | i = 1, \dots, n\}, x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}^p.$$

Așadar, pentru oricare exemplu de antrenament, $y^{(i)}$ este un vector cu p componente.

Vrem să folosim un *model liniar* pentru a prezice outputurile, asemănător celui definit prin estimarea în sensul MLE, adică prin minimizarea sumei pătratelor erorilor (engl., least squared errors, LSE),⁹⁴ și anume specificând acum (în locul vectorului d dimensional) o matrice de parametri $\beta \in \mathbb{R}^{n \times p}$ în relația de definiție a modelului de regresie:

$$y = \beta^\top x.$$

- a. Funcția de cost / pierdere în acest caz este

$$J(\beta) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^p ((\beta^\top x^{(i)})_j - y_j^{(i)})^2.$$

Similar cu modul în care am procedat în cazul univaluat,⁹⁵ scrieți $J(\beta)$ în notație vectorial-matriceală.

Sugestie: Începeți considerând așa-numita *matrice de design* (X), de dimensiune $n \times d$

$$X = \begin{bmatrix} & (x^{(1)})^\top & \\ & (x^{(2)})^\top & \\ & \vdots & \\ & (x^{(n)})^\top & \end{bmatrix}$$

și matricea de „răspunsuri“ (Y), de dimensiune $n \times p$

$$Y = \begin{bmatrix} & (y^{(1)})^\top & \\ & (y^{(2)})^\top & \\ & \vdots & \\ & (y^{(n)})^\top & \end{bmatrix},$$

iar apoi căutați să vedeați cum poate fi exprimat $J(\beta)$ cu ajutorul acestor matrice.

⁹⁴Vedeți cum am procedat la problema 3.A.

⁹⁵Adică, atunci când $y \in \mathbb{R}$; vedeți relațiile (7) și (8) de la problema 3.A.

b. Găsiți *formula analitică* (engl., the closed form solution) pentru acea valoare a lui β care minimizează expresia $J(\beta)$.⁹⁶ Altfel spus, aceasta înseamnă să obțineți aşa-numitele ecuații „normale” pentru cazul multivaluat al regresiei liniare în varianta LSE.

c. Să presupunem că în loc să considerăm vectorii $y^{(i)}$ ca atare, decidem să calculăm fiecare variabilă / componentă $y_j^{(i)}$ în mod separat, pentru $j = 1, \dots, p$. În această abordare, vom avea p modele liniare individuale, de forma

$$y_j = \beta_j^\top x^{(i)}, \text{ cu } \beta_j \in \mathbb{R}^d \text{ pentru } j = 1, \dots, p.$$

Care este relația dintre soluțiile (adică valorile optime pentru parametrii β_j) acestor p probleme independente de tip ‐least squared errors‐ (LSE) și soluția (adică valoarea optimă pentru matricea de parametri β) pentru varianta regresiei liniare LSE multivaluate?

30. (Regresie liniară folosită pentru clasificare; aplicare pe date din \mathbb{R}^2)
prelucrare de Liviu Ciortuz, după MIT, 2011 fall, Leslie Käbling, HW1, pr. 2.2

În acest exercițiu veți lucra cu un algoritm de regresie liniară de tipul sumei celor mai mici pătrate (engl., least squared errors, LSE), folosind ca etichete / ieșiri y doar valorile ± 1 . În acest fel, reducem problema de *clasificare* la una de *regresie*.⁹⁷

Concepți un set de exemple de antrenament în planul euclidian (\mathbb{R}^2), care să fie separabile printr-o dreaptă care trece prin originea sistemului de coordonate, astfel încât, atunci când aceste exemple sunt folosite pentru antrenarea unui model de regresie liniară de tip LSE cu etichete binare, să rezulte [în urma antrenării] că unele dintre exemplele de antrenament vor fi clasificate în mod eronat.

1.2.2 Regresia logistică

31. (Regresia logistică: particularizare în \mathbb{R}^2 ; deducerea regulilor de actualizare, folosind metoda gradientului; regularizare L_1)
CMU, 2018 spring, Nina Balcan, HW3, pr. 1, 4

În această problemă, vom elabora algoritmul de regresie logistică bazat pe metoda gradientului, în cazul bidimensional. Considerăm setul de date de antrenament $\{(x_i, y_i), i = 1, \dots, n\}$ în care fiecare $x_i \in \mathbb{R}^2$ este un vector de trăsături, iar $y_i \in \{0, 1\}$ este o etichetă binară. Presupunând că folosim un model parametrizat de forma

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w_0 - w_1 x_1 - w_2 x_2)} = \frac{\exp(w_0 + w_1 x_1 + w_2 x_2)}{1 + \exp(w_0 + w_1 x_1 + w_2 x_2)},$$

⁹⁶Această formulă va constitui o generalizare a relației (9) de la regresia liniară LSE univaluată (problemă 3.A).

⁹⁷Această metodă este descrisă în secțiunea 4.1.3 din cartea *Pattern Recognition and Machine Learning* de Ch. Bishop (Springer, 2006) și în secțiunea 4.2 din cartea *The Elements of Statistical Learning* de T. Hastie, R. Tibshirani and J. Friedman (Springer, 2009).

obiectivul nostru este să găsim valorile \hat{w}_i ale parametrilor w_i care maximizează verosimilitatea condițională (M(C)LE) a setului de date de antrenament.

a. Mai jos, vom face *noi* calculul pentru log-verosimilitatea condițională [a datelor de antrenament]. Relativ la acest calcul, *dumneavoastră* veți elabora căte o scurtă justificare pentru fiecare linie din demonstrație, explicând cum anume decurge ea din linia precedentă.

$$\ell(w) = \ln \prod_{j=1}^n p(y^j|x^j, w) \quad (56)$$

$$= \sum_{j=1}^n \ln p(y^j|x^j, w) \quad (57)$$

$$= \sum_{j=1}^n \ln (p(y^j = 1|x^j, w)^{y^j} p(y^j = 0|x^j, w)^{1-y^j}) \quad (58)$$

$$= \sum_{j=1}^n [y^j \ln p(y^j = 1|x^j, w) + (1 - y^j) \ln p(y^j = 0|x^j, w)] \quad (59)$$

$$= \sum_{j=1}^n \left[y^j \ln \frac{\exp(w_0 + w_1 x_1^j + w_2 x_2^j)}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} + (1 - y^j) \ln \frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \right] \quad (60)$$

$$= \sum_{j=1}^n \left[y^j \ln (\exp(w_0 + w_1 x_1^j + w_2 x_2^j)) + \ln \frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \right] \quad (61)$$

$$= \sum_{j=1}^n [y^j (w_0 + w_1 x_1^j + w_2 x_2^j) - \ln (1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j))]. \quad (62)$$

b. Acum vom calcula gradientul expresiei de mai sus în raport cu w_0, w_1, w_2 , adică $\frac{\partial \ell(w)}{\partial w_i}$, unde $\ell(w)$ este funcția de log-verosimilitate condițională de la punctul a. Vom elabora *noi* mai mulți pași ai demonstrației, iar apoi vă vom cere să elaborați *dumneavoastră* pasul final.

Calculând derivatele expresiei (62) în raport cu w_i pentru $i \in \{1, 2\}$, vom obține:⁹⁸

$$\begin{aligned} \frac{\partial \ell(w)}{\partial w_i} &= \\ \frac{\partial}{\partial w_i} \sum_{j=1}^n &[y^j (w_0 + w_1 x_1^j + w_2 x_2^j)] - \frac{\partial}{\partial w_i} \sum_{j=1}^n \ln [1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)]. \end{aligned} \quad (63)$$

Prima parte a expresiei de mai sus este liniară în raport cu w_i , aşadar după derivare ea ne va da $\sum_{j=1}^n y^j x_i^j$. Pentru cea de-a doua jumătate a aceleiasi expresii, vom folosi regula de derivare pentru compuneri de funcții (engl., chain rule), după cum urmează (vom considera mai întâi un singur $j \in \{1, \dots, n\}$):

⁹⁸Pentru derivarea în raport cu w_0 , se poate considera că se introduce în mod artificial componenta $x_0^j = 1$ în fiecare instanță de antrenament x^j . Cu aceasta, rezultatele care urmează la acest punct se extind în mod natural și la w_0 .

$$\begin{aligned}
& \frac{\partial}{\partial w_i} \ln(1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)) \\
&= \frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \cdot \frac{\partial}{\partial w_i} (1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)) \\
&= \frac{1}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)} \cdot \exp(w_0 + w_1 x_1^j + w_2 x_2^j) \cdot \frac{\partial}{\partial w_i} (w_0 + w_1 x_1^j + w_2 x_2^j) \\
&= x_i^j \cdot \frac{\exp(w_0 + w_1 x_1^j + w_2 x_2^j)}{1 + \exp(w_0 + w_1 x_1^j + w_2 x_2^j)}
\end{aligned} \tag{64}$$

Folosiți expresia (64) (și discuția de mai sus) pentru a arăta că în final expresia (63) devine

$$\frac{\partial \ell(w)}{\partial w_i} = \sum_{j=1}^n x_i^j (y^j - p(y^j = 1 | x^j; w)). \tag{65}$$

Sugestie: Fracția din expresia (64) corespunde unei probabilități cu care sunteți deja familiar.

Comentariu: Întrucât funcția de log-verosimilitate condițională ℓ este concavă,⁹⁹ este ușor să determinăm maximul ei folosind metoda gradientului ascendent. În final, *algoritmul* va fi după cum urmează. Vom alege mai întâi rata de învățare $\eta > 0$, iar apoi vom executa un număr de iterații de forma

$$\begin{aligned}
w_0^{(t+1)} &= w_0^{(t)} + \eta \sum_j [y^j - p(y^j = 1 | x^j; w^{(t)})] \\
w_i^{(t+1)} &= w_i^{(t)} + \eta \sum_j x_i^j [y^j - p(y^j = 1 | x^j; w^{(t)})],
\end{aligned}$$

până când diferența dintre valorile funcției ℓ la două iterații consecutive scade sub un anumit prag (fixat dinainte) $\varepsilon > 0$.

c. Explicați de ce regresia logistică este un clasificator de tip *discriminativ*, spre deosebire de clasificatorii de tip *generativ*, cum este de exemplu algoritmul Bayes Naiv (pentru acesta din urmă, vedeți capitolul *Clasificare bayesiană*).

d. Vă readucem aminte că regula de predicție / decizie pentru regresia logistică este următoarea:

dacă $p(y^j = 1 | x^j) > p(y^j = 0 | x^j)$, atunci alege predicția 1,
altfel alege predicția 0.

Cum arată *granița de decizie* (engl., decision boundary) corespunzătoare modelului de regresie logistică? Justificați răspunsul dumneavoastră. (Ați putea, de exemplu, să scrieți ecuația corespunzătoare graniței de decizie în funcție de w_0, w_1, w_2 și x_1^j, x_2^j .)

În ultima parte a acestei probleme ne vom concentra asupra estimării parametrilor regresiei logistice în sensul probabilității maxime a posteriori (engl., maximum a posteriori probability, MAP).

⁹⁹Pentru demonstrația acestei afirmații în cazul general (deci nu doar pentru cazul datelor din \mathbb{R}^2), vedeți problema 13.

e. Considerând o distribuție a priori de tip Laplace $p(w) = \prod_i \frac{1}{2b} e^{-\frac{|w_i|}{b}}$, calculați valoarea parametrului w care maximizează expresia funcției de probabilitate a posteriori. Așadar, veți calcula

$$\hat{w} = \operatorname{argmax}_w \ln[p(w) \prod_j p(y^j | x^j, w)].$$

Sugestie: Rezultatul pe care îl veți obține ar trebui să fie foarte asemănător cu expresia (62), cu singura diferență că noul rezultat va conține un termen suplimentar, care corespunde distribuției a priori.

f. Care este expresia pe care ar trebui să o aibe acum derivata parțială a estimării de probabilitate condiționată maximă a posteriori (M(C)AP)?

Sugestie: Ar trebui să puteți identifica / separa în expresia funcției de probabilitate condiționată maximă a posteriori (M(C)AP) termenul corespunzător distribuției a priori de termenul corespunzător funcției de verosimilitate condițională (M(C)LE). La final ar trebui să obțineți o expresie similară cu expresia (65), având însă un termen suplimentar.

32.

(Regresia logistică:
analiza efectului duplicării atributelor)
CMU, 2011 spring, Tom Mitchell, midterm, pr 5.3

Vom considera o problemă de clasificare binară pe date caracterizate de variabila / atributul $X_1 \in \{0, 1\}$ și eticheta $Y \in \{0, 1\}$.

Fie un set de date de antrenament format din n exemple: $D_1 = \{(x_1^1, y^1), \dots, (x_1^n, y^n)\}$. Presupunem că generăm un alt set de date de antrenament, D_2 , format tot din n exemple, $D_2 = \{(x_1^1, x_2^1, y^1), \dots, (x_1^n, x_2^n, y^n)\}$, unde în fiecare exemplu x_1 și y sunt aceleași ca în D_1 , iar x_2 este copia lui x_1 .

Învățăm un model de regresie logistică folosind setul de date D_1 ; acest model va avea doi parametri: w_0 și w_1 . De asemenea, învățăm un alt model de regresie logistică folosind setul de date D_2 ; acesta va avea trei parametri: w_0 , w_1 și w_2 .

Mai întâi, scrieți *funcția obiectiv* pe care o folosim pentru a estima (în sensul maximizării verosimilității condiționale) parametrii (w_0, w_1) și respectiv (w_0, w_1, w_2) , pe baza datelor de antrenament.

Apoi, analizând cele două funcții obiectiv, precizați care este *relația* dintre seturile de parametri (w_0, w_1) și (w_0, w_1, w_2) , care se estimatează din datele D_1 și respectiv D_2 . Pornind de la această relație, argumentați de ce (sau cum anume) va fi afectată regresia logistică (sau *nu* va fi afectată) de duplicarea variabilei X_1 .

33.

(Regresia liniară și regresia logistică:
definiții [,revizitate"],
o interesantă proprietate comună)
■ CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 4

Dată fiind o instanță (sau, un input) $X \in \mathbb{R}^d$ împreună cu „răspunsul” / outputul corespunzător $Y \in \mathbb{R}$, *regresia liniară* [cu „zgomot” gaussian] construiește un model de forma

$$Y|X \sim \text{Normal}(\mu(X), \sigma^2),$$

unde media $\mu(X)$ este o funcție liniară de componente / atributelor inputului: $\mu(X) = \theta^\top X = \theta_0 + \theta_1 X_1 + \dots + \theta_d X_d$.

Dacă $Y \in \{0, 1\}$, regresia logistică — care, spre deosebire de regresia liniară servește pentru clasificare — modelează outputul Y astfel:

$$Y|X \sim \text{Bernoulli}(h_\theta(X)),$$

unde $h_\theta(X)$, parametrul acestei distribuții Bernoulli, este obținut din $\theta^\top X$ aplicând funcția logistică / sigmoidală:

$$h_\theta(X) = g(\theta^\top X),$$

unde prin g am notat funcția logistică

$$g(z) \stackrel{\text{def.}}{=} \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

sau, echivalent, folosind funcția logit (care este inversa funcției logistice / sigmoidale):¹⁰⁰

$$\text{logit}(h_\theta(X)) \stackrel{\text{def.}}{=} \ln \frac{h_\theta(X)}{1 - h_\theta(X)} = \theta^\top X$$

Comentariu: Definițiile date mai sus pun în evidență un anumit „paralelism“ pentru cele două modele de regresie. În această problemă,

- mai întâi, veți putea vedea încă o similaritate, și anume între *vectorii gradient* corespunzători funcțiilor de log-verosimilitate condițională pentru cele două metode de regresie. Ne referim aici la expresia $\nabla_\theta \ell(\theta) = \sum_{i=1}^n (y_i - h_\theta(x_i))x_i$ (care corespunde relației (32) de la problema 12) pentru gradientul regresiei logistice și la expresia $\nabla_\theta \ell(\theta) = \sum_{i=1}^n (y_i - \theta^\top x_i)x_i$ pentru gradientul regresiei liniare, conform termenului principal din relația (17), care a fost obținută la problema 6;
- iar acum veți demonstra o proprietate de tip probabilist, care se scrie în mod identic(!) pentru cele două modele de regresie și care folosește media condițională a outputului Y în raport cu inputul X și cu $\hat{\theta}$, estimarea în sens MLE pentru parametrul θ .

Veți considera setul de date de antrenament $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$.

Demonstrați că pentru fiecare dintre cele două modele de regresie de mai sus, estimarea de verosimilitate maximă a parametrului θ (notație: $\hat{\theta}$) satisfac următoarea proprietate:

$$\sum_{i=1}^n y_i x_i = \sum_{i=1}^n E[Y|X = x_i, \theta = \hat{\theta}] x_i.$$

Observații:

1. Remarcați faptul că lui y_i din partea stângă a acestei egalități îi corespunde în partea dreapta o medie, și anume media condițională a outputului Y , dat fiind inputul $X = x_i$ și $\hat{\theta}$, estimarea în sens MLE pentru parametrul θ .
2. Datorită faptului că $y_i \in \{0, 1\}$, rezultă că membrul stâng al acestei egalități este de fapt suma acelor instanțe x_i pentru care $y_i = 1$. Membrul drept al egalității este o sumă ponderată a tuturor instanțelor x_i . Ponderile respective sunt mediile variabilelor aleatoare condiționate (de tip gaussian, respectiv de tip Bernoulli) scrise în mod unitar sub forma $Y|X = x_i, \theta = \hat{\theta}$.

¹⁰⁰Intr-adevăr, $\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z} = y \in (0, 1) \Rightarrow e^z(1 - y) = y \Rightarrow z = \ln \frac{y}{1 - y}$.

34. (Regresia logistică n -ară (i.e., regresia softmax): echivalență cu un caz particular de mixtură de distribuții gaussiene multidimensionale)
MIT, 2001 fall, Tommi Jaakkola, HW2, pr. 3.7

Introducere: La problema 17 am introdus modelul de regresie logistică n -ară (i.e., regresia softmax). Distribuția gaussiană multidimensională a fost introdusă la problema 31 de la capitolul *Fundamente*. Definiția modelului de tip mixtură de distribuții gaussiene multidimensionale a fost prezentată la problema 107 tot de la capitolul *Fundamente*.

Demonstrați că, pentru orice $n \in \mathbb{N}$, $n \geq 2$, modelul probabilist învățat de regresia softmax este echivalent cu un anumit tip de mixtură de distribuții gaussiene multi-variate. Veți specifica / determina în mod riguros ce tip anume de mixtură de distribuții gaussiene multi-variate este echivalent cu modelul învățat de regresia softmax.

35. (Comparații între regresia logistică și alți clasificatori...
 și încă o chestiune, legată de regresia liniară)
CMU, 2009 spring, Tom Mitchell, midterm, pr. 1.1

Explicați pe scurt de ce

- a. pentru a rezolva o anumită problemă de clasificare automată, s-ar putea să fie mai bine să folosim [un algoritm de învățare de] arbori de decizie decât regresia logistică.
- b. pentru a rezolva o anumită problemă de clasificare automată, s-ar putea să fie mai bine să folosim regresia logistică decât algoritmul Bayes Naiv.
- c. atunci când aplicăm regresia liniară, selectăm acele valori ale parametrilor care minimizează suma pătratelor erorilor la antrenare.

36. (Teorema de reprezentare generalizată)
prelucrare de Liviu Ciortuz după Stanford, John Duchi, Supplemental lecture notes

O generalizare a *teoremei de reprezentare* care a fost demonstrată la problema 18.a a fost publicată de Bernhard Schölkopf, Ralf Herbrich și Alex Smola în anul 2001.¹⁰¹ Generalizarea se referă — pe lângă faptul că nu vom mai cere ca funcția de cost $L(z, y)$ să fie derivabilă în raport cu argumentul z , aşa cum s-a considerat la rezolvarea problemei 18 — la faptul că termenul de regularizare $\frac{\lambda}{2} \|w\|^2$ din expresia (48) se înlocuiește cu termenul mai general $\rho(\|w\|)$, unde ρ este o funcție nedescrescătoare.¹⁰²

Similar cu modul în care am procedat la problema 18, vom considera setul de date de antrenament $(x_i, y_i)_{i=1,\dots,n}$, cu $x_i \in \mathbb{R}^d$ și $y_i \in \mathbb{R}$. Acum vom desemna *costul* (sau, *riscul empiric*) regularizat prin funcția $J_\rho(w) \stackrel{\text{not.}}{=} \frac{1}{n} \sum_{i=1}^n L(w \cdot x_i, y_i) + \rho(\|w\|)$.

¹⁰¹Vedeți articolul *A Generalized Representer Theorem* din volumul Computational Learning Theory, Lecture Notes in Computer Science, 2111: 416–426, 2001.

¹⁰²Evident, pentru cazul particular $\rho(\|w\|) = \frac{\lambda}{2} \|w\|^2$ cu $\lambda > 0$, se obține regularizarea de normă L_2 . Într-un alt caz particular remarcabil, ρ poate fi considerată chiar funcția constantă 0, deci *teorema de reprezentare generalizată* acoperă și cazul când nu se face deloc regularizare.

Concluzia teoremei de reprezentare generalizată este formulată astfel:

$$\exists \alpha \in \mathbb{R}^n \text{ a. i. pentru } w^{(\alpha)} \stackrel{\text{not.}}{=} \sum_{i=1}^n \alpha_i x_i \in \mathbb{R}^d \text{ și } \forall w \in \mathbb{R}^d \text{ are loc inegalitatea} \\ J_\rho(w^{(\alpha)}) \leq J_\rho(w).$$

Observație: Importanța (foarte mare!) a acestor *teoreme de reprezentare* constă în faptul că datorită lor rezolvarea problemelor de învățare automată constând în minimizarea unei funcții de tip $J_\rho(w)$ se poate reduce la căutarea unor tupluri $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$,¹⁰³ iar aceasta se poate realiza aplicând algoritmi clasici pentru minimizarea funcțiilor.¹⁰⁴

Demonstrați teorema de reprezentare generalizată.

37.

(Regresia liniară și regresia logistică;
comparație între metoda gradientului și metoda lui Newton:
Adevărat sau Fals?)

Stanford, 2014 fall, Andrew Ng, midterm, pr. 6.gba

a. Presupunem că tu și prietenul tău vreți să folosiți un model de *regresie liniară* pentru a prezice prețul locuințelor. Tu folosești în modelul tău trăsăturile $x_0 = 1$, $x_1 =$ mărimea locuinței în metri pătrați și $x_2 =$ înălțimea acoperișului măsurată în metri. Să zicem că prietenul tău face aceeași analiză, folosind exact același set de date de antrenament, doar că el reprezintă datele folosind trăsăturile $x'_0 = 1$, $x'_1 = x_1$ și $x'_2 =$ înălțimea acoperișului măsurată în cm (așadar, $x'_2 = 100 x_2$).

i. Să zicem că atât tu cât și prietenul tău rezolvați problema de regresie liniară folosind metoda analitică (adică, așa-numitele *ecuații normale*, conform problemei 3). Presupunem că nu suntem într-un *cas de degenerare*, așadar aceste ecuații ne dau o soluție unică pentru parametrii regresiei liniare. Tu obții parametrii / soluția $\theta_0, \theta_1, \theta_2$, în vreme ce prietenul tău obține $\theta'_0, \theta'_1, \theta'_2$. Urmează că $\theta'_0 = \theta_0$, $\theta'_1 = \theta_1$, $\theta'_2 = \frac{1}{100} \theta_2$. Adevărat sau Fals?

ii. Presupunem că atât tu cât și prietenul tău rezolvați problema de regresie liniară initializând parametrii cu 0 și comparând rezultatele pe care le-ați obținut după rularea unei singure iterații a algoritmului / metodei *gradientului [descendent]*, varianta *batch*. [LC: Amândoi folosiți aceeași rată a învățării η .] Tu obții parametrii / soluția $\theta_0, \theta_1, \theta_2$, iar prietenul tău obține $\theta'_0, \theta'_1, \theta'_2$. Urmează că $\theta'_0 = \theta_0$, $\theta'_1 = \theta_1$, $\theta'_2 = \frac{1}{100} \theta_2$. Adevărat sau Fals?

Observație: Punctele b și c care urmează în continuare nu au nicio legătură cu punctul a de mai sus.

b. Dacă utilizăm metoda *gradientului [ascendent]* în varianta *stochastică* pentru antrenarea *regresiei logistică* și folosim o rată fixă a învățării, atunci algoritmul va converge neapărat [în mod exact] la soluția optimă pentru parametrii θ , și anume

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^n p(y^{(i)} | x^{(i)}; \theta),$$

¹⁰³ Altfel spus, căutarea soluțiilor se poate limita la *spațiul liniar generat* de instanțele de antrenament x_i .

¹⁰⁴ Vedeti https://en.wikipedia.org/wiki/Representer_theorem, accesat la data de 30.05.2018.

presupunând că s-a ales o rată a învățării rezonabilă. Adevărat sau Fals?

- c. Presupunem că dispuneți de o implementare pentru *metoda lui Newton* și metoda *gradientului [descendent]* pentru aflarea punctului de optim al unei funcții convexe. Să zicem că o iterație a metodei lui Newton durează dublu față de o iterație a metodei gradientului [descendent]. Prin urmare, rezultă că metoda gradientului [descendent] va converge mai repede la soluția optimă. Adevărat sau Fals?

*Introducere: Modele liniare generalizate*¹⁰⁵

Considerăm că avem o *problemă* de clasificare sau de regresie, în care am dori să prezicem valoarea unei variabile aleatoare y în funcție de [o altă variabilă aleatoare] x . Pentru a deriva un model liniar generalizat (engl., Generalized Linear Model, GLM) pentru această problemă, vom considera următoarele *trei presupozitii* în legătură cu distribuția condițională a lui y în raport cu x , precum și în legătură cu modelul nostru:

1. $y|x; \theta \sim ExponentialFamily(\eta)$. Așadar, fiind x și θ , variabila y urmează o anumită distribuție din familia exponențială, caracterizată de parametrul η :¹⁰⁶

$$p(y|\eta) = b(y) \exp(\eta^\top T(y) - a(\eta)).$$

2. *Scopul* nostru este ca, dat fiind x , să prezicem media valorii lui $T(y)$ în raport cu x . În cele mai multe dintre exemplele didactice, se consideră $T(y) = y$. Așadar, dorim ca predicția $h(x)$ generată (ca output) de către *ipoteza învățată* h să satisfacă relația $h(x) = E[y|x]$.¹⁰⁷
3. Legătura dintre *parametrul natural* η și variabila de intrare x este de tip liniar: $\eta = \theta^\top x$. (Sau, dacă η este vector, atunci $\eta_i = \theta_i^\top x$.)

Cea de-a treia presupozitie poate părea mai puțin justificată decât celelalte două, și ar fi probabil mai indicat să o gândim ca fiind un element de *concepție* (engl., design choice) pentru modelele liniare generalizate (engl., Generalized Linear Models, GLM).

Aceste trei presupozitii / elemente de concepție ne permit să deducem o foarte elegantă clasă de algoritmi de învățare, numită GLM, care au multe proprietăți dezirabile, cum ar fi de pildă facilitatea învățării. Mai mult, modelele rezultate sunt adeseori foarte potrivite pentru a „caracteriza“ diverse tipuri de distribuții peste y . De exemplu, se poate arăta că atât regresia logistică cât și regresia liniară (în varianta celor mai mici pătrate) pot fi văzute ca fiind instanțe ale clasei de algoritmi de învățare automată GLM.

Vom mai adăga câteva *chestiuni de terminologie*, precizăm că funcția g care returnează media distribuției în raport cu parametrul natural η (adică, $g(\eta) = E[T(y); \eta]$) este numită *funcția de răspuns canonică* (engl., canonical response function). Inversa sa, g^{-1} , este numită *funcția de legătură canonică* (engl., canonical link function). Se poate demonstra ușor că funcția de răspuns canonică pentru distribuția gaussiană [cu varianță 1] este funcția identitate, iar pentru distribuția Bernoulli este funcția logistică.

¹⁰⁵Cf. Andrew Ng, Stanford University, Machine learning course, Lecture notes, Part III — Generalized Linear Models.

¹⁰⁶Vedeți problema 111 de la capitolul de *Fundamente*.

¹⁰⁷Veți putea observa că această presupunere este satisfăcută atât în cazul regresiei logistice cât și în cazul regresiei liniare, de către definiția respectivă aleasă pentru $h_\theta(x)$. De exemplu, la regresia logistică avem $h_\theta(x) \stackrel{\text{def.}}{=} \sigma(\theta^\top x) = p(y = 1|x; \theta) = 0 \cdot p(y = 0|x; \theta) + 1 \cdot p(y = 1|x; \theta) = E[y|x; \theta]$.

38. (GLM: particularizare pentru cazul distribuției geometrice și al distribuției Poisson)

Stanford, 2014 fall, Andrew Ng, midterm, pr. 3

Stanford, 2008 fall, Andrew Ng, HW1, pr. 3.abc

University of Chicago, 2004 spring, Michael Eichler, HW3, pr. 4.bd

Vă readucem aminte că forma standard a familiei de distribuții exponentiale este

$$p(y|\eta) = b(y) \exp(\eta^\top T(y) - a(\eta)).$$

De asemenea, vă readucem aminte că distribuția geometrică de parametru ϕ are funcția masă de probabilitate (p.m.f.) definită în felul următor:

$$p(y|\phi) = (1-\phi)^{y-1} \phi, \text{ pentru } y = 1, 2, \dots.$$

- a. Arătați că distribuția geometrică face parte din familia de distribuții exponentiale și precizați valorile corespunzătoare pentru $b(y)$, η , $T(y)$ și $a(\eta)$.
- b. Presupunem că vrem să facem regresie folosind un model liniar generalizat (engl., Generalized Liniar Model, GLM) cu o variabilă de răspuns geometrică. Precizați care este – în acest caz – funcția canonica de răspuns. (Puteti utiliza faptul că media distribuției geometrice este $1/\phi$.)
- c. Vă readucem aminte că în modelul GLM, se consideră că $\eta = \theta^\top x$, cu θ și x din \mathbb{R}^d . De asemenea, dat fiind un set de date de antrenament $\{(x_i, y_i)\}_{i=1}^m$ cu $x_i \in \mathbb{R}^d$ și $y_i \in \mathbb{R}$, se știe că log-verosimilitatea unui exemplu (x_i, y_i) este $\ln p(x_i, y_i)$. Calculați derivata acestei log-verosimilități în raport cu θ_j , pentru a obține regula gradientului ascendent în varianta stochastică, în vederea antrenării unui model GLM cu răspunsuri geometrice y și funcție canonica de răspuns.
- d. Calculați matricea hessiană H pentru $\ell(\theta) = \sum_{i=1}^m \ln p(y_i|x_i; \theta)$, funcția de log-verosimilitate a datelor de antrenament, și precizați cum va arăta aplicarea unui pas al metodei lui Newton pentru maximizarea acestei funcții de log-verosimilitate.
- e. Vă cerem să satisfaceti din nou cerințele de la punctele $a - d$ de mai sus, însă de data aceasta lucrând cu distribuția Poisson de parametru λ . Vă readucem aminte că pentru această distribuție funcția masă de probabilitate are forma

$$p(y|\lambda) = \frac{e^{-\lambda} \lambda^y}{y!},$$

iar media este egală cu λ .

39. (GLM: particularizare pentru cazul distribuției gaussiene unidimensionale)

Stanford, 2015 fall, Andrew Ng, midterm, pr. 3

Vă readucem aminte că forma standard a familiei de distribuții exponentiale este

$$p(y|\eta) = b(y) \exp(\eta^\top T(y) - a(\eta)).$$

De asemenea, vă readucem aminte că funcția densitate de probabilitate pentru distribuția gaussiană [unidimensională] este definită în felul următor:

$$p(y|\mu, \sigma^2) = \frac{1}{\sigma^2} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right).$$

a. Arătați că distribuția gaussiană face parte din familia de distribuții exponențiale. Specificați valorile corespunzătoare pentru $b(y)$, η , $T(y)$, și $a(\eta)$.

Indicație: η și $T(y)$ vor fi vectori bidimensionali; veți folosi notația $\eta = (\eta_1, \eta_2)^\top$. Veți exprima $a(\eta)$ în funcție de η_1 și η_2 .

b. Presupunem că disponem de setul de date de antrenament $\{(x_i, y_i)\}_{i=1}^m$, cu $x_i \in \mathbb{R}^d$ și $y \in \mathbb{R}$. Pornind de la $p(y|\eta)$, forma standard a familiei exponențiale dată mai sus, calculați expresia [generală a] matricei hessiene pentru funcția de log-verosimilitate $\ell(\theta) = \sum_{i=1}^m \ln p(y_i|x_i; \theta)$, cu $\theta \in \mathbb{R}^d$. (Vă readucem aminte că în modelul GLM se lucrează cu $\eta = \theta^\top x$.) Veți scrie expresia acestei matrice hessiene în raport cu x , η_1 și η_2 .

c. Folosind rezultatul de la punctul b, arătați că matricea hessiană [pentru funcția de log-verosimilitate $\ell(\theta)$] este negativ semidefinită, adică $z^\top Hz \leq 0$ pentru orice $z \in \mathbb{R}^d$.

40. (GLM: calculul funcției de log-verosimilitate în cazul general,
condiții suficiente pentru concavitate;
calculul gradientului; calculul matricei hessiene)

*prelucrare de Liviu Ciortuz, după
Stanford, 2009 fall, Andrew Ng, practice midterm, pr. 1
Stanford, 2008 fall, Andrew Ng, HW1, pr. 3.d*

Vă readucem aminte că în modelul liniar generalizat (engl., generalized linear models, GLM) se presupune că variabila de răspuns y (condiționată în raport cu x) urmează o distribuție care face parte din *familia de distribuții exponențiale*:

$$p(y|\eta) = b(y) \exp(\eta^\top T(y) - a(\eta)), \quad (66)$$

unde $\eta = \theta^\top x$. În acest exercițiu vom presupune că $\eta \in \mathbb{R}$.

a. Știți că fiind dat setul de date de antrenament $\{(x_i, y_i)\}_{i=1}^m$, funcția de log-verosimilitate [condițională] se definește astfel:

$$\ell(\theta) = \sum_{i=1}^m \ln p(y_i|x_i; \theta).$$

Formulați un set de condiții pe care trebuie să le satisfacă $b(y)$, $T(y)$ și $a(\eta)$ din relația (66) în așa fel încât funcția $\ell(\theta)$ să fie concavă (asigurându-ne astfel că ea are o valoare maximă unică). Condițiile pe care le veți formula trebuie să fie rezonabile și, de asemenea, trebuie să fie cât se poate de laxe (engl., weak).¹⁰⁸

b. Verificați că în cazul în care variabila de răspuns urmează distribuția gaussiană standard, avem $b(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$, $T(y) = y$ și $a(\eta) = \frac{\eta^2}{2}$. Verificați de asemenea că setul

¹⁰⁸De pildă, răspunsul „orice $b(y)$, $T(y)$ și $a(\eta)$ astfel încât funcția $\ell(\theta)$ să fie concavă“ nu este rezonabil. Tot așa, condiții foarte restrictive, care se aplică – de exemplu – doar pentru unele modele particulare de GLMs, nu sunt rezonabile.

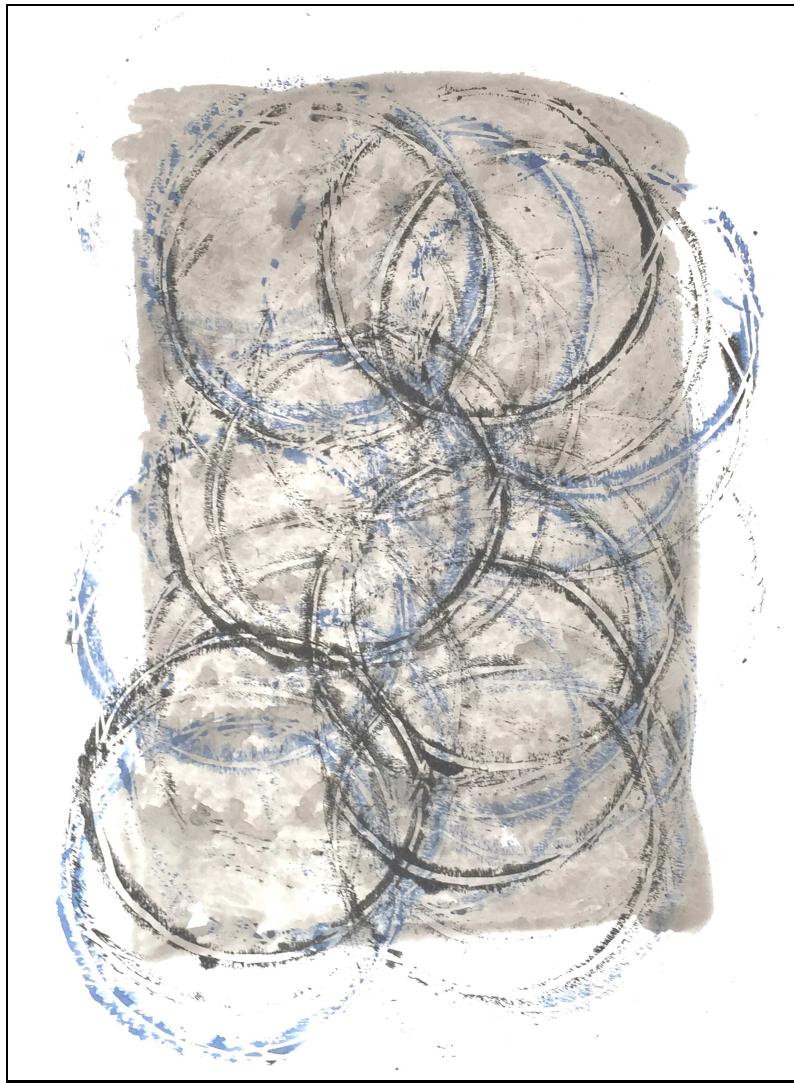
/ sistemul de condiții pe care le-ați formulat la punctul a este satisfăcut în acest caz particular.

c. Presupunem că folosim un model GLM având ca *variabilă de răspuns* orice membru al familiei de distribuții exponentiale, cu $T(y) = y$ și *funcția de răspuns canonic* pentru această familie.

Arătați că atunci când trebuie să aplicăm metoda gradientului ascendent în varianta stochastică pentru a determina maximul funcției de log-verosimilitate $\ln p(y|X, \theta)$, obținem următoarea *regulă de actualizare*:

$$\theta_i \leftarrow \theta_i - \alpha(h(x) - y)x_i,$$

unde α este *rata de învățare*.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

2 Clasificare bayesiană

Sumar

Noțiuni preliminare

- probabilități și probabilități condiționate;
- formula lui Bayes: ex. 5.b;
cap. *Fundamente*, ex. 6, ex. 7, ex. 83, ex. 84;
- independența [condițională a] evenimentelor aleatoare:
cap. *Fundamente*, ex. 4, ex. 80, ex. 81;
- independența [condițională a] variabilelor aleatoare: ex. 9, ex. 10, ex. 12, ex. 31-38; vedeti și cap. *Fundamente*, ex. 15, ex. 27, ex. 88.b, ex. 97, ex. 95;
- distribuții probabiliste comune, marginale și condiționale: ex. 8, ex. 10, ex. 12, ex. 31; vedeti și cap. *Fundamente*, ex. 13, ex. 14;
- distribuția gaussiană: de la cap. *Fundamente*, ex. 29, ex. 30 (pentru cazul unidimensional), ex. 32 (pentru cazul bidimensional), ex. 20, ex. 31, ex. 33, ex. 34 (pentru cazul multidimensional);
- estimarea parametrilor pentru distribuții de tip Bernoulli, categorial și gaussian (ultimul doar pentru cazul clasificării bayesiene de tip gaussian);¹⁰⁹
- ipoteze MAP vs. ipoteze ML:
formulare [ca soluții la] probleme de optimizare;¹¹⁰ ex. 25;
exemplificare: ex. 1, ex. 2, ex. 3, ex. 24, ex. 37;
exemplificare în cazul arborilor de decizie: ex. 4;
- regresia logistică, chestiuni introductive;¹¹¹ de la cap. *Metode de regresie*, ex. 12.

Algoritmi de clasificare bayesiană

- Algoritmul Bayes Naiv și algoritmul Bayes Optimal;¹¹²
formulare ca probleme de optimizare / estimare în sens MAP: cartea ML, pag. 167;
pseudo-cod: cartea ML, pag. 177; vedeti și slide-urile lui Tom Mitchell;
exemple de aplicare: ex. 5, ex. 7, ex. 8, ex. 9, ex. 26, ex. 27, ex. 28;
- aplicarea / adaptarea algoritmului Bayes Naiv pentru clasificare de texte;¹¹³ ex. 6, ex. 29; folosirea regulii “add-one” [a lui Laplace] pentru „netezirea” parametrilor: ex. 6, ex. 30;

¹⁰⁹ De la cap. *Fundamente*, pentru estimarea parametrului unei distribuții Bernoulli vedeti ex. 40 și ex. 113.a, pentru estimarea parametrilor unei distribuții categoriale vedeti ex. 115, iar pentru estimarea parametrilor unei distribuții gaussiane vedeti ex. 45, ex. 46, ex. 122 (pentru cazul unidimensional) și ex. 48 (pentru cazul multidimensional).

¹¹⁰ Vedeti cartea ML, pag. 156-157.

¹¹¹ Vedeti draftul capitolului suplimentar pentru cartea ML a lui T. Mitchell, *Generative and discriminative classifiers: Naive Bayes and logistic regression* (în special secțiunea 3).

¹¹² La secțiunea aceasta, precum și la următoarea secțiune, considerăm (implicit) că toate variabilele de intrare sunt de tip Bernoulli sau, mai general, de tip categorial. După aceea vom considera și variabile de intrare de tip continuu, în genere de tip gaussian. Variabila de ieșire se consideră întotdeauna de tip Bernoulli / categorial.

¹¹³ Atenție: Noi am folosit aici versiunea de bază a algoritmului Bayes Naiv; varianta “bag of words” (vedeti cartea Machine Learning a lui Tom Mitchell, pag. 183) diferă ușor de aceasta.

- calculul ratei medii a erorilor pentru algoritmii Bayes Naiv și Bayes Optimal: ex. 10, ex. 11, ex. 31, ex. 32, ex. 33, ex. 34, ex. 38;
- evidențierea grafică a neconcordanței predicțiilor făcute de clasificatorii Bayes Naiv și Bayes Optimal: ex. 12.

Proprietăți ale algoritmilor Bayes Naiv și Bayes Optimal

- (P0) dacă proprietatea de independentă condițională a atributelor de intrare în raport cu variabila de ieșire se verifică, atunci rezultatele produse de către cei doi algoritmi (Bayes Naiv și Bayes Optimal) în faza de testare coincid;
- (P1) numărul de parametri necesari de estimat din date: liniar pentru Bayes Naiv ($2d+1$) și exponential pentru Bayes Optimal ($2^{d+1} - 1$);¹¹⁴ ex. 7.e, ex. 28.ab, ex. 33.ac;
- (P2) complexitatea algoritmului Bayes Naiv:
 - complexitatea de spațiu: $\mathcal{O}(dn)$
 - complexitatea de timp:
 - la antrenare: $\mathcal{O}(dn)$
 - la testare: $\mathcal{O}(d')$,

unde n este numărul de exemple, iar d este numărul de atribute de intrare [LC: d' este numărul de atribute de intrare din instanța de test];
- (P3) algoritmul Bayes Optimal poate produce eroare [la clasificare] din cauza faptului că ia decizia în sensul unui vot majoritar. Algoritmul Bayes Naiv are și el această „sursă“ de eroare; în plus el poate produce eroare și din cauza faptului că lucrează cu presupoziția de independentă condițională (care nu este satisfăcută în mod neapărat);
- (P4) acuratețea [la clasificare] al algoritmului Bayes scade atunci când unul sau mai multe atribute de intrare sunt duplicate: ex. 10.d, ex. 31.def;
- (P5) în cazul „învățării“ unei funcții booleene (oarecare), rata medie a erorii produse la antrenare de către algoritmul Bayes Optimal (spre deosebire de Bayes Naiv!) este 0: ex. 33.d;
- (P6) complexitatea de eșantionare: de ordin logaritmic pentru Bayes Naiv și de ordin exponential pentru Bayes Optimal: ex. 13;
- (P7) corespondența dintre regula de decizie a algoritmului Bayes Naiv (când toate variabilele de intrare sunt de tip Bernoulli) și regula de decizie a *regresiei logistice* și, în consecință, liniaritatea granițelor de decizie: ex. 14.
- *comparații* între algoritmul Bayes Naiv și alți algoritmi de clasificare automată: ex. 36, ex. 38.

Algoritmii Bayes Naiv și Bayes Optimal cu variabile de intrare de tip gaussian

- Aplicare: G[N]B: ex. 15, ex. 40 și ex. 48; GJB: ex. 45, ex. 46 și ex. 47; GNB vs. GJB: ex. 21.
- Numărul de parametri necesari de estimat din date: ex. 42.
- Proprietăți:

¹¹⁴Numărul de parametri indicați în paranteze se referă la cazul când atât atributele de intrare cât și atributul de ieșire sunt de tip Bernoulli.

- (P0') presupunem că variabila de ieșire este booleană, i.e. ia valorile 0 sau 1; dacă pentru orice atribut de intrare, variabilele condiționale $X_i|Y = 0$ și $X_i|Y = 1$ au distribuții gaussiene de varianțe egale ($\sigma_{i0} = \sigma_{i1}$), atunci regula de decizie GNB (Gaussian Naive Bayes) este echivalentă (ca formă) cu cea a regresiei logistice, deci separarea realizată de către algoritmul GNB este de formă liniară: demonstrație: ex. 17; exemplificare în \mathbb{R} : ex. 40.a; exemplificare în \mathbb{R}^2 : ex. 41.c;
 - (P1') similar, presupunem că variabila de ieșire este booleană; dacă variabilele de intrare (notație: $X = (X_1, \dots, X_d)$) au distribuțiile [comune] condiționale $X|Y = 0$ și $X|Y = 1$ de tip gaussian [multidimensional], cu matricele de covarianță egale ($\Sigma_0 = \Sigma_1$), atunci regula de decizie a algoritmului “full” / Joint Gaussian Bayes este și ea echivalentă (ca formă) cu cea a regresiei logistice, deci separarea realizată este tot de formă liniară: ex. 18, ex. 20.a.i – ii;
 - (P2') când variabilele de intrare satisfac condiții mixte de tip (P0') sau (P7), atunci concluzia – separare liniară – se menține: ex. 44.b;
 - (P3') dacă în condițiile de la propozițiile (P0')-(P2') presupozitia de independentă condițională este satisfăcută, iar numărul de instanțe de antrenament tinde la infinit, atunci rezultatul de clasificare obținut de către algoritmul Bayes Naiv gaussian este identic cu cel al regresiei logistice: ex. 22.a.
- Atunci când presupozitia de independentă condițională nu este satisfăcută, iar numărul de instanțe de antrenament tinde la infinit, regresia logistică se comportă mai bine decât algoritmul Bayes Naiv [gaussian]: ex. 22.b;
- (P4') nu există o corespondență 1-la-1 între parametrii calculați de regresia logistică și între parametrii calculați de algoritmul Bayes Naiv [gaussian]: ex. 23.a;
 - (P5') atunci când varianțele distribuțiilor gaussiene care corespund probabilităților condiționale $P(X_i|Y = k)$ depind și de eticheta k , separatorul decizional determinat de algoritmul Bayes Naiv gaussian nu mai are (în mod necesar) forma regresiei logistice: ex. 40.bc, ex. 43; similar, pentru algoritmul Bayes Optimal gaussian, atunci când $\Sigma_0 \neq \Sigma_1$, ecuația separatorului decizional este de ordin pătratic: ex. 19, ex. 20.a.iii – vi, ex. 42.e (separatorul decizional este un cerc), ex. 42.f (o hiperbolă), ex 47 (o reunioare de două drepte);
 - (P6') parametrii algoritmilor Bayes Naiv gaussian și Bayes Optimal gaussian se pot estima în timp liniar în raport cu numărul de instanțe din setul de date de antrenament: ex. 41.bd, ex. 23.b.

2.1 Probleme rezolvate

2.1.1 Ipoteze de probabilitate maximă a posteriori (MAP)

1. (Formula lui Bayes; medii ale unor variabile aleatoare discrete; [ipoteze MAP;] măsuri statistice folosite în clasificare)

■ CMU, 2009 fall, Geoff Gordon, HW1, pr. 2

O anumită boală afectează una din 500 de persoane în medie. Identificarea persoanelor care au această boală se poate face cu ajutorul unei analize a săngelui, care costă 100 de

dolari de persoană. Această analiză indică în cazul unui rezultat *pozitiv* faptul că se *poate* ca persoana respectivă să suferă de acea boală.

Testul / analiza are o *sensibilitate* (engl., *sensitivity* sau *recall*) perfectă — adică raportul dintre numărul instanțelor pozitive identificate ca atare de acel test și numărul total de instanțe pozitive este 1 —, ceea ce înseamnă că pentru orice persoană care are boala respectivă, rezultatul testului este pozitiv cu probabilitate de 100%. Pe de altă parte, testul are o *specificitate* — raportul dintre numărul instanțelor negative identificate ca atare de acel test și numărul total de instanțe negative — de 99%, adică o persoană care nu suferă de acea boală va avea cu probabilitate de 1% rezultatul testului pozitiv.

- Se testează o persoană selectată în mod aleatoriu, iar rezultatul este pozitiv. Care este probabilitatea ca persoana respectivă să suferă de acea boală?
- Există și un al doilea test, care costă 10.000 de dolari și are atât sensibilitatea cât și specificitatea de 100%. Dacă am cere ca toate persoanele detectate pozitiv la testul precedent să fie supuse acestui test mult mai scump, care ar fi costul mediu pentru testarea / analiza unui individ?
- O companie farmaceutică încearcă să reducă prețul celui de-al doilea test (care este perfect), adică are atât *sensibilitatea* cât și *specificitatea* de 100%. Cât ar trebui să fie prețul acesta pentru ca primul test să nu mai fie necesar? (Adică, la ce preț va rezulta că este mai ieftin să se utilizeze doar testul al doilea, decât să se facă ambele teste, ca la punctul b?)

Răspuns:

Definim următoarele variabile aleatoare:

B : ia valoarea 1 / adevărat pentru persoanele care suferă de această boală și 0 / fals în caz contrar

T_1 : rezultatul primului test, care poate fi + (în caz de boală) sau -

T_2 : rezultatul celui de-al doilea test, care poate fi tot + sau -.

Folosind aceste variabile aleatoare, datele problemei se pot scrie astfel:

$$\begin{aligned} P(B) &= \frac{1}{500} \\ P(T_1 = + | B) &= 1 \\ P(T_1 = + | \bar{B}) &= \frac{1}{100} \\ P(T_2 = + | B) &= 1 \\ P(T_2 = + | \bar{B}) &= 0 \end{aligned}$$

- Probabilitatea ca o persoană oarecare să suferă de boala respectivă, știind că rezultatul primului test este pozitiv, este $P(B | T_1 = +)$ și se calculează cu ajutorul formulei lui Bayes:

$$\begin{aligned} P(B | T_1 = +) &= \frac{P(T_1 = + | B) \cdot P(B)}{P(T_1 = + | B) \cdot P(B) + P(T_1 = + | \bar{B}) \cdot P(\bar{B})} \\ &= \frac{1 \cdot \frac{1}{500}}{1 \cdot \frac{1}{500} + \frac{1}{100} \cdot \frac{499}{500}} = \frac{100}{599} \approx 0.1669 \end{aligned}$$

Observație: Remarcați faptul că $P(\bar{B} | T_1 = +) = 0.8331$. Este imediat care dintre cele două probabilități, $P(B | T_1 = +)$ și $P(\bar{B})$, este mai mare. În mod echivalent, pentru a stabili acest fapt era suficient să comparăm $P(B | T_1 = +)$ cu $1/2$, sau să stabilim care dintre produsele $P(T_1 = + | B) \cdot P(B)$ și $P(T_1 = + | \bar{B}) \cdot P(\bar{B})$ este mai mare. Aceste observații sunt utile pentru că ele fac legătura cu noțiunea de *ipoteză de probabilitate maximă a posteriori* (vedeți pr. 3).

b. Vom calcula costul mediu al testării unui individ folosind o nouă variabilă aleatoare, notată cu C , care reprezintă costul total de testare al unei persoane. Notând cu c_1 și c_2 costurile celor două teste, putem scrie:

$$C = \begin{cases} c_1 & \text{dacă persoana este testată doar cu primul test} \\ c_1 + c_2 & \text{dacă persoana este testată cu ambele teste} \end{cases}$$

O persoană este testată cu al doilea test doar dacă are rezultatul pozitiv la primul test, deci probabilitățile pentru variabila aleatoare C sunt:

$$P(C = c_1) = P(T_1 = -) \text{ și } P(C = c_1 + c_2) = P(T_1 = +)$$

Costul mediu cerut de problemă este media variabilei aleatoare C , deci se poate calcula astfel:

$$\begin{aligned} E[C] &= c_1 \cdot P(C = c_1) + (c_1 + c_2) \cdot P(C = c_1 + c_2) \\ &= c_1 \cdot P(T_1 = -) + (c_1 + c_2) \cdot P(T_1 = +) \end{aligned}$$

Stim că $P(T_1 = -) = 1 - P(T_1 = +)$, iar din formula probabilității totale avem:

$$\begin{aligned} P(T_1 = +) &= P(T_1 = + | B) \cdot P(B) + P(T_1 = + | \bar{B}) \cdot P(\bar{B}) \\ &= 1 \cdot \frac{1}{500} + \frac{1}{100} \cdot \frac{499}{500} = \frac{599}{50000} = 0.01198 \end{aligned}$$

Așadar, vom obține:

$$\begin{aligned} E[C] &= c_1 \cdot (1 - P(T_1 = +)) + (c_1 + c_2) \cdot P(T_1 = +) \\ &= c_1 - c_1 \cdot P(T_1 = +) + c_1 \cdot P(T_1 = +) + c_2 \cdot P(T_1 = +) \\ &= c_1 + c_2 \cdot P(T_1 = +) \\ &= 100 + 10000 \cdot \frac{599}{50000} = 219.8 \approx 220\$ \end{aligned}$$

c. Notăm cu c_n noul preț pentru al doilea test (T'_2). Acest preț trebuie să fie mai mic sau egal cu costul mediu de aplicare al ambelor teste (T_1 și T'_2), deci:

$$\begin{aligned} c_n \leq E[C'] &= c_1 \cdot P(C = c_1) + (c_1 + c_n) \cdot P(C = c_1 + c_n) \\ &= c_1 + c_n \cdot P(T_1 = +) \\ &= 100 + c_n \cdot \frac{599}{50000} \end{aligned}$$

Rezolvând ecuația $c_n = 100 + c_n \cdot 0.01198$, obținem $c_n \approx 101.2125$.

Așadar, dacă al doilea test ar costa cel mult 101.21 dolari, atunci primul test nu-ar mai fi necesar.

2. (Formula lui Bayes; [ipoteze MAP;] inferențe statistice
prelucrare de Liviu Ciortuz, după CMU, 2009 fall, Geoff Gordon, HW1, pr. 1
 (“Monty’s haunted house” problem)

Fără să ştii cum s-a întâmplat, ai nimerit într-o casă plină de fantome. Acum ești blocat în fața unui perete care are 3 uși (pentru conveniență, le vom nota cu numerele 1, 2, 3). Apare o fantomă care îți spune: „Scăparea ta este să ieși din casă printr-una din aceste uși. Însă doar una dintre ele dă în afară; celelalte două sunt păzite de câte un monstru care te va ucide imediat dacă încerci să ieși pe acolo. Trebuie să alegi o ușă!“

Decizi să alegi la întâmplare una din cele trei uși, să zicem ușa 1.

Observație: Probabilitatea *a priori* ca ușa aceasta să dea în afară este 1/3. (La fel este și în cazul celorlalte două uși.) Prin adăgarea altor informații — vedetă continuarea problemei — probabilitatea *a posteriori* a aceluiasi eveniment se poate modifica, deci într-un caz fericit ea poate crește.

Într-adevăr, tocmai când ai pus mâna pe clanță ca să o deschizi, fantoma îți spune: „Așteaptă puțin! Îți voi mai da o informație.“ Zicând aceasta, fantoma întredeschide o altă ușă (să zicem ușa 2) și îți arată că în spatele ei se află un monstru groaznic. Apoi fantoma te întreabă: „Vrei acum să alegi ultima ușă (adică ușa 3) sau consideri că este mai bine să rămâi la alegerea pe care ai făcut-o inițial?“

Fantomă te mai ajută spunându-ți că în alegerea ei, ea a urmat o *strategie* bazată pe două *principii*:

P1. După ce tu ai făcut alegerea inițială, fantoma a ales una din celelalte două uși, mai precis o ușă în spatele căruia se află un monstru. (Este evident că întotdeauna există o astfel de ușă, indiferent de alegerea ta.)

P2. În cazul în care care ambele uși între care are de ales fantoma au în spate câte un monstru, ea procedează după *una* din următoarele trei *variante* (pe care o alege *a priori* și îi-o aduce la cunoștință):

- Fantomă alege una din cele două uși cu probabilitate egală (1/2).
 - Dacă, aşa cum a fost cazul mai sus, tu ai ales ușa 1, fantoma alege ușa 2 (cu probabilitate 1).
 - Dacă, tot aşa, tu ai ales ușa 1, fantoma alege ușa 3 (cu probabilitate 1).
- (*Notă:* Alte variante nu interesează pentru rezolvarea care se cere mai jos.)

Se cere ca, pentru fiecare din aceste 3 variante în parte, să determini probabilitățile ca ieșirea să se afle în spatele ușii 1, respectiv în spatele ușii 3, dacă fantoma a deschis ușa 2.

Indicație: Pentru rezolvare, vom folosi două *variabile aleatoare*:

- *O* (de la engl. outside), cu valori în mulțimea {1, 2, 3}, indicând unde este ieșirea cea bună;
- *G* (de la engl. ghost), pentru a desemna ușa aleasă de fantomă.

Pentru fiecare din variantele de strategie ale fantomei (*a, b, c*), folosind formula lui Bayes se calculează $P(O = 1 | G = 2)$ și $P(O = 3 | G = 2)$.

Răspuns:

Pentru variabila aleatoare *O* avem probabilități *a priori* egale pentru toate ieșirile:

$$P(O = 1) = P(O = 2) = P(O = 3) = \frac{1}{3}. \quad (67)$$

Folosind formula lui Bayes combinată cu formula probabilității totale, probabilitățile (*a posteriori*) cerute vor putea fi calculate astfel:

$$P(O = 1 | G = 2) = \frac{P(G = 2 | O = 1) \cdot P(O = 1)}{P(G = 2 | O = 1) \cdot P(O = 1) + P(G = 2 | O = 3) \cdot P(O = 3)}$$

$$P(O = 3 | G = 2) = \frac{P(G = 2 | O = 3) \cdot P(O = 3)}{P(G = 2 | O = 3) \cdot P(O = 3) + P(G = 2 | O = 1) \cdot P(O = 1)}.$$

Remarcăm că la fiecare din numitorii celor două fracții de mai sus ar fi trebuit să mai scriem $P(G = 2 | O = 2) \cdot P(O = 2)$, însă acesta este 0 fiindcă $P(G = 2 | O = 2) = 0$, conform principiului P1. Deci $P(O = 3 | G = 2) = 1 - P(O = 1 | G = 2)$.

Pentru a exprima succint probabilitățile condiționate necesare calculului probabilităților $P(O = 1 | G = 2)$ și $P(O = 3 | G = 2)$ folosind formulele de mai sus, completăm următorul tabel:¹¹⁵

G	O	$P(G O)$		
		varianta a	varianta b	varianta c
2	1	1/2	1	0
3	1	1/2	0	1
2	2	0	0	0
3	2	1	1	1
2	3	1	1	1
3	3	0	0	0

În aceste condiții putem calcula ușor probabilitățile cerute în fiecare din cele trei variante:

Varianta a:

$$P(O = 1 | G = 2) = \frac{\frac{1}{2} \cdot \frac{1}{3}}{\frac{1}{2} \cdot \frac{1}{3} + 1 \cdot \frac{1}{3}} = \frac{1}{3} \text{ și } P(O = 3 | G = 2) = 1 - \frac{1}{3} = \frac{2}{3},$$

deci vei alege ușa a treia, fiindcă ea are probabilitatea mai mare de a te duce afară.

Varianta b:

$$P(O = 1 | G = 2) = \frac{1 \cdot \frac{1}{3}}{1 \cdot \frac{1}{3} + 1 \cdot \frac{1}{3}} = \frac{1}{2} \text{ și } P(O = 3 | G = 2) = 1 - \frac{1}{2} = \frac{1}{2},$$

deci vei alege la întâmplare oricare din cele două uși rămase (ușa 1 și ușa 3), ele având aceeași probabilitate de salvare.

Varianta c:

$$P(O = 1 | G = 2) = 0 \text{ și } P(O = 3 | G = 2) = 1 - 0 = 1,$$

deci vei alege ușa a treia, care este cu siguranță ieșirea cea bună.

Observații:

- Echivalent, pentru a determina maximul dintre $P(O = 1 | G = 2)$ și $P(O = 3 | G = 2)$

¹¹⁵Se observă că pe fiecare dintre cele trei coloane din tabel, ultimele patru linii sunt identice, ceea ce este natural, conform principiilor P1 și P2.

ar fi fost suficient, conform formulei lui Bayes, să comparăm $P(G = 2 | O = 1) \cdot P(O = 1)$ și $P(G = 2 | O = 3) \cdot P(O = 3)$. Mai mult, ținând cont de relația (67), aceasta revine la a compara $P(G = 2 | O = 1)$ și $P(G = 2 | O = 3)$. Răspunsul poate fi citit imediat din tabelul de mai sus (vedeți prima linie și penultima linie): în cazul variantei a , $O = 3$ este varianta pentru care se obține probabilitatea [a posteriori] maximă. Altfel spus, pentru variantei a , $O = 3$ este *ipoteza de probabilitate maximă a posteriori* (engl., maximum a posteriori probability (MAP) hypothesis). Absolut similar se poate proceda și pentru variantele b și c .¹¹⁶

2. Alternativ, pentru variantele b și c putem răspunde la întrebare și făcând un raționament care nu folosește formula lui Bayes. Ieșirea cea bună se poate găsi în spatele uneia dintre cele trei uși (vedeți figura alăturată). Cum fantoma a deschis deja ușa cu numărul 2, una dintre aceste situații (și anume, a doua din figură) este eliminată, fiindcă în spatele ei este un monstru. În continuare, putem raționa în felul următor:

1	2	3
	M	M
M		M
M	M	

Varianta b: Întrucât fantoma alege ușa 2 cu probabilitate 1, vom putea afirma că ambele variante – 1 și 3 – au probabilități egale, și anume $\frac{1}{2}$. Într-adevăr,

- fie ușa 1, cea aleasă de mine, dă înspre afară, iar atunci fantoma trebuie, conform principiului P2, să aleagă ușa 2;
- fie ușa 3 dă înspre afară, iar atunci, din nou conform principiului P2, fantoma trebuie să aleagă ușa 2;
- conform principiului P1, nu există o a treia posibilitate;
- nu dispun de alte informații pentru a decide între cele două situații de mai sus.

Varianta c: Știind că fantoma nu a deschis ușa 3 (care ar fi opțiunea corespunzătoare principiului P2), ci a ales ușa 2, înseamnă că nu a putut face altfel, deci ușa 3 reprezintă ieșirea.

3.

(Formula lui Bayes; inferențe statistice; exemplificarea noțiunii de ipoteză / ipoteze MAP (“Maximum A posteriori Probability”))

■ CMU, 2012 spring, Ziv Bar-Joseph, HW1, pr. 1.5

Mickey dă cu zarul de mai multe ori, sperând să obțină un 6. Secvența celor 10 rezultate obținute de el în urma acestor aruncări este următoarea: 1, 3, 4, 2, 3, 3, 2, 5, 1, 6. Mickey se întreabă dacă nu cumva zarul este măsluit (având tendința să producă de mai multe ori față 3 decât ar fi normal dacă zarul ar fi perfect).

Concepți o analiză simplă bazată pe teorema lui Bayes care să-i furnizeze lui Mickey informația care-l interesează: [în ce măsură putem spune că] zarul este măsluit [sau nu]? Explicați raționamentul dumneavoastră.

Veți presupune că în general fiecare set de 100 de zaruri conține 5 zaruri măsluite (engl., unfair) în așa fel încât este favorizată apariția feței 3, rezultând următoarea distribuție de probabilitate a celor șase fețe, (1, 2, 3, 4, 5, 6): $P = [0.1, 0.1, 0.5, 0.1, 0.1, 0.1]$.

¹¹⁶Observație: În cazuri precum cel de mai sus ($P(O = 1) = P(O = 2) = P(O = 3) = 1/3$), ipoteza MAP coincide cu *ipoteza de verosimilitate maximă* (engl., maximul likelihood (ML) hypothesis).

Răspuns:

Acesta este un exercițiu [tipic] de punere în evidență a noțiunii de ipoteză de *probabilitate maximă a posteriori* (engl., Maximum A posteriori Probability, MAP).

Vă reamintim definiția:

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D),$$

unde D este setul de date cu care se lucrează, iar H este mulțimea de ipoteze considerate.

În cazul nostru, $D = \{1, 3, 4, 2, 3, 3, 2, 5, 1, 6\}$, iar $H = \{FD, LD\}$, unde am notat cu FD zarul corect / cinsit (engl., fair dice) și cu LD zarul măsluit (engl., loaded dice).

Folosind formula lui Bayes, definiția de mai sus, poate fi „rafinată“ astfel:

$$h_{MAP} \stackrel{\text{def.}}{=} \operatorname{argmax}_{h \in H} P(h|D) \stackrel{F.B.}{=} \operatorname{argmax}_{h \in H} \frac{P(D|h) \cdot P(h)}{P(D)} = \operatorname{argmax}_{h \in H} P(D|h) \cdot P(h),$$

ultima egalitate având loc datorită faptului că $P(D)$ este o cantitate pozitivă care nu depinde de h .¹¹⁷

Așadar, în cazul de față, a determină ipoteza de probabilitate maximă a posteriori (h_{MAP}) revine la a determina maximul dintre două produse: $P(D|FD) \cdot P(FD)$ și $P(D|LD) \cdot P(LD)$. Facem observația că pentru a calcula $P(D|FD)$ și $P(D|LD)$, vom ține cont de faptul că aruncările zarului au fost independente unele de altele. Prin urmare, notând $D = \{x_1, x_2, \dots, x_{10}\}$, vom putea scrie:

$$\begin{aligned} P(D|FD) \cdot P(FD) &= P(x_1, x_2, \dots, x_{10}|FD) \cdot P(FD) \stackrel{i.i.d.}{=} \left(\prod_{i=1}^{10} P(x_i|FD) \right) \cdot P(FD) \\ &= \left(\frac{1}{6} \right)^{10} \cdot \frac{95}{100} = \frac{1}{2^{10} \cdot 3^{10}} \cdot \frac{19}{20}. \end{aligned}$$

Similar,

$$\begin{aligned} P(D|LD) \cdot P(LD) &= P(x_1, x_2, \dots, x_{10}|LD) \cdot P(LD) \stackrel{i.i.d.}{=} \left(\prod_{i=1}^{10} P(x_i|LD) \right) \cdot P(LD) \\ &= \left(\frac{1}{10} \cdot \frac{1}{2} \cdot \frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{10} \right) \cdot \frac{5}{100} \\ &= \frac{1}{10^7 \cdot 2^3} \cdot \frac{1}{20} = \frac{1}{2^{10} \cdot 5^7} \cdot \frac{1}{20}. \end{aligned}$$

Așadar, a vedea care dintre produsele $P(D|FD) \cdot P(FD)$ și $P(D|LD) \cdot P(LD)$ este mai mare revine la a compara fracțiile $\frac{19}{3^{10}}$ și $\frac{1}{5^7}$. În loc să facem ridicările la putere (3^{10} și 5^7), este mai convenabil să logaritmăm, folosind ca bază un număr supra-unitar:

$$\ln \frac{19}{3^{10}} = \ln 19 - \ln 3^{10} = \ln 19 - 10 \ln 3 = 2.9444 - 10.9861 = -8.0417$$

și

$$\ln \frac{1}{5^7} = -\ln 5^7 = -7 \ln 5 = -11.2661.$$

¹¹⁷Folosind formula probabilității totale, atunci când H este o mulțime discretă, putem exprima $P(D)$ ca $\sum_{h' \in H} P(D|h') \cdot P(h')$. Ar fi util să calculați această probabilitate *a priori* în cazul datelor din acest exercițiu.

Conchidem că ipoteza de probabilitate maximă a posteriori este FD , deci că zarul lui Mickey nu este măsluit.

Observație: Se obișnuieste ca, în loc să se lucreze cu cele două produse ($P(D|FD) \cdot P(FD)$ și $P(D|LD) \cdot P(LD)$) în mod separat, aşa cum am procedat noi mai sus, să se facă raportul lor,

$$\frac{P(D|LD) \cdot P(LD)}{P(D|FD) \cdot P(FD)} = \frac{P(LD|D)}{P(FD|D)}.$$

Acest raport se numește *raportul de șanse* (engl., odds ratio). (Dacă acest raport este supra-unitar, înseamnă că ipoteza LD este mai plauzibilă decât ipoteza FD .) Mai departe, aplicând logaritmul — fiindcă la calcule probabilitatea $P(D|\dots)$ se exprimă ca produs de n factori —, se obține ceea ce în limba engleză se numește *log-odds ratio*. (Evident, dacă *log-odds ratio* are valoare pozitivă, ipoteza LD este mai plauzibilă.) Pe datele noastre,

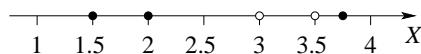
$$\ln \frac{P(LD|D)}{P(FD|D)} = -11.2661 - (-8.0417) = -3.2244 < 0.$$

În concluzie, folosind un astfel de raport (de „potrivire“), vom putea spune nu doar care ipoteză este mai plauzibilă, ci și *în ce măsură* acea ipoteză (în cazul nostru, FD) este mai plauzibilă decât cealaltă ipoteză (LD).

4. (Arbore ID3 cu decizii probabiliste, ca ipoteze ML și respectiv MAP)

■ CMU, 2009 spring, T. Mitchell, midterm, pr. 2

Se consideră următorul set de date de antrenament din spațiul real unidimensional:



Este vorba de 5 date caracterizate de atributul real X , împărțite în două clase: clasa 0 constituie din mulțimea $\{3, 3.5\}$, și clasa 1 – mulțimea $\{1.5, 2, 3.75\}$.

Pe acest set de date se va aplica algoritmul ID3 pentru construirea unor arbori de decizie. Deoarece atributul are valori reale, testelete vor fi de forma $X > t$, unde t reprezintă o valoare-prag.

Se notează cu DT^* algoritmul care construiește arborele de decizie cu număr minim de noduri necesare pentru clasificarea perfectă a datelor de antrenament, și cu $DT1$ algoritmul care construiește un arbore de decizie cu un singur nod de test.

Indicație: Veți presupune că atunci când algoritmul de învățare de arbori de decizie găsește două praguri astfel încât testelete $x < t_1$ și $x < t_2$ produc același câstig de informație, el (adică, algoritmul) va alege pragul cel mai din stânga. (Așadar, dacă $t_1 < t_2$, atunci el va alege testul ($x < t_1$)).

- Care este eroarea la antrenare a lui $DT1$ pe datele specificate? Dar eroarea la cross-validation folosind metoda “Leave-One-Out”?
- Care este eroarea la antrenare a lui DT^* pe datele specificate? Dar eroarea la cross-validation folosind metoda “Leave-One-Out”?

În continuare se consideră o nouă clasă de arbori de decizie, care au *etichete probabiliste*. Fiecare nod frunză specifică probabilitatea fiecărei etichete posibile, probabilitate scrisă

sub forma raportului dintre datele cu acea etichetă din nodul respectiv și toate datele din acel nod.

De exemplu, un arbore de decizie neavând niciun nod de test, construit pe datele specificate mai sus, clasifică astfel: $P(Y = 1) = 3/5$ și $P(Y = 0) = 2/5$. Un arbore de decizie cu un singur nod de test (în raport cu valoarea / pragul 2.5) conține probabilitățile: $P(Y = 1) = 1$ dacă $X \leq 2.5$, și $P(Y = 1) = 1/3$ dacă $X > 2.5$.

c. Pentru setul de date de mai sus, determinați arborele de decizie de tip ML (engl., maximum likelihood), adică acel arbore cu decizii probabiliste care maximizează *verosimilitatea* datelor de antrenament:

$$T_{ML} = \operatorname{argmax}_T P_T(D),$$

unde

$$P_T(D) \stackrel{\text{def}}{=} P(D|T) \stackrel{\text{indep.}}{=} \prod_{i=1}^5 P(Y = y_i | X = x_i, T),$$

y_i fiind eticheta / clasa instanței $x_i \in \{1.5, 2, 3, 3.5, 3.75\}$.

d. Se consideră o distribuție a priori $P(T)$ care penalizează numărul de teste / split-uri din arborele de decizie T , și anume:

$$P(T) \propto \left(\frac{1}{4}\right)^{\text{splits}(T)^2}$$

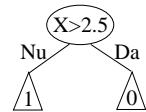
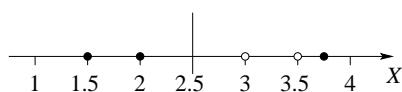
unde $\text{splits}(T)$ reprezintă numărul nodurilor de test din arborele T , iar simbolul \propto înseamnă „este proporțional cu”.

Pentru același set de date, folosind această distribuție a priori $P(T)$, găsiți arborele de decizie de tip MAP (engl., Maximum A posteriori Probability):

$$T_{MAP} = \operatorname{argmax}_T P_T(T|D)$$

Răspuns:

a. Dacă se aplică algoritmul DT1, întrucât câștigurile de informație calculate pentru testele $X > 2.5$ și $X > 3.625$ sunt 0.419 și respectiv 0.171, rezultatul este:



Eroarea la antrenare este $1/5$, deoarece punctul 3.75 este clasificat greșit.

Eroarea la cross-validation cu metoda “Leave-One-Out” se determină astfel:

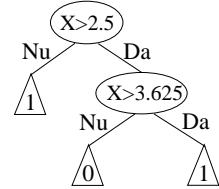
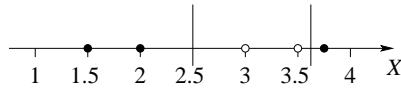
- $x_1 = 1.5$. Testul se face relativ la pragul 2.5 — ținând cont de *Indicația* din enunț —, deci punctul 1.5 este clasificat corect.
- $x_2 = 2$. Testul se face relativ la pragul 2.25 — ținând din nou cont de *Indicația* din enunț —, deci punctul $x_2 = 2$ este clasificat corect.
- $x_3 = 3$. Testul se face relativ la pragul 2.75, fiindcă se verifică imediat că IG-ul acestui prag este $1/2$, deci mai mare decât IG-ul pragului 3.625, și anume 0,311. Prin urmare, punctul $x_3 = 3$ va fi clasificat fie corect (în cazul în care se consideră

că decizia arborelui DT_1 este 0 pentru $X > 2.75$, fie eronat (în cazul în care se consideră că decizia arborelui DT_1 este 1 pentru $X > 2.75$). Însă în al doilea caz arborele DT_1 s-ar reduce la un singur nod (care este nod frunză), ceea ce este contrar definiției sale, aşa că vom reține doar primul caz.

- $x_4 = 3.5$. Testul se face din nou relativ la pragul 2.5 — justificarea este similară cu cea de la punctul precedent —, iar punctul $x_4 = 3.5$ este clasificat fie corect (în cazul în care se consideră că decizia arborelui DT_1 este 0 pentru $X > 2.5$), fie eronat (în cazul în care se consideră că decizia arborelui DT_1 este 1 pentru $X > 2.5$). Însă (din nou!) în al doilea caz arborele DT_1 s-ar reduce la un singur nod (frunză), aşa că vom reține doar primul caz.
- $x_5 = 3.75$. Testul se face tot relativ la pragul 2.5, iar punctul $x_5 = 3.75$ este clasificat greșit.

Deci eroarea la cross-validation cu metoda “Leave-One-Out” este de 1/5, punctul 3.75 fiind clasificat eronat.

- b. Dacă se aplică algoritmul DT^* , rezultatul este:



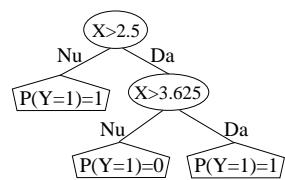
Eroarea la antrenare este bineînțeles 0, deoarece datele de antrenament nu conțin inconistențe.

La cross-validation cu metoda “Leave-One-Out”, punctele care ar putea genera probleme sunt:

- $x_2 = 2$. Cele două teste se fac la pragurile 2.25 și la 3.625 (ordinea nu contează), deci punctul $x_2 = 2$ este corect clasificat.
- $x_3 = 3$. Primul test se face la pragul 2.75, deci punctul $x_3 = 3$ este corect clasificat.
- $x_4 = 3.5$. Al doilea test se face la pragul 3.375, deci punctul $x_4 = 3.5$ este clasificat greșit.
- $x_5 = 3.75$. Se face un singur test (la pragul 2.5), iar punctul $x_5 = 3.75$ este clasificat greșit.

Deci eroarea la cross-validation pentru arborele DT^* folosind metoda “Leave-One-Out” este 2/5.

- c. Un arbore de decizie care maximizează probabilitățile datelor de antrenament va fi unul care clasifică în mod perfect aceste date. Deci un arbore de decizie ML poate fi obținut din arborele construit de DT^* , extins cu etichete probabiliste în frunze, conform figurii alăturate.



- d. Pentru a determina arborele de decizie MAP folosind distribuția a priori $P(T)$, va trebui să comparăm probabilitățile a posteriori ale celor 3 arbori de decizie posibili pe setul de date de antrenament. Vom scrie aceste probabilități a posteriori folosind formula lui Bayes:

$$P(T_j | D) = \frac{P(D | T_j) \cdot P(T_j)}{P(D)} = \frac{\prod_{i=1}^5 P(Y = y_i | T_j, X = x_i) \cdot P(T_j)}{P(D)}$$

Evident, la compararea propriu-zisă a probabilităților $P(T_j \mid D)$ cu $j = 0, 1, 2$, nu vom avea nevoie de numitorul $P(D)$. Așadar,

- pentru 0 noduri de test:

După cum s-a precizat și în enunț, acest arbore va avea $P(Y = 1) = 3/5$ și $P(Y = 0) = 2/5$. Deci

$$P(T_0 \mid D) \propto \left(\frac{3}{5}\right)^3 \cdot \left(\frac{2}{5}\right)^2 \cdot \left(\frac{1}{4}\right)^0 = \frac{3^3 \cdot 2^2}{5^5} = \frac{108}{3125} = 0.0336.$$

- pentru un nod de test:

Acest arbore va avea probabilitățile: $P(Y = 1) = 1$ dacă $X < 2.5$, și $P(Y = 1) = 1/3$ dacă $X \geq 2.5$. Prin urmare,

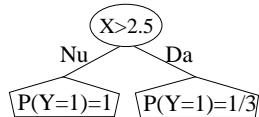
$$P(T_1 \mid D) \propto 1^2 \cdot \left(\frac{2}{3}\right)^2 \cdot \frac{1}{3} \cdot \left(\frac{1}{4}\right)^1 = \frac{1}{27} = 0.037$$

- pentru două noduri de test:

Este vorba de arborele construit la punctul c . Aceasta clasifică perfect datele, dar

$$P(T_2) \propto \left(\frac{1}{4}\right)^4 \Rightarrow P(T_2 \mid D) \propto 1 \cdot \left(\frac{1}{4}\right)^4 = \frac{1}{256} = 0.0039$$

Probabilitatea a posteriori maximă o are arborele T_1 , deci acesta va fi arborele MAP. Reprezentarea grafică a acestui arbore este:



2.1.2 Algoritmii Bayes Naiv și Bayes Optimal

5.

(Algoritmul Bayes Naiv: aplicare; comparație cu estimarea MLE)

prelucrare de Liviu Ciortuz, după CMU, 2004 fall, T. Mitchell Z. Bar-Joseph, final exam, pr. 2

Fie setul de date alăturat, cu trei variabile booleene de intrare a, b, c și o variabilă booleană de ieșire K .

a. Estimați probabilitățile $P(K = 1 \mid a = 1, b = 1)$ și $P(K = 1 \mid a = 1, b = 1, c = 0)$ în sensul verosimilității maxime (engl., Maximum Likelihood Estimation, MLE).

b. Cum clasifică algoritmul Bayes Naiv instanța $(a = 1, b = 1)$? Dar instanța $(a = 1, b = 1, c = 0)$?

a	b	c	K
1	0	1	1
1	1	1	1
0	1	1	0
1	1	0	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	1	0

Răspuns:

a. $P(K = 1 | a = 1, b = 1) = \frac{1}{2}$, fiindcă avem două instanțe în care ambele atrbute a, b sunt adevărate, dintre care una este clasificată $K = 1$, iar cealaltă $K = 0$.

$P(K = 1 | a = 1, b = 1, c = 0) = 0$, fiindcă există o singură instanță cu $a = 1, b = 1, c = 0$ în setul de antrenament și ea este clasificată $K = 0$.

b. Cazul $(a = 1, b = 1)$:

$$\begin{aligned}\hat{k}_{MAP} &= \operatorname{argmax}_{k \in \{0,1\}} P(K = k | a = 1, b = 1) = \\ &= \operatorname{argmax}_{k \in \{0,1\}} \frac{P(a = 1, b = 1 | K = k) \cdot P(K = k)}{P(a = 1, b = 1)} = \\ &= \operatorname{argmax}_{k \in \{0,1\}} P(a = 1, b = 1 | K = k) \cdot P(K = k) = \\ &= \operatorname{argmax}_{k \in \{0,1\}} P(a = 1 | K = k) \cdot P(b = 1 | K = k) \cdot P(K = k)\end{aligned}$$

Avem:

$$p_0 = P(a = 1 | K = 0) \cdot P(b = 1 | K = 0) \cdot P(K = 0) = \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{4}{8} = \frac{1}{8}$$

$$p_1 = P(a = 1 | K = 1) \cdot P(b = 1 | K = 1) \cdot P(K = 1) = \frac{2}{4} \cdot \frac{1}{4} \cdot \frac{4}{8} = \frac{1}{16}$$

Prin urmare, $p_0 > p_1$. Așadar, clasificatorul Bayes Naiv va prezice $K = 0$ pentru instanța $(a = 1, b = 1)$ cu probabilitatea

$$\begin{aligned}P(K = 0 | a = 1, b = 1) &= \frac{P(a = 1, b = 1 | K = 0) \cdot P(K = 0)}{P(a = 1, b = 1 | K = 0) \cdot P(K = 0) + P(a = 1, b = 1 | K = 1) \cdot P(K = 1)} \\ &= \frac{p_0}{p_0 + p_1} = \frac{\frac{1}{8}}{\frac{1}{8} + \frac{1}{16}} = \frac{2}{3}\end{aligned}$$

Observație: Se constată că probabilitatea $P(K = 1 | a = 1, b = 1)$ estimată în sensul MLE (adică, $1/2$; vedeti punctul a) diferă de probabilitatea (*a posteriori*) cu care algoritmul Bayes Naiv clasifică instanța $(a = 1, b = 1)$ ca aparținând clasei $K = 1$ (adică, $1/3$). Explicația rezidă în faptul că presupozitia de independentă condițională asumată de către algoritmul Bayes Naiv nu este validă. Într-adevăr, $P(a = 0 | K = 1) = 2/4$ și $P(a = 0 | b = 1, K = 1) = 0$, deci $P(a = 0 | K = 1) \neq P(a = 0 | b = 1, K = 1)$.

Cazul $(a = 1, b = 1, c = 0)$:

$$\begin{aligned}\hat{k}_{MAP} &= \operatorname{argmax}_{k \in \{0,1\}} P(K = k | a = 1, b = 1, c = 0) = \\ &= \operatorname{argmax}_{k \in \{0,1\}} \frac{P(a = 1, b = 1, c = 0 | K = k) \cdot P(K = k)}{P(a = 1, b = 1, c = 0)} = \\ &= \operatorname{argmax}_{k \in \{0,1\}} P(a = 1, b = 1, c = 0 | K = k) \cdot P(K = k) = \\ &= \operatorname{argmax}_{k \in \{0,1\}} P(a = 1 | K = k) \cdot P(b = 1 | K = k) \cdot P(c = 0 | K = k) \cdot P(K = k)\end{aligned}$$

Avem:

$$\begin{aligned}
 p_0 &= P(a = 1|K = 0) \cdot P(b = 1|K = 0) \cdot P(c = 0|K = 0) \cdot P(K = 0) = \\
 &= \frac{2}{4} \cdot \frac{2}{4} \cdot \frac{1}{4} \cdot \frac{4}{8} = \frac{1}{32} \\
 p_1 &= P(a = 1|K = 1) \cdot P(b = 1|K = 1) \cdot P(c = 0|K = 1) \cdot P(K = 1) = \\
 &= \frac{2}{4} \cdot \frac{1}{4} \cdot \frac{2}{4} \cdot \frac{4}{8} = \frac{1}{32}
 \end{aligned}$$

Întrucât $p_0 = p_1$, clasificatorul Bayes Naiv va prezice $K = 0$ sau $K = 1$ cu aceeași probabilitate ($1/2$).

Notă: Observația de mai sus este valabilă și pentru cazul $(a = 1, b = 1, c = 0)$.

6.

(Algoritmul Bayes Naiv: aplicație la filtrarea emailurilor spam)

■ CMU, 2009 spring, Ziv Bar-Joseph, midterm, pr. 2

Circa 2/3 dintre emailurile tale sunt spam, aşadar te-ai decis să descarci de pe internet un filtru spam open-source care utilizează un clasificator Bayes Naiv.

Presupunem că ai strâns următoarele emailuri spam și non-spam (engl., regular), și de asemenea că doar trei cuvinte sunt discriminative pentru această clasificare, deci fiecare email este reprezentat ca un vector de 3 componente binare, fiecare dintre ele indicând dacă respectivul cuvânt este conținut (sau nu) în email.

'study'	'free'	'money'	Category	count
1	0	0	Regular	1
0	0	1	Regular	1
1	0	0	Regular	1
1	1	0	Regular	1
0	1	0	Spam	4
0	1	1	Spam	4

a. Descoperi că filtrul spam open-source folosește o probabilitate a priori $P(\text{spam}) = 0.1$. Explică în mod succint de ce crezi că această alegere este rezonabilă.

b. Calculează următorii parametri ai modelului prin metoda estimării de verosimilitate maximă (MLE), folosind netezire (engl., smoothing) de tip “add-one” (regula lui Laplace).

$$P(\text{study}|\text{spam}) =$$

$$P(\text{study}|\text{regular}) =$$

$$P(\text{free}|\text{spam}) =$$

$$P(\text{free}|\text{regular}) =$$

$$P(\text{money}|\text{spam}) =$$

$$P(\text{money}|\text{regular}) =$$

- c. Folosind probabilitatea a priori și probabilitățile condiționate de mai sus, calculează probabilitatea ca mesajul s = “money for psychology study” să fie spam, adică $P(\text{spam} | s)$.
- d. Care ar trebui să fie valoarea probabilității a priori $P(\text{spam})$ în cazul în care dorim ca mesajul de mai sus să aibă aceeași probabilitate de fi spam respectiv non-spam (i.e., el va fi clasificat ca spam cu probabilitatea 0.5)?

Răspuns:

- a. Diferența dintre $P_{MLE}(\text{Category}|\text{Spam}) = 2/3$ și probabilitatea a priori indicată în enunț (0.1) se explică prin faptul că se preferă trecerea prin filtru a unor emailuri spam, decât să fie marcate ca spam unele emailuri non-spam și astfel să nu ajungă în Inbox.
- b. Dacă nu am folosi regula de tip “add-one” a lui Laplace (pentru „netezirea“ probabilităților), parametrii modelului ar avea valorile (obținute prin metoda estimării de verosimilitate maximă – MLE) care apar mai jos în partea stângă. Folosind regula lui Laplace, parametrii primesc valorile indicate în partea dreaptă. Observați că aparițiile lui 2 de la numitorul fracțiilor corespund numărului de valori pentru fiecare dintre atrbutele / variabilele de intrare.

$$\begin{array}{ll}
 P(\text{study}|\text{spam}) = \frac{0}{8} = 0 & P(\text{study}|\text{spam}) \stackrel{\text{Laplace}}{=} \frac{0+1}{8+2} = \frac{1}{10} \\
 P(\text{study}|\text{regular}) = \frac{3}{4} & P(\text{study}|\text{regular}) \stackrel{\text{Laplace}}{=} \frac{3+1}{4+2} = \frac{2}{3} \\
 P(\text{free}|\text{spam}) = \frac{8}{8} = 1 & P(\text{free}|\text{spam}) \stackrel{\text{Laplace}}{=} \frac{8+1}{8+2} = \frac{9}{10} \\
 P(\text{free}|\text{regular}) = \frac{1}{4} & P(\text{free}|\text{regular}) \stackrel{\text{Laplace}}{=} \frac{1+1}{4+2} = \frac{1}{3} \\
 P(\text{money}|\text{spam}) = \frac{4}{8} = \frac{1}{2} & P(\text{money}|\text{spam}) \stackrel{\text{Laplace}}{=} \frac{4+1}{8+2} = \frac{1}{2} \\
 P(\text{money}|\text{regular}) = \frac{1}{4} & P(\text{money}|\text{regular}) \stackrel{\text{Laplace}}{=} \frac{1+1}{4+2} = \frac{1}{3}
 \end{array}$$

- c. Avem mesajul s = “money for psychology study”, deci trebuie să calculăm $P(\text{spam} | s) = P(\text{spam} | \text{study}, \neg\text{free}, \text{money})$.

$$\begin{aligned}
 P(\text{spam} | s) &\stackrel{F. Bayes}{=} \\
 &= \frac{P(\text{study}, \neg\text{free}, \text{money} | \text{spam}) \cdot P(\text{spam})}{P(\text{study}, \neg\text{free}, \text{money} | \text{spam})P(\text{spam}) + P(\text{study}, \neg\text{free}, \text{money} | \text{reg})P(\text{reg})}
 \end{aligned}$$

Calculăm probabilitățile folosind ipoteza de independență condițională:

$$P(\text{study}, \neg\text{free}, \text{money} | \text{spam}) = P(\text{study} | \text{spam}) \cdot P(\neg\text{free} | \text{spam}) \cdot P(\text{money} | \text{spam})$$

$$= \frac{1}{10} \cdot \frac{1}{10} \cdot \frac{1}{2} = \frac{1}{200}$$

$$P(\text{study}, \neg\text{free}, \text{money} | \text{reg}) = P(\text{study} | \text{reg}) \cdot P(\neg\text{free} | \text{reg}) \cdot P(\text{money} | \text{reg})$$

$$= \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{27}$$

$$\text{Înlocuind valorile în formulă, obținem: } P(\text{spam} | s) = \frac{\frac{1}{200} \cdot \frac{1}{10}}{\frac{1}{200} \cdot \frac{1}{10} + \frac{4}{27} \cdot \frac{9}{10}} \approx 0.0037$$

Aceasta este o probabilitate mică. Se observă însă că dacă nu am fi folosit regula lui Laplace, probabilitatea ca emailul s să fie spam ar fi fost 0. Aceasta se datorează faptului că niciunul dintre emailurile spam din datele de antrenament nu conține cuvântul *study*, care apare însă în emailul s .

d. Dacă notăm cu p probabilitatea a priori cerută, $P(\text{spam})$, știind că $P(\text{spam} | s) = 0.5$, putem scrie:

$$\begin{aligned} 0.5 &= \frac{\frac{1}{200} \cdot p}{\frac{1}{200} \cdot p + \frac{4}{27} \cdot (1-p)} \Leftrightarrow \frac{1}{2} = \frac{\frac{p}{200}}{\frac{p}{200} + \frac{4}{27} - \frac{4p}{27}} \Leftrightarrow \frac{2p}{200} = \frac{p}{200} + \frac{4}{27} - \frac{4p}{27} \\ &\Leftrightarrow 54p = 27p + 800 - 800p \Leftrightarrow p = \frac{800}{827} \approx 0.9673 \end{aligned}$$

7. (Algoritmul Bayes Naiv și algoritmul Bayes Optimal;
comparație relativ la numărului de parametri)

■ CMU, 2008 fall, Eric Xing, HW1, pr. 2

Fie următoarea problemă de clasificare:

X_1 și X_2 sunt variabile aleatoare observabile, Y este eticheta clasei asignate fiecărei instanțe observate, conform tabelului alăturat.

În acest exercițiu veți compara rezultatele care se obțin în urma antrenării pe acest set de date de către doi algoritmi de clasificare: Bayes Naiv și Bayes Comun (engl., Joint Bayes), care este numit adeseori și *Bayes Optimal* (engl., Optimal Bayes). (În cele ce urmează, vom opta pentru cea de-a doua denumire.)

X_1	X_2	Y	Nr. apariții
0	0	0	2
0	0	1	18
1	0	0	4
1	0	1	1
0	1	0	4
0	1	1	1
1	1	0	2
1	1	1	18

Comentariu: Pentru cel de-al doilea algoritm, denumirea *Bayes Comun* se datorează faptului că acest clasificator lucrează cu distribuția comună a variabilelor / atributelor de intrare, în vreme ce denumirea *Bayes Optimal* este justificată de faptul că nicio altă distribuție probabilistă asupra datelor de intrare nu poate conduce la o eroare medie la clasificare mai mică decât cea obținută de către acest clasificator. (În cele ce urmează, vom opta pentru cea de-a două denumire.)

- Clasificați instanța $X_1 = 0, X_2 = 0$ folosind clasificatorul Bayes Naiv.
- Clasificați instanța $X_1 = 0, X_2 = 0$ folosind clasificatorul Bayes Optimal.
- Notăm cu P_{NB} și respectiv P_{JB} valoarea probabilității $P(Y = 1 | X_1 = 0, X_2 = 0)$ calculate pentru clasificatorul Bayes Naiv, respectiv pentru clasificatorul Bayes Optimal. De ce diferă cele două valori? Sugestie: Calculați $P(X_1, X_2 | Y)$.
- Care ar fi situația pentru P_{NB} și P_{JB} de la întrebarea precedentă în situația în care datele observate ar proveni din tabelul următor?

X_1	X_2	Y	Nr. apariții
0	0	0	3
0	0	1	9
1	0	0	3
1	0	1	9
0	1	0	3
0	1	1	9
1	1	0	3
1	1	1	9

e. De câți parametri independenți (i.e., probabilități estimate) este nevoie în total pentru a construi clasificatorul Bayes Naiv? Dar în cazul clasificatorului Bayes Optimal?

Răspundeți la aceste întrebări și în cazul general, cînd se folosesc n variabile binare observate. Comentați rezultatul.

Răspuns:

a. Clasificatorul Bayes Naiv face predicția pentru valoarea lui Y după formula de mai jos:

$$\hat{y}_{NB} = \operatorname{argmax}_{y \in \{0,1\}} P(X_1 = 0|Y = y) \cdot P(X_2 = 0|Y = y) \cdot P(Y = y)$$

Avem:

$$p_0 \stackrel{\text{not.}}{=} P(X_1 = 0|Y = 0) \cdot P(X_2 = 0|Y = 0) \cdot P(Y = 0) \\ \stackrel{\text{MLE}}{=} \frac{6}{12} \cdot \frac{6}{12} \cdot \frac{12}{50} = \frac{3}{50} = \frac{6}{100}$$

$$p_1 \stackrel{\text{not.}}{=} P(X_1 = 0|Y = 1) \cdot P(X_2 = 0|Y = 1) \cdot P(Y = 1) \\ \stackrel{\text{MLE}}{=} \frac{19}{38} \cdot \frac{19}{38} \cdot \frac{38}{50} = \frac{19}{100}$$

Întrucât $p_0 < p_1$, clasificatorul Bayes Naiv va prezice $Y = 1$ pentru instanța $(X_1 = 0, X_2 = 0)$.

b. Deoarece clasificatorul Bayes Optimal nu lucrează cu presupunerea de independență condițională a atributelor de intrare în raport cu atributul de ieșire, el va face predicția folosind formula de mai jos:

$$\hat{y}_{JB} = \operatorname{argmax}_{y \in \{0,1\}} P(X_1 = 0, X_2 = 0|Y = y) \cdot P(Y = y)$$

Probabilitățile din formulă sunt, ca și în cazul clasificatorului Bayes Naiv, cele estimate din distribuția datelor de antrenament.

Așadar, avem:

$$p'_0 \stackrel{\text{not.}}{=} P(X_1 = 0, X_2 = 0 | Y = 0) \cdot P(Y = 0) \stackrel{\text{MLE}}{=} \frac{2}{12} \cdot \frac{12}{50} = \frac{2}{50}$$

$$p'_1 \stackrel{\text{not.}}{=} P(X_1 = 0, X_2 = 0 | Y = 1) \cdot P(Y = 1) \stackrel{\text{MLE}}{=} \frac{18}{38} \cdot \frac{38}{50} = \frac{18}{50}$$

Fiindcă $p'_0 < p'_1$, clasificatorul Bayes Optimal va prezice tot $Y = 1$.

c. Vom calcula cele două valori pentru $P(Y = 1 | X_1 = 0, X_2 = 0)$:

$$\begin{aligned}
 P_{NB} &\stackrel{\text{not.}}{=} P(Y = 1 | X_1 = 0, X_2 = 0) \\
 &\stackrel{F. Bayes}{=} \frac{P(X_1 = 0, X_2 = 0 | Y = 1) \cdot P(Y = 1)}{P(X_1 = 0, X_2 = 0 | Y = 1)P(Y = 1) + P(X_1 = 0, X_2 = 0 | Y = 0)P(Y = 0)} \\
 &\stackrel{\text{indep. cdt.}}{=} \frac{P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1)}{P(X_1 = 0 | Y = 0) \cdot P(X_2 = 0 | Y = 0) \cdot P(Y = 0) + P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1)} \\
 &= \frac{\frac{p_1}{p_0 + p_1}}{\frac{6}{100} + \frac{19}{100}} = \frac{\frac{19}{100}}{\frac{6}{100} + \frac{19}{100}} = \frac{19}{25} \\
 P_{JB} &\stackrel{\text{not.}}{=} P(Y = 1 | X_1 = 0, X_2 = 0) \\
 &\stackrel{F. Bayes}{=} \frac{P(X_1 = 0, X_2 = 0 | Y = 1) \cdot P(Y = 1)}{P(X_1 = 0, X_2 = 0 | Y = 1)P(Y = 1) + P(X_1 = 0, X_2 = 0 | Y = 0)P(Y = 0)} \\
 &= \frac{\frac{p'_1}{p'_0 + p'_1}}{\frac{2}{50} + \frac{18}{50}} = \frac{\frac{18}{50}}{\frac{2}{50} + \frac{18}{50}} = \frac{18}{20}
 \end{aligned}$$

Cele două valori diferă deoarece presupunerea de independentă condițională a variabilelor X_1 și X_2 în raport cu Y făcută de clasificatorul Bayes Naiv este falsă. Acest lucru se poate vedea ușor din valorile estimate pentru $P(X_1 = 0, X_2 = 0 | Y = 0)$, $P(X_1 = 0 | Y = 0)$ și $P(X_2 = 0 | Y = 0)$:

$$\begin{aligned}
 &P(X_1 = 0, X_2 = 0 | Y = 0) \stackrel{MLE}{=} \frac{2}{12} \\
 &P(X_1 = 0 | Y = 0) \cdot P(X_2 = 0 | Y = 0) \stackrel{MLE}{=} \frac{6}{12} \cdot \frac{6}{12} = \frac{1}{4} \\
 \Rightarrow &P(X_1 = 0, X_2 = 0 | Y = 0) \neq P(X_1 = 0 | Y = 0) \cdot P(X_2 = 0 | Y = 0) \Rightarrow \\
 \Rightarrow &P(X_1, X_2 | Y) \neq P(X_1 | Y) \cdot P(X_2 | Y)
 \end{aligned}$$

Așadar, variabilele X_1 și X_2 nu sunt independente condițional în raport cu variabila de ieșire Y .

d. Vom calcula cele două valori pentru $P(Y = 1 | X_1 = 0, X_2 = 0)$ în cazul noilor date:

$$\begin{aligned}
 P_{NB} &= P(Y = 1 | X_1 = 0, X_2 = 0) \\
 &= \frac{P(X_1 = 0, X_2 = 0 | Y = 1) \cdot P(Y = 1)}{P(X_1 = 0, X_2 = 0 | Y = 1)P(Y = 1) + P(X_1 = 0, X_2 = 0 | Y = 0)P(Y = 0)} \\
 &= \frac{P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1)}{P(X_1 = 0 | Y = 0) \cdot P(X_2 = 0 | Y = 0) \cdot P(Y = 0) + P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1)} \\
 &= \frac{\frac{18}{36} \cdot \frac{18}{36} \cdot \frac{36}{48}}{\frac{6}{12} \cdot \frac{6}{12} \cdot \frac{12}{48} + \frac{18}{36} \cdot \frac{18}{36} \cdot \frac{36}{48}} = \frac{\frac{9}{48}}{\frac{3}{12} + \frac{9}{48}} = \frac{\frac{9}{48}}{\frac{12}{48}} = \frac{3}{4}
 \end{aligned}$$

$$\begin{aligned}
P_{JB} &= P(Y = 1 \mid X_1 = 0, X_2 = 0) \\
&= \frac{P(X_1 = 0, X_2 = 0 \mid Y = 1) \cdot P(Y = 1)}{P(X_1 = 0, X_2 = 0 \mid Y = 1)P(Y = 1) + P(X_1 = 0, X_2 = 0 \mid Y = 0)P(Y = 0)} \\
&= \frac{\frac{9}{36} \cdot \frac{36}{48}}{\frac{9}{36} \cdot \frac{36}{48} + \frac{9}{36} \cdot \frac{36}{48}} = \frac{\frac{9}{48}}{\frac{9}{48} + \frac{9}{48}} = \frac{9}{12} = \frac{3}{4}
\end{aligned}$$

Așadar, în acest caz avem $P_{NB} = P_{JB}$.

De fapt, se poate constata ușor că în cazul distribuției probabiliste date la acest punct al problemei se verifică independența condițională a variabilelor X_1 și X_2 în raport cu Y . Prin urmare, predicțiile făcute de cei doi clasificatori, Bayes Naiv și Bayes Optimal, vor coincide întotdeauna.

e. În contextul problemei noastre, clasificatorul Bayes Naiv are nevoie de estimările următoarelor probabilități:

$$\begin{aligned}
P(Y = 0) &\Rightarrow P(Y = 1) = 1 - P(Y = 0) \\
P(X_1 = 0 \mid Y = 0) &\Rightarrow P(X_1 = 1 \mid Y = 0) = 1 - P(X_1 = 0 \mid Y = 0) \\
P(X_1 = 0 \mid Y = 1) &\Rightarrow P(X_1 = 1 \mid Y = 1) = 1 - P(X_1 = 0 \mid Y = 1) \\
P(X_2 = 0 \mid Y = 0) &\Rightarrow P(X_2 = 1 \mid Y = 0) = 1 - P(X_2 = 0 \mid Y = 0) \\
P(X_2 = 0 \mid Y = 1) &\Rightarrow P(X_2 = 1 \mid Y = 1) = 1 - P(X_2 = 0 \mid Y = 1)
\end{aligned}$$

Avem nevoie, prin urmare, doar de 5 valori pentru a construi complet clasificatorul Bayes Naiv.

În cazul general, dacă avem n variabile de intrare, avem nevoie de estimări pentru probabilitățile $P(Y)$, $P(X_i \mid Y)$ și $P(X_i \mid \neg Y)$ pentru $i = 1, \dots, n$, deci $2n + 1$ valori.

Pentru clasificatorul Bayes Optimal avem nevoie de:

$$\begin{aligned}
P(Y = 0) &\quad P(Y = 1) = 1 - P(Y = 0) \\
P(X_1 = 0, X_2 = 0 \mid Y = 0) &\quad P(X_1 = 1, X_2 = 1 \mid Y = 0) \text{ se poate determina din} \\
P(X_1 = 0, X_2 = 1 \mid Y = 0) &\quad \text{celelalte 3 valori, aşa cum vom arăta mai jos.} \\
P(X_1 = 1, X_2 = 0 \mid Y = 0) & \\
P(X_1 = 0, X_2 = 0 \mid Y = 1) &\quad \text{Similar, } P(X_1 = 1, X_2 = 1 \mid Y = 1) \text{ se poate deter-} \\
P(X_1 = 0, X_2 = 1 \mid Y = 1) &\quad \text{mina din celelalte 3 valori.} \\
P(X_1 = 1, X_2 = 0 \mid Y = 1) &
\end{aligned}$$

Notăm evenimentul $X_1 = 0$ cu A , $X_2 = 0$ cu B și $Y = 0$ cu C . Știm că:

$$\begin{aligned}
\Omega &= (A \wedge B) \vee (\neg A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge \neg B) \Rightarrow \\
\Omega \wedge C &= ((A \wedge B) \vee (\neg A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge \neg B)) \wedge C \Rightarrow \\
C &= ((A \wedge B) \wedge C) \vee ((\neg A \wedge B) \wedge C) \vee ((A \wedge \neg B) \wedge C) \vee ((\neg A \wedge \neg B) \wedge C)
\end{aligned}$$

De asemenea, deoarece toate evenimentele din partea dreaptă a egalității sunt disjuncte două câte două, putem scrie egalitatea de mai sus și cu probabilități:

$$\begin{aligned}
P(C) &= P((A \wedge B) \wedge C) + P((\neg A \wedge B) \wedge C) + P((A \wedge \neg B) \wedge C) + P((\neg A \wedge \neg B) \wedge C) \\
&\Rightarrow 1 = \frac{P((A \wedge B) \wedge C)}{P(C)} + \frac{P((\neg A \wedge B) \wedge C)}{P(C)} + \frac{P((A \wedge \neg B) \wedge C)}{P(C)} + \frac{P((\neg A \wedge \neg B) \wedge C)}{P(C)} \\
&\Rightarrow 1 = P(A, B \mid C) + P(\neg A, B \mid C) + P(A, \neg B \mid C) + P(\neg A, \neg B \mid C) \\
&\Rightarrow P(\neg A, \neg B \mid C) = 1 - (P(A, B \mid C) + P(\neg A, B \mid C) + P(A, \neg B \mid C))
\end{aligned}$$

Prin urmare, știind 3 valori o putem afla și pe a patra. La fel și pentru $\neg C$.

Pentru a avea un clasificator Bayes Optimal complet avem deci nevoie de 7 valori diferite. În cazul general, pentru n variabile de intrare, avem nevoie de probabilitățile $P(Y)$, $P(\tilde{X}_1, \dots, \tilde{X}_n | Y)$ și $P(\tilde{X}_1, \dots, \tilde{X}_n | \neg Y)$, unde

$$\tilde{X}_i \in \{X_i, \neg X_i\} \quad \forall i \in \overline{1, n} \quad \text{și} \quad (\tilde{X}_1, \dots, \tilde{X}_n) \neq (\neg X_1, \dots, \neg X_n).$$

Avem, deci, $2(2^n - 1) + 1 = 2^{n+1} - 1$ valori.

Se observă că algoritmul Bayes Naiv folosește un număr liniar de parametri (în raport cu n , numărul de atrbute de intrare), în vreme ce algoritmul Bayes Optimal folosește un număr exponențial de parametri (în raport cu același n).

8. (Calculul parametrilor pentru clasificatorul Bayes Naiv
pornind de la distribuția comună a variabilelor;
comparație între algoritmii Bayes Naiv și Bayes Optimal)
■ prelucrare de L. Ciortuz, după
CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm, pr. 2.1

Fie P o distribuție de probabilitate comună peste variabilele aleatoare booleene x_1, x_2 și y .

- a. Exprimăți $P(y = 0 | x_1, x_2)$ în funcție de $P(x_1, x_2, y = 0)$ și $P(x_1, x_2, y = 1)$.

x_1	x_2	y	$P(x_1, x_2, y)$
0	0	0	0.15
0	0	1	0.25
0	1	0	0.05
0	1	1	0.08
1	0	0	0.10
1	0	1	0.02
1	1	0	0.20
1	1	1	0.15

În cele ce urmează vom considera distribuția P , definită conform tabelului alăturat.

- b. Pornind de la această distribuție comună, calculați probabilitățile necesare pentru clasificare bayesiană naivă. Completăți tabelele de mai jos:

y	$P(y)$	$P(x_1 y)$	$x_1 = 0$	$x_1 = 1$	$P(x_2 y)$	$x_2 = 0$	$x_2 = 1$
$y = 0$		$y = 0$			$y = 0$		
$y = 1$		$y = 1$			$y = 1$		

- c. Cât este probabilitatea $P(y = 1 | x_1 = 1, x_2 = 0)$ calculată de către clasificatorul Bayes Naiv?

- d. Cât este probabilitatea $P(y = 1 | x_1 = 1, x_2 = 0)$ calculată de către clasificatorul Bayes Optimal? (Vă readucem aminte că acest algoritm este similar cu algoritmul Bayes Naiv, însă nu folosește presupozitia de independentă condițională a atrbutorilor.)

- e. Răspunsurile la precedentele două întrebări ar trebui să fie diferite. Care este motivul? Justificați.

Răspuns:

a. Aplicând definiția probabilității condiționate și apoi proprietatea de aditivitate numărabilă din definiția funcției de probabilitate, obținem:

$$P(y = 0 | x_1, x_2) = \frac{P(x_1, x_2, y = 0)}{P(x_1, x_2)} = \frac{P(x_1, x_2, y = 0)}{P(x_1, x_2, y = 0) + P(x_1, x_2, y = 1)}$$

b. Probabilitățile cerute în enunț sunt $P(x_1 | y)$, $P(x_2 | y)$ și $P(y)$.

$P(y)$ este o *probabilitate marginală* a distribuției comune, deci se calculează astfel:

$$\begin{aligned} P(y = 0) &= P(x_1 = 0, x_2 = 0, y = 0) + P(x_1 = 0, x_2 = 1, y = 0) + \\ &\quad + P(x_1 = 1, x_2 = 0, y = 0) + P(x_1 = 1, x_2 = 1, y = 0) \\ &= 0.15 + 0.05 + 0.1 + 0.2 = 0.5 \\ P(y = 1) &= 1 - P(y = 0) = 0.5 \end{aligned}$$

$P(x_1 | y)$ se calculează folosind din nou definiția probabilității condiționate și formula probabilității totale:

$$P(x_1 = 0 | y = 0) = \frac{P(x_1 = 0, y = 0)}{P(y = 0)} = \frac{P(x_1 = 0, y = 0)}{P(x_1 = 0, y = 0) + P(x_1 = 1, y = 0)}$$

Probabilitățile implicate în formulă sunt probabilități marginale ale distribuției comune și se calculează astfel:

$$\begin{aligned} P(x_1 = 0, y = 0) &= P(x_1 = 0, x_2 = 0, y = 0) + P(x_1 = 0, x_2 = 1, y = 0) = 0.15 + 0.05 = 0.2 \\ P(x_1 = 1, y = 0) &= P(x_1 = 1, x_2 = 0, y = 0) + P(x_1 = 1, x_2 = 1, y = 0) = 0.1 + 0.2 = 0.3 \end{aligned}$$

Prin urmare,

$$P(x_1 = 0 | y = 0) = \frac{0.2}{0.2 + 0.3} = 0.4,$$

iar

$$P(x_1 = 1 | y = 0) = 1 - P(x_1 = 0 | y = 0) = 0.6$$

Analog se calculează și celelalte probabilități corespunzătoare lui $P(x_1 | y)$:

$$\begin{aligned} P(x_1 = 0 | y = 1) &= \frac{P(x_1 = 0, y = 1)}{P(y = 1)} = \frac{0.25 + 0.08}{0.5} = 0.66 \\ P(x_1 = 1 | y = 1) &= 1 - P(x_1 = 0 | y = 1) = 0.34 \end{aligned}$$

$P(x_2 | y)$ se calculează în același mod.

Putem completa tabelele următoare cu valorile numerice ale probabilităților calculate la acest punct:

y	$P(y)$	$P(x_1 y)$	$x_1 = 0$	$x_1 = 1$	$P(x_2 y)$	$x_2 = 0$	$x_2 = 1$
$y = 0$	0.5	$y = 0$	0.40	0.60	$y = 0$	0.50	0.50
$y = 1$	0.5	$y = 1$	0.66	0.34	$y = 1$	0.54	0.46

c. Clasificatorul Bayes Naiv face presupunerea de *independență condițională* a variabilelor, deci:

$$P(y = 1 | x_1 = 1, x_2 = 0) \stackrel{\text{Bayes}}{=} \frac{P(x_1 = 1, x_2 = 0 | y = 1) \cdot P(y = 1)}{P(x_1 = 1, x_2 = 0)}$$

$$\begin{aligned}
&= \frac{P(x_1 = 1, x_2 = 0 | y = 1) \cdot P(y = 1)}{P(x_1 = 1, x_2 = 0 | y = 1)P(y = 1) + P(x_1 = 1, x_2 = 0 | y = 0)P(y = 0)} \stackrel{\text{indep. cdt.}}{=} \\
&= \frac{P(x_1 = 1 | y = 1) \cdot P(x_2 = 0 | y = 1) \cdot P(y = 1)}{P(x_1 = 1 | y = 1)P(x_2 = 0 | y = 1)P(y = 1) + P(x_1 = 1 | y = 0)P(x_2 = 0 | y = 0)P(y = 0)} \\
&= \frac{0.34 \cdot 0.54 \cdot 0.5}{0.34 \cdot 0.54 \cdot 0.5 + 0.6 \cdot 0.5 \cdot 0.5} \approx 0.3796
\end{aligned}$$

d. Clasificatorul Bayes Optimal nu face niciun fel de presupunere, deci folosind o formulă similară cu cea obținută la punctul a, vom avea:

$$\begin{aligned}
P(y = 1 | x_1 = 1, x_2 = 0) &= \frac{P(x_1 = 1, x_2 = 0, y = 1)}{P(x_1 = 1, x_2 = 0, y = 1) + P(x_1 = 1, x_2 = 0, y = 0)} \\
&= \frac{0.02}{0.02 + 0.1} = 0.1(6).
\end{aligned}$$

e. Valorile calculate de clasificatorul Bayes Naiv și de clasificatorul Bayes Optimal pentru $P(y = 1 | x_1 = 1, x_2 = 0)$ sunt diferite deoarece presupunerea de independentă condițională făcută de clasificatorul Bayes Naiv nu este adevărată. Într-adevăr, se observă că variabilele x_1 și x_2 nu sunt independente condițional în raport cu variabila y :

$$\begin{aligned}
P(x_1 = 1, x_2 = 0 | y = 1) &= \frac{P(x_1 = 1, x_2 = 0, y = 1)}{P(y = 1)} = \frac{0.02}{0.5} = 0.04 \\
P(x_1 = 1 | y = 1) \cdot P(x_2 = 0 | y = 1) &= 0.34 \cdot 0.54 = 0.1836 \neq 0.04
\end{aligned}$$

9.

(Clasificare bayesiană: un caz particular)

CMU, 2010 spring, E. Xing, A. Singh, T. Mitchell, pr. 2.2

Se consideră variabilele aleatoare X_1, X_2, X_3 și X_4 . Aceste variabile sunt independente condițional două câte două în raport cu variabila Y , cu excepția perechii X_3, X_4 . (Așadar, dacă am aplica algoritmul Bayes Naiv, acesta ar produce erori de clasificare.)

Cum am putea modifica regula de decizie a algoritmului Bayes Naiv pentru a ține cont de această particularitate a datelor?

Răspuns:

Întrucât are loc egalitatea

$$\begin{aligned}
\operatorname{argmax}_y P(Y = y | X_1, X_2, X_3, X_4) &= \\
\operatorname{argmax}_y P(X_1 | Y = y) \cdot P(X_2 | Y = y) \cdot P(X_3, X_4 | Y = y) \cdot P(Y = y),
\end{aligned}$$

urmează că regula de decizie a algoritmului Bayes Naiv în cazul dat este:

$$\operatorname{argmax}_y P_{MLE}(X_1 | Y = y) \cdot P_{MLE}(X_2 | Y = y) \cdot P_{MLE}(X_3, X_4 | Y = y) \cdot P_{MLE}(Y = y).$$

Pentru a justifica egalitatea de mai sus, se folosește regula de înlățuire condițională:

$$P(A_1, A_2, A_3 | B) = P(A_3 | B) \cdot P(A_2 | A_3, B) \cdot P(A_1 | A_2, A_3, B),$$

care se demonstrează ușor. Varianta necondițională a regulii de înlățuire este:

$$P(A_1, A_2, A_3) = P(A_3) \cdot P(A_2 | A_3) \cdot P(A_1 | A_2, A_3).$$

Apoi, regula de înlățuire condițională se particularizează pentru evenimentele $A_1 = (X_1 = x_1)$, $A_2 = (X_2 = x_2)$, $A_3 = (X_3 = x_3, X_4 = x_4)$ și $B = (Y = y)$. În fine, se va ține cont că $P(X_1 = x_1 | X_2 = x_2, X_3 = x_3, X_4 = x_4, Y = y) = P(X_1 = x_1 | Y = y)$ și $P(X_2 = x_2 | X_3 = x_3, X_4 = x_4, Y = y) = P(X_2 = x_2 | Y = y)$ datorită proprietății de independentă condițională din enunț.

10.

(Algoritmul Bayes Naiv:
calculul ratei medii a erorii la antrenare)
CMU, 2006 fall, T. Mitchell, E. Xing, midterm, pr. 6

Considerăm o problemă de clasificare binară în care fiecare exemplu X are două attribute binare $X_1, X_2 \in \{0, 1\}$ și eticheta $Y \in \{0, 1\}$. Vom presupune că X_1 și X_2 sunt independente condițional în raport cu Y ,¹¹⁸ și că $P(Y = 0) = P(Y = 1) = 0,5$. De asemenea, probabilitățile condiționate sunt date în tabelele următoare:

$P(X_1 Y)$	$Y = 0$	$Y = 1$	$P(X_2 Y)$	$Y = 0$	$Y = 1$
$X_1 = 0$	0,7	0,2	$X_2 = 0$	0,9	0,5
$X_1 = 1$	0,3	0,8	$X_2 = 1$	0,1	0,5

a. Calculați predicția \hat{Y} făcută de clasificatorul Bayes Naiv pentru fiecare din cele patru combinații posibile de valori ale variabilelor X_1 și X_2 . Completați următorul tabel:

X_1	X_2	$P(X_1, X_2, Y = 0)$	$P(X_1, X_2, Y = 1)$	$\hat{Y}(X_1, X_2)$
0	0	$0,7 \cdot 0,9 \cdot 0,5$	$0,2 \cdot 0,5 \cdot 0,5$	0
0	1			
1	0			
1	1			

b. Presupunând că se folosesc o infinitate de exemple, calculați *rata medie a erorii* (engl., the expected error rate) făcute de acest clasificator *la antrenare*, folosind formula:

$$P(Y \neq \hat{Y}(X_1, X_2)) = \sum_{X_1=0}^1 \sum_{X_2=0}^1 P(X_1, X_2, Y = 1 - \hat{Y}(X_1, X_2))$$

c. Care din următorii doi clasificatori are rata medie a erorii la antrenare mai mică:

- clasificatorul Bayes Naiv care prezice Y având ca input doar X_1 ;
- clasificatorul Bayes Naiv care prezice Y având ca input doar X_2 .

d. Presupunem că definim un nou atribut X_3 , care este o copie a lui X_2 . Care este rata medie a erorii la antrenare a clasificatorului Bayes Naiv care prezice Y folosind toate attributele X_1, X_2, X_3 ? (Se presupune că datele de antrenament sunt în număr infinit.)

e. Explicați de ce rata erorii de la punctul *d* diferă față de cea de la punctul *a*.

Răspuns:

a. Predicția \hat{Y} făcută de clasificatorul Bayes Naiv pentru fiecare din cele patru combinații posibile de valori ale variabilelor X_1 și X_2 este înregistrată în ultima coloană a tabelului de mai jos. Calculele necesare pentru justificare — folosind formula $P(X_1, X_2, Y) =$

¹¹⁸Așadar, în această situație rezultatele algoritmilor Bayes Naiv și Bayes Optimal vor coincide.

$P(X_1, X_2|Y) \cdot P(Y) = P(X_1|Y) \cdot P(X_2|Y) \cdot P(Y)$ — sunt conținute în coloanele a treia și a patra.

X_1	X_2	$P(X_1, X_2, Y = 0)$	$P(X_1, X_2, Y = 1)$	$\hat{Y}(X_1, X_2)$
0	0	$0,7 \cdot 0,9 \cdot 0,5 = 0,315$	$0,2 \cdot 0,5 \cdot 0,5 = \mathbf{0,05}$	0
0	1	$0,7 \cdot 0,1 \cdot 0,5 = \mathbf{0,035}$	$0,2 \cdot 0,5 \cdot 0,5 = 0,05$	1
1	0	$0,3 \cdot 0,9 \cdot 0,5 = \mathbf{0,135}$	$0,8 \cdot 0,5 \cdot 0,5 = 0,2$	1
1	1	$0,3 \cdot 0,1 \cdot 0,5 = \mathbf{0,015}$	$0,8 \cdot 0,5 \cdot 0,5 = 0,2$	1

Observație (1):

În acest tabel putem vedea valorile distribuției comune $P(X_1, X_2, Y)$. Spre deosebire de problema 8, unde distribuția comună era dată iar distribuțiile marginale condiționale erau calculate pornind de la aceasta, aici se procedează invers, ținând cont [și] de presupozitia de independentă condițională.

b. Rata medie a erorii este:

$$\begin{aligned}
 P(Y \neq \hat{Y}(X_1, X_2)) &= \\
 &= \sum_{X_1=0}^1 \sum_{X_2=0}^1 P(X_1, X_2, Y = 1 - \hat{Y}(X_1, X_2)) \\
 &= P(X_1 = 0, X_2 = 0, Y = 1 - 0) + P(X_1 = 0, X_2 = 1, Y = 1 - 1) \\
 &\quad + P(X_1 = 1, X_2 = 0, Y = 1 - 1) + P(X_1 = 1, X_2 = 1, Y = 1 - 1) \\
 &= 0,05 + 0,035 + 0,135 + 0,015 = 0,235
 \end{aligned}$$

Pentru justificarea penultimei egalități, vedeți tabelul de la punctul precedent.

c. Făcând prezicerea doar cu X_1 ca atribut de intrare — folosind formula de multiplicare $P(X_1, Y) = P(X_1|Y) \cdot P(Y)$ —, obținem:

X_1	$P(X_1, Y = 0)$	$P(X_1, Y = 1)$	$\hat{Y}_1(X_1, X_2)$
0	$0,7 \cdot 0,5 = 0,35$	$0,2 \cdot 0,5 = \mathbf{0,1}$	0
1	$0,3 \cdot 0,5 = \mathbf{0,15}$	$0,8 \cdot 0,5 = 0,4$	1

Rata medie a erorii în acest caz va fi:

$$\begin{aligned}
 P(Y \neq \hat{Y}_1(X_1, X_2)) &= \\
 &= \sum_{X_1=0}^1 \sum_{X_2=0}^1 P(X_1, X_2, Y = 1 - \hat{Y}_1(X_1, X_2)) \\
 &= P(X_1 = 0, X_2 = 0, Y = 1 - 0) + P(X_1 = 0, X_2 = 1, Y = 1 - 0) \\
 &\quad + P(X_1 = 1, X_2 = 0, Y = 1 - 1) + P(X_1 = 1, X_2 = 1, Y = 1 - 1) \\
 &= 0,05 + 0,05 + 0,135 + 0,015 = 0,1 + 0,15 = 0,25
 \end{aligned}$$

Similar, dacă luăm în considerare doar variabila X_2 , avem:

X_2	$P(X_2, Y = 0)$	$P(X_2, Y = 1)$	$\hat{Y}_2(X_1, X_2)$
0	$0,9 \cdot 0,5 = 0,45$	$0,5 \cdot 0,5 = \mathbf{0,25}$	0
1	$0,1 \cdot 0,5 = \mathbf{0,05}$	$0,5 \cdot 0,5 = 0,25$	1

Acum, rata medie a erorii va fi:

$$\begin{aligned}
 P(Y \neq \hat{Y}_2(X_1, X_2)) &= \\
 &= \sum_{X_1=0}^1 \sum_{X_2=0}^1 P(X_1, X_2, Y = 1 - \hat{Y}_2(X_1, X_2)) \\
 &= P(X_1 = 0, X_2 = 0, Y = 1 - 0) + P(X_1 = 0, X_2 = 1, Y = 1 - 1) \\
 &\quad + P(X_1 = 1, X_2 = 0, Y = 1 - 0) + P(X_1 = 1, X_2 = 1, Y = 1 - 1) \\
 &= 0,05 + 0,035 + 0,2 + 0,015 = 0,25 + 0,05 = 0,3
 \end{aligned}$$

Prin urmare, rata medie a erorii la antrenare este mai mică pentru clasificatorul Bayes Naiv care prezice Y având ca input doar X_1 (decât pornind doar de la X_2).¹¹⁹

Observație (2):

Atât în cazul lui X_1 cât și în cazul lui X_2 , rata medie a erorii (pentru algoritmul Bayes Naiv) putea fi calculată folosind direct probabilitățile din cele două tabele de mai sus. Justificarea ține de faptul că distribuțiile calculate de Bayes Naiv în aceste două tabele sunt distribuții marginale în raport cu distribuția comună (reală!) din tabelul de la punctul a . La punctul d veți vedea că acolo trebuie procedat altfel, fiindcă în cazul respectiv distribuția calculată de către Bayes Naiv nu mai coincide cu distribuția reală a datelor!

d. Clasificatorul Bayes Naiv care prezice valoarea lui Y în funcție de toate cele trei variabilele $X_1, X_2, X_3 = X_2$ va lua deciziile conform tabelului următor:¹²⁰

X_1	X_2	X_3	$P(X_1, X_2, X_3, Y = 0)$	$P(X_1, X_2, X_3, Y = 1)$	$\hat{Y}_3(X_1, X_2)$
0	0	0	$0,7 \cdot 0,9 \cdot 0,9 \cdot 0,5 = 0,2835$	$0,2 \cdot 0,5 \cdot 0,5 \cdot 0,5 = 0,025$	0
0	1	1	$0,7 \cdot 0,1 \cdot 0,1 \cdot 0,5 = 0,0035$	$0,2 \cdot 0,5 \cdot 0,5 \cdot 0,5 = 0,025$	1
1	0	0	$0,3 \cdot 0,9 \cdot 0,9 \cdot 0,5 = 0,1215$	$0,8 \cdot 0,5 \cdot 0,5 \cdot 0,5 = 0,1$	0
1	1	1	$0,3 \cdot 0,1 \cdot 0,1 \cdot 0,5 = 0,0015$	$0,8 \cdot 0,5 \cdot 0,5 \cdot 0,5 = 0,1$	1

Observație (3):

Este util să observați că distribuția de probabilitate comună calculată aici de către algoritmul Bayes Naiv nu mai coincide cu distribuția „reală“ (vedeți tabelul de la punctul a). Rata erorii se calculează în raport cu distribuția „reală“ a datelor, nu cu cea calculată de către Bayes Naiv (deși pentru a identifica situațiile în care $\hat{Y} \neq Y$ se folosește ultimul tabel)!

Rata medie a erorii produse la antrenare va fi:

$$\begin{aligned}
 P(Y \neq \hat{Y}_3(X_1, X_2)) &= \\
 &= \sum_{X_1=0}^1 \sum_{X_2=0}^1 P(X_1, X_2, Y = 1 - \hat{Y}_3(X_1, X_2)) \\
 &= P(X_1 = 0, X_2 = 0, Y = 1 - 0) + P(X_1 = 0, X_2 = 1, Y = 1 - 1) \\
 &\quad + P(X_1 = 1, X_2 = 0, Y = 1 - 0) + P(X_1 = 1, X_2 = 1, Y = 1 - 1) \\
 &= 0,025 + 0,0035 + 0,1 + 0,0015 = 0,13
 \end{aligned}$$

¹¹⁹Însă ambele clasificatori au rata medie a erorii mai mare decât clasificatorul Bayes Naiv care folosește atât X_1 cât și X_2 , ceea ce era de așteptat întrucât în condițiile date Bayes Naiv are același comportament ca și Bayes Optimal.

¹²⁰Este bine de observat că în această situație presupozitia de independentă conditională este încălcată. Așadar, Bayes Naiv nu va mai furniza aceleași rezultate ca Bayes Optimal.

- e. Diferența dintre cele două rate medii ale erorilor care au fost calculate la punctele a și d se datorează faptului că presupunerea de independentă condițională a variabilelor nu este adevărată. Într-adevăr, X_2 nu este independent condițional față de X_3 deoarece cele două variabile au tot timpul valori identice.

Observații importante:

1. Este imediat că atunci când datele satisfac presupoziția de independentă condițională, algoritmii Bayes Naiv și Bayes Optimal produc aceleași rezultate și au aceeași rată medie a erorilor.
2. Se poate arăta ușor că algoritmul Bayes Optimal nu produce în mod neapărat o rată medie a erorilor nulă, chiar dacă lucrează cu distribuția reală a datelor. De exemplu, în condițiile definite inițial de problema noastră, algoritmul Bayes Optimal produce rata medie 0.235, ca și algoritmul Bayes Naiv (vedeți punctul b). Erorile produse de către algoritmul Bayes Optimal se datorează faptului că el aplică operatorul arg max, echivalent cu luarea unui vot majoritar (impus deci minorității).

11. (Cât de naiv / prost este algoritmul Bayes Naiv?)

■ CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW2, pr. 1.2

În mod evident, clasificatorul Bayes Naiv lucrează cu o presupozitie foarte restrictivă (engl., strong presupposition). Însă ne putem întreba dacă acest clasificator nu este totuși destul de folositor chiar și în cazul în care respectiva presupozitie nu este satisfăcută.

În consecință, în acest exercițiu ne propunem să folosim un exemplu simplu pentru a explora limitările algoritmului Bayes Naiv.

Fie X_1 și X_2 variabile aleatoare binare de tip Bernoulli de parametru $p = 0.5$, iar Y o funcție deterministă în raport cu valorile lui X_1 și X_2 , luând valori în mulțimea $\{1, 2\}$.

- a. Definiți Y astfel încât (pe setul de date respectiv) algoritmul Bayes Naiv să aibă rata medie a erorii de 50%.

Pe acest caz, observați cum se coreleză valorile lui X_1 și X_2 când valoarea lui Y este fixată. (Altfel spus, observați cât de (in)dependente sunt în acest context valorile lui X_1 și X_2 , dată fiind valoarea lui Y .)

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

- b. Există în total $2^4 = 16$ moduri în care poate fi definită funcția Y . Însă, datorită simetriei (relativ la valorile lui Y), problema se reduce la doar 4 cazuri,

X_1	X_2	Y									
0	0	1	0	0	1	0	0	1	0	0	1
0	1	1	0	1	1	0	1	2	0	1	2
1	0	1	1	0	1	1	0	1	1	0	2
1	1	1	1	1	2	1	1	2	1	1	1

dintre care un caz corespunde punctului a de mai sus. În fiecare din acele trei cazuri rămase după rezolvarea de la punctul a, arătați că rata erorii înregistrate de algoritmul Bayes Naiv este 0.

Răspuns:

- a. Considerăm Y definit conform tabelului de mai jos.

Observație: Dacă se consideră valoarea lui Y fixată (fie 1, fie 2), atunci putem să stabilim o regulă astfel încât dacă îl cunoaștem pe X_1 să-l determinăm pe X_2 (și invers).¹²¹ Altfel spus, X_1 este unic determinat de X_2 (și invers), dată fiind o valoare fixată a lui Y . Deci condiția de independență condițională este încălcată. Mai mult, în acest caz avem maximul posibil de „dependență” între cele două variabile (în raport cu Y).

Dorim să calculăm rata erorii înregistrate de algoritmul Bayes Naiv pe datele din tabelul de mai sus. Bayes Naiv estimează valoarea lui Y astfel:

$$\hat{y} = \operatorname{argmax}_{y \in \{1,2\}} P(X_1 | Y = y) \cdot P(X_2 | Y = y) \cdot P(Y = y)$$

Pentru $X_1 = 0, X_2 = 0$, algoritmul compară următoarele două valori:

$$\begin{aligned} p_1 &= P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \\ p_2 &= P(X_1 = 0 | Y = 2) \cdot P(X_2 = 0 | Y = 2) \cdot P(Y = 2) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \end{aligned}$$

Cum $p_1 = p_2$, algoritmul va alege una dintre ele cu o probabilitate de 0.5. Deoarece valoarea lui Y din tabel este 1, înseamnă că algoritmul va alege greșit în 50% din cazuri. Pentru celelalte 3 cazuri, $(X_1 = 0, X_2 = 1)$, $(X_1 = 1, X_2 = 0)$ și $(X_1 = 1, X_2 = 1)$, se observă ușor că se obțin de asemenea valori egale, iar algoritmul va alege pentru Y una dintre valorile 1 sau 2 cu o probabilitate de 0.5.

Deci pentru această definiție a lui Y rata erorii este de 50%.

b. Vom calcula rata erorii pentru fiecare dintre cele 3 moduri de definire a lui Y care nu a fost studiat.

Cazul 1: $\begin{array}{ c c c } \hline X_1 & X_2 & Y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}$			Este similar cu cazul: $\begin{array}{ c } \hline Y \\ \hline 2 \\ 2 \\ 2 \\ 2 \\ \hline \end{array}$		
---	--	--	---	--	--

- Pentru $X_1 = 0, X_2 = 0$, algoritmul compară:

$$\begin{aligned} p_1 &= P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1) = \frac{2}{4} \cdot \frac{2}{4} \cdot 1 = \frac{1}{4} \\ p_2 &= P(X_1 = 0 | Y = 2) \cdot P(X_2 = 0 | Y = 2) \cdot P(Y = 2) = 0 \cdot 0 \cdot 0 = 0 \end{aligned}$$

Cum $p_1 > p_2$ algoritmul alege pentru Y valoarea 1, ceea ce este corect.

- Pentru celelalte 3 cazuri, $(X_1 = 0, X_2 = 1)$, $(X_1 = 1, X_2 = 0)$ și $(X_1 = 1, X_2 = 1)$, se observă că se obțin aceleași valori pentru p_1 și p_2 ca mai sus, deci algoritmul alege (în mod corect) pentru Y valoarea 1.

Așadar, am obținut că rata erorii este în acest caz 0.

Cazul 2: $\begin{array}{ c c c } \hline X_1 & X_2 & Y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 2 \\ \hline \end{array}$			Cazuri similare: $\begin{array}{ c c c c c } \hline Y & Y & Y & Y & Y \\ \hline 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \\ \hline \end{array}$		
---	--	--	---	--	--

¹²¹Pentru $Y = 1$, regula este: X_2 are aceeași valoare ca și X_1 . Pentru $Y = 2$, regula este: X_1 și X_2 au valori complementare.

- Pentru $X_1 = 0, X_2 = 0$:

$$\left. \begin{aligned} p_1 &= P(X_1 = 0 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1) = \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{3}{4} = \frac{1}{3} \\ p_2 &= P(X_1 = 0 | Y = 2) \cdot P(X_2 = 0 | Y = 2) \cdot P(Y = 2) = 0 \cdot 0 \cdot \frac{1}{4} = 0 \end{aligned} \right\} \Rightarrow \hat{y} = 1$$

- Pentru $X_1 = 0, X_2 = 1$:

$$\left. \begin{aligned} p_1 &= P(X_1 = 0 | Y = 1) \cdot P(X_2 = 1 | Y = 1) \cdot P(Y = 1) = \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{3}{4} = \frac{1}{6} \\ p_2 &= P(X_1 = 0 | Y = 2) \cdot P(X_2 = 1 | Y = 2) \cdot P(Y = 2) = 0 \cdot 1 \cdot \frac{1}{4} = 0 \end{aligned} \right\} \Rightarrow \hat{y} = 1$$

- Pentru $X_1 = 1, X_2 = 0$:

$$\left. \begin{aligned} p_1 &= P(X_1 = 1 | Y = 1) \cdot P(X_2 = 0 | Y = 1) \cdot P(Y = 1) = \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{3}{4} = \frac{1}{6} \\ p_2 &= P(X_1 = 1 | Y = 2) \cdot P(X_2 = 0 | Y = 2) \cdot P(Y = 2) = 1 \cdot 0 \cdot \frac{1}{4} = 0 \end{aligned} \right\} \Rightarrow \hat{y} = 1$$

- Pentru $X_1 = 1, X_2 = 1$:

$$\left. \begin{aligned} p_1 &= P(X_1 = 1 | Y = 1) \cdot P(X_2 = 1 | Y = 1) \cdot P(Y = 1) = \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{3}{4} = \frac{1}{12} \\ p_2 &= P(X_1 = 1 | Y = 2) \cdot P(X_2 = 1 | Y = 2) \cdot P(Y = 2) = 1 \cdot 1 \cdot \frac{1}{4} = \frac{1}{4} \end{aligned} \right\} \Rightarrow \hat{y} = 2$$

Deci rata erorii este 0 pentru acestă definiție a lui \hat{y} .

<i>Cazul 3:</i>	X_1	X_2	Y
	0	0	1
	0	1	2
	1	0	1
	1	1	2

<i>Cazuri similare:</i>	Y	Y	Y
	2	1	2
	1	1	2
	2	2	1
	1	2	1

- Pentru $X_1 = 0, X_2 = 0$:

$$\left. \begin{aligned} p_1 &= \frac{1}{2} \cdot 1 \cdot \frac{1}{2} = \frac{1}{4} \\ p_2 &= \frac{1}{2} \cdot 0 \cdot \frac{1}{2} = 0 \end{aligned} \right\} \Rightarrow p_1 > p_2 \Rightarrow \hat{y} = 1 \text{ (corect)}$$

- Pentru $X_1 = 0, X_2 = 1$:

$$\left. \begin{aligned} p_1 &= \frac{1}{2} \cdot 0 \cdot \frac{1}{2} = 0 \\ p_2 &= \frac{1}{2} \cdot 1 \cdot \frac{1}{2} = \frac{1}{4} \end{aligned} \right\} \Rightarrow p_1 < p_2 \Rightarrow \hat{y} = 2 \text{ (corect)}$$

- Pentru $X_1 = 1, X_2 = 0$:

$$\left. \begin{aligned} p_1 &= \frac{1}{2} \cdot 1 \cdot \frac{1}{2} = \frac{1}{4} \\ p_2 &= \frac{1}{2} \cdot 0 \cdot \frac{1}{2} = 0 \end{aligned} \right\} \Rightarrow p_1 > p_2 \Rightarrow \hat{y} = 1 \text{ (corect)}$$

- Pentru $X_1 = 1, X_2 = 1$:

$$\left. \begin{aligned} p_1 &= \frac{1}{2} \cdot 0 \cdot \frac{1}{2} = 0 \\ p_2 &= \frac{1}{2} \cdot 1 \cdot \frac{1}{2} = \frac{1}{4} \end{aligned} \right\} \Rightarrow p_1 < p_2 \Rightarrow \hat{y} = 2 \text{ (corect)}$$

Prin urmare, rata erorii este 0 și în acest caz.

Cazul 4: (cel de la punctul a)		
X_1	X_2	Y
0	0	1
0	1	2
1	0	2
1	1	1

Este similar cu cazul:	Y
	2
	1
	1
	2

În concluzie, doar pentru 2 moduri (cazul 4) de definire a lui Y rata erorii este de 50%; pentru celelalte 14 moduri (cazurile 1, 2, 3) rata erorii este 0.

12.

(O reprezentare grafică a neconcordanței deciziilor luate de algoritmi Bayes Naiv și Bayes Optimal)

*CMU, 2009 fall, Geoff Gordon, HW4, pr. 1
CMU, 2009 fall, Carlos Guestrin, HW1, pr. 4.1.5*

Pentru un task de clasificare se consideră atributele X_1 , X_2 și X_3 și eticheta Y . Toate acestea sunt variabile aleatoare binare. X_1 și X_2 sunt independente condițional în raport cu Y , iar X_3 este o copie a lui X_2 (așadar, întotdeauna $X_2 = X_3$).

Sunt date următoarele probabilități condiționate:

$$\begin{aligned} P(X_1 = T \mid Y = T) &= p, & P(X_1 = T \mid Y = F) &= 1 - p, \\ P(X_2 = F \mid Y = T) &= q, & P(X_2 = F \mid Y = F) &= 1 - q, \\ P(Y = T) &= 0.5. \end{aligned}$$

Se dă instanța de test $X_1 = T$, $X_2 = X_3 = F$. Vrem să clasificăm această instanță, adică să prezicem valoarea lui Y pentru ea.

a. Arătați că dacă se folosește clasificatorul Bayes Naiv, atunci eticheta pentru instanța aceasta de test este T — ceea ce revine la $P(Y = T \mid X_1 = T, X_2 = F, X_3 = F) \geq 0.5$ — dacă $p \geq \frac{(1-q)^2}{q^2 + (1-q)^2}$.

b. Ce devine inegalitatea de la punctul precedent dacă în schimbul clasificatorului Bayes Naiv se utilizează clasificatorul Bayes Optimal?

c. Desenați cele două curbe de decizie obținute la punctele a și b . Pe axa Ox marcați valorile lui q , iar pe axa Oy marcați valorile lui p . Atât p cât și q variază în intervalul $[0, 1]$. Indicați pe grafic zona în care clasificatorul Bayes Naiv produce un output (Y) diferit de cel al algoritmului Bayes Optimal.

Atenție! Acest exercițiu nu studiază în ce condiții cei doi algoritmi clasifică corect (ori dimpotrivă, eronat) instanța $X_1 = T, X_2 = X_3 = F$, ci doar când anume (în funcție de p și q) produc ei clasificări diferite pentru această instanță. Se va vedea (grafic!) că algoritmul Bayes Naiv nu se comportă deloc rău în comparație cu algoritmul Bayes Optimal.

Răspuns:

Se lucrează cu variabile aleatoare binare, iar pentru claritatea calculelor vom nota prin X faptul că valoarea variabilei aleatoare X este T , iar prin $\neg X$ faptul că $X = F$.

Folosind acestea notății, putem transcrie datele din enunț astfel:

$$\begin{aligned} P(X_1 \mid Y) &= p, & P(X_1 \mid \neg Y) &= 1 - p, \\ P(\neg X_2 \mid Y) &= q, & P(\neg X_2 \mid \neg Y) &= 1 - q, \\ P(Y) &= 0.5. \end{aligned}$$

a. Eticheta pentru instanța aceasta de test este T dacă $P(Y | X_1, \neg X_2, \neg X_3) \geq 0.5$. Vom calcula această probabilitate utilizând regula lui Bayes precum și ipoteza de independentă condițională făcută de algoritmul Bayes Naiv:

$$\begin{aligned} P(Y | X_1, \neg X_2, \neg X_3) &\stackrel{\text{form. Bayes}}{=} \\ &= \frac{P(X_1, \neg X_2, \neg X_3 | Y)P(Y)}{P(X_1, \neg X_2, \neg X_3 | Y)P(Y) + P(X_1, \neg X_2, \neg X_3 | \neg Y)P(\neg Y)} \stackrel{\text{indep. cdt.}}{=} \\ &= \frac{P(X_1 | Y)P(\neg X_2 | Y)P(\neg X_3 | Y)P(Y)}{P(X_1 | Y)P(\neg X_2 | Y)P(\neg X_3 | Y)P(Y) + P(X_1 | \neg Y)P(\neg X_2 | \neg Y)P(\neg X_3 | \neg Y)P(\neg Y)} \\ &= \frac{p \cdot q \cdot q \cdot 0.5}{p \cdot q \cdot q \cdot 0.5 + (1-p) \cdot (1-q) \cdot (1-q) \cdot 0.5} = \frac{pq^2}{pq^2 + (1-p)(1-q)^2} \end{aligned}$$

Deci eticheta este T dacă $\frac{pq^2}{pq^2 + (1-p)(1-q)^2} \geq 0.5$, ceea ce înseamnă că

$$\begin{aligned} pq^2 \geq 0.5(pq^2 + (1-p)(1-q)^2) &\Leftrightarrow pq^2 - 0.5pq^2 \geq 0.5(1-p)(1-q)^2 \\ &\Leftrightarrow pq^2 \geq (1-q)^2 - p(1-q)^2 \Leftrightarrow p(q^2 + (1-q)^2) \geq (1-q)^2 \\ &\Leftrightarrow p \geq \frac{(1-q)^2}{q^2 + (1-q)^2} \end{aligned}$$

b. Dacă în locul clasificatorului Bayes Naiv se folosește clasificatorul Bayes Optimal, nu se mai folosește presupunerea de independentă condițională. În locul acesteia se folosesc informațiile furnizate în enunț, și anume că X_1 și X_2 sunt independente condițional în raport cu Y , iar X_3 este o copie a lui X_2 .

$$\begin{aligned} P(Y | X_1, \neg X_2, \neg X_3) &\stackrel{\text{form. Bayes}}{=} \\ &= \frac{P(X_1, \neg X_2, \neg X_3 | Y)P(Y)}{P(X_1, \neg X_2, \neg X_3 | Y)P(Y) + P(X_1, \neg X_2, \neg X_3 | \neg Y)P(\neg Y)} \\ &= \frac{P(X_1 | Y)P(\neg X_2, \neg X_3 | Y)P(Y)}{P(X_1 | Y)P(\neg X_2, \neg X_3 | Y)P(Y) + P(X_1 | \neg Y)P(\neg X_2, \neg X_3 | \neg Y)P(\neg Y)} \\ &= \frac{P(X_1 | Y)P(\neg X_2 | Y)P(Y)}{P(X_1 | Y)P(\neg X_2 | Y)P(Y) + P(X_1 | \neg Y)P(\neg X_2 | \neg Y)P(\neg Y)} \\ &= \frac{p \cdot q \cdot 0.5}{p \cdot q \cdot 0.5 + (1-p) \cdot (1-q) \cdot 0.5} = \frac{pq}{pq + (1-p)(1-q)} \end{aligned}$$

S-a folosit egalitatea

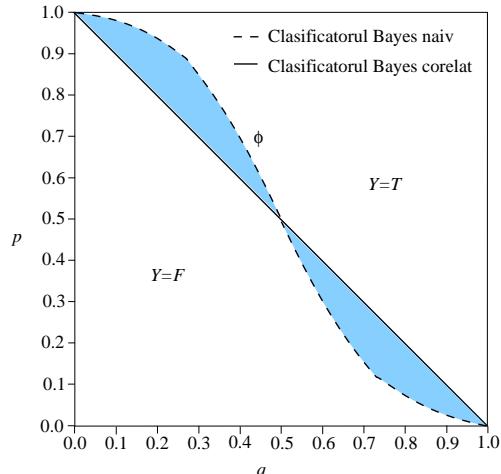
$$P(X_1, \neg X_2, \neg X_3 | Y) = P(X_1 | \neg X_2, \neg X_3, Y) \cdot P(\neg X_2, \neg X_3 | Y) = P(X_1 | Y) \cdot P(\neg X_2 | Y).$$

În acest caz eticheta este T dacă $\frac{pq}{pq + (1-p)(1-q)} \geq 0.5$, adică

$$\begin{aligned} pq \geq 0.5(pq + (1-p)(1-q)) &\Leftrightarrow pq - 0.5pq \geq 0.5(1-p-q+pq) \\ &\Leftrightarrow pq \geq 1-p-q+pq \Leftrightarrow p \geq 1-q. \end{aligned}$$

c. Dreapta de ecuație $p = 1 - q$ este ușor de reprezentat.

Notând $\phi(q) = \frac{(1-q)^2}{q^2 + (1-q)^2}$, se observă imediat că $\phi(q) + \phi(1-q) = 1 \Leftrightarrow \phi(1-q) = 1 - \phi(q)$. De aici, cu ajutorul unui raționament geometric simplu, se ajunge imediat la concluzia că graficul funcției ϕ este simetric față de punctul de coordonate $(1/2, 1/2)$. Așadar, va fi suficient să studiem graficul lui ϕ pe intervalul $[0, 1/2]$. Proprietățile funcției ϕ necesare elaborării graficului sunt ușor de studiat. Adițional, se poate arăta imediat că $\phi(q) \geq 1 - q$ pentru orice $q \in [0, 1/2]$ și $\phi(q) \leq 1 - q$ pentru orice $q \in [1/2, 1]$.



În figura de mai sus am reprezentat curbele $p = 1 - q$ și $p = \phi(q)$, precum și zonele de decizie pentru cei doi clasificatori obținuți la punctele a și b . Se observă ușor zona în care rezultatul produs de clasificatorul Bayes Naiv (Y) este diferit / „eronat“ în raport cu algoritmul Bayes Optimal.

13.

(Cât de multe date de antrenament necesită algoritmul Bayes Naiv vs. algoritmul Bayes Optimal?
[LC: complexitatea la eșantionare])

■ CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW2, pr. 1.1

Unul dintre motivele pentru care folosim clasificatorul Bayes Naiv este faptul că el neceșită mult mai puține date de antrenament (în vederea estimării parametrilor) decât clasificatorul Bayes Optimal.

Acest exercițiu te va ajuta să înțelegi cât de importantă este această diferență dintre cei doi algoritmi.

Presupunem că o *observație / instanță* este o valoare generată în mod aleatoriu de către variabilă aleatoare comună $\bar{X} = (X_1, \dots, X_{d-1}, X_d)$, unde fiecare X_i este o variabilă aleatoare urmând distribuția probabilistă Bernoulli de parametru $p = 0.5$. Considerăm X_1, \dots, X_{d-1} variabilele de intrare, iar $X_d = Y$ variabila de ieșire.

Pentru a estima în sensul verosimilității maxime (MLE) parametrii clasificatorului Bayes Optimal, avem nevoie să *observăm / întâlnim* fiecare valoare a lui \bar{X} de un număr rezonabil de ori. Similar, pentru a antrena clasificatorul Bayes Naiv avem nevoie să întâlnim fiecare valoare a fiecărei variabile X_i ($i = \overline{1, d}$) de un număr rezonabil de ori.

Ne întrebăm cât de multe observații sunt necesare (a fi generate) pentru ca fiecare valoare a variabilei comune \bar{X} în cazul algoritmului Bayes Optimal, și respectiv fiecare valoare a variabilelor X_i ($i = \overline{1, d}$) în cazul Bayes Naiv să fie întâlnită cel puțin o dată. (În practică este nevoie de mult mai multe observații, dar în acest exercițiu ne limităm la câte o singură observație pentru fiecare valoare în parte.)

Indicație: La rezolvarea punctelor de mai jos vă sugerăm să folosiți următoarele două inegalități:

- pentru orice evenimente E_1, \dots, E_n , avem $P(E_1 \cup \dots \cup E_n) \leq \sum_{i=1}^n P(E_i)$.¹²²
- $(1 - \frac{1}{k})^k \leq \frac{1}{e}$ pentru orice $k \geq 1$, unde $e \approx 2.71828$ este baza logaritmului natural.

- a. Începem cu algoritmul Bayes Naiv. Fie $i \in \{1, \dots, d\}$ fixat. Arătați că dacă s-au făcut N observații (având forma $\bar{x}_j = (x_1^j, \dots, x_{d-1}^j, x_d^j)$ cu $j = 1, \dots, N$), atunci probabilitatea să nu fi întâlnit ambele valori ale variabilei X_i este $\frac{1}{2^{N-1}}$. (Observați că această fracție reprezintă un număr foarte mic atunci când N este suficient de mare.)
- b. Fie $\varepsilon > 0$ fixat. Folosind prima inegalitate din *indicația* de mai sus, arătați că dacă au fost făcute câte $N_{NB} = 1 + \log_2 \frac{d}{\varepsilon}$ observații pentru fiecare din variabilele X_i ($i = \overline{1, d}$), atunci probabilitatea să nu fi întâlnit ambele valori pentru fiecare dintre aceste variabile este mai mică sau egală cu ε .
- c. Acum trecem la algoritmul Bayes Optimal. Fie \bar{x} o instanță (fixată) a variabilei comune \bar{X} . Folosind a doua inegalitate din *indicația* de mai sus, arătați că dacă s-au făcut N observații (fiecare observație implicând simultan toate variabilele X_i cu $i = \overline{1, d}$), atunci probabilitatea ca să nu se fi întâlnit niciodată \bar{x} este mai mică sau egală cu $e^{-\frac{N}{2^d}}$.
- d. Arătați că dacă au fost făcute cel puțin $N_{JB} = 2^d \ln \frac{2^d}{\varepsilon}$ observații, atunci probabilitatea ca să nu se fi întâlnit toate instanțele variabilei comune \bar{X} este mai mică sau egală cu ε .
- e. Dacă se fixează $\varepsilon = 0.1$, calculați valorile N_{NB} și N_{JB} pentru $d = 2$, $d = 5$ și $d = 10$. Ce concluzie puteți trage? (Altfel spus, cum interpretați rezultatele?)

Observații:

1. Știm că pentru clasificatorul Bayes Naiv, este necesar să estimăm din date probabilitățile $P(Y = y)$ — adică, $P(X_d = x_d)$ — și $P(X_i = x_i | Y = y)$, pentru $i = 1, \dots, d-1$. Putem considera că este suficient să întâlnim cu o probabilitate de cel puțin $1 - \varepsilon$ toate valorile y , precum și perechile de forma (x_i, y) , cu $i = 1, \dots, d-1$. În mod implicit, problema noastră simplifică și mai mult cerințele, considerând că este suficient să găsim cu probabilitate de cel puțin $1 - \varepsilon$ toate valorile x_i , pentru $i = 1, \dots, d$ (subînțelegând că atunci vor apărea în date, cu probabilități semnificative, atât valorile y cât și fiecare dintre combinațiile (x_i, y) , cu $i = 1, \dots, d-1$).
2. Pentru clasificatorul Bayes Optimal, în mod similar, este necesar să estimăm probabilitățile $P(Y = y)$ și $P(X_1 = x_1, \dots, X_{d-1} = x_{d-1} | Y = y)$, ceea ce, evident, va permite calcularea probabilităților de forma $P(x_1, \dots, x_{d-1}, y)$. Problema noastră consideră în mod implicit că este suficient să găsim cu probabilitate de cel puțin $1 - \varepsilon$ toate combinațiile (x_1, \dots, x_{d-1}, y) .

Răspuns:

- a. Dacă s-au facut N observații și nu s-au întâlnit ambele valori ale variabilei X_i , înseamnă că ea are aceeași valoare în toate aceste observații (adică ea este fie 0 în toate observațiile, fie 1 în toate observațiile). Pentru fiecare dintre aceste două cazuri probabilitatea este $1/2^N$, deci:

$$\begin{aligned} P(\text{doar una dintre valorile variabilei } X_i \text{ a apărut în } N \text{ observații}) \\ = \left(\frac{1}{2}\right)^N + \left(\frac{1}{2}\right)^N = \frac{2}{2^N} = \frac{1}{2^{N-1}} \end{aligned}$$

¹²²Aceasta se numește *proprietatea de subaditivitate* a probabilităților.

b. Se cere să se calculeze probabilitatea să nu fi întâlnit ambele valori pentru fiecare dintre variabilele X_1, \dots, X_{d-1}, X_d . Pentru acesta vom folosi prima inegalitate din indicația dată în enunț:

$P(\text{nu toate valorile variabilelor } X_i, i = \overline{1, d}, \text{ au apărut în } N_{NB} \text{ observații})$

$$\begin{aligned} &\leq \sum_{i=1}^d P(\text{numai una dintre valorile variabilei } X_i \text{ a apărut în } N_{NB} \text{ observații}) \\ &= \sum_{i=1}^d \frac{1}{2^{N_{NB}-1}} = d \cdot \frac{1}{2^{N_{NB}-1}} = d \cdot \frac{1}{2^{1+\log_2 \frac{d}{\varepsilon}-1}} = d \cdot \frac{1}{2^{\log_2 \frac{d}{\varepsilon}}} = d \cdot \frac{1}{\frac{d}{\varepsilon}} = d \cdot \frac{\varepsilon}{d} = \varepsilon. \end{aligned}$$

c. Pentru algoritmul Bayes Optimal s-au făcut N observații. Trebuie să calculăm probabilitatea ca să nu se fi întâlnit niciodată instanța \bar{x} .

Cum există în total 2^d posibile observații, probabilitatea ca instanța \bar{x} să nu fie obținută la *una* dintre observații este $1 - \frac{1}{2^d}$. Observațiile fiind independente, probabilitatea ca \bar{x} să nu fie obținută *după* N observații este $\left(1 - \frac{1}{2^d}\right)^N$. Așadar,

$$\begin{aligned} P(\text{instanța } \bar{x} \text{ n-a fost întâlnită în } N \text{ observații}) &= \left(1 - \frac{1}{2^d}\right)^N \\ &= \left[\left(1 - \frac{1}{2^d}\right)^{2^d}\right]^{N/2^d} \leq \left(\frac{1}{e}\right)^{N/2^d} = e^{-N/2^d} \end{aligned}$$

d. Vom calcula probabilitatea ca să nu se fi întâlnit toate instanțele variabilei \bar{X} utilizând din nou prima inegalitate din *indicație*:

$P(\text{nu toate instanțele variabilei } \bar{X} \text{ au fost întâlnite în } N_{JB} \text{ observații})$

$$\begin{aligned} &\leq \sum_{\bar{x}} P(\text{instanța } \bar{x} \text{ n-a fost întâlnită în } N_{JB} \text{ observații}) \\ &\leq \sum_{\bar{x}} e^{-N_{JB}/2^d} = 2^d \cdot e^{-N_{JB}/2^d} = 2^d \cdot e^{-\ln \frac{2^d}{\varepsilon}} = 2^d \cdot \frac{1}{e^{\ln \frac{2^d}{\varepsilon}}} = \frac{2^d}{\varepsilon} = \varepsilon. \end{aligned}$$

e. Vom înlocui datele numerice în formulele $N_{NB} = 1 + \log_2 \frac{d}{\varepsilon}$, $N_{JB} = 2^d \ln \frac{2^d}{\varepsilon}$.

$$\begin{aligned} \varepsilon = 0.1, d = 2 \Rightarrow &\begin{cases} N_{NB} = 1 + \log_2 \frac{2}{0.1} = 1 + \log_2 20 \approx 5.32 \\ N_{JB} = 2^2 \cdot \ln \frac{2^2}{0.1} = 4 \cdot \ln 40 \approx 14.75 \end{cases} \\ \varepsilon = 0.1, d = 5 \Rightarrow &\begin{cases} N_{NB} = 1 + \log_2 \frac{5}{0.1} = 1 + \log_2 50 \approx 6.64 \\ N_{JB} = 2^5 \cdot \ln \frac{2^5}{0.1} = 32 \cdot \ln 320 \approx 184.58 \end{cases} \\ \varepsilon = 0.1, d = 10 \Rightarrow &\begin{cases} N_{NB} = 1 + \log_2 \frac{10}{0.1} = 1 + \log_2 100 \approx 7.64 \\ N_{JB} = 2^{10} \cdot \ln \frac{2^{10}}{0.1} = 1024 \cdot \ln 10240 \approx 9455.67 \end{cases} \end{aligned}$$

Acum se observă ușor [diferența dintre] numărul de date de antrenament necesare pentru cei doi algoritmi: de ordin logaritmic pentru Bayes Naiv, respectiv de ordin exponențial pentru Bayes Optimal.

14. (Algoritmul Bayes Naiv: raportul cu regresia logistică și natura separatorului decizional; cazul când variabilele de intrare sunt de tip boolean)
- CMU, 2005 fall, T. Mitchell, A. Moore, HW2, pr. 2
 - CMU, 2009 fall, Carlos Guestrin, HW1, pr. 4.1.2
 - CMU, 2009 fall, Geoff Gordon, HW4, pr. 1.2-3
 - CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 3.a

a. [Bayes Naiv și Regresia Logistică: relația dintre regulile de decizie]¹²³

Fie Y o variabilă aleatoare Bernoulli, iar $X = (X_1, \dots, X_d)$ un vector de variabile booleene. Demonstrați că distribuția condițională $P(Y|X)$ are forma funcției logistice de argument $z = -(w_0 + w_1 X_1 + \dots + w_d X_d)$, cu parametrii $w_0, w_1, \dots, w_d \in \mathbb{R}$,¹²⁴ adică

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^d w_i X_i)}$$

și, prin urmare

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^d w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^d w_i X_i)}.$$

Vă reamintim că *funcția logistică* (sau *sigmoidală*) este definită prin expresia $\sigma(z) = 1/(1 + e^{-z})$ pentru orice $z \in \mathbb{R}$.

Comentariu:

Regresia logistică (și, mai general, *clasificatorii „discriminativi“*) învață [în mod] direct parametrii distribuției $P(Y|X)$,¹²⁵ pe când algoritmul Bayes Naiv (și, mai general, *clasificatorii „generativi“*) învață [parametrii pentru] distribuțiile $P(X|Y)$ și $P(Y)$, cu ajutorul cărora va calcula apoi $P(Y|X)$ și cea mai probabilă valoare pentru Y (atunci când X are o valoare fixată / dată). Vom spune că regresia logistică este corespondentul „discriminativ“ al clasificatorului „generativ“ Bayes Naiv.

Indicații:

1. Vom introduce o *notație* simplă, care ne va fi de folos în continuare. Întrucât variabilele X_i sunt booleene, odată fixată o valoare y_k pentru variabila Y , vom avea nevoie de un

¹²³ Pentru o introducere în chestiunea regresiei logistice, vedeți Tom Mitchell, *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*, draft pentru un capitol suplimentar pentru o nouă ediție a cărții *Machine Learning*, 2016. Puteți vedea de asemenea problema 12 de la capitolul *Metode de regresie* din prezenta culegere.

¹²⁴ LC: Prin urmare, *separatorul decizional* (sau, *granița de decizie*) pentru algoritmul Bayes Naiv este — într-o astfel de situație — liniar (în funcție de argumentele X_1, \dots, X_d). Ecuatia separatorului decizional va fi $w_0 + w_1 X_1 + \dots + w_d X_d = 0$.

¹²⁵ LC: Parametrii distribuției $P(Y|X)$ sunt în acest caz $w_i \in \mathbb{R}$, cu $i = 0, 1, \dots, d$, iar învățarea lor se face prin maximizarea funcției de verosimilitate $\mathcal{L}(w) \stackrel{\text{not.}}{=} P(D|w)$, unde D este setul de date de antrenament. La rândul ei, maximizarea aceasta se realizează prin aplicarea unei metode de optimizare, de exemplu metoda gradientului ascendent sau metoda lui Newton. Vedeți de exemplu problemele 12 și 13 de la capitolul *Metode de regresie*.

singur parametru pentru a defini distribuția condițională $P(X_i|Y = y_k)$, pentru fiecare $i = 1, \dots, d$. Așadar, vom desemna cu θ_{i1} probabilitatea $P(X_i = 1|Y = 1)$ și, prin urmare $P(X_i = 0|Y = 1) = 1 - \theta_{i1}$. În mod similar, vom desemna cu θ_{i0} probabilitatea $P(X_i = 1|Y = 0)$.

2. Remarcați că odată ce am introdus notațiile de mai sus, vom putea scrie $P(X_i|Y = 1)$ după cum urmează:

$$P(X_i|Y = 1) = \theta_{i1}^{X_i} (1 - \theta_{i1})^{(1-X_i)}, \quad (68)$$

bineînțeles, cu excepția cazurilor când $\theta_{i1} = 0$ și $X_i = 0$, respectiv $\theta_{i1} = 1$ și $X_i = 1$. Observați că atunci când X_i are valoarea 1, cel de-al doilea factor din partea dreaptă a egalității (68) este 1, pentru că exponentul lui este zero. Deci $P(X_i|Y = 1) = \theta_{i1}^{X_i} = \theta_{i1}$ pentru $X_i = 1$. În mod similar, atunci când $X_i = 0$ primul factor este egal cu 1, pentru că exponentul lui este zero. Deci $P(X_i|Y = 1) = (1 - \theta_{i1})^{1-X_i} = 1 - \theta_{i1}$ pentru $X_i = 0$.

b. [Relaxarea presupozitiei de independentă condițională]¹²⁶

Pentru a putea exprima interacțiunile dintre trăsături, modelul regresiei logistice poate fi extins cu niște termeni suplimentari. De exemplu, putem adăuga un termen care să exprime dependența dintre trăsăturile X_1 și X_2 :

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + w_{1,2}X_1X_2 + \sum_{i=1}^d w_i X_i)}.$$

În mod similar, presupozitia de independentă condițională asumată de către algoritmul Bayes Naiv poate fi relaxată astfel încât trăsăturile X_1 și X_2 să nu mai trebuiască să satisfacă independentă condițională. Așadar, vom putea scrie:

$$P(Y|X) = \frac{P(Y) P(X_1, X_2|Y) \prod_{i=3}^d P(X_i|Y)}{P(X)}.$$

Demonstrați că în acest caz distribuția $P(Y|X)$ are aceeași formă ca și modelul de regresie logistică augmentat cu un termen suplimentar, care exprimă dependența dintre X_1 și X_2 (și, în acest fel, modelul extins al regresiei logistice rămâne corespondentul discriminativ al clasificatorului nostru generativ).

Indicații:

3. De data aceasta o altă notație simplă ne va ajuta. Vom avea nevoie de mai mulți parametri decât la punctul a pentru a defini distribuția comună $P(X_1, X_2|Y)$. Așa că vom nota $\beta_{ijk} = P(X_1 = i, X_2 = j|Y = k)$, pentru fiecare combinație posibilă de valori pentru indicii i, j și k .
4. Această nouă notație poate fi folosită acum pentru a exprima probabilitatea $P(X_1, X_2|Y = k)$ după cum urmează:

$$P(X_1, X_2|Y = k) = (\beta_{11k})^{X_1 X_2} (\beta_{10k})^{X_1 (1-X_2)} (\beta_{01k})^{(1-X_1) X_2} (\beta_{00k})^{(1-X_1)(1-X_2)} \quad (69)$$

pentru $k \in \{0, 1\}$, cu excepția următoarelor cazuri: i. $\beta_{11k} = 0$ și $X_1 X_2 = 0$, ii. $\beta_{10k} = 0$ și $X_1 (1 - X_2) = 0$, iii. $\beta_{01k} = 0$ și $(1 - X_1) X_2 = 0$ și iv. $\beta_{00k} = 0$ și $(1 - X_1)(1 - X_2) = 0$.

Răspuns:

¹²⁶Vedeți de exemplu problema 9.

a. Mai întâi vom scrie probabilitatea $P(Y = 1|X = x)$ ca o fracție, folosind formula lui Bayes, compusă cu formula probabilității totale, apoi vom împărți atât numărătorul cât și numitorul fracției astfel obținute cu expresia de la numărător:¹²⁷

$$\begin{aligned} P(Y = 1|X = x) &\stackrel{FB}{=} \frac{P(X = x|Y = 1)P(Y = 1)}{\sum_{y' \in \{0,1\}} P(X = x|Y = y')P(Y = y')} \\ &= \frac{1}{1 + \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)}}. \end{aligned}$$

După aceea, folosind formula $e^{\ln a} = a$ (valabilă pentru orice $a > 0$), vom forța punerea fracției de mai sus într-o formă apropiată de cea a funcției sigmoidale ($\sigma(x) = 1/(1 + e^{-x})$):¹²⁸

$$\begin{aligned} P(Y = 1|X = x) &= \frac{1}{1 + \exp\left(\ln \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)}\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{P(X_1 = x_1, \dots, X_d = x_d|Y = 0)P(Y = 0)}{P(X_1 = x_1, \dots, X_d = x_d|Y = 1)P(Y = 1)}\right)}. \end{aligned}$$

Mai departe, ținând cont de presupozitia de independentă condițională și de proprietățile funcției logaritm, obținem:¹²⁹

$$P(Y = 1|X = x) = \frac{1}{1 + \exp\left(\ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \ln \frac{P(X_i = x_i|Y = 0)}{P(X_i = x_i|Y = 1)}\right)}.$$

Vom nota probabilitățile a priori $P(Y = 1)$ și $P(Y = 0)$ cu π și respectiv $1 - \pi$. Apoi, conform *indicației* 2, vom scrie $P(X_i|Y = 1)$ ca $\theta_{i1}^{X_i}(1 - \theta_{i1})^{(1-X_i)}$ și $P(X_i|Y = 0)$ ca $\theta_{i0}^{X_i}(1 - \theta_{i0})^{(1-X_i)}$. În consecință,

$$\begin{aligned} P(Y = 1|X = x) &= \frac{1}{1 + \exp\left(\ln \frac{1 - \pi}{\pi} + \sum_{i=1}^d \ln \frac{\theta_{i0}^{X_i}(1 - \theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i}(1 - \theta_{i1})^{(1-X_i)}}\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{1 - \pi}{\pi} + \sum_{i=1}^d \left(X_i \ln \frac{\theta_{i0}}{\theta_{i1}} + (1 - X_i) \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}}\right)\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{1 - \pi}{\pi} + \sum_{i=1}^d \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}} + \sum_{i=1}^d X_i \left(\ln \frac{\theta_{i0}}{\theta_{i1}} - \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}}\right)\right)}. \end{aligned}$$

Pentru a pune această ultimă expresie sub forma dorită, adică $P(Y = 1|X = x) = 1/(1 + \exp(w_0 + \sum_{i=1}^d w_i X_i))$, vom alege valorile parametrilor w_i în mod natural:

$$w_0 = \ln \frac{1 - \pi}{\pi} + \sum_{i=1}^d \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}} \quad \text{și} \quad w_i = \ln \frac{\theta_{i0}}{\theta_{i1}} - \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}} \text{ pentru } i = 1, \dots, d.$$

¹²⁷Cu excepția cazului când $P(X = x|Y = 1)P(Y = 1) = 0$.

¹²⁸Cu excepția cazului când $P(X = x|Y = 0)P(Y = 0) = 0$.

¹²⁹Cu excepția cazurilor când $P(X = x_i|Y = 0) = 0$ sau $P(X = x_i|Y = 1) = 0$ pentru $i = 1, \dots, d$.

b. Vom începe ca și la punctul precedent, prin a pune probabilitatea $P(Y = 1|X = x)$ sub o formă apropiată de cea a funcției sigmoidale:¹³⁰

$$\begin{aligned} P(Y = 1|X) &\stackrel{FB}{=} \frac{P(X|Y = 1)P(Y = 1)}{P(X|Y = 1)P(Y = 1) + P(X|Y = 0)P(Y = 0)} \\ &= \frac{1}{1 + \frac{P(X|Y = 0)P(Y = 0)}{P(X|Y = 1)P(Y = 1)}} \\ &= \frac{1}{1 + \exp\left(\ln \frac{P(X|Y = 0)P(Y = 0)}{P(X|Y = 1)P(Y = 1)}\right)}. \end{aligned}$$

Până aici nu avem încă nicio diferență în raport cu calculul de la punctul a. Însă acum vom ține cont că toate variabilele X_i ci $i = 1, \dots, d$ sunt independente condițional două câte două în raport cu Y , cu excepția perechii X_1, X_2 :¹³¹

$$\begin{aligned} P(Y = 1|X) &= \frac{1}{1 + \exp\left(\ln \frac{P(X_1, X_2|Y = 0)}{P(X_1, X_2|Y = 1)} \frac{\prod_{i=3}^d P(X_i|Y = 0)P(Y = 0)}{\prod_{i=3}^d P(X_i|Y = 1)P(Y = 1)}\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{1 - \pi}{\pi} + \sum_{i=3}^d \ln \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)} + \ln \frac{P(X_1, X_2|Y = 0)}{P(X_1, X_2|Y = 1)}\right)}. \end{aligned}$$

Ca și la punctul a, vom ține cont că

$$\ln \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)} = \ln \frac{\theta_{i0}^{X_i}(1 - \theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i}(1 - \theta_{i1})^{(1-X_i)}}$$

atunci când condițiile asociate cu relația (68) sunt satisfăcute. Mai departe, folosind *indicația 4*, vom putea înlocui $P(X_1, X_2|Y = 0)$ și $P(X_1, X_2|Y = 1)$ în funcție de β_{ijk} , obținând (atunci când condițiile asociate cu relația (69) sunt satisfăcute):

$$\ln \frac{P(X_1, X_2|Y = 0)}{P(X_1, X_2|Y = 1)} = \ln \frac{(\beta_{110})^{X_1 X_2} (\beta_{100})^{X_1(1-X_2)} (\beta_{010})^{(1-X_1)X_2} (\beta_{000})^{(1-X_1)(1-X_2)}}{(\beta_{111})^{X_1 X_2} (\beta_{101})^{X_1(1-X_2)} (\beta_{011})^{(1-X_1)X_2} (\beta_{001})^{(1-X_1)(1-X_2)}}.$$

Așadar, va rezulta:

$$P(Y = 1|X) = \frac{1}{1 + \exp\left(w_0 + \sum_{i=3}^d w_i X_i + w_1 X_1 + w_2 X_2 + w_{1,2} X_1 X_2\right)},$$

unde

$$\begin{aligned} w_0 &= \ln \frac{1 - \pi}{\pi} + \sum_{i=3}^d \ln \frac{1 - \theta_{i1}}{1 - \theta_{i0}} + \ln \frac{\beta_{000}}{\beta_{001}} \\ w_1 &= \ln \frac{\beta_{100}}{\beta_{101}} + \ln \frac{\beta_{001}}{\beta_{000}} \\ w_2 &= \ln \frac{\beta_{010}}{\beta_{011}} + \ln \frac{\beta_{001}}{\beta_{000}} \end{aligned}$$

¹³⁰Cu excepția cazurilor când $P(X|Y = 1)P(Y = 1) = 0$ sau $P(X|Y = 0)P(Y = 0) = 0$.

¹³¹Cu excepția cazurilor când $P(Y = 0) = 0$ sau $P(Y = 1) = 0$, respectiv $P(X_1 X_2|Y = 0) = 0$ sau $P(X_1 X_2|Y = 1) = 0$ și încă $P(X_i|Y = 0) = 0$ sau $P(X_i|Y = 1) = 0$ pentru $i = 3, \dots, d$.

$$\begin{aligned} w_{1,2} &= \ln \frac{\beta_{110}}{\beta_{111}} + \ln \frac{\beta_{101}}{\beta_{100}} + \ln \frac{\beta_{011}}{\beta_{010}} + \ln \frac{\beta_{000}}{\beta_{001}} \\ w_i &= \ln \frac{\theta_{i0}}{\theta_{i1}} + \ln \frac{1 - \theta_{i1}}{1 - \theta_{i0}} \text{ pentru } i = 3, \dots, d. \end{aligned}$$

2.1.3 Clasificare bayesiană [cu attribute de intrare] de tip gaussian

15. (Algoritmul Bayes [Naiv] gaussian: aplicare pe date din \mathbb{R})
*prelucrare de Liviu Ciortuz, după
 ■ CMU, 2001 fall, Andrew Moore, midterm, pr. 3.a*

Presupunem că disponem de setul de date de antrenament din tabelul alăturat; singurul atribut de intrare (X) ia valori reale, iar atributul de ieșire (Y) este de tip Bernoulli, deci ia două valori, notate cu A și respectiv B .

X	Y
0	A
2	A
3	B
4	B
5	B
6	B
7	B

- a. Pornind de la acest set de date, va trebui mai întâi să învățați parametrii clasificatorului Bayes gaussian, prin metoda estimării de verosimilitate maximă (MLE).¹³² Centralizați rezultatele, completând tabelul următor:

$\mu_A =$	$\sigma_A^2 =$	$P(Y = A) =$
$\mu_B =$	$\sigma_B^2 =$	$P(Y = B) =$

- b. Notăm $\alpha = p(X = 2|Y = A)$ și $\beta = p(X = 2|Y = B)$.

- Cât este $p(X = 2, Y = A)$ în funcție de α ?
- Cât este $p(X = 2, Y = B)$ în funcție de β ?
- Cât este $p(X = 2)$ în funcție de α și β ?
- Cât este $p(Y = A|X = 2)$ în funcție de α și β ?

- c. Cum va clasifica algoritmul Bayes [Naiv] gaussian punctul $X = 2$? Puteți exprima răspunsul fie în funcție de α și β , fie — mai bine! — calculând în prealabil valorile lui α și β în funcție de parametrii calculați / estimați la punctul precedent.

Răspuns:

- a. Pentru a estima mediile μ_A și μ_B , vom folosi formula demonstrată la problema 45.a de la capitolul de *Fundamente*:

$$\mu_{MLE} = \frac{\sum_{i=1}^n x_i}{n},$$

¹³²Vedeți secțiunea corespunzătoare din capitolul de *Fundamente*, în spate (pentru acest caz problemele 45.a și 46.a).

unde n este numărul instanțelor de antrenament. Așadar, $\mu_A = \frac{\sum_{i=1}^2 X_i}{2} = \frac{0+2}{2} = 1$, iar $\mu_B = \frac{\sum_{i=3}^7 X_i}{5} = \frac{3+4+5+6+7}{5} = 5$.

Similar, pentru calculul varianțelor σ_A^2 și σ_B^2 , vom folosi formula care a fost demonstrată la problema 46.a de la capitolul de *Fundamente*:

$$\sigma_{MLE}^2 = \frac{\sum_{i=1}^n (x_i - \mu_{MLE})^2}{n}.$$

Așadar, $\sigma_A^2 = \frac{1}{2}[(0-1)^2 + (2-1)^2] = 1$, iar $\sigma_B^2 = \frac{1}{5}[(3-5)^2 + (4-5)^2 + 0^2 + (6-5)^2 + (7-5)^2] = \frac{1}{5} \cdot 2 \cdot [4+1] = 2$.

Pentru calculul probabilităților $P(Y = A)$ și $P(Y = B)$ se folosește formula clasică (numărul de cazuri favorabile împărțit la numărul de cazuri posibile; vedeti problema 40 sau problema 113 de la capitolul de *Fundamente*), fiindcă Y este variabilă de tip Bernoulli. Așadar, $P(Y = A) = 2/7$ și $P(Y = B) = 5/7$.

Centralizând aceste estimări, obținem:

$\mu_A = 1$	$\sigma_A^2 = 1$	$P(Y = A) = 2/7$
$\mu_B = 5$	$\sigma_B^2 = 2$	$P(Y = B) = 5/7$

b. Folosind formula de multiplicare [a probabilităților], calculăm $p(X = 2, Y = A) = p(X = 2|Y = A) \cdot P(Y = A) = \frac{2\alpha}{7}$ și $p(X = 2, Y = B) = p(X = 2|Y = B) \cdot P(Y = B) = \frac{5\beta}{7}$.

Probabilitatea $p(X = 2)$ se poate obține aplicând formula probabilității totale: $p(X = 2) = p(X = 2|Y = A) \cdot P(Y = A) + p(X = 2|Y = B) \cdot P(Y = B) = \frac{1}{7}(2\alpha + 5\beta)$.

Probabilitatea condiționată $p(Y = A|X = 2)$ se calculează folosind definiția: $p(Y = A|X = 2) = \frac{p(Y = A, X = 2)}{p(X = 2)} = \frac{2\alpha}{2\alpha + 5\beta}$.

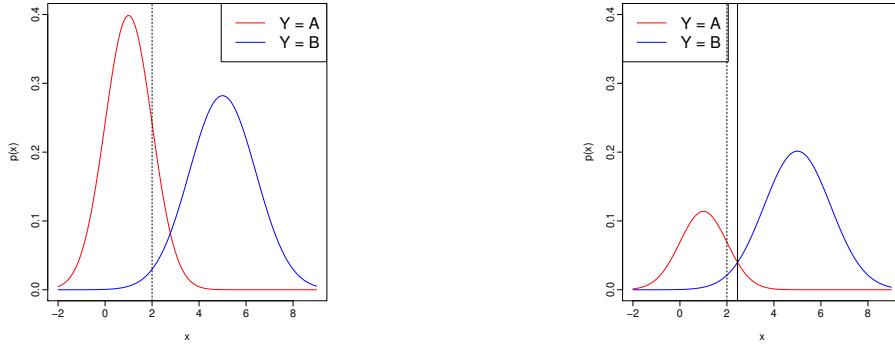
c. Algoritmul Bayes [Naiv] gaussian va asocia punctului $X = 2$ eticheta $Y = A$ dacă $p(Y = A|X = 2) \geq p(Y = B|X = 2) \Leftrightarrow \frac{2}{7}\alpha \geq \frac{5}{7}\beta$.

Folosind valorile estimate pentru parametrii μ_A , μ_B , σ_A și σ_B la punctul a , vom putea scrie: $\alpha = \frac{1}{\sqrt{2\pi}} e^{-\frac{(2-1)^2}{2}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}} = 0.24197$ și $\beta = \frac{1}{\sqrt{2\pi} \cdot \sqrt{2}} e^{-\frac{(2-5)^2}{2 \cdot 2}} = \frac{1}{2\sqrt{\pi}} e^{-\frac{9}{4}} = 0.029732$. Deci,

$$\frac{2}{7}\alpha \geq \frac{5}{7}\beta \Leftrightarrow \frac{2}{7} \cdot 0.24197 \geq \frac{5}{7} \cdot 0.029732 \Leftrightarrow 0.06913 \geq 0.02123 \text{ (adevărat).}$$

Prin urmare, algoritmul Bayes [Naiv] gaussian va asocia punctului $X = 2$ eticheta $Y = A$, cu probabilitatea $\frac{0.06913}{0.06913 + 0.02123} = 0.76499$.

Observație: În următoarele două grafice am reprezentat $\mathcal{N}(\mu_A, \sigma_A^2)$ și $\mathcal{N}(\mu_B, \sigma_B^2)$, p.d.f.-urile gaussianelor asociate claselor A și respectiv B (în partea stângă), precum și funcțiile obținute din aceste două p.d.f.-uri prin înmulțirea cu factorii de selecție 2/7 și respectiv 5/7 (în partea dreaptă).



Se poate constata relativ ușor că există două puncte de intersecție ($x_1 = -8.451$ și $x_2 = 2.451$) pentru graficele funcțiilor $\frac{2}{7}\mathcal{N}(\mu_A, \sigma_A^2)$ și $\frac{5}{7}\mathcal{N}(\mu_B, \sigma_B^2)$. Toate instanțele de test x situate între aceste puncte de intersecție ($x_1 < x < x_2$) vor apartine clasei A (acolo curba roșie este situată deasupra celei albastre). Instanțele situate fie la stânga lui x_1 fie la dreapta lui x_2 vor apartine clasei B (acolo curba albastră este situată deasupra celei roșii). *Separatorul decizional* este de tip pătratic, fiind constituit din punctele x_1 și x_2 .¹³³

16.

(Clasificatorul Bayes [Naiv] gaussian,
cazul când se folosește un singur atribut de intrare:
zone de decizie și granițe de decizie;
analiza diferențelor *cazuri specifice*)

□ Sergiu Dinu, Liviu Ciortuz, 2020

În acest exercițiu ne propunem să identificăm *zonele de decizie* și *granițele de decizie* determinate de clasificatorul Bayes [Naiv] gaussian (abreviat *GB*), atunci când folosim un singur atribut de intrare X , iar atributul de ieșire, pe care îl vom nota cu Y , este binar, deci poate lua două valori, desemnate în continuare cu A și B. Vom desemna prin $p_A = p$ probabilitatea de selecție pentru clasa A. Prin urmare, probabilitatea de selecție pentru clasa B este $p_B = 1 - p$. Vom presupune că $p \in (0, 1)$ și $X|(Y = A) \sim \mathcal{N}(x|\mu_A, \sigma_A^2)$, iar $X|(Y = B) \sim \mathcal{N}(x|\mu_B, \sigma_B^2)$.

a. Arătați că *regula de decizie* a acestui clasificator Bayes [Naiv] gaussian

$$\hat{Y}_{GB}(X = x) = A \Leftrightarrow p \cdot \mathcal{N}(x|\mu_A, \sigma_A^2) \geq (1 - p) \cdot \mathcal{N}(x|\mu_B, \sigma_B^2) \quad (70)$$

devine echivalentă cu

$$(\sigma_A^2 - \sigma_B^2)(x - x_1)(x - x_2) \geq 0,$$

unde

$$x_1 = \frac{\sigma_A^2 \mu_B - \sigma_B^2 \mu_A - \sqrt{\Delta'}}{\sigma_A^2 - \sigma_B^2} \quad \text{și} \quad x_2 = \frac{\sigma_A^2 \mu_B - \sigma_B^2 \mu_A + \sqrt{\Delta'}}{\sigma_A^2 - \sigma_B^2},$$

¹³³LC: Multumesc studentului MSc Dinu Sergiu pentru această observație.

cu

$$\Delta' \stackrel{not.}{=} \sigma_A^2 \sigma_B^2 \left[(\mu_A - \mu_B)^2 + (\sigma_A^2 - \sigma_B^2) \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right)^2 \right],$$

în condițiile în care $\sigma_A^2 \neq \sigma_B^2$ și $\Delta' \geq 0$.

b. Arătați că atunci când $\sigma_A = \sigma_B \stackrel{not.}{=} \sigma$ și $\mu_A \neq \mu_B$, regula de decizie (70) este echivalentă cu

$$x \geq x_0 \text{ dacă } \mu_A > \mu_B$$

și, respectiv

$$x \leq x_0 \text{ dacă } \mu_A < \mu_B,$$

unde

$$x_0 = \frac{\mu_A + \mu_B}{2} + \frac{\sigma^2}{\mu_A - \mu_B} \cdot \ln \frac{1-p}{p}.$$

c. Arătați că este posibil ca în anumite cazuri să avem $\Delta' < 0$, adică inegalitatea (70) să fie ori adevărată pentru orice $x \in \mathbb{R}$, ori falsă pentru orice $x \in \mathbb{R}$. Altfel spus, arătați că pot exista combinații de valori pentru parametrii σ_A și σ_B (ambii strict pozitivi), μ_A și μ_B din \mathbb{R} , precum și $p \in (0, 1)$, astfel încât

$$(\mu_A - \mu_B)^2 + (\sigma_A^2 - \sigma_B^2) \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right)^2 < 0. \quad (71)$$

Răspuns:

a. Putem scrie următoarea succesiune de echivalențe:

$$\begin{aligned} p \cdot \mathcal{N}(x|\mu_A, \sigma_A^2) \geq (1-p) \cdot \mathcal{N}(\mu_B, \sigma_B^2) &\Leftrightarrow \\ p \cdot \frac{1}{\sqrt{2\pi}\sigma_A} \exp \left(-\frac{(x-\mu_A)^2}{2\sigma_A^2} \right) &\geq (1-p) \cdot \frac{1}{\sqrt{2\pi}\sigma_B} \exp \left(-\frac{(x-\mu_B)^2}{2\sigma_B^2} \right) \Leftrightarrow \\ \exp \left(\frac{1}{2} \left[\left(\frac{x-\mu_B}{\sigma_B} \right)^2 - \left(\frac{x-\mu_A}{\sigma_A} \right)^2 \right] \right) &\geq \frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \Leftrightarrow \\ \left(\frac{x-\mu_B}{\sigma_B} \right)^2 - \left(\frac{x-\mu_A}{\sigma_A} \right)^2 &\geq \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right)^2 \Leftrightarrow \\ (\sigma_B^2 - \sigma_A^2)x^2 + 2(\sigma_B^2\mu_A - \sigma_A^2\mu_B)x + \sigma_A^2\mu_B^2 - \sigma_B^2\mu_A^2 - \sigma_A^2\sigma_B^2 \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right)^2 &\geq 0. \quad (72) \end{aligned}$$

Membrul stâng al acestei ultime inegalități este o funcție polinomială de gradul al doilea. Prin urmare, discriminantul ei „pe jumătate“ (adică, $\Delta' \stackrel{not.}{=} \Delta/2$) este:

$$\Delta' = (\sigma_B^2\mu_A - \sigma_A^2\mu_B)^2 - (\sigma_A^2 - \sigma_B^2) \left[\sigma_A^2\mu_B^2 - \sigma_B^2\mu_A^2 - \sigma_A^2\sigma_B^2 \ln \left(\frac{1-p}{p} \frac{\sigma_A}{\sigma_B} \right)^2 \right]$$

Însă,

$$\begin{aligned} (\sigma_B^2\mu_A - \sigma_A^2\mu_B)^2 - (\sigma_A^2 - \sigma_B^2)(\sigma_A^2\mu_B^2 - \sigma_B^2\mu_A^2) &= \\ \cancel{\sigma_B^4\mu_A^2} + \cancel{\sigma_A^4\mu_B^2} - 2\sigma_A^2\sigma_B^2\mu_A\mu_B - \cancel{\sigma_A^4\mu_B^2} + \sigma_A^2\sigma_B^2\mu_A^2 + \sigma_A^2\sigma_B^2\mu_B^2 - \cancel{\sigma_B^4\mu_A^2} &= \\ \sigma_A^2\sigma_B^2(\mu_A - \mu_B)^2. & \end{aligned}$$

Așadar,

$$\Delta' = \sigma_A^2 \sigma_B^2 \left[(\mu_A - \mu_B)^2 + (\sigma_A^2 - \sigma_B^2) \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right)^2 \right],$$

ceea ce ne conduce imediat la concluzia din enunț.

Consecință: În acest caz (a), separatorul decizional este format din punctele x_1 și x_2 . Dacă $\sigma_A^2 > \sigma_B^2$, atunci zona de decizie corespunzătoare clasei A este $(-\infty, x_1] \cup [x_2, +\infty)$, iar zona de decizie corespunzătoare clasei B este intervalul (x_1, x_2) . Similar, dacă $\sigma_A^2 < \sigma_B^2$, atunci zona de decizie corespunzătoare clasei A este (x_1, x_2) , iar zona de decizie corespunzătoare clasei B este intervalul $(-\infty, x_1] \cup [x_2, +\infty)$.

b. Dacă $\sigma_A = \sigma_B = \sigma$, iar $\sigma > 0$, atunci este ușor de observat că inegalitatea (72) devine

$$(\mu_A - \mu_B)x + \mu_B^2 - \mu_A^2 - \sigma^2 \ln \frac{1-p}{p} \geq 0,$$

ceea ce fundamentează concluzia din enunț.

Consecință: În acest caz (b), separatorul decizional este punctul x_0 . Zonele de decizie se stabilesc imediat conform relațiilor din enunț.

c. Observăm că dacă vom considera cazul particular $\mu_A = \mu_B$, relația (71) va căpăta o formă mai simplă:

$$(\sigma_A^2 - \sigma_B^2) \ln \left(\frac{1-p}{p} \cdot \frac{\sigma_A}{\sigma_B} \right) < 0. \quad (73)$$

Această inegalitate este satisfăcută, de pildă atunci când σ_A și σ_B (care se consideră, prin convenție, strict pozitive) sunt în relația $\sigma_A > \sigma_B$, iar $\frac{1-p}{p} < \frac{\sigma_B}{\sigma_A}$. Se poate constata imediat că a două inegalități este echivalentă cu $p > \frac{\sigma_A}{\sigma_A + \sigma_B}$. Se observă că această ultimă fracție are valori în intervalul $(1/2, 1)$, fiindcă $\sigma_A > \sigma_B$.

Similar, dacă $\sigma_A < \sigma_B$, atunci pentru a satisface inegalitatea (73) se impune condiția $\frac{1-p}{p} > \frac{\sigma_B}{\sigma_A}$, care revine la $p < \frac{\sigma_A}{\sigma_A + \sigma_B}$. Această ultimă fracție ia valori în intervalul $(0, 1/2)$, însăcă $\sigma_A < \sigma_B$.

Consecință: În acest caz (c), nu există separator decizional. Una din cele două zone de decizie este vidă, iar cealaltă zonă coincide cu multimea tuturor numerelor reale, \mathbb{R} .

17.

(Algoritmul Bayes Naiv gaussian:
deducerea [formei liniare a] regulei de decizie în cazul
matricelor de covarianță diagonale și identice,
i.e., $\sigma_{i0} = \sigma_{ii}$, pentru $i = 1, \dots, d$)

■ CMU, 2009 spring, Ziv Bar-Joseph, HW2, pr. 2

Considerăm un model de tip Bayes Naiv cu două clase ($Y \in \{0, 1\}$), definit pe spațiul real \mathbb{R}^d al atributelor de intrare X_1, \dots, X_d . Presupunem că în acest model distribuția [comună] condiționată $X|Y = 0$, unde $X = (X_1, \dots, X_d) \in \mathbb{R}^d$, poate fi definită ca un vector de distribuții gaussiene unidimensionale independente și-l vom desemna prin notația

$$Gaussian(\mu_0 = <\mu_{10}, \dots, \mu_{d0}>, \sigma = <\sigma_1, \dots, \sigma_d>)$$

și analog pentru $X|Y = 1$:

$$\text{Gaussian}(\mu_1 = \langle \mu_{11}, \dots, \mu_{d1} \rangle, \sigma = \langle \sigma_1, \dots, \sigma_d \rangle).$$

Observați că intrările X_1, \dots, X_d au — la condiționare în raport cu clasa — medii diferite dar varianțe (de fapt, matrice de covarianță, diagonale) identice pentru ambele clase.

În acest exercițiu vă vom arăta că, în modelul specificat mai sus, probabilitatea condiționată $P(Y = 1|X = x)$, unde $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, se poate scrie ca valoare a unei funcții sigmoidale / „logistice“, $f(x) = \frac{1}{1 + e^{-(w_0 + w \cdot x)}}$, cu parametrii $w_0 \in \mathbb{R}$ și $w = (w_1, \dots, w_d) \in \mathbb{R}^d$ aleși în mod convenabil.¹³⁴

a. Folosiți regula lui Bayes (compusă cu formula probabilității totale) pentru a scrie $P(Y = 1|X = x)$ sub forma unei fracții. Împărțiți atât numărătorul cât și numitorul fracției astfel obținute cu expresia de la numărător.

b. La punctul a ar fi trebuit să ajungeți la un rezultat de forma

$$\frac{1}{1 + f(x, X, Y)},$$

unde f este o anumită funcție având argumentele x, X și Y . Folosind formula $e^{\ln a} = a$ (valabilă pentru orice $a > 0$), putem forța punerea acestei fracții într-o formă apropiată de funcția sigmoidală:

$$\frac{1}{1 + e^{\ln f(x, X, Y)}}.$$

Vă cerem să scrieți sub o astfel de formă rezultatul de la punctul a.

c. Explicitați presupoziția Bayes „naivă“ în cadrul modelului probabilist dat. Apoi folosiți-o pentru a converti exponentul care apare în fracția rezultată la punctul b la o sumă de forma

$$\ln g(Y) + \sum_{i=1}^d \ln h(x_i, X_i, Y).$$

Precizați expresiile funcțiilor g și h .

d. Acum folosiți specificul modelului dat — și anume, de tip gaussian, având pentru componentele condiționate (și anume, variabilele aleatoare condiționate $X_i|Y = 1$, respectiv $X_i|Y = 0$, pentru $i = 1, \dots, n$) medii diferite dar varianțe egale —, pentru a aduce expresia de la punctul c la o formă mai convenabilă.

e. Rescriind $P(Y = 1|X = x)$ conform expresiilor obținute la punctele b și d, rezultatul ar trebui să semene cu un alt model pe care l-am întâlnit la curs. Care anume? Exprimăți parametrii aceluia model în raport cu $P(Y = 1)$, μ_{i0} , μ_{i1} și σ_i , cu $i = 1, \dots, d$.

Răspuns:

a. Procedând conform cerințelor, vom scrie:

$$\begin{aligned} P(Y = 1|X = x) &= \frac{P(X = x|Y = 1) P(Y = 1)}{\sum_{y \in \{0,1\}} P(X = x|Y = y) P(Y = y)} \\ &= \frac{1}{1 + \frac{P(X = x|Y = 0) P(Y = 0)}{P(X = x|Y = 1) P(Y = 1)}}. \end{aligned}$$

¹³⁴În consecință, [se poate arăta imediat că] regula de decizie a clasificatorului Bayes Naiv pentru acest model este de tip *liniar*.

Considerând $f(x, X, Y) = \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)}$, se observă că ultima expresie obținută are într-adevăr forma $\frac{1}{1 + f(x, X, Y)}$.

b. Folosind formula $a = e^{\ln a}$, obținem:

$$P(Y = 1|X = x) = \frac{1}{1 + \exp\left(\ln \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)}\right)}.$$

c. Conform presupozitiei specifice algoritmului Naive Bayes, vom considera că atributele X_i sunt independente condițional două câte două în raport cu variabila de ieșire. Așadar, vom avea egalitățile următoare:

$$\begin{aligned} P(X = x|Y = 1) &= \prod_{i=1}^d P(X_i = x_i|Y = 1) \\ P(X = x|Y = 0) &= \prod_{i=1}^d P(X_i = x_i|Y = 0) \end{aligned}$$

Prin urmare, vom scrie argumentul funcției $\exp()$ din fracția de la punctul b astfel:

$$\ln \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)} = \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \ln \frac{P(X_i = x_i|Y = 0)}{P(X_i = x_i|Y = 1)}.$$

În consecință, funcțiile g și h care au fost cerute în enunț vor fi:

$$g(Y) = \frac{P(Y = 0)}{P(Y = 1)} \quad h(x_i, X_i, Y) = \frac{P(X_i = x_i|Y = 0)}{P(X_i = x_i|Y = 1)} \text{ pentru } i = 1, \dots, d.$$

d. Înținând cont de specificul gaussian al modelului din enunț, vom putea scrie rezultatul de la punctul b astfel:

$$\begin{aligned} &\ln \frac{P(X = x|Y = 0)P(Y = 0)}{P(X = x|Y = 1)P(Y = 1)} \\ &= \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \ln \left(\frac{\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_i - \mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_i - \mu_{i1})^2}{2\sigma_i^2}\right)} \right) \\ &= \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \left(\frac{(x_i - \mu_{i1})^2}{2\sigma_i^2} - \frac{(x_i - \mu_{i0})^2}{2\sigma_i^2} \right) \\ &= \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \frac{2x_i(\mu_{i0} - \mu_{i1}) + (\mu_{i1}^2 - \mu_{i0}^2)}{2\sigma_i^2} \\ &= \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \left(\frac{x_i(\mu_{i0} - \mu_{i1})}{\sigma_i^2} + \frac{(\mu_{i1}^2 - \mu_{i0}^2)}{2\sigma_i^2} \right) \\ &= \ln \frac{P(Y = 0)}{P(Y = 1)} + \sum_{i=1}^d \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} + \sum_{i=1}^d \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i. \end{aligned}$$

e. Notând în expresia obținută la punctul d

$$w_0 = \ln \frac{P(Y=0)}{P(Y=1)} + \sum_{i=1}^d \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \quad \text{și} \quad w_i = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} \text{ pentru } i = 1, \dots, d$$

și apoi revenind la expresia de la punctul b, vom putea scrie:

$$P(Y=1|X=x) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^d w_i x_i)}.$$

Așadar, expresia probabilității a posteriori este exact cea a funcției sigmoidale, $\frac{1}{1 + e^{-x}}$. Rezultatul seamănă foarte bine cu modelul de *regresie logistică*.¹³⁵

18.

(Algoritmul Bayes Optimal gaussian:
raportul față de regresia logistică
în cazul $\Sigma_0 = \Sigma_1$)

■ CMU, 2011 spring, Tom Mitchell, HW2, pr. 2.2

În cele ce urmează, vom considera:

1. Y , o variabilă booleană care urmează o distribuție de tip Bernoulli, cu parametrul $\pi = P(Y=1)$, ceea ce implică $P(Y=0) = 1 - \pi$;
2. $X = (X_1, X_2, \dots, X_d)^\top$, un vector de variabile aleatoare care nu sunt independente condițional în raport cu variabila Y , probabilitatea [comună și] condiționată $P(X|Y=k)$ urmând o *distribuție gaussiană multidimensională*, $\mathcal{N}(\mu_k, \Sigma)$, unde $k \in \{0, 1\}$.

Rețineți faptul că μ_k , media acestei distribuții multidimensionale, este un vector-coloană (altfel spus, o matrice $d \times 1$) și există două astfel de medii, câte una pentru fiecare dintre cele două valori ale variabilei Y . De asemenea, Σ , matricea de covarianță are dimensiunea $d \times d$, iar ea nu depinde de valorile variabilei Y .¹³⁶

În rezolvarea problemei, veți folosi funcția de densitate [de probabilitate] a distribuției gaussiene multidimensionale în notație matricială.¹³⁷

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1} (x - \mu)\right),$$

unde simbolul \top desemnează operația de transpunere a matricelor.

Răspundeți la următoarea întrebare:

Este oare distribuția $P(Y|X)$ corespunzătoare acestui clasificator de tip Bayes Optimal gaussian (nu naiv!) de aceeași formă cu cea a *regresiei logistică*?

Sugestie: Rafinați expresia distribuției probabiliste $P(Y|X)$.

Observație: La problema 17 am arătat că în cazul (particular!) în care intrările X_1, X_2, \dots, X_d sunt independente condițional în raport cu ieșirea Y — ceea ce, conform

¹³⁵Pentru o introducere la modelul regresiei logistice, vedeți problema 12 de la capitolul *Metode de regresie*.

¹³⁶Altfel spus, presupunând că Σ_k , pentru $k \in \{0, 1\}$, desemnează matricea de covarianță a distribuției gaussiene multidimensionale $\mathcal{N}(\mu_k, \Sigma_k)$, atunci considerăm că $\Sigma_0 = \Sigma_1$.

¹³⁷Vedeți problema 34 de la capitolul de *Fundamente*.

problemei 31 de la capitolul de *Fundamente*, este echivalent cu a spune că matricea Σ este diagonală —, răspunsul la întrebarea pusă în enunț este pozitiv.

Răspuns:

Vom demara calculele în maniera standard (adică, similar cu prima parte a rezolvării problemelor 14 și 17):

$$\begin{aligned} P(Y = 1|X) &= \frac{P(X|Y = 1) P(Y = 1)}{P(X|Y = 1) P(Y = 1) + P(X|Y = 0) P(Y = 0)} \\ &= \frac{1}{1 + \frac{P(Y = 0) P(X|Y = 0)}{P(Y = 1) P(X|Y = 1)}} = \frac{1}{1 + \exp\left(\ln \frac{P(Y = 0) P(X|Y = 0)}{P(Y = 1) P(X|Y = 1)}\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{P(Y = 0)}{P(Y = 1)} + \ln \frac{P(X|Y = 0)}{P(X|Y = 1)}\right)}. \end{aligned}$$

Acum ne vom concentra atenția asupra termenului $\ln \frac{P(X|Y = 0)}{P(X|Y = 1)}$, ținând cont de faptul că $X|Y = 0 \sim \mathcal{N}(\mu_0, \Sigma)$ și $X|Y = 1 \sim \mathcal{N}(\mu_1, \Sigma)$:

$$\begin{aligned} \ln \frac{P(X|Y = 0)}{P(X|Y = 1)} &= \ln \frac{\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}}}{\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}}} + \ln \exp\left(\frac{1}{2} \left[(X - \mu_1)^\top \Sigma^{-1} (X - \mu_1) - (X - \mu_0)^\top \Sigma^{-1} (X - \mu_0) \right]\right) \\ &= \frac{1}{2} \left[(X - \mu_1)^\top \Sigma^{-1} (X - \mu_1) - (X - \mu_0)^\top \Sigma^{-1} (X - \mu_0) \right] \\ &= \frac{1}{2} \left[-X^\top \Sigma^{-1} \mu_1 - \mu_1^\top \Sigma^{-1} X + \mu_1^\top \Sigma^{-1} \mu_1 + X^\top \Sigma^{-1} \mu_0 + \mu_0^\top \Sigma^{-1} X - \mu_0^\top \Sigma^{-1} \mu_0 \right] \\ &= \frac{1}{2} \left[\mu_1^\top \Sigma^{-1} \mu_1 - \mu_0^\top \Sigma^{-1} \mu_0 + X^\top \Sigma^{-1} (\mu_0 - \mu_1) + (\mu_0^\top - \mu_1^\top) \Sigma^{-1} X \right] \\ &= \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1 - \frac{1}{2} \mu_0^\top \Sigma^{-1} \mu_0 + (\mu_0 - \mu_1)^\top \Sigma^{-1} X. \end{aligned}$$

Remarcați faptul că $((\mu_0^\top - \mu_1^\top) \Sigma^{-1} X)^\top = ((\mu_0 - \mu_1)^\top \Sigma^{-1} X)^\top = X^\top (\Sigma^{-1})^\top (\mu_0 - \mu_1) = X^\top (\Sigma^\top)^{-1} (\mu_0 - \mu_1) = X^\top \Sigma^{-1} (\mu_0 - \mu_1)$, întrucât matricea Σ^{-1} este simetrică (ca urmare a faptului că Σ însăși, ca matrice de covarianță, este simetrică; vedeți problema 20 de la capitolul de *Fundamente*).

Prin urmare,

$$\begin{aligned} P(Y = 1|X) &= \frac{1}{1 + \exp\left(\ln \frac{1 - \pi}{\pi} + \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1 - \frac{1}{2} \mu_0^\top \Sigma^{-1} \mu_0 + (\mu_0 - \mu_1)^\top \Sigma^{-1} X\right)} \\ &= \frac{1}{1 + \exp(w_0 + w^\top X)}, \end{aligned}$$

cu $w_0 = \ln \frac{1 - \pi}{\pi} + \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1 - \frac{1}{2} \mu_0^\top \Sigma^{-1} \mu_0$ și $w = \Sigma^{-1} (\mu_0 - \mu_1)$. Evident, w_0 este un număr real (constant), iar w un vector-coloană (mai precis, o matrice de dimensiune $d \times 1$).

În concluzie, distribuția probabilistă $P(Y|X)$ are (și în acest caz!) aceeași formă cu cea din modelul regresiei logistice.

19.

(Clasificatorul Bayes Optimal Gaussian:
natura separatorului decizional în cazul în care $\Sigma_0 \neq \Sigma_1$)

■ Stanford, 2014 fall, Andrew Ng, midterm, pr. 2.b

Fie setul de date de antrenament $\{(x_1, y_1), \dots, (x_n, y_n)\}$, cu x_i vectori-coloană din \mathbb{R}^d și $y_i \in \{0, 1\}$ pentru orice i . Algoritmul Bayes Optimal gaussian estimează următorii parametri: $\phi \in (0, 1)$, vectorii-coloană μ_0 și μ_1 din \mathbb{R}^d , precum și matricele de covarianță Σ_0 și Σ_1 de dimensiune $d \times d$, corespunzători următoarelor distribuții:

$$\begin{aligned} p(y) &= \phi^y(1 - \phi)^{1-y}, \text{ unde } \phi = p(y = 1) \\ p(x|y = 0) &= \frac{1}{(2\pi)^{d/2}|\Sigma_0|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_0)^\top \Sigma_0^{-1}(x - \mu_0)\right) \\ p(x|y = 1) &= \frac{1}{(2\pi)^{d/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^\top \Sigma_1^{-1}(x - \mu_1)\right) \end{aligned}$$

Vă readucem aminte *regula de decizie* a acestui clasificator:

$$y = 1 \text{ dacă } p(y = 1|x) \geq p(y = 0|x) \text{ și } y = 0 \text{ în caz contrar.} \quad (74)$$

Arătați că atunci când $\Sigma_0 \neq \Sigma_1$, *separatorul decizional* determinat de algoritmul Bayes Optimal gaussian este de ordin pătratic, adică inegalitatea dintre probabilitățile condiționate a posteriori, $p(y = 1|x) \geq p(y = 0|x)$, este echivalentă cu o inegalitate numerică de forma

$$x^\top Ax + B^\top x + C \geq 0, \quad (75)$$

unde A este o anumită matrice de dimensiune $d \times d$, cu $A \neq 0$ (matricea nulă de dimensiune $d \times d$), $B \in \mathbb{R}^d$ (un anumit vector-coloană) și $C \in \mathbb{R}$ (o anumită constantă). Veți specifica în mod clar valorile pentru A , B și C .

Răspuns:

Elaborând relația dintre probabilitățile care determină decizia luată de către clasificatorul Bayes Optimal gaussian pentru o instanță oarecare de test x (vedeți relația (74)), putem obține următoarele echivalențe:

$$\begin{aligned} p(y = 1|x) \geq p(y = 0|x) &\Leftrightarrow \ln p(y = 1|x) \geq \ln p(y = 0|x) \\ &\Leftrightarrow \ln p(y = 1|x) - \ln p(y = 0|x) \geq 0 \Leftrightarrow \ln \frac{p(y = 1|x)}{p(y = 0|x)} \geq 0 \\ &\stackrel{F. Bayes}{\Leftrightarrow} \ln \frac{p(x|y = 1)p(y = 1)}{p(x|y = 0)p(y = 0)} \geq 0 \Leftrightarrow \ln \frac{p(y = 1)}{p(y = 0)} + \ln \frac{p(x|y = 1)}{p(x|y = 0)} \geq 0 \\ &\Leftrightarrow \ln \frac{\phi}{1 - \phi} - \ln \frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}} - \frac{1}{2} \left((x - \mu_1)^\top \Sigma_1^{-1}(x - \mu_1) - (x - \mu_0)^\top \Sigma_0^{-1}(x - \mu_0) \right) \geq 0 \\ &\Leftrightarrow -\frac{1}{2} \left(x^\top (\Sigma_1^{-1} - \Sigma_0^{-1})x - 2(\mu_1^\top \Sigma_1^{-1} - \mu_0^\top \Sigma_0^{-1})x + \mu_1^\top \Sigma_1^{-1}\mu_1 - \mu_0^\top \Sigma_0^{-1}\mu_0 \right) \\ &\quad + \ln \frac{\phi}{1 - \phi} - \ln \frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}} \geq 0 \end{aligned}$$

$$\Leftrightarrow x^\top \left(\frac{1}{2} (\Sigma_0^{-1} - \Sigma_1^{-1}) \right) x + \left(\mu_1^\top \Sigma_1^{-1} - \mu_0^\top \Sigma_0^{-1} \right) x + \ln \frac{\phi}{1-\phi} + \ln \frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} + \frac{1}{2} \left(\mu_0^\top \Sigma_0^{-1} \mu_0 - \mu_1^\top \Sigma_1^{-1} \mu_1 \right) \geq 0. \quad (76)$$

Coroborând rezultatul pe care tocmai l-am obținut cu relația (74) din enunț, observăm că putem considera

$$\begin{aligned} A &= \frac{1}{2} (\Sigma_0^{-1} - \Sigma_1^{-1}) \\ B^\top &= \mu_1^\top \Sigma_1^{-1} - \mu_0^\top \Sigma_0^{-1} \Leftrightarrow \\ B &= (\mu_1^\top \Sigma_1^{-1} - \mu_0^\top \Sigma_0^{-1})^\top = (\Sigma_1^{-1})^\top \mu_1 - (\Sigma_0^{-1})^\top \mu_0 = \Sigma_1^{-1} \mu_1 - \Sigma_0^{-1} \mu_0 \\ C &= \ln \frac{\phi}{1-\phi} + \ln \frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}} + \frac{1}{2} \left(\mu_0^\top \Sigma_0^{-1} \mu_0 - \mu_1^\top \Sigma_1^{-1} \mu_1 \right). \end{aligned} \quad (77)$$

Se poate constata imediat că ipoteza $\Sigma_0 \neq \Sigma_1$ din enunț implică faptul că $\Sigma_0^{-1} \neq \Sigma_1^{-1}$, deci $A \neq 0$. Prin urmare, granița de separare (engl., decision boundary) determinată în acest caz de către algoritm Bayes Optimal gaussian este de ordin pătratic.

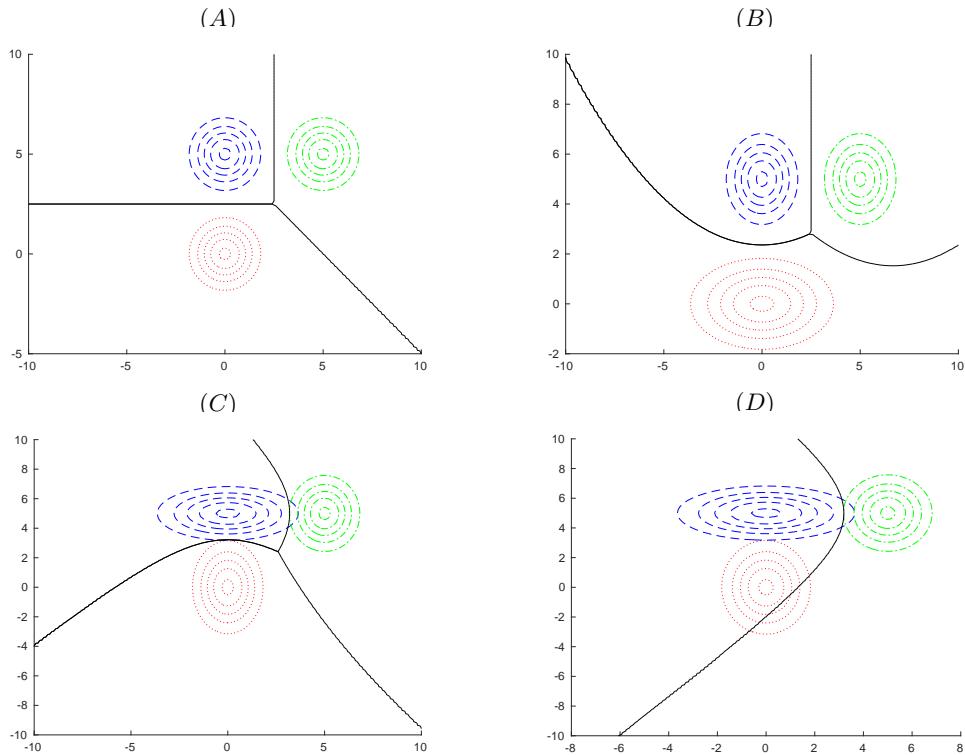
20. (Clasificatorul Bayes Optimal gaussian:
aplicare pe date din \mathbb{R}^2 ;
raportul cu regresia logistică)
*prelucrare de Liviu Ciortuz, după
MIT, 2001 fall, Tommi Jaakkola, HW2, pr. 2.abe*

a. Precizați separatorii decizionali determinați de algoritm Bayes Optimal gaussian pentru fiecare din cazurile de mai jos. Veți menționa în mod explicit, în fiecare caz în parte, tipul — liniar, pătratic (cerc, elipsă, parabolă, hiperbolă etc.), etc. — și ecuația acestor separatori decizionali.

(i.)	(ii.)	(iii.)
$P_0 = 0.5, P_1 = 0.5$	$P_0 = 0.995, P_1 = 0.005$	$P_0 = 0.5, P_1 = 0.5$
$\mu_0 = (1, 1)^\top$	$\mu_0 = (1, 1)^\top$	$\mu_0 = (1, 1)^\top$
$\mu_1 = (-1, -1)^\top$	$\mu_1 = (-1, -1)^\top$	$\mu_1 = (-1, -1)^\top$
$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$
(iv.)	(v.)	(vi.)
$P_0 = 0.5, P_1 = 0.5$	$P_0 = 0.5, P_1 = 0.5$	$P_0 = 0.5, P_1 = 0.5$
$\mu_0 = (0, 0)^\top$	$\mu_0 = (1, 1)^\top$	$\mu_0 = (-2, 1)^\top$
$\mu_1 = (0, 0)^\top$	$\mu_1 = (1, 1)^\top$	$\mu_1 = (1, 1)^\top$
$\Sigma_0 = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}$	$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1/2 \end{pmatrix}$
$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$	$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$	$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$

Sugestie: În cazul general, ecuația separatorului decizional determinat de clasificatorul Bayes Optimal gaussian este $x^\top Ax + B^\top x + C = 0$ (vedeți realția (75) de la problema 19), unde valorile pentru A , B și C au fost determinate prin relațiile (77).

- b. Arătați că este posibil ca întreg planul euclidian să fie alocat de către algoritmul Bayes Optimal gaussian¹³⁸ unei singure clase, situație în care, practic, nu avem separator decizional.
- c. Care dintre *separatorii decizionali* de la punctul a poate corespunde și unui model de tip *regresie logistică*? Justificați răspunsul dumneavoastră.
- d. Care dintre *zonele de decizie* din figurile de mai jos poate să corespundă unui model de tip *regresie logistică multinomială (softmax)*? Justificați răspunsul dumneavoastră.¹³⁹



Răspuns:

- a. Folosind formulele (75), se calculează coeficienții A , B și C din ecuația $x^\top Ax + B^\top x + C = 0$, pentru fiecare dintre cazurile date:¹⁴⁰

¹³⁸LC: De fapt, este suficient să ne gândim la cazul $\Sigma_0 = \Sigma_1$. (Menționăm că acest caz particular de clasificare bayesiană gaussiană este numit în literatura de specialitate *analiză gaussiană discriminativă*.)

¹³⁹Pentru o introducere în regresia logistică multinomială, vedeți problema 17 de la capitolul *Metode de regresie*.

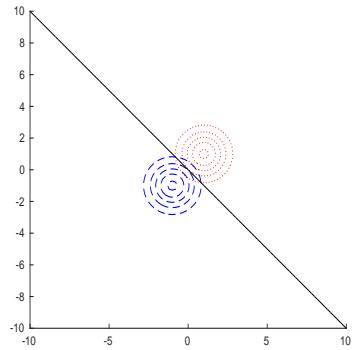
¹⁴⁰Am omis calculele efective, fiindcă nu sunt dificile. Invităm studentul să verifice că este capabil să obțină el însuși aceste rezultate.

i. $A = 0$ – matricea nulă de dimensiune 2×2 , $B^\top = (2, 2)$, $C = 0$.

Ecuatăia separatorului decizional este

$$2x_1 + 2x_2 = 0 \Leftrightarrow x_2 = -x_1.$$

Această ecuație reprezintă o dreaptă care trece prin originea sistemului de coordonate. Observați că această dreaptă este mediatoarea segmentului de dreaptă care unește punctele reprezentate de μ_1 și μ_2 (mediile celor două gaussiene).

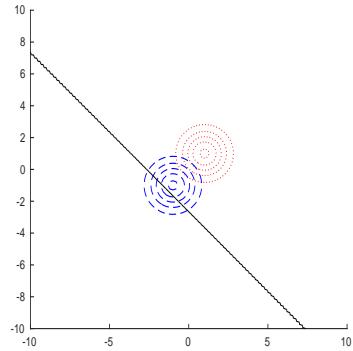


ii. $A = 0$, $B^\top = (2, 2)$, $C = \ln 199$.

Ecuatăia separatorului decizional este

$$2x_1 + 2x_2 + \ln 199 = 0 \Leftrightarrow x_2 = -x_1 - \underbrace{\ln \sqrt{199}}_{\approx 2.6466}.$$

Această ecuație reprezintă o dreaptă paralelă cu cea de la punctul precedent (i). Ea este situată acum mult mai aproape de punctul μ_1 — datorită faptului că în acest caz probabilitatea P_0 este mult mai mare decât probabilitatea P_1 .

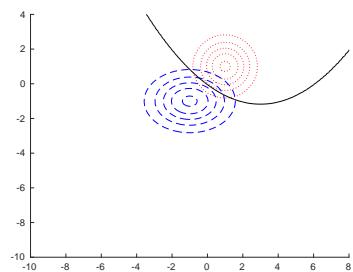


iii. $A = \begin{pmatrix} 1/4 & 0 \\ 0 & 0 \end{pmatrix}$, $B^\top = (-3/2, -2)$, $C = 1/4$.

În acest caz, spre deosebire de cazurile precedente, ecuația separatorului decizional este de ordin pătratic:

$$\frac{1}{4}x_1^2 - \frac{3}{2}x_1 - 2x_2 + \frac{1}{4} = 0 \Leftrightarrow x_2 = \frac{1}{8}x_1^2 - \frac{3}{4}x_1 + \frac{1}{8}.$$

Această ecuație reprezintă o parabolă, al cărei vârf are coordonatele $(3, -1)$.



iv. $A = \begin{pmatrix} -3/8 & 0 \\ 0 & 0 \end{pmatrix}$, $B^\top = 0$ – vectorul nul din \mathbb{R}^2 , $C = \ln 2$.

Ecuația separatorului decizional este

$$-\frac{3}{8}x_1^2 + \ln 2 = 0 \Leftrightarrow x_1^2 = \frac{8}{3} \ln 2.$$

Așadar, în acest caz, separatorul decizional este tot de ordin pătratic, dar este reprezentat de două drepte, având ecuațiile $x_1 = -\sqrt{\frac{8}{3} \ln 2} \approx -1.3595$ și respectiv $x_1 = \sqrt{\frac{8}{3} \ln 2} \approx 1.3595$.

Tinând cont de relația (76) care a fost demonstrată la problema 19, deducem că zona de decizie corespunzătoare clasei 1 este situată între aceste două drepte, iar zona de decizie corespunzătoare clasei 0 este în exteriorul lor.

$$v. A = \begin{pmatrix} 1/4 & 0 \\ 0 & 1/4 \end{pmatrix}, B^\top = (-1/2, -1/2), C = -\ln 2 + 1/2.$$

Ecuația separatorului decizional este

$$\begin{aligned} \frac{1}{4}(x_1^2 + x_2^2) - \frac{1}{2}(x_1 + x_2) - \ln 2 + \frac{1}{2} = 0 \Leftrightarrow \\ \frac{1}{4}(x_1^2 - 2x_1 + x_2^2 - 2x_2) = \ln 2 - \frac{1}{2} \Leftrightarrow \\ (x_1 - 1)^2 + (x_2 - 1)^2 = 4(\ln 2 - \frac{1}{2}) + 2. \end{aligned}$$

Această ecuație — tot de ordin pătratic — reprezintă un cerc cu centru în punctul $(1, 1)$ și raza de $2\sqrt{\ln 2} \approx 1.3862$.

$$vi. A = \begin{pmatrix} 1/4 & 0 \\ 0 & 7/8 \end{pmatrix}, B^\top = (5/2, -7/4), C = -\ln(1/4) + 21/8.$$

Ecuația separatorului decizional este

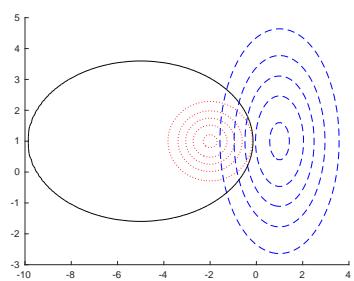
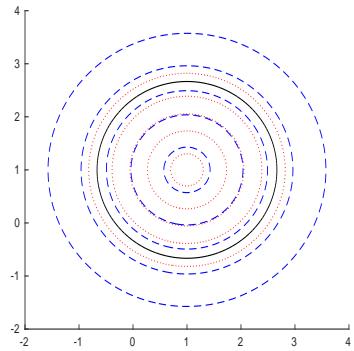
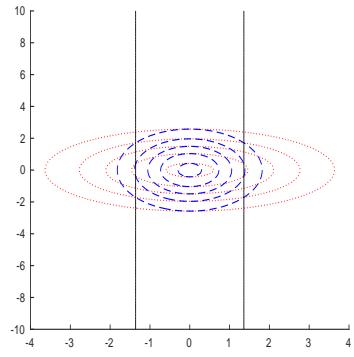
$$\begin{aligned} \frac{1}{4}x_1^2 + \frac{7}{8}x_2^2 + \frac{5}{2}x_1 - \frac{7}{4}x_2 - 2\ln 2 + \frac{21}{8} = 0 \Leftrightarrow \\ \frac{1}{4}(x_1^2 + 10x_1) + \frac{7}{8}(x_2^2 - 2x_2) = 2\ln 2 - \frac{21}{8} \Leftrightarrow \\ \frac{(x_1 + 5)^2}{4} + \frac{(x_2 - 1)^2}{\frac{8}{7}} = 2\ln 2 - \frac{21}{8} + \frac{25}{4} + \frac{7}{8}. \end{aligned}$$

Această ecuație — tot de ordin pătratic — reprezintă o elipsă care are centru de simetrie în punctul $(-5, 1)$.

b. Pornind de la regula de decizie $y_{GNB} = \operatorname{argmax}_{y \in \{0,1\}} P(Y = y|X = x)$, putem rationa astfel:

$$\begin{aligned} P(Y = 1|X = x) \geq P(Y = 0|X = x) &\stackrel{F. Bayes}{\Leftrightarrow} \\ P(X = x|Y = 1)P(Y = 1) \geq P(X = x|Y = 0)P(Y = 0). \end{aligned}$$

Dacă alegem ca distribuțiile gaussiene corespunzătoare claselor $Y = 1$ și $Y = 0$ să fie identice — adică să aibă aceiași parametri μ și Σ —, în inegalitatea de mai sus factorii



$P(X = x|Y = 1)$ și $P(X = x|Y = 0)$ vor fi egali. Prin urmare, atunci când $P(Y = 1) \neq P(Y = 0)$, va rezulta că întreg planul / spațiul va fi alocat unei singure clase, și anume acelei clase $y \in \{0, 1\}$ care are probabilitatea a priori ($P(Y = y)$) mai mare.

- c. Se știe că regresia logistică determină separatori decizionali liniari.¹⁴¹ Prin urmare, doar separatorii din cazurile i și ii pot corespunde (și) unor modele de regresie logistică.
- d. Justificarea este similară cu cea de la punctul c: doar în cazul (A) separatorii decizionali pot corespunde (și) unui model de regresie logistică multinomială / softmax.

21. (Clasificatori de tip Bayes gaussian,
cazul datelor cu atrbute de intrare corelate:
exemplificare în \mathbb{R}^2 ; comparație)

■ CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, HW2, pr. 5.c

În cazul bidimensional, putem vizualiza modul în care se comportă algoritmul Bayes Naiv gaussian atunci când atrbutele de intrare (adică, trăsăturile; engl., features) sunt corelate (engl., correlated).¹⁴²

Fie setul de date din figura (A) de mai jos, în care instanțele roșii sunt din clasa 0, iar instanțele albastre din clasa 1. Distribuțiile comune condiționale $((X_1, X_2)|Y = 0)$ și $((X_1, X_2)|Y = 1)$ sunt de tip gaussian bidimensional. Elipsele din figurile (B), (C) și (D) reprezintă curbe de izocontur pentru [diverse] distribuții condiționale asociate celor două clase. Centrele elipselor corespund mediilor, iar curbele de izocontur sunt situate la o distanță de două deviații standard față de medii.¹⁴³

- a. Care anume dintre perechile de elipse din figurile (B), (C) și (D) corespunde cel mai probabil distribuțiilor condiționale care au generat datele din figura (A)?
- b. Care anume dintre aceste elipse corespunde [cel mai probabil] estimărilor de parametri făcute de către algoritmul Bayes Naiv gaussian?
- c. Dacă presupunem că probabilitățile a priori pentru cele două clase sunt egale, care dintre modelele (B), (C) și (D) va obține o acuratețe mai mare pe datele de antrenament? Care va fi natura separatorului decizional în acest caz?

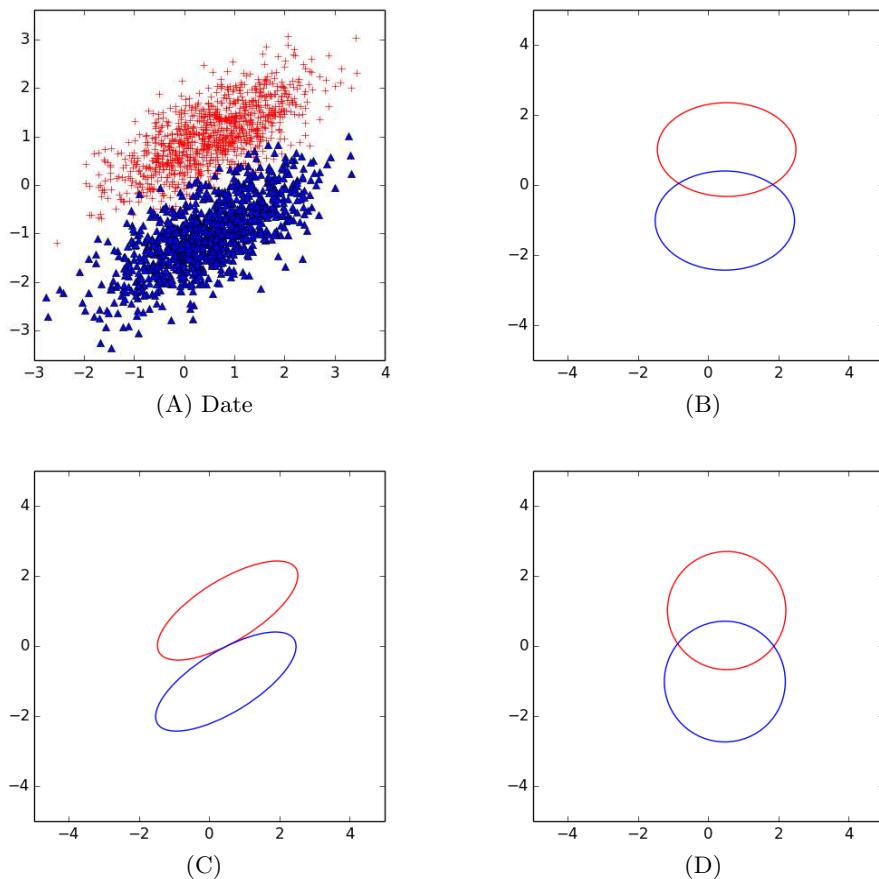
¹⁴¹Vedeți problema 12 de la capitolul *Metode de regresie*, în special relațiile (29), din care rezultă:

$$P(Y = 1|X = x) \geq P(Y = 0|X = x) \Leftrightarrow \sigma(z) \geq 1/2 \Leftrightarrow z \geq 0,$$

unde $\sigma(z) \stackrel{\text{def.}}{=} \frac{1}{1+e^{-z}}$, $z \stackrel{\text{not.}}{=} w_0 + \sum_{i=1}^d w_i x_i$ și $x \stackrel{\text{not.}}{=} (x_1, \dots, x_d)$. Așadar, în cazul regresiei logistice separatorul decizional are ecuația de forma $w_0 + \sum_{i=1}^d w_i x_i = 0$, deci este de tip liniar.

¹⁴²Pentru definiția noțiunii de *corelare* pentru două variabile aleatoare, vedeți problema 19 de la capitolul de *Fundamente*.

¹⁴³Mai exact, o astfel de curbă de izocontur este constituită din punctele x din planul euclidian pentru care $\Sigma^{-1/2}(x - \mu) = 2 \Leftrightarrow (x - \mu)^\top \Sigma^{-1}(x - \mu) = 4$, unde μ este media distribuției gaussiene considerate. Vedeți *Pattern Classification*, R. Duda, P. Hart, D. Stork, 2nd ed. (Wiley-Interscience, 2000), Appendix A, pag. 625.



Răspuns:

- a. Se observă în figura (A) că datele au fost generate de distribuții gaussiene bidimensionale având matrice de covariantă nediagonale și identice, însă având medii diferite. În plus, dacă notăm o instanță oarecare cu $x = (x_1, x_2)$ este evident că x_1 și x_2 nu sunt independente, ci există o anumită corelare între ele (mai precis, x_1 și x_2 sunt într-o relație de dependență de tip liniar). În consecință, curbele de izocontur pentru aceste distribuții sunt elipse identice ca mărime, având axe de simetrie neparallele cu axele sistemului de coordinate. Evident, doar desenul (C) corespunde acestei situații.

b. Clasificatorul Bayes Naiv gaussian presupune independentă condițională a celor două atribute (X_1 și X_2) în raport cu eticheta / variabila de ieșire (Y). Această independentă corespunde unor matrice de covariantă diagonale, respectiv unor curbe de izocontur reprezentate de elipse ale căror axe de simetrie sunt parallele cu axele sistemului de coordinate. Atât elipsele din desenul (B) cât și cele din desenul (D) satisfac aceste specificații, însă în cazul (D) elipsele sunt chiar cercuri, în vreme ce datele din figura (A) sunt dispuse în elipse având deviații standard diferite pe cele două axe de simetrie. Așadar, cazul (B) corespunde estimărilor făcute de clasificatorul Bayes Naiv gaussian pe aceste date.

c. Evident, cazul (C) furnizează cea mai mică eroare la antrenare, deci cea mai mare acuratețe. În acest caz, se lucrează cu algoritmul Bayes Optimal gaussian. Întrucât matricele de covarianță sunt egale, separatorul decizional este de tip liniar. (Pentru justificare riguroasă, vedeți rezultatul teoretic demonstrat la problema 18.)

22. (Algoritmul Bayes Naiv [gaussian] vs. regresia logistică — comparații)

CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 1.f, 3.b-d

Fie un set de date caracterizate de atributele X_1, \dots, X_n (pe care le vom considera ca fiind fie toate Bernoulli, fie toate gaussiene) și de eticheta Y .¹⁴⁴ Modelul Bayes Naiv [eventual de tip gaussian] va fi identificat în continuare cu abrevierea NB, iar modelul regresiei logistice cu LR.

- a. Presupunem că datele satisfac presupozitia de independentă condițională de tip Bayes Naiv. Atunci când numărul de exemple de antrenament tinde la infinit, care dintre cei doi clasificatori va produce rezultate mai bune, NB sau LR? Justificați.
- b. Presupunem acum că datele *nu* satisfac presupozitia de independentă condițională de tip Bayes Naiv. Ne punem aceeași întrebare ca mai sus: atunci când numărul de exemple de antrenament tinde la infinit, care dintre cei doi clasificatori va produce rezultate mai bune, NB sau LR? Justificați.
- c. Este oare posibil să calculăm distribuția $P(X)$ cu ajutorul parametrilor estimati de către algoritmul Bayes Naiv? Explicați în mod succint.
- d. Este oare posibil să calculăm distribuția $P(X)$ cu ajutorul parametrilor w calculați de către regresia logistică? Explicați în mod succint.

Răspuns:

- a. Regresia logistică este un clasificator probabilist de tip *discriminativ*, adică aproximăza / modelează $P(Y|X)$ cu ajutorul funcției logistice de argument $w \cdot X$ (unde w este vectorul de parametri, $w \in \mathbb{R}^d$, sau $w \in \mathbb{R}^{d+1}$ dacă extinem fiecare instanță X cu componenta $X_0 = 1$). În contrast cu acesta, NB este un clasificator probabilist de tip *generativ*, deci calculează distribuțiile $P(X|Y = y)$ (care în cazul GNB sunt de tip gaussian multidimensional) și de asemenea $P(Y)$, estimând parametrii acestor distribuții.

Atunci când numărul de instanțe tinde la infinit, pe de o parte aproximarea calculată de regresia logistică pentru distribuția $P(Y|X)$ tinde la distribuția reală $P(Y|X)$, iar pe de altă parte estimările făcute de clasificatorul Bayes Naiv pentru distribuțiile $P(Y)$ și $P(X|Y)$ vor tinde la distribuțiile reale corespunzătoare, dat fiind (în cazul lui $P(X|Y)$) că datele de antrenament satisfac presupozitia de independentă condițională. Corespondența dintre distribuțiile reale $P(Y|X)$ (pe de o parte) și $P(Y)$ și $P(X|Y)$ (pe de altă parte) este dată de formula lui Bayes. Prin urmare, în aceste condiții cei doi clasificatori vor produce rezultate echivalente.

¹⁴⁴Așa este cazul problemei 14, în care variabilele condiționate $X_i|Y$ sunt de tip Bernoulli și sunt independente condițional două câte două, sau cel al problemei 17, în care toate variabilele condiționate, $X_i|Y$ pentru $i = 1, \dots, n$ urmează distribuții gaussiene — având varianțele $\sigma_{i0} = \sigma_{i1}$ — și sunt, de asemenea, independente condițional două câte două. (Similar este și cazul problemei 44, care constituie o combinație a precedentelor două tipuri.)

- b. Regresia logistică va produce rezultate mai bune, fiindcă ea nu lucrează cu presupozitia de independență condițională. (Vedeți pe de o parte problema 32 de la capitolul *Metode de regresie*, iar pe de altă parte problema 10 de la prezentul capitol.)
- c. Da, algoritmul Bayes Naiv este un clasificator de tip *generativ* (engl., generative classifier). Putem calcula $P(X)$ prin „marginalizarea“ distribuției condiționate $P(X|Y)$ în raport cu eticheta / clasa Y , și anume: $P(X) = \sum_y P(X|Y = y) \cdot P(Y = y)$.
- d. Nu, nu este posibil. Așa cum am precizat la punctul a, regresia logistică estimează $P(Y|X)$ (nu $P(X|Y)$ și $P(Y)$, cum calculează clasificatorii de tip Bayes Naiv).

23. (Clasificarea bayesiană gaussiană vs. regresia logistică:
Adevărat sau Fals?)

*CMU, 2010 fall, Aarti Singh, midterm, pr. 1.2
CMU, (?) 15-781, midterm example questions, pr. 1.d*

- a. Corespondența dintre regresia logistică și clasificatorul Bayes Naiv de tip gaussian înseamnă — în cazul în care matricele de covarianță corepunzătoare claselor sunt toate egale cu matricea-identitate¹⁴⁵ — că există o corespondență 1-la-1 între parametrii celor doi clasificatori.
- b. Presupunând că lucrăm cu un număr fix de atribute, putem învăța un clasificator Bayes Optimal de tip gaussian în timp liniar în raport cu numărul de instanțe din setul de date de antrenament.

Răspuns:

- a. Fals. Se poate preciza de la început că deși cei doi clasificatori învață separatori decizionali care au aceeași formă (și anume, o formă liniară, vedeți problema 17) nu rezultă în mod neapărat că pe un același set de date cei doi separatori decizionali învățați coincid. Faptul că matricele de covarianță sunt, toate, matrice identitate / diagonale înseamnă că presupozitia de independență condițională este satisfăcută.¹⁴⁶ Suntem, aşadar, în condiții similare cu cele de la problema 22.a, însă aici nu avem neapărat satisfăcută ipoteza că numărul de instanțe de antrenament tinde la infinit, caz în care rezultatele de clasificare furnizate de LR și NB ar fi echivalente. Așadar, nu există (în general) o corespondență de tip 1-la-1 între parametrii w_{LR} calculați de regresia logistică¹⁴⁷ și parametrii w_{NB} corespunzători clasificatorului Bayes Naiv gaussian.¹⁴⁸

Observație: Rezultatul acesta este valabil și în cazul algoritmului Bayes Naiv cu variabile categoriale.

¹⁴⁵LC: Chiar mai general, putem considera că aceste matrice sunt diagonale.

¹⁴⁶Vedeți problema 31 de la capitolul de *Fundamente*.

¹⁴⁷Vedeți problema 12 de la capitolul *Metode de regresie*.

¹⁴⁸În primul rând, din punctul de vedere al terminologiei, dacă ne referim la *parametrii* calculați de NB ca fiind parametrii distribuțiilor $P(X|Y = y)$ și $P(Y)$, este imediat că nu există o corespondență 1-la-1 între aceștia și parametrii w_{LR} calculați de LR, care servesc la estimarea / aproximarea distribuției $P(Y|X)$.

b. Adevărat. În cazul clasificatorului Bayes Optimal de tip gaussian, regula de calcul pentru clasificarea unei instanțe noi $x \in \mathbb{R}^d$ este următoarea:

$$\begin{aligned} y_{GNB} &\stackrel{\text{def.}}{=} \operatorname{argmax}_{y \in Val(Y)} P(Y = y | X = x) = \operatorname{argmax}_{y \in Val(Y)} \frac{P(X = x | Y = y)P(Y = y)}{P(X = x)} \\ &= \operatorname{argmax}_{y \in Val(Y)} P(X = x | Y = y)P(Y = y) = \operatorname{argmax}_{y \in Val(Y)} \mathcal{N}(x; \mu_y, \Sigma_y)P(Y = y). \end{aligned}$$

Așadar, pentru fiecare $y \in Val(Y)$ vom avea de estimat câte o pereche de parametri, care caracterizează o distribuție gaussiană multidimensională: vectorul de medii μ_y și matricea de covarianță Σ_y . Conform problemei 48 de la capitolul de *Fundamente*, estimările celor doi parametri sunt media la eșantionare și respectiv matricea de covarianță la eșantionare, deci se calculează în timp liniar în raport cu numărul de instanțe de antrenament.

2.2 Probleme propuse

2.2.1 Ipoteze de probabilitate maximă a posteriori (MAP)

24. (Formula lui Bayes; inferențe statistice; ilustrarea noțiunii de ipoteze MAP)
CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW1, pr. 5

Imaginează-ți că te află în fața a trei cutii, care sunt etichetate cu literele A, B, C . Două dintre ele sunt goale, iar una conține un premiu. Tu nu știi în care dintre ele se află premiul; trebuie să ghicești. Procedezi în felul următor:

Mai întâi alegi la întâmplare o cutie X (să zicem că $X = A$). Totuși, chiar înainte de a deschide cutia X observi că altcineva, înaintea ta, a deschis cutia Y (să zicem că $Y = B$). Ai dreptul să privești în cutia Y , ca să vezi dacă ea conține sau nu premiul. Dacă ea nu conține premiul, vei avea dreptul să schimbi alegerea pe care ai făcut-o inițial.

În vederea luării unei decizii cât mai bune, îți se comunică *strategia* după care a fost aleasă cutia Y , și anume, folosind una dintre următoarele trei *variante*:

a. Dacă cutia pe care ai ales-o inițial conține premiul, atunci cutia Y este aleasă cu probabilitate de $1/2$ una dintre cele două cutii goale (diferite de cutia X). Dacă X este vidă, atunci Y se alege ca fiind cutia goală diferită de X .

b. Se alege aleatoriu cu probabilitate de $1/2$ una dintre cutiile diferite de cea pe care ai ales-o tu inițial, X . (În consecință, cutia Y poate sau nu să conțină premiul. Dacă Y conține premiul, ai pierdut jocul.)

c. Se alege în mod aleatoriu cu probabilitate de $1/2$ una dintre cutiile goale. (Deci este posibil ca $Y = X$. Așadar, în cazul $Y = X$ observi că a fost deschisă anterior chiar cutia X . În continuare vei putea alege una dintre celelalte două cutii.)

Considerând (pentru simplitate) că $X = A$, $Y = B$, iar cutia B este vidă, pentru fiecare dintre cele trei *variante* de mai sus decide ce cutie ar trebui să alegi în final pentru a-ți maximiza şansele de a obține premiul. Justifică-ți decizia, elaborând calculul probabilistic aferent.

25.

(Adevărat sau Fals?)

CMU, 2002 fall, Andrew Moore, final exam, pr. 11.a

Fie D un set de exemple (date de antrenament), iar H o mulțime de ipoteze pentru (un algoritm de) învățare automată pe datele D . Precizați care este *valoarea de adevăr* a următoarelor afirmații:

$\text{argmax}_{h \in H} P(D|h)$ este ipoteză de probabilitate maximă a posteriori, [iar]
 $\text{argmax}_{h \in H} P(h|D)$ este ipoteză de verosimilitate maximă.

2.2.2 Algoritmii Bayes Naiv și Bayes Optimal

26.

(Algoritmii Bayes Naiv și Bayes Optimal; aplicare; numărul minimal de parametri de estimat)

Liviu Ciortuz, 2017, pornind de la setul de date din Machine Learning, Tom Mitchell, 1997, ch. Decision Trees, page 59

Considerăm următorul set de date de antrenament, în care variabila de ieșire este *Enjoy-Tennis*:

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

a. Determinați decizia luată de către algoritmul Bayes Naiv pentru instanța de test

$$X = \langle \text{Outlook} = \text{sunny}, \text{Temp} = \text{cool}, \text{Humidity} = \text{high}, \text{Wind} = \text{strong} \rangle,$$

precum și probabilitatea cu care este luată această decizie.

b. Care este numărul *minim* de parametri pe care trebuie să-l estimeze algoritmul Bayes Naiv pe aceste date [pentru a face apoi predicții pe un set oarecare de instanțe de test]? Dar în cazul clasificatorului Bayes Optimal?

c. Implementați algoritmul Bayes Naiv, iar apoi cu ajutorul acestei implementări calculați eroarea la antrenare și eroarea la CVLOO pe acest set de date.

27. (Algoritmul Bayes Naiv: aplicare)
CMU, 2004 fall, T. Mitchell Z. Bar-Joseph, midterm, pr. 6.a

A	B	C	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1

Se dă setul de date din tabelul alăturat, în care A, B, C sunt attribute (de intrare) binare, iar Y este atribut de ieșire.
Care va fi răspunsul algoritmului de clasificare Bayes Naiv pentru intrarea $A = 0, B = 0, C = 1$?

28. (Algoritmul Bayes Naiv și algoritmul Bayes Optimal: aplicare)
prelucrare de Liviu Ciortuz, după CMU, 2002 fall, Andrew Moore, final exam, pr. 4.b-e

A	B	C	nr. aparitii
0	0	1	3
0	1	0	1
0	1	1	4
1	0	0	5
1	1	0	2
1	1	1	1

- Se dă setul de date alăturat, cu A și B variabile de intrare, iar C variabilă de ieșire.
- Care este numărul minim de probabilități ce trebuie estimate pentru a putea construi după aceea (pe acest set de date) un clasificator de tip Bayes Naiv? Justificați.
 - Similar, pentru clasificatorul Bayes Optimal. Justificați.
 - Care este decizia clasificatorului Bayes Naiv pentru $A = 0, B = 1$? Precizați cu ce probabilitate este luată această decizie.
 - Care este decizia clasificatorului Bayes Optimal pentru $A = 0, B = 1$? Precizați cu ce probabilitate este luată această decizie.
 - Dacă rezultatele obținute la punctele c și d diferă (fie și numai în privința probabilităților cu care sunt luate deciziile), care este explicația? Justificați în mod riguros.

29. (Aplicarea algoritmului Bayes Naiv la clasificarea de texte)
Edinburgh, 2009 fall, C. Williams, V. Lavrenko, tutorial 2, pr. 2

Firma Whizzco decide să implementeze un clasificator de texte. Pentru început, ei vor să clasifice documente aparținând fie clasei *sport* fie clasei *politică*. Ei decid să reprezinte fiecare document ca un vector de attribute descriind prezența ori absența unor cuvinte-cheie:

goal, football, golf, defence, offence, wicket, office, strategy.

Datele de antrenament sunt reprezentate folosind o matrice în care fiecare linie este un vector de valori (0 sau 1) pentru cele 8 attribute.

```

xP=[1 0 1 1 1 0 1 1; % Politica
    0 0 0 1 0 0 1 1;
    1 0 0 1 1 0 1 0;
    0 1 0 0 1 1 0 1;
    0 0 0 1 1 0 1 1;
    0 0 0 1 1 0 0 1]

xS=[1 1 0 0 0 0 0 0; % Sport
     0 0 1 0 0 0 0 0;
     1 1 0 1 0 0 0 0;
     1 1 0 1 0 0 0 1;
     1 1 0 1 1 0 0 0;
     0 0 0 1 0 1 0 0;
     1 1 1 1 1 0 1 0]

```

Folosind algoritmul Bayes Naiv, care este probabilitatea cu care documentul $x = (1, 0, 0, 1, 1, 0, 1, 0)$ va fi clasificat ca aparținând clasei *politica*?

30. (Aplicarea algoritmului Bayes Naiv:
chestiunea valorilor lipsă (engl., missing values)
în datele de antrenament)

CMU, 2013 fall, A. Smola, G. Gordon, midterm practice, pr. 9

Fie setul de date de antrenament (x, y) și datele de test z :

$$\begin{aligned}
 x_1 &= (0, 0, 0, 1, 0, 0, 1) & y_1 &= 1 \\
 x_2 &= (0, 0, 1, 1, 0, 0, 0) & y_2 &= 1 \\
 x_3 &= (1, 1, 0, 0, 0, 1, 0) & y_3 &= -1 \\
 x_4 &= (1, 0, 0, 0, 1, 1, 0) & y_4 &= -1 \\
 \\
 z_1 &= (1, 0, 0, 0, 0, 1, 0) \\
 z_2 &= (0, 1, 1, 0, 0, 1, 1)
 \end{aligned}$$

Ce problemă va întâmpina clasificatorul Bayes Naiv pe aceste date?

(Indicație: Pentru ca răspunsul dumneavoastră să fie cât mai bine justificat, veți estima toți parametrii necesari și veți aplica algoritmul pe cele două instanțe de test. Veți nota attributele cu $A1, A2, \dots$)

La curs am prezentat un „remediu“ standard pentru o astfel de problemă. Precizați cum se numește „tehnica“ respectivă și aplicați-o pe aceste date. După aceea, veți aplica algoritmul Bayes Naiv pentru a clasifica instanțele de test z_1 și z_2 .

31. (Algoritmului Bayes Naiv:
calculul ratei medii a erorilor)

■ *CMU, 2010 fall, Aarti Singh, HW1, pr. 4.2*

Considerăm următoarea problemă de clasificare:

Fie variabila aleatoare $Y: Hike \in \{T, F\}$ care denotă faptul că Alice și Bob merg sau nu în drumeție în funcție de condițiile vremii: $X_1: Sunny \in \{T, F\}$ și $X_2: Windy \in \{T, F\}$.

Se presupune că au fost estimați următorii parametri:

$$\begin{aligned}
 P(Hike) &= 0.5 \\
 P(Sunny | Hike) &= 0.8, \quad P(Sunny | \neg Hike) = 0.7 \\
 P(Windy | Hike) &= 0.4, \quad P(Windy | \neg Hike) = 0.5
 \end{aligned}$$

De asemenea, se consideră că este satisfăcută presupozitia de independentă condițională a algoritmului Bayes Naiv.

- a. Care este probabilitatea (comună) ca Alice și Bob să meargă în drumeție atunci când vremea este însorită și bate vântul, adică

$$P(\text{Sunny} = T, \text{Windy} = T, \text{Hike} = T) = ?$$

Care este decizia luată de algoritmul Bayes Naiv în acest caz?

- b. Completați tabelul următor:

X_1	X_2	Y	$P(X_1, X_2, Y)$	$P_{NB}(Y X_1, X_2)$	<i>decizia algoritmului Bayes Naiv</i>
F	F	F			
F	F	T			
F	T	F			
F	T	T			
T	F	F			
T	F	T			
T	T	F			
T	T	T			

Observație: Calculele de la punctul a corespund ultimei linii din tabelul de mai sus.

- c. Care este rata medie a erorilor (engl., expected error rate) produse de algoritmul Bayes Naiv? Vă reamintim că această (rată) medie este definită ca fiind suma probabilităților $P(X_1, X_2, Y)$ pentru acele (triplete de) valori ale variabilelor X_1, X_2, Y pentru care decizia luată de algoritmul Bayes Naiv diferă de valoarea variabilei Y .

În cele ce urmează se presupune că se obțin mai multe informații despre vreme. Se introduce o nouă trăsătură $X_3: \text{Rainy} \in \{T, F\}$. Se presupune că în fiecare zi vremea poate fi fie *Rainy* fie *Sunny*, dar nu și *Rainy* și *Sunny*. Similar, se presupune că vremea nu poate fi într-o zi $\neg \text{Rainy}$ și $\neg \text{Sunny}$.

- d. În noile condiții, presupoziția de independentă condițională rămâne oare adevărată? Justificați.
e. Calculați $P(\text{Sunny} = T, \text{Windy} = T, \text{Rainy} = F, \text{Hike} = T)$.
f. Care este rata medie a erorilor produse de clasificatorul Bayes Naiv când se folosesc toate cele 3 atrbute de intrare?
g. S-a îmbunătățit performanța algoritmului Bayes Naiv prin adăugarea atrbutului *Rainy*? Explicați de ce.

32.

(Algoritmul Bayes Naiv:
calculul ratei medii a erorii – exemplificare;
comparație cu regresia logistică)
*prelucrare de Liviu Ciortuz, după
CMU, 2009 fall, Carlos Guestrin, HW1, pr. 4.1.4*

Considerăm o problemă de clasificare binară în care fiecare exemplu de antrenament are două atrbute binare $X_1, X_2 \in \{T, F\}$ și eticheta / clasa $Y \in \{T, F\}$. Presupunem că $P(Y = T) = 0.5$, iar $P(X_1 = T|Y = T) = 0.8$, $P(X_1 = F|Y = F) = 0.7$, $P(X_2 =$

$T|Y = T) = 0.5$ și $P(X_2 = F|Y = F) = 0.9$. (Se poate observa că atributul X_1 furnizează / constituie un indiciu întrucâtva mai puternic decât atributul X_2 în ce privește determinarea clasei unei instanțe oarecare.)

În cele ce urmează vom presupune că X_1 și X_2 sunt independente în raport cu Y .

a. Calculați probabilitățile $P(X_1 = F|Y = T)$, $P(X_1 = T|Y = F)$, $P(X_2 = F|Y = T)$ și $P(X_2 = T|Y = F)$. Asociați răspunsului dumneavoastră o justificare generală, sub forma unei formule din teoria probabilităților:

$$P(\neg A|B) = \dots, \text{ unde } A \text{ și } B \text{ sunt evenimente aleatoare oarecare.}$$

b. Scrieți regula de decizie a algoritmului Bayes Naiv pentru $X_1 = x_1$ și $X_2 = x_2$, justificând în mod succint obținerea ei.

c. Calculați rata medie a erorii produse de algoritmul Bayes Naiv, atunci când se folosesc ambele attribute, X_1 și X_2 . (Veți da în prealabil definiția ratei medii a erorii.) Este oare această rată mai bună decât în cazul în care se folosește un singur atribut (X_1 sau X_2)? De ce?

d. Să presupunem acum că se crează un nou atribut, X_3 , care este o copie exactă a lui X_2 . Așadar, pentru fiecare exemplu de antrenament, attributele X_2 și X_3 au aceeași valoare, $X_2 = X_3$. Răspundeți la următoarele întrebări:

- Sunt X_2 și X_3 independente condițional în raport cu Y ?
- Cât este rata medie a erorii pentru Bayes Naiv acum? (Atenție! Distribuția „adevărată“ a datelor nu s-a modificat.)
- Explicați ce se întâmplă cu algoritmul Bayes Naiv. Oare *regresia logistică* are aceeași problemă? Explicați de ce.

33. (Clasificare bayesiană: calculul ratei medii a erorilor pentru diversi clasificatori bayesieni)

prelucrare de L. Ciortuz, după CMU, 2004 fall, T. Mitchell Z. Bar-Joseph, HW3, pr. 1.2

Fie funcția $Y = (A \wedge B) \vee \neg(B \vee C)$, unde A, B și C sunt variabile aleatoare binare independente, fiecare dintre ele având posibilitatea să ia valoarea 0 cu probabilitate de 50%.

a. Câtă parametri trebuie să estimeze clasificatorul Bayes Naiv pentru a învăța funcția Y ? Enumerați acești parametri. Atenție: $P(\neg x)$ nu va fi socotit ca parametru dacă $P(x)$ a fost deja estimat ca parametru.

b. Care este rata medie a erorii la antrenare pentru clasificatorul Bayes Naiv la învățarea conceptului Y , presupunând că avem o infinitate de date de antrenament?

Indicație: Scrieți mai întâi tabela de adevară a funcției Y , apoi estimați valorile parametrilor (în sensul verosimilității maxime, MLE). Pentru conveniență, centralizați toate calculele făcute de către algoritmul Bayes Naiv într-un tabel. (Sau, altfel, puteți adăuga niște coloane suplimentare la tabela de adevară a funcției Y .)

Convenție: În cazul în care, pentru o setare oarecare a variabilelor A, B și C , cele două probabilități calculate de către algoritmul Bayes Naiv în vederea determinării valorii y_{NB} sunt egale, convenim că algoritmul va lua decizia $y_{NB} = 1$.

- c. Căți parametri trebuie să estimeze clasificatorul Bayes Optimal pentru a „învăța“ funcția Y ? Justificați în detaliu.
- d. Care este rata medie a erorii la antrenare pentru clasificatorul Bayes Optimal la învățarea conceptului Y , presupunând același lucru ca mai sus? *Atenție:* Nu este nevoie să calculați efectiv această rată; este suficient să indicați valoarea ei și să o justificați printr-un *raționament calitativ*.¹⁴⁹
- e. Considerăm un alt clasificator de tip Bayes, care presupune că A este independent de C , condiționat de B și Y — în contrast cu clasificatorul Bayes Naiv, care presupune că variabilele A , B și C sunt independente două câte două în raport cu Y . Arătați că acest clasificator Bayes va avea nevoie să estimeze mai puțini parametri decât clasificatorul Bayes Optimal la învățarea conceptului Y , și totuși va obține aceeași rată medie a erorii la antrenare (considerând că este valabilă aceeași presupozitie în legătură cu datele de antrenament).

Indicații:

- Scrieți tabela de adevăr a funcției Y .
- Folosind această tabelă de adevăr, dovediți că într-adevăr proprietatea de independență condițională formulată mai sus este satisfăcută. și anume: pentru fiecare pereche de valori b și y ale variabilelor aleatoare B și respectiv Y , arătați că

$$P(A = a, C = c | B = b, Y = y) = P(A = a | B = b, Y = y) \cdot P(C = c | B = b, Y = y),$$

pentru $\forall a \in Val(A)$ și $\forall c \in Val(C)$.

- Scrieți regula de decizie pentru acest nou clasificator de tip Bayes și arătați că ea este echivalentă cu regula de decizie a clasificatorului Bayes Optimal.
- Calculați numărul minim de parametri de estimat de către noul nostru clasificator Bayes și enumerați-i.

34.

(Algoritmul Bayes Naiv:
independența [condițională a] atributelor de intrare;
calculul ratei medii a erorii)

CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW3, pr. 1.1

Presupunem că A și B sunt variabile aleatoare binare independente, fiecare dintre ele având posibilitatea de a lua valoarea 0 cu o probabilitate de 50%.

Definiți o funcție booleană $y = f(A, B)$ în aşa fel încât variabila A să nu fie independentă de variabila B în raport cu y (văzut și el ca variabilă aleatoare), însă clasificatorul Bayes Naiv să producă o rată medie a erorii de 0% (presupunând că datele de antrenament sunt în număr infinit).

Demonstrați că acest clasificator are într-adevăr rata erorii de 0%.

¹⁴⁹LC: Totuși, este recomandabil să procedați aşa după ce în prealabil ati văzut ce valoare produce algoritmul Bayes Optimal pentru [măcar] una dintre combinațiile de valori ale variabilelor, de exemplu, $A = B = C = 0$.

35. (Algoritmul Bayes Naiv:
comparație cu alți clasificatori)
*prelucrare de L. Ciortuz, după
CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, midterm, pr. 6.b*

Considerăm un clasificator Bayes Naiv care lucrează pe un set de date descrise de atribuibile de intrare A și B și atributul de ieșire Y . (Exemplu: $Y = A \text{ XOR } B$). A și B sunt variabile aleatoare independente între ele.

- Este posibil ca, în situația aceasta, vreun alt clasificator — de exemplu, ID3, regresia logistică (eventual kernel-izată) sau SVM — să lucreze mai bine decât clasificatorul Bayes Naiv?
- Care este motivul?

36. (Comparație între clasificatorul Bayes Naiv
și algoritmul ID3)
CMU, 2010 fall, Ziv Bar-Joseph, midterm, pr. 5.b

Care dintre afirmațiile de mai jos sunt adevărate atât pentru clasificatorul Bayes Naiv cât și pentru algoritmul ID3 pentru învățarea de arbori de decizie? (Veți putea alege nu neapărat una singură dintre aceste afirmații.)

- În cazul ambilor clasificatori se presupune că orice pereche de atrbute X_i și X_j cu $i \neq j$ — văzute ca variabile aleatoare — sunt independente.
- În cazul ambilor clasificatori se presupune că orice pereche de atrbute X_i și X_j cu $i \neq j$ sunt dependente.
- În cazul ambilor clasificatori se presupune că orice pereche de atrbute sunt independente în raport cu eticheta (adică variabila care reprezintă clasa).
- În cazul ambilor clasificatori se presupune că orice pereche de atrbute sunt dependente în raport cu eticheta.

37. (Algoritmul Bayes Naiv – clasificator de tip MAP;
o condiție [suficientă] pentru echivalență cu clasificarea de tip ML)
CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW3, pr. 4.2

Algoritmul Bayes Naiv asociază unui exemplu x clasa c dacă aceasta maximizează probabilitatea $P(c|x)$.

Când este această condiție [din formularea algoritmului Bayes Naiv] echivalentă cu selecționarea acelei clase c care maximizează probabilitatea $P(x|c)$?

38. (Algoritmii Bayes Naiv și Bayes Optimal: Adevărat sau Fals?)
CMU, 2005 spring, C. Guestrin, T. Mitchell, midterm, pr. 2.b.5
CMU, 2011 spring, Tom Mitchell, midterm, pr. 1.1.ab

- Clasificatorul Bayes Optimal poate să obțină rata de eroare 0 [la antrenare] pentru orice set de date. Justificați.

- b. Dacă antrenăm un clasificator Bayes Naiv folosind un număr infinit de date de antrenament care satisfac toate presupozиtiile luate în calcul de acest tip de modelare (de exemplu, independența condițională), atunci acest clasificator va produce *eroare* zero pe setul de exemple de antrenament considerat.
- c. Dacă antrenăm un clasificator Bayes Naiv folosind un număr infinit de date de antrenament care satisfac toate presupозиtiile luate în calcul de acest tip de modelare (de exemplu, independența condițională), atunci acest clasificator va produce *eroare „adevărată“* zero pentru exemple de test generate conform aceleiași distribuții.

2.2.3 Clasificare bayesiană [cu atribute de intrare] de tip gaussian

39. (Distribuția gaussiană unidimensională: exemplificare pentru estimarea mediei în sens MLE și calcularea unei probabilități a posteriori)

U. Edinburgh, 2009 fall, C. Williams, V. Lavrenko, HW2, pr. 1

Considerăm un set de date [de antrenament] care constă din *exemplu*e pentru două *clase*. Exemplile pentru clasa 1 sunt 0.5, 0.1, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.35, 0.25, iar exemplile pentru clasa 2 sunt 0.9, 0.8, 0.75, 1.0.

- a. Vă cerem să antrenați (engl., fit) căte o distribuție gaussiană unidimensională pentru fiecare din cele două clase, folosind metoda estimării în sensul verosimilității maxime (engl., Maximum Likelihood estimation, MLE). Veți presupune că varianța distribuției corespunzătoare clasei 1 este 0.0149, iar varianța distribuției pentru clasa 2 este 0.0092.
- b. Estimați de asemenea în sens MLE probabilitățile *a priori* de selecție pentru [generarea de exemple din] cele două clase.
- c. Care este probabilitatea *a posteriori* ca punctul $x = 0.6$ să aparțină clasei 1?

40. (Algoritmul Bayes [Naiv] gaussian: exemplificare pe date din \mathbb{R} , granițe de decizie)

CMU, 2009 spring, Tom Mitchell, midterm, pr. 5

În acest exercițiu vom considera mai mulți clasificatori de tip Bayes Naiv gaussian (GNB) pentru un set de date având un singur atribut x și două clase, 0 și 1.¹⁵⁰ Ca de obicei pentru clasificatori bayesieni, vom clasifica o instanță x ca aparținând clasei 1 dacă

$$P(y = 1|x) \geq P(y = 0|x) \Leftrightarrow \ln \frac{P(y = 1|x)}{P(y = 0|x)} \geq 0.$$

Vom folosi notația $\mathcal{N}(\mu, \sigma^2)$ pentru a desemna o distribuție normală / gaussiană de medie μ și varianță σ^2 . Vi se va cere să scrieți expresia analitică a separatorului decizional (sau, a graniței de decizie) pentru fiecare model de tip GNB prezentat mai jos.

¹⁵⁰Fiind dat un singur atribut de intrare, putem renunța la termenul „Naiv“ din expresia care desemnează tipul clasificatorului.

a. Fie următorul clasificator Bayes [Naiv] gaussian:

$$\begin{aligned} x|y=0 &\sim \mathcal{N}(0, 1) \\ x|y=1 &\sim \mathcal{N}(2, 1) \\ P(y=1) &= 0.5 \end{aligned}$$

Este oare granița de decizie a acestui clasificator GNB liniară? Altfel spus, puteți scrie o expresie de forma $w_0 + w_1x \geq 0$ care reprezintă granița de decizie a acestui model GNB? În cazul afirmativ, calculați valorile w_0 și w_1 .¹⁵¹ În cazul negativ, justificați.

b. Acum vom considera un alt clasificator de tip GNB. Noii parametri pentru cele două distribuții gaussiene sunt:

$$\begin{aligned} x|y=0 &\sim \mathcal{N}(0, 1/4) \\ x|y=1 &\sim \mathcal{N}(0, 1) \\ P(y=1) &= 0.5 \end{aligned}$$

Este granița de decizie a acestui model GNB liniară? Dacă da, marcați care dintre opțiunile de mai jos constituie granița de decizie corectă. Dacă alegeti opțiunea (v), dați o scurtă explicație.

- (i) Alege clasa 1 dacă $x \geq 1/2$.
- (ii) Alege clasa 1 dacă $x \leq -1/2$.
- (iii) Alege clasa 1 dacă $x \leq 1$.
- (iv) Alege clasa 1 dacă $x \geq -1$.
- (v) Granița de decizie nu este liniară.

c. Acum vom considera o graniță de decizie *pătratică*. La o graniță de decizie pătratică adăugăm o nouă trăsătură, x^2 , instanței de antrenament $\langle x, y \rangle$. Astfel, instanța $\langle x, y \rangle$ va fi transformată în $\langle x^2, x, y \rangle$. O graniță de decizie *liniară* pentru acest set de date modificat — aşadar, determinată de $w_0, w_1, w_2 \in \mathbb{R}$ cu proprietatea $w_0 + w_1x + w_2x^2 \geq 0 \Leftrightarrow P(y=1|x) \geq P(y=0|x)$ pentru $\forall x \in \mathbb{R}$ — produce o graniță de decizie *pătratică* pentru setul de date original.

Este oare posibil să găsim o graniță de decizie pătratică care corespunde exact graniței de decizie a modelului GNB de la punctul b? Dacă da, marcați care dintre opțiunile de mai jos constituie granița de decizie corectă. Dacă alegeti opțiunea (iv), dați o scurtă explicație.

- (i) Alege clasa 1 dacă $x \leq -0.68$ or $x \geq 0.68$.
- (ii) Alege clasa 1 dacă $-0.95 \leq x \leq 0.95$.
- (iii) Alege clasa 1 dacă $-0.68 \leq x \leq 0.68$.
- (iv) Granița de decizie nu este pătratică.

¹⁵¹Adică, găsiți w_0 și $w_1 \in \mathbb{R}$ astfel încât $P(y=1|x) \geq P(y=0|x) \Leftrightarrow w_0 + w_1x \geq 0$ pentru $\forall x \in \mathbb{R}$.

41. (Algoritmul Bayes Naiv gaussian: chestiuni legate de estimarea parametrilor distribuțiilor condiționale; aplicare; algoritmul Bayes Optimal gaussian: numărul parametrilor necesari de estimat)
 ■ prelucrare de Liviu Ciortuz, după CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, HW2, pr. 5.ab CMU, 2007 spring, midterm, pr. 4.A

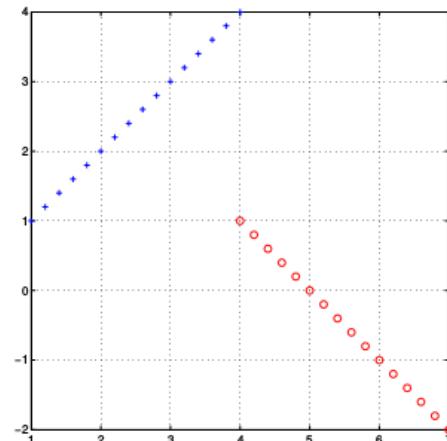
Fie un set de date de antrenament pentru care $Y \in \{0, 1\}$ reprezintă clasele / etichetele, iar $X \in \mathbb{R}^d$ reprezintă vectorii de trăsături (d -dimensionali).

La punctele a-c ale acestui exercițiu vom lucra cu modelul Bayes Naiv gaussian, în care distribuția probabilistă condițională pentru fiecare trăsătură este o distribuție gaussiană unidimensională / univariată.

- Considerând n instanțe de antrenament generate în mod independent, $\{(X^{(1)}, Y^{(1)}), \dots, (X^{(n)}, Y^{(n)})\}$, calculați estimările de verosimilitate maximă (MLE) pentru parametrii distribuțiilor probabiliste următoare de variabilele condiționale $X^{(j)}|Y$, pentru $j = 1, \dots, d$.
- Câtări parametri trebuie estimati pentru modelul Bayes Naiv gaussian? Veți presupune că parametrul pentru distribuția a priori a lui Y este deja cunoscut.

- c. Considerăm setul de date din figura alăturată, în care sunt două clase, desemnate cu + și respectiv o. Număr de instanțe din cele două clase este același. Fiecare instanță are două trăsături, care corespund coordonatelor X_1 și X_2 , ambele având valori numere reale.

Trasați granița de decizie pe care o va învăța clasificatorul Bayes Naiv gaussian pe acest set de date.



- d. În modelul Bayes Optimal [complet] Gaussian, presupunem că distribuția probabilistă condițională $Pr(X|Y)$ este o gaussiană multidimensională / multivariată, $X|Y \sim \mathcal{N}(\mu_Y, \Sigma_Y)$, unde μ_Y este vectorul medie, iar $\Sigma_Y \in \mathbb{R}^{d \times d}$ este matricea de covarianță.

Câtări parametri trebuie estimati pentru modelul Bayes complet gaussian? Veți presupune, ca și la punctul b, că parametrul pentru distribuția a priori a lui Y este deja cunoscut.

Observație: Prin numărul de parametri se va înțelege aici *nu* numărul de vectori de medii — din \mathbb{R}^d — și numărul de matrice de covarianță — din $\mathbb{R}^{d \times d}$ — de estimat, *ci* numărul total de componente (adică, numere reale) ale tuturor acestor vectori de medii și matrice de covarianță, care sunt de estimat.

42. (Algoritmul Bayes Naiv gaussian:
legătura dintre estimarea parametrilor și
maximizarea log-verosimilității datelor)
CMU, 2017 fall, Nina Balcan, Practice questions, pr. 1.2

După cum știți, în clasificarea binară obiectivul este să putem prezice un „target“ binar $Y \in \{0, 1\}$ pentru o instanță oarecare de test $X = (X_1, \dots, X_d)$. Aici vom presupune că $X \in \mathbb{R}^d$. Pentru a prezice Y , este suficient să estimăm funcția $P(Y|X)$. Algoritmul Bayes Naiv calculează $P(Y|X)$ astfel:

$$P(Y|X) = \frac{P(Y, X)}{P(X)} = \frac{P(Y) \prod_{j=1}^d P(X_j|Y)}{P(X)}.$$

Aici vom presupune că toate distribuțiile condiționate în raport cu clasa sunt de tip gaussian.¹⁵² Aceasta înseamnă:

$$P(X_j|Y = k) = \mathcal{N}(X_j|\mu_{j,k}, \sigma_{j,k}^2),$$

unde $\mu_{j,k}$ și $\sigma_{j,k}^2$ sunt media și respectiv varianța. Să presupunem că avem n instanțe de antrenament $\{X^{(i)}, Y^{(i)}\}_{i=1}^n$. Pentru a estima parametrii modelului pornind de la aceste date, vom învăța (engl., fit) căte o distribuție gaussiană univariată (sau, unidimensională) pentru fiecare trăsătură / atribut de intrare (X_j) și pentru fiecare clasă ($Y = k$).

Arătați că acest estimator de tip Bayes Naiv gaussian de fapt rezolvă următoarea *problemă* de estimare în sensul verosimilității maxime (engl., maximum likelihood estimation, MLE):

$$\max \frac{1}{n} \sum_{i=1}^n \ln P(X^{(i)}, Y^{(i)}).$$

43. (Algoritmul Bayes Naiv gaussian, cazul $\sigma_{i0} \neq \sigma_{i1}$:
raportul cu regresia logistică)
CMU, 2011 spring, Tom Mitchell, HW2, pr. 2.1

În această problemă vom modifica clasificatorul Bayes Naiv gaussian care a fost prezentat la problema 17 și-l vom face ceva mai general, eliminând presupunerea că σ_i , deviația standard pentru distribuția urmată de $P(X_i|Y = k)$ nu depinde de eticheta k . Prin urmare, pentru fiecare X_i și fiecare k , unde $i \in \{1, 2, \dots, n\}$ și $k \in \{0, 1\}$, distribuția urmată de $P(X_i|Y = k)$ este de tipul $\mathcal{N}(\mu_{ik}, \sigma_{ik}^2)$.

Este oare și noua formă a probabilității condiționate $P(Y|X)$, care este implicată de acest clasificator Bayes Naiv gaussian [mai general], la fel cu forma regresiei logistice? Pentru a justifica răspunsul dumneavoastră, calculați noua formă a probabilității condiționate $P(Y|X)$.

¹⁵²LC: De fapt, concluzia pe care o vom formula mai jos se menține și atunci când în locul distribuțiilor gaussiene lucrăm cu distribuții Bernoulli (sau, mai general, cu distribuții categoriale).

44. (Algoritmul Bayes Naiv: deducerea regulii de decizie pentru cazul când toate atributele sunt gaussiene, în afară de unul singur, care este de tip boolean)
CMU, 2010 fall, Aarti Singh, HW1, pr. 4.1

Considerăm funcția de învățare $X \rightarrow Y$, unde $Y \in \{T, F\}$ reprezintă eticheta clasei, iar X este n -uplul (X_1, X_2, \dots, X_n) , cu X_1 variabilă booleană și X_2, \dots, X_n variabile continue. Presupunem că în cazul fiecărei variabile continue X_i , distribuția $P(X_i|Y = y)$ este de tip gaussian de medie $\mu_{i,y}$ și varianță $\sigma_{i,y}^2$. Vrem ca în acest cadru să antrenăm un clasificator de tip Bayes Naiv, deci care să lucreze cu presupozitia de independentă condițională a variabilelor X_i (cu $i = 1, \dots, n$) în raport cu variabila Y .

- Enumerați toți parametrii care trebuie estimati pentru a putea clasifica o instanță nouă. Care este numărul total al acestor parametri?
- Elaborați formula de calcul pentru $P(Y|X)$ în funcție de acești parametri și de atribuțele / trăsăturile X_i . Tratați cazul particular când $\sigma_{i,T}^2 = \sigma_{i,F}^2$.

45. (Algoritmul Bayes Optimal gaussian: câteva chestiuni de bază)
CMU, 2017 fall, Nina Balcan, midterm, pr. 4.1-4,6-7

Fie două variabile aleatoare, $X \in \mathbb{R}^d$ și $Y \in \{0, 1\}$. Vom considera că atât $p_0(X = x) \stackrel{\text{not.}}{=} p(x|Y = 0)$ cât și $p_1(X = x) \stackrel{\text{not.}}{=} p(x|Y = 1)$ sunt distribuții gaussiene multidimensionale:

$$p_y(x) = \frac{1}{(2\pi)^d |\Sigma_y|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y) \right), \text{ pentru } y \in \{0, 1\}.$$

Așadar, putem scrie $X|(Y = y) \sim \mathcal{N}(\mu_y, \Sigma_y)$. Mai departe, vom nota probabilitățile de selecție ale celor două clase cu $\pi_y \stackrel{\text{not.}}{=} P(Y = y)$ pentru $y \in \{0, 1\}$. Vom lucra cu un set de n „realizări“ i.i.d. ale cuplului de variabile (X, Y) , și anume $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$.

- Clasificatorul Bayes Optimal gaussian va prezice eticheta / clasa unei noi instanțe x conform următoarei regule de decizie: $\hat{y} = I(\pi_1 p_1(x) > \pi_0 p_0(x))$, unde $I(\cdot)$ este funcție-indicator.¹⁵³ Demonstrați că această regulă se poate scrie în mod echivalent astfel:

$$\begin{aligned} \hat{y} = & \\ I((x - \mu_1)^\top \Sigma_1^{-1} (x - \mu_1) - 2 \ln \pi_1 + \ln(|\Sigma_1|) < (x - \mu_0)^\top \Sigma_0^{-1} (x - \mu_0) - 2 \ln \pi_0 + \ln(|\Sigma_0|)). \end{aligned} \tag{78}$$

- (Adevărat sau fals?)

Este oare granița de decizie determinată de algoritmul Bayes Optimal gaussian liniară? Dacă da, explicați de ce este așa. Dacă nu, precizați în ce condiții / cazuri această graniță de decizie este liniară.

- Dat fiind setul de date de antrenament S , precizați care este estimarea de verosimilitate maximă (MLE) pentru vectorii de medii μ_y , precum și pentru probabilitățile de selecție π_y , pentru $y \in \{0, 1\}$.

Sugestie: Pentru estimarea mediilor μ_y , vă sugerăm să faceți referire la problema 48.a de la capitolul *Fundamente*, iar pentru estimarea probabilităților de selecție π_y , vedeti problema 113.a (sau problema 40.c) de la capitolul *Fundamente*.

¹⁵³Aceasta înseamnă că $\hat{y} = 1$ dacă x satisfacă condiția care a fost scrisă ca argument al funcției $I(\cdot)$, și $\hat{y} = 0$ în cazul contrar.

d. (Adevărat sau fals?)

Dacă fiind setul de date de antrenament S , un *estimator nedeplasat* (engl, unbiased) al matricelor de covarianță este

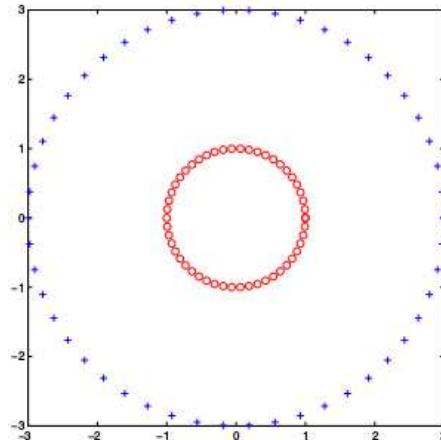
$$\hat{\Sigma}_y = \frac{1}{n_y - 1} \sum_{i: y_i=y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^\top, \text{ pentru } y \in \{0, 1\},$$

unde $n_1 = \sum_{i=1}^n y_i$ și $n_0 = \sum_{i=1}^n (1 - y_i)$.

Sugestie: Vedeti problemele 113.a și 46 de la capitolul *Fundamente*.

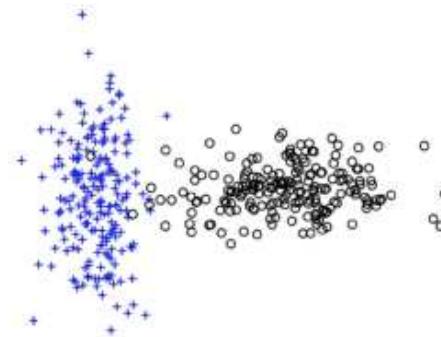
e. Presupunem că $\pi_0 = \pi_1 = 0.5$. Desenați granița de decizie determinată de algoritm Bayes Optimal gaussian pentru setul de instanțe [ale celor două clase] din figura alăturată. Justificați cât se poate de riguros.

Sugestie: Faceți referire la relația (78).



f. Similar cu punctul precedent, veți presupune că $\pi_0 = \pi_1 = 0.5$ și veți desena granița de decizie determinată de algoritm Bayes Optimal gaussian pentru setul de instanțe din figura alăturată.

Care dintre cele două trăsături (corespunzătoare celor două coordonate din planul euclidian) are o putere discriminativă mai mare?



46.

(Algoritmul Bayes Optimal gaussian:
aplicare pe setul de date XOR)

CMU, (?) 15-781, midterm example questions, pr. 3.2.b

În acest exercițiu ne vom referi la învățarea conceptualui XOR. Așadar, vom considera că avem două atrbute de intrare, x și y , care iau valorile -1 și $+1$, iar outputul va fi pozitiv dacă și numai dacă $x \neq y$.

Ce se întâmplă dacă încercăm să aplicăm clasificatorul Bayes Optimal gaussian pe setul de date corespunzător conceptualui XOR? (Vom presupune că acest clasificator este capabil să estimateze orice matrice de covarianță.)

47. (Clasificatorul Bayes Optimal gaussian: aplicare în \mathbb{R}^2 ; granițe de decizie, în cazul $\Sigma_0 \neq \Sigma_1$)
CMU, 2010 fall, Aarti Singh, midterm, pr. 2.4

Fie următoarea problemă de clasificare în \mathbb{R}^2 :

Mai întâi presupunem că $P(y = 0) = P(y = 1) = 1/2$. De asemenea, presupunem că funcțiile densitate de probabilitate (p.d.f.) condiționale în raport cu clasa / eticheta sunt de tip gaussian, cu media $\mu_0 \in \mathbb{R}^2$ și matricea de covarianta Σ_0 pentru clasa 0, respectiv media $\mu_1 \in \mathbb{R}^2$ și matricea de covarianta Σ_1 pentru clasa 1. Mai mult, presupunem că $\mu_0 = \mu_1 \stackrel{\text{not.}}{=} (\mu^1, \mu^2) \in \mathbb{R}^2$.

Pentru cazul

$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

trasați într-un sistem de coordonate cartezian *curbe de izocontur* corespunzătoare celor două p.d.f.-uri condiționale, pe care le veți nota cu $p(x|y = 0)$ și respectiv $p(x|y = 1)$.

Apoi trasați *granițele de decizie* (engl., decision boundaries) determinate de clasificatorul Bayes Optimal gaussian. Determinați în mod riguros — adică, plecând de la *regula de decizie* a acestui clasificator — ecuația satisfăcută de aceste granițe de decizie.

În final, indicați regiunile în care clasificatorul va prezice clasa 0, respectiv clasa 1.

Indicație: Vă reamintim că pentru o variabilă aleatoare $X : \Omega \rightarrow \mathbb{R}^n$, care este reprezentată pe componente ca vector-colonă $X = (X_1, \dots, X_n)^\top$ și care urmează o distribuție gaussiană având media $\mu \in \mathbb{R}^n$ și matricea de covarianta Σ , funcția densitate de probabilitate are forma analitică următoare:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right),$$

unde \top reprezintă operația de transpunere.¹⁵⁴

48. (Clasificarea bayesiană gaussiană vs. regresia logistică și regresia liniară: Da sau nu?)
CMU, (?) 15-781, midterm example questions, pr. 3.3
CMU, 2009 spring, Tom Mitchell, midterm, pr. 1.b

a. Ne întrebăm dacă există vreun clasificator de tip Bayes gaussian pentru [date cu] un singur atribut de intrare x astfel încât, atunci când va fi folosit, să facă următoarele predicții:

- clasa 1 dacă $x < -1$;
- clasa 2 dacă $-1 < x < 1$;
- clasa 1 dacă $x > 1$.

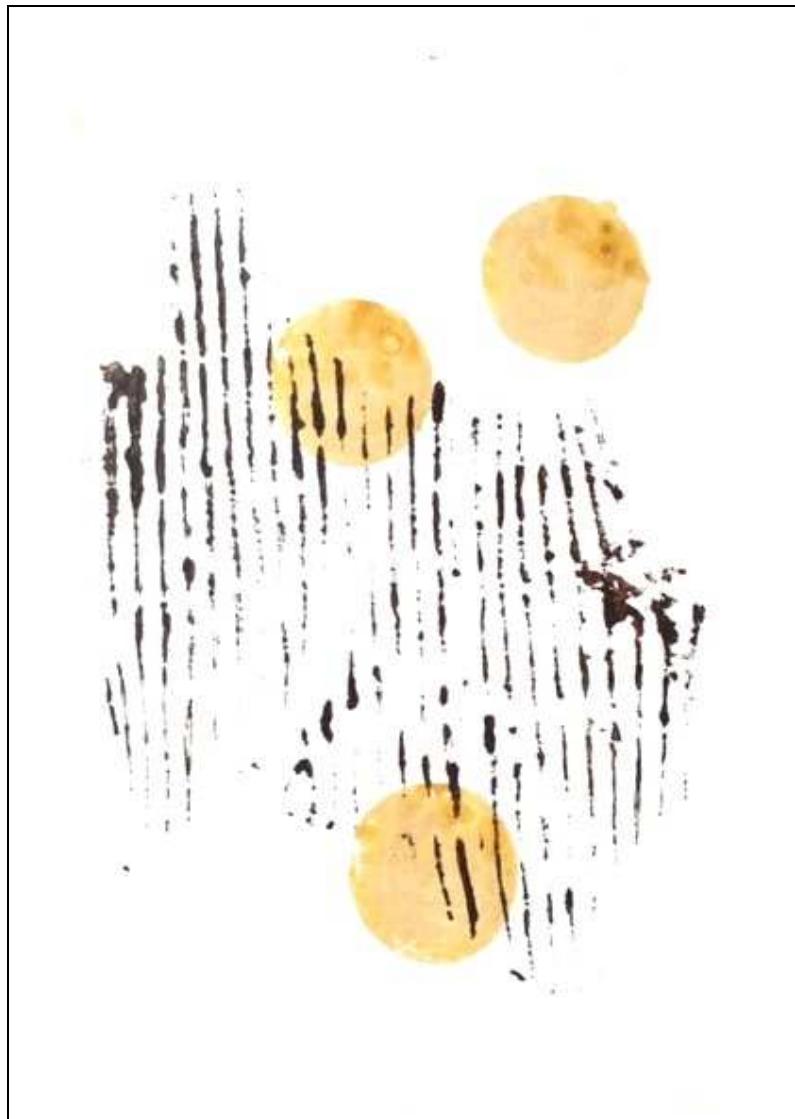
¹⁵⁴Observați că notația $(x - \mu)^\top \Sigma^{-1}(x - \mu)$ este matriceală. În urma efectuării operațiilor de înmulțire rezultă o matrice de tip 1×1 . În expresia $\exp(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu))$, matricea menționată mai sus este de fapt substituită cu un număr real, și anume cu unicul ei element.

În cazul afirmativ, precizați cum anume se poate construi un astfel de clasificator.

b. Presupunem că antrenăm mai mulți clasificatori pentru a învăța funcția $f : X \rightarrow Y$, unde $X = \langle X_1, X_2, X_3 \rangle$ este vectorul de trăsături. Pentru fiecare dintre următorii clasificatori,

- i. algoritmul Bayes Naiv [gaussian]
- ii. regresia logistică
- iii. regresia liniară

indicați dacă respectivul clasificator oferă suficiente informații ca să putem calcula probabilitățile $P(X_1, X_2, X_3, Y)$. Atunci când răspundeți cu *da*, precizați detaliile. Invers, atunci când răspundeți cu *nu*, spuneți [ș] ce anume lipsește pentru a calcula $P(X_1, X_2, X_3, Y)$.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

3 Învățare bazată pe memorare

Sumar

Noțiuni preliminare

- măsuri de distanță, măsuri de similaritate: ex. 2;
- normă într-un spațiu vectorial; [măsura de] distanță indusă de către o normă: ex. 7;
- k -NN vecinătate a unui punct din \mathbb{R}^d .

Algoritmul k -NN

- pseudo-cod: cartea ML, pag. 232;
- bias-ul inductiv: „Cine se aseamănă se adună“ (sau: „Spune-mi cu cine te împrietenești, ca să-ți spun cine ești“): ex. 15.a;
- exemple (simple) de aplicare: ex. 1, ex. 2;
- complexitate de spațiu: $\mathcal{O}(dn)$
complexitate de timp:
 - la antrenare: $\mathcal{O}(dn)$
 - la testare: $\mathcal{O}(dn \log n)$
[LC: $\mathcal{O}(dn k \log k)$ pt. $k > 1$ (worst case) și $\mathcal{O}(dn)$ pt. $k = 1$],
unde d este numărul de atribute, iar n este numărul de exemple;
- arbori kd (engl., kd -trees): *Statistical Pattern Recognition*, Andrew R. Webb, 3rd ed., 2011, Wiley, pag. 163-173;
- k -NN ca algoritm ML “lazy” (vs. “eager”):
suprafețe de decizie și granițe de decizie:
diagrame Voronoi pentru 1-NN: ex. 4, ex. 11.a, ex. 18, ex. 19, ex. 20.a;
- analiza erorilor:
 - 1-NN pe date consistente: eroarea la antrenare este 0: ex. 2, ex. 12.a;
 - variația numărului de erori (la antrenare și respectiv testare) în funcție de valorile lui k : ex. 22, ex. 23.ab;
 k -NN ca metodă neparametrică; alegerea lui k : CV: ex. 23.c;
 - CVLOO: ex. 3, ex. 12.b, ex. 15.bc, ex. 16, ex. 24.a, ex. 20.b;
 - sensibilitatea / robustețea la „zgomote“: ex. 5, ex. 15;
 - eroarea asimptotică: ex. 10, ex. 25.
- efectul trăsăturilor redundante sau irelevante;
- alegerea valorii convenabile pentru k : ex. 21.

Proprietăți ale algoritmului k -NN

- (P0) output-ul algoritmului k -NN pentru o instanță oarecare de test x_q depinde de valoarea lui k : ex. 1;
- (P1) pe seturi de date de antrenament *consistent*, eroarea la antrenare produsă de algoritmul 1-NN este 0: ex. 2, ex. 12.a;

- (P2) output-ul algoritmului k -NN, precum și suprafețele de decizie și separatorii decizionali depind de *măsura de distanță* folosită: ex. 7;
- (P3) „blestemul marilor dimensiuni” (engl., the curse of dimensionality): în anumite condiții, numărul de instanțe de antrenament necesare pentru a avea un *cel mai apropiat vecin* situat la distanță *rezonabilă* față de instanța de test x_q crește exponential în funcție de numărul de atrbute folosite: ex. 9;
- (P4) în anumite condiții, rata medie a *erorii asymptotice* a algoritmului 1-NN este mărginită superior de dublul ratei medii a erorii algoritmului Bayes Optimal: ex. 10, ex. 25.

Comparări cu alți algoritmi de clasificare automată

- ID3: ex.11.b, ex. 13.ab;
- SVM: ex.12, ex. 13.c, ex. 24.b;
- regresia logistică: ex. 24.b;
- 1-NN cu mapare cu RBF: ex. 14.

Variante ale algoritmului k -NN

- k -NN folosind alte măsuri de distanță (decât distanța euclidiană): ex. 7;
- k -NN cu *ponderarea distanțelor* (engl., distance-weighted k -NN):
cartea ML, pag. 236-238 (formulele 8.2, 8.3, 8.4);¹⁵⁵
- algoritmul lui Shepard: ex.8.

Alte metode de tip IBL

- rețele RBF: cartea ML, pag. 238-240;
- raționare bazată pe cazuri (engl., case-based reasoning): cartea ML, pag. 240-244.

3.1 Probleme rezolvate

1.

(Algoritmul k -NN: aplicare în \mathbb{R}^2
pentru diferite valori ale lui k)

■ CMU, 2006 fall, final exam, pr. 2

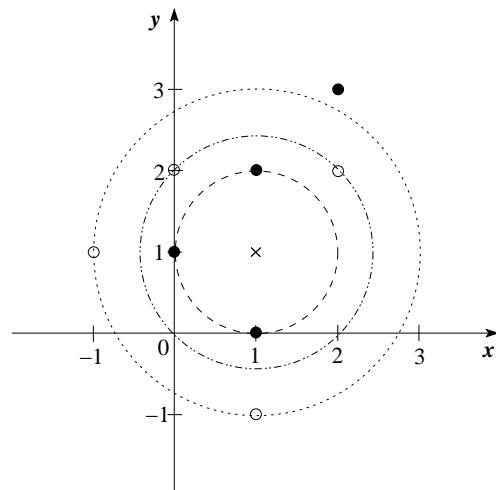
Fie setul de instanțe de antrenament din tabelul alăturat:

- Vizualizați datele într-un sistem de axe din \mathbb{R}^2 .
- Presupunând că se folosește distanța euclidiană, care va fi predicția făcută pentru punctul $(1,1)$ de către
 - clasificatorul 3-NN?
 - clasificatorul 5-NN?
 - clasificatorul 7-NN?

x	y	
-1	1	-
0	1	+
0	2	-
1	-1	-
1	0	+
1	2	+
2	2	-
2	3	+

¹⁵⁵Sectiunea 8.3 din cartea ML (pag. 236-238) se referă la regresia [liniară] local-ponderată ca o formă mai generală de aproximare a [valorilor] funcțiilor, în raport cu cele calculate de către algoritmul k -NN atunci când se folosește ponderarea distanțelor.

Răspuns:



În figura alăturată am reprezentat:

- datele de antrenament, folosind cerculete albe pentru cele clasificate cu – și cerculete negre pentru cele clasificate cu +;
- punctul care trebuie clasificat, marcat cu ×;
- vecinătățile luate în considerare de către cei trei clasificatori precizați în enunț; aceste vecinătăți sunt reprezentate prin cele trei cercuri concentrice având centrul în punctul (1,1).

Analizând pe rând etichetele instanțelor din aceste trei vecinătăți, ajungem la concluzia că rezultatele obținute de cei trei clasificatori vor fi următoarele:

- 3-NN: +
- 5-NN: +
- 7-NN: –

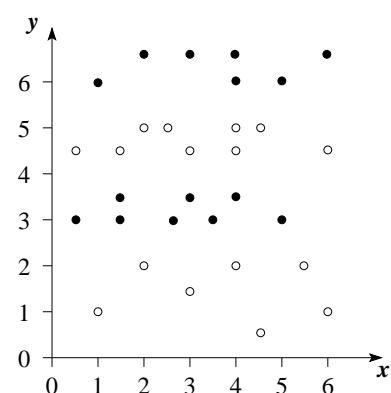
Observație: Acest exercițiu pune în evidență faptul că la clasificarea cu algoritmul k -NN rezultatul poate doli în funcție de diversele valori ale lui k , adică în funcție de cum se lărgește (sau se restrâne) vecinătatea punctului de test considerat.

2.

(Algoritmul 1-NN: eroarea la antrenare)
CMU, 2002 fall, Andrew Moore, final exam, pr. 6.e

Figura alăturată prezintă un set de date având două atribute de intrare x și y , și un atribut de ieșire, ale cărui valori sunt reprezentate prin culoarea punctului (alb sau negru).

Putem alege o metrică (adică, măsură sau funcție de distanță) astfel încât, folosind algoritmul de învățare 1-NN (engl., [one] nearest neighbour), să obținem eroare 0 la antrenare pe setul acesta de date?



Răspuns:

Fie $d : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$ o metrică oarecare. Conform definiției matematice, d îndeplinește următoarele *condiții*:

$$\begin{array}{ll} d(x, y) \geq 0, \forall x, y \in \mathbb{R}^2 & (\text{neneagativitatea}) \\ d(x, y) = 0 \Leftrightarrow x = y & (\text{identitatea indiscernabilor}) \\ d(x, y) = d(y, x) & (\text{simetria}) \\ d(x, y) \leq d(x, z) + d(z, y), \forall x, y, z \in \mathbb{R}^2 & (\text{inegalitatea triunghiului}) \end{array}$$

În particular, d poate fi metrika euclidiană, definită astfel:

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2},$$

pentru orice $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$.

În general, când algoritmul 1-NN primește o instanță de test (x, y) , el caută în setul de antrenament punctul (x', y') cu proprietatea că distanța $d((x, y), (x', y'))$ este minimă.

În cazul particular în care punctul (x, y) însuși aparține datelor de antrenament, urmează că $(x', y') = (x, y)$, datorită primelor două proprietăți ale lui d enunțate mai sus.

Cum setul de date din enunț este *consistent* — adică nu există nicio instanță care să apară de două sau mai multe ori, însă cu etichete diferite —, vom avea de luat în considerare o singură valoare pentru calculul atributului de ieșire pentru punctul de test (x, y) . Evident, algoritmul 1-NN o va folosi pe aceasta ca rezultat al clasificării. Concluzia acestui raționament este că eroarea la antrenare produsă de algoritmul 1-NN pe acest set de date este 0.

Facem *observația* că nu doar pe acest set de date ci pe orice set de date de antrenament fără zgromote / inconsistențe în ce privește etichetarea, algoritmul 1-NN va avea eroarea la antrenare 0, indiferent de metrika folosită (bineînțeles, dacă există o metrică în spațiul instanțelor respective).

3.

(Algoritmul k -NN: calculul erorii la CVLOO
pentru diferite valori ale lui k)

CMU, 2003 fall, T. Mitchell, A. Moore, final exam, pr. 5.ab

Fie setul de date de antrenament din tabelul alăturat.

Vom folosi algoritmul k -NN cu distanță euclidiană (neponderată) pentru a prezice valorile atributului de ieșire Y (de tip boolean) plecând de la atributul de intrare X , care ia valori reale.

Care este eroarea produsă de algoritmul k -NN la cross-validation cu metoda “Leave-One-Out” în cazurile următoare:

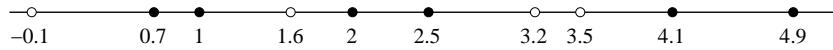
- a. $k = 1$.
- b. $k = 3$.

Expremați răspunsul sub forma numărului de instanțe clasificate eronat.

X	Y
-0.1	-
0.7	+
1.0	+
1.6	-
2.0	+
2.5	+
3.2	-
3.5	-
4.1	+
4.9	+

Răspuns:

Figura următoare vizualizează pe o axă datele de antrenament, precum și clasificările acestora (○ pentru instanțe negative și ● pentru instanțe pozitive):



Comportamentul celor doi clasificatori la cross-validation prin metoda “Leave-One-Out” este cel explicat mai jos:

- Clasificatorul 1-NN:

Data	Eticheta	Vecinătate	Clasificare la CVLOO	Eroare?
-0.1	-	{0.7}	+	da
0.7	+	{1.0}	+	nu
1.0	+	{0.7}	+	nu
1.6	-	{2.0}	+	da
2.0	+	{1.6}	-	da
2.5	+	{2.0}	+	nu
3.2	-	{3.5}	-	nu
3.5	-	{3.2}	-	nu
4.1	+	{3.5}	-	da
4.9	+	{4.1}	+	nu

- Clasificatorul 3-NN:

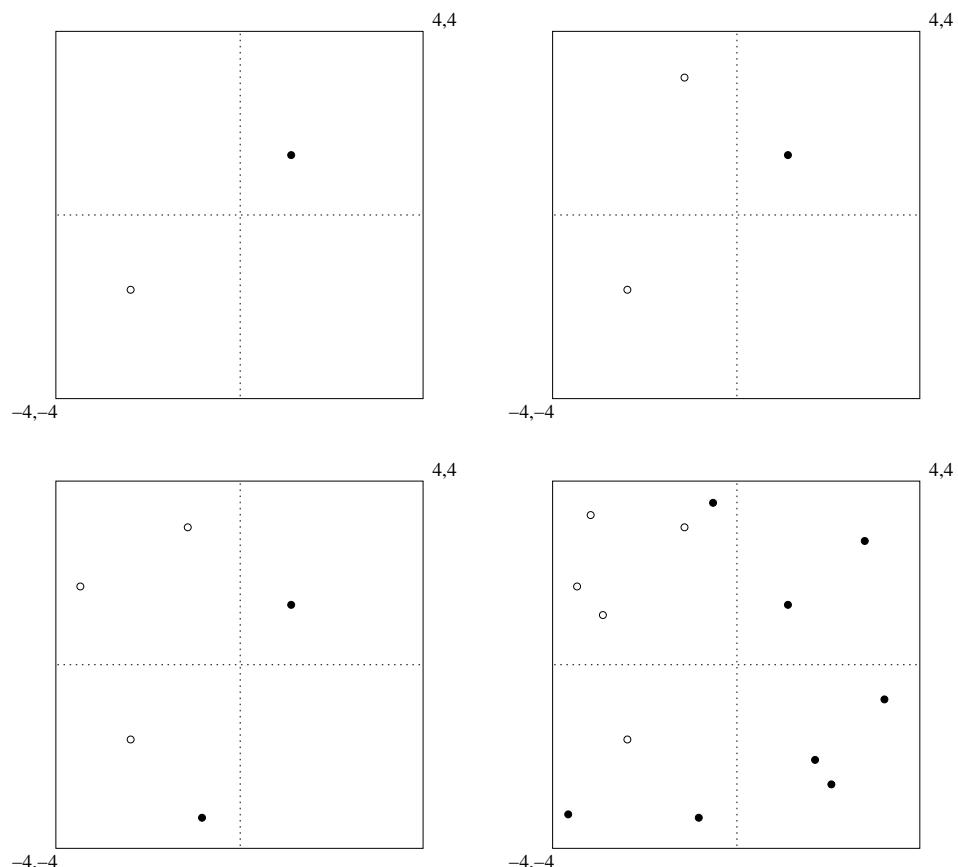
Data	Eticheta	Vecinătate	Clasificare la CVLOO	Eroare?
-0.1	-	{0.7; 1.0; 1.6}	+	da
0.7	+	{-0.1; 1.0; 1.6}	-	da
1.0	+	{0.7; 1.6; 2.0}	+	nu
1.6	-	{1.0; 2.0; 0.7/2.5}	+	da
2.0	+	{1.0; 1.6; 2.5}	+	nu
2.5	+	{1.6; 2.0; 3.2}	-	da
3.2	-	{2.5; 3.5; 4.1}	+	da
3.5	-	{2.5; 3.2; 4.1}	+	da
4.1	+	{3.2; 3.5; 4.9}	-	da
4.9	+	{3.2; 3.5; 4.1}	-	da

În concluzie, la cross-validation cu metoda “Leave-One-Out” avem 4 erori la clasificarea cu algoritmul 1-NN și 8 erori la clasificarea cu algoritmul 3-NN. Putem concluza că rezultatul algoritmului 3-NN este foarte afectat de către puternica „mixare“ (adică, de frecvențele schimbări de clasă, de la o instanță oarecare la vecinii ei) din setul de antrenament.

- (Algoritmul 1-NN: granițe / suprafețe de decizie; diagrame Voronoi ca modalitate de învățare rapidă / “eager”)

■ CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW1, pr. 3.1-2

În fiecare din figurile următoare se dau câteva puncte în spațiul euclidian bidimensional. Fiecare dintre aceste puncte este etichetat fie pozitiv (cerc plin) fie negativ (cerc simplu).



a. Presupunând că folosim ca metrică distanța euclidiană, desenați suprafețele de decizie corespunzătoare clasificatorului 1-NN, în fiecare din aceste patru cazuri.

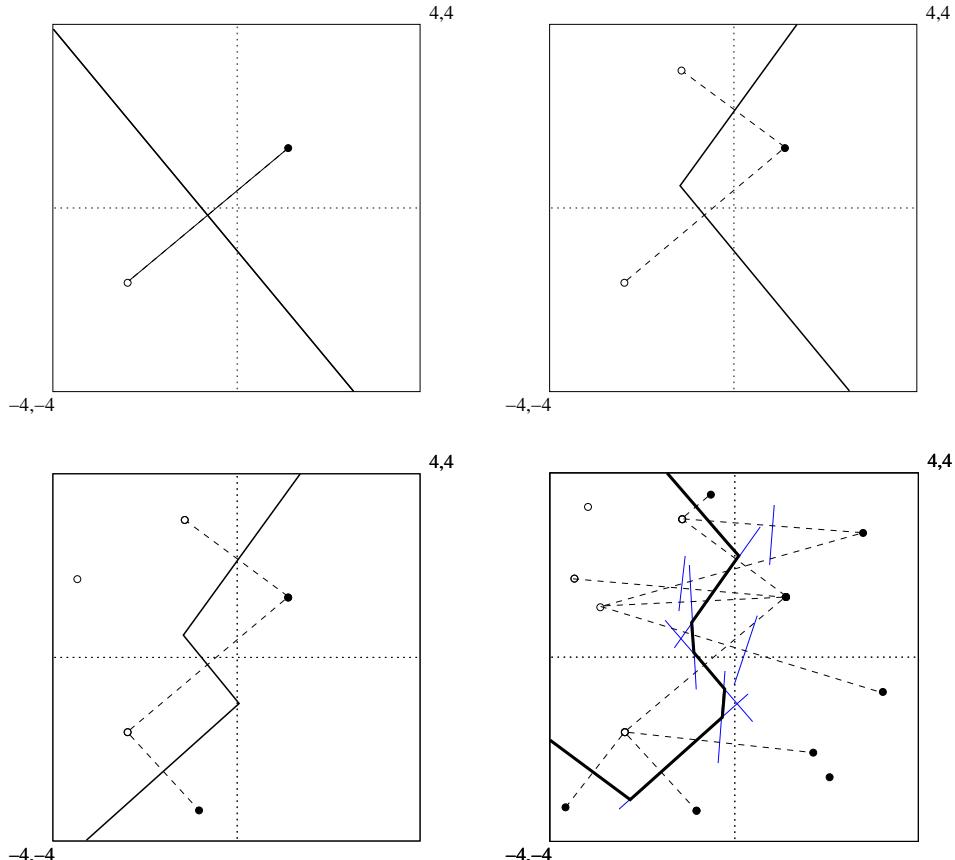
b. La curs am afirmat despre k -NN că este un clasificator lent (engl., *lazy*), care doar memorează toate instanțele de antrenament până ajunge la faza de test.

Totuși, la punctul precedent am văzut că putem să trasăm suprafețele de decizie pentru clasificatorul 1-NN, îmânte de a intra în faza de test / generalizare. Atunci, în această fază, în loc să calculăm diverse distanțe și apoi să determinăm care sunt cei mai apropiati vecini față de punctul de test dat (notat x_q), pur și simplu i se va asigna lui x_q clasa / eticheta corespunzătoare suprafetei de decizie în care se placează.

Dacă am decide să memorăm toate suprafețele de decizie (ca linii poligonale) în loc să memorăm toate datele de antrenament, am obține *întotdeauna* o îmbunătățire în ceea ce privește consumul de memorie necesar pentru acest clasificator?

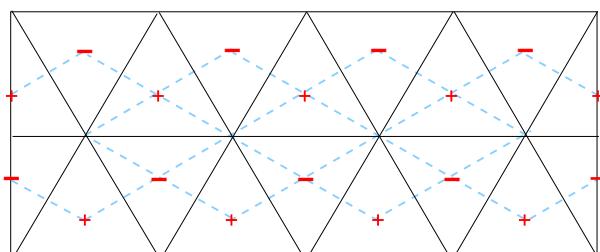
Răspuns:

a. Suprafețele de decizie corespunzătoare clasificatorului 1-NN pentru cele patru cazuri sunt:



b. Răspunsul este negativ, adică: nu întotdeauna memorarea separatorului decizional este mai convenabilă decât memorarea datelor de antrenament. Vom justifica dând un exemplu, care reprezintă o situație-limită, și anume, cazul care apare atunci când se creează tot atâtea suprafețe de decizie căte instanțe sunt în setul de antrenament.

În figura alăturată, pentru cele n instanțe de antrenament avem nevoie să memorăm $\frac{n}{2}$ puncte care determină triunghiurile — suprafețele de decizie pentru clasa +. (Clasa – va fi determinată prin excluziune / complementaritate.)

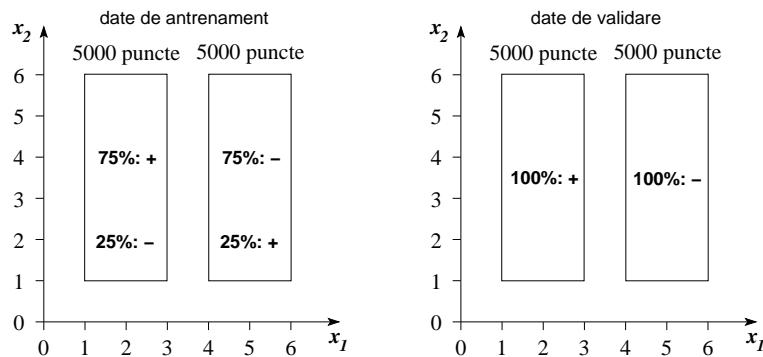


Se poate demonstra ușor următoarea proprietate: Chiar dacă am memora o singură dată cele $\frac{n}{2}$ puncte — fiecare punct având căte 2 coordonate — care determină contururile suprafețelor de decizie și apoi am folosi indecsi pentru a indica ce puncte determină fiecare suprafață de decizie, consumul de memorie ar fi mai mare decât dacă am memora doar

instanțele de antrenament.

5. (Algoritmul k -NN:
rata medie a erorii la antrenare, la CVLOO și la testare
în prezența unor „zgomote“ în datele de antrenament)
CMU, 2002 fall, Andrew Moore, final exam, pr. 7

Se constituie un set de date de antrenament și un set de date de validare, alegând în mod uniform aleatoriu puncte situate în anumite regiuni dreptunghiulare, conform imaginii următoare:



Se observă că datele de antrenament sunt „bruiate“ (engl., noisy): în fiecare regiune, 25% din date provin din clasa adversă. Datele de validare nu sunt bruiate.

Pentru fiecare dintre întrebările următoare, încercuiți fracția care este *cea mai apropiată de media / rata erorii* în cazul respectiv. Justificați în mod riguros alegerea făcută.

- Care este rata medie a erorii la antrenare pentru clasificatorul 1-NN ?
0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1
- Care este rata medie a erorii la cross-validation cu metoda “Leave-One-Out” pentru clasificatorul 1-NN pe setul de antrenare?
0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1
- Care este rata medie a erorii la testare pentru clasificatorul 1-NN pe setul de validare?
(Antrenarea se face pe setul de antrenare.)
0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1
- Care este rata medie a erorii la antrenare pentru clasificatorul 21-NN ?
0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1
- Care este rata medie a erorii la cross-validation cu metoda “Leave-One-Out” pentru clasificatorul 21-NN pe setul de antrenare?
0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1
- Care este rata medie a erorii la testare pentru clasificatorul 21-NN pe setul de validare?
(Antrenarea se face pe setul de antrenare.)

0 1/8 1/4 3/8 1/3 1/2 5/8 2/3 3/4 7/8 1

Răspuns:

a. În acest caz, rata medie a erorii este 0, întrucât orice punct din datele de antrenament este cel mai apropiat vecin în raport cu el însuși, iar probabilitatea ca două puncte care au fost generate în mod aleatoriu (în oricare din cele două dreptunghiuri) să aibă exact aceleași coordonate și aceeași etichetă este foarte mică (este practic 0).

b. 3/8.

Să analizăm cazul primului dreptunghi. Constituie eroare cazul când un exemplu pozitiv (care apare cu probabilitatea „așteptată“ de 3/4) este clasificat negativ (iar aceasta se întâmplă cu probabilitatea 1/4) sau, invers, când un exemplu negativ (probabilitate: 1/4) este clasificat pozitiv (probabilitate: 3/4). Așadar, probabilitatea de a clasifica gresit un exemplu din primul dreptunghi o aflăm astfel: $3/4 \cdot 1/4 + 1/4 \cdot 3/4 = 3/8$. Pentru dreptunghiul din dreapta raționamentul este similar, iar selecția exemplelor din cele două dreptunghiuri se face cu aceeași probabilitate (1/2), deci rezultatul final este 3/8.

c. 1/4.

Probabilitatea de eroare este 1/4 pentru fiecare dreptunghi, fiindcă (în general) pentru un punct (x_1, x_2) selectat în mod aleatoriu din datele de validare din dreptunghiul respectiv aceasta (adică 1/4) este probabilitatea ca vecinul cel mai apropiat de (x_1, x_2) în setul de date de antrenament să aibă semnul opus (față de semnul lui (x_1, x_2)). Așadar, media erorii la testare cu clasificatorul 1-NN pe setul de validare este $1/2 \cdot 1/4 + 1/2 \cdot 1/4 = 1/4$.

d. 1/4.

Folosind algoritmul 21-NN, în dreptunghiul din stânga 3/4 din datele de antrenament sunt (în general) clasificate corect, iar restul de 1/4 (și anume, cele având semnul –) sunt clasificate eronat. Într-adevăr, la testarea unui punct oarecare (x_1, x_2) din acel dreptunghi, în 21-NN vecinătatea lui (x_1, x_2) , unul (cel mai apropiat vecin) este însuși (x_1, x_2) , iar dintre ceilalți 20 cei mai apropiati vecini 3/4 sunt (în general) de semn +, iar 1/4 de semn –. Analog se raționează pentru dreptunghiul din dreapta. Deci rata medie a erorii la antrenare este:

$$\frac{1}{2} \left(\frac{3}{4} \cdot 0 + \frac{1}{4} \cdot 1 \right) + \frac{1}{2} \left(\frac{3}{4} \cdot 0 + \frac{1}{4} \cdot 1 \right) = \frac{1}{4}.$$

e. 1/4.

În cazul primului dreptunghi, apare eroare atunci când un exemplu pozitiv (probabilitate de selectare: 3/4) este clasificat negativ (probabilitate: 0, fiind vorba de 21-NN) sau când un exemplu negativ (probabilitate de selectare: 1/4) este clasificat pozitiv (probabilitate: 1). ¹⁵⁶ Așadar, probabilitatea de a clasifica gresit un exemplu din primul dreptunghi este $3/4 \cdot 0 + 1/4 \cdot 1 = 1/4$. Cazul dreptunghiului din dreapta este similar, prin urmare rezultatul final este 1/4.

f. 0.

Datorită distribuției uniforme a datelor de antrenament, fiecare instanță (x_1, x_2) din setul de validare va avea majoritatea vecinilor — dintre cei mai apropiati, selectați de către algoritmul 21-NN din setul de date de antrenament — de același semn cu semnul lui (x_1, x_2) . (Și anume, în medie de 3 ori mai mulți vecini decât cei de semn contrar.) Prin urmare, fiecare punct din setul de validare va fi clasificat corect.

¹⁵⁶În 21-NN vecinătatea punctului considerat (pentru testare), spre deosebire de cazul erorii la antrenare, la CVLOO nu se mai include punctul respectiv.

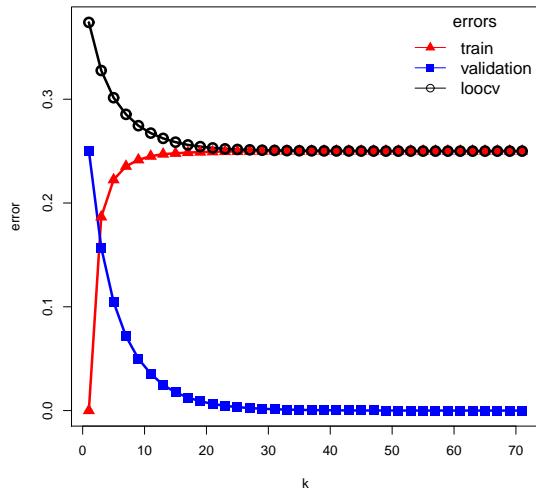
Observație (1): Comparând rezultatele obținute la punctele a și c pe de o parte cu cele de la punctele d și f (sau chiar b și e) pe de altă parte, se observă că are loc o relație exact de același gen cu cea din definiția fenomenului de *overfitting* (sau, *supra-specializare*).¹⁵⁷ erorile produse de algoritmii 1-NN și 21-NN la antrenare sunt în relația $0 < 1/4$ dar în relație inversă ($1/4 > 0$) la testare (respectiv $3/8 > 1/4$ la cross-validation).

Observație (2): Remarcați „sublinarea“ din expresia „fracția cea mai apropiată de media / rata erorii“ din enunț. Generarea aleatorie a datelor poate conduce la rezultate ușor diferite față de cele pe care le-am obținut mai sus. În urma realizării unei implementări,¹⁵⁸ au fost obținute următoarele rezultate de tip *eroare medie*, calculată în urma repetării de 100 de ori a generării datelor și aplicării algoritmilor k -NN, conform cerințelor din enunț:

- a. 0, b. 0.374022, c. 0.250472, d. 0.249342, e. 0.253088, f. 0.006436.

Constatăm că se verifică „previziunile“ noastre din rezolvarea de mai sus.

Evoluția celor trei tipuri de eroare (la antrenare, la validare și cross-validation cu metoda “leave-one-out”), pentru $k = 1, 3, \dots, 69, 71$ este prezentată în figura alăturată. Se observă convergența la aceeași valoare (aprox., 0.25) pentru eroarea la antrenare și eroarea CVLOO (pe setul de date de antrenare) începând din jurul valorii $k = 31$, precum și convergența la valori foarte apropiate de 0 pentru eroarea la testare pe setul de date de validare, începând din jurul valorii $k = 35$.



6.

(O versiune ipotetică pentru algoritmul k -NN:
selectarea celor mai apropiati vecini
nu pe baza calculării funcției de distanță,
ci folosind un oracol / “black box”)

CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 2.2-3

Se încearcă clasificarea unor puncte în spațiul euclidian bidimensional. Sunt date n instanțe P_1, P_2, \dots, P_n , precum și etichetările corespunzătoare c_1, c_2, \dots, c_n , unde c_1, c_2, \dots, c_n reprezintă valori dintr-o mulțime C . În schema de clasificare k -NN, fiecare element

¹⁵⁷Vă readucem aminte definiția lui Tom Mitchell pentru fenomenul de *overfitting* (vedeți cartea *Machine Learning*, pag. 67): două ipoteze h și h' obținute (eventual cu un același algoritm de clasificare automată) pe un set de date sunt în raport de overfitting dacă

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h'), \quad \text{dar} \quad \text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h'),$$

unde \mathcal{D} este distribuția reală a datelor.

¹⁵⁸Implementarea a fost făcută de către studentul Sebastian Ciobanu de la Facultatea de Informatică a Universității „Al. I. Cuza“ din Iași în semestrul I al anului universitar 2016-2017.

nou Q este clasificat cu eticheta majoritară obținută în cadrul vecinătății formate din cei mai apropiati k vecini.

Să presupunem că măsura de distanță nu este dată în mod explicit. În locul acesteia, aveți la dispozitie un “black box”. Dacă se introduc instanțele $P_{i_1}, P_{i_2}, \dots, P_{i_l}$ (unde l este un număr natural oarecare) și un punct Q , black box-ul returnează cel mai apropiat vecin al lui Q , adică un element $P_{i_0} \in \{P_{i_1}, P_{i_2}, \dots, P_{i_l}\}$, precum și clasificarea corespunzătoare (c_{i_0}).

- Este posibil să se construiască un algoritm de tip k -NN bazat doar pe acest black box? Dacă da, explicați cum anume, iar dacă nu, explicați de ce nu este posibil.
- Dacă, în schimb, acel black box returnează cei mai apropiati j vecini (și etichetele corespunzătoare), iar $j \neq k$, este posibil să se construiască un algoritm de tip k -NN bazat doar pe acest black box? Dacă da, explicați cum anume, iar dacă nu, explicați de ce nu este posibil.

Răspuns:

- Da. Se folosește black box-ul dându-i ca intrare mai întâi mulțimea de exemple P_1, P_2, \dots, P_n ; se obține cel mai apropiat vecin al lui Q și eticheta sa. Apoi se scoate instanța / punctul returnat din mulțimea de exemple. Se repetă acest procedeu de k ori, iar la final se va alege pentru Q eticheta corespunzătoare majorității din mulțimea de k vecini care au fost identificați de către black box.
- Dacă $j < k$, atunci se folosește black box-ul de $[k/j]$ ori, obținându-se $j * [k/j]$ cei mai apropiati vecini și clasificările acestora. Notăm cu V_1 mulțimea formată din acești vecini. În cazul în care $k \neq j * [k/j]$, pentru a obține restul de vecini necesari pentru k -NN, adică încă $k - j * [k/j]$ vecini, vom folosi black box-ul încă o dată. Vom nota cu V_2 mulțimea acestor noi j vecini. Dintre aceștia va trebui să alegem doar $k - j * [k/j]$ instanțe. Vom proceda astfel: considerăm o nouă mulțime de instanțe alcătuită din elementele lui V_2 și $j - (k - j * [k/j])$ dintre toți ceilalți vecini obținuți anterior (V_1). Aplicăm black box-ul pe acest nou set de date și vom obține cei mai apropiati j vecini pentru punctul Q . Printre aceștia se vor afla cei $k - j * [k/j]$ vecini căutați.

Dacă $j > k$ iar black box-ul nu acceptă intrări duplicate, atunci el nu poate fi folosit pentru a determina cei mai apropiati k vecini ai instanței de test.

Dar dacă $j > k$ și black box-ul acceptă intrări duplicate, este posibil să rezolvăm problema, cel puțin în situația în care cei mai apropiati j vecini ai lui Q se află toți la distanțe diferite față de acesta.¹⁵⁹ De exemplu, pentru $k = 1$ și $j = 3$ vom proceda astfel:

Pasul 1: Aplicăm black box-ul inputului P_1, \dots, P_n . Vom obține outputul $P_{i_1}, P_{i_2}, P_{i_3}$.

Pasul 2: Corespunzător fiecărui punct $P_{i_1}, P_{i_2}, P_{i_3}$, vom aplica pe rând black box-ul inputului

- $P_{i_1}, P_{i_1}, P_{i_2}, P_{i_3}$,
- $P_{i_1}, P_{i_2}, P_{i_2}, P_{i_3}$ și respectiv
- $P_{i_1}, P_{i_2}, P_{i_3}, P_{i_3}$.

Se poate constata că într-unul singur din aceste cazuri black box-ul returnează outputul $P_{i_1}, P_{i_2}, P_{i_3}$.¹⁶⁰ De pildă, dacă outputurile în aceste trei cazuri sunt

¹⁵⁹Rămâne de analizat varianta contrară.

¹⁶⁰Am presupus că black box-ul returnează exact j instanțe, chiar dacă între P_1, \dots, P_n există mai multe instanțe egal depărtate față de punctul Q , în speță mai multe instanțe situate la maximumul distanțelor dintre fiecare din cele j pe de o parte și punctul Q pe de altă parte.

- $P_{i_1}, P_{i_1}, P_{i_2}$,
- $P_{i_1}, P_{i_2}, P_{i_2}$,
- $P_{i_1}, P_{i_2}, P_{i_3}$,

rezultă că P_{i_3} este cel mai distanță dintre cei trei vecini ai lui Q .

Pasul 3: Considerând că lucrurile stau ca la finalul pasului precedent, pentru fiecare dintre punctele P_{i_1}, P_{i_2} vom aplica black box-ului următorul input:

- $P_{i_1}, P_{i_1}, P_{i_1}, P_{i_2}$ și respectiv
- $P_{i_1}, P_{i_2}, P_{i_2}, P_{i_2}$.

Dacă vom obține outputul $P_{i_1}, P_{i_1}, P_{i_1}$, va rezulta că P_{i_1} este mai apropiat de Q decât punctul P_{i_2} . Invers, dacă obținem outputul $P_{i_2}, P_{i_2}, P_{i_2}$, va rezulta că P_{i_2} este mai apropiat de Q decât punctul P_{i_1} .

7. (Algoritmul 1-NN: suprafețele de decizie
[și separatorii decizionali] depind de măsurile de distanță folosite)
CMU, 2008 fall, Eric Xing, HW1, pr. 3.1.2

Se dau două puncte din spațiul euclidian bidimensional: punctul $(-1, 0)$ clasificat negativ și punctul $(1, 0)$ clasificat pozitiv.

Clasificatorul 1-NN care folosește distanța euclidiană și are ca set de date de antrenament cele două puncte de mai sus are următoarea *formă analitică* (ușor de dedus):

- dat fiind un punct arbitrar (x, y) ,
în cazul în care $x > 0$, eticheta asignată punctului respectiv este $+$, iar în cazul $x < 0$ eticheta asignată este $-$;
 - dreapta $x = 0$ este *granița de decizie* (engl., decision boundary) corespunzătoare acestui clasificator.
- a. Care va fi forma analitică a clasificatorului 1-NN dacă în locul distanței euclidiene (indusă de normă L_2) se folosește distanța Manhattan (indusă de normă L_1)? Vă reamintim că distanța Manhattan dintre două puncte (x_1, y_1) și (x_2, y_2) este $|x_1 - x_2| + |y_1 - y_2|$.
- b. Dar dacă se folosește distanța [indusă de normă] L_∞ , definită în \mathbb{R}^2 prin

$$d((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\} ?$$

Răspuns:

a. Putem exprima distanța Manhattan dintre un punct oarecare (x, y) din plan și de fiecare dintre cele două puncte date în enunț — punctul $(1, 0)$ clasificat pozitiv și punctul $(-1, 0)$ clasificat negativ — astfel:

$$\begin{aligned} d_+ &\stackrel{\text{not.}}{=} d((x, y), (1, 0)) = |x - 1| + |y| \\ d_- &\stackrel{\text{not.}}{=} d((x, y), (-1, 0)) = |x + 1| + |y| \end{aligned}$$

În consecință, a stabili care dintre cele două distanțe (d_+ și d_-) este mai mică revine la a compara (doar) expresiile $|x - 1|$ și $|x + 1|$.

Audem următoarele cazuri:

- dacă $x > 1$, atunci

$$\left. \begin{array}{l} d_+ = x - 1 + |y| \\ d_- = x + 1 + |y| \end{array} \right\} \Rightarrow d_+ < d_- \Rightarrow \text{punctul } (x, y) \text{ va fi clasificat +}$$

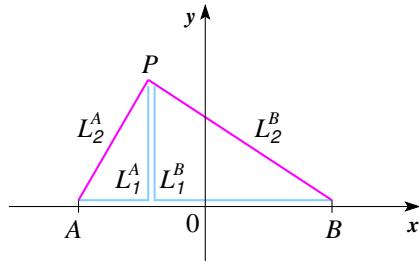
- dacă $x < -1$:

$$\left. \begin{array}{l} d_+ = -x + 1 + |y| \\ d_- = -x - 1 + |y| \end{array} \right\} \Rightarrow d_+ > d_- \Rightarrow \text{punctul } (x, y) \text{ va fi clasificat -}$$

- dacă $-1 \leq x \leq 1$:

$$\left. \begin{array}{l} d_+ = -x + 1 + |y| \\ d_- = x + 1 + |y| \end{array} \right\} \Rightarrow \begin{cases} d_+ < d_- \text{ pentru } x > 0 \text{ deci } (x, y) \text{ va fi clasificat +} \\ d_+ > d_- \text{ pentru } x < 0 \text{ deci } (x, y) \text{ va fi clasificat -} \\ d_+ = d_- \text{ dacă și numai dacă } x = 0. \end{cases}$$

Rezultă că granița de decizie a acestui clasificator este dreapta $x = 0$. Chiar mai mult: forma analitică a clasificatorului 1-NN care folosește distanța Manhattan pe setul de date din enunț este aceeași cu a clasificatorului 1-NN care folosește distanța euclidiană.



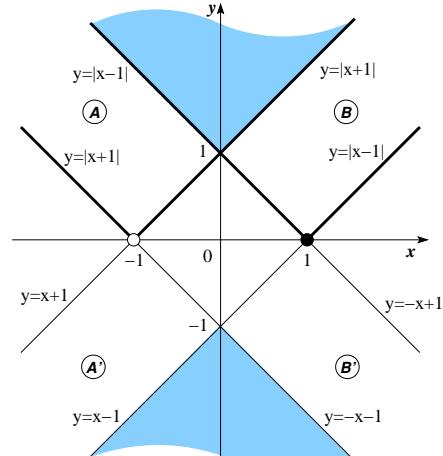
Observație: La concluzia de mai sus se putea ajunge mult mai ușor ținând cont de următoarea proprietate (demonstrabilă imediat pe cale geometrică): pentru orice două puncte A și B din \mathbb{R}^2 au loc următoarele relații de echivalență:

$$\begin{aligned} L_1(P, A) < L_1(P, B) &\Leftrightarrow L_2(P, A) < L_2(P, B) \text{ pentru } \forall P \in \mathbb{R}^2 \text{ și} \\ L_1(P, A) = L_1(P, B) &\Leftrightarrow L_2(P, A) = L_2(P, B) \text{ pentru } \forall P \in \mathbb{R}^2. \end{aligned}$$

b. Dacă se folosește distanța L_∞ , atunci conform definiției acestei metriki vom avea:

$$\begin{aligned} d_+ &= d((x, y), (1, 0)) \\ &= \max\{|x - 1|, |y|\} \\ d_- &= d((x, y), (-1, 0)) \\ &= \max\{|x + 1|, |y|\}. \end{aligned}$$

Mai întâi vom trasa graficele funcțiilor $y = |x - 1|$ și $y = |x + 1|$, și vom obține rezultatul din figura următoare.



Cazul i: Se observă că $|y| \geq |x - 1|$ și $|y| \geq |x + 1|$ pentru toate punctele (x, y) situate în zonele unghiulare hașurate. În consecință, d_+ va fi egal cu d_- pentru orice punct

din aceste zone. Așadar, zonele hașurate vor apartine graniței / suprafeței de decizie a clasificatorului nostru.

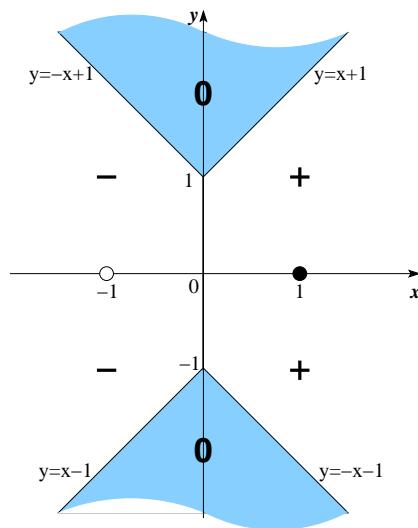
Cazul ii: Considerăm acum zonele identificate prin literele A, A', B și B' în figura de mai sus.¹⁶¹ Se observă ușor că pentru zonele A și A' avem $d_+ = \max\{|x - 1|, |y|\} = |x - 1|$ și $d_- = \max\{|x + 1|, |y|\} = |y|$, iar pentru zonele B și B' avem $d_+ = \max\{|x - 1|, |y|\} = |y|$ și $d_- = \max\{|x + 1|, |y|\} = |x + 1|$. În consecință, $d_- = |y| < |x - 1| = d_+$ pentru zonele A și A' , iar $d_+ = |y| < |x + 1| = d_-$ pentru zonele B și B' . Așadar, zonele A și A' vor fi clasificate negativ, iar zonele B și B' vor fi clasificate pozitiv.

Cazul iii: Pentru toate celelalte zone rămase în discuție — adică pentru orice punct (x, y) situat în afara celor două zone hașurate și în afara zonelor A, A', B, B' —, vom avea $d_+ = \max\{|x - 1|, |y|\} = |x - 1|$ și $d_- = \max\{|x + 1|, |y|\} = |x + 1|$. Din grafic se observă că $|x + 1| > |x - 1|$ pentru $x > 0$ și $|x - 1| > |x + 1|$ pentru $x < 0$, iar $|x + 1| = |x - 1|$ pentru $x = 0$. Așadar, sumarizând, în zone avem: $d_+ = \min\{d_+, d_-\}$ pentru $x > 0$ și $d_- = \min\{d_+, d_-\}$ pentru $x < 0$, iar $d_+ = d_-$ pentru $x = 0$.

Concluzionând, forma analitică a clasificatorului 1-NN care folosește distanța L_∞ este următoarea:

- (x, y) va fi etichetat cu $+$ dacă $x > 0$ și $-x - 1 < y < x + 1$;
- (x, y) va fi etichetat cu $-$ dacă $x < 0$ și $x - 1 < y < -x + 1$;
- în rest este vorba de suprafață de separare, adică locul geometric al punctelor (x, y) pentru care distanța față de cele două puncte din enunț este egală.

Reprezentarea grafică a suprafețelor de decizie este dată în figura următoare:



Este de remarcat faptul că pentru acest clasificator granița / suprafața de decizie nu este formată doar din drepte, ci este reuniunea unei drepte (axa Oy) cu două intersecții de (câte două) semiplane.

¹⁶¹Analitic, zona A este definită de punctele (x, y) care satisfac inecuațiile $|x + 1| < |y| < |x - 1|$ și $y > 0$; zona A' : $|x + 1| < |y| < |x - 1|$ și $y < 0$; zona B : $|x - 1| < |y| < |x + 1|$ și $y > 0$; zona B' : $|x - 1| < |y| < |x + 1|$ și $y < 0$.

8. (Algoritmul / metoda lui Shepard – aplicare)
CMU, 2002 fall, Andrew Moore, final exam, pr. 6.a-d

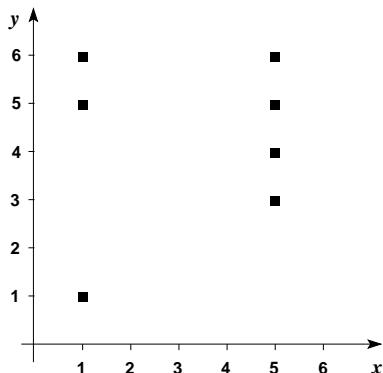
Figura de mai jos prezintă un set de date de antrenament cu un atribut de intrare $x \in \mathbb{R}$ și un atribut de ieșire $y \in \mathbb{R}$.

Vom estima din aceste date câteva valori ale unei funcții continue $f : \mathbb{R} \rightarrow \mathbb{R}$, folosind *metoda lui Shepard*. Aceasta este o variantă (de tip *regresie*) a algoritmului *k-NN* în care se iau în considerare toate punctele de antrenament, dar se aplică ponderi în funcție de distanță:

$$\hat{f}(x) \leftarrow \frac{\sum_i w(x, x_i) f(x_i)}{\sum_i w(x, x_i)}$$

Se va considera

$$w(x, x_i) = \begin{cases} 1, & \text{dacă } |x - x_i| \leq 3 \\ 0, & \text{în rest.} \end{cases}$$



Care va fi valoarea prezisă pentru funcția f pentru

- a. $x = 1?$ c. $x = 5?$
 b. $x = 3?$ d. $x = 6?$

Răspuns:

Din modul cum au fost definite ponderile w ajungem la concluzia că valoarea lui f pentru un punct oarecare x va fi calculată ca medie aritmetică a valorilor / componentelor y din acele date de antrenament pentru care abscisa (x') este situată la distanță de cel mult 3 unități de punctul x care ne interesează.

a. Avem $|1 - 1| = 0 \leq 3$ și $|1 - 5| = 4 > 3$, prin urmare vor fi luate în considerare doar valorile învățate pentru $x = 1$.

$$\hat{f}(1) = \frac{1 + 5 + 6}{3} = \frac{12}{3} = 4.$$

b. Avem $|3 - 1| = 2 \leq 3$ și $|3 - 5| = 2 \leq 3$, prin urmare vor fi luate în considerare și valorile învățate pentru $x = 1$ și cele pentru $x = 5$.

$$\hat{f}(3) = \frac{1 + 5 + 6 + 3 + 4 + 5 + 6}{7} = \frac{30}{7}.$$

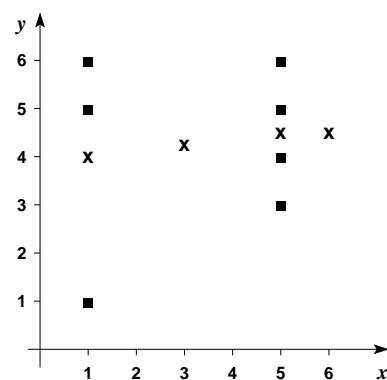
c. Avem $|5 - 1| = 4 > 3$ și $|5 - 5| = 0 \leq 3$, prin urmare vor fi luate în considerare doar valorile învățate pentru $x = 5$.

$$\hat{f}(5) = \frac{3 + 4 + 5 + 6}{4} = \frac{18}{4} = \frac{9}{2}.$$

d. Avem $|6 - 1| = 5 > 3$ și $|6 - 5| = 1 \leq 3$, prin urmare vor fi luate în considerare doar valorile învățate pentru $x = 5$, ca și în cazul precedent.

$$\hat{f}(6) = \hat{f}(5) = 4.5$$

Dacă vom plasa rezultatele de mai sus pe grafic și vom reprezenta punctele $(x, \hat{f}(x))$ sub forma unor cruciulițe, vom obține figura alăturată.



9.

(Asupra folosirii algoritmului k -NN
în spații (\mathbb{R}^p) de dimensiune (p) mare:
un avertisment: „blestemul marilor dimensiuni“)
■ CMU, 2010 fall, Aarti Singh, HW2, pr. 2.2

Considerăm punctele x_1, x_2, \dots, x_n distribuite în mod independent și uniform într-o sferă (notată cu B) care are raza egală cu unitatea¹⁶² și centrul în O , originea spațiului \mathbb{R}^p . Așadar, $B = \{x : \|x\|^2 \leq 1\} \subset \mathbb{R}^p$, unde $\|x\| = \sqrt{x \cdot x}$, iar operatorul \cdot desemnează produsul scalar din \mathbb{R}^p .

În această problemă veți studia „mărimea“ vecinătății de tip 1-NN pentru originea O și cum anume variază ea în raport cu dimensiunea p . În acest fel, veți putea vedea care sunt dezavantajele folosirii algoritmului k -NN într-un spațiu de dimensiune mare.

Din punct de vedere formal, „mărimea“ menționată mai sus va fi identificată cu d^* , distanța de la O la cel mai apropiat vecin din mulțimea $\{x_1, x_2, \dots, x_n\}$:

$$d^* \stackrel{\text{not.}}{=} \min_{1 \leq i \leq n} \|x_i\|.$$

Observație: Din moment ce eșantionul $\{x_1, x_2, \dots, x_n\}$ este generat în mod aleatoriu, distanța d^* poate fi văzută ca fiind [produsă de către] o variabilă aleatoare.

a. În cazul particular $p = 1$, calculați expresia *funcției de distribuție cumulative*¹⁶³ a lui d^* (văzută ca variabilă aleatoare), și anume $P(d^* \leq t)$ pentru $t \in [0, 1]$.

b. Determinați expresia *funcției de distribuție cumulative* (c.d.f.) a lui d^* în cazul general, adică pentru $p \in \{1, 2, 3, \dots\}$.

Sugestie: Puteți folosi următoarea formulă pentru volumul unei sfere de rază r din \mathbb{R}^p :

$$V_p(r) = \frac{(r\sqrt{\pi})^p}{\Gamma\left(\frac{p}{2} + 1\right)},$$

unde Γ reprezintă funcția Gamma a lui Euler,¹⁶⁴ care are proprietățile:

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}, \quad \Gamma(1) = 1, \quad \text{iar } \Gamma(x+1) = x\Gamma(x) \text{ pentru } x > 0.$$

c. Care este *mediana* variabilei aleatoare d^* (adică, valoarea lui t pentru care $P(d^* \leq t) = 1/2$)? Va trebui ca răspunsul să fie formulat în funcție de n și p (dimensiunea eșantionului și, respectiv, dimensiunea spațiului din care se face extragerea instantelor, \mathbb{R}^p).

Pentru $n = 100$, alcătuți un grafic cu valorile [funcției] mediane pentru $p = 1, 2, 3, \dots, 100$. Valorile lui p vor fi plasate pe axa Ox , iar valorile medianei pe axa Oy . Ce observați?

d. Folosind funcția de distribuție cumulative (c.d.f.) de la punctul b, determinați cât de mare ar trebui să fie n (mărimea eșantionului) astfel încât

$$P(d^* \leq 0.5) \geq 0.9,$$

¹⁶²Termenul folosit în limba engleză pentru o astel de sferă este *unit ball*.

¹⁶³Engl., cumulative distribution function, c.d.f.

¹⁶⁴Vedeți problema 28.b de la capitolul de *Fundamente*.

¹⁶⁵Se verifică ușor că pentru $p = 3$ se obține volumul sferei: $V_3(r) = \frac{(r\sqrt{\pi})^3}{\frac{3}{4}\sqrt{\pi}} = \frac{4\pi r^3}{3}$. Pentru demonstrarea unora dintre proprietățile funcției Γ indicate mai sus, vedeți problema 41 de la capitolul de *Fundamente*.

adică, cu probabilitate de cel puțin 9/10, distanța d^* de la originea O la cel mai apropiat vecin să fie mai mică decât 1/2 (adică, jumătate din distanța de la O la marginea sferei). Va trebui să formulați răspunsul ca expresie a unei funcții în raport cu variabila p . Reprezentați grafic valorile acestei funcții pentru $p = 1, 2, \dots, 20$, plasând valorile lui p pe axa Ox și valorile funcției pe axa Oy . Ce observați?

Sugestie: Pentru $\ln(1 - x)$, puteți face apel la dezvoltarea sa sub formă de serie Taylor:

$$\ln(1 - x) = -\sum_{i=1}^{\infty} \frac{x^i}{i} \text{ pentru } -1 \leq x < 1.$$

e. În urma rezolvării punctelor de mai sus, ce puteți spune despre dezavantajele algoritmului k -NN în raport cu [diferitele valori posibile pentru] n și p ?

Răspuns:

a. Pentru $p = 1$, sfera de rază 1 este intervalul $[-1, 1]$, iar funcția de distribuție cumulativă va avea expresia:

$$F_{n,1}(t) \stackrel{\text{not.}}{=} P(d^* \leq t) = 1 - P(d^* > t) = 1 - P(\|x_i\| > t, i = 1, 2, \dots, n)$$

Tinem cont de presupozitia de independentă la generarea punctelor x_i , rezultă:

$$F_{n,1}(t) = 1 - \prod_{i=1}^n P(\|x_i\| > t) = 1 - (1 - t)^n.$$

b. În cazul general, adică pentru p un număr natural oarecare nenul, fixat, vom exprima, mai întâi $P(d^* \leq t)$ exact ca mai înainte:

$$\begin{aligned} F_{n,p}(t) \stackrel{\text{not.}}{=} P(d^* \leq t) &= 1 - P(d^* > t) = 1 - P(\|x_i\| > t, i = 1, 2, \dots, n) \\ &\stackrel{\text{indep. cdt.}}{=} 1 - \prod_{i=1}^n P(\|x_i\| > t). \end{aligned}$$

Apoi, ținând cont de presupozitia de uniformitate la generarea punctelor x_i și, folosind notația $V_p(t)$ pentru volumul sferei de rază t , obținem:

$$F_{n,p}(t) = 1 - \left(\frac{V_p(1) - V_p(t)}{V_p(1)} \right)^n = 1 - \left(1 - \frac{V_p(t)}{V_p(1)} \right)^n.$$

În sfârșit, folosind formula sugerată în enunț pentru V_p , rezultă imediat că $F_{n,p}(t) = 1 - (1 - t^p)^n$.

c. Pentru a afla valoarea mediană corespunzătoare variabilei aleatoare d^* , vom rezolva ecuația $P(d^* \leq t) = 1/2$ în funcție de t :

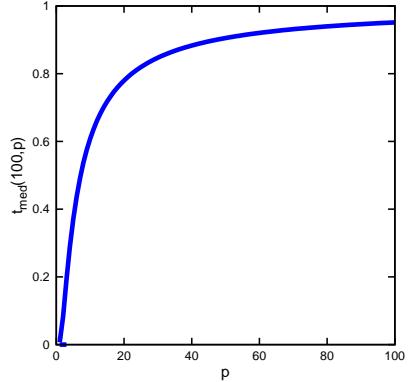
$$\begin{aligned} P(d^* \leq t) = \frac{1}{2} &\Leftrightarrow F_{n,p}(t) = \frac{1}{2} \stackrel{b}{\Leftrightarrow} 1 - (1 - t^p)^n = \frac{1}{2} \\ &\Leftrightarrow (1 - t^p)^n = \frac{1}{2} \Leftrightarrow 1 - t^p = \frac{1}{2^{1/n}} \\ &\Leftrightarrow t^p = 1 - \frac{1}{2^{1/n}} \end{aligned}$$

Prin urmare,

$$t_{med}(n, p) = \left(1 - \frac{1}{2^{1/n}}\right)^{1/p}.$$

Graficul funcției $t_{med}(100, p)$ pentru $p = 1, 2, \dots, 100$ este cel din figura alăturată.

Să observă că sfera minimală care conține cel mai apropiat vecin (un x_i , cu $i \in \{1, 2, \dots, n\}$) al originii O se lărgește foarte repede pe măsură ce p crește. Pentru valori ale lui p mai mari decât 10, majoritatea dintre cele 100 de instanțe de antrenament sunt mai aproape de conturul sferei de rază 1 decât de originea O .



d. Putem scrie următorul sir de echivalențe:

$$\begin{aligned} P(d^* \leq 0.5) \geq 0.9 &\Leftrightarrow F_{n,p}(0.5) \geq 0.9 \Leftrightarrow \\ &\Leftrightarrow 1 - \left(1 - \frac{1}{2^p}\right)^n \geq \frac{9}{10} \Leftrightarrow \left(1 - \frac{1}{2^p}\right)^n \leq \frac{1}{10} \\ &\Leftrightarrow n \cdot \ln\left(1 - \frac{1}{2^p}\right) \leq -\ln 10 \\ &\Leftrightarrow n \geq \frac{\ln 10}{-\ln\left(1 - \frac{1}{2^p}\right)} \end{aligned}$$

Se poate vedea imediat că membrul din partea dreaptă a inegalității de mai sus tinde la $+\infty$ pentru $p \rightarrow \infty$. Este necesar să vedem *cât de repede* are loc această tindere la infinit. Pentru aceasta, vom folosi descompunerea lui $-\ln(1 - 1/2^p)$ sub forma unei serii Taylor (luând $x = 1/2^p$):¹⁶⁶

$$\begin{aligned} P(d^* \leq 0.5) \geq 0.9 &\Rightarrow n \geq (\ln 10) 2^p - \frac{1}{1 + \frac{1}{2} \cdot \frac{1}{2^p} + \frac{1}{3} \cdot \frac{1}{2^{2p}} + \dots + \frac{1}{n} \frac{1}{2^{(n-1)p}} + \dots} \\ &\Rightarrow n > \frac{4}{3} 2^{p-1} \ln 10. \end{aligned}$$

Pentru obținerea ultimei inegalități de mai sus am ținut cont de faptul că inegalitatea $\frac{1}{n \cdot 2^{(n-1)p}} \leq \frac{1}{2^n} \Leftrightarrow 2^n \leq n \cdot 2^{(n-1)p}$ are loc pentru orice $p \geq 1$, și $n \geq 2$,¹⁶⁷ deci

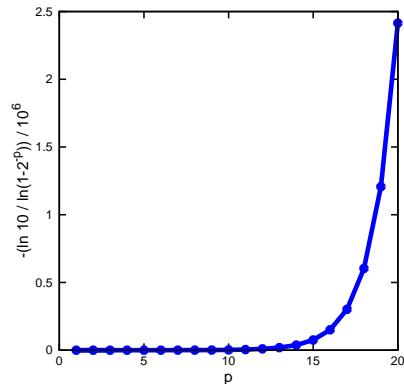
$$\begin{aligned} 1 + \frac{1}{2} \cdot \frac{1}{2^p} + \frac{1}{3} \cdot \frac{1}{2^{2p}} + \dots + \frac{1}{n} \cdot \frac{1}{2^{(n-1)p}} + \dots \\ \leq 1 + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^n} + \dots \end{aligned}$$

¹⁶⁶ $-\ln(1 - x) = x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \dots + \frac{1}{n}x^n + \dots$ pentru orice $x \in (-1, +1)$.

¹⁶⁷ Demonstrația se poate face prin inducție după valorile lui p .

$$\begin{aligned}
 &< \left[1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^n} + \dots \right] - \frac{1}{2} \\
 &\rightarrow \frac{1}{1 - \frac{1}{2}} - \frac{1}{2} = 2 - \frac{1}{2} = \frac{3}{2}.
 \end{aligned}$$

Observați că limita de mai sus este o limită *superioară*, de aceea inegalitatea se păstrează până la final.



Așadar, rezultă că n crește în mod exponential în raport cu p .¹⁶⁸ Graficul pentru marginea inferioară dedusă mai sus ($-\ln 10 / \ln(1 - 2^{-p})$) este cel din figura alăturată.

Se observă că, într-adevăr, creșterea acestei margini inferioare (deci și a lui n) este exponentială.

e. Conform intuiției, clasificatorul k -NN se comportă bine atunci când instanța de test x_q este situată într-o vecinătate densă de instanțe de antrenament. Totuși, analiza teoretică de mai sus ne arată că pentru a ne asigura că punctul x_q are o vecinătate densă, numărul tuturor instanțelor de antrenament trebuie să crească exponential în raport cu p , ceea ce nu este fezabil pentru valori mari ale lui p . (Parametrul p este dimensiunea spațiului în care se lucrează, adică numărul de trăsături ale instanțelor de antrenament și, respectiv, de test).

În consecință, pentru aplicațiile practice în care se folosesc date cu multe attribute, este recomandat ca execuția algoritmului k -NN să fie precedată de efectuarea unei „selectii de trăsături“ (engl., feature selection).

10. (Algoritmul 1-NN [comparativ cu clasificatorul Bayes Optimal]: o margine superioară pentru eroarea medie asimptotică [la antrenare])
■ CMU, 2005 spring, C. Guestrin, T. Mitchell, HW3, pr. 1

Un rezultat interesant obținut de Cover și Hart (1967) arată că, atunci când numărul datelor de antrenament tinde la infinit, iar datele de antrenament umplu spațiul în mod dens, rata medie a erorii produsă de către clasificatorul 1-NN este mărginită superior de dublul ratei medii a erorii pentru clasificatorul Bayes Comun (engl., Joint Bayes), care este numit adeseori și *Bayes Optimal* (engl., Optimal Bayes).

¹⁶⁸Mai detaliat: n , numărul de instanțe de antrenament necesare pentru a ne asigura că d^* (distanța până la cel mai apropiat vecin al originii O) este cu o probabilitate mare (și anume, $9/10$) mai mică decât 0.5 crește în mod exponential în raport cu p .

La acest exercițiu vi se va arăta, pas cu pas, cum se demonstrează rezultatul lui Cover și Hart în cazul particular al clasificării binare. Așadar, fie x_1, x_2, \dots instanțele de antrenament, iar y_1, y_2, \dots etichetele corespunzătoare, cu $y_i \in \{0, 1\}$. Putem considera instanțele x_i ca fiind puncte într-un spațiu euclidian d -dimensional.

Notăm $p_y(x) = P(X = x | Y = y)$ probabilitatea condiționată care reprezintă distribuția instanțelor din clasa y . Vom presupune că aceste probabilități condiționate sunt continue în raport cu variabila x și că $p_y(x) \in (0, 1)$ pentru orice x și orice y . Notăm cu θ probabilitatea ca un exemplu de antrenament selectat în mod aleatoriu să fie din clasa 1, așadar $\theta \stackrel{\text{not.}}{=} P(Y = 1)$. Din nou, presupunem că $\theta \in (0, 1)$.

- Calculați probabilitatea ca o instanță oarecare x să aparțină clasei 1: $q(x) \stackrel{\text{not.}}{=} P(Y = 1 | X = x)$. Exprimăți $q(x)$ în funcție de $p_0(x), p_1(x)$ și θ .
- Clasificatorul Bayes Optimal asignează unui punct dat x cea mai probabilă clasă, $\operatorname{argmax}_y P(Y = y | X = x)$. (Aceasta implică faptul că algoritmul Bayes Optimal maximizează probabilitatea clasificării corecte a tuturor datelor.) Considerând o instanță oarecare x , calculați probabilitatea ca x să fie clasificat greșit folosind clasificatorul Bayes Optimal, în funcție de probabilitatea $q(x) \stackrel{\text{not.}}{=} P(Y = 1 | X = x)$ care tocmai a fost calculată la punctul precedent. Veți desemna această nouă probabilitate cu $Error_{Bayes}(x)$.
- Acum considerăm clasificatorul 1-NN. Acesta îi asignează unei instanțe oarecare de test x eticheta celei mai apropiate instanțe de antrenament x' . Dată fiind o instanță de antrenament x (aleasă în mod arbitrar, dar fixată), calculați eroarea „așteptată“ (engl., expected error) produsă de către clasificatorul 1-NN, adică probabilitatea ca instanța x să fie clasificată greșit. Notați această probabilitate cu $Error_{1-NN}(x)$ și exprimați-o sub forma unei funcții definită în raport cu probabilitățile $q(x)$ și $q(x')$.
- În *cazul asimptotic*, numărul de exemple de antrenament al fiecărei clase tinde la infinit, iar datele de antrenament umplu spațiul în mod dens. Atunci $q(x') \rightarrow q(x)$, unde, ca și mai sus, x' este cel mai apropiat vecin al lui x .¹⁶⁹ Făcând această substituție în rezultatul obținut la punctul anterior, deduceți expresia erorii asimptotice pentru clasificatorul 1-NN în punctul x , adică $\lim_{x' \rightarrow x} Error_{1-NN}(x)$, în funcție de probabilitatea $q(x)$.
- Arătați că eroarea asimptotică obținută la punctul d este mai mică decât dublul erorii clasificatorului Bayes Optimal obținută la punctul b , adică:

$$\lim_{x' \rightarrow x} Error_{1-NN}(x) \leq 2Error_{Bayes}(x).$$

În final, din această inegalitate deduceți relația corespunzătoare între ratele medii ale erorilor:¹⁷⁰

$$E[\lim_{n \rightarrow \infty} Error_{1-NN}] \leq 2E[Error_{Bayes}].$$

Răspuns:

- Conform enunțului, $p_1(x) \stackrel{\text{not.}}{=} P(X = x | Y = 1)$, $p_0(x) \stackrel{\text{not.}}{=} P(X = x | Y = 0)$, $q(x) \stackrel{\text{not.}}{=} P(Y = 1 | X = x)$ și $\theta \stackrel{\text{not.}}{=} P(Y = 1)$. Putem calcula probabilitatea $q(x)$ în funcție de $p_1(x)$, $p_0(x)$ și θ folosind formula lui Bayes:

¹⁶⁹Adică, $P(Y = 1 | X = x') \rightarrow P(Y = 1 | X = x)$. Aceasta se justifică ținând cont de continuitatea lui $p_y(x) \stackrel{\text{not.}}{=} P(X = x | Y = y)$ care a fost asumată în enunț și, de asemenea, de rezultatul obținut la punctul a.

¹⁷⁰Cititorul atent va remarca faptul că în expresia de mai jos ($E[\lim_{n \rightarrow \infty} Error_{1-NN}]$) s-a înlocuit $\lim_{x \rightarrow x'}$ (folosită anterior) cu $\lim_{n \rightarrow \infty}$, pentru că se face trecerea la medii. Când $n \rightarrow \infty$, conform presupozitărilor din enunț, rezultă $x \rightarrow x'$ pentru orice x .

$$\begin{aligned}
q(x) &\stackrel{\text{Bayes}}{=} \frac{P(X = x|Y = 1)P(Y = 1)}{P(X = x)} \\
&= \frac{P(X = x|Y = 1)P(Y = 1)}{P(X = x|Y = 1)P(Y = 1) + P(X = x|Y = 0)P(Y = 0)} \\
&= \frac{p_1(x)\theta}{p_1(x)\theta + p_0(x)(1 - \theta)}
\end{aligned}$$

b. Este imediat faptul următor: probabilitatea ca algoritmul Bayes Optimal să greșească este $P(Y = 0|X = x)$ în cazul în care $P(Y = 1|X = x) \geq P(Y = 0|X = x)$, respectiv $P(Y = 1|X = x)$ atunci când $P(Y = 0|X = x) \geq P(Y = 1|X = x)$.¹⁷¹ Altfel spus,

$$\begin{aligned}
Error_{Bayes}(x) &= \min\{P(Y = 0|X = x), P(Y = 1|X = x)\} \\
&= \min\{1 - q(x), q(x)\} = \begin{cases} q(x) \text{ în cazul } q(x) \in [0, 1/2] \\ 1 - q(x) \text{ în cazul } q(x) \in (1/2, 1]. \end{cases}
\end{aligned}$$

c. Algoritmul 1-NN greșește atunci când instanța de antrenament x are eticheta 1 iar x' , cel mai apropiat vecin al lui x , are eticheta 0, sau invers, adică atunci când x are eticheta 0 iar x' are eticheta 1. În consecință, folosind algoritmul 1-NN, eroarea „așteptată“ la clasificarea lui x este:

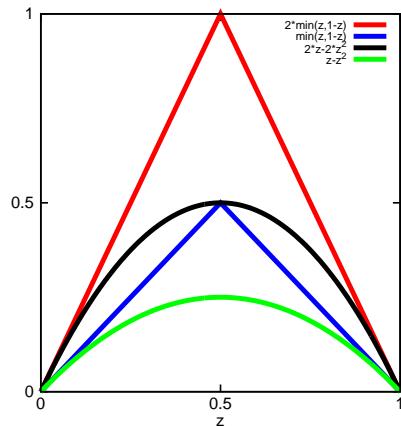
$$\begin{aligned}
Error_{1-NN}(x) &= P(Y = 1|X = x)P(Y = 0|X = x') + \\
&\quad P(Y = 0|X = x)P(Y = 1|X = x') \\
&= q(x)(1 - q(x')) + (1 - q(x))q(x').
\end{aligned}$$

d. Este imediat că $\lim_{x' \rightarrow x} Error_{1-NN}(x) = 2q(x)(1 - q(x))$.

e. Se arată imediat că $z - z^2 \leq z$ pentru $\forall z$, deci și pentru $z \in [0, 1/2]$, iar $z - z^2 \leq 1 - z$ pentru $\forall z$, deci și pentru $z \in [1/2, 1]$. Așadar, pentru orice x , vom avea:

$$q(x)(1 - q(x)) \leq \begin{cases} q(x) \text{ dacă } q(x) \in [0, 1/2] \\ 1 - q(x) \text{ dacă } q(x) \in (1/2, 1]. \end{cases}$$

Coroborând cu rezultatul de la punctul b, obținem: $2q(x)(1 - q(x)) \leq 2Error_{Bayes}(x)$ pentru orice x .



Combinând acest rezultat cu egalitatea de la punctul d, rezultă că inegalitatea

$$\lim_{n \rightarrow \infty} Error_{1-NN}(x) = \lim_{x' \rightarrow x} Error_{1-NN}(x) \leq 2Error_{Bayes}(x)$$

¹⁷¹Am ținut cont de proprietatea (evidentă) $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y) = P(Y = y|X = x)P(X = x)$, care are loc pentru orice x și y .

este adevărată pentru orice x . Înmulțind ambii membri ai acestei inegalități cu $P(x)$ și însumând după toate valorile lui x — de fapt, integrând în raport cu x —, obținem:

$$E[\lim_{n \rightarrow \infty} \text{Error}_{1\text{-NN}}] \leq 2E[\text{Error}_{\text{Bayes}}].$$

Așadar, am demonstrat că media (sau: rata medie a) erorii asimptotice a algoritmului 1-NN este cel mult dublul mediei (sau: ratei medii a) erorii algoritmului Bayes Optimal.

Observația 1: Această margine a erorii asimptotice nu se păstrează și în cazul neasimtotic, unde numărul de exemple de antrenament este finit.

Observația 2: La fel, se poate arăta că $2z - 2z^2 \geq z$ pentru $\forall z \in [0, 1/2]$ și $2z - 2z^2 \geq 1 - z$ pentru $\forall z \in [1/2, 1]$. Luând din nou $z = q(x)$ și ținând cont de rezultatul de la punctul b, obținem că $2q(x)(1 - q(x)) \geq \text{Error}_{\text{Bayes}}(x)$, pentru orice x . Combinând această inegalitate cu egalitatea de la punctul d, rezultă o nouă inegalitate:

$$\lim_{n \rightarrow \infty} \text{Error}_{1\text{-NN}}(x) = \lim_{x' \rightarrow x} \text{Error}_{1\text{-NN}}(x) \geq \text{Error}_{\text{Bayes}}(x) \text{ pentru orice } x.$$

În final, trecând la medii, obținem următoarea inegalitate (care era de altfel de așteptat):

$$E[\lim_{n \rightarrow \infty} \text{Error}_{1\text{-NN}}] \geq E[\text{Error}_{\text{Bayes}}].$$

Observația 3 (preluată din *An Elementary Introduction to Statistical Learning Theory*, de Sanjeev Kulkarni și Gilbert Harman, 2011, pag. 69): În mod intuitiv, dacă mărim valoarea lui k , ar trebui ca eroarea medie a algoritmului k -NN să se reducă. Într-adevăr, în anumite condiții (dar nu în orice condiții!) se poate arăta că are loc următoarea inegalitate dublă:

$$E[\text{Error}_{\text{Bayes}}] \leq E[\lim_{n \rightarrow \infty} \text{Error}_{k\text{-NN}}] \leq \left(1 + \frac{1}{k}\right) E[\text{Error}_{\text{Bayes}}].$$

Este de remarcat faptul că există distribuții probabilistice ale datelor pentru care clasificatorul 1-NN se comportă mai bine decât k -NN pentru orice $k \neq 1$.

Observația 4 (preluată din aceeași lucrare, *An Elementary Introduction to Statistical Learning Theory*, citată mai sus): Dacă lucrăm cu k_n -NN, adică îl fixăm pe k în funcție de n (numărul instanțelor de antrenament), se poate demonstra că în cazul în care $\frac{k_n}{n} \rightarrow 0$ pentru $n \rightarrow \infty$ (de exemplu, $k_n = \sqrt{n}$), se obține:

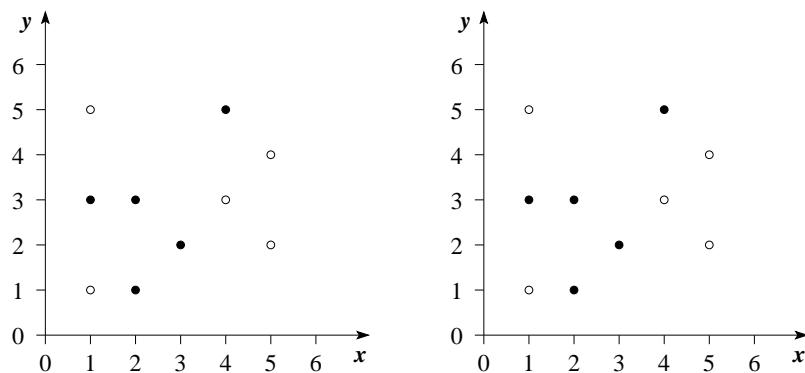
$$E[\lim_{n \rightarrow \infty} \text{Error}_{k_n\text{-NN}}] = E[\text{Error}_{\text{Bayes}}].$$

Aceasta înseamnă că, la limită, algoritmul k_n -NN se comportă la fel de bine ca algoritmul Bayes Optimal!

11. (Comparație între algoritmii 1-NN și ID3:
zone și suprafețe de decizie)
■ prelucrare de Liviu Ciortuz, după
CMU, 2007 fall, Carlos Guestrin, HW2, pr 1.4

Pe setul de date de mai jos desenați *granițele de decizie* și apoi hașurați *suprafețele de decizie* produse de

- a. algoritmul 1-NN (veți obține deci diagrama Voronoi);
 b. algoritmul ID3 extins cu capacitatea de a procesa atribute cu valori continue.



Răspuns:

a. *Algoritmul 1-NN:*

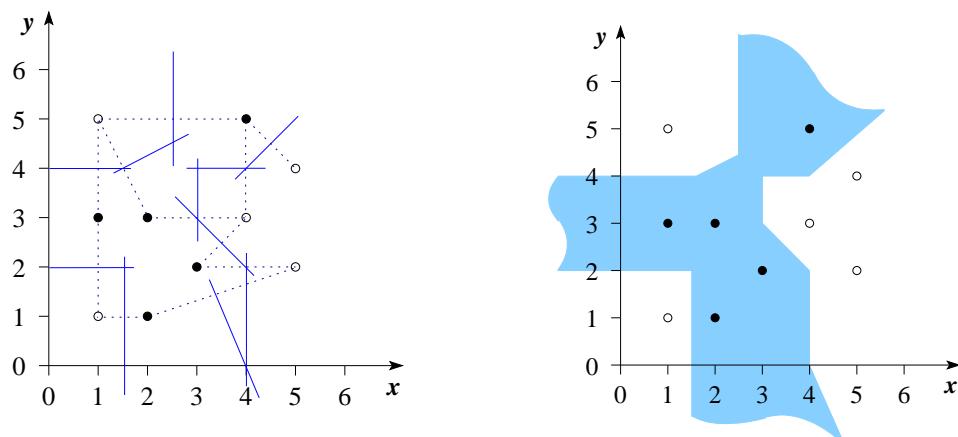
Pentru a defini suprafețele de decizie în acest caz se procedează în felul următor:

- se trasează mediatoarele segmentelor de dreaptă determinate de perechi de puncte din setul de antrenament care sunt etichetate în mod diferit;
- se stabilesc intersecțiile acestor mediatoare; acest lucru este reprezentat în figura de mai jos, partea stângă;
- apoi se marchează pe aceste mediatoare acelle segmente (determinate de intersecții) care determină zonele de decizie corespunzătoare clasificatorului 1-NN.

Observații:

1. De fapt, întrucât nu este necesar să se lucreze cu toate perechile de instanțe cu etichete diferite sunt relevante pentru clasificarea unei instanțe / zone, în timpul „execuției“ punctelor de mai sus este foarte util să se țină cont de următoarea regulă / euristică de *ghidare*: alegerea perechilor de instanțe și apoi a segmentelor de pe mediatoare se va face urmărind delimitarea zonelor corespunzătoare instanțelor negative (○) de zonele corespunzătoare instanțelor pozitive (●).
2. Suplimentar, în jurul fiecărui punct de antrenament *A* se poate identifica câte o zonă [convexă] care va constitui multimea punctelor mai apropiate de *A* decât de oricare alt punct din setul de date de antrenament. (Toate punctele din această zonă convexă vor avea aceeași clasificare / etichetă ca și punctul *A*.) Aceasta este ușor de văzut pentru instanțele negative (○) din cazul de față.

În figura următoare, în partea dreaptă am hașurat zona corespunzătoare instanțelor pozitive (●).

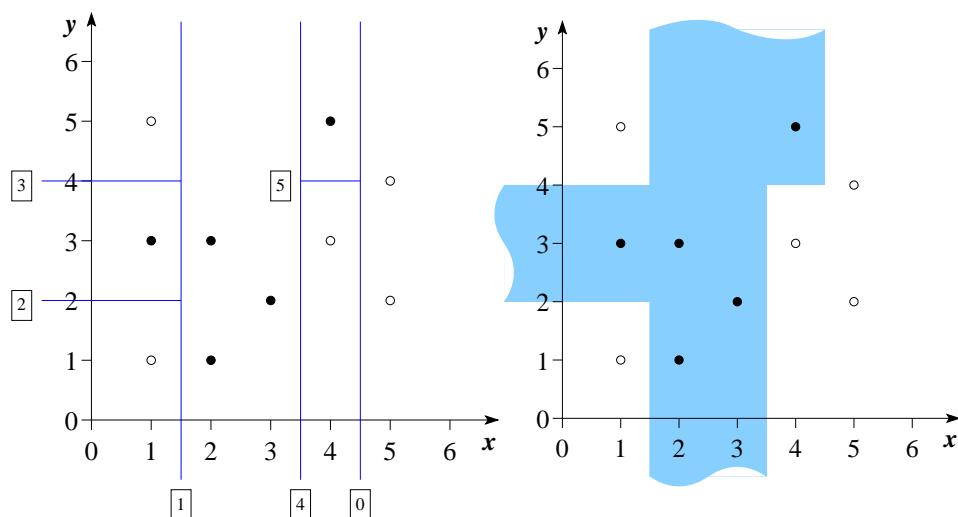


b. *Algoritmul ID3:*

Pentru construirea suprafeteelor de decizie ale algoritmului ID3,

- mai întâi se determină valorile-prag pentru teste (adică, punctele de splitare de pe fiecare axă) și apoi se alege testul corespunzător nodului rădăcină din arborele ID3; se trasează o dreaptă prin punctul respectiv, paralelă cu cealaltă axă. În cazul nostru, testul din nodul rădăcină va fi $x < 4.5$.
- pentru fiecare test / split ulterior se trasează o semidreaptă (sau un segment de dreaptă) mărginit(ă) la un capăt de dreapta corespunzătoare nodului-părinte.

În figura de mai jos, în partea stângă am vizualizat toate testele / split-urile realizate de algoritmul ID3 pentru a învăța complet arborele de decizie, iar în partea dreaptă, ca și mai sus, am păstrat doar frontierele dintre zonele cu clasificări diferite.



Se observă că suprafetele de decizie determinate de cei doi algoritmi nu sunt identice, dar sunt totuși asemănătoare într-o anumită măsură (pentru că ambele sunt consistente cu datele de antrenament).

Observații:

3. Este de reținut faptul că suprafețele de decizie produse de algoritm ID3 nu sunt neapărat unic determinate, fiindcă sunt situații în care două teste diferite pot conduce la același câștig de informație. De exemplu, dacă în exercițiul nostru am avea de partionat la un moment dat mulțimea formată din instanțele de antrenament $(1, 1), (1, 3), (2, 1), (2, 3), (3, 2)$, atunci testele $x > 1.5$ și $y > 1.5$ ar produce același câștig de informație, iar suprafețele de decizie rezultante ar fi determinate (în mod diferit!) de ce anume alegem ca prim test.
4. De asemenea, trebuie să scoatem în evidență faptul că pragurile / spliturile care sunt calculate de algoritm ID3 pentru un același atribut continuu pot差别 de la un nod de test la altul. De exemplu, la nodul rădăcină (nodul 0), atunci când se calculează câștigul de informație maxim, pentru atributul y se iau în calcul pragurile 1.5, 2.5, 3.5, și 4.5, în vreme ce la nodurile 2 și / sau 3 (vedeți figura de mai sus, partea stângă) se analizează pragurile 2 și 4, întrucât seturile / partiile de instanțe asignate acestor noduri sunt diferite!

12. (Comparație între algoritmii 1-NN, ID3 cu atribute continue și SVM:
eroarea la antrenare, eroarea la CVLOO)

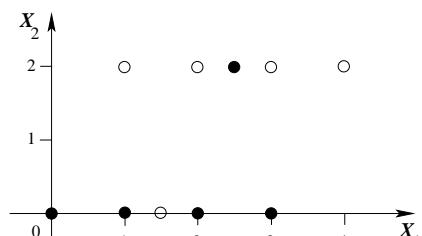
• prelucrare de Liviu Ciortuz, după
CMU, 2003 fall, T. Mitchell, A. Moore, midterm exam, pr. 5

Folosind metoda 1-NN cu distanță euclidiană, învățăm un clasificator cu două valori pentru atributul de ieșire, $Y = 0$ și $Y = 1$, pornind de la datele de antrenament din tabelul de mai jos (X_1 și X_2 sunt atribute de intrare).

	X_1	X_2	Y
a.	0	0	1
	1	0	1
	2	0	1
b.	2.5	2	1
	3	0	1
c.	1	2	0
	1.5	0	0
d.	2	2	0
	3	2	0
	4	2	0

a. Care este eroarea la antrenare (exprimată ca număr de exemple clasificate eronat)?
b. Care este eroarea la cross-validation folosind metoda "Leave-One-Out"?
c. Răspundeți la întrebările de mai sus, considerând acum arbori de decizie cu atribute numerice continue în locul metodei 1-NN.
d. Răspundeți la întrebările a și b , considerând mașini cu vectori-suport în locul metodei 1-NN. (Se vor considera doar SVM-uri în cazul liniar cu margine "soft", cu un parametru C suficient de mare pentru a minimiza numărul de instanțe de antrenament clasificate eronat.)

Răspuns:



Reprezentarea datelor în planul euclidian este
cea din figura alăturată.

- a. După cum am explicat și la exercițiul 2, întrucât datele de antrenament nu conțin inconsistențe, eroarea la antrenare produsă de algoritm 1-NN va fi 0.

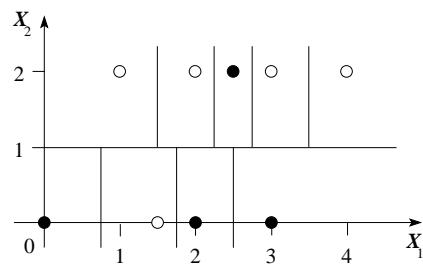
b. Comportamentul algoritmului la cross-validation cu metoda “Leave-One-Out” este cel descris în tabelul următor:

Data	Eticheta	Vecinătate	Clasificare la CVLOO	Eroare?
(0; 0)	1	(1; 0)	1	nu
(1; 0)	1	(1.5; 0)	0	da
(2; 0)	1	(1.5; 0)	0	da
(2, 5; 2)	1	(2; 2)/(3; 2)	0	da
(3; 0)	1	(2; 0)	1	nu
(1; 2)	0	(2; 2)	0	nu
(1, 5; 0)	0	(1; 0)/(2; 0)	1	da
(2; 2)	0	(2.5; 2)	1	da
(3; 2)	0	(2.5; 2)	1	da
(4; 2)	0	(3; 2)	0	nu

Deci în total avem 6 erori (din totalul de 10 instanțe), ceea ce indică faptul ca algoritmul 1-NN este foarte puțin adecvat pentru acest gen de date.

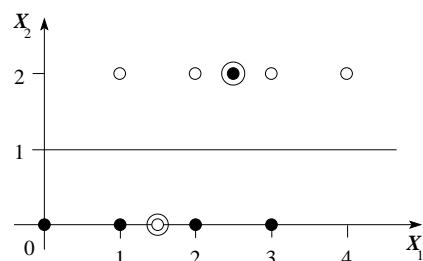
c. Eroarea la antrenare cu algoritmul ID3 pe acest set de date este 0, fiindcă instanțele sunt etichetate în mod consistent.

Eroarea la CVLOO produsă de algoritmul ID3 pe același set de date se poate calcula foarte ușor, ținând cont că este suficient să determinăm granițele de decizie — deci și zonele de decizie — corespunzătoare celor două clase. De exemplu, atunci când se va elimina instanța (1,0), se vor obține granițele de decizie ca în figura alăturată. Este imediat că *modelul* rezultat în acest caz va clasifica instanța (1,0) în mod eronat.



Se constată că eroarea de tip CVLOO produsă de ID3 cu atribute continue este de 6 (din totalul de 10) instanțe. Ba chiar, — este încă o coincidență! — ID3 produce la CVLOO exact aceleași erori (mai precis, el clasifică în mod eronat exact aceleași instanțe) ca și algoritmul 1-NN.

d. Se observă ușor că separatorul liniar care minimizează eroarea la antrenare este cel reprezentat în figura alăturată. Datele clasificate eronat de către acest separator sunt cele două puncte încercuite din figură; ele sunt de asemenea singurele puncte care generează eroare și la testare cu metoda CVLOO.



Mai concret, în cazul CVLOO folosind SVM, deoarece separatorul optimal este „susținut” de mai mulți vectori-suport pe fiecare parte, lipsa unui singur dintre ei nu influențează cu nimic construirea separatorului. În fiecare caz în parte se va învăța ca separator dreapta paralelă cu Ox1 care trece prin punctul (0, 1).

Avem deci în ambele situații, adică atât la antrenare cât și la cross-validation cu metoda “Leave-One-Out”, (doar) două puncte clasificate eronat de către SVM.

Comparând cei trei clasificatori, 1-NN, ID3 cu atribute continue și SVM, rezultă în mod clar că SVM este cel mai convenabil pe acest set de date (deși la antrenare SVM produce două erori, iar 1-NN și ID3 nicio eroare), fiindcă nu produce *overfitting* (aici!).

13. (Comparații între algoritmii 1-NN și ID3: Da sau nu?)

CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 2.1

a. Este posibil să se construiască un arbore de decizie (având în fiecare nod intern teste de forma $x > a$, $x \leq b$, $y > c$, sau $y \leq d$, unde a, b, c, d sunt numere reale oarecare) care să producă la clasificare aceleași rezultate ca și algoritmul 1-NN folosind distanța euclidiană? Justificați răspunsul.

(Învățare rapidă / “eager” vs. învățare lentă / “lazy”; k -NN vs. ID3)

CMU, 2010 spring, HW1, pr. 3.3

b. Algoritmul ID3 este o metodă de învățare de tip “batch”, care solicită ca toate datele de antrenament să-i fie puse la dispoziție pentru a putea elabora arborele de decizie. Așadar, în situația în care date de antrenament suplimentare ne sunt puse ulterior la dispoziție, acestea trebuie tratate cu atenție fiindcă ele pot modifica arborele de decizie rezultat în urma învățării.¹⁷² Algoritmul k -NN suferă și el de această problemă? Justificați.

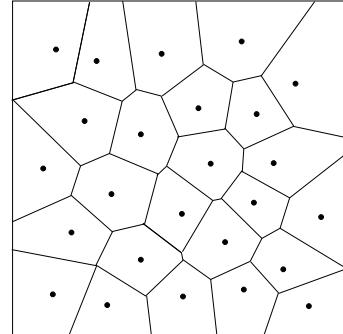
Răspuns:

a. Nu.

Suprafețele de decizie pentru algoritmul 1-NN corespund diagramei Voronoi și nu sunt neapărat paralele cu axele de coordonate, după cum se observă în figura alăturată.

Suprafețele de decizie pentru un arbore de decizie cu atribute cu valori continue sunt întotdeauna paralele cu axele de coordonate, deoarece deciziile din fiecare nod sunt de forma $x > a$, $x \leq b$, $y > c$, sau $y \leq d$, $\forall a, b, c, d \in \mathbb{R}$.

Așadar, răspunsul este negativ pentru cazul general, deși există situații în care cei doi algoritmi produc exact aceeași clasificare.



b. Nu.

Răspunsul decurge din modul de lucru a algoritmului k -NN, care este un algoritm de învățare de tip “lazy”: k -NN estimează valoarea locală a unei funcții-target \hat{f} pentru una sau mai multe instanțe de test x_q . Valoarea $\hat{f}(x_q)$ nu depinde decât de cei mai apropiati vecini ai lui x_q ; celealte instanțe de antrenament nu sunt necesare pentru calculul lui $\hat{f}(x_q)$. Evident, dacă pe măsură ce se acumulează noi date de antrenament se modifică și vecinătatea lui x_q , atunci se prea poate să se modifice și $\hat{f}(x_q)$. Însă în sine, algoritmul procedează exact la fel ca mai înainte. Se poate modifica doar outputul lui. Spre deosebire de algoritmul k -NN, la învățarea arborilor de decizie adăugarea de noi instanțe modifică în general și modelul / arborele rezultat, nu doar decizia pentru o instanță de test particulară.

¹⁷²Vedeți problema 16 de la capitolul *Arborei de decizie*.

14. (1-NN cu mapare cu RBF: Adevărat sau Fals?)
 ■ CMU, 2003 fall, T. Mitchell, A. Moore, final exam, pr. 7.f

Algoritmul 1-NN folosind distanța euclidiană neponderată este capabil să obțină rezultate mai bune dacă în prealabil intrările sale sunt mapate într-un „spațiu de trăsături“ folosind o funcție-nucleu cu baza radială (RBF).

Răspuns:

Fals.

Fie $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^n$ funcția de mapare în spațiul de trăsături, astfel încât să avem $K(x, y) \stackrel{\text{not.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}} = \phi(x) \cdot \phi(y)$, $\forall x, y \in \mathbb{R}^d$. (\mathbb{R}^d reprezintă spațiul inițial, \mathbb{R}^n spațiul de trăsături în care se face maparea, iar $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$ este funcția-nucleu cu baza radială.) Avem:

$$\begin{aligned} \|\phi(x) - \phi(y)\|^2 &= (\phi(x) - \phi(y)) \cdot (\phi(x) - \phi(y)) \\ &= \phi(x) \cdot \phi(x) + \phi(y) \cdot \phi(y) - 2 \cdot \phi(x) \cdot \phi(y) = e^{-\frac{\|x-x\|^2}{2\sigma^2}} + e^{-\frac{\|y-y\|^2}{2\sigma^2}} - 2 \cdot e^{-\frac{\|x-y\|^2}{2\sigma^2}} \\ &= e^0 + e^0 - 2 \cdot e^{-\frac{\|x-y\|^2}{2\sigma^2}} = 2 - 2 \cdot e^{-\frac{\|x-y\|^2}{2\sigma^2}} = 2 - K(x, y). \end{aligned}$$

Prin urmare, pentru orice $x, x_i, x_j \in \mathbb{R}^d$ vom avea:

$$\begin{aligned} \|\phi(x) - \phi(x_i)\|^2 \leq \|\phi(x) - \phi(x_j)\|^2 &\Leftrightarrow 2 - K(x, x_i) \leq 2 - K(x, x_j) \Leftrightarrow K(x, x_i) \geq K(x, x_j) \\ &\Leftrightarrow e^{-\frac{\|x-x_i\|^2}{2\sigma^2}} \geq e^{-\frac{\|x-x_j\|^2}{2\sigma^2}} \Leftrightarrow -\frac{\|x-x_i\|^2}{2\sigma^2} \geq -\frac{\|x-x_j\|^2}{2\sigma^2} \Leftrightarrow \|x-x_i\|^2 \leq \|x-x_j\|^2. \end{aligned}$$

Cu alte cuvinte, dacă o instanță de test x are drept cel mai apropiat vecin punctul x_i în spațiul inițial, acest lucru rămâne valabil și în spațiul de trăsături. Așadar, decizia algoritmului 1-NN este identică în ambele spații. Aceeași concluzie este valabilă și pentru k -NN.

Observație: Este însă posibil ca folosind ponderarea sau alte măsuri de distanță (decât cea euclidiană) kernel-izarea să funcționeze cu succes pentru k -NN.

3.2 Probleme propuse

15. (Algoritmul k -NN: acuratețe; comparație cu un simplu clasificator aleator)
prelucrare de Liviu Ciortuz, după CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, HW1, pr. 5.bc

- a. Enunțați [succint] *regula de decizie* a algoritmului k -NN pentru o instanță de test x_q . Care este *bias*-ul inductiv al algoritmului k -NN?

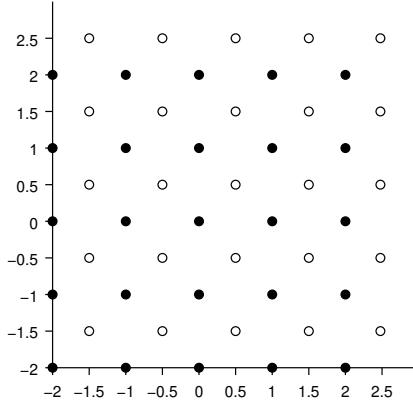
În continuare veți lucra pe datele din figura de mai jos. Veți aplica algoritmul k -NN, folosind distanța euclidiană.

k -NN funcționează bine atunci când instanțele dintr-o aceeași clasă sunt plasate într-o zonă relativ bine delimitată, fără întrepătrunderi puternice.

Obiectivul nostru acum este să analizăm ce se întâmplă atunci când datele sunt puternic mixate. Rezultatele pe care le veți obține la calculul erorilor vor fi exprimate sub formă de numere [fracționare] din intervalul $[0, 1]$.

Observație importantă:

În cazul în care există două sau mai multe instanțe situate exact pe „marginea“ [adică, pe conturul circular al] k -NN-vecinătății asociate instanței de clasificat, se va considera că toate aceste instanțe aparțin respectivei vecinătăți, iar fiecare dintre ele dispune de un vot întreg.



- b. Pentru $k = 1$, calculați eroarea la antrenare și eroarea la cross-validation cu metoda “leave-one-out” (CVLOO).

Ce puteți spune comparând cele două rezultate? (Care este legătura între *bias*-ul inductiv al lui k -NN și puterea de generalizare a lui 1-NN pe astfel de date? Se produce oare aici un *anumit* fenomen, specific multor situații din clasificarea automată?)

- c. Pentru $k = 2$, calculați eroarea la cross-validation cu metoda “leave-one-out” (CVLOO).
- d. Considerăm $k = 50$. (Remarcați faptul că în total în setul nostru de date sunt 50 de instanțe.) De această dată, vom impune ca algoritmul k -NN să ia decizia în mod *probabilist*. Aceasta înseamnă că dacă în vecinătatea k -NN a unei instanțe de test există n vecini pozitivi și m vecini negativi, atunci algoritmul k -NN va returna (pentru instanța respectivă) decizia + cu probabilitatea $n/(n+m)$ și decizia – cu probabilitatea $m/(n+m)$. În consecință, pentru întreg setul de date vom putea calcula o *eroare medie*.

Calculați *eroarea medie* la antrenare pentru algoritmul 50-NN pe datele de mai sus. Cu noașteți o metodă de clasificare foarte simplă care obține pe aceste date rezultate la fel de bune / proaste precum 50-NN?

16. (Algoritmul 1-NN: calculul erorii la CVLOO)

CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm exam, pr. 1.7

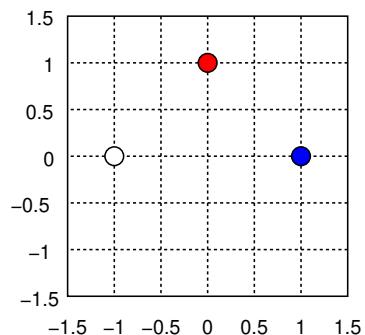
Care este eroarea clasificatorului 1-NN la cross-validation de tip “Leave-One-Out” pe setul de date următor?



17.

(Algoritmul 1-NN: granițe / suprafețe de decizie)
CMU, 2013 fall, W. Cohen, E. Xing, final exam, pr. 3.6

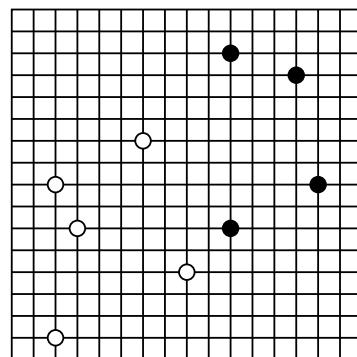
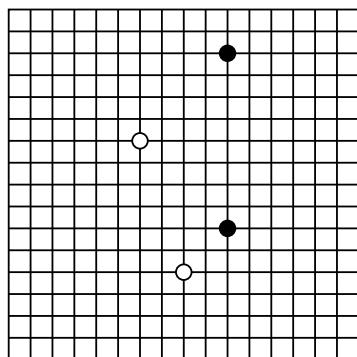
Trasați granițele de decizie (engl., *decision boundaries*) produse de clasificatorul 1-NN la aplicarea pe datele din figura alăturată. Diversele forme ale punctelor reprezintă clase diferite.



18.

(Algoritmul 1-NN: granițe / suprafețe de decizie)
CMU, 2008 fall, Eric Xing, HW1, pr. 3

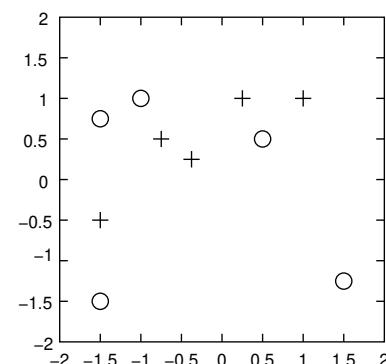
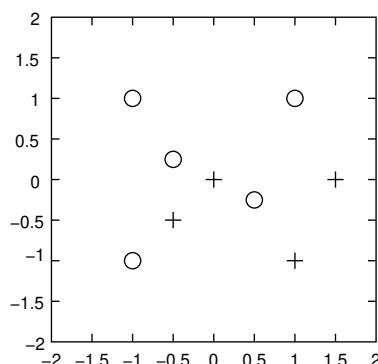
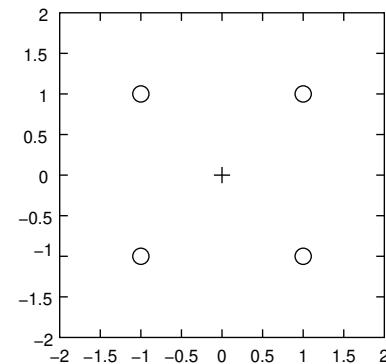
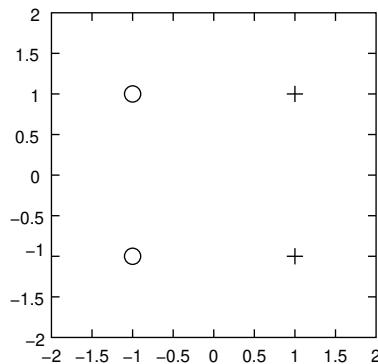
În fiecare din figurile următoare se dau câteva puncte în spațiul bidimensional, etichetate cu + sau -. Indicați în fiecare caz granițele / suprafețele de decizie pentru algoritmul 1-NN presupunând că se folosește distanța euclidiană.



19.

(Algoritmul 1-NN: diagrame Voronoi)
CMU, 2010 fall, Ziv Bar-Joseph, HW1, pr. 3.1

Desenați suprafețele de decizie pentru clasificatorul 1-NN pentru fiecare dintre seturile de date din figurile de mai jos. Folosiți distanța euclidiană. Hașurați fin zonele corespunzătoare clasei +.



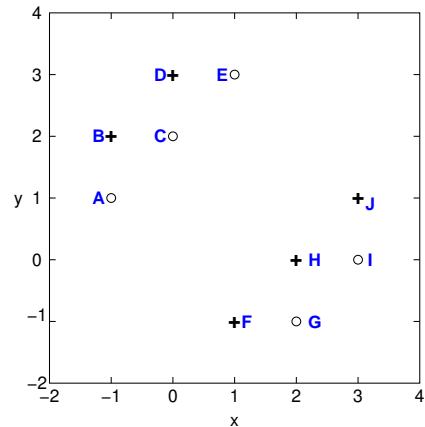
20.

(Algoritmul k -NN: diagrama Voronoi, eroarea la CVLOO; comparație pentru diferite valori ale lui k)

*prelucrare de L. Ciortuz, după
CMU, 2012 spring, Ziv Bar-Joseph, midterm exam, pr. 2*

La acest exercițiu veți aplica algoritmul k -NN folosind distanța euclidiană pe setul de date din figura de mai jos. Fiecare punct aparține la una din două clase, desemnate cu + și respectiv ○.

- a. Trasați diagrama Voronoi și hașurați zona / zonele de decizie corespunzătoare etichetei +.
- b. Care este eroarea la cross-validation cu metoda "Leave-One-Out" (CVLOO) dacă se folosește algoritmul 1-NN?
- c. Care dintre următoarele valori ale lui k va conduce la o valoare minimă a erorii de tip CVLOO: 3, 5, 7 sau 9? Comentați succint rezultatul.



Indicații:

1. k -NN-vecinătățile vor fi construite în manieră *inclusivă*.¹⁷³ Vă cerem(!) să puneti în evidență toate cazurile de acest tip. Vedeți, spre exemplu, liniile 2 și 3 din tabelul de mai jos, coloana 1-NN vecinătăților.
2. În caz de *paritate la voturi* (dar doar în acest caz!), se va considera că se aplică (în mod intuitiv) ponderarea distanțelor în sensul prezentat la curs.
3. Dacă veți ști să exploatați *simetriile*, veți avea mult mai puțin de elaborat la nivel de detaliu!

21.

(Algoritmul k -NN: alegerea valorii convenabile pentru k)

prelucrare de Liviu Ciortuz, după CMU, (?) spring, ML course 10-701, HW1, pr. 5

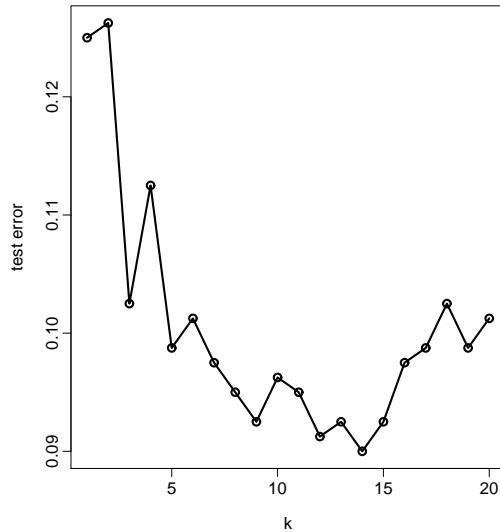
Pe un anumit set de date format din date de antrenament, date de validare și date de test, după ce fost antrenat algoritmul k -NN pentru diferite valori ale lui k , rezultatele obținute la validare au fost reprezentate în graficul care urmează.

¹⁷³Adică, dacă notăm cu

- x_1, x_2, \dots, x_n instanțele de antrenament,
- x o instanță oarecare căreia i se aplică la un moment dat procedura de cross-validation LOO cu algoritmul k -NN (unde k este fixat),
- $d(x, x_{i_1}) \leq d(x, x_{i_2}) \leq \dots \leq d(x, x_{i_n})$ secvența ordonată a distanțelor de la x la fiecare din instanțele de antrenament,

și există $l > 0$ astfel încât $x_{i_k} = x_{i_{k+1}} = \dots = x_{i_{k+l}} < x_{i_{k+l+1}}$ sau $k+l = n$, atunci în k -NN vecinătatea lui x vor fi incluse toate instanțele $x_{i_{k+1}}, \dots, x_{i_{k+l}}$.

Care este — în conformitate cu aceste rezultate — valoarea optimală care trebuie aleasă pentru k , în vederea folosirii ulterioare pe datele de test? Justificați alegerea făcută.



22.

(Algoritmul k -NN: acuratețe; comparații pentru diferite valori ale lui k)

CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, HW1, pr. 5.a

Considerăm două clase, notate cu C_1 și C_2 , în spațiul euclidian bidimensional. Datele din clasa C_1 sunt distribuite în mod uniform într-un cerc de rază r . Datele din clasa C_2 sunt distribuite în mod uniform într-un alt cerc de rază r . (*Observație:* Numărul de date din cele două clase nu este neapărat același.) Centrele celor două cercuri sunt situate la o distanță strict mai mare decât $4r$.

Arătați că este posibil ca [la antrenare] acuratețea algoritmului 1-NN aplicat pe aceste date să fie *strict* mai mare decât acuratețea algoritmului k -NN, pentru un anumit număr întreg $k \geq 3$, impar, ales în mod convenabil.

23.

(Algoritmul k -NN: întrebări de ordin calitativ)

CMU, 2012 spring, Ziv Bar-Joseph, HW1, pr. 3

Stim că la aplicarea algoritmului k -NN, clasificarea unei instanțe date se face pe baza votului majoritar obținut în „vecinătatea“ instanței respective. Presupunem că se dă două clase de instanțe, fiecare clasă având $n/2$ puncte, întrepătrunse într-o anumită măsură, într-un spațiu bidimensional.

- a. Descrieți ce se întâmplă cu *eroarea la antrenare* (folosind toate datele disponibile) când numărul k al vecinilor considerați variază de la n la 1.
- b. Schițați grafic cum anume ar evoluă *eroarea de generalizare* (de exemplu, reținând o parte din date pentru testare) atunci când k variază. Explicați modul în care ați raționat.
- c. Propuneți o metodă de determinare a unei valori adecvate pentru k .
- d. La folosirea algoritmului k -NN, odată ce s-a stabilit valoarea lui k , toți cei mai apropiati k vecini ai punctului de clasificat au ponderi egale (adică, aceeași importanță) la stabilirea

etichetei respectivului punct. Sugerați o modificare a algoritmului k -NN care elimină această limitare.

e. Dați două motive pentru care este de preferat să nu folosim algoritmul k -NN atunci când dimensiunea spațiului datelor de intrare este mare.

24.

(Algoritmul k -NN:CVLOO: comparație pentru diferite valori ale lui k ;

eroarea la antrenare: comparație cu alți clasificatori)

CMU, 2010 fall, Aarti Singh, midterm exam, pr. 2

a. Care dintre clasificatorii de mai jos realizează
o eroare de tip CVLOO (Leave-One-Out Cross-validation) mai mare pe setul de date alăturat?

<input type="checkbox"/> 1-NN	<input type="checkbox"/> 3-NN	+	+	-	-
-------------------------------	-------------------------------	---	---	---	---

Recomandare: Formulați explicit euristică pe care o veți folosi pentru a trata cazurile în care apar tot atâtea voturi pozitive cât cele negative.

b. Considerăm setul de date din figura alăturată. Care dintre clasificatorii de mai jos obține / obțin eroare nulă la antrenare pe acest set de date?

<input type="radio"/>	+
+	<input type="radio"/>

- arborii de decizie ID3 de adâncime 2
- clasificatorul 3-NN
- regresia logistică
- SVM (cu nucleu pătratic)

25.

(Compararea clasificatorilor 1-NN și Bayes Optimal:

o margine superioară mai bună
pentru *rata medie a erorii asymptotice* a lui 1-NN)

Liviu Ciortuz, 2014, bazat pe un rezultat din
■ “An Elementary Introduction to Statistical Learning Theory”,
S. Kulkarni, G. Harman, 2011, pag. 68-69

La problema 10 am demonstrat că *rata medie a erorii* clasificatorului 1-NN este mărginită asimptotic¹⁷⁴ de dublul ratei medii a erorii clasificatorului Bayes Optimal.

Arătați că — în aceleși condiții ca la problema 10 — se poate obține o margine chiar mai bună:

$$E[\lim_{n \rightarrow \infty} Error_{1-NN}] \leq 2E[Error_{Bayes}](1 - E[Error_{Bayes}]).$$

26.

(Adevărat sau Fals?)

CMU, 2010 fall, Ziv Bar-Joseph, midterm, pr. 1.bc

Care dintre următoarele afirmații sunt adevărate pentru clasificatorii k -NN? (Justificați pe scurt răspunsul, în dreptul fiecărui punct.)

¹⁷⁴Adică, anuncii când $n \rightarrow \infty$, unde n este numărul de instanțe de antrenament.

- a. Acuratețea la antrenare crește pe măsură ce crește valoarea lui k .
- b. Suprafața de decizie este mai netedă (engl., smoother) pe măsură ce valoarea lui k scade.
- c. k -NN nu necesită o procedură explicită de antrenare.
- d. Suprafața / granița de decizie este liniară.
- e. Este posibil ca un clasificator binar 1-NN să clasifice întotdeauna orice instanță de test ca fiind pozitivă, chiar dacă în setul de date de antrenament există instanțe negative.

27.

(Întrebări calitative despre design-ul unor experimente din Învățarea Automată: OK ori ...problematic?)
CMU, 2009, Geoff Gordon, midterm exam, pr. 3

Fiecare din punctele de mai jos prezintă pe scurt design-ul unui experiment practic de învățare automată. Analizați fiecare din aceste cazuri, indicând apoi dacă respectivul experiment este *ok* ori *problematic* (încercuți varianta pe care o alegeti). Dacă este *problematic*, identificați TOATE defectele [de concepție ale] design-ului respectiv.

- a. O echipă de proiectare raportează o *eroare mică la antrenare* și susține că metoda folosită este bună.

Ok

Problematic

- b. O echipă de proiectare susține că este un mare succes faptul că a obținut 98% *acuratețe la antrenare* pentru un task de clasificare binară care are următorul specific: unul din cele două cazuri se întâlnește foarte rar comparativ cu celălalt caz. (O astfel de problemă o constituie, de exemplu, identificarea tranzacțiile bancare frauduloase.) Datele lor au constat din 50 de exemple pozitive și 4950 de exemple negative.

Ok

Problematic

- c. O echipă de proiectare și-a împărțit datele de care dispune în date de antrenament și date de test. Folosind datele de antrenament, ei au construit un *model* de clasificare caracterizat de anumiți *parametri*. Apoi, făcând *cross-validation*, au ales cea mai bună setare a parametrilor. La final, au raportat *eroarea* obținută pe *datele de test*.

Ok

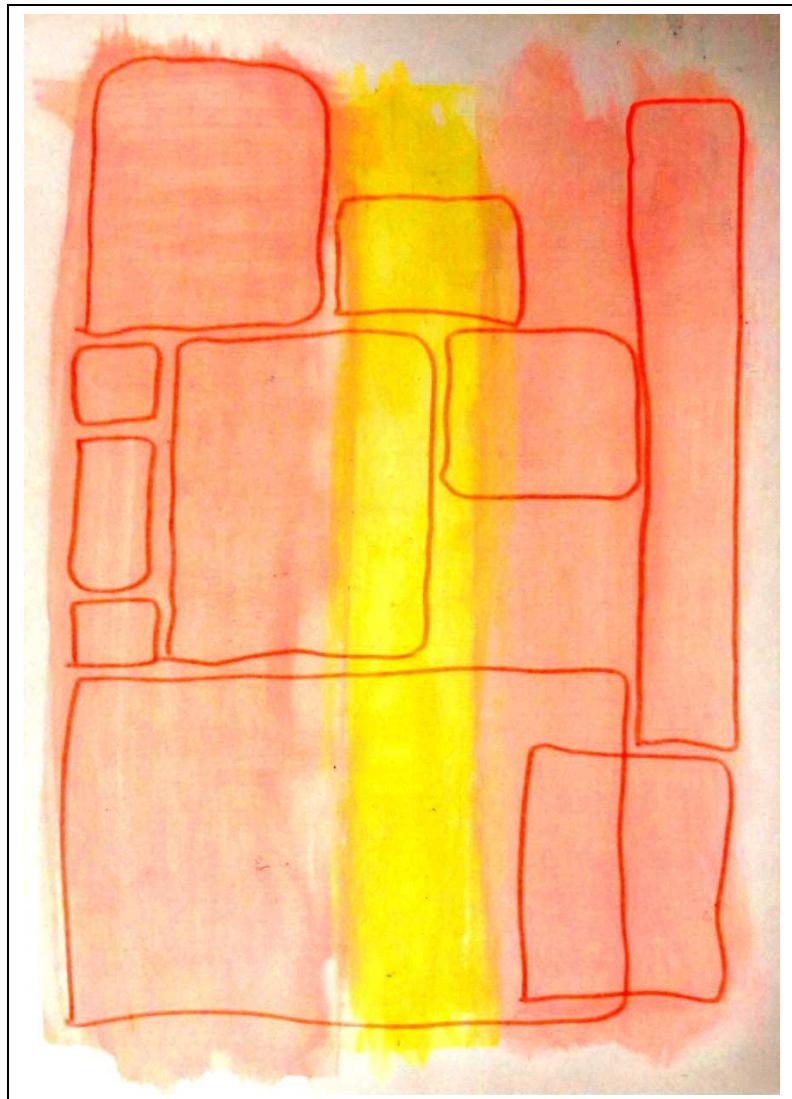
Problematic

- d. O echipă de proiectare a efectuat o procedură de *selecție a atributelor* (engl., features) pe toate datele și apoi a redus setul mare de atrbute la un set mai mic. După aceea, membrii echipei au împărțit datele în date de test și date de antrenament. Au construit *modelul* de clasificare pe datele de antrenament folosind mai multe setări ale parametrilor modelului, și au raportat cea mai bună *eroare la testare* pe care au obținut-o.

Ok

Problematic

Această pagină a fost lăsată liberă în mod intenționat.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

4 Arbori de decizie

Sumar

Noțiuni preliminare

- partiție a unei mulțimi: ex. 82 de la cap. *Fundamente*;
- proprietăți elementare ale funcției logaritm; formule uzuale pentru calcule cu logaritmi;
- Elemente de *teoria informației* (vedeți secțiunea corespunzătoare din cap. *Fundamente*):
 - entropie, definiție: T. Mitchell, *Machine Learning*, 1997 (desemnată în continuare simplu prin *cartea ML*), pag. 57; ex. 2.a, ex. 39.a, ex. 35.a;
 - entropie condițională specifică: ex. 14.a;
 - entropie condițională medie: ex. 2.cd, ex. 35.c;
 - câștig de informație (definiție: cartea ML, pag. 58): ex. 2.cd, ex. 5.a, ex. 33, ex. 39.b, ex. 35.e;
 - *arbori de decizie*, văzuți ca structură de date: ex. 1, ex. 31 și, respectiv, ca program în logica propozițiilor: ex. 2.e, ex. 38.bc;
 - (P0) expresivitatea arborilor de decizie cu privire la *funcții boolene*: ex. 32;
 - *spațiu de versiuni* pentru un concept (de învățat): ex. 1, ex. 31, ex. 37.

Algoritmul ID3

- pseudo-cod: cartea ML, pag. 56;
- *bias-ul inductiv*: *ibidem*, pag. 63-64;
- exemple simple de aplicare: ex. 2, ex. 3, ex. 4.a, ex. 5, ex. 36, ex. 37.a, ex. 38, ex. 39, ex. 40;
- ID3 ca algoritm *per se*:
 - este un *algoritm de căutare*;
spațiul de căutare — mulțimea tuturor arborilor de decizie care se pot construi cu atributele de intrare în *nodurile de test* și cu valorile atributului de ieșire în *nodurile de decizie* — este de dimensiune exponențială în raport cu numărul de atrbute: ex. 1, ex. 3, ex. 31, ex. 37;
 - ID3 are ca *obiectiv* căutarea unui arbore / *model* care *i.* să explice cât mai bine datele (în particular, atunci când datele sunt *consistent*, modelul trebuie să fie *consistent* cu acestea), *ii.* să fie cât mai *compact*, din motive de *eficiență* la generalizare / testare și *iii.* în final să aibă o [cât mai] bună putere de *generalizare*;¹⁷⁵
 - ID3 ar putea fi văzut și ca algoritm de *optimizare*;¹⁷⁶

¹⁷⁵LC: Alternativ, putem spune că algoritmul ID3 produce o *structură* de tip *ierarhie* (arbore) între diferite *partiționări* ale setului de instanțe de antrenament, această ierarhie fiind generată pe baza *corespondenței* dintre atributul de *ieșire* și atributele de *intrare*, care sunt adăugate la model câte unul pe rând.

¹⁷⁶LC: Am putea să-l interpretăm pe ID3 ca fiind un algoritm care caută între diferitele *distribuții probabiliste* discrete care pot fi definite pe setul de date de antrenament una care să satisfacă cerința de *ierarhizare*, și pentru care entropia să fie *minimală* (vedeți proprietatea de *structură-litate* de la ex. 50 de la cap. *Fundamente*). Cerința ca arboarele ID3 să fie *minimal* (ca număr de niveluri / noduri) este însă mai importantă, mai practică și mai ușor de înțeles.

- *greedy* \Rightarrow nu garantează obținerea soluției optime d.p.v. al numărului de niveluri / noduri:
ex. 4, ex. 21.a, ex. 38 (vs. ex. 37.b, ex. 3.b), ex. 46;
- de tip *divide-et-impera* (\Rightarrow “*Iterative Dichotomizer*”), recursiv;
 - *1-step look-ahead*;
- complexitate de timp, cf. Weka book.¹⁷⁷
la antrenare, în anumite condiții: $\mathcal{O}(dm \log m)$; la testare $\mathcal{O}(d)$, unde d este numărul de attribute, iar m este numărul de exemple;
- ID3 ca algoritm de învățare automată:
 - *bias-ul inductiv* al algoritmului ID3:
[dorim ca modelul să aibă structură ierarhică, să fie compatibil / consistent cu datele dacă acestea sunt consistente (adică, necontradictorii), iar] arborele produs de ID3 trebuie să aibă un număr cât mai mic de niveluri / noduri;
 - algoritm de învățare de tip “*eager*”;
 - *analiza erorilor*:
la antrenare: ex. 7.a, ex. 10.a, ex. 42;¹⁷⁸ [acuratețe la antrenare: ex. 6;]
la validare: CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 1;
la n -fold cross-validation
la cross-validation leave-one-out (CVLOO): ex. 10.b, ex. 45.bc;¹⁷⁹
 - *robustețe la „zgomote“ și overfitting*: ex. 10, ex. 21.bc, ex. 45, ex. 67.b;¹⁸⁰
 - *zone de decizie și granițe de separare / decizie* pentru arbori de decizie cu variabile continue: ex. 10, ex. 44, ex. 45, ex. 46.

Observație: Zonele de decizie produse de algoritmul ID3 nu sunt în mod neapărat unice, fiindcă arborele de decizie creat de ID3 nu este determinat în mod unic (vedeți proprietatea (P2), mai jos).

Extensii / variante ale algoritmului ID3

- *attribute cu valori continue*: ex. 10-12, ex. 14.c, ex. 43-47; cap. *Învățare bazată pe memorare*, ex. 11.b;
- attribute discrete cu multe valori: ex. 13;
- attribute cu valori nespecificate / absente pentru unele instanțe;
- attribute cu diferite costuri asociate: ex. 14;
- reducerea caracterului “*eager*” al învățării: ex. 16;
- reducerea caracterului “*greedy*” al învățării:
IG cu “*2-step look-ahead*”: ex. 17, ex. 18;
variante de tip “*look-ahead*” specifică atributelor continue: ex. 48;
- folosirea altor *măsură de „impuritate“* în locul câștigului de informație:
Gini Impurity, Misclassification Impurity: ex. 15;
- reducerea *overfitting-ului*:
reduced-error pruning (folosind un set de date de validare): carte ML, pag. 69-71; A. Cornuéjols, L. Miclet, *Apprentissage artificiel*, 2010, pag. 418-421;

¹⁷⁷“Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, Ian Witten, Eibe Frank (3rd ed.), Morgan Kaufmann Publishers, 2011.

¹⁷⁸De asemenea, ex. 4.ab de la capitolul *Clasificare bayesiană*.

¹⁷⁹De asemenea, ex. 4.ab de la capitolul *Clasificare bayesiană*.

¹⁸⁰De asemenea, ex. 4.ab de la capitolul *Clasificare bayesiană*.

rule post-pruning: cartea ML, pag. 71-72; ex. 50
 top-down vs. bottom-up pruning, folosind un criteriu bazat pe câștigul de informație: ex. 19, ex. 49;
 pruning folosind testul statistic χ^2 : ex. 20, ex. 51.

Proprietăți numerice / calitative ale arborilor ID3

- (P1) arborele produs de algoritm ID3 este *consistent* (adică, în concordanță) cu datele de antrenament, dacă acestea sunt *consistente* (adică, necontradictorii). Altfel spus, *eroarea la antrenare* produsă de algoritm ID3 pe orice set de *date consistente* este 0: ex. 2-4, ex. 37-38;
- (P2) arborele produs de algoritm ID3 nu este în mod neapărat *unic*: ex. 3, ex. 37;
- (P3) arborele ID3 nu este neapărat *optimal* (ca nr. de noduri / niveluri): ex. 4, ex. 21, ex. 38;
- (P4) influența *atributelor identice* și, respectiv, a *instanțelor multiple* asupra arborelui ID3: ex. 8;
- (P5) o *margine superioară* pentru *eroarea la antrenare* a algoritmului ID3, în funcție de numărul de valori ale variabilei de ieșire): ex. 7.b;
- (P6) o aproximare simplă a numărului de *instanțe greșit clasificate* din totalul de M instanțe care au fost asignate la un nod frunză al unui arbore ID3, cu ajutorul entropiei (H) nodului respectiv: ex. 41;
- (P7) *granițele de separare / decizie* pentru arborii ID3 cu *attribute de intrare continue* sunt întotdeauna paralele cu axele de coordonate: ex. 10, ex. 12, ex. 44, ex. 45, ex. 46 și cap. *Învățare bazată pe memorare*, ex. 11.b;
Observație: Următoarele trei proprietăți se referă la arbori de decizie în general, nu doar la arbori ID3.
- (P8) *adâncimea maximă* a unui arbore de decizie, când atributele de intrare sunt categoriale: numărul de atrbute: ex. 52.c;
- (P9) o *margine superioară* pentru *adâncimea* unui arbore de decizie când atrbutele de intrare sunt continue, iar datele de antrenament sunt (ne)separabile liniar: ex. 11;
- (P10) o *margine superioară* pentru numărul de *noduri-frunză* dintr-un arbore de decizie, în funcție de numărul de exemple și de numărul de atrbute de intrare, atunci când acestea (atributete de intrare) sunt binare: ex. 9.

Învățare automată de tip ansamblist folosind arbori de decizie: Algoritmul AdaBoost

- Noțiuni preliminare:
 distribuție de probabilitate discretă, factor de normalizare pentru o distribuție de probabilitate, ipoteze „slabe“ (engl., weak hypothesis), compas de decizie (engl., decision stump), prag de separare (engl., threshold split) pentru un compas de decizie, prag exterior de separare (engl., outside threshold split), eroare ponderată la antrenare (engl., weighted training error), vot majoritar ponderat (engl., weighted majority vote), overfitting, ansambluri de clasificatori (vedeți ex. 64), funcții de cost / pierdere (engl., loss function) (vedeți ex. 29 și ex. 23);
- pseudo-codul algoritmului AdaBoost + proprietăți de bază + convergența erorii la antrenare: ex. 22 și 23;
- exemple de aplicare: ex. 24, 56, 54, 55, 57, 58.

- AdaBoost ca algoritm *per se*:
algoritm iterativ, algoritm de căutare (spațiul de căutare este mulțimea combinațiilor liniare care se pot construi peste clasa de ipoteze „slabe“ considerate), algoritm de optimizare secvențială (minimizează o margine superioară pentru eroarea la antrenare), algoritm greedy (dacă la fiecare iterație se alege cea mai bună ipoteză „slabă“).
- învățabilitate empirică γ -slabă:
definiție: ex. 23.e
exemplificarea unor cazuri când nu există *garanție* pentru învățabilitate γ -slabă: ex. 25, 60;
- AdaBoost ca algoritm de *optimizare secvențială* în raport cu funcția de cost / „pierdere“ negativ-exponențială: ex. 26;
- marginea de votare: ex. 27, 28 și 65;
- *selectarea trăsăturilor* folosind AdaBoost; aplicare la clasificarea de documente: ex. 62;
- o variantă generalizată a algoritmului AdaBoost: ex. 29 și ex. 65;
- recapitulare (întrebări cu răspuns *adevărat / fals*): ex. 30 și 66.
- Proprietăți ale algoritmului AdaBoost:
 - (P0) AdaBoost poate produce rezultate diferite atunci când are posibilitatea să aleagă între două sau mai multe [cele mai bune] ipoteze „slabe“: ex. 24, 54;
 - (P1) $\text{err}_{D_{t+1}}(h_t) = \frac{1}{2}$ (ex. 22.vii);
ca o consecință, rezultă că ipoteza h_t nu poate fi reselecată și la iterarea $t + 1$; ea poate fi reselecată la o iterare ulterioară;
 - (P2) Din relația de definiție pentru distribuția D_{t+1} rezultă
 $Z_t = e^{-\alpha_t} \cdot (1 - \varepsilon_t) + e^{\alpha_t} \cdot \varepsilon_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$ și
 $\varepsilon_t \in (0, 1/2) \Rightarrow Z_t \in (0, 1)$ (ex. 22).
 - (P3) $D_{t+1}(i) = \frac{1}{m \prod_{t'=1}^t Z_{t'}} e^{-y_i f_t(x_i)}$, unde $f_t(x_i) \stackrel{\text{def.}}{=} -\sum_{t'=1}^t \alpha_{t'} h_{t'}(x_i)$ (ex. 23.a). Produsul $y_i f_t(x_i)$ se numește *margine algebrică*;
 - (P4) $\text{err}_S(H_t) \leq \prod_{t'=1}^t Z_t$, adică eroarea la antrenare comisă de ipoteza combinată produsă de AdaBoost este majorată de produsul factorilor de normalizare (ex. 23.b);
 - (P5) AdaBoost nu optimizează în mod direct $\text{err}_S(H_t)$, ci marginea sa superioară, $\prod_{t'=1}^t Z_t$; optimizarea se face în mod secvențial (greedy): la iterarea t se minimizează valoarea lui Z_t ca funcție de α_t , ceea ce conduce la $\alpha_t = \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}$ (ex. 23.c);
 - (P5') $\varepsilon_i > \varepsilon_j \Leftrightarrow \alpha_i < \alpha_j$ (consecință imediată din relația $\alpha_t = \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}$): ex. 22.v;
 - (P6) O consecință din relația (81) și (P5): $D_{t+1}(i) = \begin{cases} \frac{1}{2\varepsilon_t} D_t(i), & i \in M \\ \frac{1}{2(1 - \varepsilon_t)} D_t(i), & i \in C \end{cases}$
(ex. 22.iv);
 - (P7) $\text{err}_S(H_t)$ nu neapărat descrește de la o iterare la alta; în schimb, descresc marginile sale superioare: $\prod_{t'=1}^t Z_{t'}$ și $\exp(-\sum_{t'=1}^t \gamma_{t'}^2)$ (ex. 23.d);
 - (P8) O condiție suficientă pentru învățabilitate γ -slabă, bazată pe marginea de votare: marginea de votare a oricărei instanțe de antrenament să fie de cel puțin 2γ , la orice iterare a algoritmului AdaBoost (ex. 28);

(P9) Orice mulțime formată din m instanțe din \mathbb{R} care sunt etichetate în mod consistent poate fi corect clasificată de către o combinație liniară formată din cel mult $2m$ compași de decizie (ex. 64.a);

(P10) Orice mulțime de instanțe distincte [și etichetate] din \mathbb{R} este γ -slab învățabilă cu ajutorul compașilor de decizie (ex. 63).

- Alte metode de învățare ansamblistă bazate pe arbori de decizie: Bagging, Random Forests;
- Alte metode de învățare automată bazate pe arbori: arbori de regresie (CART).

4.1 Probleme rezolvate

4.1.1 Algoritmul ID3

1. (Arbori de decizie; optimalitate, relativ la numărul de noduri)

Reprezentați arborele / arborii de decizie care are / au numărul minim de noduri posibile și corespunde / corespund funcției booleene $(\neg A \vee B) \wedge \neg(C \wedge A)$ definită peste atributele booleene A, B și C .

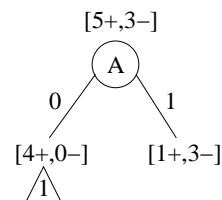
Răspuns:

Vom determina arborele de decizie optimal (ca număr de noduri) parcurgând în mod *exhaustiv spațiul de versiuni*, adică mulțimea tuturor arborilor de decizie (construiți cu variabilele A, B și C) care sunt *consistenti* cu funcția dată. Așadar, vom examina ce se întâmplă când în nodul rădăcină se pun pe rând atributele A, B și respectiv C .

Notăm cu X funcția $(\neg A \vee B) \wedge \neg(C \wedge A)$, ale cărei valori sunt date în tabelul alăturat.

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

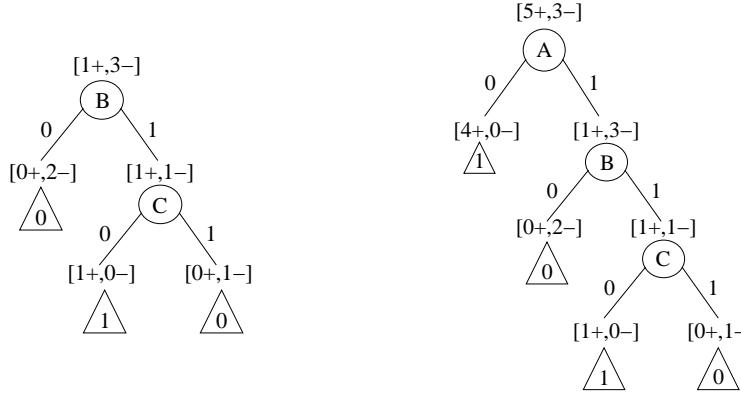
- *Cazul 1:* Dacă în nodul rădăcină se plasează atributul A , mulțimea de exemple va fi re-partiționată conform reprezentării alăturate.



Subarborele drept al acestui arbore va trebui să reprezinte arborele de decizie pentru funcția

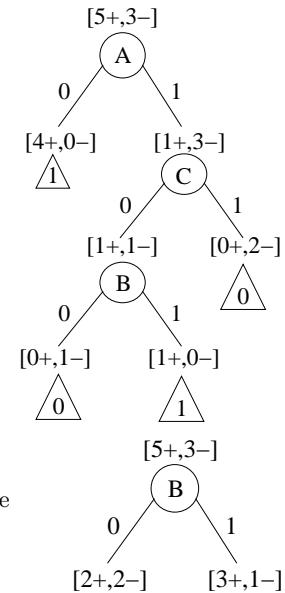
$$X_1 = X[A/1] = (\neg 1 \vee B) \wedge \neg(C \wedge 1) = B \wedge \neg C,$$

pentru care o reprezentare optimă este redată mai jos, în partea stângă:



Prin urmare, un arbore optim (ca număr de noduri) care are variabila A în nodul rădăcină este cel reprezentat mai sus, în partea dreaptă.

Observație: Evident, există încă un arbore optim care are variabila A în nodul rădăcină (el corespunde unei alte reprezentări optimale a conjuncției $B \wedge \neg C$ față de cea de mai sus). Vedeți desenul alăturat.



- *Cazul 2:* Dacă în nodul rădăcină se alege atributul B , se obține partitia reprezentată de arborele din figura alăturată.

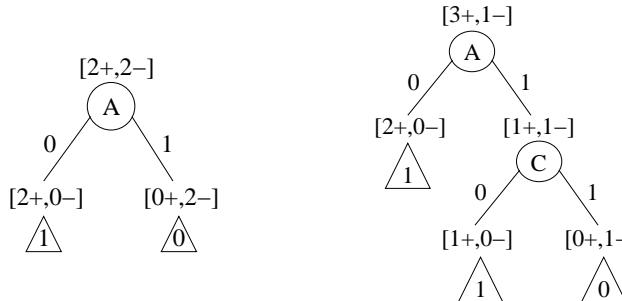
Subarborele stâng și subarborele drept trebuie să reprezinte arborele de decizie pentru funcțiile

$$X_2 = X[B/0] = (\neg A \vee 0) \wedge \neg(C \wedge A) = \neg A \wedge (\neg C \vee \neg A) = (\neg A \wedge \neg C) \vee \neg A = \neg A,$$

și respectiv

$$X_3 = X[B/1] = (\neg A \vee 1) \wedge \neg(C \wedge A) = 1 \wedge (\neg C \vee \neg A) = \neg C \vee \neg A,$$

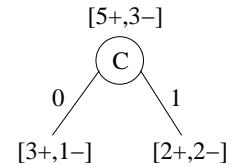
care au ca reprezentări optime arborii de mai jos:



Pentru arborele din dreapta există un arbore de decizie echivalent, obținut prin interschimbarea lui A cu C .

Prin urmare, orice arbore optim având variabila B în rădăcină are 3 niveluri și 4 noduri, aşadar cu un nod (de test) mai mult decât cel determinat în primul caz.

- *Cazul 3:* În sfârșit, când în nodul rădăcină se alege atributul C , obținem partiția reprezentată de arborele alăturat.



Subarborele stâng și subarborele drept trebuie să reprezinte arborele de decizie pentru funcțiile

$$X_4 = X[C/0] = (\neg A \vee B) \wedge \neg(0 \wedge A) = (\neg A \vee B) \wedge \neg 0 = \neg A \vee B$$

și respectiv

$$X_5 = X[C/1] = (\neg A \vee B) \wedge \neg(1 \wedge A) = (\neg A \vee B) \wedge \neg 1 = \neg A.$$

Urmând un raționament similar cu cel de la cazul anterior, putem spune că orice arbore optim cu atributul C în rădăcină are 3 niveluri și 4 noduri, cu un nod (de test) mai mult decât cel determinat în primul caz.

Așadar, putem concluziona că arborii de decizie optimi corespunzători funcției date sunt cei determinați în primul caz.

2.

(Algoritmul ID3: aplicare)

■ CMU, 2002 spring, A. Moore, midterm example questions, pr. 2

Ai naufragiat pe o insulă pustie, unde nu găsești niciun alt fel de hrănă decât ciuperci. Despre unele dintre aceste ciuperci se știe că sunt otrăvitoare, despre altele se știe că sunt comestibile, iar despre restul nu se știe ce fel sunt. Ai rămas singur pe insulă — foștii tăi camarazi, fiind epuizați de foame, au folosit metoda ‘trial and error’... — și ai la dispoziție următoarele date:

Exemplu	<i>Ușoară</i>	<i>Mirositoare</i>	<i>ArePete</i>	<i>Netedă</i>	<i>Comestibilă</i>
<i>A</i>	1	0	0	0	1
<i>B</i>	1	0	1	0	1
<i>C</i>	0	1	0	1	1
<i>D</i>	0	0	0	1	0
<i>E</i>	1	1	1	0	0
<i>F</i>	1	0	1	1	0
<i>G</i>	1	0	0	1	0
<i>H</i>	0	1	0	0	0
<i>U</i>	0	1	1	1	?
<i>V</i>	1	1	0	1	?
<i>W</i>	1	1	0	0	?

Atunci când nu vei mai avea la dispoziție pentru a supraviețui decât ciuperci U, V, sau W, ai putea estima care dintre ele sunt comestibile, folosind arbori de decizie.

În primele trei întrebări care urmează, ne vom referi la ciupercile *A – H*:

- Care este entropia atributului *Comestibilă*?
- Doar privind datele — adică fără a face explicit calculul câștigului de informație (engl., information gain) pentru cele patru atribute — poți determina ce atribut vei alege ca rădăcină a arborelui de decizie?
- Calculează câștigul de informație pentru atributul pe care l-ai ales la întrebarea precedentă.
- Elaborează întregul arbore de decizie ID3 bazat pe datele din tabel și apoi clasifică ciupercile U, V, W.
- Exprimă cu ajutorul calculului propozițional (logica predicatelor de ordinul 0) clasificarea produsă de arborele de decizie obținut. (*Comestibilă* ↔ ...)
- Există vreun risc dacă vei consuma ciuperci care au fost clasificate de arborele de decizie ca fiind comestibile? De ce da? sau, de ce nu?

Răspuns:

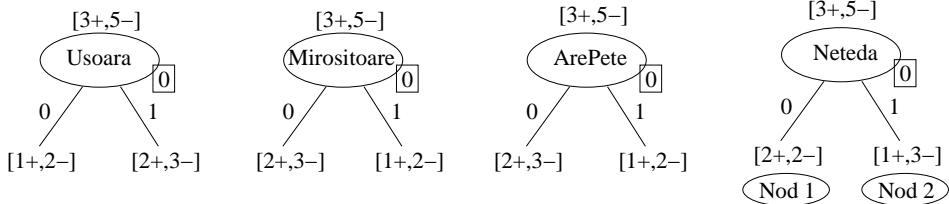
- Entropia atributului *Comestibilă* este:

$$\begin{aligned}
 H_{\text{Comestibilă}} &\stackrel{\text{not.}}{=} H[3+, 5-] \stackrel{\text{def.}}{=} -\frac{3}{8} \log_2 \frac{3}{8} - \frac{5}{8} \log_2 \frac{5}{8} = \frac{3}{8} \log_2 \frac{8}{3} + \frac{5}{8} \log_2 \frac{8}{5} = \\
 &= \frac{3}{8} 3 - \frac{3}{8} \log_2 3 + \frac{5}{8} 3 - \frac{5}{8} \log_2 5 = 3 - \frac{3}{8} \log_2 3 - \frac{5}{8} \log_2 5 \approx \\
 &\approx 0.9544
 \end{aligned}$$

Observație (1): Notația [3+, 5–] simbolizează o mulțime partitioanată în 3 exemple pozitive și 5 exemple negative. Vom folosi acest gen de notație peste tot în continuare, cu mici variații determinate de valorile pe care le poate lua atributul de ieșire. De exemplu, dacă vorbim despre o mulțime cu 5 obiecte roșii, 3 albastre și 4 verzi, am putea nota: [5R, 3A, 4V].

- În rădăcina arborelui de decizie se alege atributul care aduce cel mai mare câștig de informație. Adică, atributul care, intuitiv vorbind, partitioanează cel mai bine datele de

antrenament în raport cu atributul de ieșire. În cazul nostru, variantele pe care le avem la dispoziție pentru rădăcina arborelui (nodul 0) sunt:



Este ușor de observat că atributele *Ușoară*, *Mirosoitoare* și *ArePete* împart multimea exemplelor în mod similar: o submulțime cu 3 elemente, dintre care unul este pozitiv iar două sunt negative, și o submulțime cu 5 elemente, dintre care două sunt pozitive, iar trei sunt negative.

Dacă am considera un arbore de decizie cu un singur nod de test în care plasăm atributul *Netedă*, atunci numărul minim de erori la antrenare pe care îl putem obține este 3, utilizând următoarea clasificare:

- Netedă = 0 : Comestibilă = 1 \Rightarrow ciupercile E și H sunt clasificate greșit
 - Netedă = 1 : Comestibilă = 0 \Rightarrow ciuperca C este clasificată greșit

Dacă, în schimb, vom pune în rădăcina arborelui de decizie unul dintre celelalte trei atribute, spre exemplu atributul *Ușoară*, și dacă vom lua votul majoritar în fiecare nod descendant din nodul rădăcină, eroarea rezultată la antrenare va fi aceeași ca mai sus (3/8), însă toate instanțele vor fi clasificate la fel (și anume, negativ).

Dacă nu lucrăm cu vot majoritar pentru ambii descendenti, ci doar pentru cel cu entropie mai mică (în vreme ce pentru celălalt nod descendant luăm decizia contrară), se observă că pentru atributul *Ușoară* vom obține 4 erori pe setul de antrenament, iar pentru atributul *Netedă* vom obține 3 erori.

Sumarizând, suntem încinați să credem că ar fi o alegere sensibil mai bună să punem în rădăcină atributul *Neteda*. Pentru o justificare numerică riguroasă a acestei alegeri folosind criteriul maximizării câștigului de informație, vedeti punctul *d*.

c. Pentru a obține câștigul de informație pentru atributul *Netedă*, se fac calculele:

$$\begin{aligned}
H_{0/Neted\ddot{a}} &\stackrel{\text{def.}}{=} \frac{4}{8}H[2+, 2-] + \frac{4}{8}H[1+, 3-] = \frac{1}{2} \cdot 1 + \frac{1}{2} \left(\frac{1}{4} \log_2 \frac{4}{1} + \frac{3}{4} \log_2 \frac{4}{3} \right) \\
&= \frac{1}{2} + \frac{1}{2} \left(\frac{1}{4} \cdot 2 + \frac{3}{4} \cdot 2 - \frac{3}{4} \log_2 3 \right) = \frac{1}{2} + \frac{1}{2} \left(2 - \frac{3}{4} \log_2 3 \right) \\
&= \frac{1}{2} + 1 - \frac{3}{8} \log_2 3 = \frac{3}{2} - \frac{3}{8} \log_2 3 \approx 0.9056
\end{aligned}$$

$$IG_{0/Netedā} \stackrel{def.}{=} H_{Comestibilā} - H_{0/Netedā}$$

*Observație (2): În cele de mai sus am notat cu H_0 /*Netedă* entropia partitiei [mulțimii de exemple de antrenament] determinate de alegerea atributului *Netedă* în nodul 0*¹⁸¹ iar cu

¹⁸¹ Mai riguros, folosind terminologia din *Teoria informației*, vom spune că notația $H_0/Netedă$ se referă la *entropia condițională medie* a atributului de ieșire *Comestibilă* în raport cu atributul de intrare *Netedă*.

$IG_{0/Netedă}$ câștigul de informație corespunzător acestei alegeri. În general, prin notația $H_{n/A}$ vom înțelege entropia partiției determinate de alegerea atributului A în nodul n .

d. Arborele de decizie ID3 se construiește pornind din rădăcină și alegând atributul pentru fiecare nod de test în modul următor:

Nodul 0 (rădăcina):

Să verificăm dacă alegerea făcută la punctul b este cea corectă:

$$\begin{aligned}
 H_{0/Ușoară} &\stackrel{\text{def.}}{=} \frac{3}{8}H[1+, 2-] + \frac{5}{8}H[2+, 3-] \\
 &= \frac{3}{8}\left(\frac{1}{3}\log_2\frac{3}{1} + \frac{2}{3}\log_2\frac{3}{2}\right) + \frac{5}{8}\left(\frac{2}{5}\log_2\frac{5}{2} + \frac{3}{5}\log_2\frac{5}{3}\right) \\
 &= \frac{3}{8}\left(\frac{1}{3}\log_2 3 + \frac{2}{3}\log_2 3 - \frac{2}{3} \cdot 1\right) + \frac{5}{8}\left(\frac{2}{5}\log_2 5 - \frac{2}{5} \cdot 1 + \frac{3}{5}\log_2 5 - \frac{3}{5}\log_2 3\right) \\
 &= \frac{3}{8}\left(\log_2 3 - \frac{2}{3}\right) + \frac{5}{8}\left(\log_2 5 - \frac{3}{5}\log_2 3 - \frac{2}{5}\right) \\
 &= \frac{3}{8}\log_2 3 - \frac{2}{8} + \frac{5}{8}\log_2 5 - \frac{3}{8}\log_2 3 - \frac{2}{8} \\
 &= \frac{5}{8}\log_2 5 - \frac{4}{8} \approx 0.9512
 \end{aligned}$$

Urmărează că

$$IG_{0/Ușoară} \stackrel{\text{def.}}{=} H_{Comestibilă} - H_{0/Ușoară} = 0.9544 - 0.9512 = 0.0032,$$

deci

$$IG_{0/Ușoară} = IG_{0/Mirosoitoare} = IG_{0/ArePete} = 0.0032 < IG_{0/Netedă} = 0.0488$$

Am avut deci dreptate să alegem atributul *Netedă* la punctul b .

Observație importantă: În loc să fi calculat efectiv aceste câștiguri de informație, pentru a determina atributul cel mai „bun“, ar fi fost *suficient* să elaborăm un *raționament* de tip *relațional*, bazat pe comparația dintre valorile entropiilor condiționale medii $H_{0/Netedă}$ și $H_{0/Ușoară}$:

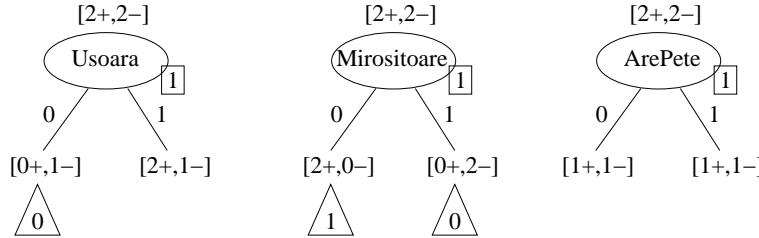
$$\begin{aligned}
 IG_{0/Netedă} > IG_{0/Ușoară} &\Leftrightarrow H_{0/Netedă} < H_{0/Ușoară} \\
 &\Leftrightarrow \frac{3}{2} - \frac{3}{8}\log_2 3 < \frac{5}{8}\log_2 5 - \frac{1}{2} \\
 &\Leftrightarrow 12 - 3\log_2 3 < 5\log_2 5 - 4 \\
 &\Leftrightarrow 16 < 5\log_2 5 + 3\log_2 3 \Leftrightarrow 16 < 11.6096 + 4.7548 \text{ (adev.)}
 \end{aligned}$$

În mod *alternativ*, ținând cont de relația (103) de la problema 35, putem proceda chiar *mai simplu* relativ la calcule (nu doar aici, ci ori de câte ori *nu* avem de-a face cu un număr mare de instanțe):

$$\begin{aligned}
 H_{0/Netedă} < H_{0/Ușoară} &\Leftrightarrow \frac{4^4}{2^2 \cdot 2^2} \cdot \frac{4^4}{3^3} < \frac{5^5}{2^2 \cdot 3^2} \cdot \frac{3^2}{2^2} \Leftrightarrow \frac{4^8}{3^3} < 5^5 \Leftrightarrow 4^8 < 3^3 \cdot 5^5 \\
 &\Leftrightarrow 2^{16} < 3^3 \cdot 5^5 \Leftrightarrow 64 \cdot 2^{10} < 27 \cdot 25 \cdot 125 \text{ (adev.)}
 \end{aligned}$$

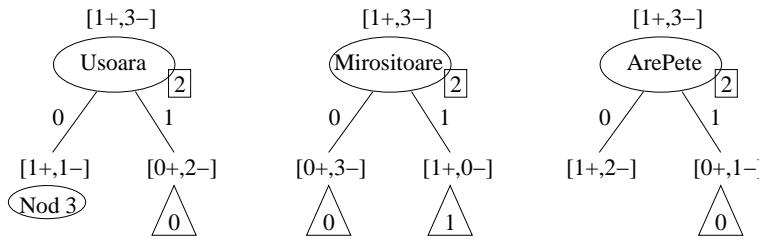
Vă sfătuim să *procedați aşa* la rezolvarea problemelor propuse din acest capitol, acolo unde este cazul.

Nodul 1: Trebuie să clasificăm acele exemple care au $Netedă = 0$; avem de ales 3 attribute - *Ușoară*, *Mirositoare* și *ArePete*.

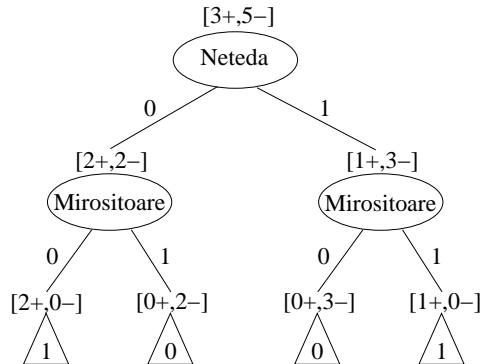


Audem $H_{1/Mirositoare} = \frac{2}{4}H[2+, 0-] + \frac{2}{4}H[0+, 2-] = 0$. Oricare ar fi valorile pentru $H_{1/ArePete}$ și $H_{1/Ușoară}$, întrucât stim că entropia are întotdeauna valori nenegative, rezultă că atributul *Mirositoare* maximizează în nodul 1 câștigul de informație. În imaginea de mai sus valorile din triunghi reprezintă decizia luată de subarborele construit în nodul frunză respectiv.

Nodul 2: Avem de clasificat exemplele pentru care $Netedă = 1$. Atributele disponibile sunt: *Ușoară*, *Mirositoare* și *ArePete*.



Evident, $H_{2/Mirositoare} = \frac{3}{4}H[0+, 3-] + \frac{1}{4}H[1+, 0-] = \frac{3}{4} \cdot 0 + \frac{1}{4} \cdot 0 = 0$. Așadar, pentru nodul 2 putem alege atributul *Mirositoare*.



Arborele complet arată astfel:

Parcurgând arborele construit, ciupercile *U*, *V* și *W* vor fi clasificate astfel:

<i>U</i>	$Netedă = 1, Mirosoitoare = 1 \Rightarrow Comestibilă = 1$
<i>V</i>	$Netedă = 1, Mirosoitoare = 1 \Rightarrow Comestibilă = 1$
<i>W</i>	$Netedă = 0, Mirosoitoare = 1 \Rightarrow Comestibilă = 0$

e. $\text{Comestibilă} \leftrightarrow (\neg\text{Netedă} \wedge \neg\text{Mirosoitoare}) \vee (\text{Netedă} \wedge \text{Mirosoitoare})$

Același lucru poate fi exprimat și sub forma unui pseudo-cod *if* ... *then* *Comestibilă* *else* \neg *Comestibilă*:

```
IF      (Netedă = 0 AND Mirosoitoare = 0) OR
       (Netedă = 1 AND Mirosoitoare = 1)
THEN   Comestibilă;
ELSE    $\neg$ Comestibilă;
```

f. Arborele de decizie produs de către algoritmul ID3 elaborat mai sus este consistent cu datele de antrenament pe care le-am avut la dispoziție (fiindcă aceste date sunt necontradictorii). Întrucât în realitate clasificarea poate depinde și de alte trăsături / informații decât cele de care dispunem noi, nu avem garanția că arborele ID3 face identificarea corectă a etichetei / clasei pentru toate instanțele din setul de test. Așadar, nu putem fi siguri că nu ne vom îmbolnăvi dacă vom consuma ciupercile *U* și *V*, sau că ne vom îmbolnăvi dacă vom consuma ciuperca *W*. În multe aplicații practice, calitatea unui model de învățare automată (în cazul de față, un arbore de decizie) se verifică pe un set de *date de validare*.

3. (Algoritmul ID3, aplicat pe expresii booleene; exploatarea simetriilor operațiilor \vee , \wedge în alegerea atributelor; analiza „optimalității“ arborelui ID3, ca număr de noduri de test)

prelucrare de Liviu Ciortuz, după Tom Mitchell, "Machine Learning", 1997, ex. 3.1.b

Considerăm următoarea funcție booleană: $A \vee (B \wedge C)$. Presupunem că această funcție este deja definită — adică valoarea ei este cea cunoscută din logica propozițiilor —, însă dorim să o reprezentăm ca arbore de decizie.

a. Aplicați algoritmul ID3 [tabelei de adevăr corespunzătoare] acestei funcții.

Observație: Dacă exploatați simetriile, veți avea nevoie doar de puține calcule, altfel vă veți complica inutil.

b. Arborele ID3 obținut la punctul precedent este optimal?

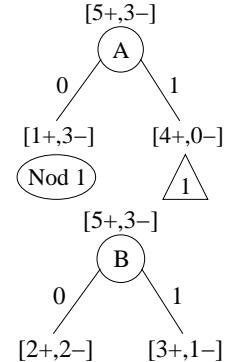
Alfel spus, puteți găsi un alt arbore de decizie, de adâncime mai mică sau cu număr mai mic de noduri (comparativ cu arborele obținut la punctul a), care să reprezinte această funcție? (Tineți cont că în fiecare nod al unui arbore de decizie se poate testa un singur atribut.)

Răspuns:

<i>A</i>	<i>B</i>	<i>C</i>	$Y = A \vee (B \wedge C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Nodul 0 (rădăcină):

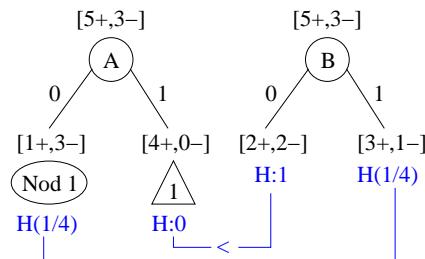
$$\begin{aligned}
 H_{0/A} &= \frac{4}{8}H[1+, 3-] + \frac{4}{8}H[4+, 0-] = \\
 &= \frac{1}{2}H[1+, 3-] + \frac{1}{2} \cdot 0 = \frac{1}{2}H[1+, 3-] \\
 H_{0/B} &= \frac{4}{8}H[2+, 2-] + \frac{4}{8}H[3+, 1-] = \\
 &= \frac{1}{2} \cdot 1 + \frac{1}{2}H[3+, 1-] = \frac{1}{2} + \frac{1}{2}H[1+, 3-]
 \end{aligned}$$



Este evident că $H_{0/A} < H_{0/B}$, deci vom alege atributul A în rădăcină.

Observații importante:

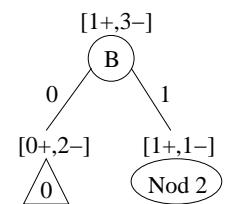
- La aceeași concluzie se putea ajunge *imediat* pe baza unui *raționament* de tip *calitativ*, și anume, comparând atent cei doi arbori („compași de de decizie“) de mai sus. Mai precis, vom compara (două căte două) entropiile condiționale specifice din nodurile descendente, precum și ponderile cu care se combină aceste entropii în scrierea entropiilor condiționale medii corespunzătoare atributelor A și B . Putem pune în evidență acest fapt în figura următoare, în care simbolul H , scris uneori însotit de un argument (așadar, ca $H(p)$), se referă la entropia unei variabile Bernoulli de parametru p .



Mai facem *precizarea* că semnele $<$ și $=$ din figura alăturată se referă de fapt nu [doar] la entropiile condiționale specifice, ci [și] la produsul acestora cu ponderile asociate în mod corespunzător: $\frac{4}{8}H[1+, 3-] = \frac{4}{8}H[3+, 1-]$ și respectiv $\frac{4}{8}H[4+, 0-] < \frac{4}{8}H[2+, 2-]$.

- Pentru câteva formule de calcul convenabile pentru entropii și câștiguri de informație relative la compași de decizie, atunci când se folosește calculatorul de buzunar, dar numărul de instanțe de antrenament asociate nu este prea mare, vedeți problema 35.

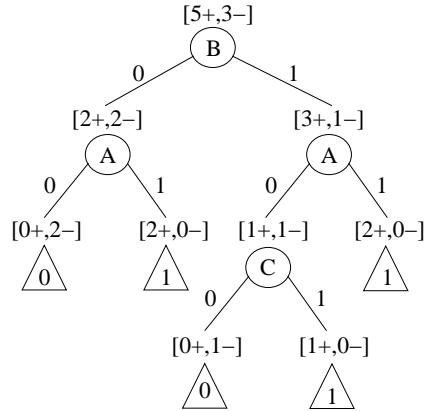
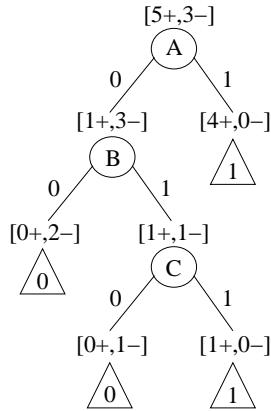
Nodul 1: Avem de clasificat instanțele care au $A = 0$ și putem alege între atributele B și C . Datorită simetriei, îl putem alege pe oricare dintre ele. Pentru fixare, îl alegem pe B .



Nodul 2: La acest punct a mai rămas disponibil doar atributul C .

Arborele construit de ID3 este cel reprezentat mai jos, în partea stângă:¹⁸²

¹⁸²Un alt arbore ID3 este cel obținut din acesta interschimbând atributele B și C . (Vedeți



b. Pentru a vedea dacă arborele construit de algoritmul ID3 este cel optimal, trebuie să reconsiderăm toate deciziile pe care le-am luat în construirea acestuia:

– La nodul 1 al arborelui avem de clasificat exemplele pentru care $A = 0$, deci funcția care trebuie reprezentată de subarborele în cauză este $f' = f[A/0] = 0 \vee (B \wedge C) = B \wedge C$, funcție care este reprezentată în mod optimal de subarborele construit de ID3. (Notația $A/0$ semnifică faptul că variabila logică A este instantiată la valoarea 0.) Prin urmare, nu există un arbore mai bun care să reprezinte funcția dată și să aibă în rădăcină atributul A .

– În rădăcină am ales atributul A în detrimentul celorlalte două attribute deoarece am demonstrat că aduce cel mai mare câștig de informație. Să vedem ce se întâmplă dacă alegem unul dintre atributele B sau C . După cum am discutat mai sus, datorită simetriei, pe oricare dintre cele două l-am alege, arborele rezultat ar avea aceeași formă. Pentru fixare, să-l alegem pe B .

Subarborele stâng și drept vor trebui să reprezinte funcțiile:

$$f'' = f[B/0] = A \vee (0 \wedge C) = A \vee 0 = A$$

și respectiv

$$f''' = f[B/1] = A \vee (1 \wedge C) = A \vee C.$$

Arborele minimal care poate fi construit în aceste circumstanțe este cel reprezentat mai sus în partea dreaptă. Acest arbore are 3 niveluri și 4 noduri, cu un nod în plus față de cel construit de algoritmul ID3.

Putem deci conchide că arborele construit respectând specificațiile algoritmului ID3 este cel optimal.

Observație:

Această problemă pune în evidență două modalități de parcursere a spațiului de versiuni pentru un concept, în particular unul din logica propozițiilor, care este reprezentat cu ajutorul arborilor de decizie. Pe de o parte avem explorarea (incompletă) făcută de algoritmul ID3 care este de tip "greedy", iar pe de altă parte avem explorarea exhaustivă. Prima strategie de explorare procedează la o căutare „orientată” a soluției (și din această cauză este mai eficientă, dar se va vedea, ca revers, că nu asigură întotdeauna găsirea

Observația din enunț.)

optimului), iar cea de-a doua strategie de explorare, deși asigură găsirea optimului, nu este utilizabilă în cazurile (frecvente!) în care spațiul de versiuni este foarte mare.

4. (ID3, ca algoritm “greedy”;
un exemplu când arborele ID3 nu este optimal
ca număr de noduri și de niveluri)
prelucrare de Liviu Ciortuz, după
■ CMU, 2003 fall, T. Mitchell, A. Moore, midterm exam, pr. 9.a

Fie atributele binare de intrare A, B, C , atributul de ieșire Y și următoarele exemple de antrenament:

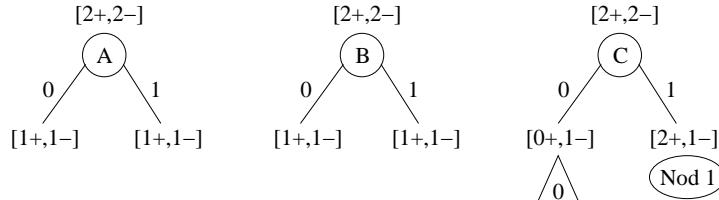
A	B	C	Y
1	1	0	0
1	0	1	1
0	1	1	1
0	0	1	0

- a. Determinați arborele de decizie calculat de algoritmul ID3. Este acest arbore de decizie *consistent* cu datele de antrenament?
b. Există un arbore de decizie de adâncime mai mică (decât cea a arborelui ID3) consistent cu datele de mai sus? Dacă da, ce concept (logic) reprezintă acest arbore?

Răspuns:

- a. Se construiește arborele de decizie cu algoritmul ID3 astfel:

Nodul 0 (rădăcina):



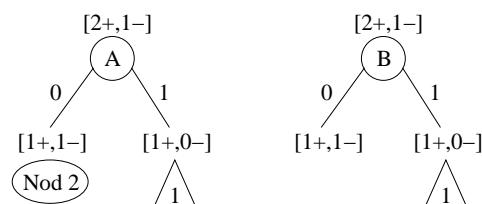
Observăm că $H_{0/A} = H_{0/B} = \frac{2}{4}H[1+, 1-] + \frac{2}{4}H[1+, 1-] = H[1+, 1-] = 1$, care este valoarea maximă a entropiei [condiționale medii a] unei variabile booleene. Prin urmare, $H_{0/C}$ nu poate fi decât mai mică sau egală cu $H_{0/A}$ și $H_{0/B}$. Deci vom alege în nodul rădăcină atributul C .

Nodul 1: Avem de clasificat instanțele cu $C = 1$, deci alegerea se face între atributele A și B .

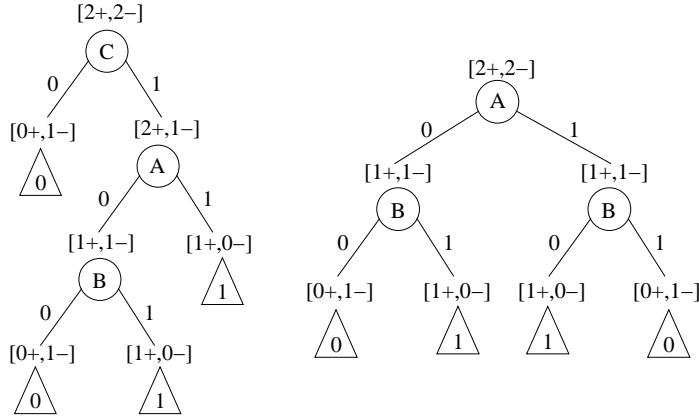
Cele două entropii condiționale medii sunt egale:

$$H_{1/A} = H_{1/B} = \frac{2}{3}H[1+, 1-] + \frac{1}{3}H[1+, 0-]$$

Așadar, putem alege oricare dintre cele două atrbute. Pentru fixare, îl alegem pe A .



Nodul 2: La acest nod nu mai avem decât atributul B , deci îl vom pune pe acesta. Arborele complet este reprezentat alăturat (în partea stângă):



Prin construcție, arborele ID3 este consistent cu datele de antrenament dacă acestea sunt consistente (i.e., necontradictorii). În cazul nostru, se verifică imediat că datele de antrenament sunt consistente.

b. Se observă că atributul de ieșire Y reprezintă de fapt funcția logică $A \text{ XOR } B$. Reprezentând această funcție ca arbore de decizie, vom obține arborele desenat mai sus în partea dreaptă. Acest arbore are cu un nivel mai puțin decât arborele construit cu algoritmul ID3. Prin urmare, arborele obținut de algoritmul ID3 pe datele din enunț *nu* este *optim* din punctul de vedere al numărului de niveluri. Aceasta este o *consecință* a caracterului “greedy” al algoritmului ID3, datorat faptului că la fiecare iterare alegem „cel mai bun” atribut în raport cu criteriul câștigului de informație. Se știe că algoritmii de tip “greedy” nu grantează obținerea optimului global.

5.

(Clasificare ternară: “decision stump” produs de ID3, pe date care conțin duplicări și „zgomote“)

Presupunem că se dau şase date de antrenament (precizate în tabel) pentru o problemă de clasificare cu două atrbute binare și trei clase $Y \in \{1, 2, 3\}$. Se va crea un arbore ID3, bazat pe câștigul de informație.

	X_1	X_2	Y	
	1	1	1	
	1	1	1	
	1	1	2	
a.	Calculați câștigul de informație atât pentru X_1 cât și pentru X_2 . Se va folosi aproximarea $\log_2 3 = 19/12$ și se va scrie câștigul de informație sub formă de fracții.	1	0	3
	0	0	2	
	0	0	3	

b. Pe baza rezultatelor anterioare, ce atribut va fi folosit pentru primul nod al arborelui ID3? Desenați arborele de decizie care rezultă folosind doar acest singur nod. Etichetați corespunzător nodul, ramurile și eticheta prevăzută în fiecare frunză.

c. Cum va clasifica acest arbore instanța determinată de $X_1 = 0$ și $X_2 = 1$?

Răspuns:

a. Redăm formula pentru calculul câștigului de informație în varianta folosită de Tom Mitchell:

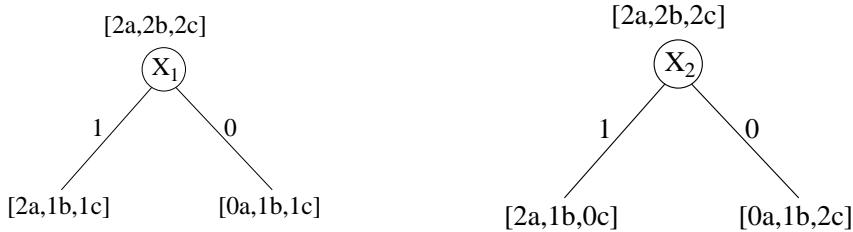
$$Gain(X_i) = Entropy(S) - \sum_{v \in Values(X_i)} \frac{|S_v|}{|S|} Entropy(S_v)$$

unde S este mulțimea celor 6 date de antrenament, iar

$$\text{Entropy}(S) = - \sum_{y \in Y} p_y \log_2 p_y$$

Notăm cu a clasa instanțelor având eticheta $Y = 1$, cu b clasa instanțelor cu $Y = 2$ și cu c clasa instanțelor cu $Y = 3$. În mulțimea S există câte 2 elemente din fiecare clasă și atunci putem scrie că $S = [2a, 2b, 2c]$.

Pentru nodul rădăcină, putem alege fie atributul X_1 , fie atributul X_2 , ceea ce determină următoarele împărțiri ale mulțimii S :



Putem calcula câștigurile de informație pentru cele două attribute:

$$\text{Gain}(X_1) = H[2a, 2b, 2c] - \left(\frac{4}{6}H[2a, 1b, 1c] + \frac{2}{6}H[0a, 1b, 1c] \right)$$

unde $H[2a, 2b, 2c]$ este o altă notație pentru entropia mulțimii compuse din două exemple de clasă a , două de clasă b și două de clasă c .

Calculăm entropiile care intervin în formulă:

$$\begin{aligned}
H[2a, 2b, 2c] &= -\frac{2}{6} \log_2 \frac{2}{6} - \frac{2}{6} \log_2 \frac{2}{6} - \frac{2}{6} \log_2 \frac{2}{6} \\
&= -3 \cdot \frac{2}{6} \log_2 \frac{2}{6} = -\log_2 \frac{1}{3} = \log_2 3 = \frac{19}{12} \\
H[2a, 1b, 1c] &= -\frac{2}{4} \log_2 \frac{2}{4} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{4} \log_2 \frac{1}{4} \\
&= +\frac{1}{2} \cdot \log_2 2 + \frac{1}{4} \log_2 4 + \frac{1}{4} \log_2 4 \\
&= \frac{1}{2} + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = \frac{3}{2} \\
H[0a, 1b, 1c] &= 0 - \frac{1}{2} \cdot \log_2 \frac{1}{2} - \frac{1}{2} \cdot \log_2 \frac{1}{2} = \log_2 2 = 1
\end{aligned}$$

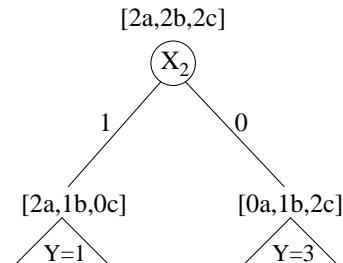
Înlocuind aceste valori numerice în formula câștigului de informație, obținem:

$$\text{Gain}(X_1) = \frac{19}{12} - \left(\frac{2}{3} \cdot \frac{3}{2} + \frac{1}{3} \cdot 1 \right) = \frac{19}{12} - \left(1 + \frac{1}{3} \right) = \frac{19}{12} - \frac{4}{3} = \frac{3}{12} = \frac{1}{4}$$

Se aplică aceleasi formule și pentru atributul X_2 :

$$\begin{aligned}
 Gain(X_2) &= H[2a, 2b, 2c] - \left(\frac{3}{6}H[2a, 1b, 0c] + \frac{3}{6}H[0a, 1b, 2c] \right) \\
 H[2a, 1b, 0c] &= H[0a, 1b, 2c] \\
 &= -\frac{1}{3}\log_2 \frac{1}{3} - \frac{2}{3}\log_2 \frac{2}{3} \\
 &= -\frac{2}{3}(\log_2 2 - \log_2 3) - \frac{1}{3}(-1)\log_2 3 \\
 &= -\frac{2}{3} \cdot 1 + \frac{2}{3} \cdot \frac{19}{12} + \frac{1}{3} \cdot \frac{19}{12} = \frac{19}{12} - \frac{2}{3} = \frac{11}{12} \\
 \Rightarrow Gain(X_2) &= \frac{19}{12} - \left(\frac{1}{2} \cdot \frac{11}{12} + \frac{1}{2} \cdot \frac{11}{12} \right) = \frac{19}{12} - \frac{11}{12} = \frac{8}{12} = \frac{2}{3}.
 \end{aligned}$$

b. Deoarece $Gain(X_1) = \frac{3}{12}$, $Gain(X_2) = \frac{8}{12}$, și deci $Gain(X_1) < Gain(X_2)$, se va alege atributul X_2 ca rădăcină a arborelui. Arborele de decizie construit din acest singur nod este cel din figura alăturată.



c. O instanță care are $X_1 = 0$ și $X_2 = 1$ va fi clasificată de acest arbore cu $Y = 1$ (cu probabilitate $2/3$).

6.

(Algoritmul ID3: aplicare pe date inconsistente, "decision stumps", calculul acurateții)

CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW1, pr. 2.ab

Tabelul de mai jos sumarizează situația celor 2201 de pasageri și membri ai echipejajului de la bordul vasului Titanic, în urma naufragiului din data de 15 Aprilie 1912. Pentru fiecare combinație de valori ale celor 3 variabile (Clasă, Sex, Vârstă) am indicat în tabel câți oameni au supraviețuit și câți nu au supraviețuit. (*Observație:* Datele originale au patru valori pentru atributul Clasă; am comasat valorile II, III, și Echipaj într-o singură valoare, denumită „Inferioară“.)

Clasa	Sexul	Vârstă	Supraviețuitori		
			Nu	Da	Total
I	Masculin	Copil	0	5	5
I	Masculin	Adult	118	57	175
I	Feminin	Copil	0	1	1
I	Feminin	Adult	4	140	144
Inferioară	Masculin	Copil	35	24	59
Inferioară	Masculin	Adult	1211	281	1492
Inferioară	Feminin	Copil	17	27	44
Inferioară	Feminin	Adult	105	176	281
			Total	1490	711
					2201

Pentru a vă ușura calculele pe care va trebui să le faceți, am făcut noi totalurile pentru fiecare variabilă:

Clasa	Supraviețuitori		
	Nu	Da	Total
I	122	203	325
Inferioară	1368	508	1876

Sexul	Supraviețuitori		
	Nu	Da	Total
Masculin	1364	367	1731
Feminin	126	344	470

Vârstă	Supraviețuitori		
	Nu	Da	Total
Copil	52	57	109
Adult	1438	654	2092

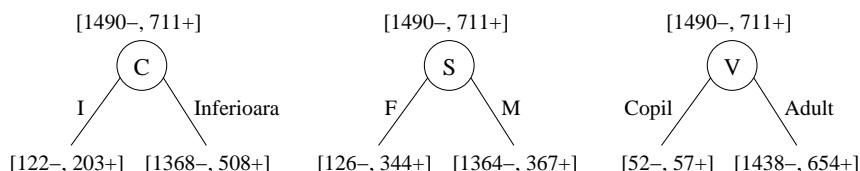
a. Folosind un arbore de decizie, dorim să prezicem variabila de ieșire Y (Supraviețuitor), pornind de la atributele de intrare C (Clasa), S (Sexul), V (Vârstă). Utilizați criteriul câștigului de informație pentru a alege care dintre aceste trei atrbute C , S sau V trebuie să fie folosit în nodul-rădăcină al arborelui de decizie.

De fapt, ce vi se cere este să învățați un arbore de decizie de adâncime 1 care folosește doar atributul din rădăcină pentru a clasifica datele. (Astfel de arbori de decizie de adâncime 1 sunt adesea numiți în terminologia de limbă engleză “decision stumps”.) Parcurgeți toate etapele rezolvării, redând inclusiv calculele pentru câștigul de informație al fiecărui atribut.

- b. Care este acuratețea [medie] obținută pe datele de antrenament de către arborele de decizie cu adâncime 1 de la punctul precedent?
- c. Dacă ați crea un arbore de decizie care folosește toate cele trei variabile, care ar fi acuratețea lui [medie] pe datele de antrenament? (*Observație:* Nu trebuie neapărat să creați arborele de decizie pentru a afla răspunsul!)

Răspuns:

- a. Totalurile care au fost furnizate în enunț pentru fiecare dintre variabilele C , S și V ne servesc foarte bine pentru a crea rapid cei trei “decision stumps”:



Analizând datele conform figurii de mai sus, se poate „intui” că atributul S va avea un câștig de informație (în raport cu atributul de ieșire Y – *Supraviețuitor*) mai bun decât al celorlalte două atrbute de intrare (C și V). Intuiția se verifică făcând calculele:

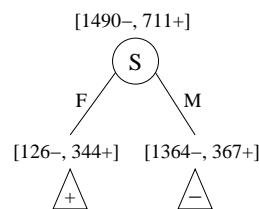
$$\begin{aligned}
 IG(Y, C) &= H[1490-, 711+] - \left(\frac{325}{2201} H[122-, 203+] + \frac{1876}{2201} H[1368-, 508+] \right) \\
 &= 0.048501 \\
 IG(Y, S) &= H[1490-, 711+] - \left(\frac{470}{2201} H[126-, 344+] + \frac{1731}{2201} H[1364-, 367+] \right) \\
 &= 0.142391
 \end{aligned}$$

$$\begin{aligned} IG(Y, V) &= H[1490-, 711+] - \left(\frac{109}{2201} H[52-, 57+] + \frac{2092}{2201} H[1438-, 654+] \right) \\ &= 0.006411. \end{aligned}$$

Deci, într-adevăr, câștigul maxim de informație se obține pentru atributul S .

- b. Arboarele de decizie de adâncime 1 care are în nodul rădăcină atributul S este cel din figura alăturată. Acuratețea [medie a] acestui arbore de decizie este:

$$\frac{470}{2201} \cdot \frac{344}{470} + \frac{1731}{2201} \cdot \frac{1364}{1731} = \frac{344 + 1364}{2201} = \frac{1708}{2201} = 0.776.$$



- c. Se poate constata imediat că arboarele ID3 produs pe datele din această problemă va avea 8 noduri-frunză, iar în fiecare dintre aceste noduri-frunză se va asigna câte una dintre mulțimile descrise (pe linie) în coloanele 4 și 5 ale tabelului principal din enunț: $[0-, 5+]$, $[118-, 57+]$, ..., $[17-, 27+]$, $[105-, 176+]$. Decizia care va fi luată în fiecare nod-frunză este dictată de votul majoritar, și anume: $+$, $-$, ..., $+$ și respectiv $+$.

Putem calcula acuratețea [medie] astfel:

$$\frac{5 + 118 + 1 + 140 + 35 + 1211 + 27 + 176}{2201} = \frac{1713}{2201} = 0.778.$$

Se observă că se produce (din păcate) o creștere foarte mică în raport cu acuratețea celui mai bun “decision stump”: doar 0.002.

7.

(Algoritmul ID3: cazul când există repetiții și inconsistențe în datele de antrenament; o margine superioară pentru eroarea la antrenare în funcție de numărul de valori ale variabilei de ieșire)
CMU, 2002 fall, Andrew Moore, midterm exam, pr. 1.fg

Presupunem că învățăm un arbore de decizie care să prezică atributul de ieșire Z pornind de la atributele de intrare A , B , C . Se folosesc datele de antrenament din tabelul de mai jos.

A	B	C	Z
0	0	0	0
0	0	1	0
0	0	1	0
0	1	0	0
0	1	1	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	0	1
1	1	1	0
1	1	1	1

- a. Care va fi eroarea la antrenare pe acest set de date? Exprimăți răspunsul sub forma fracției de înregistrări care vor fi clasificate eronat ($n/12$).

- b. Considerăm un arbore de decizie construit pe un set arbitrar de date. Dacă atributul de ieșire este cu valori discrete și poate lua k valori distincte, care este eroarea de antrenare maximă (exprimată ca fracție)?

Răspuns:

a. Este ușor de observat că datele de antrenament conțin „inconsistențe“ (contradicții, relativ la etichetare), și anume la exemplele $(0, 1, 1)$ și $(1, 1, 1)$. Fiecare dintre aceste exemple sunt etichetate o dată cu 0 și altă dată cu 1. Prin urmare, jumătate din aceste exemple vor fi clasificate eronat de arborele învățat de către algoritmul ID3. Eroarea la antrenare va fi deci $\frac{2}{12}$.

b. Vom analiza pe rând mai multe cazuri, care sunt din ce în ce mai generale.

Cazul *i*: Mai întâi vom calcula eroarea la antrenare pentru cazul în care setul de date de antrenament este compus din k instanțe care sunt identice ca tupluri de valori pentru atributele ce le caracterizează, dar sunt clasificate pe rând cu fiecare din cele k valori posibile ale atributului de ieșire. Este evident că arborele de decizie învățat va clasifica eronat $k - 1$ instanțe. Așadar, în acest caz, eroarea la antrenare va fi

$$E = \frac{k - 1}{k}$$

Cazul *ii*: Aceeași eroare [la antrenare] ca mai sus se va înregistra dacă în locul fiecărei instanțe dintre cele considerate la cazul precedent vom avea l instanțe identice, inclusiv în ce privește eticheta. (În total sunt kl instanțe de antrenament.)

$$E = \frac{(k - 1) \cdot l}{k \cdot l} = \frac{k - 1}{k}$$

Cazul *iii*: Dacă relaxăm condiția de mai sus considerând l_1, l_2, \dots, l_k instanțe identice, iar $l = \max_{i=1}^k l_i$, este imediat că eroarea maximă se va atinge în cazul $l_1 = l_2 = \dots = l_k = l$, și va avea aceeași valoare ca mai sus. Așadar, în continuare vom putea renunța la a considera factorul de multiplicare l , fără ca prin aceasta să restrângem generalitatea raționamentului.

Cazul *iv*: Fie n exemple de antrenament (instanțe etichetate) dintre care d sunt distințe (ca tupluri de valori ale atributelor de intrare). Fie $k_1, k_2 \dots k_d$ numărul de instanțe etichetate pentru fiecare caz distinct în parte din cele d . Atunci vom avea:

$$k_1 \leq k, k_2 \leq k, \dots, k_d \leq k \Rightarrow n = k_1 + k_2 + \dots + k_d \leq k \cdot d \text{ deci } n \leq k \cdot d$$

Eroarea maximă la antrenare va fi dată de formula

$$E = \frac{(k_1 - 1) + (k_2 - 1) + \dots + (k_d - 1)}{n} = \frac{n - d}{n}$$

Avem:

$$E \leq \frac{k - 1}{k} \Leftrightarrow \frac{n - d}{n} \leq \frac{k - 1}{k} \Leftrightarrow k \cdot n - k \cdot d \leq k \cdot n - n \Leftrightarrow k \cdot d \geq n \text{ (adev.)}$$

Prin urmare, eroarea maximă la antrenare care poate fi atinsă atunci când atributul de ieșire poate lua k valori distințe este $\frac{k - 1}{k}$.

8. (ID3, aspecte computaționale: influența atributelor duplicate, respectiv a instanțelor de antrenament duplicate asupra arborelui ID3 rezultat)

CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 3.1-2

Se dorește construirea unui arbore de decizie pentru n vectori, cu m atrbute.

- Să presupunem că există i și j astfel încât pentru TOTI vectorii X din datele de antrenament, aceste atrbute au valori egale (adică, $x_i = x_j$ pentru toți vectorii, unde x_i este valoarea atrbutului i în vectorul X). Să presupunem de asemenea că în cazul în care ambele atrbute duc la același câstig de informație vom folosi atrbutul i . Ștergerea atrbutului j din datele de antrenament poate schimba arborele de decizie obținut? Explicați pe scurt.
- Să presupunem că există în mulțimea de antrenament doi vectori egali X și Z (adică, toate atrbutele lui X și Z sunt exact la fel, inclusiv etichetele). Ștergerea vectorului Z din datele de antrenament poate schimba arborele de decizie obținut? Explicați pe scurt.

Răspuns:

- Nu, îndepărtarea atrbutului j nu schimbă arborele de decizie, deoarece atrbutele i și j conduc la valori egale pentru câstigul de informație în fiecare nod al arborelui.
- Da, în acest caz arborele de decizie se poate schimba, fiindcă entropia condițională — care se calculează în fiecare nod pentru a determina atrbutul cu câstigul de informație cel mai mare — depinde de numărul de instanțe de antrenament luate în considerare.

9. (Arbori de decizie: o margine superioară pentru numărul de noduri frunză, în funcție de numărul atrbutelor și numărul de exemple)

CMU, 2005 fall, T. Mitchell, A. Moore, midterm exam, pr. 2.d

Presupunem că învățăm un arbore de decizie folosind un set de R instanțe de antrenament descrise de M atrbute de intrare având valori binare.

Care este numărul maxim posibil de noduri frunză din arborele de decizie, presupunând că fiecărui nod frunză îi este asociat măcar un exemplu de antrenament? Încercuiți unul din răspunsurile de mai jos; justificați alegerea făcută.

$$\begin{aligned}
 & R, \log_2(R), R^2, 2^R, M, \log_2(M), M^2, 2^M, \\
 & \min(R, M), \min(R, \log_2(M)), \min(R, M^2), \min(R, 2^M), \\
 & \min(\log_2(R), M), \min(\log_2(R), \log_2(M)), \min(\log_2(R), M^2), \min(\log_2(R), 2^M), \\
 & \min(R^2, M), \min(R^2, \log_2(M)), \min(R^2, M^2), \min(R^2, 2^M), \\
 & \min(2^R, M), \min(2^R, \log_2(M)), \min(2^R, M^2), \min(2^R, 2^M), \\
 & \max(R, M), \max(R, \log_2(M)), \max(R, M^2), \max(R, 2^M), \\
 & \max(\log_2(R), M), \max(\log_2(R), \log_2(M)), \max(\log_2(R), M^2), \max(\log_2(R), 2^M), \\
 & \max(R^2, M), \max(R^2, \log_2(M)), \max(R^2, M^2), \max(R^2, 2^M), \\
 & \max(2^R, M), \max(2^R, \log_2(M)), \max(2^R, M^2), \max(2^R, 2^M).
 \end{aligned}$$

Răspuns:

Notăm cu \max_{frunze} valoarea căutată. Trebuie luate în considerare două aspecte:

- (a) fiecare nod frunză trebuie să clasifice măcar un exemplu de antrenament \Rightarrow nu putem să avem mai multe frunze decât exemple de antrenament $\Rightarrow \max_{frunze} \leq R$
- (b) fiecare atribut poate fi testat o singură dată pe un drum de la rădăcină la o frunză oarecare \Rightarrow arborele obținut va avea adâncimea cel mult $M \Rightarrow \max_{frunze} \leq 2^M$

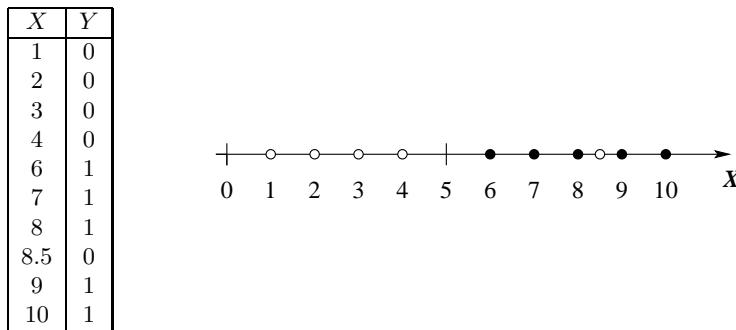
$$\left. \begin{array}{l} \max_{frunze} \leq R \\ \max_{frunze} \leq 2^M \end{array} \right\} \Rightarrow \max_{frunze} \leq \min(R, 2^M)$$

Această valoare poate fi atinsă: putem să luăm, de exemplu, cazul trivial când avem o singură instanță de antrenament. Prin urmare, avem $\max_{frunze} = \min(R, 2^M)$.

10. (Extensiile ale algoritmului ID3: variabile de intrare continue; “decision stumps”; eroarea la antrenare, eroarea la CVLOO; overfitting)

■ CMU, 2002 fall, Andrew Moore, midterm exam, pr. 3

Fie setul de date de mai jos. X este atribut de intrare și ia valori reale, iar Y este variabilă de ieșire cu valori booleene. (*Observație:* Am reprezentat acest set de date sub formă grafică, marcând valoarea / eticheta $Y = 1$ prin bulină neagră, iar valoarea / eticheta $Y = 0$ prin cerculeț alb.) Pe acest set de date se folosește algoritmul ID3 pentru învățare de arbori de decizie.



Algoritmul ID3 (extins) va trebui să decidă cum divide (engl., split) intervale de valori asociate variabilei reale X . Separarea în intervale diferite va fi stabilită cu ajutorul unor *praguri* (engl., split thresholds), determinate în felul următor:

- Mai întâi se ordonează crescător acele valori ale variabilei X care apar în datele de antrenament.
- Se stabilesc apoi perechi de valori consecutive pentru care există instanțe de antrenament care sunt etichetate în mod diferit pentru o valoare, comparativ cu cealaltă valoare. Pentru fiecare pereche de valori de acest fel, va fi plasat un prag de separare la jumătatea distanței dintre cele două valori.

Initial, se alege pragul de separare care conduce la un câștig de informație maxim. Apoi, la fiecare nouă execuție a buclei principale a algoritmului ID3 — vă readucem aminte că acest algoritm este recursiv — se va selecta câte un alt prag dintre cele rămase disponibile, aplicând același criteriu: maximizarea câștigului de informație.

De exemplu, pentru $X = 4$ avem o instanță de antrenament negativă, iar pentru $X = 6$ avem o instanță de antrenament pozitivă. Se poate arăta că algoritmul ID3 va decide să splitze mai întâi la valoarea $X = 5$ (care reprezintă jumătatea distanței dintre $X = 4$ și $X = 6$) și apoi la valoarea $X = 8.25$ (care reprezintă jumătatea distanței dintre $X = 8$ și $X = 8.5$).

Fie DT^* arborele de decizie complet, obținut de algoritm ID3 fără a face pruning, iar DT_2 arborele de decizie produs de ID3 urmat de pruning, care are doar două noduri frunză (deci DT_2 face o singură divizare de interval).

- Care este eroarea la antrenare produsă de DT_2 respectiv DT^* (exprimată ca număr de exemple clasificate eronat din totalul de 10 exemple)?
- Care este eroarea produsă de DT_2 respectiv DT^* la cross-validation folosind metoda Leave-One-Out (CVLOO)?

Răspuns:

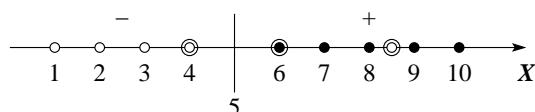
- Deoarece DT_2 reține doar un singur split, și anume la $X = 5$, regula de decizie pe care o reprezintă este:

```
IF      X ≤ 5
THEN    Y = 0
ELSE    Y = 1
```

Este evident că această regulă produce o clasificare eronată doar pentru una dintre instanțele de antrenament, și anume $X = 8.5$. Avem deci $E_{antren.}(DT_2) = 1/10$.

Întrucât exemplele nu conțin inconsistențe, arborele de decizie ID3 clasifică corect toate datele de antrenament. Avem deci $E_{antren.}(DT^*) = 0/10 = 0$.

- Figura de mai jos reprezintă împărțirea axei reale în *intervale / zone de decizie* conform arborelui (și *pragului de decizie*) învățat de către algoritmul DT_2 , folosind întregul set de exemple date. (Am încercuit acele puncte care, după cum se va vedea mai jos, vor constitui cazuri aparte la calcularea erorii de tip CVLOO cu algoritmul DT_2 .)



În ce privește cross-validation prin metoda "Leave-One-Out", se poate demonstra — calculele nu sunt arătate aici¹⁸³ — următorul fapt: pentru fiecare din cele 10 exemple ($X = 1, X = 2, \dots, X = 10$) considerate pe rând, pragul de decizie identificat de algoritmul DT_2 va fi $X = 5$, cu excepția următoarelor două cazuri:

- $X = 4$: în acest caz, splitarea se va face la mijlocul intervalului $[3, 6]$, deci la 4.5. Cum $4 \leq 4.5$, rezultă că exemplul $X = 4$ va fi clasificat corect;
- $X = 6$: splitarea se va face la mijlocul intervalului $[4, 7]$, deci la 5.5. Cum $6 > 5.5$, rezultă că exemplul $X = 6$ va fi clasificat corect.

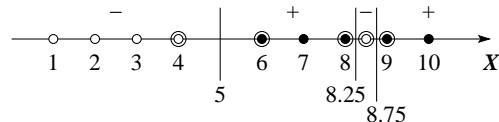
Atunci când punctul $X = 8.5$ este lăsat deoparte, pragul de decizie selectat fiind $X = 5$, va rezulta că punctul $X = 8.5$ este clasificat pozitiv (deci eronat), întrucât $8.5 > 5$.

¹⁸³Aceste calcule nu sunt dificile, mai ales dacă se utilizează raționamente de tip „calitativ“, așa cum am arătat la problema 3.

Este imediat că pentru restul de 7 cazuri ($X = 1, 2, 3, 7, 8, 9, 10$), arborele determinat de algoritmul DT2 va clasifica corect punctul X .

Așadar, pentru DT2 rezultă $E_{CVLOO} = 1/10$.

Figura de mai jos reprezintă împărțirea axei reale în *intervale / zone de decizie* conform *pragurilor de decizie* (și arborelui DT*) determinate de către algoritmul ID3 pe întregul set de date de antrenament.¹⁸⁴



În această figură am încercuit exemplele care ar putea conduce la eroare la cross-validationarea de tip "Leave-One-Out":

- $X = 4$: corect clasificat, explicația este aceeași ca în cazul DT2;
- $X = 6$: idem;
- $X = 8$: split-ul trebuie făcut la mijlocul intervalului $[7, 8.5]$, adică 7.75. Cum $8 > 7.75$, rezultă că punctul $X = 8$ va fi clasificat negativ, ceea ce este eronat;
- $X = 8.5$: nu este nevoie decât de un singur split, arborele DT* învățat în acest caz fiind identic cu cel construit de algoritmul DT2. Cum $8 > 5$, rezultă că punctul $X = 8.5$ va fi clasificat pozitiv, deci eronat;
- $X = 9$: intervalul de split devine $[8.5, 10]$, split-ul făcându-se la 9.25. Cum $9 \leq 9.25$, rezultă că punctul $X = 9$ va fi clasificat negativ, ceea ce este eronat.

Așadar, pentru DT* avem $E_{CVLOO} = 3/10$.

În *concluzie*, se observă că $E_{antren.}(DT2) = 1/10 > 0 = E_{antren.}(DT^*)$, în vreme ce $E_{CVLOO}(DT2) = 1/10 < 3/10 = E_{CVLOO}(DT^*)$. Acesta este un caz tipic de manifestare a fenomenului de *overfitting (supra-specializare)*.

11.

(Arbori de decizie cu variabile de intrare continue;
o margine superioară pentru adâncimea arborilor
în cazul (ne)separabilității liniare în \mathbb{R}^2)

CMU, 2009 spring, Ziv Bar-Joseph, midterm exam, pr. 5.cd

Se consideră n vectori bidimensionali ($x = \{x_1, x_2\}$) care pot fi clasificați folosind o funcție [de regresie] liniară, adică există $w \in \mathbb{R}^2$ și $b \in \mathbb{R}$ astfel încât:

$$y = \begin{cases} +1 & \text{if } w \cdot x + b > 0 \\ -1 & \text{if } w \cdot x + b \leq 0 \end{cases}$$

¹⁸⁴Observație: Atunci când ne raportăm doar la zonele de decizie în ansamblu, ordinea în care sunt stabilit testele în arborele ID3 este irelevantă! Acest fapt este valabil și mai jos, unde discutăm despre eroarea la CVLOO cu algoritmul DT*.

- a. Poate un arbore binar de decizie — atenție!, nu neapărat arborele ID3 — să clasifice corect acești vectori? Dacă nu, justificați. Dacă da, determinați adâncimea maximă (adică numărul maxim de niveluri de test) ale unui astfel de arbore de decizie, care este optim (ca număr de niveluri de test).
- b. Acum să presupunem că aceste n date nu sunt separabile liniar (adică nu există $w \in \mathbb{R}^2$ și $b \in \mathbb{R}$ cu proprietatea de mai sus). Poate un arbore de decizie (binar) să clasifice corect acești vectori? Dacă nu, justificați. Dacă da, care este adâncimea maximă a unui arbore de decizie corespunzător, optim ca număr de niveluri?

Răspuns:

- a. Da. O strategie posibilă pentru a construi un arbore de decizie binar care să clasifice corect aceste puncte este următoarea:

Mai întâi vom construi un arbore de decizie considerând doar atributul x_1 . În particular, făcând „înjumătățiri“ succesive ale mulțimii de valori ale acestui atribut, arborele rezultat va avea o adâncime maximă de $\lceil \log_2 n \rceil$ (adică partea întreagă superioară din $\log_2 n$) niveluri. În fiecare nod frunză al acestui arbore se va găsi o mulțime de puncte, însă nu neapărat cu aceeași clasificare. Totuși, deoarece datele sunt liniar separabile, pentru fiecare dintre aceste mulțimi de puncte se poate găsi o valoare a atributului x_2 care să o împartă în două submulțimi clasificate corect.

Prin urmare, arborele construit inițial considerând doar valoarea x_1 are nevoie să i se adauge doar cel mult câte un nod în fiecare frunză, nod care să clasifice corect datele luând în considerare valoarea x_2 . Adâncimea totală a arborelui astfel construit este $1 + \lceil \log_2 n \rceil$, adică de ordinul $O(\log n)$.

- b. Similar punctului anterior, se construiește un arbore de decizie considerând doar atributul x_1 , ceea ce înseamnă o adâncime de $\lceil \log_2 n \rceil$. În fiecare nod frunză al acestui arbore se va găsi o mulțime de puncte, posibil cu clasificări diferite. Datele de antrenament nu mai sunt liniar separabile, deci nu pot fi clasificate corect printr-un singur nod.

Pentru fiecare astfel de nod frunză în care punctele nu au aceeași clasificare, se aplică algoritmul de determinare a arborelui de decizie, de această dată luând în considerare doar valoarea lui x_2 . Aceasta presupune din nou o adâncime maximă a subarborelui de $\lceil \log_2 n \rceil$. În total, arborele obținut are adâncimea maximă $\lceil \log_2 n \rceil + \lceil \log_2 n \rceil$, deci tot de ordinul $O(\log n)$.

12.

(Algoritmul ID3 cu atribute discrete, respectiv atribute discrete și un atribut continuu: aplicare; predicție)

■ CMU, 2012 fall, E. Xing, A. Singh, HW1, pr. 1.1

Până în luna septembrie a anului 2012, 800 de planete extrasolare (numite în continuare exoplanete) au fost identificate în galaxia noastră. Niște navete spațiale super-secrete au fost trimise pentru a survola toate aceste exoplanete, cu scopul de a stabili dacă ele sunt locuibile sau nu de către oameni. Evident, a trimite căte o navetă spațială la fiecare dintre aceste exoplanete este extrem de costisitor. De aceea, în această problemă vă propunem să elaborați un arbore de decizie pentru a prezice dacă o exoplanetă este locuibilă sau nu, folosind doar trăsături / caracteristici (engl., features) observabile cu ajutorul telescopelor terestre.

- a. În tabelul de mai jos vi se dău anumite date în legătură cu toate cele 800 de planete survolate până acum. Trăsăturile observate cu ajutorul telescopelor sunt *Size* ("Big" sau "Small") și *Orbit* ("Near" sau "Far").

Fiecare linie din tabel indică valori ale acestor două trăsături, caracterul habitabil ("Yes" sau "No"), precum și de câte ori a fost identificată fiecare combinație de valori [pentru cele trei trăsături]. De exemplu, au fost identificate 20 de planete mari ("Big"), care sunt situate pe orbite apropiate ("Near") de soarele / steaua lor și sunt locuibile.

Size	Orbit	Habitable	Count
Big	Near	Yes	20
Big	Far	Yes	170
Small	Near	Yes	139
Small	Far	Yes	45
Big	Near	No	130
Big	Far	No	30
Small	Near	No	11
Small	Far	No	255

Elaborați și desenați arborele de decizie învățat de către algoritmul ID3 pe aceste date. (Folosiți criteriul câștigului de informație; nu aplicați pruning-ul.) La fiecare nod din arbore, scrieți numărul de planete locuibile și respectiv nelocuibile din datele de antrenament care sunt asociate cu nodul respectiv.

- b. Pentru doar 9 dintre aceste exoplanete, a fost măsurată o a treia trăsătură, *Temperature* (exprimată în grade Kelvin), după cum se arată în tabelul de mai jos.

Refaceti toti pașii de la punctul a, de data aceasta folosind toate cele trei trăsături de intrare. Pentru trăsătura *Temperature* (văzută ca atribut numeric cu valori continue), la fiecare iterație va trebui să faceți maximizarea în raport cu toate pragurile adecvate pentru separare binară (engl., binary thresholding splits). Iată un exemplu de test pentru o astfel de separare binară: $T \leq 250$ vs. $T > 250$.

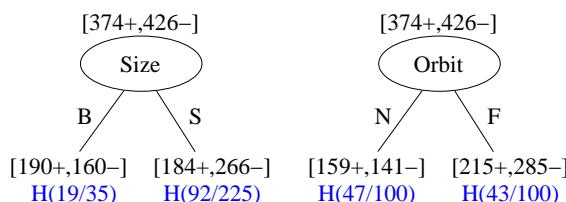
Size	Orbit	Temperature	Habitable
Big	Far	205	No
Big	Near	205	No
Big	Near	260	Yes
Big	Near	380	Yes
Small	Far	205	No
Small	Far	260	Yes
Small	Near	260	Yes
Small	Near	380	No
Small	Near	380	No

- c. Conform arborelui de decizie pe care l-ați obținut la punctul b, cum va fi clasificată o planetă având trăsăturile (Big, Near, 280), locuibilă sau nelocuibilă?

Indicație: Este posibil să aveți nevoie de următoarele valori pentru entropia ($H(p)$) unei variabile aleatoare Bernoulli de parametru p : $H(1/3) = 0.9182$, $H(2/5) = 0.9709$, $H(92/225) = 0.9759$, $H(43/100) = 0.9858$, $H(16/35) = 0.9946$, $H(47/100) = 0.9974$.

Răspuns:

- a. „Compașii de decizie“ corespunzători nodului rădăcină sunt înfățișați în desenul următor:



Sub fiecare nod descendant am notat entropia nodului respectiv, făcând referire la distribuția Bernoulli și dând (de fiecare dată) parametrului acestei distribuții valoarea corespunzătoare. Pentru nodul descendant corespunzător lui Orbit = Near am ținut cont și de faptul că entropia distribuției Bernoulli, ca funcție de parametru p , este simetrică față de valoarea $1/2$.¹⁸⁵

Entropiile condiționale medii corespunzătoare acestor doi „compasi de decizie“ sunt:

$$\begin{aligned} H(Habitable|Size) &= \frac{35}{80} \cdot H\left(\frac{19}{35}\right) + \frac{45}{80} \cdot H\left(\frac{92}{225}\right) = \frac{35}{80} \cdot 0.9946 + \frac{45}{80} \cdot 0.9759 \\ &= 0.9841 \\ H(Habitable|Orbit) &= \frac{3}{8} \cdot H\left(\frac{47}{100}\right) + \frac{5}{8} \cdot H\left(\frac{43}{100}\right) = \frac{3}{8} \cdot 0.9974 + \frac{5}{8} \cdot 0.9858 \\ &= 0.9901 \end{aligned}$$

Suntem acum în măsură să desemnăm atributul care va fi plasat în nodul rădăcină al arborelui care va fi construit de algoritmul ID3 pe datele din enunț: este atributul *Size*, întrucât $H(Habitable|Size) < H(Habitable|Orbit)$.

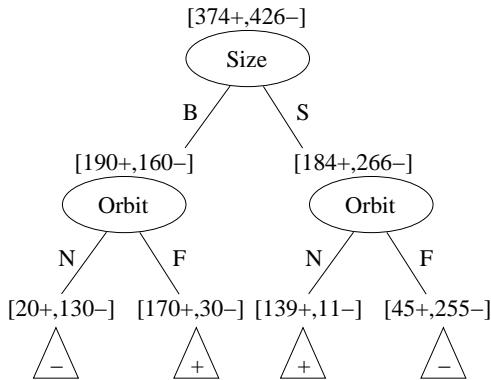
Doar cu titlu de informare, precizăm și câștigurile de informație realizate de cele două atrbute de intrare în raport cu atributul de ieșire:

$$IG(Habitable; Size) = H(Habitable) - H(Habitable|Size) = 0.9969 - 0.9841 = 0.0128$$

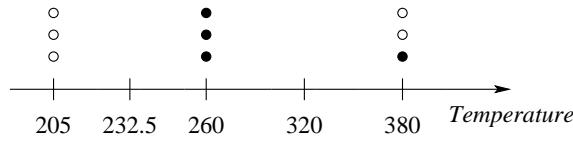
și, similar,

$$IG(Habitable; Orbit) = 0.0067.$$

Întrucât nu avem decât două atrbute de intrare, arborele de decizie va fi:



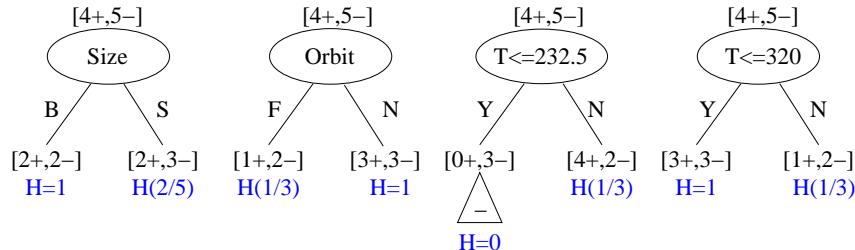
b. Pragurile de separare corespunzătoare atrbutoului *Temperature* sunt determinate conform imaginii următoare:



¹⁸⁵Este util să revedeți explicațiile date în *observația importantă* de la pagina 214.

Nivelul 1 (rădăcina):

„Compașii de decizie“ corespunzători acestui nivel sunt:



Observând cu atenție partițiile formate,¹⁸⁶ vom constata că entropia condițională medie pentru testul $Temperature \leq 232.5$ (în raport cu atributul de ieșire *Habitable*) are o valoare mai mică decât fiecare dintre entropiile condiționale medii $H(Habitable|Orbit)$ și $H(Habitable|Temperature \leq 320)$ (care, de fapt, sunt egale între ele).¹⁸⁷

Relația dintre $H(Habitable|Temperature \leq 232.5)$ și $H(Habitable|Size)$ se determină cu ajutorul calculelor:

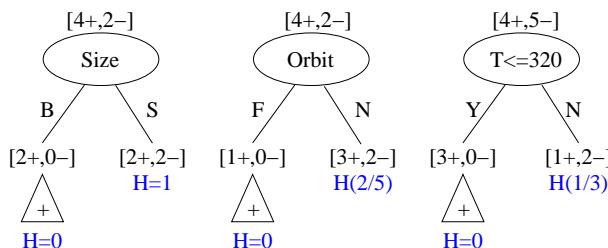
$$\begin{aligned} H(Habitable|Size) &= \frac{4}{9} + \frac{5}{9} \cdot H\left(\frac{2}{5}\right) = \frac{4}{9} + \frac{5}{9} \cdot 0.9709 = 0.9838 \\ H(Habitable|Temp \leq 232.5) &= \frac{2}{3} \cdot H\left(\frac{1}{3}\right) = \frac{2}{3} \cdot 0.9182 = 0.6121. \end{aligned}$$

Deși nu mai este necesar, indicăm și valorile câștigurilor de informație:

$$\begin{aligned} IG(Habitable; Size) &= 0.0072 \\ IG(Habitable; Orbit) &= 0.0183 \\ IG(Habitable; Temp \leq 232.5) &= 0.3788 \\ IG(Habitable; Temp \leq 320) &= 0.0183. \end{aligned}$$

Prin urmare, vom reține pentru nivelul 0 testul $Temperature \leq 232.5$.

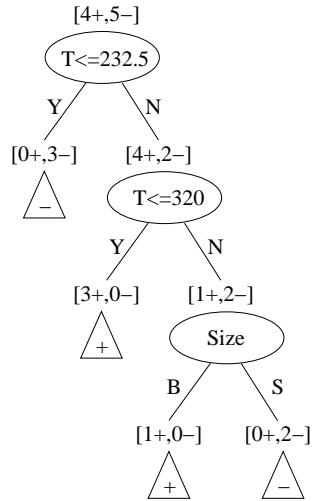
Nivelul 2 (mai exact, aici ne limităm la datele cu $Temperature > 232.5$): „Compașii de decizie“ corespunzători [completării] acestui nivel sunt:



¹⁸⁶Se compară două căte două entropiile condiționale specifice, precum și ponderile datelor respective în [raport cu] numărul total de instanțe asociate nodului părinte.

¹⁸⁷Observați de exemplu că $H(Habitable|Orbit = F) > H(Habitable|Temperature \leq 232.5)$ și $H(Habitable|Orbit = N) > H(Habitable|Temperature > 232.5)$, iar ponderile corespunzătoare acestor entropii condiționale specifice în calculul entropiilor condiționale medii $H(Habitable|Orbit = N)$ și $H(Habitable|Temperature \leq 232.5)$ sunt egale căte două (și anume, cu 3/9 și respectiv 6/9).

Este imediat că, pe aceste date, $H(\text{Habitable} | \text{Temp} \leq 320) < H(\text{Habitable} | \text{Size})$ și, de asemenea, $H(\text{Habitable} | \text{Temp} \leq 320) < H(\text{Habitable} | \text{Orbit})$.¹⁸⁸ Prin urmare, aici va fi ales testul $\text{Temp} \leq 320$.



Nivelul 3 (mai exact, aici ne limităm la datele cu $\text{Temperature} > 320$):

Se poate observa că, pentru aceste date, atributul *Size* are putere discriminativă maximă. Așadar, arborele de decizie final va fi cel reprezentat alăturat.

c. Conform arborelui de decizie obținut la punctul anterior, o exoplanetă având trăsăturile (Big, Near, 280) va fi clasificată ca fiind locuibilă.

13.

(Extensiile ale algoritmului ID3:
cazul atributelor discrete cu număr mare de valori)
CMU, 2008(?) spring, HW2, pr. 1

Se dorește antrenarea unui arbore de decizie care să clasifice exemple cu două atribută de intrare X_1, X_2 și un atribut de ieșire Y care are valorile 1 și 2. Primul atribut, X_1 , este binar, pe când al doilea atribut, X_2 , are 6 valori posibile A, B, C, D, E, F . Se dau următoarele 12 exemple de antrenament, câte 6 din fiecare clasă:

$Y = 1$	(1, A)	(0, E)	(1, B)	(1, B)	(1, F)	(0, D)
$Y = 2$	(0, A)	(0, C)	(1, E)	(0, F)	(0, B)	(1, D)

Vă reamintim formula câștigului de informație la partajarea mulțimii de exemple S în funcție de valorile atributului A :

$$\text{Gain}(S, A) = H(S) - H(S | A), \text{ cu } H(S | A) = \sum_{v \in \text{valori}(A)} \frac{|S_v|}{|S|} \cdot H(S_v),$$

unde $H(S)$ este entropia mulțimii de exemple S , iar $H(S | A)$ este entropia condițională medie a mulțimii S în raport cu atributul A , calculată aşa cum se vede mai sus, ca sumă ponderată a entropiilor submulțimilor lui S determinate de valorile atributului A .

¹⁸⁸Mai exact, ar fi trebuit să scriem: $H(\text{Habitable} | \text{Temp} > 232.5, \text{Temp} \leq 320) < H(\text{Habitable} | \text{Temp} > 232.5, \text{Size})$ și respectiv $H(\text{Habitable} | \text{Temp} > 232.5, \text{Temp} \leq 320) < H(\text{Habitable} | \text{Temp} > 232.5, \text{Orbit})$.

a. Determinați atributul ales în rădăcină folosind câștigul de informație. Folosiți aproxi-marea $\log_2 3 = 1.585$.

b. Determinați atributul ales în nodul rădăcină folosind o măsură numită *gain ratio impurity*, adică alegeti acel atribut care maximizează raportul

$$\frac{\text{Gain}(S, A)}{-\sum_v P(A=v) \cdot \log_2 P(A=v)} = \frac{H(S) - H(S|A)}{H(A)}.$$

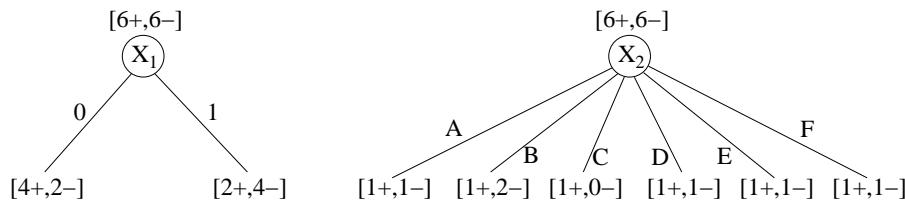
c. Având în vedere rezultatele de la punctele precedente, analizați utilitatea folosirii măsurii *gain ratio impurity* în cazurile în care attributele au numere diferite de valori posibile.

Răspuns:

a. Calculăm în primul rând entropia nodului rădăcină (sau a atributului de ieșire), care are 6 exemple pozitive și 6 negative:

$$H(Y) = H[6+, 6-] = -\frac{6}{12} \log_2 \frac{6}{12} - \frac{6}{12} \log_2 \frac{6}{12} = 2 \cdot \frac{1}{2} \log_2 2 = 1$$

În rădăcină poate fi ales fie atributul X_1 , fie atributul X_2 , obținând următoarele împărțiri:



Dacă vom pune atributul X_1 în nodul rădăcină, câștigul de informație va fi:

$$\begin{aligned} \text{Gain}(S, X_1) &= H(Y) - \frac{6}{12} H[4+, 2-] - \frac{6}{12} H[2+, 4-] = \\ &= 1 - \frac{6}{12} \left(-\frac{4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6} \right) - \frac{6}{12} \left(-\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} \right) \\ &= 1 - 2 \cdot \frac{1}{2} \left(\frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 \frac{3}{2} \right) = 1 - \left(\log_2 3 - \frac{2}{3} \right) \\ &= \frac{5}{3} - \log_2 3 \approx 0.0817 \end{aligned}$$

Altminteri, punând atributul X_2 în nodul rădăcină, câștigul de informație este:

$$\begin{aligned} \text{Gain}(S, X_2) &= H(Y) - 4 \cdot \frac{2}{12} H[1+, 1-] - \frac{3}{12} H[1+, 2-] - \frac{1}{12} H[1+, 0-] \\ &= 1 - \frac{2}{3} \cdot 1 - \frac{3}{12} \left(-\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} \right) - \frac{1}{12} \cdot 0 \\ &= 1 - \frac{2}{3} - \frac{1}{4} \left(\frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 \frac{3}{2} \right) = 1 - \frac{2}{3} - \frac{1}{4} \left(\log_2 3 - \frac{2}{3} \right) \\ &= \frac{1}{2} - \frac{1}{4} \log_2 3 \approx 0.1037 \end{aligned}$$

Folosind drept criteriu de optimizat în fiecare nod câştigul de informație, în rădăcină vom pune atributul X_2 , întrucât $Gain(S, X_1) < Gain(S, X_2)$.

b. Dacă vom pune atributul X_1 în nodul rădăcină, *gain ratio impurity* va fi:

$$\frac{Gain(S, X_1)}{-\sum_{i \in \{0,1\}} P(X_1 = i) \cdot \log_2 P(X_1 = i)} = \frac{Gain(S, X_1)}{-\frac{6}{12} \log_2 \frac{6}{12} - \frac{6}{12} \log_2 \frac{6}{12}} = \frac{Gain(S, X_1)}{1} \approx 0.0817$$

În schimb, plasând atributul X_2 în nodul rădăcină, *gain ratio impurity* va fi:

$$\begin{aligned} & \frac{Gain(S, X_2)}{-\sum_{j \in \{A, \dots, F\}} P(X_2 = j) \cdot \log_2 P(X_2 = j)} = \\ &= \frac{Gain(S, X_2)}{-4 \cdot \frac{2}{12} \log_2 \frac{2}{12} - \frac{3}{12} \log_2 \frac{3}{12} - \frac{1}{12} \log_2 \frac{1}{12}} \\ &= \frac{Gain(S, X_2)}{\frac{2}{3} \log_2 6 + \frac{1}{4} \log_2 4 + \frac{1}{12} \log_2 12} = \frac{Gain(S, X_2)}{\frac{4}{3} + \frac{3}{4} \log_2 3} \approx 0.0411 \end{aligned}$$

Prin urmare, în nodul rădăcină vom pune atributul X_1 , pentru care măsura *gain ratio impurity* are valoarea cea mai mare.

c. Câștigul de informație favorizează alegerea atributelor care au un număr mare de valori, indiferent dacă ele determină sau nu partitioarea în mod semnificativ a datelor de antrenament. În schimb, măsura *gain ratio impurity* ia în considerare, prin cantitatea de la numitor (vedeți definiția), numărul de valori ale atributului respectiv, mai exact mărimea mulțimilor de instanțe asignate nodurilor-fii, care au fost generate ca urmare a alegerii respectivului atribut. Valoarea de la numitor va crește odată cu numărul de noduri-fii, și totodată cu numărul de noduri-fii care au asignate puține exemple. Prin urmare, această măsură penalizează attributele cu multe valori, evitând favorizarea de care se face vinovat câștigul de informație într-o atare situație.

14.

(Extensiile ale algoritmului ID3: cazul când se ia în considerare costul calculării atributelor; attribute continue; “decision stumps”)

CMU, 2008 spring, T. Mitchell, W. Cohen, HW1, pr. 1

Se dă următorul set de date. Acesta reprezintă fișele a 12 pacienți ipotetici, ținând cont de sex, vîrstă (peste 60 ani sau nu), dacă suferă sau nu de diabet, dacă au pulsul mărit (sau nu) și EKG-ul anormal (sau nu). Pacienții sunt clasificați în final după cum prezintă (sau nu) aritmie.

Pacient	Sex	Peste60	Diabetic	Puls	EKG	AreAritmie
1	<i>M</i>	1	1	0	0	0
2	<i>M</i>	0	0	1	1	1
3	<i>M</i>	0	1	1	0	0
4	<i>M</i>	1	0	0	1	1
5	<i>M</i>	1	1	1	0	1
6	<i>M</i>	0	1	1	0	1
7	<i>F</i>	0	0	1	0	0
8	<i>F</i>	1	1	1	1	1
9	<i>F</i>	0	1	0	1	1
10	<i>F</i>	1	0	0	0	0
11	<i>F</i>	1	1	0	0	0
12	<i>F</i>	1	0	1	1	1

a. Calculați entropia condițională specifică $H(\text{AreAritmie} \mid \text{Sex} = F)$.

b. Dacă pentru selectarea atributelor se folosește măsura $\frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}$ în schimbul câștigului informational, care va fi atributul pus în nodul rădăcină? Se consideră că $\text{Cost}(\text{Sex}) = \text{Cost}(\text{Peste60}) = 1$, $\text{Cost}(\text{Diabetic}) = 3$, $\text{Cost}(\text{Puls}) = 2$ și $\text{Cost}(\text{EKG}) = 5$.

Observație: În calcule se vor utiliza următoarele aproximări: $\log_2 3 = 1.585$, $\log_2 5 = 2.322$ și $\log_2 7 = 2.807$.

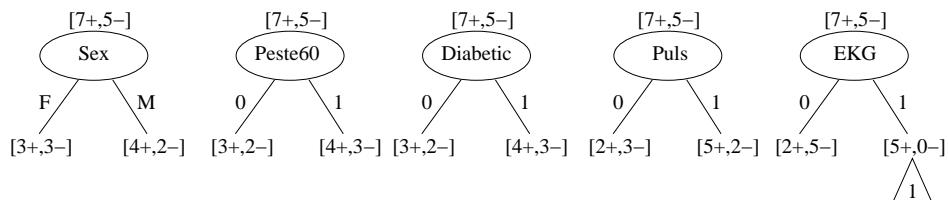
c. Să presupunem că, pentru un alt set de pacienți, se cunoaște vârstă lor exactă. Pentru exemplele pozitive, vârstele sunt: {40, 60, 62, 64, 70, 74, 75, 82}, iar pentru exemplele negative sunt: {33, 35, 42, 45, 49, 52, 58, 59, 80}. Să presupunem că toate celelalte attribute sunt predictori „slabi“, prin urmare dorim ca arborele să aibă un singur nod, rădăcina, care să împartă exemplele cu valori continue ale atributului vârstă în două: $vârstă < k$ și $vârstă \geq k$. Care va fi valoarea aleasă pentru k , bazat pe câștigul de informație?

Răspuns:

a. Entropia condițională cerută este:

$$H(\text{AreAritmie} \mid \text{Sex} = F) = H[3+, 3-] = 1$$

b. În rădăcina arborelui de decizie se alege atributul A pentru care raportul $\frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}$ este maxim. În cazul nostru, variantele pe care le avem sunt:



Se observă direct că $\text{Gain}(S, \text{EKG})$ este mai mare decât $\text{Gain}(S, A)$ pentru orice atribut $A \neq \text{EKG}$. Însă și $\text{Cost}(\text{EKG})$ este mai mare decât $\text{Cost}(A)$ pentru orice $A \neq \text{EKG}$. Așadar, trebuie să facem calculele în detaliu.

Entropia atributului de ieșire, AreAritmie, este:

$$\begin{aligned}
 H(\text{AreAritmie}) &= H[7+, 5-] = \frac{7}{12} \cdot \log_2 \frac{12}{7} + \frac{5}{12} \cdot \log_2 \frac{12}{5} \\
 &= \frac{7}{12} \cdot \log_2 12 - \frac{7}{12} \cdot \log_2 7 + \frac{5}{12} \cdot \log_2 12 - \frac{5}{12} \cdot \log_2 5 \\
 &= \log_2(3 \cdot 4) - \frac{7}{12} \cdot \log_2 7 - \frac{5}{12} \cdot \log_2 5 \\
 &= \log_2 3 + \underbrace{\log_2 4}_{=2} - \frac{7}{12} \cdot \log_2 7 - \frac{5}{12} \cdot \log_2 5 \approx 0.98
 \end{aligned}$$

Vom calcula câștigul de informație pentru fiecare din cele 5 atribute — se observă că pentru atributele Peste60 și Diabetic, câștigurile de informație sunt egale —, și apoi vom face raportul $\frac{Gain^2}{Cost}$:

Pentru atributul Sex:

$$\begin{aligned}
 Gain(S, \text{Sex}) &= H(\text{AreAritmie}) - \frac{6}{12}H[3+, 3-] - \frac{6}{12}H[4+, 2-] \\
 &= H(\text{AreAritmie}) - \frac{1}{2} \cdot 1 - \frac{1}{2} \left(-\frac{4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6} \right) \\
 &= 0.98 - \frac{1}{2} - \frac{1}{2} \left(\log_2 3 - \frac{2}{3} \right) = 0.98 - \frac{1}{2} - \frac{1}{2} \log_2 3 + \frac{1}{3} \approx 0.02
 \end{aligned}$$

Prin urmare, $\frac{Gain^2(S, \text{Sex})}{Cost(\text{Sex})} \approx \frac{0.02^2}{1} = 0.0004$.

Pentru atributele Peste60 și Diabetic:

$$\begin{aligned}
 Gain(S, \text{Peste60}) &= Gain(S, \text{Diabetic}) \\
 &= H(\text{AreAritmie}) - \frac{5}{12}H[3+, 2-] - \frac{7}{12}H[4+, 3-] \\
 &= 0.98 - \frac{5}{12} \left(\frac{3}{5} \log_2 \frac{5}{3} + \frac{2}{5} \log_2 \frac{5}{2} \right) - \frac{7}{12} \left(\frac{4}{7} \log_2 \frac{7}{4} + \frac{3}{7} \log_2 \frac{7}{3} \right) \\
 &= 0.98 - \frac{5}{12} \left(\log_2 5 - \frac{3}{5} \log_2 3 - \frac{2}{5} \cdot 1 \right) - \frac{7}{12} \left(\log_2 7 - \frac{4}{7} \cdot 2 - \frac{3}{7} \log_2 3 \right) \\
 &= 0.98 - \frac{5}{12} \log_2 5 - \frac{7}{12} \log_2 7 + \frac{1}{2} \log_2 3 + \frac{5}{6} \approx 0.0009
 \end{aligned}$$

Deci $\frac{Gain^2(S, \text{Peste60})}{Cost(\text{Peste60})} \approx \frac{0.0009^2}{1} = 81 \cdot 10^{-8}$ și $\frac{Gain^2(S, \text{Diabetic})}{Cost(\text{Diabetic})} \approx \frac{0.0009^2}{3} = 27 \cdot 10^{-8}$, ambele fiind niște valori foarte mici.

Pentru atributul Puls:

$$\begin{aligned}
 Gain(S, \text{Puls}) &= H(\text{AreAritmie}) - \frac{5}{12}H[2+, 3-] - \frac{7}{12}H[5+, 2-] \\
 &= 0.98 - \frac{5}{12} \left(\frac{2}{5} \log_2 \frac{5}{2} + \frac{3}{5} \log_2 \frac{5}{3} \right) - \frac{7}{12} \left(\frac{5}{7} \log_2 \frac{7}{5} + \frac{2}{7} \log_2 \frac{7}{2} \right) \\
 &= 0.98 - \frac{7}{12} \log_2 7 + \frac{1}{4} \log_2 3 + \frac{1}{3} \approx 0.072
 \end{aligned}$$

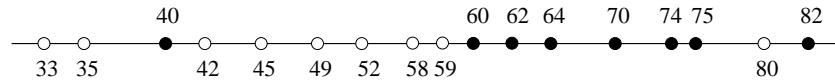
Prin urmare, $\frac{Gain^2(S, \text{Puls})}{Cost(\text{Puls})} \approx \frac{0.072^2}{2} = 0.002592$.

Pentru atributul EKG:

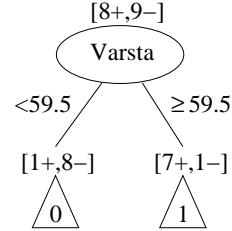
$$\begin{aligned} Gain(S, \text{EKG}) &= H(\text{AreAritmie}) - \frac{7}{12}H[2+, 5-] - \frac{5}{12}H[5+, 0-] \\ &= H(\text{AreAritmie}) - \frac{7}{12}\left(\frac{2}{7}\log_2\frac{7}{2} + \frac{5}{7}\log_2\frac{7}{5}\right) - \frac{5}{12} \cdot 0 \\ &= 0.98 + \frac{5}{12}\log_2 5 - \frac{7}{12}\log_2 7 + \frac{1}{6} \approx 0.476 \end{aligned}$$

Deci $\frac{Gain^2(S, \text{EKG})}{Cost(\text{EKG})} \approx \frac{0.476^2}{5} = 0.0453152$. Este evident că aceasta este cea mai mare valoare, de aceea în nodul rădăcină va fi ales atributul *EKG*, deși costul acestuia este cel mai mare.

c. Putem reprezenta exemplele astfel:



Se observă că alegerea cea mai bună este $k = 59.5$, arboarele de decizie fiind cel din figura alăturată.



15.

(Alte criterii posibile pentru selecția atributelor în ID3:
Gini impurity și Misclassification impurity)

prelucrare de Liviu Ciortuz, după
■ CMU, 2003 fall, T. Mitchell, A. Moore, HW1, pr. 4

Entropia este o mărime care cuantifică gradul de neomogenitate (engl., impurity) al unui set de instanțe în raport cu etichetele asignate. Algoritmul ID3 folosește entropia drept criteriu de partitioare (engl., splitting criterion), calculând *câștigul de informație* pentru a decide care este atributul care trebuie testat în nodul curent. Există, însă, și alte măsuri de neomogenitate care pot fi folosite, de asemenea, drept criterii de partitioare. În această problemă, vom investiga două astfel de măsuri.

Presupunem că nodul curent (n) din arborele de decizie aflat în curs de elaborare are asignate instanțe din k clase: c_1, c_2, \dots, c_k . Definim

$$Gini Impurity: i(n) = 1 - \sum_{i=1}^k P^2(c_i)$$

și

$$Misclassification Impurity: i(n) = 1 - \max_{i=1}^k P(c_i),$$

unde am notat cu $P(c_i)$ probabilitatea [sau: frecvența de apariție a instanțelor aparținând clasei c_i în ansamblul instanțelor asignate la nodul curent].

a. Presupunem $k = 2$. Așadar, în acest caz nodul n are două clase: c_1 și c_2 . Dese-nați un grafic în care cele trei măsuri de neomogenitate — *Entropia*, *Gini Impurity* și *Misclassification Impurity* — sunt reprezentate în funcție de $P(c_1)$.

b. Acum putem da definiția unui nou criteriu de partitioanare, bazat pe măsurile de neomogenitate *Gini* și *Misclassification*. În literatura de specialitate, acest nou criteriu este denumit uneori *Drop-of-Impurity* (pentru care propunem ca traducere în limba română termenul de *diminuarea neomogenității*). El reprezintă diferența dintre neomogenitatea nodului curent pe de o parte și suma ponderată a neomogenităților fiilor pe de altă parte. În cazul partitioanării atributelor binare, definim *Drop-of-Impurity* ca fiind:

$$\Delta i(n) = i(n) - P(n_l) i(n_l) - P(n_r) i(n_r),$$

unde n_l și n_r reprezintă fiul-stânga și, respectiv, fiul-dreapta, care au fost derivați din nodul n după partitioanare.

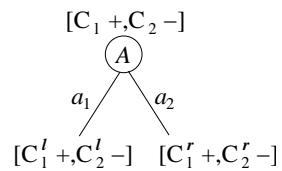
Folosind mai întâi *Gini Impurity* și apoi *Misclassification Impurity*, calculați *Drop-of-Impurity* pentru următorul set de instanțe asignate nodului pentru care se testează atributul A luând valorile a_1 și a_2 . Am notat cu C variabila de ieșire (desemnând clasa) și cu c_1 și c_2 cele două valori ale ei.

A	a_1	a_1	a_1	a_2	a_2	a_2
C	c_1	c_1	c_2	c_2	c_2	c_2

c. Se poate crea un set de date de antrenament (sau: se poate modifica setul de date de mai sus) astfel încât, pe noul set, *Drop-of-Impurity* bazat pe *Misclassification* să fie 0 dar bazat pe *Entropy* și, respectiv, *Gini* să fie diferit de 0?

Sugestie: Puteti folosi următoarea proprietate, care este ușor de demonstrat:

Dacă într-un set de date avem C_1 instanțe din clasa (sau: cu eticheta) c_1 și C_2 instanțe din clasa c_2 , cu $C_1 < C_2$, iar după partitioanarea în funcție de valorile atributului A această relație se păstrează, adică $C_1^l < C_2^l$ și $C_1^r < C_2^r$ (evident, cu $C_1 = C_1^l + C_1^r$ și $C_2 = C_2^l + C_2^r$), unde l și r desemnează nodul-fiu stâng și respectiv nodul-fiu drept, atunci *Drop-of-Impurity* pentru *Misclassification* va fi 0. Însă, în aceleși condiții, *Drop-of-Impurity* bazat pe *Gini* sau pe *Entropy* va avea, în general, valori nenule.

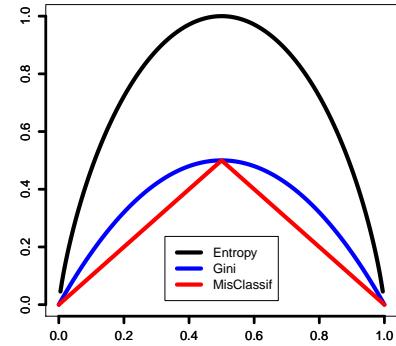


(Evident, proprietatea de mai sus se menține dacă în locul relației $<$ vom considera peste tot relația $>$.)

Răspuns:

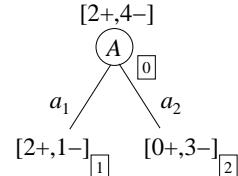
a. Vom scrie mai întâi expresiile celor trei funcții, iar apoi vom trasa graficele lor:

$$\begin{aligned}
 Entropy(p) &= -p \log_2 p - (1-p) \log_2(1-p) \\
 Gini(p) &= 1 - p^2 - (1-p)^2 = 2p(1-p) \\
 MisClassif(p) &= \\
 &= \begin{cases} 1 - (1-p), & \text{pentru } p \in [0; 1/2] \\ 1-p, & \text{pentru } p \in [1/2; 1] \end{cases} \\
 &= \begin{cases} p, & \text{pentru } p \in [0; 1/2] \\ 1-p, & \text{pentru } p \in [1/2; 1] \end{cases}
 \end{aligned}$$



Se observă că toate cele trei funcții iau valori în intervalul $[0; 1]$, sunt simetrice în raport cu punctul $p = 1/2$, sunt strict crescătoare pe intervalul $[0; 1/2]$ și strict descrescătoare pe intervalul $[1/2; 1]$, maximul fiecăreia dintre ele fiind obținut pentru $p = 1/2$.

b. Partiționarea datelor de antrenament în funcție de valorile atributului A se face aşa cum se arată în figura alăturată. (În această figură și, de asemenea, în calculele de mai jos, pentru conveniență / simplitate, am asociat semnul + etichetei c_1 și semnul - etichetei c_2 .)



Aplicând formula $\Delta i(n) = i(n) - P(n_l) i(n_l) - P(n_r) i(n_r)$, vom obține pentru *Drop-of-Impurity* următoarele valori:

Gini: $p = 2/6 = 1/3 \Rightarrow$

$$\left. \begin{aligned}
 i(0) &= 2 \cdot \frac{1}{3}(1 - \frac{1}{3}) = \frac{2}{3} \cdot \frac{2}{3} = \frac{4}{9} \\
 i(1) &= 2 \cdot \frac{2}{3}(1 - \frac{2}{3}) = \frac{4}{3} \cdot \frac{1}{3} = \frac{4}{9} \\
 i(2) &= 0
 \end{aligned} \right\} \Rightarrow \Delta i(0) = \frac{4}{9} - \frac{3}{6} \cdot \frac{4}{9} = \frac{4}{9} - \frac{2}{9} = \frac{2}{9}.$$

Misclassification: $p = 1/3 < 1/2 \Rightarrow$

$$\left. \begin{aligned}
 i(0) &= p = \frac{1}{3} \\
 i(1) &= 1 - \frac{2}{3} = \frac{1}{3} \\
 i(2) &= 0
 \end{aligned} \right\} \Rightarrow \Delta i(0) = \frac{1}{3} - \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}.$$

c. Într-adevăr, putem justifica ușor *proprietatea* din enunț:

$$\begin{aligned}
 \Delta i(n) &= \frac{C_1}{C_1 + C_2} - \left(\frac{C_1^l + C_2^l}{C_1 + C_2} \cdot \frac{C_1^l}{C_1^l + C_2^l} + \frac{C_1^r + C_2^r}{C_1 + C_2} \cdot \frac{C_1^r}{C_1^r + C_2^r} \right) \\
 &= \frac{C_1}{C_1 + C_2} - \frac{C_1^l + C_1^r}{C_1 + C_2} = \frac{C_1}{C_1 + C_2} - \frac{C_1}{C_1 + C_2} = 0.
 \end{aligned}$$

Dacă în setul de date din enunț una dintre instanțele (a_1, c_1) se modifică în (a_1, c_2) și se adaugă o instanță (a_2, c_1) , atunci *Drop-of-Impurity* va avea următoarele valori:

Entropy: $\Delta i(0) = H[5+, 2-] - \left(\frac{3}{7}H[2+, 1-] + \frac{4}{7}H[3+, 1-] \right) = 0.006 \neq 0;$

$$\begin{aligned}
 Gini: & 2 \left\{ \frac{2}{7} \left(1 - \frac{2}{7} \right) - \left[\frac{3}{7} \cdot \frac{1}{3} \left(1 - \frac{1}{3} \right) + \frac{4}{7} \cdot \frac{1}{4} \left(1 - \frac{1}{4} \right) \right] \right\} = 2 \left\{ \frac{10}{49} - \left[\frac{2}{21} + \frac{3}{28} \right] \right\} \\
 & = 2 \left(\frac{10}{49} - \frac{17}{84} \right) \neq 0; \\
 Misclassification: & \Delta i(0) = \frac{2}{7} - \left(\frac{3}{7} \cdot \frac{1}{3} + \frac{4}{7} \cdot \frac{1}{4} \right) = 0.
 \end{aligned}$$

16. (Algoritmul ID3 ca metodă de învățare de tip “eager”: posibilitatea suplimentării datelor de antrenament)
CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW1, pr. 3.4

Presupunem că, pornind de la un set de date de antrenament D , obținem un arbore de decizie ID3, notat cu T . Ulterior, cineva ne mai dă un set suplimentar de date de antrenament, D' . Putem proceda într-unul din următoarele două moduri:

- Putem rula din nou ID3, de această dată pe datele de antrenament $D \cup D'$, obținând arboarele T_1 . (Dezavantaj: dacă $|D|$ este foarte mare, această procedură poate fi costisitoare ca timp.)
- Putem extinde T , arboarele ID3 obținut pe mulțimea de antrenament D , ținând cont de datele D' . Arboarele nou, T_2 , ar putea să nu fie la fel de bun ca T_1 (vedeți cazul de mai sus), dar el este consistent cu datele din $D \cup D'$ în cazul în care aceste mulțimi de antrenament nu conțin zgomote. În special dacă $|D'|$ este mic, această metodă este acceptabilă din punct de vedere practic.

Propuneți o procedură pentru obținerea efectivă a arborelui T_2 .

Răspuns:

Instanțele din D' se atașează nodurilor frunză ale arborelui T , conform procedurii de clasificare de la arbori de decizie. Pentru fiecare dintre aceste noduri frunză, dacă instanțele atașate nodului respectiv nu sunt toate etichetate identic, se aplică algoritmul ID3 (folosind mulțimea de atribute neutilizate pe drumul care unește nodul rădăcină al arborelui cu acest nod frunză). În acest mod se obține un alt arbore de decizie T_2 care este consistent cu datele din $D \cup D'$.

17. (Reducerea caracterului “greedy” al algoritmului ID3 prin calcularea câștigului de informație în maniera “2-step look-ahead”)
CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW4, pr. 1.2-6

Învățarea automată a arborilor de decizie depinde mult de utilizarea unui mecanism “greedy” de selecție a atributelor.

- a. Dacă un set de date are A atribute booleene, calculați (ca expresie în funcție de A) numărul total de apeluri la funcția care calculează câștigul de informație pentru elaborarea întregului arbore de decizie.

Pentru *simplicitate*, se va presupune că toate atributele sunt necesare pentru clasificarea unei instanțe, iar setul de date de antrenament conține toate instanțele posibile (adică toate combinațiile posibile de perechi atribut-valoare, inclusiv pentru atributul de ieșire).

b. Este posibil să îmbunătățim algoritmul ID3, făcându-l să se comporte mai puțin "greedy", prin explorarea / prospectarea în avans (engl., look-ahead) a spațiului de căutare. La o explorare cu 2 pași înainte (engl., 2-step look-ahead), calculul câștigului de informație pentru un atribut a_i pe mulțimea de instanțe D va fi făcut cu ajutorul formulei

$$IG_{2\text{-step}}(D, a_i) = \max_{a_l, a_r} \left\{ \frac{n_l}{n_l + n_r} IG(D_l, a_l) + \frac{n_r}{n_l + n_r} IG(D_r, a_r) \right\},$$

unde

- a_l și a_r sunt attributele din nodurile descendente din nodul marcat cu atributul a_i ,
- D_l și D_r sunt seturile de instanțe asignate nodurilor descendente din a_i , iar n_l și n_r reprezintă cardinalul mulțimii D_l și respectiv D_r ;
- $IG(D_l, a_l)$ este câștigul de informație calculat (în sens clasic) pentru atributul a_l pe setul D_l ; similar, $IG(D_r, a_r)$ este câștigul de informație pentru atributul a_r pe setul D_r .

b1. Explicați pe scurt de ce sunt necesari factorii $\frac{n_l}{n_l+n_r}$ și $\frac{n_r}{n_l+n_r}$ în formula de mai sus.

b2. Dacă se folosesc A attribute booleene, câte apeluri la funcția $IG(\dots, \dots)$ sunt necesare pentru a stabili atributul din nodul rădăcină?

b3. Vom evalua acum cât de costisitoare este această explorare în avans a spațiului de căutare.

Dacă $A = 10$ și se face aceeași presupozitie ca la punctul a , calculați câte niveluri complete din arborele ID3 standard se pot calcula cu același efort de calcul — exprimat ca număr de apeluri la funcția IG — ca la stabilirea atributului rădăcină în varianta *2-step look-ahead*.

b4. Metoda de învățare a arborilor de decizie folosind *2-step look-ahead* crează o clasă de modele / ipoteze mai largă decât algoritmul ID3 simplu? Altfel spus, putem să calculăm în acest mod funcții de clasificare pe care nu le putem reprezenta cu arborii de decizie standard?

Răspuns:

a. Datorită presupunerii făcute pentru *simplitate*, arborele de decizie final va fi un arbore binar complet cu $A + 1$ niveluri (inclusiv și nodurile de decizie), notate în mod convențional de la 0 la A . Pe fiecare nivel $i = \overline{0, A - 1}$ al arborelui binar se găsesc 2^i noduri. În fiecare dintre aceste noduri poate fi ales un atribut din cele $A - i$ rămas disponibile. Deci pentru determinarea nivelului i se fac $2^i \cdot (A - i)$ apeluri ale funcției IG .

Desigur, pe penultimul nivel, $A - 1$, nu mai există decât un singur atribut rămas disponibil, prin urmare nu este necesar să se calculeze câștigul de informație. Așadar, numărul total de apeluri ale funcției IG pentru elaborarea întregului arbore de decizie este:

$$N_{IG} = A + 2(A - 1) + 4(A - 2) + \dots + 2^{A-2}(A - (A - 2)) = \sum_{i=0}^{A-2} 2^i(A - i)$$

b1. n_l și n_r reprezintă numărul de instanțe asignate nodurilor descendente din a_i , deci factorii $\frac{n_l}{n_l + n_r}$ și $\frac{n_r}{n_l + n_r}$ reprezintă ponderile acestor sub-mulțimi în raport cu reuniunea lor. Cei doi factori vor pondera corespunzător câștigurile de informație de pe cele două ramuri din arborele de decizie. Dacă nu facem astfel de ponderări, este posibil să avem un câștig de informație mare pe o mulțime mică sau invers. În consecință, simpla însumare

a celor două câștiguri de informație nu ar emula în mod veridic câștigul de informație pe întreg ansamblul lui D .

b2. Pentru stabilirea atributului din nodul rădăcină făcând o explorare cu 2 pași înainte, se calculează pentru fiecare dintre cele A atrbute câștigurile de informație corespunzătoare celor 2 descendenți, adică:

$$N_{IG_{2-step}}(\text{rădăcină}) = A \cdot 2(A - 1) = 2A^2 - 2A$$

b3. Dacă $A = 10$, atunci pentru stabilirea atributului rădăcină în varianta *2-step look-ahead* se apelează funcția IG de $N_{IG_{2-step}}(\text{rădăcină}) = 200 - 20 = 180$ de ori. Trebuie determinat numărul x de niveluri complete din arborele ID3 standard care se pot calcula folosind maxim 180 de apeluri ale funcției IG , adică argmax_x astfel încât $N_{IG}(x) = \sum_{i=0}^{x-1} 2^i(10 - i) \leq 180$.

$$\begin{aligned} x = 1 &\Rightarrow N_{IG}(1) = 2^0(10 - 0) = 10 \\ x = 2 &\Rightarrow N_{IG}(2) = 10 + 2^1(10 - 1) = 28 \\ x = 3 &\Rightarrow N_{IG}(3) = 28 + 2^2(10 - 2) = 60 \\ x = 4 &\Rightarrow N_{IG}(4) = 60 + 2^3(10 - 3) = 116 \\ x = 5 &\Rightarrow N_{IG}(5) = 116 + 2^4(10 - 4) = 212 > 180 \end{aligned}$$

Așadar, se pot calcula 4 niveluri complete din arborele ID3 standard cu același efort de calcul ca la stabilirea atributului din rădăcină în varianta (mai puțin “greedy”) a algoritmului ID3 cu *2-step look-ahead*.

b4. Nu. Clasa de modele / ipoteze pe care lucrează algoritmul ID3 cu *2-step look-ahead* este aceeași ca la algoritmul ID3 standard. Însă este foarte posibil ca arborele de decizie construit să fie mai bun dacă se face cătarea în maniera *2-step look-ahead*.

18.

(Îmbunătățirea algoritmului ID3, folosind IG cu “2-step look-ahead”)

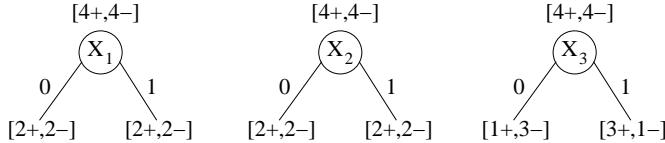
Liviu Ciortuz, folosind date de la CMU, 2008 fall, Eric Xing, HW2, pr. 3

Se consideră variabilele booleene X_1 , X_2 și X_3 , precum și clasificarea $Y = \{0, 1\}$. Fie setul de date de antrenament din tabelul de mai jos.

	X_1	X_2	X_3	Y
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	1	1
6	1	0	1	1
7	1	1	0	0
8	1	1	0	0

Răspuns:

a. Pentru nodul rădăcină vom alege unul dintre atrbutele X_1 , X_2 și X_3 . Corespunzător, vom obține următoarele partajări ale setului de date de antrenament:



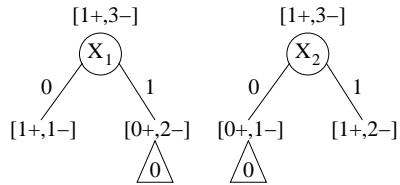
În cele ce urmează vom scrie câștigul de informație (IG) omitând primul argument, întrucât se consideră cunoscut din context. Vom calcula câștigul de informație corespunzător fiecărui atribut:

$$\begin{aligned} IG(X_1) = IG(X_2) &= H[4+, 4-] - \left(\frac{4}{8}H[2+, 2-] + \frac{4}{8}H[2+, 2-] \right) = 1 - \left(\frac{1}{2} + \frac{1}{2} \right) = 0 \\ IG(X_3) &= H[4+, 4-] - \left(\frac{4}{8}H[1+, 3-] + \frac{4}{8}H[3+, 1-] \right) \end{aligned}$$

Însă $H[1+, 3-] = H[3+, 1-]$, iar valoarea corespunzătoare este sub-unitară, deci $IG(X_3) > 0$. Așadar, în nodul rădăcină se alege atributul X_3 .

Pentru nodul corespunzător ramurii $X_3 = 0$ se poate alege dintre attributele care au mai rămas, adică X_1 sau X_2 .

Se calculează câștigurile de informație corespunzătoare:



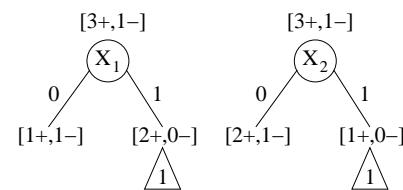
$$\begin{aligned} IG(X_1 | X_3 = 0) &= H[1+, 3-] - \left(\frac{2}{4}H[1+, 1-] + \frac{2}{4}H[0+, 2-] \right) \\ &= 2 - \frac{3}{4}\log_2 3 - \left(\frac{1}{2} + 0 \right) = \frac{3}{2} - \frac{3}{4}\log_2 3 = 0.311 \\ IG(X_2 | X_3 = 0) &= H[1+, 3-] - \left(\frac{1}{4}H[0+, 1-] + \frac{3}{4}H[1+, 2-] \right) \\ &= 2 - \frac{3}{4}\log_2 3 - \left(0 + \frac{3}{4}H[1+, 2-] \right) \\ &= 2 - \frac{3}{4}\log_2 3 - \frac{3}{4} \left(\frac{1}{3}\log_2 3 + \frac{2}{3}\log_2 \frac{3}{2} \right) \\ &= 2 - \frac{3}{4}\log_2 3 - \frac{3}{4} \left(\log_2 3 - \frac{2}{3} \right) = \frac{5}{2} - \frac{3}{2}\log_2 3 = 0.122 \end{aligned}$$

Cum $IG(X_1 | X_3 = 0) > IG(X_2 | X_3 = 0)$, se alege atributul X_1 .

În cele de mai sus, s-au folosit notațiile în maniera condițională $IG(X_1 | X_3 = 0)$ și $IG(X_2 | X_3 = 0)$ doar pentru a identifica în mod neambiguu care este nodul din arbore pentru care se calculează câștigul de informație respectiv.

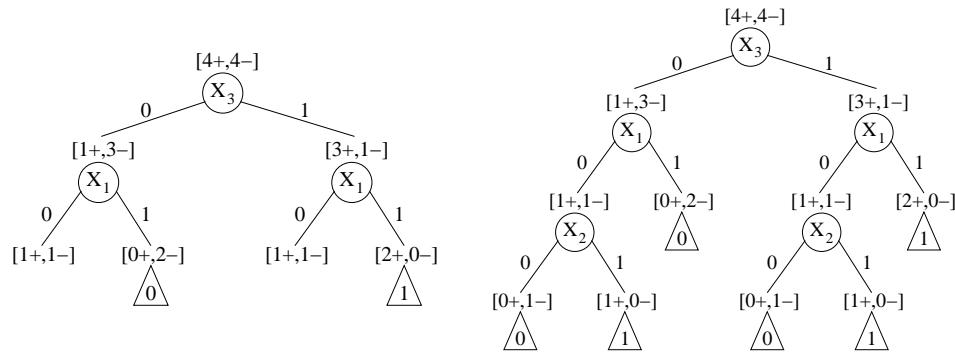
Pentru nodul corespunzător ramurii $X_3 = 1$ ce pornește din nodul rădăcină se poate alege din nou unul dintre attributele X_1 și X_2 .

Pentru aceasta, se compară câștigurile de informație corespunzătoare. Se observă că acestea sunt egale cu cele din cazul precedent. Așadar,



$$\begin{aligned} IG(X_1 \mid X_3 = 1) &= IG(X_1 \mid X_3 = 0) = \frac{3}{2} - \frac{3}{4} \log_2 3 = 0.311 \\ IG(X_2 \mid X_3 = 1) &= IG(X_2 \mid X_3 = 0) = \frac{5}{2} - \frac{3}{2} \log_2 3 = 0.122 \end{aligned}$$

Prin urmare, se alege tot atributul X_1 . Arborele de decizie construit până în acest moment este cel reprezentat mai jos în partea stângă:



În cele două noduri care au rămas, nu poate fi ales decât atributul X_2 . Deci arborele de decizie complet construit de algoritmul ID3 este cel reprezentat mai sus în partea dreaptă.

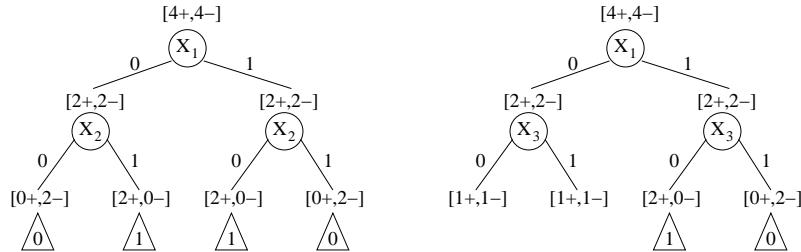
b. Dacă se folosește câștigul de informație cu *2-step look-ahead* pentru algoritmul ID3, atunci la alegerea fiecărui nod se iau în considerare [și] toate combinațiile posibile de noduri de pe nivelul următor, utilizându-se formula:

$$IG_{2\text{-step}}(D, a_i) = \max_{a_l, a_r} \left\{ \frac{n_l}{n_l + n_r} IG(D_l, a_l) + \frac{n_r}{n_l + n_r} IG(D_r, a_r) \right\}$$

Pentru nodul rădăcină se poate alege, la fel ca la punctul precedent, unul dintre atrbutele X_1 , X_2 și X_3 . Așadar, câștigul de informație corespunzător atributului X_1 va fi calculat astfel:

$$IG_{2\text{-step}}(X_1) = \max \left\{ \begin{array}{l} \frac{4}{8} IG(X_2 \mid X_1 = 0) + \frac{4}{8} IG(X_2 \mid X_1 = 1) \\ \frac{4}{8} IG(X_2 \mid X_1 = 0) + \frac{4}{8} IG(X_3 \mid X_1 = 1) \\ \frac{4}{8} IG(X_3 \mid X_1 = 0) + \frac{4}{8} IG(X_2 \mid X_1 = 1) \\ \frac{4}{8} IG(X_3 \mid X_1 = 0) + \frac{4}{8} IG(X_3 \mid X_1 = 1) \end{array} \right.$$

Pentru a determina această valoare, va trebui să calculăm cele 4 câștiguri de informație (în sens clasic) implicate în formulă, și este util să reprezentăm două din cele 4 situații posibile:



$$\begin{aligned}
 IG(X_2 | X_1 = 0) &= IG(X_2 | X_1 = 1) = IG(X_3 | X_1 = 1) \\
 &= H[2+, 2-] - \left(\frac{1}{2}H[0+, 2-] + \frac{1}{2}H[2+, 0-] \right) = 1 - 0 = 1 \\
 IG(X_3 | X_1 = 0) &= H[2+, 2-] - \left(\frac{1}{2}H[1+, 1-] + \frac{1}{2}H[1+, 1-] \right) \\
 &= 1 - \left(\frac{1}{2} + \frac{1}{2} \right) = 0
 \end{aligned}$$

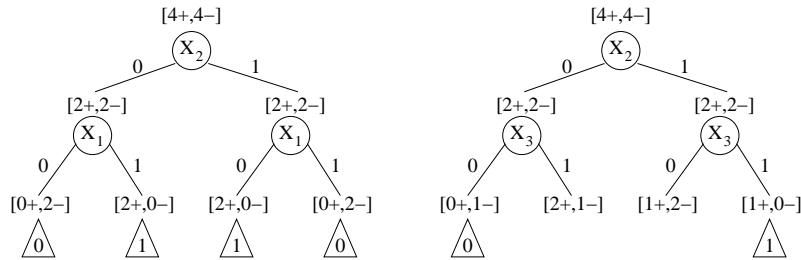
Prin urmare,

$$IG_{2\text{-step}}(X_1) = \max \begin{cases} \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 \\ \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 \\ \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 \\ \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 \end{cases} = \max \left\{ 1, 1, \frac{1}{2}, \frac{1}{2} \right\} = 1.$$

Câștigul de informație corespunzător atributului X_2 plasat în nodul rădăcină este:

$$IG_{2\text{-step}}(X_2) = \max \begin{cases} \frac{4}{8}IG(X_1 | X_2 = 0) + \frac{4}{8}IG(X_1 | X_2 = 1) \\ \frac{4}{8}IG(X_1 | X_2 = 0) + \frac{4}{8}IG(X_3 | X_2 = 1) \\ \frac{4}{8}IG(X_3 | X_2 = 0) + \frac{4}{8}IG(X_1 | X_2 = 1) \\ \frac{4}{8}IG(X_3 | X_2 = 0) + \frac{4}{8}IG(X_3 | X_2 = 1) \end{cases}$$

Vom reprezenta din nou două dintre situațiile posibile:



$$\begin{aligned}
IG(X_1 | X_2 = 0) &= IG(X_1 | X_2 = 1) = H[2+, 2-] - 0 = 1 \\
IG(X_3 | X_2 = 0) &= IG(X_3 | X_2 = 1) = H[2+, 2-] - \left(\frac{1}{4}H[0+, 1-] + \frac{3}{4}H[2+, 1-] \right) = \\
&= 1 - \frac{3}{4} \left(\log_2 3 - \frac{2}{3} \right) = \frac{3}{2} - \frac{3}{4} \log_2 3 = 0.311
\end{aligned}$$

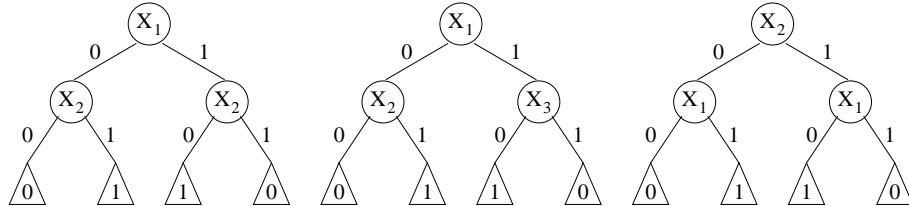
Prin urmare,

$$IG_{2\text{-step}}(X_2) = \max \begin{cases} \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 \\ \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0.311 \\ \frac{1}{2} \cdot 0.311 + \frac{1}{2} \cdot 1 \\ \frac{1}{2} \cdot 0.311 + \frac{1}{2} \cdot 0.311 \end{cases} = \max\{1, 0.655, 0.655, 0.311\} = 1$$

Pentru calculul câștigului de informație corespunzător atributului X_3 plasat în nodul rădăcină, au fost calculate la punctul a cele 4 câștiguri de informație în sens clasic implicate în formulă, deci rezultă:

$$\begin{aligned}
IG_{2\text{-step}}(X_3) &= \max \begin{cases} \frac{4}{8}IG(X_1 | X_3 = 0) + \frac{4}{8}IG(X_1 | X_3 = 1) \\ \frac{4}{8}IG(X_1 | X_3 = 0) + \frac{4}{8}IG(X_2 | X_3 = 1) \\ \frac{4}{8}IG(X_2 | X_3 = 0) + \frac{4}{8}IG(X_1 | X_3 = 1) \\ \frac{4}{8}IG(X_2 | X_3 = 0) + \frac{4}{8}IG(X_2 | X_3 = 1) \end{cases} \\
&= \max \begin{cases} \frac{1}{2} \left(\frac{3}{2} - \frac{3}{4} \log_2 3 \right) + \frac{1}{2} \left(\frac{3}{2} - \frac{3}{4} \log_2 3 \right) \\ \frac{1}{2} \left(\frac{3}{2} - \frac{3}{4} \log_2 3 \right) + \frac{1}{2} \left(\frac{5}{2} - \frac{3}{2} \log_2 3 \right) \\ \frac{1}{2} \left(\frac{5}{2} - \frac{3}{2} \log_2 3 \right) + \frac{1}{2} \left(\frac{3}{2} - \frac{3}{4} \log_2 3 \right) \\ \frac{1}{2} \left(\frac{5}{2} - \frac{3}{2} \log_2 3 \right) + \frac{1}{2} \left(\frac{5}{2} - \frac{3}{2} \log_2 3 \right) \end{cases} = \max \begin{cases} \frac{3}{2} - \frac{3}{4} \log_2 3 \\ 2 - \frac{9}{8} \log_2 3 \\ 2 - \frac{9}{8} \log_2 3 \\ \frac{5}{2} - \frac{3}{2} \log_2 3 \end{cases} \\
&= \max\{0.311, 0.216, 0.216, 0.122\} = 0.311
\end{aligned}$$

Comparând $IG_{2\text{-step}}(X_1)$, $IG_{2\text{-step}}(X_2)$ și $IG_{2\text{-step}}(X_3)$, obținem valoarea maximă 1 fie pentru X_1 , fie pentru X_2 în nodul rădăcină. Având în vedere calculele realizate pentru a obține aceste valori, nu mai sunt necesare operații suplimentare pentru determinarea arborelui de decizie construit de ID3 cu metoda *2-step look-ahead*. Există de fapt 3 arbori de decizie optimi (ca număr de niveluri și / sau noduri) pentru aceste date de antrenament, și anume:



Este important de remarcat faptul că algoritmul ID3 cu *2-step look-ahead* identifică toate aceste trei soluții optimale, în vreme ce algoritmul ID3 standard nu identifică niciuna dintre ele.

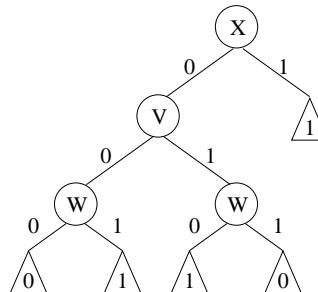
19.

(O strategie de pruning pentru arborele ID3:
eliminarea nodurilor cu $IG < \varepsilon$;
explorare top-down vs. bottom-up)

■ *prelucrare de Liviu Ciortuz, după CMU, 2006 spring, Carlos Guestrin, midterm exam, pr. 4*

Puteți constata ușor că, aplicând algoritmul ID3 pe datele din tabelul de mai jos, se va obține arborele de decizie alăturat.

V	W	X	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
1	1	1	1



- a. Pentru un astfel de arbore de decizie, o strategie simplă de trunchiere (engl., pruning) în vederea contracărării fenomenului de “overfitting” constă în a parcurge arborele de sus în jos, începând deci cu nodul-rădăcină și identificând fiecare nod de test pentru care câștigul de informație (sau un alt criteriu fixat în avans) are o valoare mai mică decât o valoare pozitivă, mică, fixată de la început, ε . Orice astfel de nod de test este imediat înlocuit — împreună cu subarborele corespunzător lui — cu un nod de decizie, conform etichetei majoritară a instanțelor asignate nodului de test. Această strategie se numește “top-down pruning”.

Care este arborele de decizie obținut aplicând această strategie pe arborele de mai sus, dacă se consideră $\varepsilon = 0.0001$? Care este eroarea la antrenare pentru noul arbore?

- b. O altă posibilitate de a face pruning este să parcurgem arborele de decizie începând cu părintii nodurilor-frunză și să eliminăm în mod recursiv acele noduri de test pentru care câștigul de informație (sau un alt criteriu ales) este mai mic decât ε . Aceasta este strategia de “bottom-up pruning”.

Observație: Spre deosebire de strategia top-down, în varianta de pruning de tip bottom-up nu vor fi eliminate noduri (cu $IG < \varepsilon$) pentru care există descendenți al căror câștig de informație este mai mare sau egal cu ε .

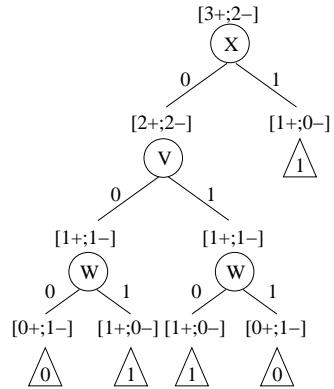
Ce arbore se obține făcând “bottom-up pruning” pe arborele dat mai sus, dacă se consideră $\varepsilon = 0.0001$? Care este eroarea la antrenare pentru arborele rezultat?

c. Stabiliti în ce situații ar fi indicat să alegem strategia “bottom-up pruning” în loc de “top-down pruning” și viceversa. Comparați acuratețea la antrenare și complexitatea computațională a celor două strategii de pruning.

d. Cât este înălțimea — adică, numărul de niveluri de test — pentru arborele returnat de ID3 urmat de “bottom-up pruning”? Puteți găsi un arbore de decizie având o înălțime mai mică, dar care clasifică perfect setul de antrenament? Ce concluzie putem trage despre calitatea [outputului] algoritmului ID3?

Răspuns:

Înainte de a rezolva efectiv punctele a și b , vom augmenta arborele de decizie dat cu informațiile referitoare la numărul de instanțe (pozitive și, respectiv, negative) asignate fiecărui nod de test. Obținem astfel figura alăturată.



a. Câștigul de informație al atributului X plasat în nodul-rădăcină este:

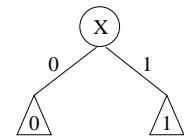
$$H[3+; 2-] - 1/5 \cdot 0 - 4/5 \cdot 1 = 0.971 - 0.8 = 0.171 > \varepsilon.$$

Prin urmare, acest nod nu va fi eliminat din arbore.

Câștigul de informație al atributului V este:

$$H[2+; 2-] - 1/2 \cdot 1 - 1/2 \cdot 1 = 1 - 1 = 0 < \varepsilon.$$

Așadar, nodul reprezentat de atributul V va fi eliminat și vom obține arborele de decizie [trunchiat], reprezentat în figura alăturată. Menționăm că am fi putut la fel de bine (din punctul de vedere al numărului de erori la antrenare) să alegem decizia $Y = 1$ în nodul-fiu stâng, însă în acel caz arborele s-ar fi redus de fapt la un singur nod de decizie (cu outputul $Y = 1$). Eroarea la antrenare produsă de acest arbore este $2/5$.



b. Câștigul de informație al celor două noduri marcate cu atributul W în arborele dat în enunț este același, și anume 1 (se poate verifica imediat). Prin urmare, aplicând strategia de “bottom-up pruning”, arborele de decizie rămâne identic cu cel inițial. Evident, eroarea la antrenare pentru acest arbore este 0, întrucât datele de antrenament nu conțin inconistențe.

c. Din cauza faptului că la top-down pruning, odată cu un nod de test pentru care câștigul de informație (IG) este mai mic decât valoarea ε se elimină întregul subarbore care are ca rădăcină acel nod de test, această strategie este mai rapidă decât (sau, în cel mai rău caz, la fel de rapidă / lentă ca și) pruning-ul de tip bottom-up.

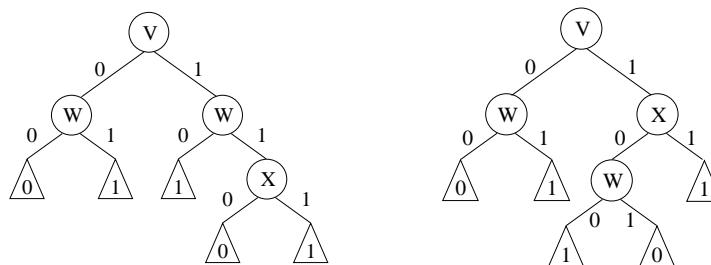
Așa cum s-a menționat în *observația* din enunț și s-a exemplificat apoi la punctele *a* și *b*, dezavantajul pruning-ului de tip top-down este că odată ce este eliminat un nod cu IG mai mic decât ε , este posibil ca între descendenții săi (eliminați) să fie și noduri care au câștigul de informație mai mare sau egal cu ε . Pruning-ul bottom-up nu are acest dezavantaj; el este deci mai „conservativ“.

O *problemă* care poate însă să apară și în cazul pruning-ului de tip bottom-up este faptul că în nodurile apropiate de nodurile de decizie (acestea din urmă fiind nodurile-frunză), câștigul de informație se calculează uneori pe mulțimi mici de exemple, deci testul $IG \geq \varepsilon$ nu este neapărat semnificativ din punct de vedere statistic. Așadar, este *recomandabil* ca în astfel de situații să se folosească un test statistic, de exemplu *testul χ^2* . (A se vedea problemele 20 și 51.)

În ce privește comparația dintre acuratețile arborilor obținuți prin aplicarea celor două variante de pruning: se observă că arboarele mai simplu (cel de la punctul *a*) are o eroare la antrenare mai mare decât arboarele mai complex (cel de la punctul *b*), însă este mai probabil ca acesta din urmă să producă overfitting.

d. Din rezolvarea dată la punctul *b*, rezultă imediat că înălțimea arborelui obținut prin aplicarea algoritmului ID3 urmat de pruning de tip bottom-up este 3. (Nu se iau în considerare nodurile frunză.) Vom demonstra — exact ca la problema 1 — că nu există un arbore de decizie consistent cu datele de antrenament, care să aibă adâncimea strict mai mică decât 3:

Se observă ușor din tabelul dat în enunț că $Y = (V \text{ XOR } W) \vee X$, așadar variabilele V și W au rol simetric în definirea funcției reprezentate de variabila Y . Punând atributul X în nodul rădăcină, arboarele de decizie minimal care se poate obține este cel dat în enunț (sau, echivalent, arboarele care obține din acesta interschimbând V și W). Dacă, în schimb, punem atributul V în nodul rădăcină, se pot obține doi arbori de decizie minimali, așa cum se arată grafic mai jos. (Și similar, dacă în nodul rădăcină punem atributul W .)



20.

(ID3 cu post-pruning: folosirea testului statistic χ^2 pentru limitarea overfitting-ului)

■ prelucrare de Liviu Ciortuz, după
CMU, 2010 fall, Ziv Bar-Joseph, HW2, pr. 2.1

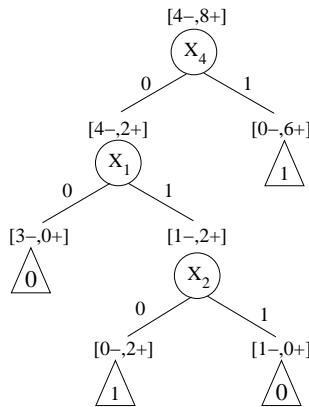
În acest exercițiu veți face pruning asupra unui arbore ID3 (după ce s-a făcut antrenarea pe toate datele disponibile), folosind o metodă statistică de testare / verificare a ipotezelor.

După ce a fost învățat arborele de decizie, vizităm fiecare nod intern (inclusiv nodul rădăcină) și testăm dacă atributul care a fost pus în nodul respectiv nu este cumva necorelat cu eticheta / clasa specificată de către atributul de ieșire.

Pentru aceasta, mai întâi presupunem că atributul din nodul respectiv este independent de atributul de ieșire (aceasta este aşa-numita „ipoteză nulă“), iar apoi folosim testul χ^2 al lui Pearson pentru a genera o „statistică“, care poate constitui temeiul pentru respingerea „ipotezei nule“. Dacă ipoteza nulă nu poate fi respinsă, eliminăm sub-arborele din nodul respectiv (de fapt, îl înlocuim cu un nod de decizie).

Pentru a ilustra aceste chestiuni, considerăm arborele de decizie de mai jos (partea dreaptă); el a fost construit pornind de la datele din tabelul alăturat lui, folosind algoritmul ID3.

X_1	X_2	X_3	X_4	<i>Class</i>
1	1	0	0	0
1	0	1	0	1
0	1	0	0	0
1	0	1	1	1
0	1	1	1	1
0	0	1	0	0
1	0	0	0	1
0	1	0	1	1
1	0	0	1	1
1	1	0	1	1
1	1	1	1	1
0	0	0	0	0



a. Pentru fiecare nod intern din arborele de decizie vom crea o *tabelă de contingență* (engl., contingency table) pentru exemplele de antrenare care sunt asignate nodului respectiv. Tabela aceasta va avea coloanele etichetate cu cele c clase / valori ale variabilei de ieșire. Similar, valorile atributului testat în nodul respectiv (având în total r valori) vor fi asignate liniilor tabelei. Dacă acceptăm o ușoară simplificare în forma de exprimare, vom putea spune că un element oarecare $O_{i,j}$ din tabela de contingență reprezintă numărul de „observații“ (adică, instanțe de antrenament asignate nodului respectiv) pentru care valoarea atributului testat este i , iar eticheta / clasa este j .

Calculați tabelele de contingență pentru cele trei noduri de test ale arborelui de decizie dat mai sus. Apoi, pornind de la datele conținute în fiecare dintre aceste matrice de contingență, estimați (în sensul verosimilității maxime) probabilitățile pentru valorile variabilei de ieșire (*Class*), precum și pentru valorile atributului din nodul corespunzător.

b. Pentru a aplica testul statistic χ^2 , avem nevoie să calculăm pentru fiecare nod intern al arborelui de decizie încă o tabelă (pe care o vom nota cu E), în care să consemnăm *numărul așteptat* de apariții (engl., expected counts) ale instanțelor de antrenament la nodul respectiv, pentru fiecare pereche de indici i, j având semnificația de mai sus. Acest număr așteptat este numărul (mediu) de instanțe de antrenament pe care le-am „observat“ în nodul respectiv dacă atributul selectat și clasa (variabila de ieșire) ar fi independente.

Derivați o formulă pentru calculul fiecărui element (notat $E_{i,j}$) din această a doua tabelă.

Întrebări ajutătoare: Care este probabilitatea ca exemplele de antrenament asignate nodului respectiv să aibă o anumită etichetă (j)? Înținând cont de această probabilitate,

precum și de presupoziția de independență formulată prin ipoteza „nulă“, care este numărul de exemple cu o anumită valoare (i) pentru atributul selectat în nodul respectiv, care ar trebui (i.e., „ne aşteptăm“) să aibă acea etichetă / clasă (j)?

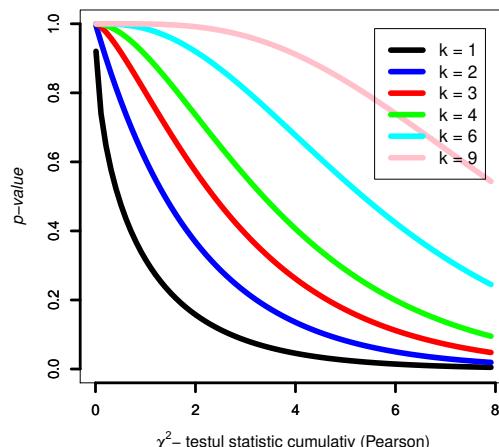
Folosind formula pe care tocmai ati derivat-o, calculati matricea E pentru fiecare dintre cele trei noduri de test ale arborelui de decizie dat mai sus.

c. Date fiind cele două tabele pentru nodul considerat, puteți calcula acum testul statistic χ^2 -pătrat:

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

Puteți introduce valoarea calculată χ^2 precum și numărul de grade de libertate $(r-1)(c-1)$ într-un program¹⁸⁹ sau într-un calculator on-line¹⁹⁰ pentru a calcula o așa-numită p -valoare (engl., *p-value*).¹⁹¹

În general, dacă $p < 0.05$, se va considera că nu avem suficientă evidență în favoarea ipotezei „nule“, care afirma că atributul selectat și clasa (variabila de ieșire) sunt independente, și o vom respinge. Într-o astfel de situație, spunem că testul [din nodul respectiv] este *semnificativ* din punct de vedere statistic.



Pentru fiecare dintre cele trei noduri interne din arborele de decizie dat mai sus, găsiți p -valoarea corespunzătoare și precizați dacă testul din nodul respectiv este sau nu semnificativ d.p.v. statistic. Cât de multe noduri interne vor fi eliminate din arbore dacă la pruning impunem condiția $p \geq 0.05$ [pentru a elimina un nod de test din arbore și a-l înlocui cu un nod de decizie]?

Răspuns:

¹⁸⁹Folosiți `1-chi2cdf(x,df)` în MATLAB sau `CHIDIST(x,df)` în Excel.

¹⁹⁰<http://faculty.vassar.edu/lowry/tabs.html#csq> este un astfel de calculator.

¹⁹¹Statistica χ^2 este aproximată de distribuția χ^2 cu numărul corespunzător (k) de grade de libertate. (Distribuția χ^2 reprezintă suma pătratelor a k variabile gaussiene standard independente.)

p -valoarea despre care este vorba mai sus reprezintă probabilitatea ca distribuția χ^2 să ia valori mai mari sau egale cu valoarea considerată (i.e., valoarea calculată pentru statistică χ^2). Așadar, p -valoarea pentru testul χ^2 se calculează făcând diferența dintre 1 și valoarea funcției de distribuție cumulative (c.d.f.) pentru distribuția χ^2 cu k de grade de libertate. Vedeti site-ul https://en.m.wikipedia.org/wiki/Chi-square_distribution#Table_of_Chi-square_distributions (accesat la 5.09.2015).

a. Pentru fiecare nod din arborele ID3 vom alcătui matricea de contingență asociată, pornind de la partitioanările multimilor de exemple care au fost asignate (de către algoritm ID3) descendenților nodului respectiv. Apoi, din fiecare matrice de contingență vom estima (în sensul verosimilității maxime) probabilitățile pentru valorile variabilei de ieșire (*Class*), precum și probabilitățile pentru valorile atributului din acel nod. (Atenție la condiționarea probabilităților!)

$$\begin{array}{c|cc} O_{X_4} & \text{Class} = 0 & \text{Class} = 1 \\ \hline X_4 = 0 & 4 & 2 \\ X_4 = 1 & 0 & 6 \end{array} \xrightarrow{N=12} \begin{cases} P(X_4 = 0) = \frac{6}{12} = \frac{1}{2}, P(X_4 = 1) = \frac{1}{2} \\ P(\text{Class} = 0) = \frac{4}{12} = \frac{1}{3}, P(\text{Class} = 1) = \frac{2}{3} \end{cases}$$

$$\begin{array}{c|cc} O_{X_1|X_4=0} & \text{Class} = 0 & \text{Class} = 1 \\ \hline X_1 = 0 & 3 & 0 \\ X_1 = 1 & 1 & 2 \end{array} \xrightarrow{N=6} \begin{cases} P(X_1 = 0 | X_4 = 0) = \frac{3}{6} = \frac{1}{2} \\ P(X_1 = 1 | X_4 = 0) = \frac{1}{2} \\ P(\text{Class} = 0 | X_4 = 0) = \frac{4}{6} = \frac{2}{3} \\ P(\text{Class} = 1 | X_4 = 0) = \frac{1}{3} \end{cases}$$

$$\begin{array}{c|cc} O_{X_2|X_4=0, X_1=1} & \text{Class} = 0 & \text{Class} = 1 \\ \hline X_2 = 0 & 0 & 2 \\ X_2 = 1 & 1 & 0 \end{array} \xrightarrow{N=3} \begin{cases} P(X_2 = 0 | X_4 = 0, X_1 = 1) = \frac{2}{3} \\ P(X_2 = 1 | X_4 = 0, X_1 = 1) = \frac{1}{3} \\ P(\text{Class} = 0 | X_4 = 0, X_1 = 1) = \frac{1}{3} \\ P(\text{Class} = 1 | X_4 = 0, X_1 = 1) = \frac{2}{3} \end{cases}$$

b. Considerăm i o valoare arbitrar aleasă (dar fixată) pentru atributul de intrare A care este testat în nodul curent, iar j o valoare arbitrar aleasă (de asemenea, fixată) pentru atributul de ieșire *Class* (renotat cu C). Înținând cont de presupozitia de independentă stipulată de ipoteza „nulă“, putem scrie:

$$P(A = i, C = j) = P(A = i) \cdot P(C = j)$$

Probabilitățile $P(A = i)$ și $P(C = j)$ pot fi estimate — în sensul verosimilității maxime (MLE) —, cu ajutorul celor N instanțe de antrenament asignate nodului respectiv. Instanțele pentru care atributul A are valoarea i sunt tocmai cele din linia i a matricei. În mod similar, instanțele care au eticheta / clasa j sunt cele din coloana j a matricei de count-uri observate. Așadar,

$$P(A = i) = \frac{\sum_{k=1}^c O_{i,k}}{N} \text{ și } P(C = j) = \frac{\sum_{k=1}^r O_{k,j}}{N}$$

În consecință,

$$P(A = i, C = j) = \frac{(\sum_{k=1}^c O_{i,k}) (\sum_{k=1}^r O_{k,j})}{N^2},$$

iar valoarea așteptată — repetăm, în condițiile presupozitiei de independentă — pentru numărul de instanțe având atributul $A = i$ și clasa $C = j$ va fi dată de formula

$$E_{i,j} = N \cdot P(A = i, C = j) = \frac{(\sum_{k=1}^c O_{i,k}) (\sum_{k=1}^r O_{k,j})}{N}$$

c. Folosind probabilitățile calculate la punctul precedent și ținând cont de presupozitia de independentă, calculăm numărul de observații așteptate în fiecare nod, pentru a completa matricele E :

E_{X_4}	$Class = 0$	$Class = 1$	$E_{X_1 X_4=0}$	$Class = 0$	$Class = 1$
$X_4 = 0$	2	4	$X_1 = 0$	2	1
$X_4 = 1$	2	4	$X_1 = 1$	2	1
$E_{X_2 X_4=0, X_1=1}$	$Class = 0$	$Class = 1$			
$X_2 = 0$	$\frac{2}{3}$	$\frac{4}{3}$			
$X_2 = 1$	$\frac{1}{3}$	$\frac{2}{3}$			

Ca să exemplificăm cum am procedat, detaliem mai jos calculul pentru primul element din matricea E_{X_4} :

$$N = 12, P(X_4 = 0) = \frac{1}{2} \text{ și } P(Class = 0) = \frac{1}{3} \Rightarrow$$

$$N \cdot P(X_4 = 0, Class = 0) = N \cdot P(X_4 = 0) \cdot P(Class = 0) = 12 \cdot \frac{1}{2} \cdot \frac{1}{3} = 2$$

Acum putem aplica pentru fiecare nod din arborele de decizie formula de calcul a valorilor / statisticilor χ^2 care ne-a fost dată în enunț:

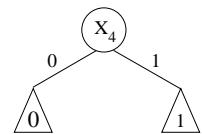
$$\chi^2_{x_4} = \frac{(4-2)^2}{2} + \frac{(2-4)^2}{4} + \frac{(0-2)^2}{2} + \frac{(6-4)^2}{4} = 2+2+1+1=6$$

$$\chi^2_{x_1|x_4=0} = \frac{(3-2)^2}{2} + \frac{(0-1)^2}{1} + \frac{(1-2)^2}{2} + \frac{(2-1)^2}{1} = 3$$

$$\chi^2_{x_2|x_4=0, x_1=1} = \frac{\left(0-\frac{2}{3}\right)^2}{\frac{2}{3}} + \frac{\left(2-\frac{4}{3}\right)^2}{\frac{4}{3}} + \frac{\left(1-\frac{1}{3}\right)^2}{\frac{1}{3}} + \frac{\left(0-\frac{2}{3}\right)^2}{\frac{2}{3}} = \frac{4}{9} \cdot \frac{27}{4} = 3$$

Accesând pagina web indicată în enunț, am obținut p -valorile următoare: 0.0143, 0.0833 și 0.0833.

În consecință, cu un grad de încredere de cel puțin 95%, nodurile situate pe nivelurile 1 și 2 din arborele ID3 pot fi eliminate. Pentru nodul rădăcină, ipoteza „nulă“ nu se verifică, adică variabilele X_4 și $Class$ nu sunt independente. Arborele obținut în urma pruning-ului este cel din figura alăturată.



Observație: Este de remarcat faptul că arborele ID3 furnizat în enunț are (întâmplător) atât pentru atributul X_4 (din nodul rădăcină) cât și pentru nodul X_1 (de pe primul nivel)

același câștig de informație, 0.4591. În urma testului χ^2 , se va elimina însă doar nodul care-l conține pe X_1 . Valoarea statisticii χ^2 asociate nodului X_1 este mult mai mică (3, față de 6, cât este pentru nodul care-l conține pe X_4), deci vom avea suficientă „evidență“ pentru a trage concluzia, cu un grad de încredere de 95%, că variabila de ieșire, *Class*, și $X_1|X_4 = 0$ sunt independente.¹⁹² Dacă am fi aplicat o metodă de pruning bazată pe câștigul de informație, aceasta n-ar fi putut să trateze în mod diferit cele două noduri, adică să-l păstreze pe unul și să-l eliminate pe celălalt.

21. (Adevărat sau Fals?)

a.

Liviu Ciortuz

Algoritmul ID3 garantează obținerea arborelui de decizie optimal (ca număr de niveluri sau de noduri).

CMU, 2002 spring, A. Moore, midterm example questions, pr. 1.c

b. Întrucât arborii de decizie pot învăța să clasifice instanțe într-un număr discret de clase (deci nu învăță funcții cu valori reale), este imposibil ca ei să manifeste fenomenul de overfitting.

CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm, pr. 1.4

c. Fie A și B doi algoritmi de clasificare automată. Algoritmul A este mai bun decât algoritmul B dacă eroarea la antrenare a algoritmului A este mai mică decât eroarea la antrenare a algoritmului B . Justificați.

CMU, 2005 spring, C. Guestrin, T. Mitchell, midterm, pr. 2.c

d. Presupunem că avem m instanțe și că vom folosi jumătate dintre ele pentru antrenarea unui clasificator oarecare (nu neapărat ID3) și jumătate pentru testare. Diferența dintre eroarea la antrenare și eroarea la testare descrește pe măsură ce numărul m crește.

Răspuns:

a. Fals. ID3 nu garantează obținerea arborelui optim (relativ la numărul de niveluri și / sau noduri), ci încearcă să găsească o soluție convenabilă, însă fără să caute în mod exhaustiv în tot spațiul soluțiilor. Mai exact, căutarea soluției se face în manieră “greedy”, maximizând un anumit criteriu (e.g., câștigul de informație) la fiecare iterație. O astfel de căutare nu garantează obținerea optimului.

b. Fals. Arborii de decizie manifestă fenomenul de overfitting. Prima parte a afirmației din enunț este adevarată — arborii de decizie pot învăța să clasifice instanțe într-un număr discret de clase —, însă a doua parte este falsă, deci avem o implicație de forma: $T \rightarrow F \equiv \neg T \vee F \equiv F$.

c. Fals, fiindcă la testare algoritmul B poate să aibă o eroare mai mică decât algoritmul A . Într-un astfel de caz, se spune că algoritmul A este “overfit” (rom., supra-antrenat).

d. Adevărat. Pe măsură ce dispunem de tot mai multe date de antrenament, dacă datele de antrenament sunt inconsistente, eroarea la antrenare va crește, fiindcă va fi din ce în ce mai greu ca modelul învățat să se adapteze la „zgomotele“ din date. Similar, eroarea la testare va descrește, fiindcă producem un clasificator care este din ce în ce mai puțin

¹⁹²Observați și faptul că nodul care conține atributul X_1 are mai puține instanțe asociate (și anume, jumătate) față de cele asociate nodului rădăcină.

afectat de “overfitting” pe datele de antrenament. Cele două erori vor converge la aşa-numita *eroare adevărată* (engl., true error) fiindcă diferențele statistice dintre datele de antrenament și datele de testare vor dispărea.

4.1.2 Algoritmul AdaBoost

22.

(Algoritmul AdaBoost: formulare; demonstarea unor relații de bază)

prelucrare de Liviu Ciortuz, după

■ CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1
CMU, 2009 fall, Carlos Guestrin, HW2, pr. 3.1

Fie o mulțime de m exemple de antrenament, $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, cu $x_i \in \mathcal{X}$ (unde \mathcal{X} poate fi de exemplu \mathbb{R}^d , cu $d \in \mathbb{N}^*$) și $y_i \in \{-1, 1\}$ pentru $i = 1, \dots, m$.

Presupunem că dispunem de un algoritm A care este un *clasificator* automat „slab“ (engl., weak classifier). A primește ca *input* o distribuție probabilistă oarecare D definită peste mulțimea S de exemple de antrenament și produce ca *output* o ipoteză $h : \mathcal{X} \rightarrow \{-1, 1\}$. Faptul că algoritmul A este un clasificator „slab“ înseamnă că orice astfel de ipoteză h produsă de el este doar cu puțin mai bună decât ghicirea / alegerea aleatorie (engl., random guessing).

Algoritmul AdaBoost (Y. Freund, R. Shapire, 1997) este un algoritm iterativ care livrează ca *output* [un clasificator bazat pe] o *combinație liniară de ipoteze* care sunt produse de către clasificatorul „slab“ A , câte una la fiecare iterație. Concret, AdaBoost lucrează astfel:

- Inițial (adică pentru $t = 1$), se folosește distribuția uniformă $D_1(i) = \frac{1}{m}, i = 1, \dots, m$.
- La fiecare iterație $t = 1, \dots, T$,
 - folosind distribuția probabilistă D_t , rulează algoritmul „slab“ de clasificare A pe setul de exemple de antrenament S , obținând ipoteza h_t ;
 - calculează *eroarea ponderată la antrenare* produsă de ipoteza h_t ,

$$\varepsilon_t \stackrel{\text{not.}}{=} \text{err}_{D_t}(h_t) \stackrel{\text{def.}}{=} \Pr_{D_t}(\{x|y \neq h_t(x)\}) \quad (79)$$

iar după aceea,¹⁹³ ponderea / „votul“

$$\alpha_t \stackrel{\text{not.}}{=} \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} \quad (80)$$

asociat ipotezei h_t ;¹⁹⁴

¹⁹³Dacă la o iterație $t \leq T$ clasificatorul slab A nu poate produce nicio ipoteză mai bună decât alegerea aleatorie (altfel spus, $\varepsilon_t = 1/2$) sau dacă, dimpotrivă, ipoteza h_t este perfectă (adică $\varepsilon_t = 0$), atunci algoritmul AdaBoost trebuie oprit.

¹⁹⁴Am notat cu $\Pr_{D_t}(E)$ probabilitatea lui E (eveniment aleatoriu oarecare) în raport cu distribuția probabilistă D_t . Remarcați faptul că din $\varepsilon_t \in (0, 1/2)$ rezultă $(1 - \varepsilon_t)/\varepsilon_t > 1$, deci $\alpha_t > 0$.

- definește o nouă distribuție probabilistă D_{t+1} , folosind „regula de actualizare“ (engl., update rule)

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t y_i h_t(x_i)}, \text{ pentru } i = 1, \dots, m \text{ și } \forall t \geq 1, \quad (81)$$

unde Z_t este factorul de normalizare.

- În final, algoritmul AdaBoost va livra — ca ipoteză învățată — funcția $H_T \stackrel{\text{def.}}{=} \text{sign}\left(\sum_{t=1}^T \alpha_t h_t\right)$; ea va acționa asupra instanțelor de test x conform [principiului] *votului ponderat majoritar* (engl., weighted majority vote).

Observație importantă: Această formulare a algoritmului AdaBoost nu impune nicio restricție asupra ipotezei h_t furnizate de către clasificatorul slab A la iterată t , cu excepția condiției $\varepsilon_t < 1/2$. Totuși, într-o formulare ulterioară a algoritmului AdaBoost, ca în problema 28, dar și într-un cadru mai extins, ca în problema 29, se poate cere / recomanda în mod explicit¹⁹⁵ ca ipoteza h_t să fie aleasă *minimizând* (eventual aproximativ) criteriul erorii ponderate la antrenare pe o întreagă clasă de ipoteze (separatori decizionali). Astfel de ipoteze pot fi, de exemplu, arborii de decizie de adâncime 1 (compași de decizie; engl., decision stumps). În mod implicit la toate exercițiile unde vom exemplifica aplicarea algoritmului AdaBoost, *recomandarea* aceasta va fi aplicată.

Demonstrați următoarele relații:

- i. $Z_t = e^{-\alpha_t} \cdot (1 - \varepsilon_t) + e^{\alpha_t} \cdot \varepsilon_t$ (consecință din relația (81))
- ii. $Z_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$ (consecință din i și din relația (80))
- iii. $0 < Z_t < 1$ (consecință din ii.)
- iv. $D_{t+1}(i) = \begin{cases} \frac{D_t(i)}{2\varepsilon_t}, & i \in M \stackrel{\text{not.}}{=} \{i | y_i \neq h_t(x_i)\} \\ \frac{D_t(i)}{2(1 - \varepsilon_t)}, & i \in C \stackrel{\text{not.}}{=} \{i | y_i = h_t(x_i)\} \end{cases}$
(consecință din relația (81) și ii.)
- v. $\varepsilon_i > \varepsilon_j \Rightarrow \alpha_i < \alpha_j$ (consecință din relația (80))
- vi. din relația (79) rezultă $\text{err}_{D_{t+1}}(h_t) = \frac{1}{Z_t} \cdot e^{\alpha_t} \cdot \varepsilon_t$,
unde $\text{err}_{D_{t+1}}(h_t) \stackrel{\text{not.}}{=} \Pr_{D_{t+1}}(\{x_i | h_t(x_i) \neq y_i\})$
- vii. $\text{err}_{D_{t+1}}(h_t) = 1/2$ (consecință din ii. și vi.)

Observații:

1. Vom explica acum de ce anume se folosește regula de actualizare a distribuțiilor D_t sub forma dată (81). Dubla inegalitate $0 < \varepsilon_t < \frac{1}{2}$ implică $\alpha_t > 0$ și, prin urmare, rezultă că $e^{\alpha_t} > 1$, iar $e^{-\alpha_t} < 1$. Pentru instanțe corect clasificate de către ipoteza h_t , vom avea $y_i h_t(x_i) = 1$, iar pentru instanțe incorect clasificate $y_i h_t(x_i) = -1$. Rezultă că la calculul distribuției D_{t+1} , algoritmul AdaBoost mărește probabilitatea alocată instanțelor incorect clasificate și diminuează probabilitatea alocată instanțelor corect clasificate. Faptul că $\text{err}_{D_{t+1}}(h_t) = 1/2$ (egalitate care trebuie demonstrată la punctul vii.) înseamnă că

¹⁹⁵Vedeți de exemplu MIT, 2006 fall, Tommi Jaakkola, HW4, problema 3.

algoritmul ia exact „masa“ de probabilitate $\gamma_t \stackrel{not.}{=} \frac{1}{2} - \varepsilon_t$ (care reprezintă cu cât este mai bună ipoteza h_t decât alegerea aleatorie) de la mulțimea de instanțe care au fost corect clasificate de către ipoteza h_t și o distribuie în mod proporțional la mulțimea de instanțe incorect clasificate de către același h_t , pentru ca algoritmul slab A să se concentreze ulterior asupra „învățării“ unui model (sau a unei ipoteze) care să clasifice corect și aceste instanțe. Termenul de *boosting adaptiv* — din care a derivat numele algoritmului AdaBoost — corespunde exact acestei strategii, care este pusă de lucru prin creșterea probabilităților asociate instanțelor incorect clasificate.

2. Prin ipoteză, clasificatorul slab A produce pentru distribuția D_{t+1} o ipoteză (h_{t+1}) având eroarea ε_{t+1} strict mai mică decât $1/2$. Prin urmare, rezultatul $err_{D_{t+1}}(h_t) = 1/2$ (care va fi demonstrat la acest punct) va avea drept consecință faptul că în mod cert ipoteza h_t nu va fi [produsă și, deci, nici] aleasă la iterată $t + 1$.

Răspuns:

i. Întrucât Z_t este factorul de normalizare în scrierea probabilităților $D_{t+1}(i)$, conform relației (81) putem scrie:¹⁹⁶

$$\begin{aligned} Z_t &= \sum_{i=1}^m D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)} = \sum_{i \in C} D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)} + \sum_{i \in M} D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)} \\ &= \sum_{i \in C} D_t(i) \cdot e^{-\alpha_t} + \sum_{i \in M} D_t(i) \cdot e^{\alpha_t} = e^{-\alpha_t} \cdot \underbrace{\sum_{i \in C} D_t(i)}_{1-\varepsilon_t} + e^{\alpha_t} \cdot \underbrace{\sum_{i \in M} D_t(i)}_{\varepsilon_t} \\ &= e^{-\alpha_t} \cdot (1 - \varepsilon_t) + e^{\alpha_t} \cdot \varepsilon_t. \end{aligned} \quad (82)$$

ii. Înținând cont de relația de definiție dată pentru ponderea α_t în enunț, vom avea:

$$e^{\alpha_t} = e^{\frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}} = e^{\ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}} = \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}, \quad (83)$$

ceea ce implică

$$e^{-\alpha_t} = \frac{1}{e^{\alpha_t}} = \sqrt{\frac{\varepsilon_t}{1 - \varepsilon_t}}. \quad (84)$$

În consecință, folosind relația (82), factorul de normalizare Z_t poate fi exprimat în funcție de ε_t (eroarea ponderată comisă la antrenare de către ipoteza h_t în raport cu distribuția D_t) astfel:

$$Z_t = \sqrt{\frac{\varepsilon_t}{1 - \varepsilon_t}} \cdot (1 - \varepsilon_t) + \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \cdot \varepsilon_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}. \quad (85)$$

Observație: La punctul d vom arăta că valoarea $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$ este punct de minim pentru expresia $e^{-\alpha_t} \cdot (1 - \varepsilon_t) + e^{\alpha_t} \cdot \varepsilon_t$, dacă aceasta este văzută ca funcție de argumentul α_t .

¹⁹⁶Vom uza de faptul (specificat în enunț la proprietatea iv.) că s-a notat cu C mulțimea indicilor acelor exemple care sunt corect clasificate la iterată t (adică, $C = \{i : y_i h_t(x_i) \geq 0\}$) și cu M mulțimea indicilor acelor exemple care sunt incorect clasificate la aceeași iterată t (adică, $M = \{i : y_i h_t(x_i) < 0\}$).

iii. Pe baza cunoștințelor de analiză matematică de liceu, deducem imediat că maximul funcției de gradul al doilea $\varepsilon_t(1-\varepsilon_t)$ se atinge pentru $\varepsilon_t = 1/2$, iar valoarea acestui maxim este $1/4$. Înădă cont că $\varepsilon_t \in (0, 1/2)$, din relația (85) rezultă că $Z_t > 0$ și $Z_t < 2\sqrt{\frac{1}{4}} = 1$.

iv. Pe baza relației (81), putem scrie imediat:

$$D_{t+1}(i) = \frac{1}{Z_t} \cdot D_t(i) \cdot \begin{cases} e^{\alpha_t}, & \text{pentru } i \in M \\ e^{-\alpha_t}, & \text{pentru } i \in C. \end{cases}$$

Prin urmare,

$$\begin{aligned} i \in M \Rightarrow D_{t+1}(i) &= \frac{1}{Z_t} \cdot D_t(i) \cdot e^{\alpha_t} \stackrel{(83)}{=} \frac{1}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}} \cdot D_t(i) \cdot \frac{\sqrt{1-\varepsilon_t}}{\sqrt{\varepsilon_t}} = \frac{D_t(i)}{2\varepsilon_t} \\ i \in C \Rightarrow D_{t+1}(i) &= \frac{1}{Z_t} \cdot D_t(i) \cdot e^{-\alpha_t} \stackrel{(84)}{=} \frac{1}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}} \cdot D_t(i) \cdot \frac{\sqrt{\varepsilon_t}}{\sqrt{1-\varepsilon_t}} = \frac{D_t(i)}{2(1-\varepsilon_t)}. \end{aligned}$$

v. Plecând de la definiția $\alpha_t = \ln \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}$ care a fost dată în pseudo-codul algoritmului AdaBoost, putem scrie mai întâi

$$\alpha_i < \alpha_j \Leftrightarrow \ln \sqrt{\frac{1-\varepsilon_i}{\varepsilon_i}} < \ln \sqrt{\frac{1-\varepsilon_j}{\varepsilon_j}}.$$

Apoi, întrucât atât funcția \ln cât și funcția rădăcină pătrată sunt strict crescătoare, rezultă că

$$\begin{aligned} \alpha_i < \alpha_j &\Leftrightarrow \frac{1-\varepsilon_i}{\varepsilon_i} < \frac{1-\varepsilon_j}{\varepsilon_j} \stackrel{\varepsilon_i, \varepsilon_j > 0}{\Leftrightarrow} \varepsilon_j(1-\varepsilon_i) < \varepsilon_i(1-\varepsilon_j) \\ &\Leftrightarrow \varepsilon_j - \varepsilon_i \varepsilon_j < \varepsilon_i - \varepsilon_i \varepsilon_j \Leftrightarrow \varepsilon_i > \varepsilon_j \end{aligned}$$

Așadar, are loc echivalența $\alpha_i < \alpha_j \Leftrightarrow \varepsilon_i > \varepsilon_j$.

vi. Putem exprima eroarea ponderată produsă la antrenare de către ipoteza h_t , în raport cu distribuția probabilistă D_{t+1} , în felul următor:

$$\begin{aligned} err_{D_{t+1}}(h_t) &\stackrel{\text{def.}}{=} \Pr_{D_{t+1}}(\{x_i | h_t(x_i) \neq y_i\}) = \sum_{i \in M} \frac{1}{Z_t} \cdot D_t(i) \cdot e^{\alpha_t} \\ &= \frac{1}{Z_t} \cdot e^{\alpha_t} \cdot \underbrace{\sum_{i \in M} D_t(i)}_{\varepsilon_t} = \frac{1}{Z_t} \cdot e^{\alpha_t} \cdot \varepsilon_t. \end{aligned} \tag{86}$$

vii. Înlocuind în formula (86) valoarea lui Z_t dată de expresia (85), obținem:

$$err_{D_{t+1}}(h_t) = \frac{1}{Z_t} \cdot e^{\alpha_t} \cdot \varepsilon_t = \frac{1}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}} \cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \cdot \varepsilon_t = \frac{1}{2}.$$

23. (Algoritmul AdaBoost:
analiza teoretică a convergenței erorii la antrenare;
învățabilitate empirică γ -slabă)
 prelucrare de Liviu Ciortuz, după
 ■ CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.2-5
 CMU, 2009 fall, Carlos Guestrin, HW2, pr. 3.1
 CMU, 2005 spring, T. Mitchell, C. Guestrin, HW2, pr. 1.1.3

Acest exercițiu vă va ghida pas cu pas ca să demonstrați că $\text{err}_S(H_T)$, eroarea la antrenare a algoritmului AdaBoost, prezentat la problema 22 — văzută ca numărul de instanțe greșit clasificate din totalul de m instanțe —, descrește foarte repede (adică, cu o rată foarte mare) în raport cu numărul de iterații efectuate (T), iar în anumite condiții converge la 0.

Observație: În acest exercițiu se lucrează fără a cere ca la fiecare iterație a algoritmului AdaBoost să se aleagă cea mai bună ipoteză „slabă”.¹⁹⁷

- Notăm cu f_T combinația liniară de ipoteze $\sum_{t=1}^T \alpha_t h_t$.¹⁹⁸ Arătați că din relația (81) rezultă $D_{T+1}(i) = \frac{1}{m \cdot \prod_{t=1}^T Z_t} e^{-y_i f_T(x_i)}$. (Această expresie reprezintă de fapt o formulă de calcul nerecursivă pentru valorile distribuțiilor probabiliste D_t .)
- Folosind rezultatul de la punctul a, arătați că $\text{err}_S(H_T) \leq \prod_{t=1}^T Z_t$, unde $\text{err}_S(H_T)$ desemnează eroarea produsă la antrenare de către AdaBoost: $\frac{1}{m} \sum_{i=1}^m 1_{\{H_T(x_i) \neq y_i\}}$.¹⁹⁹ Notația $1_{\{H_T(x_i) \neq y_i\}}$ desemnează, ca de obicei, *funcția-indicator* pentru testul $H_T(x_i) \neq y_i$; și, astăzi, $1_{\{H_T(x_i) \neq y_i\}} = 1$ pentru acei i pentru care $H_T(x_i) \neq y_i$ și 0 în caz contrar.²⁰⁰

Sugestie (1): Puteți folosi inegalitatea: $1_{\{x < 0\}} \leq e^{-x}$.²⁰¹

- La acest punct vom vedea că algoritmul AdaBoost — în loc să optimizeze în mod direct eroarea produsă la antrenare, $\text{err}_S(H_T)$ —, se mulțumește să minimizeze în mod *greedy* (deci, secvențial) produsul $\prod_{t=1}^T Z_t$, care reprezintă o *margine superioară* (engl., upper bound) pentru *eroarea la antrenare*, după cum am arătat la punctul b.

Se observă în pseudo-codul algoritmului AdaBoost că factorii de normalizare Z_1, \dots, Z_{t-1} sunt determinați în cadrul primelor $t-1$ iterații și nu pot fi modificați la iterația t . Așadar, a minimiza $\prod_{t=1}^T Z_t$ la iterația t în mod *greedy* revine la a minimiza Z_t .

Dacă facem abstracție de valoarea atribuită ponderii α_t la iterația t (și anume, $\frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$), atunci factorul de normalizare Z_t pentru distribuția D_{t+1} va putea fi scris ca o funcție de parametru α_t , pornind de la relația (81). Arătați că valoarea pentru care se atinge

¹⁹⁷Vedeți *Observația importantă* de la problema 22.

¹⁹⁸Observați că expresia funcției H_T din finalul pseudo-codului algoritmului AdaBoost — vedeți enunțul problemei 22 — se poate scrie ca $H_T(x) = \text{sign}(f_T(x))$.

¹⁹⁹Vă reamintim că S este setul de date de antrenament.

²⁰⁰Observație: $H_T(x_i) \neq y_i \stackrel{\text{def.}}{\Leftrightarrow} \text{sign}(\sum_{t=1}^T \alpha_t h_t(x_i)) \neq y_i \stackrel{\text{def.}}{\Leftrightarrow} \text{sign}(f_T(x_i)) \neq y_i \Leftrightarrow y_i \text{sign}(f_T(x_i)) = -1 \Leftrightarrow y_i f_T(x_i) < 0$. Așadar, funcțiile $1_{\{H_T(x_i) \neq y_i\}}$ și $1_{\{y_i f_T(x_i) < 0\}}$ sunt identice.

²⁰¹Putem interpreta această inegalitate sub forma următoare: funcția de cost / „pierdere“ [negativ] exponentială reprezintă o margine superioară pentru funcția de cost / pierdere 0 – 1 (engl., the 0 – 1 loss function). Pentru o definiție [de lucru] pentru funcțiile de cost / pierdere, vedeți *Explicația* de la problema 29.

minimul acestei funcții (în raport cu toate valorile posibile pentru α_t) este exact valoarea $\frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$, pe care am întâlnit-o în formularea algoritmului AdaBoost.

d. Arătați că $\prod_{t=1}^T Z_t \leq e^{-2 \sum_{t=1}^T \gamma_t^2}$, unde $\gamma_t \stackrel{\text{not.}}{=} \frac{1}{2} - \varepsilon_t$.²⁰²

Sugestie (2): De data aceasta, puteți folosi o altă margine inferioară pentru funcția de pierdere [negativ] exponentială: $1 - x \leq e^{-x}$.

e. Combinând rezultatele de la punctele c și e, observăm că eroarea la antrenare produsă de algoritmul AdaBoost se micșorează (cu o rată exponentială) pe măsură ce T crește. Întrucât rata aceasta este cuprinsă în intervalul $(0, 1)$, convergența ei este asigurată.²⁰³ La acest punct vom pune în evidență o condiție suficientă pentru ca eroarea la antrenare produsă de AdaBoost să conveargă la 0.

Presupunem că există $\gamma > 0$ astfel încât $\gamma \leq \gamma_t$ pentru orice $t = 1, 2, \dots$.²⁰⁴ (Acestă proprietate se numește „învățabilitate empirică”²⁰⁵ γ -slabă“ (engl., empirical γ -weak learnability), iar γ se numește *garanție de învățabilitate empirică slabă*.) Considerând $\varepsilon > 0$ fixat, determinați (în funcție de γ și ε) o margine superioară pentru cât de multe iterații sunt necesare pentru ca eroarea la antrenare produsă de algoritmul AdaBoost să fie mai mică decât ε , adică $\text{err}_S(H_T) < \varepsilon$. Vă cerem să exprimați răspunsul sub forma $T = \mathcal{O}(\cdot)$.

Răspuns:

a. Pentru a demonstra egalitatea din enunț, vom porni de la relația (81), exprimând probabilitatea $D_{T+1}(i)$ recursiv în funcție de $D_T(i)$, apoi de $D_{T-1}(i)$, până ajungem la $D_1(i)$:

$$\begin{aligned} D_{T+1}(i) &\stackrel{\text{def.}}{=} \frac{1}{Z_T} D_T(i) e^{-\alpha_T y_i h_T(x_i)} = D_T(i) \frac{1}{Z_T} e^{-\alpha_T y_i h_T(x_i)} \\ &\stackrel{\text{def.}}{=} D_{T-1}(i) \frac{1}{Z_{T-1}} e^{-\alpha_{T-1} y_i h_{T-1}(x_i)} \frac{1}{Z_T} e^{-\alpha_T y_i h_T(x_i)} \\ &= D_{T-1}(i) \frac{1}{Z_{T-1} Z_T} e^{-y_i (\alpha_{T-1} h_{T-1}(x_i) + \alpha_T h_T(x_i))} \\ &\quad \vdots \\ &= D_j(i) \frac{1}{Z_j \dots Z_{T-1} Z_T} e^{-y_i (\alpha_j h_j(x_i) + \dots + \alpha_{T-1} h_{T-1}(x_i) + \alpha_T h_T(x_i))} \\ &\quad \vdots \\ &= D_1(i) \frac{1}{\prod_{t=1}^T Z_t} e^{-\sum_{t=1}^T y_i \alpha_t h_t(x_i)} = \frac{1}{m \cdot \prod_{t=1}^T Z_t} e^{-y_i f_T(x_i)}. \end{aligned}$$

Produsul de forma $y_i f_T(x_i)$ (mai general, $y_i f_t(x_i)$) se numește *magine algebraică* a instanței x_i .

²⁰²LC: Putem numi variabila γ_t „ecart“, fiindcă ea reprezintă diferența dintre $1/2$, eroarea alegerii aleatorii, și ε_t , eroarea ponderată a ipotezei h_t la antrenare.

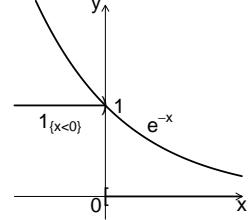
²⁰³Se știe că orice sir mărginit și monoton este convergent.

²⁰⁴LC: De fapt, este suficient ca această condiție să fie îndeplinită de la o iterație oarecare t_0 încolo.

²⁰⁵Adjectivul *empiric* se referă la faptul că eroarea analizată este *eroarea la antrenare*.

b. După cum s-a precizat în enunț, putem exprima eroarea produsă de ipoteza H_T (outputul algoritmului AdaBoost) pe setul de date de antrenament S cu ajutorul funcției de cost / pierdere $0 - 1$:

$$\text{err}_S(H_T) = \frac{1}{m} \sum_{i=1}^m 1_{\{y_i f_T(x_i) < 0\}}.$$



Folosind inegalitatea menționată în *Sugestia* (1) din enunț, putem scrie:

$$\text{err}_S(H_T) \leq \frac{1}{m} \sum_{i=1}^m e^{-y_i f_T(x_i)}.$$

Conform rezultatului de la punctul *a*, vom putea substitui $e^{-y_i f_T(x_i)}$ cu produsul $D_{T+1}(i) \cdot m \cdot \prod_{t=1}^T Z_t$ și vom obține:

$$\begin{aligned} \text{err}_S(H_T) &\leq \frac{1}{m} \sum_{i=1}^m \left(D_{T+1}(i) \cdot m \cdot \prod_{t=1}^T Z_t \right) = \sum_{i=1}^m \left(D_{T+1}(i) \prod_{t=1}^T Z_t \right) \\ &= \underbrace{\left(\prod_{t=1}^T Z_t \right)}_1 \cdot \underbrace{\left(\sum_{i=1}^m D_{T+1}(i) \right)}_1 = \prod_{t=1}^T Z_t. \end{aligned}$$

Egalitatea $\sum_{i=1}^m D_{T+1}(i) = 1$ se justifică prin faptul că D_{T+1} reprezintă o distribuție probabilistă.

c. Vom porni de la relația (82), care a fost demonstrată la problema 22.i, și anume $Z_t = \varepsilon_t \cdot e^{\alpha_t} + (1 - \varepsilon_t) \cdot e^{-\alpha_t}$. Pentru a facilita înțelegerea raționamentului care urmează, întrucât aici α_t este lăsat liber, îl vom renota cu α , deci expresia anterioară va fi văzută ca funcție de acest α , adică $Z(\alpha) = \varepsilon_t \cdot e^\alpha + (1 - \varepsilon_t) \cdot e^{-\alpha}$. (Observați că Z_t dinainte tocmai a fost și el renotat mai simplu: Z .)

Ca de obicei, pentru a găsi minimul expresiei $\varepsilon_t \cdot e^\alpha + (1 - \varepsilon_t) \cdot e^{-\alpha}$, în care ε_t (eroarea produsă de ipoteza tocmai a fost produsă de clasificatorul „slab” A) este considerată constantă, vom calcula derivata ei în raport cu α și apoi vom egala cu 0 această derivată:

$$\begin{aligned} \frac{\partial}{\partial \alpha} (\varepsilon_t \cdot e^\alpha + (1 - \varepsilon_t) \cdot e^{-\alpha}) = 0 &\Leftrightarrow \varepsilon_t \cdot e^\alpha - (1 - \varepsilon_t) \cdot e^{-\alpha} = 0 \Leftrightarrow \\ \varepsilon_t \cdot (e^\alpha)^2 = 1 - \varepsilon_t &\Leftrightarrow e^{2\alpha} = \frac{1 - \varepsilon_t}{\varepsilon_t} \Leftrightarrow 2\alpha = \ln \frac{1 - \varepsilon_t}{\varepsilon_t} \Leftrightarrow \alpha = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} = \ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}. \end{aligned}$$

Remarcați faptul că fracția $\frac{1 - \varepsilon_t}{\varepsilon_t}$ este pozitivă (deci i se poate aplica logaritmul), fiindcă $\varepsilon_t \in (0, 1/2)$. Chiar mai mult, $\alpha > 0$, fiindcă $\frac{1 - \varepsilon_t}{\varepsilon_t} > 1$, datorită aceluiasi motiv ca mai înainte. Se poate verifica imediat (analizând semnele derivatei de mai sus) că $\alpha = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$ este într-adevăr punctul în care se atinge *minimul* expresiei $\varepsilon_t \cdot e^\alpha + (1 - \varepsilon_t) \cdot e^{-\alpha}$, deci și al lui Z_t (văzut ca funcție de α):

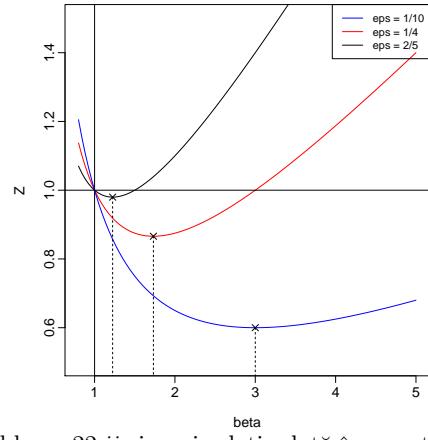
$$\varepsilon_t \cdot e^\alpha - (1 - \varepsilon_t) \cdot e^{-\alpha} > 0 \Leftrightarrow e^{2\alpha} - \frac{1 - \varepsilon_t}{\varepsilon_t} > 0 \Leftrightarrow \alpha > \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t},$$

ceea ce înseamnă că funcția $Z(\alpha) = \varepsilon_t \cdot e^\alpha - (1 - \varepsilon_t) \cdot e^{-\alpha}$ este strict crescătoare pe intervalul $\left(\frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}, +\infty\right)$ și strict descrescătoare pe intervalul $\left(-\infty, \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}\right]$.

Pentru a ilustra grafic acest rezultat, vom nota e^α cu β , deci vom scrie Z ca o funcție de argumentul β :

$$Z(\beta) = \varepsilon_t \cdot \beta + (1 - \varepsilon_t) \cdot \frac{1}{\beta}.$$

Graficul funcției $Z(\beta)$ pentru trei valori diferite ale lui ε_t este prezentat în figura alăturată. Conform raționamentului de mai sus, minimul funcției $Z(\beta)$ este atins pentru valoarea $\beta_{min} = \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}$. Vechea expresie a lui Z_t o vom regăsi astfel: $Z_t \stackrel{not.}{=} Z(\beta_{min}) \stackrel{calcul}{=} 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$.²⁰⁶

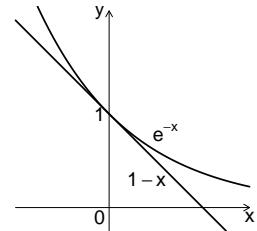


d. Folosind relația (85) care a fost dedusă la problema 22.ii și apoi relația dată în enunț pentru a exprima legătura dintre ε_t și „ecartul” γ_t , vom putea scrie:

$$\begin{aligned} \prod_{t=1}^T Z_t &= \prod_{t=1}^T 2 \cdot \sqrt{\varepsilon_t(1 - \varepsilon_t)} = \prod_{t=1}^T 2 \cdot \sqrt{\left(\frac{1}{2} - \gamma_t\right) \left(1 - \left(\frac{1}{2} - \gamma_t\right)\right)} \\ &= \prod_{t=1}^T 2 \cdot \sqrt{\left(\frac{1}{2} - \gamma_t\right) \left(\frac{1}{2} + \gamma_t\right)} = \prod_{t=1}^T 2 \cdot \sqrt{\left(\frac{1}{4} - \gamma_t^2\right)} = \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2}. \end{aligned}$$

Utilizând inegalitatea din Sugestia (2) din enunț, rezultă $1 - 4\gamma_t^2 \leq e^{-4\gamma_t^2}$ și, mai departe:

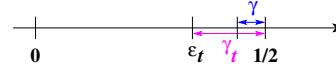
$$\begin{aligned} \prod_{t=1}^T Z_t &\leq \prod_{t=1}^T \sqrt{e^{-4\gamma_t^2}} = \prod_{t=1}^T \sqrt{(e^{-2\gamma_t^2})^2} = \prod_{t=1}^T e^{-2\gamma_t^2} \\ &= e^{-2 \sum_{t=1}^T \gamma_t^2}. \end{aligned}$$



²⁰⁶Iată alte câteva proprietăți ale funcției $Z(\beta)$, care pot fi demonstreate cu ușurință (și care sunt reflectate în graficul de mai sus):

- $\lim_{\beta \searrow 0} Z(\beta) = +\infty$.
- $\lim_{\beta \rightarrow 1} Z(\beta) = Z(1) = \varepsilon_t + 1 - \varepsilon_t = 1$.
- $\lim_{\beta \rightarrow +\infty} Z(\beta) = +\infty$.
- Dreapta de ecuație $y = \varepsilon_t \beta$ este asimptotă oblică la $+\infty$ pentru funcția $Z(\beta)$, fiind că $\lim_{\beta \rightarrow +\infty} \frac{Z(\beta)}{\beta} = \varepsilon_t$, iar $\lim_{\beta \rightarrow +\infty} Z(\beta) - \varepsilon_t \beta = 0$.
- Atunci când $\varepsilon_t \nearrow \frac{1}{2}$, rezultă că $\sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \searrow 1$, deci $\ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \searrow 0$, adică $\alpha_t \searrow 0$ (și, deci, $\beta_{min} \searrow 1$). Totodată, $Z_t \stackrel{not.}{=} Z(\beta_{min}) = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \nearrow 1$.
- Atunci când $\varepsilon_t \searrow 0$, rezultă pe de o parte că $\sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \nearrow +\infty$, deci $\alpha_t \rightarrow +\infty$ (și $\beta_{min} \nearrow +\infty$), iar pe de altă parte că $Z_t \stackrel{not.}{=} Z(\beta_{min}) = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \searrow 0$.

e. Combinând rezultatele pe care le-am obținut la punctele c și e , rezultă $\text{err}_S(H_T) \leq e^{-2\sum_{t=1}^T \gamma_t^2}$. Apoi, înținând cont că $\gamma \leq \gamma_t$ pentru $t = 1, 2, \dots$, vom avea:²⁰⁷



$$\text{err}_S(H_T) \leq e^{-2T\gamma^2}. \quad (87)$$

Prin urmare, pentru ca inegalitatea $\text{err}_S(H_T) < \varepsilon$ să aibă loc, este suficient să impunem restricția următoare:

$$-2T\gamma^2 < \ln \varepsilon \Leftrightarrow 2T\gamma^2 > -\ln \varepsilon \Leftrightarrow 2T\gamma^2 > \ln \frac{1}{\varepsilon} \Leftrightarrow T > \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon}.$$

Așadar, $T = \mathcal{O}\left(\frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}\right)$.

Observație: Din relația (87) rezultă că $\text{err}_S(H_T) \rightarrow 0$ atunci când $T \rightarrow \infty$.

24.

(Algoritmul AdaBoost, folosind “decision stumps”: aplicare pe date din \mathbb{R}^2)

■ *prelucrare de L. Ciortuz, după CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.6*

Considerăm setul de date de antrenament din tabelul alăturat. Rulați $T = 3$ iterații ale algoritmului AdaBoost, folosind drept clasificatori slabii *compași de decizie* (engl., decision stumps), care determină separatori liniari paraleli cu axele de coordinate (engl., axis-aligned separators). Reprezentați aceste date în planul euclidian; pe figura obținută veți trasa dreptele corespunzătoare ipotezelor „slabe“ h_t ,²⁰⁸ iar la final veți completa tabelul de mai jos. Pentru pseudo-codul algoritmului AdaBoost, vedeti problema 22. Vă rugăm să citiți și să rețineți *Observația importantă* pe care am redactat-o imediat după acel pseudo-cod.

x_i	X_1	X_2	y_i
x_1	1	2	+1
x_2	2	3	+1
x_3	3	4	-1
x_4	3	2	-1
x_5	3	1	-1
x_6	4	4	-1
x_7	5	4	-1
x_8	5	2	+1
x_9	5	1	+1

t	ε_t	α_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$	$D_t(9)$	$\text{err}_S(H_T)$
1												
2												
3												

Recomandare: Rolul acestui exercițiu este de a vă ajuta să înțelegeți pas cu pas cum anume lucrează în practică algoritmul AdaBoost. Vă sugerăm ca, după ce veți fi înțeles

²⁰⁷Întrucât $\gamma_t \geq \gamma > 0$, au loc următoarele echivalențe: $\gamma_t \geq \gamma \Leftrightarrow \gamma_t^2 \geq \gamma^2 \Leftrightarrow -2\gamma_t^2 \leq -2\gamma^2 \Leftrightarrow e^{-2\gamma_t^2} \leq e^{-2\gamma^2}$. Considerând această ultimă inegalitate pentru fiecare valoare a lui $t \in \{1, 2, \dots, T\}$ și înmulțind membru cu membru, rezultă: $\prod_{t=1}^T e^{-2\gamma_t^2} \leq \prod_{t=1}^T e^{-2\gamma^2} \Leftrightarrow e^{-2\sum_{t=1}^T \gamma_t^2} \leq (e^{-2\gamma^2})^T \Leftrightarrow e^{-2\sum_{t=1}^T \gamma_t^2} \leq e^{-2T\gamma^2}$.

²⁰⁸Însoțiti fiecare dintre dreptele respective cu o pereche de semne + și -, corespunzătoare zonelor de decizie determinate.

rezolvarea acestui exercițiu, să implementați mai întâi un program / o funcție, care să calculeze *eroarea ponderată la antrenare* (engl., weighted training error) produsă de un anumit compas de decizie, în raport cu o distribuție de probabilitate (D) definită pe setul de date de antrenament. Ulterior, puteți extinde acest program la o implementare completă a algoritmului AdaBoost, conform pseudo-codului din problema 22.

Răspuns:

Față de reprezentarea grafică (de tip arbore de adâncime 1) cu care ne-am obișnuit în trecut pentru *compașii de decizie*, aici vom lucra cu *reprezentarea analitică* următoare: pentru un atribut de tip continuu X care ia valori $x \in \mathbb{R}$ și pentru un prag (engl., threshold) oarecare $s \in \mathbb{R}$, putem defini doi compași de decizie:

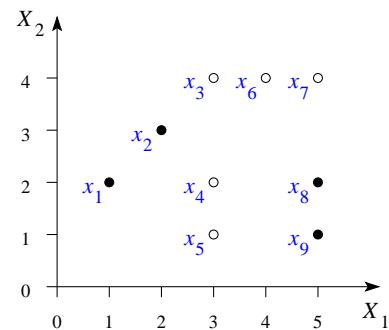
$$\text{sign}(x - s) = \begin{cases} 1 & \text{dacă } x \geq s \\ -1 & \text{dacă } x < s \end{cases} \quad \text{și} \quad \text{sign}(s - x) = \begin{cases} -1 & \text{dacă } x \geq s \\ 1 & \text{dacă } x < s. \end{cases}$$

Pentru conveniență, în cele ce urmează — atunci când nu riscăm să creăm ambiguități — vom renota primul compas de decizie cu $X \geq s$ și al doilea cu $X < s$.

Faptul că în acest exercițiu se cere aplicarea algoritmului AdaBoost cu compași de decizie înseamnă că la fiecare iterare (t) clasificatorul „slab“ (notat cu A) din pseudo-codul algoritmului AdaBoost selectează un compas de decizie, și anume — conform *Observației importante* din enunțul problemei 22 — unul care are eroarea ponderată la antrenare minimă în raport cu distribuția probabilistă curentă (D_t).²⁰⁹

Exact ca atunci când am lucrat cu algoritmul ID3 cu atribute continue, vom considera căte un prag intermediar pentru fiecare pereche de valori succesive [luate de un atribut continuu X] corespunzătoare schimbărilor de etichete. Formal, pentru orice astfel de perechi de valori x_i, x_{i+1} , cu $y_i y_{i+1} < 0$ și $x_i < x_{i+1}$, dar fără să mai existe vreun $x_j \in \text{Val}(X)$ cu proprietatea $x_i < x_j < x_{i+1}$, vom considera pragul $(x_i + x_{i+1})/2$.²¹⁰ De asemenea, vom considera și un *prag exterior* intervalului de valori ale atributului X în multimea de instanțe de antrenament.²¹¹

Așadar, la *prima iterare* a algoritmului AdaBoost (adică, pentru $t = 1$), *pragurile* de separare pentru valorile celor două variabile continue (X_1 și X_2) care corespund celor două coordonate ale instanțelor de antrenament (x_1, \dots, x_9) sunt $\frac{1}{2}, \frac{5}{2}$ și $\frac{9}{2}$ pentru X_1 , și respectiv $\frac{1}{2}, \frac{3}{2}, \frac{5}{2}$ și $\frac{7}{2}$ pentru X_2 .



²⁰⁹Deci algoritmul slab A nu este algoritmul ID3 care produce un compas de decizie cu câștig de informație minim!

²¹⁰În cazul algoritmului ID3, există un *rezultat teoretic* (vedeți problema 43) care demonstrează că nu este necesar să considerăm alte praguri pentru un atribut continuu X decât cele situate între perechi de valori succesive având etichete [de semne] contrare, fiindcă valoarea IG-ului celorlalte praguri este situată în mod cert sub valoarea IG-ului maximal pentru acel atribut. LC: În cazul algoritmului AdaBoost, se poate demonstra un rezultat similar care, în consecință, ne va permite să simplificăm aplicarea clasificatorului „slab“ (A).

²¹¹Compașii de decizie corespunzători acestui prag „exterior“ pot fi puși în corespondență cu arborii de decizie de adâncime 0 pe care i-am întâlnit în precedent.

Observăm însă că se poate renunța la pragul „exterior“ $\frac{1}{2}$ pentru X_2 , deoarece compașii de decizie corespunzător lui se comportă identic cu compașii de decizie corespunzător pragului „exterior“ $\frac{1}{2}$ pentru X_1 .²¹²

Pentru compașii de decizie corespunzători acestei prime iterații, *erorile ponderate la antrenare* sunt cele prezentate centralizat în tabelele de mai jos. Pentru calcule, am folosit egalitățile $err_{D_t}(X_1 \geq s) = 1 - err_{D_t}(X_1 < s)$ și, similar, $err_{D_t}(X_2 \geq s) = 1 - err_{D_t}(X_2 < s)$, pentru orice prag s și orice iterație $t = 1, 2, \dots$. Aceste egalități sunt foarte ușor de demonstrat.

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_1}(X_1 < s)$	$\frac{4}{9}$	$\frac{2}{9}$	$\frac{4}{9} + \frac{2}{9} = \frac{2}{3}$
$err_{D_1}(X_1 \geq s)$	$\frac{5}{9}$	$\frac{7}{9}$	$\frac{1}{3}$

s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_1}(X_2 < s)$	$\frac{4}{9}$	$\frac{1}{9} + \frac{3}{9} = \frac{4}{9}$	$\frac{2}{9} + \frac{1}{9} = \frac{1}{3}$	$\frac{2}{9}$
$err_{D_1}(X_2 \geq s)$	$\frac{5}{9}$	$\frac{5}{9}$	$\frac{2}{3}$	$\frac{7}{9}$

Se observă că eroarea ponderată minimă la antrenare ($\varepsilon_1 = 2/9$) este obținută pentru compașii de decizie $X_1 < 5/2$ și $X_2 < 7/2$.²¹³ Alegem drept *cea mai bună ipoteză* la această iterație pe $h_1 = sign\left(\frac{7}{2} - X_2\right)$, separatorul corespunzător fiind dreapta de ecuație $X_2 = \frac{7}{2}$. Ipoteza h_1 clasifică greșit instanțele x_4 și x_5 . Vom avea:

$$\begin{aligned}\gamma_1 &= \frac{1}{2} - \frac{2}{9} = \frac{5}{18} \\ \alpha_1 &= \frac{1}{2} \ln \frac{1 - \varepsilon_1}{\varepsilon_1} = \ln \sqrt{\left(1 - \frac{2}{9}\right) : \frac{2}{9}} = \ln \sqrt{\frac{7}{2}} \approx 0.626\end{aligned}$$

Acum algoritmul trebuie să „pregătească“ o nouă distribuție (D_2), pentru iterația următoare. Distribuția D_2 va fi obținută prin modificarea vechii distribuții (D_1), în așa fel încât algoritmul să se poată concentra cu preponderență asupra instanțelor care au fost greșit clasificate (engl., misclassified). Conform relației (81), vom scrie:

$$D_2(i) = \frac{1}{Z_1} D_1(i) \underbrace{\left(e^{-\alpha_1}\right)^{y_i h_1(x_i)}}_{\sqrt{2/7}} = \begin{cases} \frac{1}{Z_1} \cdot \frac{1}{9} \cdot \sqrt{\frac{2}{7}} & \text{pentru } i \in \{1, 2, 3, 6, 7, 8, 9\}; \\ \frac{1}{Z_1} \cdot \frac{1}{9} \cdot \sqrt{\frac{7}{2}} & \text{pentru } i \in \{4, 5\}. \end{cases}$$

²¹²De aceea în tabelele de mai jos, pentru aceste cazuri se folosește culoarea verde.

²¹³În tabele, pentru erorile minime am folosit culoarea albastră.

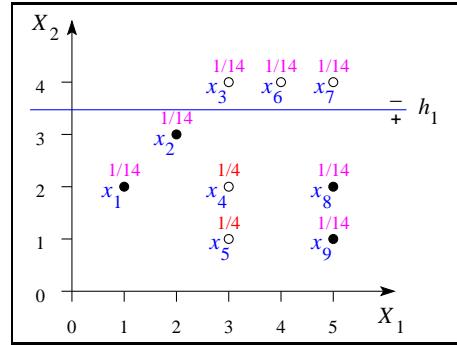
Vă reamintim că Z_1 este așa-numitul *factor de normalizare* pentru distribuția probabilistă D_2 . Așadar, din relația $\sum_i D_2(i) = 1$ rezultă:

$$Z_1 = \frac{1}{9} \left(7 \cdot \sqrt{\frac{2}{7}} + 2 \cdot \sqrt{\frac{7}{2}} \right) = \frac{2\sqrt{14}}{9} = 0.8315$$

Prin urmare,

$$D_2(i) = \begin{cases} \frac{9}{2\sqrt{14}} \cdot \frac{1}{9} \cdot \sqrt{\frac{2}{7}} = \frac{1}{14} & \text{pentru } i \notin \{4, 5\}; \\ \frac{9}{2\sqrt{14}} \cdot \frac{1}{9} \cdot \sqrt{\frac{7}{2}} = \frac{1}{4} & \text{pentru } i \in \{4, 5\}. \end{cases}$$

Figura alăturată prezintă grafic exemplele de antrenament (x_i, y_i) , pentru $i = 1, \dots, 9$, împreună cu probabilitățile care tocmai le-au fost asociate ($D_2(x_i)$), precum și separatorul „învățat” la această primă iterare (h_1). Probabilitățile corespunzătoare instanțelor clasificate eronat de către h_1 au fost scrise cu culoarea roșie. Se observă că aceste probabilități sunt mult mai mari decât probabilitățile celorlalte instanțe și că împreună se sumează la valoarea 1/2. Am notat cu + și respectiv – cele două zone de decizie determinate de h_1 .



Observație (1): Dacă, drept ipoteză h_1 , în locul lui $\text{sign}\left(\frac{7}{2} - X_2\right)$ am fi ales $\text{sign}\left(\frac{5}{2} - X_1\right)$, atunci cursul rezolvării ulterioare ar fi fost altul (deși ambele ipoteze au aceeași eroare ponderată – minimală – la antrenare): x_8 și x_9 ar fi primit ponderile 1/4, iar x_4 și x_5 ar fi primit ponderile 1/14. Așadar, outputul algoritmului AdaBoost nu este în mod neapărat unic determinat!

Iterația $t = 2$:

Vom proceda similar cu iterarea precedentă, doar că, de data aceasta, vom folosi distribuția D_2 .

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_2}(X_1 < s)$	$\frac{4}{14}$	$\frac{2}{14}$	$\frac{2}{14} + \frac{2}{4} + \frac{2}{14} = \frac{11}{14}$
$err_{D_2}(X_1 \geq s)$	$\frac{10}{14}$	$\frac{12}{14}$	$\frac{3}{14}$

s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_2}(X_2 < s)$	$\frac{4}{14}$	$\frac{1}{4} + \frac{3}{14} = \frac{13}{28}$	$\frac{2}{4} + \frac{1}{14} = \frac{8}{14}$	$\frac{2}{4} = \frac{1}{2}$
$err_{D_2}(X_2 \geq s)$	$\frac{10}{14}$	$\frac{15}{28}$	$\frac{6}{14}$	$\frac{1}{2}$

Observație (2): Conform rezultatului teoretic de la punctul *vii* al problemei 22, luarea în calcul a compasului de decizie corespunzător testului $X_2 \geq 7/2$ este aici superfluă, întrucât acest compas de decizie a fost ales ca ipoteză optimală la iterația precedentă. L-am pus totuși în tabel (folosind culoarea roșie), de dragul realizării unei prezentări exhaustive.

Cea mai bună ipoteză este acum $h_2 = \text{sign}\left(\frac{5}{2} - X_1\right)$; separatorul corespunzător acestei ipoteze este dreapta de ecuație $X_1 = \frac{5}{2}$. Urmează să explicităm cum am calculat eroarea ε_2 și, aferent, să calculăm „ecartul” γ_2 , precum și ponderea α_2 , după care vom trece la determinarea noii distribuții (D_3).

$$\begin{aligned}\varepsilon_2 &= P_{D_2}(\{x_8, x_9\}) = \frac{2}{14} = \frac{1}{7} = 0.143, \quad \text{deci } \gamma_2 = \frac{1}{2} - \frac{1}{7} = \frac{5}{14} \\ \alpha_2 &= \ln \sqrt{\frac{1 - \varepsilon_2}{\varepsilon_2}} = \ln \sqrt{\left(1 - \frac{1}{7}\right) : \frac{1}{7}} = \ln \sqrt{6} = 0.896 \\ D_3(i) &= \frac{1}{Z_2} \cdot D_2(i) \cdot (\underbrace{e^{-\alpha_2}}_{1/\sqrt{6}})^{y_i h_2(x_i)} = \begin{cases} \frac{1}{Z_2} \cdot D_2(i) \cdot \frac{1}{\sqrt{6}} & \text{dacă } h_2(x_i) = y_i; \\ \frac{1}{Z_2} \cdot D_2(i) \cdot \sqrt{6} & \text{în caz contrar} \end{cases} \\ &= \begin{cases} \frac{1}{Z_2} \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} & \text{pentru } i \in \{1, 2, 3, 6, 7\}; \\ \frac{1}{Z_2} \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} & \text{pentru } i \in \{4, 5\}; \\ \frac{1}{Z_2} \cdot \frac{1}{14} \cdot \sqrt{6} & \text{pentru } i \in \{8, 9\}. \end{cases}\end{aligned}$$

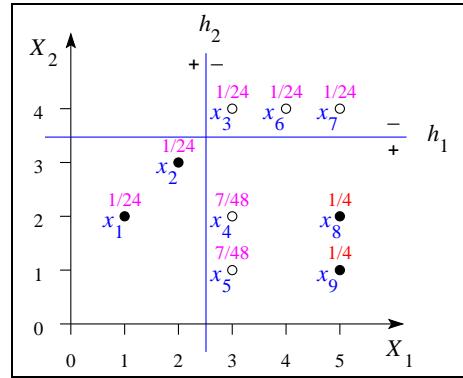
Deci

$$\begin{aligned}Z_2 &= 5 \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} + 2 \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} + 2 \cdot \frac{1}{14} \cdot \sqrt{6} = \frac{5}{14\sqrt{6}} + \frac{1}{2\sqrt{6}} + \frac{\sqrt{6}}{7} = \frac{12 + 12}{14\sqrt{6}} \\ &= \frac{24}{14\sqrt{6}} = \frac{2\sqrt{6}}{7} \approx 0.7\end{aligned}$$

și, prin urmare

$$D_3(i) = \begin{cases} \frac{7}{2\sqrt{6}} \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} = \frac{1}{24} & \text{pentru } i \in \{1, 2, 3, 6, 7\}; \\ \frac{7}{2\sqrt{6}} \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} = \frac{7}{48} & \text{pentru } i \in \{4, 5\}; \\ \frac{7}{2\sqrt{6}} \cdot \frac{1}{14} \cdot \sqrt{6} = \frac{1}{4} & \text{pentru } i \in \{8, 9\}. \end{cases}$$

Facem din nou reprezentarea grafică a datelor de antrenament, împreună cu noile probabilități asociate (D_3) și cei doi separatori care au fost „învățați“ până acum (h_1 și h_2); vedeti figura alăturată. Este instructiv să comparăm această figură cu cea dinainte (adică, pentru $t = 1$), pentru a observa evoluția probabilităților de la o iterare la alta. Instantele x_1, x_2, x_3, x_6 și x_7 au acum ponderile foarte mici, pentru că au fost clasificate corect atât de către h_1 cât și de către h_2 .



Iterația $t = 3$:

Erorile ponderate la antrenare pentru compașii de decizie, în raport cu distribuția D_3 sunt:

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_3}(X_1 < s)$	$\frac{2}{24} + \frac{2}{4} = \frac{7}{12}$	$\frac{2}{4}$	$\frac{2}{24} + 2 \cdot \frac{7}{48} + 2 \cdot \frac{1}{4} = \frac{21}{24}$
$err_{D_3}(X_1 \geq s)$	$\frac{5}{12}$	$\frac{2}{4}$	$\frac{3}{24} = \frac{1}{8}$

s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_3}(X_2 < s)$	$\frac{7}{12}$	$\frac{7}{48} + \frac{2}{24} + \frac{1}{4} = \frac{23}{48}$	$2 \cdot \frac{7}{48} + \frac{1}{24} = \frac{1}{3}$	$2 \cdot \frac{7}{48} = \frac{7}{24}$
$err_{D_3}(X_2 \geq s)$	$\frac{5}{12}$	$\frac{25}{48}$	$\frac{2}{3}$	$\frac{17}{24}$

Similar cu *Observația* precedentă (vedeti iterare $t = 2$), facem mențiunea că am fi putut renunța la elaborarea compasului de decizie corespunzător testului $X_1 < 5/2$.

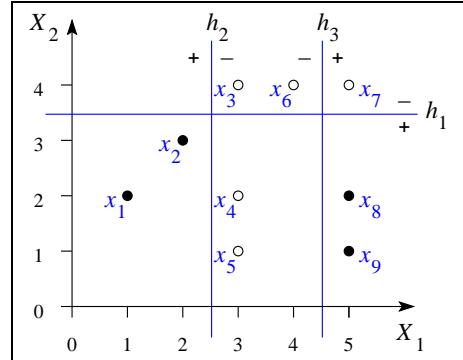
Acum cea mai bună ipoteză este $h_3 = \text{sign}\left(X_1 - \frac{9}{2}\right)$, iar separatorul corespunzător este dreapta de ecuație $X_1 = \frac{9}{2}$. Vom explicita cum am calculat eroarea ε_3 și apoi vom calcula „ecartul“ γ_3 , precum și ponderea α_3 :

$$\varepsilon_3 = P_{D_3}(\{x_1, x_2, x_7\}) = 2 \cdot \frac{1}{24} + \frac{1}{24} = \frac{3}{24} = \frac{1}{8}$$

$$\gamma_3 = \frac{1}{2} - \frac{1}{8} = \frac{3}{8}$$

$$\alpha_3 = \ln \sqrt{\frac{1 - \varepsilon_3}{\varepsilon_3}} = \ln \sqrt{\left(1 - \frac{1}{8}\right) : \frac{1}{8}} = \ln \sqrt{7} = 0.973$$

Reprezentarea grafică a datelor de antrenament împreună cu cei trei separatori învățați este furnizată în figura alăturată.



În final, vom pune rezultatele pe care le-am obținut până acum în tabelul care a fost dat în enunț.

t	ε_t	α_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$	$D_t(9)$	$err_S(H_t)$
1	$2/9$	$\ln \sqrt{7/2}$	$1/9$	$1/9$	$1/9$	$1/9$	$1/9$	$1/9$	$1/9$	$1/9$	$1/9$	$2/9$
2	$2/14$	$\ln \sqrt{6}$	$1/14$	$1/14$	$1/14$	$1/4$	$1/4$	$1/14$	$1/14$	$1/14$	$1/14$	$2/9$
3	$1/8$	$\ln \sqrt{7}$	$1/24$	$1/24$	$1/24$	$7/48$	$7/48$	$1/24$	$1/24$	$1/4$	$1/4$	0

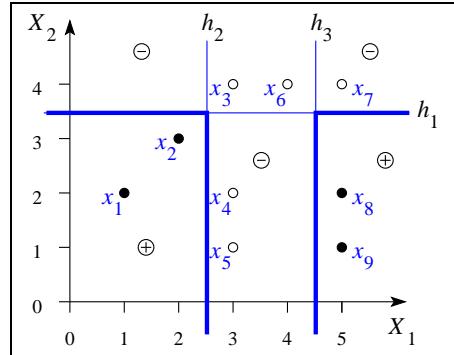
Observație (3): Tabelul de mai jos vă servește ca să înțelegeți cum anume se calculează $err_S(H_t)$. În acest tabel, am marcat cu culoarea roșie clasificările eronate făcute de ipotezele h_t , pentru $t = 1, 2, 3$.

t	α_t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1	0.626	+1	+1	-1	+1	+1	-1	-1	+1	+1
2	0.896	+1	+1	-1	-1	-1	-1	-1	-1	-1
3	0.973	-1	-1	-1	-1	-1	-1	+1	+1	+1
	$H_T(x_i)$	+1	+1	-1	-1	-1	-1	-1	+1	+1

Se observă că H_1 (deci, la finalul primei iterații a algoritmului AdaBoost) a clasificat greșit instanțele x_4 și x_5 , iar H_2 (deci, la finalul celei de-a doua iterații) instanțele x_8 și x_9 . La finalul celei de-a treia iterații, eroarea produsă pe datele de antrenament (de către ipoteza H_3) este 0. Vă readucem aminte că $H_T(x_i) \stackrel{\text{def.}}{=} \text{sign}(\sum_{t=1}^T \alpha_t h_t(x_i))$, deci notând $x_i = (x_{i,1}, x_{i,2})$, putem scrie

$$H_3(x_i) = \text{sign}\left(\ln \sqrt{\frac{7}{2}} \cdot \text{sign}\left(\frac{7}{2} - x_{i,2}\right) + \ln \sqrt{6} \cdot \text{sign}\left(\frac{5}{2} - x_{i,1}\right) + \ln \sqrt{7} \cdot \text{sign}\left(x_{i,1} - \frac{9}{2}\right)\right).$$

Observație (4): Se poate constata imediat că o instanță nouă aleasă în mod arbitrar în partea din stânga sus a figurii care prezintă datele de antrenament (de exemplu, instanța $(1, 4)$) va fi clasificată de către ipoteza H_3 — care a fost „învățată“ de către algoritmul AdaBoost după cele trei iterări — ca fiind negativă, fiindcă $-\alpha_1 + \alpha_2 - \alpha_3 = -0.626 + 0.896 - 0.973 < 0$. Făcând și alte raționamente similare, putem conchide că *zonele de decizie și granițele de decizie* produse de către AdaBoost pentru acest set de date de antrenament vor fi cele indicate în figura alăturată.

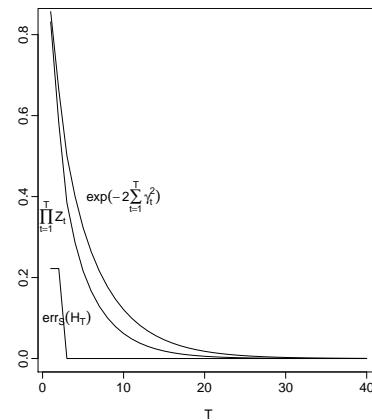
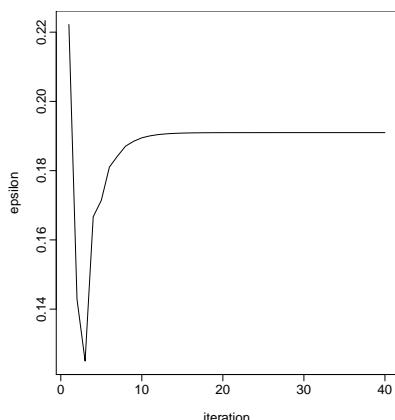


Observație (5): Execuția algoritmului AdaBoost ar putea continua (dacă am fi fixat inițial un $T > 3$), chiar dacă am obținut $err_S(H_t) = 0$ la iterarea $t = 3$. Dacă veți elabora detaliile, veți vedea că pentru $t = 4$ am obține ca ipoteză optimă $X_2 < \frac{7}{2}$ (care a fost selectată și la iterarea $t = 1$). Această ipoteză ar produce acum la antrenare eroarea ponderată $\varepsilon_4 = \frac{1}{6}$, deci ar primi în noul output H_4 factorul $\alpha_4 = \ln \sqrt{5}$ (care s-ar alătura factorului $\alpha_1 = \ln \sqrt{7/2}$). Astfel se va întări încrederea în ipoteza $X_2 < \frac{7}{2}$. Să reținem, deci, că algoritmul AdaBoost poate selecta de mai multe ori o aceeași ipoteză „slabă“ (însă niciodată la iterări consecutive, conform pr. 22.vii).

În final prezentăm două grafice utile, care au fost realizate de către studentul Sebastian Ciobanu:

Evoluția erorii ε_t în raport cu t :

Cele două margini superioare (eng., upper bounds) pentru eroarea empirică a ipotezei combinate H_T :

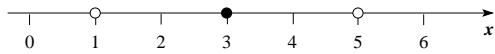


25.

(AdaBoost și non-învățabilitate γ -slabă:
exemplificare pe un set de date din \mathbb{R})

■ prelucrare de Liviu Ciortuz, după
CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final, pr. 8.a-e

În această problemă vom studia modul în care se comportă algoritmul AdaBoost pe un set de date foarte simplu din \mathbb{R} , $x_1 = 1, x_2 = 3, x_3 = 5$, ilustrat în figura alăturată.



Vă readucem aminte *convenția* noastră de notare: simbolul \bullet desemnează instanțe pozitive, iar simbolul \circ instanțe negative.

Vom folosi ca ipoteze „slabe“ (h_t) compași de decizie. Vă readucem aminte că un compas de decizie alege o valoare reală constantă s și clasifică toate punctele $x > s$ ca fiind într-o clasă, iar toate celelalte puncte ($x \leq s$) în cealaltă clasă.

- a. i. Care este valoarea ponderilor / probabilităților asignate inițial fiecărei instanțe?
 - ii. Marcați pe desenul de mai sus separatorul decizional (granița) corespunzătoare primului compas de decizie. Indicați zona pozitivă și zona negativă, de o parte și de alta a separatorului decizional.
 - iii. Încercuiți instanța a cărei pondere / probabilitate crește la execuția primei iterări de boosting.
 - iv. Calculați ponderile / probabilitățile asignate fiecărei instanțe după prima iterărie.
- b. Poate oare algoritmul AdaBoost să clasifice în mod perfect toate exemplele de antrenament? Dacă răspunsul este *nu*, justificați în detaliu. Dacă răspunsul este *da*, care este numărul minim de iterări în care se atinge eroarea la antrenare 0?

Răspuns:

Observație: La stabilirea distribuțiilor probabiliste D_{t+1} (pentru $t = 1, 2, \dots$), vom ține cont

- în primul rând de rezultatul de la exercițiul 22.vii, și anume că $D_{t+1}(h_t) = 1/2$,
- iar apoi de faptul — specificat în relația de definiție (81) pentru distribuția D_{t+1} — că probabilitățile instanțelor de antrenament care au fost corect (respectiv incorect) catalogate de către ipoteza slabă h_t vor fi modificate (pentru iterăția $t + 1$) proporțional, și anume cu factorul $e^{-\alpha_t}$ (respectiv e^{α_t}).²¹⁴

- a. La iterăția $t = 1$, se lucrează cu distribuția probabilistă $D_1(i) = 1/3$ pentru $i \in \{1, 2, 3\}$ și, indiferent de cum am fixa pragul compasului de decizie h_1 ,²¹⁵ instanța care este incorect clasificată de către ipoteza h_1 — în legătură cu care nu precizăm deocamdată decât că trebuie să aibă eroarea ponderată la antrenare minimală, ca de obicei — este $x_2 = 3$. Pentru iterăția $t + 1 = 2$, instanța x_2 va primi (cf. rezultatul teoretic de la ex. 22.vii) ponderea / probabilitatea $1/2$, în vreme ce probabilitatea care rămâne, și anume $1 - (1/2) = 1/2$ va fi împărțită în mod egal între instanțele $x_1 = 1$ și $x_3 = 5$; aşadar acestea vor primi acum ponderile $1/4$.

²¹⁴Mai precis, folosind notațiile de la ex. 22, $D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{-\alpha_t}$, pentru $i \in C$, și

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{\alpha_t}, \text{ pentru } i \in M.$$

²¹⁵Atât pentru varianta când se folosește prag exterior (engl., outside threshold) cât și pentru varianta când *nu* se folosește prag exterior. Vedeti precizările de mai jos.

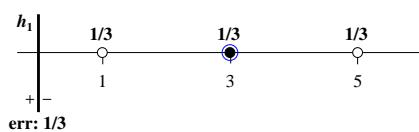
Dacă folosim praguri exterioare, pentru ipoteza slabă h_1 putem alege pragul 0 (vedeți primul desen de mai jos).

b. La iterată $t = 2$, instanța incorrect clasificată de către ipoteza slabă h_2 (vedeți detaliile din al doilea desen de mai jos) este $x_3 = 5$. În continuare, la iterată $t + 1 = 3$, instanța x_3 va primi probabilitatea $1/2$, iar restul masei de probabilitate ($1/2$) va fi împărțit în mod proporțional între instanțele $x_1 = 1$ și $x_2 = 3$, deci acestea vor avea acum probabilitățile $\frac{1}{6} = \frac{1}{2} \cdot \frac{1}{3}$ și respectiv $\frac{1}{3} = \frac{1}{2} \cdot \frac{2}{3}$.

Erorile ponderate ε_t se calculează ușor, la fel și ponderile α_t , aşa că vom reda acum în mod direct rezultatele obținute la fiecare iteratăie (presupunând, ca și mai sus, că folosim praguri exterioare).

Se observă că după $T = 3$ iterării, algoritmul AdaBoost obține eroarea la antrenare $err_S(H_3) = 0$.

• $t = 1 :$



• $t = 1 :$

$$\varepsilon_1 = 1/3 \Rightarrow \alpha_1 = \ln \sqrt{2} = 0.3465, \\ err_S(H_1) = 1/3$$

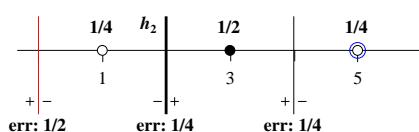
• $t = 2 :$

$$\varepsilon_2 = 1/4 \Rightarrow \alpha_2 = \ln \sqrt{3} = 0.5493$$

	x_1	x_2	x_3
α_1	-	-	-
α_2	-	+	+
$H_2(x_i)$	-	+	+

$$\Rightarrow err_S(H_2) = 1/3$$

• $t = 2 :$



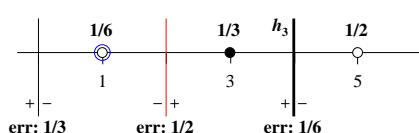
• $t = 3 :$

$$\varepsilon_3 = 1/6 \Rightarrow \alpha_3 = \ln \sqrt{5} = 0.8047$$

	x_1	x_2	x_3
α_1	-	-	-
α_2	-	+	+
α_3	+	+	-
$H_3(x_i)$	-	+	-

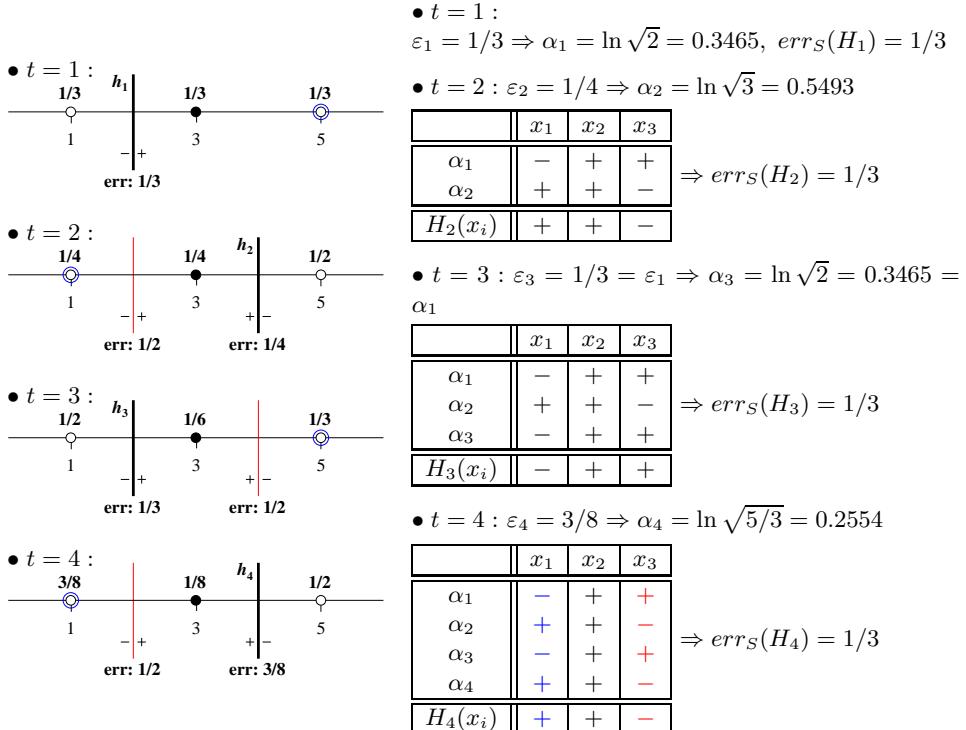
$$\Rightarrow err_S(H_3) = 0$$

• $t = 3 :$



Atenție! Întrucât se va dovedi a fi un fapt foarte instructiv, vom elabora acum o *a doua soluție*, pentru care însă ne vom auto-impune din start să nu folosim praguri exterioare.²¹⁶ Iată soluția obținută de data aceasta (mai exact, rezultatul primelor 4 iterării):

²¹⁶Aceasta a fost, în esență, soluția dată pentru acest exercițiu la CMU, însă ea este greșită, iar când spunem aceasta ne bazăm pe rezultatul teoretic de la problema 63, cu care această soluție (de la CMU) vine în contradicție.



Se observă că

- la oricare două iterații consecutive, pragurile alese ($s = 2$ și $s = 4$) vor alterna, conform unei consecințe imediate de la relația demonstrată la ex. 22.vii (vedeți *Observația* 2 de la pagina 259);
- instanța $x_2 = 3$ este clasificată corect de către oricare ipoteză selectată, h_t , de aceea, conform relației (81) ponderea ei va scădea mereu (observați valorile $1/3, 1/4, 1/6, 1/8$). Mai mult, se poate demonstra că această probabilitate tinde descrescător la 0.²¹⁷
- conform aceleiași relații de la ex. 22.vii, va rezulta că la fiecare iterație una dintre instanțele $x_1 = 1$ și $x_3 = 5$ va avea probabilitatea $1/2$, iar cealaltă instanță va avea probabilitatea $1/2$ minus probabilitatea instanței (corect clasificată) $x_2 = 3$. Așadar, această ultimă probabilitate va tinde crescător la $1/2$. În consecință, în acest caz — adică, atunci când nu se lucrează cu praguri exterioare, pe acest set de date de antrenament — nu există *garanție de învățabilitate empirică slabă* ($\gamma > 0$).

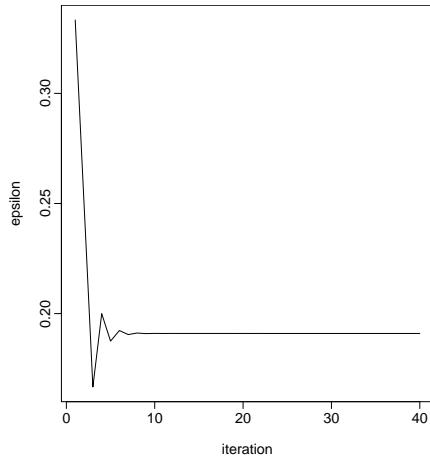
Observație: Se poate constata — din tabelele pentru $\text{err}_S(H_T)$ și în special din ultimul tabel — că $H_T(x_1)$ va avea întotdeauna semnul contrar semnului lui $H_T(x_3)$, deoarece la fiecare iterație (t) instanțele x_1 și x_3 vor fi clasificate cu semne contrare de către ipoteza slabă selectată, h_t . Prin urmare, în acest caz — repetăm, atunci când nu se lucrează cu praguri exterioare, pe acest set de date de antrenament — nu este posibil ca AdaBoost să obțină eroare la antrenare 0.

²¹⁷Vedeți graficul de mai jos. Analitic, la această concluzie se poate ajunge prin reducere la absurd, pornind de la rezultatul teoretic obținut la ex. 23.e, coroborat cu *Observația* de mai jos.

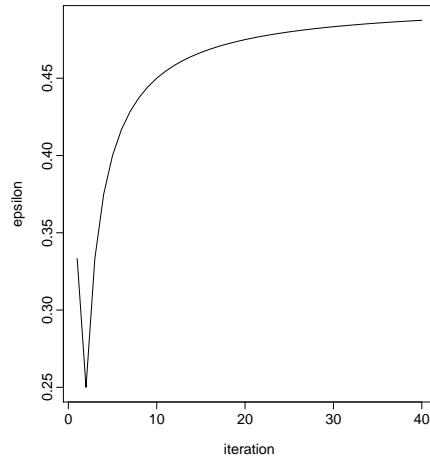
În *concluzie*, este absolut necesar ca în toate cazurile, la aplicarea algoritmului AdaBoost să fie folosite parguri exterioare (mai precis, este suficient un prag exterior). Altfel, este posibil ca AdaBoost să nu poată să clasifice perfect datele de antrenament.

Grafcile următoare, făcute de Sebastian Ciobanu, arată evoluția valorilor *erorilor pondereate la antrenare* care au fost produse atunci când am rulat 40 de iterații cu algoritmul AdaBoost pe aceste date, în cele două variante: cu și respectiv fără prag exterior. Se observă clar existența unui *prag $\gamma > 0$* de *învățabilitate empirică slabă* în primul caz, și lipsa unui astfel de prag în cel de-al doilea caz.

cu prag exterior:

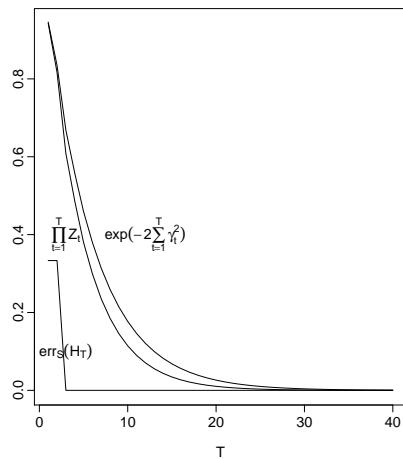


fără prag exterior:

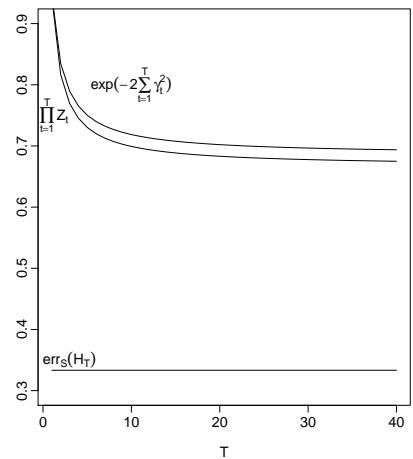


În sfârșit, următoarele grafice, făcute tot de Sebastian Ciobanu, pun în evidență legătura dintre rezultatele obținute mai sus și cele două *margini superioare* pentru *eroarea la antrenare* produsă de *ipoteza combinată* H_T livrată la ieșire de algoritmul AdaBoost pe aceste date (vedeți ex. 23.bd).

cu prag exterior:



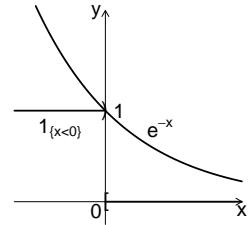
fără prag exterior:



26. (Algoritmul AdaBoost ca algoritm de *optimizare [secvențială]*: deducerea regulii de calcul a ponderilor α_t folosind *metoda minimizării secvențiale* a costului / pierderii [negativ] exponentiale)
prelucrare de Liviu Ciortuz, după CMU, 2008 fall, Eric Xing, HW3, pr. 4.1.1 CMU, 2008 fall, Eric Xing, midterm, pr. 5.1 CMU, 2012 fall, E. Xing, A. Singh, HW4, pr. 3.ab

La problema 23.c am văzut că în algoritmul AdaBoost se urmărește (în mod indirect) să se minimizeze eroarea la antrenare $err_S(H_T)$ prin minimizarea *secvențială* a marginii sale superioare $\prod_{t=1}^T Z_t$. Aceasta înseamnă că la fiecare iterație t (unde $1 \leq t \leq T$) alegem valoarea ponderii α_t astfel încât să minimizăm Z_t (văzut ca funcție de α_t , conform relației (82) de la problema problema 22.i).

Aici veți vedea că o altă cale de a explica modul în care funcționază algoritmul AdaBoost este determinată de *obiectivul* de a minimiza în mod *greedy* (deci secvențial) *costul / pierdere exponențială* (engl., the exponential loss).²¹⁸



$$J_T \stackrel{\text{def.}}{=} \frac{1}{m} \sum_{i=1}^m \exp(-y_i f_T(x_i)) \stackrel{\text{not.}}{=} \frac{1}{m} \sum_{i=1}^m \exp(-y_i \sum_{t=1}^T \alpha_t h_t(x_i)). \quad (88)$$

Aceasta revine la a spune că la fiecare iterație $t \in \{1, \dots, T\}$ urmărim să alegem — pe lângă cea mai bună ipoteză „slabă” h_t — o valoare pentru ponderea α_t astfel încât costul / „pierdere” J_t să fie minimizată.

În acest execuțiu veți se demonstrează că această [nouă] strategie va conduce la aceeași regulă de actualizare pentru ponderea α_t folosită în algoritmul AdaBoost, adică $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$.

a. Arătați că

$$J_t = c_t \cdot \underbrace{\sum_{i=1}^m D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i))}_{\text{not.: } \tilde{J}_t}$$

unde c_t este un factor constant, care nu depinde de α_t și nici de h_t , dar pe care îl veți preciza, iar

$$\tilde{J}_t = (1 - \varepsilon_t) \cdot e^{-\alpha_t} + \varepsilon_t \cdot e^{\alpha_t},$$

²¹⁸Pentru o definiție [de lucru] pentru funcțiile de cost / pierdere, vedeti *Explicația* de la problema 29.

Veți observa că pentru orice h_t și α_t fixați, $J_t \geq err_S(H_t) \stackrel{\text{def.}}{=} \frac{1}{m} \sum_{i=1}^m 1_{\{y_i \neq \text{sign}(f_t(x_i))\}}$. Pentru aceasta, analizați separat cazurile $y_i f_t(x_i) < 0$ și $y_i f_t(x_i) \geq 0$. Pentru a minimiza $err_S(H_t)$, aici vom urmări (în principal) să minimizăm J_t în funcție de α_t .

unde ε_t este eroarea pondertă la antrenare pentru ipoteza „slabă“ h_t .

Sugestie: Veți putea folosi rezultatul obținut la problema 23.a, și anume că $D_t(i) \propto \exp(-y_i f_{t-1}(x_i))$, adică probabilitatea $D_t(i)$ este proporțională cu $\exp(-y_i f_{t-1}(x_i))$.²¹⁹

b. Arătați că dacă se consideră $\alpha_t > 0$ fixat, atunci pentru a minimiza expresia / valoarea lui J_t trebuie să alegem dintre toate ipotezele „slabe“ pe aceea (notată cu h_t) pentru care eroarea ponderată la antrenare (notată cu ε_t) este minimă.

c. Arătați că dacă se consideră h_t fixat, atunci minimizarea lui J_t revine la a atribui lui α_t valoarea $\ln \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}$.

d. Rescrieți pseudo-codul algoritmului AdaBoost ca un algoritm de *optimizare secvențială* a costului [negativ] exponențial al clasificării instanțelor de antrenament.

e. Dacă notăm cu J_t^* valoarea optimă lui J_t (văzut ca funcție de ipoteza „slabă“ h_t și votul $\alpha_t > 0$), ce relație există în general între J_t^* și J_{t-1}^* pentru $t > 1$?

Dar între J_t^* pe de o parte și Z_1, Z_2, \dots, Z_{t-1} , pe de altă parte?

Răspuns:

a. La iterația t vom avea:

$$\begin{aligned} J_t &= \frac{1}{m} \sum_{i=1}^m \exp(-y_i f_t(x_i)) = \frac{1}{m} \sum_{i=1}^m \exp \left(-y_i \left(\sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x_i) \right) - y_i \alpha_t h_t(x_i) \right) \\ &= \frac{1}{m} \sum_{i=1}^m \exp(-y_i f_{t-1}(x_i)) \cdot \exp(-y_i \alpha_t h_t(x_i)) \\ &= \frac{1}{m} \sum_{i=1}^m \left(m \prod_{t'=1}^{t-1} Z_{t'} \right) \cdot D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \\ &= \left(\prod_{t'=1}^{t-1} Z_{t'} \right) \cdot \sum_{i=1}^m D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \end{aligned} \quad (89)$$

$$\begin{aligned} &\propto \sum_{i=1}^m D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \stackrel{\text{not.}}{=} \tilde{J}_t. \end{aligned} \quad (90)$$

Mai departe, ținând cont de faptul că $y_i \in \{-1, +1\}$ și $h_t(x_i) \in \{-1, +1\}$ pentru $i = 1, \dots, m$ și $t = 1, \dots, T$, putem scrie expresia \tilde{J}_t astfel:

$$\begin{aligned} \tilde{J}_t &= \sum_{i=1}^m D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \\ &= \sum_{i \in C} D_t(i) \exp(-\alpha_t) + \sum_{i \in M} D_t(i) \exp(\alpha_t) \\ &= \exp(-\alpha_t) \underbrace{\sum_{i \in C} D_t(i)}_{1 - \varepsilon_t} + \exp(\alpha_t) \underbrace{\sum_{i \in M} D_t(i)}_{\varepsilon_t} \\ &= (1 - \varepsilon_t) \cdot e^{-\alpha_t} + \varepsilon_t \cdot e^{\alpha_t}, \end{aligned} \quad (91)$$

²¹⁹Mai exact, $D_t(i) = \frac{1}{mZ_1 \dots Z_{t-1}} \exp(-y_i f_{t-1}(x_i))$.

unde, ca și la problema 22, C este mulțimea [indicilor] exemplelor de antrenament care sunt corect clasificate de către ipoteza h_t , iar M este mulțimea [indicilor] exemplelor de antrenament care sunt incorect clasificate de către h_t .

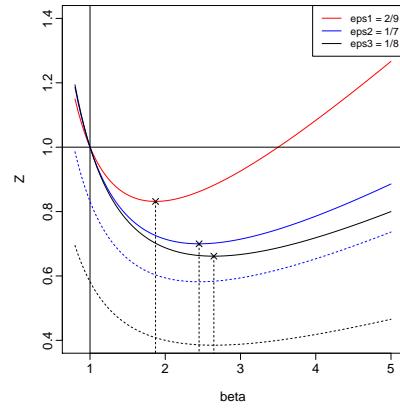
b. Dacă rescriem expresia (91) sub forma

$$e^{-\alpha_t} + \varepsilon_t \underbrace{(e^{\alpha_t} - e^{-\alpha_t})}_{>0},$$

se observă că dacă fixăm α_t atunci a minimiza expresia J_t în raport cu h_t revine la a minimiza ε_t (care nu depinde de α_t).

c. Expresia (91) este identică cu expresia (82) pentru factorul de normalizare Z_t de la problema 22.i.²²⁰ Acolo am arătat că expresia (82) își atinge minimul pentru $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$. Prin urmare, ținând cont de faptul că ε_t se consideră fixat, „costul“ J_t va fi minimizat pentru același $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$.

Exemplificare: În graficul alăturat, realizat de către Sebastian Ciobanu, sunt reprezentate funcțiile \tilde{J}_t și J_t corespunzătoare celor trei iterări realizate de către algoritmul AdaBoost, care au fost descrise la problema 24. Pentru conveniență, am folosit notația $\beta = e^\alpha$ și am reprezentat funcțiile în raport cu acest argument. În acest grafic, liniile continue corespund funcțiilor \tilde{J}_t , iar liniile discontinue corespund funcțiilor J_t asociate lor: $J_t(\beta) = (\prod_{t'=1}^{t-1} Z_{t'}) \cdot \tilde{J}_t(\beta)$. Remarcați că din relația (89) rezultă că $J_1(\beta) = \tilde{J}_1(\beta), \forall \beta$.



d. Ca o consecință a punctelor b și c , algoritmul AdaBoost poate fi reformulat ca un algoritm de *optimizare secvențială* a costurilor / „pierderilor“ J_t : La fiecare iterație a algoritmului se adaugă în mod „greedy“ căte o ipoteză „slabă“ $h \in \mathcal{H}$ la ipoteza combinată curentă pentru a minimiza costul determinat de funcția ϕ , și cu *obiectivul* final de a minimiza eroarea la antrenare $err_S(H_T)$. (De aceea această metodă de optimizare se numește *optimizare secvențială*.) Iată care este noua formulare a algoritmului AdaBoost:

Intrare: $\{(x_i, y_i)\}_{i=1,\dots,m}$ – setul de date de antrenament, T – numărul de iterării de executat, \mathcal{H} – setul de ipoteze „slabe“, $\phi(y, y') \stackrel{\text{def.}}{=} \exp(-yy')$ – funcția de cost / pierdere.

Procedura:

```

Initializare:  $f_0(x) = 0$  și  $D_1(i) = 1/m$  pentru  $i = 1, \dots, m$ 
pentru  $t$  de la 1 la  $T$  execută:
begin
  1. Calculează

```

²²⁰[S. Ciobanu:] De fapt, se putea vedea chiar din relația (90) că $\tilde{J}_t = Z_t$ (văzut ca funcție de α_t), ținând cont de faptul că Z_t este factor de normalizare în definiția distribuției D_{t+1} (vedeți relația (81)).

$$(h_t, \alpha_t) = \arg \min_{h \in \mathcal{H}, \alpha \in \mathbb{R}_+} \underbrace{\sum_{i=1}^m \phi(y_i, f_{t-1}(x_i) + \alpha h(x_i))}_{J_t(h, \alpha)}$$

2. Actualizează clasificatorul $f_t(x) = f_{t-1}(x) + \alpha_t h_t(x)$
și calculează noua distribuție, D_{t+1}
end

returnează clasificatorul $\text{sign}(f_T(x))$

Precizăm faptul că minimizarea efectuată la pasul 1 din corpul iterativ al acestui algoritm este făcută *pe coordonate*. Aceasta înseamnă că mai întâi minimizăm funcția J_t în raport cu argumentul h (care nu depinde de α) și apoi — cu h fixat la valoarea pe care tocmai am găsit-o, h_t — minimizăm funcția J_t în raport cu argumentul α (care depinde de h_t).

e. Valorile minime ale funcțiilor J_t nu cresc niciodată de la o iterare la alta, fiindcă $\min_{h \in \mathcal{H}, \alpha \in \mathbb{R}_+} J_t(h, \alpha) \leq \underbrace{\phi(y_i, f_{t-1}(x_i))}_{J_t(-, 0)}$. Introducând notația $J_t^* = \min_{h \in \mathcal{H}, \alpha \in \mathbb{R}_+} J_t(h, \alpha)$,

putem scrie aceasta inegalitate sub forma $J_t^* \leq J_{t-1}^*, \forall t \geq 1$. Din relația (89) putem deduce imediat că $J_t^* = \prod_{t'=1}^{t-1} Z_{t'}$, fiindcă expresia (91) are valoarea 1 atunci cînd îi atribuim lui α_t valoarea $\frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$.²²¹

Observație importantă: La problema 29 vom arăta că putem adapta algoritmul nou introdus astfel încât să lucrăm cu orice funcție de cost dorim în locul funcției de cost [negativ] exponențiale ϕ .

27.

(Noțiunea de *margine de votare* în conexiune cu algoritmul AdaBoost;
o referire la chestiunea overfitting-ului)
prelucrare de Liviu Ciortuz, după
■ CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.3

Deși la aplicarea algoritmului AdaBoost *complexitatea modelului* produs crește la fiecare iterare, *în general nu se produce overfitting*. Motivul este că modelul capătă din ce în ce mai multă „încredere“ pe măsură ce numărul de iterări executate crește. Această „încredere“ (engl., confidence) poate fi exprimată din punct de vedere matematic cu ajutorul noțiunii de *margine de votare* (engl., voting margin).²²²

Vă readucem aminte că,²²³ după efectuarea celor T iterării, algoritmul AdaBoost livrează la ieșire clasificatorul

$$H_T(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right).$$

În mod similar, putem defini *clasificatorul ponderat intermediu* (engl., intermediate weighted classifier) după k iterării:

$$H_k(x) = \text{sign}\left(\sum_{t=1}^k \alpha_t h_t(x)\right).$$

²²¹LC: Multumesc lui Sebastian Ciobanu pentru dialogul în urma căruia am identificat această proprietate.

²²²În mod concret, „încrederea“ în clasificarea lui x_i la iterarea t va fi definită ca $|\text{Margin}_k(x_i)|$, unde $\text{Margin}_k(x_i)$ urmează să fie definită mai jos.

²²³Vedeți enunțul problemei 22.

Întrucât outputul acestei funcții este $+1$ sau -1 , aceasta nu ne comunică nimic despre încrederea [pe care o putem avea] în deciziile acestui clasificator. De aceea, fără a afecta regula de decizie în sine, o putem redefini sub forma

$$H_k(x) = \text{sign}\left(\sum_{t=1}^k \bar{\alpha}_t h_t(x)\right),$$

unde $\bar{\alpha}_t = \frac{\alpha_t}{\sum_{t'=1}^k \alpha_{t'}}$, deci ponderile (engl., weights) ipotezelor „slabe“ sunt acum normalizate, în sensul că $\sum_{t=1}^k \bar{\alpha}_t = 1$ și, desigur, $\bar{\alpha}_t > 0$.²²⁴

Marginea de votare a instanței de antrenament x_i după iterată k se definește ca fiind diferența dintre suma ponderilor / voturilor normalizate ale acelor h_t — evident, $t \in \{1, \dots, k\}$ — care clasifică corect instanța x_i și suma ponderilor / voturilor normalizate ale acelor h_t care îl clasifică incorrect pe x_i .²²⁵ Așadar,

$$\text{Margin}_k(x_i) = \sum_{t:h_t(x_i)=y_i} \bar{\alpha}_t - \sum_{t:h_t(x_i) \neq y_i} \bar{\alpha}_t.$$

Observație: Este imediat că $\text{Margin}_k(x_i) \in [-1, 1]$, pentru orice instanță x_i , cu $i = 1, \dots, m$ și pentru orice iterată k a algoritmului AdaBoost.

a. Fie $f_k(x) \stackrel{\text{not.}}{=} \sum_{t=1}^k \bar{\alpha}_t h_t(x)$. Arătați că $\text{Margin}_k(x_i) = y_i f_k(x_i)$ pentru orice instanță de antrenament x_i , cu $i = 1, \dots, m$.

Așadar, avem în acest fel o legătură foarte semnificativă între noțiunea de *margine de votare* nou-introdusă și noțiunea de *margine algebrică* introdusă la rezolvarea problemei 23.a: atunci când toate voturile α_t sunt normalizate, noțiunea de *margine de votare* coincide cu noțiunea de *margine algebrică*. (De aceea, în cele ce urmează, atunci când nu este niciun pericol de confuzie ne vom referi la aceste două noțiuni pur și simplu prin termenul *margine*.)

Consecință imediată: x_i , o instanță de antrenament oarecare, este corect clasificată de către ipoteza combinată H_k produsă de către AdaBoost la iterată k dacă și numai dacă $\text{Margin}_k(x_i) \geq 0$.

b. Dacă $\text{Margin}_k(x_i) > \text{Margin}_k(x_j)$, care dintre probabilitățile asociate celor două instanțe va fi mai mare la iterată $k + 1$ (adică, $D_{k+1}(i)$ sau $D_{k+1}(j)$)?

Sugestie: Din relația $D_{k+1}(i) = \frac{1}{m \cdot \prod_{t=1}^k Z_t} \cdot \exp(-y_i \cdot \sum_{t=1}^k \alpha_t \cdot h_t(x_i))$,²²⁶ care a fost demonstrată la problema 23.a, putem deduce $D_{k+1}(i) \propto \exp(-y_i f_k(x_i))$.²²⁷

Răspuns:

²²⁴Aceasta normalizare, referitoare la ponderile α_t (asociate ipotezelor „slabe“ h_t), nu trebuie confundată cu normalizarea sau, de fapt, normalizările reprezentate de factorii Z_t din pseudocodul algoritmului AdaBoost — vedeti enunțul problemei 22 —, care se referă la distribuțiile probabiliste D_t (asociate, la fiecare iterată, setului de instanțe de antrenament).

²²⁵Noțiunea de *margine geometrică* (nu ca aici) este specifică clasificatorului SVM. Vedeti capitolul *Mașini cu vectori-suport*.

²²⁶Atenție! Spre deosebire de definiția funcției f_k care a fost dată la punctul a, definiția funcției f_k de la problema 22 nu include normalizarea „voturilor“ α_t .

²²⁷Observați că o sugestie similară a fost făcută și la finalul enunțului problemei 26.

a. Vom demonstra egalitatea cerută pornind de la termenul din partea dreaptă:

$$\begin{aligned} y_i f_k(x_i) &= y_i \sum_{t=1}^k \bar{\alpha}_t h_t(x_i) = \sum_{t=1}^k \bar{\alpha}_t y_i h_t(x_i) = \sum_{t:h_t(x_i)=y_i} \bar{\alpha}_t - \sum_{t:h_t(x_i)\neq y_i} \bar{\alpha}_t \\ &= Margin_k(x_i). \end{aligned}$$

b. Conform relației pe care tocmai am demonstrat-o la punctul precedent, inegalitatea $Margin_k(x_i) > Margin_k(x_j)$ este echivalentă cu $y_i f_k(x_i) > y_j f_k(x_j)$. La rândul ei, aceasta din urmă este echivalentă cu $-y_i f_k(x_i) < -y_j f_k(x_j)$, deci și cu $\exp(-y_i f_k(x_i)) < \exp(-y_j f_k(x_j))$. Înănd cont de Sugestia din enunț, rezultă imediat că $D_{k+1}(i) < D_{k+1}(j)$.

Așadar, se verifică *intuiția* conform căreia este natural ca instanțele care sunt [cel] mai bine clasificate, deci care au o margine [mai] mare la o anumită iterare, să primească la iterarea următoare o probabilitate mai mică, algoritmul AdaBoost concentrându-se asupra instanțelor care sunt incorrect (sau, mai puțin bine) clasificate.

Observație importantă: Se poate constata practic²²⁸ că în cursul execuției sale, algoritmul AdaBoost trebuie să mărească per ansamblu, de la o iterare la alta, marginile corespunzătoare exemplelor de antrenament,²²⁹ iar o margine mai mare [pentru aceste exemple] implică în mod obișnuit o eroare la testare / generalizare mai mică. Aceasta este o explicație pentru faptul că, deși numărul de „parametri“ ai modelului rezultat crește cu 2 la fiecare iterare a algoritmului AdaBoost (și, deci, complexitatea crește),²³⁰ în general acest algoritm *nu produce overfitting*.²³¹

28. (AdaBoost: o condiție suficientă pentru învățabilitate γ -slabă:
marginea de votare, pentru orice exemplu de antrenament,
să fie de cel puțin 2γ , la fiecare iterare a algoritmului AdaBoost)
■ prelucrare de Liviu Ciortuz, după
CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.1.4

Introducere: La problema 23.e am făcut cunoștință cu noțiunea de *învățabilitate empirică γ -slabă* (engl., empirical γ -weak learnability). Concret, am demonstrat acolo că eroarea la antrenare produsă de algoritmul AdaBoost descrește [rapid] la 0 atunci când există $\gamma > 0$ astfel încât $\gamma \leq \gamma_t$ pentru orice t , unde $\gamma_t \stackrel{\text{def.}}{=} \frac{1}{2} - \varepsilon_t$, iar ε_t este eroarea ponderată la antrenare a ipotezei „slabe“ h_t .²³² În acest exercițiu vom demonstra o nouă *condiție suficientă* pentru ca să aibă loc învățabilitatea empirică γ -slabă, făcând apel la noțiunea de

²²⁸Vedeți de exemplu problemele de tip implementare MIT, 2001 fall, Tommi Jaakkola, HW3, pr. 2.4 și MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.4.

²²⁹Acest fapt se poate deduce din relația (88) de la problema 26, care dă expresia *costului* de minimizat de către algoritmul AdaBoost. Concret, se minimizează $E_T = \sum_{i=1}^m \exp(-y_i f_T(x_i))$ implicață, în idee, a maximiza, pe cât posibil, fiecare *margine* $y_i f_T(x_i)$.

²³⁰Cei doi parametri adăugați la fiecare iterare a algoritmului AdaBoost sunt unul pentru identificarea ipotezei h_t (de exemplu, în cazul compașilor de decizie, pragul corespunzător, s_t), iar celălalt ponderea α_t .

²³¹Există totuși situații în care algoritmul AdaBoost produce overfitting. Vedeți de exemplu CMU, 2011 fall, Eric Xing, HW5, pr. 3.2.cd.

²³²Totuși, această condiție nu este satisfăcută întotdeauna. Vedeți, de exemplu, soluția a două de la problema 25.b (adică soluția care a fost obținută fără a folosi prag exterior).

margine de votare (engl., voting margin), care a fost prezentată la problema 27. Concret, vom arăta că atunci când există $\theta > 0$ astfel încât marginile de votare ale instanțelor de antrenament sunt mărginită inferior de θ la fiecare iterare a algoritmului AdaBoost,²³³ proprietatea de învățabilitate empirică γ -slabă este „garantată”, cu $\gamma = \theta/2$.

Presupunem că dispunem de setul de date de antrenament $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ și că există ipotezele „slabe” h_1, \dots, h_k din spațiul de ipoteze \mathcal{H} , precum și coeficienții nenegativi $\alpha_1, \dots, \alpha_k$ cu proprietatea $\sum_{j=1}^k \alpha_j = 1$ și, de asemenea, că există $\theta > 0$ astfel încât

$$y_i \left(\underbrace{\sum_{j=1}^k \alpha_j h_j(x_i)}_{f_k(x_i)} \right) \geq \theta \text{ pentru } \forall (x_i, y_i) \in S. \quad (92)$$

Observații:

1. În termenii problemei 27.b, inegalitatea de mai sus se poate scrie sub forma $y_i f_k(x_i) \geq \theta$ pentru $i = 1, \dots, m$ și, mai departe, ca $\text{Margin}_k(x_i) \geq \theta$ pentru $i = 1, \dots, m$.
2. Știm din enunțul problemei 27 că $\text{Margin}_k(x_i) \in [-1, 1]$, deci ipoteza $\text{Margin}_k(x_i) > \theta > 0$ implică $\theta \in (0, 1]$ și, în consecință, $\frac{\theta}{2} \in (0, 1/2]$.

Vom demonstra că atunci când condiția (92) este satisfăcută, pentru orice distribuție probabilistă D peste setul de date de antrenament S există [cel puțin] o ipoteză $h_l \in \{h_1, \dots, h_k\}$ pentru care eroarea ponderată la antrenare în raport cu distribuția D este de cel mult $\frac{1}{2} - \frac{\theta}{2}$, adică $\varepsilon_l \leq \frac{1}{2} - \frac{\theta}{2}$.²³⁴ Demonstrația va fi făcută în doi pași, care corespund punctelor a și b care urmează.

a. Arătați că dacă $\text{Margin}_k(x_i) \geq \theta$ pentru $i = 1, \dots, m$, iar D este o distribuție probabilistă oarecare definită pe S , atunci

- $E_{i \sim D}[y_i f_k(x_i)] \geq \theta$, adică media [ponderată a] marginilor algebrice pentru instanțele x_1, \dots, x_m este mărginită și ea, inferior, de θ ,
- și, în consecință, există [cel puțin] o ipoteză „slabă” h_l din mulțimea $\{h_1, \dots, h_k\}$ astfel încât $E_{i \sim D}[y_i h_l(x_i)] \geq \theta$.

Sugestie: Țineți cont de faptul că, dată fiind o inegalitate, dacă aplicăm operatorul $E[\cdot]$ (care desemnează media probabilistă) la ambele termeni ai inegalității respective, atunci inegalitatea se menține.

b. Arătați că inegalitatea $E_{i \sim D}[y_i h_l(x_i)] \geq \theta$ este echivalentă cu inegalitatea $\text{err}_D(h_l) \leq \frac{1}{2} - \frac{\theta}{2}$, unde, ca și la problema 22, $\text{err}_D(h_l) \stackrel{\text{not.}}{=} \Pr_{i \sim D}[y_i \neq h_l(x_i)]$ desemnează eroarea ponderată la antrenare a ipotezei „slabe” h_l în raport cu distribuția probabilistă D .

Răspuns:

²³³De fapt, este suficient ca această condiție ($\text{Margin}_k(x_i) \geq \theta$ pentru $i = 1, \dots, m$) să fie satisfăcută de la o anumită iterare t_0 încolo.

²³⁴Așadar, va rezulta că $\gamma_t \stackrel{\text{def.}}{=} \frac{1}{2} - \varepsilon_t \geq \frac{1}{2} - \frac{1}{2} + \frac{\theta}{2} = \frac{\theta}{2}$. Altfel spus, setul de date de antrenament S este γ -slab învățabil [în sens empiric], cu $\gamma = \frac{\theta}{2}$.

a. Pe de o parte, inegalitățile $y_i \left(\sum_{j=1}^k \alpha_j h_j(x_i) \right) \geq \theta$ pentru $i = 1, \dots, m$, rescrite în mod echivalent sub forma $y_i f_k(x_i) \geq \theta$ pentru $i = 1, \dots, m$, implică imediat prin aplicarea operatorului $E_{i \sim D}[\cdot]$ (care desemnează media probabilistă, în funcție de distribuția D), următoarea inegalitate:²³⁵

$$E_{i \sim D}[y_i f_k(x_i)] \geq \theta,$$

care se rescrică echivalent sub forma

$$\sum_{i=1}^m y_i h_l(x_i) \cdot D(i) \geq \theta. \quad (93)$$

Acum vom demonstra — folosind *metoda reducerii la absurd* — că există cel puțin un indice $l \in \{1, \dots, k\}$ astfel încât $E_{i \sim D}[y_i h_l(x_i)] \geq \theta$. Presupunem că pentru orice $l = 1, \dots, k$ are loc inegalitatea $E_{i \sim D}[y_i h_l(x_i)] < \theta$, adică $\sum_{i=1}^m y_i h_l(x_i) \cdot D(i) < \theta$. Prin înmulțirea ambilor termeni ai acestei inegalități cu $\alpha_l > 0$ rezultă

$$\sum_{i=1}^m y_i h_l(x_i) \cdot D(i) \cdot \alpha_l < \theta \cdot \alpha_l \text{ pentru } l = 1, \dots, k.$$

Însumând membru cu membru aceste inegalități pentru $l = 1, \dots, k$, rezultă

$$\begin{aligned} \sum_{l=1}^k \sum_{i=1}^m y_i h_l(x_i) \cdot D(i) \cdot \alpha_l &< \sum_{l=1}^k \theta \cdot \alpha_l \Leftrightarrow \sum_{i=1}^m y_i D(i) \left(\sum_{l=1}^k h_l(x_i) \alpha_l \right) < \theta \sum_{l=1}^k \alpha_l \Leftrightarrow \\ &\sum_{i=1}^m y_i f_k(x_i) \cdot D(i) < \theta, \end{aligned} \quad (94)$$

fiindcă $\sum_{l=1}^k \alpha_l = 1$ și $f_k(x_i) \stackrel{\text{not.}}{=} \sum_{l=1}^k \alpha_l h_l(x_i)$.

Inegalitatea (94) se rescrică sub forma $E_{i \sim D}[y_i f_k(x_i)] < \theta$. Evident, aceasta contrazice relația (93). Prin urmare, presupunerea făcută anterior este falsă. Rezultă că există $l \in \{1, \dots, k\}$ astfel încât $E_{i \sim D}[y_i h_l(x_i)] \geq \theta$.

b. După cum am menționat deja la rezolvarea punctului a, inegalitatea $E_{i \sim D}[y_i h_l(x_i)] \geq \theta$ se rescrică în mod echivalent sub forma $\sum_{i=1}^m y_i h_l(x_i) \cdot D(i) \geq \theta$. Înținând cont de faptul că $y_i \in \{-1, +1\}$ și $h_l(x_i) \in \{-1, +1\}$ pentru $i = 1, \dots, m$ și $l \in \{1, \dots, k\}$, putem scrie următorul sir de echivalențe:²³⁶

²³⁵În acest context, aplicarea operatorului $E_{i \sim D}$ revine practic la înmulțirea fiecărei inegalități $y_i f_k(x_i) \geq \theta$ cu $D(i) \geq 0$, pentru $i = 1, \dots, m$, urmată de însumarea acestor inegalități membru cu membru:

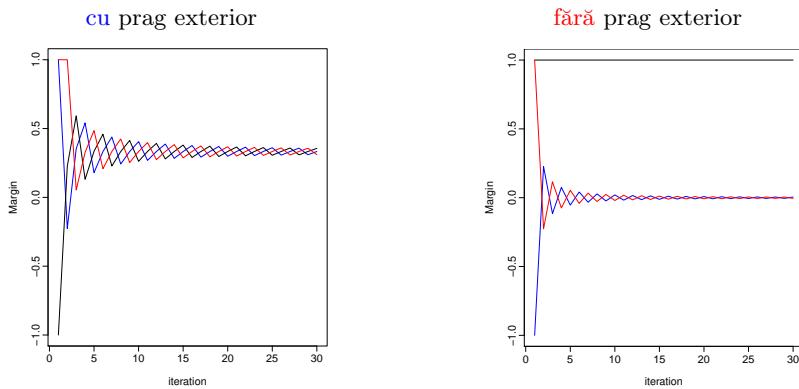
$$\sum_{i=1}^m y_i f_k(x_i) \cdot D(i) \geq \sum_{i=1}^m \theta \cdot D(i) = \theta \underbrace{\sum_{i=1}^m \theta D(i)}_1 = \theta$$

²³⁶Ca și la problema 22, vom nota cu ε_l eroarea ponderată produsă la antrenare de către ipoteza h_l .

$$\begin{aligned}
\sum_{i=1}^m y_i h_l(x_i) \cdot D(i) \geq \theta &\Leftrightarrow \\
\sum_{i:y_i=h_l(x_i)} D(x_i) - \sum_{i:y_i \neq h_l(x_i)} D(x_i) &\geq \theta \Leftrightarrow \\
(1 - \varepsilon_l) - \varepsilon_l \geq \theta &\Leftrightarrow 1 - 2\varepsilon_l \geq \theta \Leftrightarrow 2\varepsilon_l \leq 1 - \theta \Leftrightarrow \varepsilon_l \leq \frac{1}{2} - \frac{\theta}{2} \Leftrightarrow \\
&\stackrel{\text{def.}}{\Leftrightarrow} \text{err}_D(h_l) \leq \frac{1}{2} - \frac{\theta}{2}.
\end{aligned}$$

În consecință, o condiție suficientă pentru a asigura γ -învățabilitate slabă pe un dataset de antrenament S este ca marginea de votare să fie de cel puțin $2\gamma \stackrel{\text{not.}}{=} \theta$, pentru orice exemplu de antrenament, la fiecare iterare a algoritmului AdaBoost (de fapt, este destul ca această condiție să fie îndeplinită de la o anumită iterare t_0 încolo).²³⁷

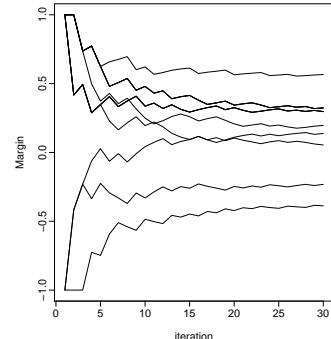
Exemplificare: Următoarele grafice, făcute de Sebastian Ciobanu, pun în evidență evoluția marginilor de votare pentru datele de la problema 25, pentru fiecare dintre cele două variante de rezolvare care au fost prezentate acolo:



Se constată ușor că în prima variantă (și anume, când se folosește prag exterior) există un prag inferior θ pentru toate marginile de votare, începând chiar cu una din primele iterării. În schimb, în a doua variantă (și anume, când nu se folosește prag exterior), este evident că nu va exista un prag inferior θ pentru [toate] marginile de votare, întrucât la fiecare iterare una dintre instanțele $x_1 = 1$ și $x_3 = 5$ este clasificată eronat. (Se observă însă că instanța $x_2 = 3$ este clasificată corect la fiecare iterare, ba chiar cu „maximum“ de voturi, de aceea marginea ei de votare este întotdeauna 1.)

²³⁷ Observați că această condiție suficientă este mai „tare“ / restrictivă decât condiția pentru învățabilitate γ -slabă de la problema 23.e, fiindcă prezenta condiție o implică pe cea de la problema 23.e.

Graficul alăturat, făcut tot de Sebastian Ciobanu, prezintă evoluția marginilor de votare pentru datele de la problema 60, pentru care se poate arăta că nu admite [nici măcar direct, adică în maniera de la problema 23.e] garanție de învățabilitate γ -slabă.



Observație: Știm de la problema 27 că existența pragului inferior $\theta > 0$ pentru $\text{Margin}_t(x_i)$, pentru orice $i = 1, \dots, m$, implică faptul că toate instanțele de antrenament x_i sunt corect clasificate (de către ipoteza combinată $H_S(t)$) la iterarea t . Totuși, asta nu înseamnă că toate instanțele vor fi corect clasificate și la iterarea următoare (de către $H_S(t+1)$), fiindcă eroarea la antrenare $\text{errs}(H_t)$ nu este în mod neapărat descrescătoare (în raport cu t).²³⁸ Însă, dacă pragul inferior θ pentru $\text{Margin}_t(x_i)$ se menține pentru orice $i = 1, \dots, m$ și pentru orice $t = t_0, t_0 + 1, \dots, t_0 + T$, unde T se calculează ca la problema 59, atunci avem certitudinea că eroarea la antrenare $\text{errs}(H_t)$ va rămâne 0 pentru orice $t \geq t_0 + T$.²³⁹

29.

(Generalizarea algoritmului AdaBoost,
pentru diverse funcții de pierdere / cost)

■ prelucrare de Liviu Ciortuz, după
MIT, 2003 fall, Tommi Jaakkola, HW4, pr. 2.1-3

În această problemă vom deriva un *algoritm de boosting, mai general* decât algoritmul AdaBoost din problema 22.²⁴⁰ Noul algoritm va putea fi aplicat la o întreagă clasă de *funcții de cost / pierdere* (engl., loss functions), în particular pentru funcția de cost [negativ] exponențială.

Scopul nostru este să generăm *funcții de discriminare* (engl., discriminant functions) de forma următoare:

$$f_T(x) = \alpha_1 h(x; \theta_1) + \dots + \alpha_T h(x; \theta_T).$$

În această expresie, $x \in \mathbb{R}^d$, iar θ_i este un parametru sau set / vector de parametri (tot din un spațiu de tip \mathbb{R}^n), depinzând de tipul ipotezelor „slabe“ h . În continuare veți putea presupune că aceste ipoteze $h(x; \theta_i)$ sunt compași de decizie (engl., decision

²³⁸Vedeți de exemplu problema de tip implementare de la Stanford, 2016 fall, A. Ng, J. Duchi, HW2, pr. 6.d.

²³⁹Chiar dacă această condiție — există $\theta > 0$ astfel încât $\text{Margin}_t(x_i) \geq \theta$ pentru orice $i = 1, \dots, m$ și pentru orice t de la un t_0 încolo — nu este satisfăcută, totuși studiul evoluției marginilor de votare are o mare valoare informativă. (Vedeți de exemplu problemele de tip implementare MIT, 2001 fall, Tommi Jaakkola, HW3, pr. 2.4 și MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.4.) La capitolul *Masini cu vectori-suport* (SVM) se arată că pentru o altă categorie de margini — și anume, *margini geometrice* —, se poate formula în mod riguros o *problemă de optimizare* care se pretează (inclusiv) la tratarea *excepțiilor* de la clasificare. În urma rezolvării acelei probleme de optimizare se obține un clasificator / separator care este foarte convenabil, din multe puncte de vedere.

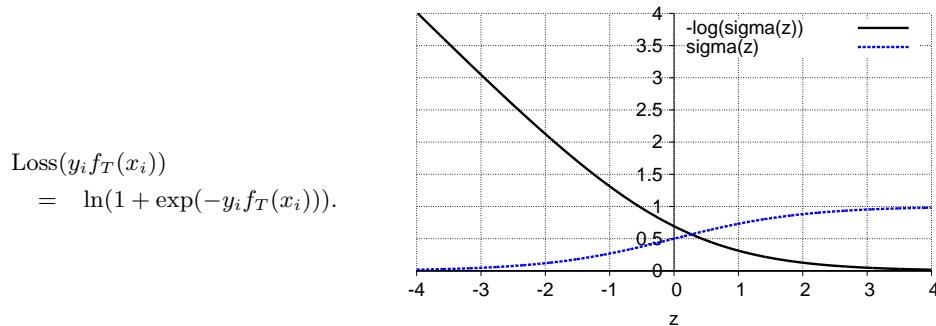
²⁴⁰Sămânța pentru acest nou algoritm a fost sădită la *Observația importantă* de la finalul rezolvării problemei 26.

stumps), ale căror predicții pot fi $+1$ sau -1 . Orice alte categorii de ipoteze „slabe“ vor putea fi folosite în acest cadru. Vom adăuga în mod *secvențial* componente la funcția generală de discriminare, într-o manieră care va separa (atât cât este posibil) *estimarea* parametrilor θ ai ipotezelor „slabe“ de *setarea* voturilor / ponderilor α .

Explicație: Vom începe prin a defini noțiunea de *funcție de cost / pierdere*, pe care o vom folosi mai jos.²⁴¹ Singurele *restricții* pe care o astfel de funcție va trebui să le îndeplinească sunt următoarele: *i.* să fie nenegativă, *ii.* să fie *monoton descrescătoare* și *iii.* să fie *derivabilă*.²⁴² În contextul nostru, argumentul unei funcții de pierdere va fi expresia $y_i f_T(x_i)$, care reprezintă o măsură a acordului dintre eticheta y_i asociată instanței de antrenament x_i pe de o parte și [semnul dat de] valoarea funcției de discriminare f_T pe de altă parte. Vrem ca, pentru fiecare instanță x_i , „pierderea“ să fie cu atât mai mică cu cât valoarea funcției de discriminare este mai mult în acord cu valoarea ± 1 a etichetei y_i (adică, cu cât *marginea algebrică* $y_i f_T(x_i)$ are o valoare mai mare). Funcția de pierdere [negativ] *exponențială* pe care am întâlnit-o deja la problema 23, adică

$$\text{Loss}(y_i f_T(x_i)) = \exp(-y_i f_T(x_i))$$

satisfac cele două restricții pe care tocmai le-am menționat mai sus. La fel și *funcția [de pierdere] logistică*:²⁴³



Observație (legătura cu regresia logistică): Funcția de cost / pierdere logistică are o interpretare interesantă, și anume [valoarea ei este] negativul unei log-probabilități. Într-adevăr, [vă reamintim că] pentru un model aditiv de *regresie logistică*,²⁴⁴ avem

$$-\ln P(y = 1|x, w) = -\ln \frac{1}{1 + \exp(-z)} = \ln(1 + \exp(-z)),$$

unde $z = w_1 \phi_1(x) + \dots + w_T \phi_T(x)$.²⁴⁵ Înlocuind combinația aditivă de *funcții de bază* ($\phi_i(x)$) cu combinația aditivă de ipoteze „slabe“ ($h(x; \theta_i)$), vom obține un *model aditiv de*

²⁴¹Veți vedea că definiția pe care o dăm aici pentru funcții de cost / pierdere diferă ușor de cea pe care am folosit-o la problema 18 de la capitolul *Metode de regresie*.

²⁴²Într-o acceptiune mai largă, condiția *ii.* din definiția funcției de pierdere / cost devine: să fie *convexă*. (Acestă acceptiune va fi folosită la punctul *c* al acestei probleme.)

²⁴³La problema 18 de la capitolul *Metode de regresie* am ilustrat / reprezentat pe un același grafic trei funcții de cost: funcția de cost [negativ] exponențială pentru AdaBoost, funcția de cost logistică pentru regresia logistică și funcția de cost *hinge* pentru mașini cu vectori-suport (SVM).

²⁴⁴Pentru o introducere succintă în chestiunea regresiei logistice, vedeți problema 12 de la capitolul *Metode de regresie*.

²⁴⁵Mentionăm că am omis termenul liber (engl., bias) w_0 — care apare la regresia logistică — din motive care ţin de simplitate.

regresie logistică, în care ipotezele „slabe“ operează ca funcții de bază. *Diferența* este că în cazul algoritmului AdaBoost [generalizat] vom „estima“ atât [parametrii pentru] funcțiile de bază (ipotezele „slabe“) cât și coeficienții cu care acestea vor fi înmulțite. În modelul [clasic] de regresie logistică avem de-a face în mod tipic cu un set fixat de funcții de bază.

Criteriul de estimare / evaluare pentru combinația de ipoteze „slabe“ este pur și simplu *costul / pierderea empirică* (engl., empirical loss), definită prin expresia

$$J_T = \frac{1}{m} \sum_{i=1}^m \text{Loss}(y_i f_T(x_i)),$$

unde sumarea se face parcurgând toate exemplele de antrenament disponibile.

Așadar, vrem să derivăm algoritmul de boosting în aşa fel încât să putem lucra cu *orice funcție de pierdere* de tipul discutat mai sus. În acest scop, la *iterația curentă* (t) presupunem că am inclus deja [în combinația aditivă] un număr de $t - 1$ componente (ipoteze „slabe“):

$$f_{t-1}(x) = \alpha_1 h(x; \hat{\theta}_1) + \dots + \alpha_{t-1} h(x; \hat{\theta}_{t-1}) \quad (95)$$

și că dorim să mai adăugăm încă una, $h(x; \theta)$. *Criteriul de estimare* pentru funcția de discriminare — inclusiv ultima componentă adăugată, al cărei „vot“ asociat este α —, este dat de expresia

$$J_t(\alpha, \theta) = \frac{1}{m} \sum_{i=1}^m \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \theta)).$$

Remarcați faptul că urmărim să explicăm / explorăm doar modul în care acest „obiectiv“ / criteriu depinde de alegerea ultimei componente ($h(x_i; \theta)$) și de votul corespunzător (α), fiindcă parametrii precedentelor $t - 1$ componente, împreună cu voturile asociate lor au fost deja setate și nu vor mai fi modificate.

Mai întâi vom urmări să identificăm noua componentă, mai precis spus valoarea parametrului θ în aşa fel încât să maximizăm potențialul ei (adică, al noii componente) de a reduce pierderea empirică — potențial în sensul că ulterior vom putea să ajustăm „votul“ asociat acestei componente, ca să reducem pierderea empirică. Concret, la *Pasul 1* al iterăției t vom seta θ astfel încât să minimizăm valoarea derivatei²⁴⁶

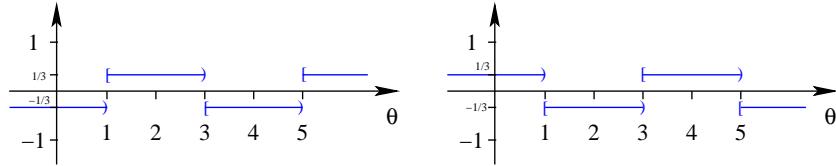
$$\begin{aligned} \frac{\partial}{\partial \alpha} J_t(\alpha, \theta) |_{\alpha=0} &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial \alpha} \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \theta)) |_{\alpha=0} \\ &= \frac{1}{m} \sum_{i=1}^m dL(y_i f_{t-1}(x_i)) y_i h(x_i; \theta), \end{aligned} \quad (96)$$

unde prin $dL(z)$ am notat derivata $\frac{\partial \text{Loss}(z)}{\partial z}$. (Am aplicat formula de derivare a funcțiilor compuse.)

Exemplificare: Graficele următoare, corespunzătoare datelor de la problema 25, reprezintă cele două variante ale expresiei / funcției (de argumentul θ) $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta) |_{\alpha=0}$ calculate

²⁴⁶Vom arăta că această minimizare (în funcție de θ) revine la identificarea ipotezei slabe cu cea mai bună eroare ponderată la antrenare (ε_t). Vedeți relația (98).

pentru iterația $t = 1$ al algoritmului AdaBoost folosind funcția de cost [negativ] exponentială. Aceste două variante corespund semnelor care pot fi alese pentru zonele de decizie determinate de compasul de decizie de prag (generic) θ : $+|-$ și respectiv $-|+$.



Remarcați faptul că expresia $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta)|_{\alpha=0}$ exprimă în mod precis cât de „repede“ (în funcție de θ) vom putea începe să reducem pierderea empirică dacă vom crește în mod gradual „votul“ (α) pentru noua componentă, $h(x; \theta)$. Minimizarea expresiei (96) în raport cu θ pare a fi o modalitate rezonabilă de „estimare“ a parametrului θ al noii componente, $h(x; \theta)$. Acest plan ne permite ca mai întâi să setăm valoarea lui θ și apoi, la *Pasul 2* al iterației t , să alegem valoarea lui α în aşa fel încât să minimizăm pierderea empirică.

Acum vom modifica ușor algoritmul schițat mai sus în aşa fel încât să semene mai mult cu un *algoritm de boosting* (pregătind elaborarea *Pasului 3* al iterației curente).²⁴⁷ Vom defini următoarele *ponderi* (engl., weights) și *ponderi normalizezate* asociate exemplelor de antrenament:²⁴⁸

$$\begin{aligned} W_i^{(t-1)} &= -dL(y_i f_{t-1}(x_i)), \text{ pentru } i = 1, \dots, m \text{ și} \\ \tilde{W}_i^{(t-1)} &= \frac{W_i^{(t-1)}}{\sum_{j=1}^m W_j^{(t-1)}}, \text{ pentru } i = 1, \dots, m. \end{aligned} \quad (97)$$

ACESTE PONDERI SUNT, ÎN MOD EVIDENT, NENEGATIVE, ÎNTRUCÂT FUNCȚIA DE COST / PIERDERE ESTE DESCRESCĂTOARE ȘI DERIVABILĂ (DECI DERIVATA EI TREBUIE SĂ FIE NEGATIVĂ SAU ZERO). *Intuiția* este că ponderea unei instanțe x_i este cu atât mai mică (respectiv mai mare) cu cât marginea algebraică $y_i f_{t-1}(x_i)$ este mai mare (respectiv mai mică), adică cu cât x_i este mai bine (respectiv mai prost) clasificat.

Revenind la relația (96), o vom putea rescrive — folosind notația (97) — sub forma următoare:

$$\begin{aligned} \frac{\partial}{\partial \alpha} J_t(\alpha, \theta)|_{\alpha=0} &= -\frac{1}{m} \sum_{i=1}^m W_i^{(t-1)} y_i h(x_i; \theta) \\ &= -\frac{1}{m} \left(\sum_j W_j^{(t-1)} \right) \cdot \sum_{i=1}^m \frac{W_i^{(t-1)}}{\sum_j W_j^{(t-1)}} y_i h(x_i; \theta) \\ &= -\frac{1}{m} \left(\sum_j W_j^{(t-1)} \right) \cdot \sum_{i=1}^m \tilde{W}_i^{(t-1)} y_i h(x_i; \theta). \end{aligned}$$

²⁴⁷Termenul de *boosting* se referă la faptul că instanțele incorect clasificate la o iterație (de către ipoteza „slabă“ aleasă la acea iterație) vor avea la iterația următoare probabilități mai mari, pentru ca algoritmul să se concentreze cu preponderență asupra lor.

²⁴⁸Ponderilor normalizezate $\tilde{W}_i^{(t-1)}$ de aici le corespund în pseudo-codul algoritmului AdaBoost dat la problema 22 probabilitățile $D_t(i)$. Este îndreptățit să folosim pentru $\tilde{W}_i^{(t-1)}$ și denumirea de probabilități (atenție!, pentru iterația t), fiindcă $\tilde{W}_i^{(t-1)} \in [0, 1]$ și $\sum_{i=1}^m \tilde{W}_i^{(t-1)} = 1$.

Întrucât factorul $\frac{1}{m} \sum_j W_j^{(t-1)}$ este constant în raport cu θ , la *Pasul 1* de la iterarea t vom estima valoarea lui θ *minimizând* valoarea expresiei

$$-\sum_{i=1}^m \tilde{W}_i^{(t-1)} y_i h(x_i; \theta).$$

Acest lucru este echivalent cu a minimiza eroarea ponderată la antrenare²⁴⁹

$$\varepsilon_t \stackrel{\text{not.}}{=} \sum_{i: y_i \neq h(x_i; \hat{\theta}_t)} \tilde{W}_i^{(t-1)} y_i h(x_i; \theta) \stackrel{\text{calcul}}{=} \frac{1}{2} \left(1 - \sum_{i=1}^m \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t) \right). \quad (98)$$

Acum putem începe să formulăm pașii [corfului iterativ al] noului algoritm de boosting într-un pseudo-cod similar cu algoritmul AdaBoost care a fost prezentat la problema 22.²⁵⁰

Pasul 1: Identifică o ipoteză / un clasificator $h(x; \hat{\theta}_t)$ care are o eroare ponderată la antrenare (engl., weighted training error) ε_t mai bună decât alegerea aleatorie, adică $\varepsilon_t < 1/2$, unde:

$$\varepsilon_t \stackrel{\text{not.}}{=} \sum_{i: y_i \neq h(x_i; \hat{\theta}_t)} \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t). \quad (99)$$

Pasul 2: Setează „votul” α_t pentru noua componentă, minimizând pierderea empirică totală (engl., overall empirical loss):

$$J_t(\alpha, \hat{\theta}_t) = \frac{1}{m} \sum_{i=1}^m \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \hat{\theta}_t)),$$

adică

$$\alpha_t = \arg \min_{\alpha > 0} J_t(\alpha, \hat{\theta}_t).$$

Se consideră că $f_0(x_i) = 0$ pentru $i = 1, \dots, m$.

²⁴⁹Notând, ca la problema 22, $C = \{i = 1, \dots, m \mid y_i = h(x_i; \hat{\theta}_t)\}$ și $M = \{i = 1, \dots, m \mid y_i \neq h(x_i; \hat{\theta}_t)\}$, este imediat că

$$\begin{aligned} \sum_{i=1}^m \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t) &= \sum_{i \in C} \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t) + \sum_{i \in M} \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t) \\ &= \underbrace{\sum_{i \in C} \tilde{W}_i^{(t-1)}}_{1 - \varepsilon_t} - \underbrace{\sum_{i \in M} \tilde{W}_i^{(t-1)}}_{\varepsilon_t} = 1 - 2\varepsilon_t \\ \Rightarrow \varepsilon_t &= \frac{1}{2} \left(1 - \sum_{i=1}^m \tilde{W}_i^{(t-1)} y_i h(x_i; \hat{\theta}_t) \right) \end{aligned}$$

²⁵⁰Pentru partea de inițializare, ca și în cazul algoritmului AdaBoost, vom folosi $\tilde{W}_i^{(0)} = \frac{1}{m}$, pentru $i = 1, \dots, m$. După aceea, corpul iterativ al algoritmului, format din pașii 1, 2 și 3, se execută pentru $t = 1, \dots, T$. Ca și în cazul algoritmului AdaBoost de la problema 22, algoritmul generalizat va fi oprit dacă la o iterație oarecare t nu există nicio ipoteză $h(x; \theta)$ având eroarea ponderată la antrenare $\varepsilon_t \in (0, 1/2)$.

Pasul 3: Recalculează ponderile normalizate pentru următoarea iterație, astfel:

$$\tilde{W}_i^{(t)} = -c_t \cdot dL(\underbrace{y_i f_{t-1}(x_i) + y_i \alpha_t h(x_i; \hat{\theta}_t)}_{y_i f_t(x_i)}) \text{ pentru } i = 1, \dots, m,$$

unde constanta c_t este aleasă astfel încât $\sum_{i=1}^m \tilde{W}_i^{(t)} = 1$.

- a. Arătați că, atunci când funcția de pierdere este cea [negativ] *exponențială* $\text{Loss}(z) = \exp(-z)$, cei trei pași din noul algoritm sunt în corespondență directă cu pașii algoritmului AdaBoost (de la problema 22). Mai concret, arătați că în acest caz setarea lui α_t (la Pasul 2) bazat pe noua ipoteză „slabă“ $h(x; \hat{\theta}_t)$, precum și actualizarea ponderii $\tilde{W}_i^{(t)}$ (la Pasul 3) conduc la un rezultat identic cu AdaBoost. (După cum am precizat deja, în problema 22 corespondentul lui $\tilde{W}_i^{(t)}$ este $D_{t+1}(i)$.)

MIT, 2003 fall, Tommi Jaakkola, HW4, pr. 2.3

- b. Arătați că, pentru *orice funcție de pierdere validă* de tipul discutat mai sus, componenta $h(x; \hat{\theta}_t)$, care a fost adăugată la iterată t , are eroarea ponderată la antrenare în raport cu ponderile actualizate $\tilde{W}_i^{(t)}$ exact $1/2$.²⁵¹

CMU, 2008 fall, Eric Xing, HW3, pr. 4.1.1

CMU, 2008 fall, Eric Xing, midterm, pr. 5.1

- c. Să presupunem acum că schimbăm funcția obiectiv J_t în *media pătratelor erorilor*, adică $J_t = \frac{1}{m} \sum_{i=1}^m (y_i - f_t(x_i))^2$ — sau, mai simplu, $\sum_{i=1}^m (y_i - f_t(x_i))^2$ — și că urmărim să o optimizăm, ca și mai înainte, în mod secvențial.²⁵² Care este noua regulă de actualizare (engl., update rule) pentru α_t ?

MIT, 2006 fall, Tommi Jaakkola, HW4, pr. 3.a

MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.1

- d. Arătați că, atunci când folosim funcția de *pierdere logistică* [în locul celei invers exponențiale], ponderile nenormalizate $\tilde{W}_i^{(t)}$ au valori mărginite superior de 1 (adică, $\tilde{W}_i^{(t)} < 1$). (Sugestie: Veți exprima aceste ponderi în funcție de $y_i f_t(x_i)$.)

MIT, 2003 fall, Tommi Jaakkola, HW4, pr. 2.2

MIT, 2011 fall, L. P. Kaelbling, HW5, pr. 1.1

- e. În cazul funcției de *pierdere logistică*, care este valoarea / expresia ponderilor normalizate, $\tilde{W}_i^{(t)}$? Ce puteți spune despre ponderile normalizate pentru exemplele care sunt clasificate eronat în mod flagrant, comparativ cu cele care sunt clasificate ușor eronat de către combinația curentă („ansamblul“ curent), $f_T(x) \stackrel{\text{not}}{=} \sum_{t=1}^T \alpha_t h(x; \hat{\theta}_t)$? Dacă datele de antrenament conțin exemple care au fost etichetate în mod greșit (engl., mislabeled), de ce credeți că se preferă funcția de pierdere logistică în locul celei [negativ] exponențiale, $\text{Loss}(z) = \exp(-z)$?

L. Ciortuz, 2020

- f. Presupunem că lucrăm cu funcția de *pierdere logistică*. Care este în acest caz regula de actualizare pentru α_t ?

²⁵¹Acest nou rezultat generalizează proprietatea care a fost demonstrată la problema 22.vii.

²⁵²Observați că $(y_i - f(x_i))^2 = [y_i(1 - y_i f(x_i))]^2 = (1 - y_i f(x_i))^2 = (1 - z_i)^2$, unde am notat $z_i = y_i f(x_i)$. Funcția $(1 - z)^2$ este derivabilă și convexă; este descrescătoare pe intervalul $(-\infty, 1]$ și crescătoare pe intervalul $[1, +\infty)$. (Unii autori consideră în locul acestei funcție $(1 - z)^2$ funcția $(1 - z)^2 \cdot 1_{\{z < 1\}}$, care este descrescătoare pe tot domeniul de definiție.)

*MIT, 2006 fall, Tommi Jaakkola, HW4, pr. 3.b
MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.2*

g. Presupunem din nou că lucrăm cu funcția de pierdere logistică. Considerăm un set de date de antrenament liniar-separabil. Am dori să utilizăm o mașină cu vectori-suport liniară cu margine “hard” — adică, fără variabile / penalizări ecart (engl. slack penalties) — pe post de clasificator „slab“.²⁵³ Presupunând că la *Pasul 1* al algoritmului AdaBoost generalizat se minimizează eroarea ponderată la antrenare ε_t , care va fi valoarea ponderii α_1 la prima iterare de boosting?

Răspuns:

a. Pentru prima parte, vom arăta că atunci când se folosește $\text{Loss}(z) = e^{-z}$, valoarea obținută pentru coeficientul α la minimizarea de la *Pasul 2* din algoritmul general (vedeți partea stângă a egalității următoare) este aceeași cu valoarea lui α rezultată prin minimizarea efectuată de către algoritmul AdaBoost (vedeți partea dreaptă a egalității următoare), adică

$$\arg \min_{\alpha > 0} \sum_{i=1}^m \text{Loss}(y_i f_{t-1}(x_i) + \alpha y_i h(x_i; \hat{\theta}_t)) = \arg \min_{\alpha > 0} \sum_{i=1}^m \tilde{W}_i^{(t-1)} \exp(-\alpha y_i h(x_i; \hat{\theta}_t)),$$

unde, conform asignării de la execuția *Pasului 3* al iterării $t - 1$,

$$\tilde{W}_i^{(t-1)} = c_{t-1} \cdot \exp(-y_i f_{t-1}(x_i)),$$

c_{t-1} fiind constanta de normalizare (astfel încât ponderile însumate să dea valoarea 1). Calculând expresia funcției obiectiv din partea stângă a egalității de mai sus, obținem:

$$\begin{aligned} \sum_{i=1}^m \text{Loss}(y_i f_{t-1}(x_i) + \alpha y_i h(x_i; \hat{\theta}_t)) &= \sum_{i=1}^m \exp(-y_i f_{t-1}(x_i)) - \alpha y_i h(x_i; \hat{\theta}_t) \\ &= \sum_{i=1}^m \exp(-y_i f_{t-1}(x_i)) \exp(-\alpha y_i h(x_i; \hat{\theta}_t)) \\ &= \frac{1}{c_{t-1}} \sum_{i=1}^m \tilde{W}_i^{(t-1)} \exp(-\alpha y_i h(x_i; \hat{\theta}_t)) \\ &= \frac{1}{c_{t-1}} \left[\sum_{i \in \{i | y_i h(x_i; \hat{\theta}_t) = 1\}} \left(\tilde{W}_i^{(t-1)} \right) e^{-\alpha} + \sum_{i \in \{i | y_i h(x_i; \hat{\theta}_t) = -1\}} \left(\tilde{W}_i^{(t-1)} \right) e^{\alpha} \right], \end{aligned}$$

cantitate care este proporțională cu funcția obiectiv minimizată de către algoritmul AdaBoost (vedeți de asemenea problema 26). Prin urmare, minimizarea în raport cu α conduce la același rezultat în ambii algoritmi.

Pentru partea a doua, observați faptul că la *Pasul 3* (de la iterare t) din algoritmul general asignarea ponderilor se face astfel:

$$\tilde{W}_i^{(t)} = -c_t \cdot dL(y_i f_t(x_i)) = c_t \cdot \exp(-y_i f_t(x_i)),$$

ceea ce coincide cu modul cum procedează algoritmul AdaBoost (vedeți problema 23.a).

²⁵³LC: În locul acestui tip de SVM putem considera orice alt separator liniar consistent cu astfel de date de antrenament (adică, liniar separabile).

b. La iterația t , coeficientul α_t este ales minimizând $J_t(\alpha, \hat{\theta}_t)$, ceea ce implică $\frac{\partial J_t(\alpha, \hat{\theta}_t)}{\partial \alpha} = 0$. Întrucât

$$\begin{aligned}\frac{\partial}{\partial \alpha} J_t(\alpha, \hat{\theta}_t) &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial \alpha} \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \hat{\theta}_t)) \\ &= \frac{1}{m} \sum_{i=1}^m \underbrace{dL(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \hat{\theta}_t))}_{\tilde{W}_i^{(t)}} y_i h(x_i; \hat{\theta}_t) \\ &\quad - \frac{\tilde{W}_i^{(t)}}{c_t} \\ &\propto \sum_{i=1}^m \tilde{W}_i^{(t)} y_i h(x_i; \hat{\theta}_t),\end{aligned}$$

rezultă că trebuie să avem

$$\tilde{W}_i^{(t)} y_i h(x_i; \hat{\theta}_t) = 0.$$

După aceasta, eroarea ponderată la antrenare produsă de $h(x; \hat{\theta}_t)$ (corespunzător ponderilor actualizate $\tilde{W}_i^{(t)}$ determinate de α_t) poate fi calculată într-un mod similar cu relația (98):

$$\frac{1}{2} \left(1 - \underbrace{\sum_{i=1}^m \tilde{W}_i^{(t)} y_i h(x_i; \hat{\theta}_t)}_0 \right) = \frac{1}{2} (1 - 0) = \frac{1}{2}.$$

c. Pentru a găsi valoarea lui α_t , vom calcula derivata lui $J_t = \frac{1}{m} \sum_{i=1}^m (y_i - f_t(x_i))^2$ în raport cu α_t și o vom egala cu zero.

$$\frac{\partial J_t}{\partial \alpha_t} = \frac{1}{m} \frac{\partial \sum_{i=1}^m (y_i - f_t(x_i))^2}{\partial \alpha_t} = \frac{1}{m} \sum_{i=1}^m 2(y_i - f_t(x_i)) \frac{\partial (y_i - f_t(x_i))}{\partial \alpha_t}.$$

Stim că

$$f_t(x_i) = f_{t-1}(x_i) + \alpha_t h_t(x_i).$$

În această expresie, f_{t-1} nu depinde de α_t . Înlocuind în expresia derivatei, obținem

$$\frac{\partial J_t}{\partial \alpha_t} = \frac{2}{m} \sum_{i=1}^m (y_i - f_t(x_i)) \frac{\partial (y_i - f_{t-1}(x_i) - \alpha_t h_t(x_i))}{\partial \alpha_t} = \frac{2}{m} \sum_{i=1}^m (y_i - f_t(x_i)) (-h_t(x_i)).$$

Egalând această derivată cu zero, vom avea:

$$\begin{aligned}\frac{\partial J_t}{\partial \alpha_t} = 0 &\Leftrightarrow \sum_{i=1}^m (y_i - f_t(x_i)) h_t(x_i) = 0 \Leftrightarrow \sum_{i=1}^m (y_i - f_{t-1}(x_i) - \alpha_t h_t(x_i)) h_t(x_i) = 0 \\ \sum_{i=1}^m (y_i - f_{t-1}(x_i)) h_t(x_i) &= \alpha_t \sum_{i=1}^m h_t^2(x_i) \Leftrightarrow \alpha_t = \frac{\sum_{i=1}^m (y_i - f_{t-1}(x_i)) h_t(x_i)}{\sum_{i=1}^m h_t^2(x_i)} \\ &\Leftrightarrow \alpha_t = \frac{1}{m} \sum_{i=1}^m (y_i - f_{t-1}(x_i)) h_t(x_i).\end{aligned}$$

Observație: Egalitatea $h_t^2(x_i) = 1$ (pe care am folosit-o mai sus) decurge din presupunerea că ipotezele „slabe“ sunt compașii de decizie.

d. Ponderea / probabilitatea $W_i^{(t)}$ a fost definită ca fiind $-dL(y_i f_t(x_i))$, unde $dL(z) \stackrel{not.}{=} \frac{\partial}{\partial z}(\ln(1 + e^{-z}))$. Așadar,

$$W_i^{(t)} = \frac{e^{-z_i}}{1 + e^{-z_i}} = \frac{1}{1 + e^{z_i}} < 1, \text{ unde } z_i = y_i f_t(x_i).$$

e. Ponderile normalizate au fost definite astfel:

$$\tilde{W}_i^{(t)} = c_t \cdot \frac{\exp(-y_i f_t(x_i))}{1 + \exp(-y_i f_t(x_i))},$$

unde constanta de normalizare este

$$c_t = \left(\sum_{i=1}^m \frac{\exp(-y_i f_t(x_i))}{1 + \exp(-y_i f_t(x_i))} \right)^{-1}.$$

Pentru exemplele de antrenament care sunt clasificate eronat în mod flagrant, $y_i f_t(x_i)$ are o valoare negativă mare în valoare absolută, așadar $W_i^{(t)}$ va fi în acest caz aproape de [dar mai mic decât] 1. Pentru exemplele de antrenament care sunt clasificate ușor eronat, $W_i^{(t)}$ va fi aproape de [și mai mare decât] 1/2. Așadar, ponderile normalizate pentru aceste două cazuri se vor afla într-un raport de cel mult 2 : 1. Aceasta înseamnă că un singur outlier care este clasificat eronat în mod flagrant va avea întotdeauna o pondere care este de cel mult de două ori mai mare decât ponderea unei instanțe care este clasificată ușor eronat. Din acest motiv, algoritmul AdaBoost cu funcție de cost logistică este robust la outliers.

În cazul funcției de pierdere logistice, dacă avem un x_i care ar avea de drept eticheta $y_i = +1$, dar se consideră (în mod eronat) $y_i = -1$, pierderea este de aproximativ $f_t(x_i)$ dacă $f_t(x_i) > 0$,²⁵⁴ în vreme ce în cazul funcției de pierdere [negativ] exponențiale pierderea este $\exp(f_t(x_i))$ care este în general mult mai mare decât $f_t(x_i)$. Cazurile simetrice ($f_t(x_i) \leq 0$ și apoi $y_i = -1 \rightarrow +1$) se tratează în mod similar.

f.²⁵⁵ În loc să minimizăm J_t în raport cu argumentul α , vom urmări să minimizăm în raport cu α o *margine superioară* pentru diferența dintre J_t și J_{t-1}^* , unde, așa cum am definit la problema 26, $J_{t-1}^* \stackrel{not.}{=} \min_{h \in \mathcal{H}, \alpha \in \mathbb{R}_+} J_{t-1}(h, \alpha)$.²⁵⁶

$$\begin{aligned} J_t - J_{t-1}^* &= \sum_{i=1}^m \ln(1 + \exp(-y_i f_t(x_i))) - \sum_{i=1}^m \ln(1 + \exp(-y_i f_{t-1}(x_i))) \\ &= \sum_{i=1}^m \ln \frac{1 + \exp(-y_i f_t(x_i))}{1 + \exp(-y_i f_{t-1}(x_i))} \\ &= \sum_{i=1}^m \ln \frac{1 + \exp(-y_i f_{t-1}(x_i)) - \exp(-y_i f_{t-1}(x_i)) + \exp(-y_i f_t(x_i))}{1 + \exp(-y_i f_{t-1}(x_i))} \end{aligned}$$

²⁵⁴Vedeți graficul funcției de *pierdere logistică* din enunțul problemei de față.

²⁵⁵Soluția care urmează a fost redactată inițial de către studentul Ștefan Matcovici.

²⁵⁶Spre deosebire de J_t care depinde de α , J_{t-1}^* este constant, deci nu depinde de α . Prin urmare, a minimiza J_t în raport cu α este echivalent cu a minimiza $J_t - J_{t-1}^*$ în raport cu α . La finalul problemei 26.e am arătat că $J_t^* \geq J_{t-1}^*$ și $J_t(h, 0) = J_{t-1}^*$.

$$\begin{aligned}
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i f_t(x_i)) - \exp(-y_i f_{t-1}(x_i))}{1 + \exp(-y_i f_{t-1}(x_i))} \right) \\
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i f_{t-1}(x_i) - y_i \alpha h(x_i)) - \exp(-y_i f_{t-1}(x_i))}{1 + \exp(-y_i f_{t-1}(x_i))} \right) \\
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i f_{t-1}(x_i)) \exp(-y_i \alpha h(x_i)) - \exp(-y_i f_{t-1}(x_i))}{1 + \exp(-y_i f_{t-1}(x_i))} \right) \\
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i f_{t-1}(x_i)) [\exp(-y_i \alpha h(x_i)) - 1]}{1 + \exp(-y_i f_{t-1}(x_i))} \right) \\
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i \alpha h(x_i)) - 1}{\frac{1}{\exp(-y_i f_{t-1}(x_i))} + 1} \right) \\
&= \sum_{i=1}^m \ln \left(1 + \frac{\exp(-y_i \alpha h(x_i)) - 1}{\exp(y_i f_{t-1}(x_i)) + 1} \right) \tag{100}
\end{aligned}$$

$$\leq \sum_{i=1}^m \frac{\exp(-y_i \alpha h(x_i)) - 1}{\exp(y_i f_{t-1}(x_i)) + 1} \tag{101}$$

fiindcă $\ln(1+z) \leq z, \forall z > -1$.

Remarcați faptul că logaritmul din expresia (100) există, întrucât

$$\begin{aligned}
\frac{\exp(-y_i \alpha h(x_i)) - 1}{\exp(y_i f_{t-1}(x_i)) + 1} > -1 &\Leftrightarrow \exp(-y_i \alpha h(x_i)) - 1 > -\exp(y_i f_{t-1}(x_i)) - 1 \\
&\Leftrightarrow \underbrace{\exp(-y_i \alpha h(x_i))}_{>0} > \underbrace{-\exp(y_i f_{t-1}(x_i))}_{<0}, \\
&\text{inegalitate adevarată pentru } \forall \alpha.
\end{aligned}$$

Expresia (101) pe care tocmai am obținut-o mai sus este *marginea superioară* (engl., upper bound) pentru $J_t - J_{t-1}^*$ pe care o vom minimiza în raport cu α .

Un calcul simplu ne arată că

$$\frac{\partial}{\partial z} \ln(1 + e^{-z}) = \frac{-e^{-z}}{1 + e^{-z}} = -\frac{1}{\frac{1}{e^{-z}} + 1} = -\frac{1}{e^z + 1}. \tag{102}$$

Stim din enunț că $W_i^{(t-1)} = -dL(y_i f_{t-1}(x_i))$. Conform relației (102), rezultă că

$$W_i^{(t-1)} = -\left(-\frac{1}{\exp(y_i f_{t-1}(x_i)) + 1} \right) = \frac{1}{\exp(y_i f_{t-1}(x_i)) + 1},$$

iar

$$\tilde{W}_i^{(t-1)} = c_{t-1} \cdot \frac{1}{\exp(y_i f_{t-1}(x_i)) + 1},$$

unde c_{t-1} este constanta de normalizare a ponderilor $W_i^{(t-1)}$.

Prin urmare,

$$\begin{aligned}
 J_t - J_{t-1}^* &\leq \sum_{i=1}^m \frac{1}{\exp(y_i f_{t-1}(x_i)) + 1} (\exp(-y_i \alpha h(x_i)) - 1) \\
 &= \frac{1}{c_{t-1}} \sum_{i=1}^m \tilde{W}_i^{(t-1)} \cdot (\exp(-y_i \alpha h(x_i)) - 1) \\
 &\propto \sum_{i=1}^m \tilde{W}_i^{(t-1)} \cdot \exp(-y_i \alpha h(x_i)) - \underbrace{\sum_{i=1}^m \tilde{W}_i^{(t-1)}}_1.
 \end{aligned}$$

Așadar, a minimiza marginea superioară pentru $J_t - J_{t-1}^*$ revine la a minimiza (în raport cu α) expresia $\sum_{i=1}^m \tilde{W}_i^{(t-1)} \cdot \exp(-y_i \alpha h(x_i))$, pe care am întâlnit-o și la optimizarea costului [negativ] exponential. Putem conchide că și în cazul funcției de cost logistice putem alege $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$.

g. Conform enunțului, setul de date de antrenament este liniar-separabil. O mașină cu vectori-suport cu margine “hard” va produce pe un astfel de set de date un separator (liniar) care are eroarea ponderată la antrenare 0. Ca și în cazul lui AdaBoost, algoritmul trebuie oprit.²⁵⁷

30. (Algoritmul AdaBoost: Adevărat sau Fals?)

MIT, 2003 fall, Tommi Jaakkola, final, pr. 3.1-2

MIT, 2001 fall, Tommi Jaakkola, midterm, pr. 4.3

MIT, 2002 fall, Tommi Jaakkola, midterm, pr. 5.4

CMU, 2011 spring, Eric Xing, HW5, pr. 3.1.b

a. ε_t , eroarea ponderată produsă la antrenare de către ipoteza h_t / clasificatorul „slab“ A (măsurată relativ la ponderile de la începutul iterăției t) trebuie să crească în raport cu t .

²⁵⁷Dacă am gândi în mod miop, am judeca astfel:

La *Pasul 1*, alegem $\hat{\theta}_1$, astfel încât să minimizăm $\frac{\partial J_t(\alpha, \theta)}{\partial \alpha}|_{\alpha=0}$. Echivalent, putem gândi că acest $\hat{\theta}_1$ este ales astfel încât să minimizăm suma ponderată $2\varepsilon_1 - 1 = -\sum_{i=1}^m \tilde{W}_0(i)y_i h(x_i; \theta)$, unde $\tilde{W}_0(i) = \frac{1}{m}$ pentru $i = 1, 2, \dots, m$. Setul de date de antrenament fiind liniar-separabil, mașina cu vectori-suport cu margine “hard” va produce un separator $h(\cdot; \hat{\theta}_1)$ care satisfacă inegalitatea $y_i h(x_i; \hat{\theta}_1) \geq 1$ pentru $i = 1, 2, \dots, m$.

La *Pasul 2*, trebuie să alegem α_1 astfel încât să minimizăm $J_t(\alpha_1, \hat{\theta}_1) = \frac{1}{m} \sum_{i=1}^m L(y_i h_0(x_i) + \alpha_1 y_i h(x_i; \hat{\theta}_1)) = \frac{1}{m} \sum_{i=1}^m L(\alpha_1 y_i h(x_i; \hat{\theta}_1))$. Remarcăți faptul că $J_t(\alpha_1, \hat{\theta}_1)$ este o sumă de termeni care sunt strict decrescători în raport cu α_1 (pentru că $y_i h(x_i; \hat{\theta}_1) \geq 1$). Așadar, și suma $J_t(\alpha_1, \hat{\theta}_1)$ este strict decrescătoare în raport cu α_1 . Aceasta ar implica faptul că algoritmul AdaBoost generalizat folosind funcție de perdere logistică va lua $\alpha_1 = \infty$.

În general, dacă am avea un clasificator „de bază“ care separă în mod perfect datele de antrenament, îi vom asocia un vot oricărui putem de mare, ca să minimizăm pierderea / costul. Lețția este simplă: atunci când facem boosting, trebuie să folosim clasificatori „de bază“ care nu sunt atât de puternici încât să separe în mod perfect datele de antrenament.

- b. În decursul iterațiilor executate de algoritmul AdaBoost, erorile ponderate ε_t (produse la antrenare, pe rând, de către ipotezele „slabe“ h_t , în ocurență, compașii de decizie) pe de o parte, și erorile produse la antrenare de către clasificatorii combinați H_t pe de altă parte, variază aproximativ la fel.
- c. Ponderile / „voturile“ α_t asignate de către algoritmul AdaBoost clasificatorilor „slabi“ h_t asamblați sunt întotdeauna nenegative.
- d. Probabilitățile / ponderile $D_t(i)$ alocate de către algoritmul AdaBoost exemplelor de antrenament care au fost clasificate eronat [de către ipoteza h_t] vor crește cu un același factor multiplicativ.
- e. Întotdeauna după ce algoritmul AdaBoost execută suficient de multe iterații, eroarea la antrenare produsă de ipoteza combinată H_t descrește la o valoare care este oricât [dorim să fie] de apropiată de zero, indiferent de tipul de clasificatori „slabi“ folosiți.

Răspuns:

- a. Adevărat (în general). Modul de definire a probabilităților / ponderilor $D_t(i)$ asignate exemplelor de antrenament face ca algoritmul AdaBoost să se concentreze asupra exemplelor care sunt dificil de clasificat corect (foarte puține ipoteze „slabe“ clasifică aceste exemple în mod corect). După câteva iterații, cea mai mare parte a „masei“ de probabilitate va fi alocată acestor exemple „dificele“, iar eroarea ponderată la antrenare comisă de următoarea ipoteză „slabă“ va fi mai apropiată de 1/2 (care reprezintă eroarea corespunzătoare alegării aleatorii).
- b. Fals. În vreme ce eroarea la antrenare produsă de către clasificatorul combinat H_t îmmod tipic descrește ca funcție de t (numărul de iterații executate de AdaBoost), erorile ponderate la antrenare ε_t produse de ipotezele „slabe“ h_t îmmod tipic devin din ce în ce mai mari (așa cum am justificat deja la punctul precedent), fiindcă ponderile / probabilitățile $D_t(i)$ se alocă din ce în ce mai mult exemplelor care sunt dificil de clasificat.
- c. Adevărat. După cum s-a specificat în pseudo-codul din problema 22, algoritmul AdaBoost alege la fiecare iterație (t) ipoteze „slabe“, care au o eroare ponderată la antrenare ε_t strict mai mică decât 1/2. Prin urmare, $\ln((1-\varepsilon_t)/\varepsilon_t) > 0$, ceea ce înseamnă că „votul“ α_t este pozitiv.
- d. Adevărat. Puteți verifica acest fapt analizând formula (81) din problema 22, pentru actualizarea probabilităților $D_t(i)$. Întrucât pentru toate exemplele incorect clasificate avem $y_i \neq h_t(x_i)$, iar y_i și $h_t(x_i)$ pot fi doar ± 1 , rezultă că probabilitățile / ponderile alocate lor vor fi multiplicate cu factorul $\exp(-\alpha_t y_i h_t(x_i)) = \exp(\alpha_t)$. După aceea se face normalizarea, cu un același factor, Z_t .
- e. Fals. Dacă la o anumită iterație t algoritmului AdaBoost clasificatorul „slab“ A nu poate produce nicio ipoteză care să aibă eroarea ponderată la antrenare $\varepsilon_t < 1/2$, atunci algoritmul AdaBoost se oprește.²⁵⁸ La momentul respectiv, cea mai mică dintre erorile produse la antrenare de către clasificatorii [combinăți] $H_{t'}$ cu $t' = 1, \dots, t$, poate fi 0 ori strict pozitivă.

²⁵⁸ Altminteri, adică atunci când $\varepsilon_t = 1/2$, ponderea ipotezei h_t respective ar fi $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right) = \frac{1}{2} \ln 1 = 0$, deci h_t nu mai poate îmbunătăți combinația liniară învățată $f(x)$ și, prin urmare, eroarea clasificatorului combinat H_t rămâne neschimbată.

Însă, chiar și în cazul în care clasificatorul „slab“ A produce la orice iterație ipoteze h_t cu eroare $\varepsilon_t < 0.5$ (deci $\gamma_t \stackrel{\text{not.}}{=} \frac{1}{2} - \varepsilon_t > 0$), deși avem certitudinea că *marginea superioară* $\exp(-2 \sum_{t'=1}^t \gamma_t^2)$ pentru $\text{err}_S(H_t)$ descrește ca funcție de t (vedeți pr. 23.d), nu putem avea și certitudinea că ea converge la 0, și nici certitudinea că *eroarea* $\text{err}_S(H_t)$ însăși descrește mereu. Vedeți, de exemplu, soluția a două de la problema 25.b (cea care a fost obținută fără a folosi prag exterior).

4.2 Probleme propuse

4.2.1 Algoritmul ID3

31. (Arborei de decizie; optimalitate, ca număr minim de noduri)

Reprezentați arborele / arborii de decizie care are / au numărul minim posibil de noduri (de test) și corespunde / corespund funcției booleene $(A \text{ XOR } B) \wedge C$ definită peste atributele booleene A, B și C .

32. (Expresivitatea arborilor de decizie: un rezultat privind funcțiile booleene)

Orice funcție booleană (care primește n argumente din mulțimea $\{0, 1\}$ și întoarce un element din mulțimea $\{0, 1\}$) poate fi reprezentată cu ajutorul unui arbore de decizie. Adevărat sau fals?

În cazul afirmativ, explicați succint cum anume poate fi construit arborele de decizie respectiv.

În cazul negativ, dați un exemplu de funcție booleană pentru care nu se poate construi un arbore de decizie consistent cu funcția respectivă.

33. (Calcularea câștigului de informație pe “decision stumps”)
 ■ CMU, 2013 fall, W. Cohen, E. Xing, Sample Questions, pr. 4

Studentul Timmy dorește să știe cum [ar trebui să procedeze cel mai bine ca] să promoveze examenul de învățare automată. Pentru aceasta, a cules informații de la studenții care au urmat acest curs în anii precedenți și apoi a decis să-și construiască un *model* bazat pe arbori de decizie. A colectat în total nouă *instanțe* / exemple, descrise cu ajutorul a două *trăsături* (văzute în cele ce urmează ca două variabile aleatoare, S și A): „este bine să stai și să înveți până noaptea târziu înainte de examen“ (S) și „este bine să mergi la toate cursurile și seminariile“ (A). Timmy dispune acum de următoarele „statistici“ (care sunt de fapt *partiționări* ale datelor sale):

$$\begin{aligned}Set(all) &= [5+, 4-] \\Set(S+) &= [3+, 2-], Set(S-) = [2+, 2-] \\Set(A+) &= [5+, 1-], Set(A-) = [0+, 3-]\end{aligned}$$

Presupunând că se folosește drept criteriu de selecție a celei mai bune trăsături câștigul maxim de informație, ce trăsătură va alege Timmy? Care este valoarea câștigului de informație?

Puteți folosi la calcule următoarele aproximații:

N	3	5	7
$\log_2 N$	1.5850	2.3219	2.8073

34.

(Implementare: compas de decizie,
entropie, entropie condițională specifică,
entropie condițională medie, câștig de informație)
Liviu Ciortuz, 2016

Folosind limbajul de programare pe care-l preferați, implementați un program care, pornind de la o structură de date de tip compas de decizie (engl., decision stump), calculează entropia, entropiile condiționale specifice, entropia condițională medie, precum și câștigul de informație aferent.

În mod concret, programul va primi ca *input*

- m — numărul de valori posibile ale etichetei / atributului de ieșire (în mod implicit, se va considera $m = 2$);
- n — numărul de valori ale atributului (notat mai jos cu A) în raport cu care se face partiționarea mulțimii de instanțe asociate nodului-rădăcină al compasului de decizie (valoarea implicită: $n = 2$);
- partițiile (de fapt, count-urile) corespunzătoare nodurilor descendente. Pornind de la aceste partiții, programul va calcula partiția asociată nodul-rădăcină al compasului de decizie.

De *exemplu*, pentru primul compas de decizie de la problema 33, inputul va avea forma $[3, 2]$, $[2, 2]$, în vreme ce pentru al doilea compas de decizie va fi $[5, 1]$, $[0, 3]$.

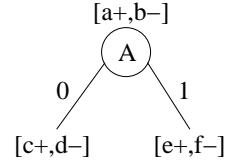
Programul va calcula și apoi va afișa

- entropia atributului / variabilei de ieșire (notată aici cu Y);
- entropiile condiționale specifice pentru fiecare descendent din nodul-rădăcină;
- entropia condițională medie a atributului A ;
- câștigul de informație al atributului [de ieșire] Y în raport cu atributul [de intrare] A .

35. (Compași de decizie; entropie, entropia condițională medie și câștigul de informație: formule de calcul adecvate pentru folosirea calculatorului de buzunar)

■ UAIC, Iași, 2017, Sebastian Ciobanu, Liviu Ciortuz

Fie compasul de decizie din figura alăturată. a, b, c, d, e și f reprezintă count-uri corespunzătoare unui set de date de antrenament. După cum se observă, eticheta (sau, variabila de ieșire), notată cu Y , este binară, iar atributul (sau, variabila de intrare) A este de asemenea binar. Evident, $a = c + e$ și $b = d + f$.



- a. Arătați că entropia [variabilei de ieșire] corespunzătoare partitiei asociate nodului de test este

$$H[a+, b-] = \frac{1}{a+b} \log_2 \frac{(a+b)^{a+b}}{a^a b^b} \text{ dacă } a \neq 0 \text{ și } b \neq 0.$$

- b. Cum s-ar scrie formula corespunzătoare entropiei variabilei de ieșire în cazul când ea este ternară, iar partitia din nodul de test [al compasului de decizie] este $[a+, b-, c*]$?

Atenție! Nu există nicio legătură între acest ultim c și count-ul c din compasul de decizie de mai sus.

- c. Presupunând că niciunul dintre c, d, e și f nu este nul, arătați că entropia condițională medie corespunzătoare compasului de decizie din desenul de mai sus este

$$H_{nod|atribut} = \frac{1}{a+b} \log_2 \left(\frac{(c+d)^{c+d}}{c^c d^d} \cdot \frac{(e+f)^{e+f}}{e^e f^f} \right).$$

- d. Să presupunem acum că unul dintre count-urile c, d, e și f este 0; pentru fixarea ideilor vom considera $c = 0$. Elaborați formula entropiei condiționale medii pentru compasul de decizie în acest caz.

- e. Demonstrați următoarea formulă pentru câștigul de informație corespunzătoare compasului de decizie de mai sus, presupunând că a, b, c, d, e și f sunt strict pozitive:

$$IG_{nod;atribut} = \frac{1}{a+b} \log_2 \left(\frac{(a+b)^{a+b}}{a^a b^b} \cdot \frac{c^c d^d}{(c+d)^{c+d}} \cdot \frac{e^e f^f}{(e+f)^{e+f}} \right).$$

Observație (1): Întrucât majoritatea calculatoarelor de buzunar nu au funcția \log_2 ci funcțiile \ln și \lg , în formulele prezentate sau deduse la punctele $a - e$ ar fi de dorit să schimbăm baza logaritmului. Aceasta revine – pe lângă înlocuirea lui \log_2 cu \ln sau \lg – la înmulțirea membrului drept cu $1/\ln 2$, respectiv $1/\lg 2$.

Observație (2): Întrucât, la aplicarea algoritmului ID3, pentru alegerea celui mai bun atribut de pus în nodul curent este suficient să calculăm entropiile condiționale medii, va fi suficient să comparăm produsele de forma

$$\frac{(c+d)^{c+d}}{c^c d^d} \cdot \frac{(e+f)^{e+f}}{e^e f^f} \quad (103)$$

pentru compașii de decizie considerați la nodul respectiv și să alegem minimul dintre aceste produse.

Atenție! O problemă importantă care poate apărea la folosirea acestor formule în lucrul cu calculatorul de buzunar este *depășirea capacitatei de reprezentare* a rezultatelor intermedii. Spre exemplu, la un calculator Sharp EL-531VH, putem lucra cu 56^{56} dar nu și cu 57^{57} . Similar, pe calculatorul disponibil în [meniul *Accessories* din] sistemul de operare Linux Mint putem lucra cu 179^{179} dar nu și cu 180^{180} . Din acest motiv, în cazul depășirii capacitatei de reprezentare pe calculatoare de buzunar, trebuie să utilizați formulele de bază pentru entropii și pentru câștigul de informație, întrucât ele folosesc mult mai mult funcția log.

36.

(Calcularea unor entropii;
aplicarea algoritmului ID3)

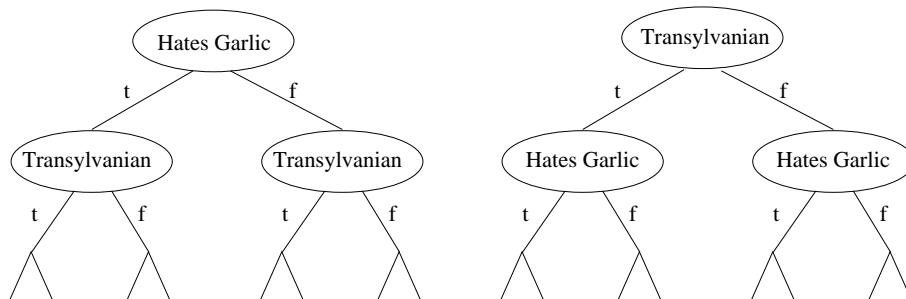
*prelucrare de Liviu Ciortuz, după
CMU, 2014 spring, Seyoung Kim, HW2, pr. 1.4.1-5*

Centrul [de Medicină] pentru Controlul și Prevenția Maladiilor a fost sesizat în legătură cu o creștere surprinzătoare a aparițiilor de vampiri. Acest centru a colectat date preliminare referitoare la anumite *caracteristici*, atât pentru vampiri [deja] cunoscuți cât și pentru non-vampiri, și acum ar dori să construiască un arbore de decizie ca să-i ajute pe cetățeni să identifice noi vampiri. Datele culese sunt prezentate în tabelul următor.

V (Vampire)	G (Hates Garlic)	T (Transylvanian)	Nr. apariții
+	t	t	9
+	t	f	4
+	f	t	3
+	f	f	0
-	t	t	1
-	t	f	3
-	f	t	1
-	f	f	6

Fiecare linie indică ce caracteristici au fost „observate“, și de câte ori a fost observată fiecare combinație de caracteristici. De exemplu, combinația (+, t, t) a fost observată de 9 ori, pe când combinația (+, f, f) n-a fost observată niciodată. (Simbolii 't' și 'f' au fost folosiți în locul lui True și False, pentru a evita confuzia cu atributul T .)

- a. Completati arborii de mai jos cu informații (count-uri) referitoare la partitioarea datelor (sub forma $[+n, -m]$) în fiecare nod, precum și cu deciziile care trebuie luate (în sens majoritar) în nodurile frunză.



b. Calculați entropia condițională medie $H(V|G)$. (Toate calculele intermediare trebuie făcute cu o precizie de cel puțin 4 zecimale, pentru a ne asigura că răspunsul final are primele 3 zecimale corecte.)

c. Calculați entropia condițională medie $H(V|T)$. (Din nou, toate calculele intermediare trebuie făcute cu o precizie de cel puțin 4 zecimale.)

d. Care dintre cei doi arbori de decizie de mai sus reprezintă rezultatul învățării realizate de algoritm ID3 pe aceste date?

e. Adevărat sau Fals: Arborele produs de către ID3 va clasifica o persoană căreia-i displace usturoiul (engl., hates garlic) dar nu este transilvănean ca fiind vampir.

Indicație: Este posibil să aveți nevoie de următoarele valori pentru entropia ($H(p)$) unei variabile aleatoare Bernoulli de parametru p : $H(1/7) = 0.5916$, $H(4/17) = 0.7871$, $H(3/10) = 0.8812$, $H(4/13) = 0.8904$, $H(11/27) = 0.9751$.

37.

(Algoritmul ID3: aplicare pe expresii booleene; exploatarea simetriilor operațiilor \vee , \wedge în alegerea atributelor; analiza „optimalității“ arborelui ID3)

prelucrare de Liviu Ciortuz, după Tom Mitchell, "Machine Learning", 1997, ex. 3.1.d

Considerăm următoarea funcție booleană: $(A \wedge B) \vee (C \wedge D)$. Valorile pe care le ia această funcție, calculate conform diferitelor valori de adevăr atribuite variabilelor / atributelor A , B , C și D sunt cele cunoscute din logica propozițiilor. Dorim însă să reprezentăm această funcție ca arbore de decizie.

a. Aplicați algoritmul ID3 acestei funcții.

Observație: Dacă exploatați simetriile, este nevoie doar de puține calcule, altfel vă complicați în mod inutil.

b. Arborele ID3 obținut la punctul precedent este optimal?

Alfel spus, puteți găsi alt arbore de decizie de adâncime mai mică sau cu număr mai mic de noduri (de test) pentru această funcție? (Țineți cont că în fiecare nod al unui arbore de decizie se poate testa un sigur atribut.)

38.

(Algoritmul ID3: aplicare; analiza „optimalității“ arborelui ID3)

CMU, 2005 spring, C. Guestrin, T. Mitchell, midterm, pr. 4

Agenția spațială NASA dorește să distingă între marțieni (M) și pământeni (H) folosind următoarele caracteristici: $Green \in \{N, Y\}$, $Legs \in \{2, 3\}$, $Height \in \{S, T\}$, $Smelly \in \{N, Y\}$.

Datele de antrenament de care dispunem sunt prezentate în tabelul alăturat.

	Species	Green	Legs	Height	Smelly
1	M	N	3	S	Y
2	M	Y	2	T	N
3	M	Y	3	T	N
4	M	N	2	S	Y
5	M	Y	3	T	N
6	H	N	2	T	Y
7	H	N	2	S	N
8	H	N	2	T	N
9	H	Y	2	S	N
10	H	N	2	T	Y

- a. Învătați un arbore de decizie folosind algoritmul ID3 și trasați arborele respectiv.
 b. Descrieți conceptul M (marțian) ca un set de reguli conjunctive din logica propozițiilor. Spre exemplu:

```
if Green = Y and Legs = 2 and Height = T and Smelly = N then M;
else
  if ... then M; else H.
```

- c. Soluția de la punctul *b* de mai sus folosește cel mult 4 atrbute în fiecare conjuncție. Găsiți un set de reguli conjunctive care folosesc doar 2 atrbute pentru fiecare conjuncție, păstrând însă eroarea la antrenare zero. Această ipoteză mai simplă poate fi reprezentată ca un arbore de decizie de adâncime 2? Justificați răspunsul.

39.

(Algoritmul ID3: aplicare; cazul instanțelor de antrenament cu multiple apariții)
CMU, 2010 fall, Aarti Singh, HW2, pr. 5.1

Tabelul de mai jos descrie instanțe (înregistrări) pozitive și instanțe negative pentru persoane cărora banca le-a acordat (sau nu le-a acordat) un card de credit.

Fiecare linie din tabel indică niște combinații de valori observate pentru atrbutele considerate (*Gender*, *Income* și *Approved*) și de câte ori a fost înregistrată respectiva combinație de valori. De exemplu, (F, Low, +) a apărut de 10 ori, iar (F, Low, -) de 80 de ori.

<i>Gender</i>	<i>Income</i>	<i>Approved</i>	<i>Counts</i>
F	Low	+	10
F	High	+	95
M	Low	+	5
M	High	+	90
F	Low	-	80
F	High	-	20
M	Low	-	120
M	High	-	30

- a. Calculați entropia atrbutului *Approved* pe acest set de date de antrenament (folosind logaritmul cu baza 2).
 b. Calculați de asemenea câștigurile de informație $IG(Approved, Gender)$ și $IG(Approved, Income)$.
 c. Desenați un arbore de decizie produs de către algoritmul ID3 (fără post-pruning) pe baza acestui set de date de antrenament.

40.

("Decision stump" produs de ID3: raționament calitativ pe un exemplu simplu)
CMU, 2010 fall, Ziv Bar-Joseph, midterm exam, pr. 5.a

Vrem să construim un arbore de decizie care să ne ajute să prezicem întârzierile avioanelor. Timp de câteva luni am colectat informații, iar un rezumat al acestora este prezentat în tabelul alăturat.

Atribut	Valoare = <i>Da</i>		Valoare = <i>Nu</i>	
	#Zboruri amânate	#Zboruri neamânate	#Zboruri amânate	#Zboruri neamânate
Ploaie	30	10	10	30
Vânt	25	15	15	25
Vara	5	35	35	5
Iarna	20	10	20	30
Ziua	20	20	20	20
Noaptea	15	10	25	30

- a. Pe baza acestui tabel precizați ce atribut ar trebui să fie pus în rădăcina arborelui de decizie, folosind criteriul câștigului de informație. Justificați riguros; nu este însă necesar să elaborați în detaliu toate calculele.
- b. Pe baza acelaiași tabel, precizați care dintre attribute ar trebui să apară pe al doilea nivel (nivelul de sub rădăcină) al arborelui de decizie.
41. (O aproximare a numărului de instanțe greșit clasificate care au fost asignate la un nod frunză dintr-un arbore ID3)
CMU, 2003 fall, T. Mitchell, A. Moore, midterm exam, pr. 9.b
 Învățăm un arbore de decizie folosind un set de date de antrenament cu atributul de ieșire (*class*) având valorile 0 sau 1.
 Presupunând că pentru un nod frunză l din acest arbore,
 - există M instanțe de antrenament asignate la acel nod, iar
 - entropia sa este H ,
 schițați un algoritm simplu care ia ca valori de intrare M și H și furnizează la ieșire numărul de exemple de antrenament clasificate greșit de către nodul frunză l .
Sugestie: Folosiți o aproximare simplă (polinomială) pentru funcția entropie $H(p)$.

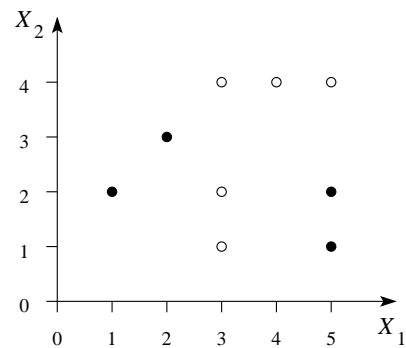
42. (Algoritmul ID3: eroarea la antrenare)
CMU, 2003 fall, T. Mitchell, A. Moore, HW1, pr. 2.1
 Un student mi-a spus următoarele:
 - el poate să construiască un set de instanțe cu attributele de intrare discrete și atributul de ieșire binar;
 - mie îmi dă voie să aleg o parte din acest set de instanțe (dar nu toate!) pentru a antrena un arbore de decizie;
 - indiferent de modul cum mi-aș alege datele de antrenament din setul construit de el, eroarea de clasificare pe care arborele de decizie (obținut în urma antrenării cu algoritmul ID3) o va face pe instanțele care nu au fost incluse în setul de antrenament va fi de cel puțin 50%.
 Credeți că studentul are dreptate? Explicați de ce sau dați un exemplu.

43. (Extensiile ale algoritmului ID3: variabile de intrare continue; determinarea celui mai bun prag de separare: o proprietate)
USC, 2008 fall, Sofus Macskassy, HW2, pr. 3
 Atunci când construim arbori de decizie, selecția atributelor se face folosind de obicei criteriul câștigul de informație maxim.
 Arătați că în cazul variabilelor de intrare continue, pragurile de separare (engl., split-point) pentru care de o parte și de alta etichetele sunt de același tip nu vor conduce niciodată la câștig de informație maxim.
Sugestie: Vă recomandăm să citiți [secțiunea 4 din] lucrarea *Technical note: On the handling of continuous-valued attributes in decision tree generation*, Usama M. Fayyad, Keki B. Irani, *Journal of Machine Learning*, nr. 8, 1992, pages 87-102.

44.

(ID3 cu atribute continue:
zone de decizie și separatori decizionali)■ Liviu Ciortuz, 2017,
folosind datele de la problema 24

Fie setul de date de antrenament din figura alăturată. X_1 și X_2 sunt considerate atribute numerice continue. Vă readucem aminte *convenția* noastră de notare: simbolul \bullet desemnează instanțe pozitive, iar simbolul \circ instanțe negative. Aplicați algoritmul ID3 pe acest set de date. (Faceți toate calculele necesare, în mod detaliat; precizați la fiecare pas care sunt pragurile de splitare pentru cele două atribute.) Desenați arborele de decizie rezultat. La final, reprezentați grafic zonele de decizie și separatoare decizionali, marcând clar zona pozitivă (sau zonele pozitive) și zona negativă (sau zonele negative).

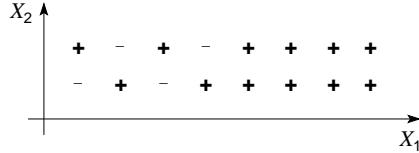


45.

(Extinderea algoritmului ID3 cu atribute continue;
eroarea la antrenare, eroarea la CVLOO; overfitting)

CMU, 2005 fall, T. Mitchell, A. Moore, midterm exam, pr. 2

Figura alăturată prezintă un set de date cu două intrări X_1 și X_2 , variabile cu valori reale, și o ieșire Y care poate lua valori pozitive (+) sau negative (-).



Testăm doi algoritmi extremi de învățare de arbori de decizie. Algoritmul *OVERFIT* construiește un arbore de decizie în maniera standard a algoritmului ID3, fără a face pruning. Algoritmul *UNDERFIT* refuză complet să-și asume riscul splitării intervalelor de valori pentru X_1 și X_2 și construiește un arbore de decizie alcătuit doar dintr-un singur nod (care va fi simultan și rădăcină și nod frunză, deci și nod de decizie).

- Câte noduri frunză vor fi în arborele de decizie învățat de *OVERFIT* pe aceste date?
- Care va fi eroarea de clasificare la cross-validation cu metoda “Leave-One-Out” pe acest set de date atunci când folosim algoritmul *OVERFIT*? (Indicați datele incorect clasificate.)
- Similar punctului anterior, pentru cazul în care se folosește algoritmul *UNDERFIT*.

46.

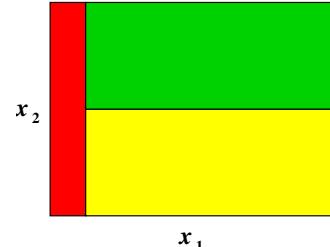
(Arbore de decizie cu variabile continue:
ID3 ca algoritm “greedy”)

CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, HW1, pr. 8.abcf

Fie următoarea problemă de clasificare ternară. Considerăm că în figura de mai jos regiunea dreptunghiulară este populată în mod dens cu puncte caracterizate de două

attribute numerice (continue), x_1 și x_2 . Cele trei sub-dreptunghiuri (roșu, verde și galben) reprezintă trei clase de puncte, C_1 , C_2 , și C_3 .

Dimensiunile $x_1 \times x_2$ ale dreptunghiurilor roșu, verde și galben sunt 1×6 , 7×3 și respectiv 7×3 . Dreptunghiul roșu este populat în mod uniform cu 6000 de puncte din clasa C_1 . Dreptunghiul verde este populat în mod uniform cu 42000 de puncte din clasa C_2 . Dreptunghiul galben este populat în mod uniform cu 42000 de puncte din clasa C_3 . Pentru simplitate, nu vom considera alte puncte decât acestea.



- Care este numărul minim de noduri de test pe care trebuie să le aibă un arbore de decizie pentru a clasifica în mod corect acest set de date?
- Câte noduri de test are arborele de decizie obținut în urma antrenării algoritmului ID3 pe acest set de date, folosind criteriul maximizării câștigului de informație?
- Avem același număr de noduri în cele două cazuri de mai sus, sau nu? Care credeți că este explicația?
- Un arbore de decizie poate să clasifice setul de date din figura de mai sus cu 100% acuratețe (presupunând că nu există zgomote la nivel de etichete). Ce condiții trebuie să satisfacă în general un set de date de acest gen astfel încât arborele de decizie rezultat în urma antrenării să fie cât mai compact și să producă o acuratețe de 100%?

Indicație: Fiecare nod intern al arborelui de decizie corespunde unui test bazat pe o singură trăsătură. Gândiți-vă ce fel de clase de funcții / granițe de separare corespund unui astfel de arbore de decizie.

47.

(Un exemplu de aplicare a algoritmului ID3:
cazul când se folosesc atât variabile discrete
cât și variabile continue)

CMU, 2010 fall, Ziv Bar-Joseph, HW2, pr. 1

Cursul de Învățare Automată pe care l-am urmat în acest semestrul, și-a insuflat dorința ca după absolvirea facultății să fondeze pe cont propriu o firmă (engl., start-up company). Pentru a-ți evalua șansele de succes — adică, mai exact, dacă vei deveni milionar sau nu —, ai colectat date de la absolvenții de la CMU, referitoare la foști studenți care și-au înființat propriile lor companii start-up. Pentru fiecare start-up, știi acum ce anume produce compania respectivă, ce fonduri de capital de investiție a obținut (exprimat în milioane de dolari), dacă directorul companiei a studiat la CMU o disciplină din domeniul științelor exacte și, în final, dacă directorul a devenit milionar. Tabelul de mai jos centralizează datele pe care le-ai cules.

Produs	Capital atras	Ştiinţe exacte	Milionar
SiteDeSocializare	2.9	Da	Nu
SiteDeSocializare	1.7	Da	Nu
SiteDeSocializare	3.4	Da	Nu
SiteDeSocializare	2.3	Nu	Da
MaşinaAlimentatăCuCombustibilBio	3.4	Da	Da
MaşinaAlimentatăCuCombustibilBio	6.1	Da	Da
MaşinaAlimentatăCuCombustibilBio	5.6	Nu	Nu
MaşinaAlimentatăCuCombustibilBio	0.6	Nu	Nu
NanoVaccin	1.9	Nu	Nu
NanoVaccin	2.9	Nu	Da
NanoVaccin	3.1	Da	Da
NanoVaccin	0.3	Da	Nu

Acum ai vrea să construiești un arbore de decizie pornind de la aceste date. Referitor la atributul cu valori continue *CapitalAtras*, știm că arborele de decizie poate conține teste (partiționări binare) de forma $\text{CapitalAtras} \leq v$ și $\text{CapitalAtras} > v$ și că pot exista mai multe teste de acest fel în arbore.

a. Câte „praguri“ distințe v trebuie să considerăm pentru *CapitalAtras* atunci când căutăm atributul (optim) care trebuie pus în nodul rădăcină?

b. Desenați arborele de decizie care va fi învățat de către algoritm ID3 extins cu attribute cu valori continue, aşa cum a fost prezentat la curs. Ne vom referi ulterior la acest arbore ca fiind *arborele original*. Adnotați fiecare nod intern (i.e., nod de test) din arbore cu câștigul de informație obținut în urma aplicării testului respectiv.

Indicație: Pentru punctele c și d de mai jos, deși nu este neapărat necesar, este recomandabil să folosiți o implementare a algoritmului ID3 (extins cu attribute numerice continue). Vă sugerăm să abordați mai întâi problemele 34 și 53, care vă ghidează cum să construjiți propria dumneavoastră implementare.

c. Schimbați eticheta (i.e., valoarea clasei binare *Milionar*) pentru o instanță din tabelul de mai sus astfel încât arborele care va fi învățat ulterior să conțină cel puțin încă un nod de test în raport cu arborele de la punctul b .

d. Un exemplu de antrenament este consistent cu arborele învățat dacă este clasificat corect de către acel arbore. Întrebarea pe care v-o adresăm acum este următoarea:

Este posibil să adăugăm la setul de date de antrenament de mai sus noi exemple care sunt consistente cu arborele original, dar care fac totuși ca algoritmul ID3 extins, executat pe noul set de date de antrenament, să învețe un arbore cu o rădăcină diferită de cea originală și cu mai multe noduri decât arborele original?

Dacă răspunsul dumneavoastră este afirmativ, indicați noile exemple pe care ați putea să le adăugați, precum și arborele de decizie rezultat. Dacă răspunsul este negativ, explicați de ce este așa.

48.

(Variante ale algoritmului ID3:
proceduri de tip “look-ahead”
specifice variabilelor continue)*CMU, 2007 spring, Andrew Moore, final exam, pr. 4*

La curs am arătat cum lucrează ID3, un algoritm “greedy” pentru invățarea arborilor de decizie, pornind de la un set de date de antrenament.

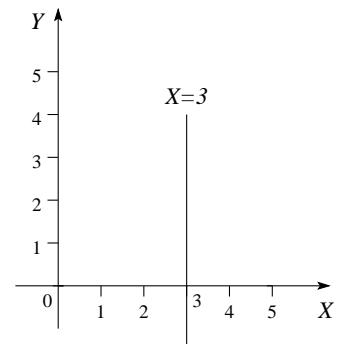
Un astfel de algoritm partiționează în mod recursiv *spațiul de trăsături* (engl., feature space) în regiuni etichetate, optimizând [la fiecare iterare] în mod “greedy” o anumită *măsură* numerică și anume câștigul de informație. *Obiectivul* urmărit este acela de a produce arbori de decizie cât mai simpli (pe cât posibil), care să partiționeze spațiul de trăsături în anumite regiuni, în care datele de antrenament să fie clasificate perfect.

Ca și în cazul celor mai multe abordări de tip “greedy”, această abordare nu ne oferă garanția că vom produce un set de regiuni care maximizează în cel mai bun mod posibil această *măsură*.

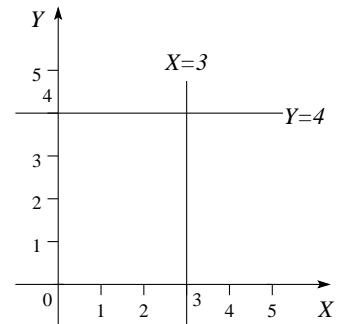
Totuși, întotdeauna putem proceda mai puțin “greedy”. În loc să luăm o primă decizie în mod “greedy” și apoi o a doua decizie tot în mod “greedy”, putem identifica dintre toate perechile posibile de decizii care este cea mai bună pereche și apoi să ne fixăm la aceea.

În această problemă vom explora care sunt avantajele dar și costurile în cazul acestei abordări, mai puțin “greedy”.

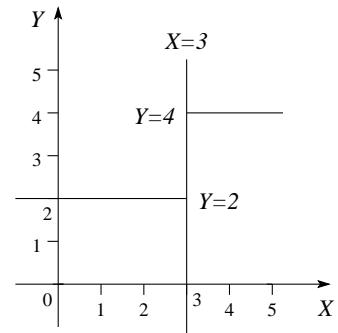
- În cazul unui *arbore de decizie standard*, la fiecare nivel de recursie care lucrează cu un atribut numeric continuu se va alege un *prag* decizional (de exemplu, $X = 3$), care împarte spațiul de trăsături în două regiuni (de exemplu, $X \leq 3$ și $X > 3$) astfel încât să maximizăm *măsura* fixată (câștigul de informație). Fiecare din aceste două regiuni va fi apoi împărțită recursiv, folosind aceeași procedură.



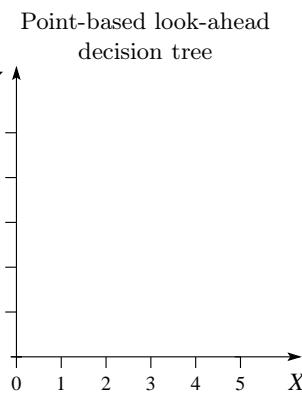
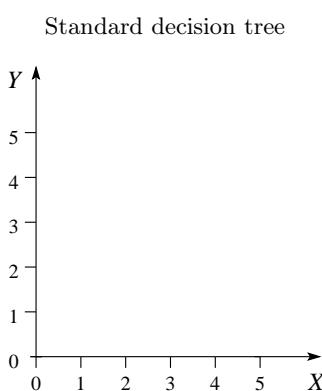
- Pentru un *arbore de decizie cu explorare în avans bazată pe puncte* (engl., point-based look-ahead), spațiul trăsăturilor este partiționat la fiecare iterare în patru regiuni, pe baza alegerii unui anumit punct. Spre exemplu, punctul $(X, Y) = (3, 4)$ va determina regiunile $[X > 3, Y > 4]$, $[X \leq 3, Y > 4]$, $[X \leq 3, Y \leq 4]$, și $[X > 3, Y \leq 4]$.



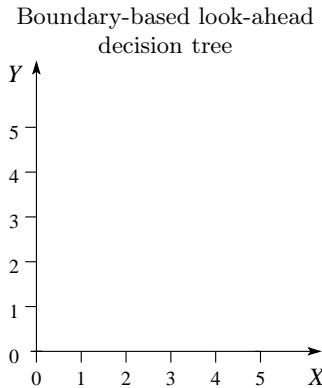
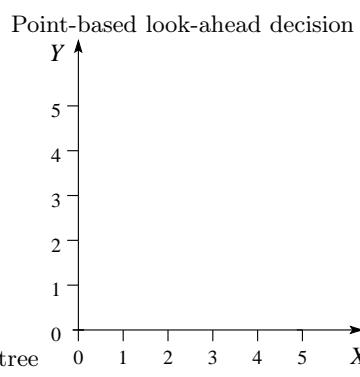
- În cazul unui *arbore de decizie cu explorare în avans bazată pe praguri* (engl., boundary-based look-ahead), la fiecare nivel al construcției recursive a arborelui de decizie sunt luate în considerare trei praguri decizionale. Primul prag decizional împarte spațiul de trăsături în două regiuni, iar cele două praguri adiționale împart aceste două regiuni într-un total de 4 regiuni. Spre exemplu, putem lua $X = 3$ ca prim prag și $Y = 2$ pentru regiunea $X \leq 3$, precum și $Y = 4$ pentru regiunea $X > 3$. Așadar, vom obține regiunile $[X \leq 3, Y \leq 2]$, $[X \leq 3, Y > 2]$, $[X > 3, Y \leq 4]$, și $[X > 3, Y > 4]$.



- a. Concepți un set de date pe următoarele două grafice astfel încât un arbore de decizie standard cu două praguri (4 regiuni) va clasifica destul de prost datele, dar un arbore de decizie cu explorare în avans bazată pe puncte având un singur nivel (4 regiuni) va clasifica perfect datele. Folosiți semnele + și – pentru a indica clasa fiecărui punct și marcați pe desen *granițele de decizie* (engl., decision boundaries) pentru fiecare dintre acești arbori de decizie.



- b. Concepți un set de date pentru următoarele două grafice astfel încât un arbore de decizie cu explorare în avans bazată pe puncte având un singur nivel (deci 4 regiuni) va clasifica greșit datele, dar un arbore de decizie cu explorare în avans bazată pe praguri având tot un nivel (deci 4 regiuni) va clasifica perfect datele. Ca și la punctul precedent, folosiți semnele + și – pentru a indica clasa fiecărui punct și marcați pe desen granițele de decizie pentru fiecare dintre acești arbori de decizie.



- c. Calculați timpul de rulare necesar pentru un nivel de partitioare în cazul fiecăreia dintre cele trei variante de arbori de decizie. Vom presupune că există D puncte în setul de date antrenament și că toate aceste puncte au valori unice pentru trăsăturile / atributele X și Y . Elaborați raționamentul.

49.

(Deficiențe ale pruning-ului top-down și respectiv bottom-up)

CMU, 2009 fall, Carlos Guestrin, HW1, pr. 2.4

- a. Știm că algoritmul ID3 alege în fiecare nod de test atributul care are cel mai mare câștig de informație dintre toate atributurile de intrare rămase disponibile pentru acel nod. O posibilă metodă de *pruning* în manieră *top-down* este următoarea: dacă toate atributurile despre care am vorbit au câștig de informație 0 atunci oprim aplicarea algoritmului ID3 (adică elaborarea arborelui de decizie) pe ramura respectivă. Această strategie este însă problematică, întrucât ea poate conduce la arbori de decizie inconsistenti cu datele, chiar și atunci când datele sunt consistente / necontradictorii. Vă cerem să furnizați un mic set de exemple de antrenament care să demonstreze acest fapt. *Sugestie:* Este suficient să folosiți două atrbute de intrare booleene.

- b. La problema 19.b, am discutat [și] despre *pruning bottom-up* ca modalitate de control asupra „complexității“ arborelui creat de algoritmul ID3. Totuși, și această metodă are anumite neajunsuri.

Pornind de la soluția pe care ati dat-o la punctul precedent, clonați fiecare exemplu în parte de K ori (unde K este un număr suficient de mare). Apoi adăugați N noi atrbute de intrare care iau valorile True / False cu o probabilitate uniformă. Cum va arăta arborele de decizie care rezultă? Explicați acum de ce poate să eșueze pruning-ul de tip bottom-up.

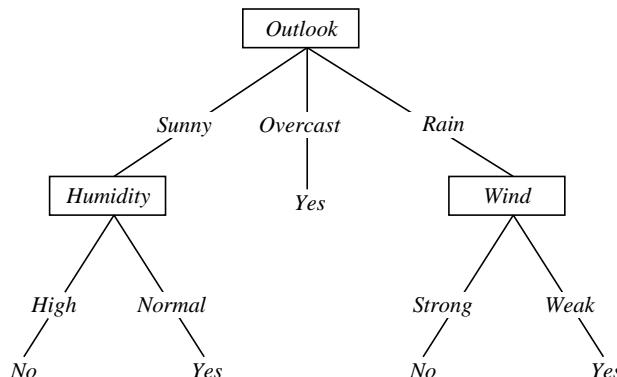
50.

(Algoritmul ID3: întrebări recapitulative; compararea strategiilor de pruning: reduced-error pruning vs. rule post-pruning)

CMU, 2005 fall, T. Mitchell, A. Moore, HW1, pr. 1

Considerăm următorul set de date de antrenament, împreună cu arborele de decizie care a fost învățat pe aceste date de către algoritmul ID3 (fără a se aplica vreo strategie de post-pruning).

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



a. Demonstrați că alegerea atributului *Wind* pe nivelul secund din arbore este corectă, arătând că în cazul lui câștigul de informație este superior față de cazurile / posibilitățile alternative.

b. Orice arbore de decizie poate fi exprimat sub forma unui set de reguli, scriind căte o regulă pentru fiecare nod-frunză (de fapt, pentru întreaga cale începând cu nodul-rădăcină și continuând până la nodul-frunză respectiv). *Pre-condițiile* dintr-o astfel de regulă corespund secvenței / succesiunii de atribute testate de-a lungul căii respective. De exemplu, nodul-frunză cel mai din stânga din arborele de mai sus corespunde regulii următoare:

IF (Outlook = sunny AND Humidity=High) THEN PlayTennis = No

Scrieți regulile corespunzătoare celorlalte noduri-frunză. Observați faptul că acest set de reguli produce clasificări care sunt identice cu cele din arborele de decizie de mai sus, pentru orice instanță posibilă [LC: de antrenament sau de test].

c. Este posibil să „traducem” orice arbore de decizie într-un set de reguli care reprezintă un clasificator echivalent. Invers, este oare posibil să „traducem” orice set de reguli într-un arbore echivalent? Explicați succint sau, dacă este cazul, dați un contraexemplu.

d. La curs am discutat despre strategii de post-pruning pentru arbori de decizie, folosind abordarea *reduced-error pruning* relativ la un set de date de validare, pentru a evita

overfitting-ul. Aici vom considera o strategie alternativă, care constă în a converti / „traduce“ arborele în setul de reguli echivalent și făcând apoi pruning pe reguli. În particular, vom considera că pruning-ul se execută în mod independent pe fiecare regulă în parte, parcurgând iterativ următorii pași:

- i. Determină în regula respectivă acele pre-condiții care, dacă sunt înlăturate (în mod individual) produc o îmbunătățire a acurateții pe setul de date de validare.²⁵⁹
- ii. Dacă există astfel de pre-condiții,
identific-o pe cea care îmbunătășește cel mai mult acuratețea pe setul de date de validare,
șterge-o din regulă și apoi *iterează* mai departe;
în caz contrar *oprește* pruning-ul pe această regulă.

Considerați cele două strategii de pruning (adică pruning pe arbore, versus pruning pe setul de reguli echivalent). Este oare adevărat faptul că aceste două strategii de pruning produc clasificatori trunchiați (engl., pruned) care sunt echivalenți? Adică, produc oare arborele de decizie trunchiat și setul de reguli trunchiate clasificări identice (engl., equivalent) pentru orice instanță posibilă? Explicați de ce da, sau de ce nu.

51.

(Post-pruning pentru arborele ID3:
folosirea testului statistic χ^2 pentru identificarea
partiționărilor care sunt semnificative d.p.v. statistic)
CMU, 2007 fall, Carlos Guestrin, HW1, pr. 1.3.1

La curs am menționat că arborii de decizie pot să manifeste fenomenul de “overfitting” și, în consecință, pentru a generaliza cât mai corect, trebuie să limităm „complexitatea“ arborilor pe care îi învățăm.

O modalitate de a face pruning pe arbori de decizie — de obicei, prin parcursare *de jos în sus*, după antrenare — este să analizăm din punct de vedere statistic instanțele asignate fiecărui nod de test din arbore. Mai precis, vom analiza etichetele acestor instanțe, cu *obiectivul* de a vedea cât de probabil este d.p.v. statistic să „observăm“ proporția acestor etichete în cazul în care atributul de intrare ales în nodul respectiv nu se află de fapt în corelație cu valorile atributului-țintă (etichetele).

În acest sens putem folosi *testul χ^2* al lui Pearson, care este un exemplu clasic de test bazat pe o *ipoteză statistică*. Vom pleca de la ipoteza că atributul de intrare asociat nodului respectiv nu este corelat cu atributul de ieșire (reprezentat de etichete), și vom verifica dacă datele „observate“ conduc în mod ferm la respingerea acestei ipoteze.

Mai exact, pornind de la datele de antrenament, vom calcula o anumită *mărime statistică*, despre care știm că are o distribuție de tipul χ^2 în cazul în care cele două entități (atributul și etichetele) nu sunt corelate. Apoi, vom verifica dacă acea *mărime statistică* calculată are (sau nu) o *valoare* prea puțin probabil să fi fost generată de o distribuție χ^2 . În cazul afirmativ, vom respinge ipoteza că cele două entități nu sunt corelate. În cazul negativ, vom înlocui nodul de test respectiv, împreună cu întregul subarbore atașat lui, cu un nod de decizie.

Concret, să presupunem că am învățat un arbore de decizie. Notăm cu S setul de exemple de antrenament ale căror drumuri de clasificare trec prin nodul a cărui semnificație

²⁵⁹Acuratețea unei reguli este [o fracție, dată de] numărul de *predicții corecte* pe care le face această regulă, raportat la numărul total de *predicții*.

statistică dorim să o testăm. Să zicem că în nodul respectiv se testează un atribut discret X care poate lua valorile $1, 2, \dots, k$. Fie p numărul de exemple din S care au eticheta $+$, iar $n = |S| - p$ numărul de exemple din S având eticheta $-$. Fie

S_i submulțimea lui S formată din instanțele care au $X = i$,

p_i numărul de exemple din S_i având eticheta $+$,

$n_i = |S_i| - p_i$ numărul de exemple din S_i având eticheta $-$.

În plus, fie $\bar{p}_i = p \cdot \frac{|S_i|}{|S|}$ și $\bar{n}_i = |S_i| - \bar{p}_i$. Acestea sunt valorile „așteptate“ ale lui p_i și respectiv n_i , în ipoteza că atributul X nu este corelat cu atributul de ieșire (*clasa*).

Folosind datele de mai sus, vom calcula *mărimea numerică corespunzătoare testului statistic χ^2* :

$$\chi^2 = \sum_{i=1}^k \left(\frac{(p_i - \bar{p}_i)^2}{\bar{p}_i} + \frac{(n_i - \bar{n}_i)^2}{\bar{n}_i} \right)$$

În ipoteza că atributul X nu este corelat cu *clasa*, mărimea calculată mai sus urmează *distribuția χ^2* .

Un *parametru* al distribuției χ^2 este așa-numitul *grad de libertate*. Pentru X , gradul de libertate este $k - 1$ atunci când se lucrează cu doar două etichete (+ și -).²⁶⁰ În particular, pentru un atribut binar gradul de libertate este 1.

După ce vom calcula valoarea mărimii statistice χ^2 pornind de la date, o vom compara cu o *valoare critică*, care reprezentă un prag astfel încât probabilitatea ca mărimea statistică calculată să depășească acel prag este de cel mult α în cazul în care variabilele nu sunt corelate. α este *nivelul de încredere* (engl., confidence level) și de obicei se consideră $\alpha = 0.05$, fiind astfel siguri în proporție de 95% că o partitioare (engl., split) care trece testul este semnificativă d.p.v. statistic.

De exemplu, atunci când gradul de libertate este 1 iar $\alpha = 0.05$, valoarea critică este 3.841; pentru 2 grade de libertate și același α , ea este 5.991. Spunem că o partitioare este *statistic semnificativă* dacă mărimea calculată χ^2 trece de valoarea critică.

Y	X_1	X_2	Nr. apariții
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

52.

(Algoritmul ID3: Adevărat sau Fals?)

CMU, 2004, fall, final exam, pr. 1.b

Învățăm un arbore de decizie folosind algoritmul ID3 standard, fără pruning. Atributele de intrare (X_1, X_2, \dots, X_m) sunt categoriale, iar atributul de ieșire (Y) este de asemenea categorial.

²⁶⁰Dacă în loc de două etichete am fi considerat n etichete [de clasă] diferite, atunci am fi folosit o statistică asemănătoare, cu $(n - 1)(k - 1)$ grade de libertate.

Marcați cu A (adevărat) sau F (fals) fiecare din afirmațiile de mai jos și dați în fiecare caz o explicație succintă, însotită eventual de un exemplu sau un contraexemplu minimalist.

a. Dacă X_i și Y văzute ca variabile aleatoare sunt independente (raportat la distribuția probabilistă care a generat datele de antrenament), atunci X_i nu va apărea în arborele de decizie.

b. Dacă $IG(Y, X_i) = 0$, atunci atributul X_i nu va apărea în arborele de decizie.

c. Adâncimea maximă a arborelui de decizie este de cel mult m .

Notă: Dacă arborele este format doar din nodul rădăcină, ceea ce corespunde cazului în care toate exemplele de antrenament sunt identic clasificate, atunci se consideră că adâncimea arborelui este 0.

d. Dacă sunt R exemple de antrenament, atunci adâncimea maximă a arborelui de decizie este de cel mult $1 + \log_2 R$.

e. Dacă sunt R exemple de antrenament, iar unul dintre atrbutele de intrare are R valori distincte și ia valori v_1, \dots, v_R în mod injectiv (pe mulțimea formată de exemplele de antrenament), atunci arborele de decizie va avea adâncimea 0 sau 1.

53.

(Implementare: algoritmul ID3)

Liviu Ciortuz, 2016

Pornind de la pseudo-codul algoritmului ID3 în varianta de bază (adică, folosind doar atrbute cu valori discrete) dată în cartea *Machine Learning* de Tom Mitchell la pagina 56, elaborați o implementare, în limbajul de programare preferat. „Inima“ acestui algoritm este calculul câștigului de informație (a se vedea pr. 34). Programul va conține o funcție de antrenare și una de testare. Ca input (la linia de comandă), programul va primi

- numele unui fișier în format SSV (engl., space-separated values), conținând instanțele de antrenament, câte una pe fiecare linie, variabila de ieșire aflându-se pe ultima poziție;
- numele unui al doilea fișier, tot în format SSV, conținând instanța de test sau instanțele de test (de asemenea, câte una pe fiecare linie).

Optional, pe prima linie a fișierului de antrenament vor fi indicate pentru fiecare atrbut numele lui, precum și numărul de valori pe care le ia respectivul atrbut (drept separatori, se vor folosi tot spații). În acest caz, primul caracter din această linie-comentariu va fi %. Folosiți o modalitate convenabilă pentru a afișa arborele ID3 obținut, de exemplu ca program în logica propozițiilor cu variabile cu valori multiple.

Pentru a testa programul, puteți folosi de pildă datele de la problema 2.

Ulterior, veți putea adăuga programului una dintre următoarele extensii (sau chiar ambele):

- variabile cu atrbute continue; testați programul pe datele de la problema 10 din prezentul capitol și pe cele de la problema 11.b de la capitolul *Învățare bazată pe memorare*,²⁶¹
- o funcție care, în vederea contracarării fenomenului de overfitting, face trunchierea / pruning-ul arborelui ID3, în varianta “reduced-error”, folosind un al treilea set / fișier de instanțe, pentru validare; a se vedea cartea *Machine Learning* de Tom Mitchell, pag. 69-70.²⁶²

²⁶¹Pentru testarea variantei algoritmului ID3 care folosește atât atrbute discrete cât și atrbute continue, puteți folosi datele de la problema 47, precum și cele de la problema 12.

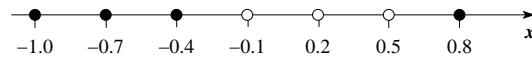
²⁶²Pentru testare, vedeti problemele CMU, 2008 spring, T. Mitchell, W. Cohen, HW1, pr. 2, CMU, 2012 spring, Roni Rosenfeld, HW3 și / sau CMU, 2011 fall, T. Mitchell, A. Singh, HW1, pr. 2.

Indicație: Pentru a lucra în mod convenabil cu atributele discrete, mai ales atunci când aceste valori nu sunt numerice, în loc să folosiți (în timpul procesărilor) valorile lor aşa cum apar în fișierele de date, puteți crea un *vocabulary* (ca vector în care se memorează valorile tuturor acestor variabile) și apoi să înregistrați în tabelele de date (în locul valorile atributelor discrete) indecșii corespunzători „intrărilor“ din *vocabulary*.

4.2.2 Algoritmul AdaBoost

54. (Algoritmul AdaBoost: întrebări în legătură cu aplicarea algoritmului pe un set de date din \mathbb{R})
CMU, 2011 fall, T. Mitchell, A. Singh, HW6, pr. 3.2-8

Considerăm setul de date de antrenament din figura următoare. Vom folosi algoritmul AdaBoost cu compași de decizie în rolul de ipoteze „slabe“.



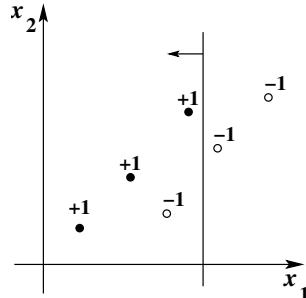
- Determinați separatorul decizional corespunzător primei ipoteze „slabe“, h_1 . Desenați-l pe figura de mai sus și indicați [eventual printr-o mică săgeată perpendiculară pe acest separator] care este zona clasificată cu +.
- Calculați ε_1 și α_1 . Cât este acuratețea obținută de AdaBoost la antrenare dacă oprim acum algoritmul?
- Cât va fi valoarea noilor probabilități / ponderi $D_2(i)$ pentru fiecare dintre cele şapte exemple de antrenament? (Atenție! Nu uitați să faceți normalizarea cu ajutorul factorului Z_1 .)
- Desenați separatorul decizional corespunzător celei de-a doua ipoteze „slabe“, h_2 . Indicați iarăși zona de decizie corespunzătoare clasei +.
- Care sunt exemplele de antrenament cărora le va fi asignată cea mai mică pondere / probabilitate după ce algoritmul AdaBoost va fi terminat cea de-a doua sa iteratăie?
- Se îmbunătățește oare acuratețea la antrenare obținută de AdaBoost la a doua iteratăie în raport cu cea obținută la prima iteratăie?

55. (AdaBoost: întrebări în legătură cu aplicarea algoritmului pe un set de date din \mathbb{R}^2)
CMU, 200X spring, midterm, pr. 3
MIT, 2006 fall, Tommi Jaakkola, final, pr. 2

Folosind algoritmul AdaBoost, vrem să obținem un ansamblu de compași de decizie (engl., decision stumps) h_t , de forma

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right).$$

În figura alăturată sunt desenate câteva puncte (instante) etichetate în planul bidimensional, precum și primul compas de decizie care a fost ales de către algoritmul AdaBoost. Un compas de decizie oarecare produce valori binare ± 1 , ținând cont doar de un anumit *prag* (engl., the split point). Săgeata mică din figură, care este perpendiculară pe dreapta care reprezintă compasul de decizie indică *zona de decizie* pentru care compasul de decizie va produce valoarea $+1$.



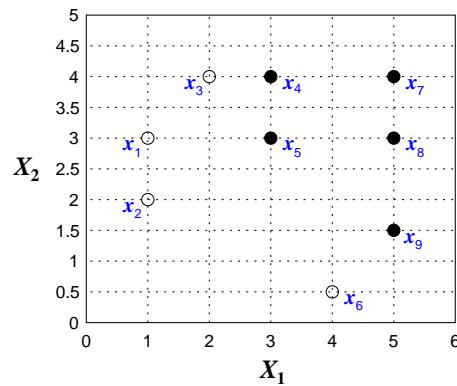
- Încercuiți toate acele instanțe din figură pentru care ponderea / probabilitatea [atribuită de către AdaBoost] va crește ca urmare a incorporării [în ipoteza combinată H a] primului compas de decizie. Justificați răspunsul în mod riguros.
- Desenați pe aceeași figură un compas de decizie care va putea fi selectat la următoarea iterare a algoritmului AdaBoost. Veți trasa atât dreapta care reprezintă compasul de decizie cât și [o săgeată care să indice] zona sa de decizie pozitivă. Justificați în mod riguros.
- Va fi oare coeficientul / votul α_2 , care este asociat celui de-al doilea compas de decizie mai mare decât α_1 , coeficientul [din ansamblul H] pentru primul compas de decizie? Cu alte cuvinte, vom avea oare $\alpha_2 > \alpha_1$? Justificați în mod riguros.

56.

(AdaBoost: aplicare pe un set de date din \mathbb{R}^2)

*adaptare facută de Liviu Ciortuz, după
MIT, 2004 fall, Tommi Jaakkola, final, pr. 1*

Fie setul de exemple de antrenament (instante etichetate) din figura de mai jos. Simbolurile \bullet și \circ desemnează etichete pozitive și respectiv etichete negative. Pentru a rezolva această problemă de clasificare, vă cerem să aplicați algoritmul AdaBoost folosind drept ipoteze „slabe” compași de decizie (engl., decision stump).



La fiecare iterare de boosting veți selecta acel compas de decizie care minimizează eroarea ponderată la antrenare (engl., weighted training error). În cazul în care există mai mulți compași de decizie care au [o aceeași!] cea mai bună eroare ponderată la antrenare, veți putea alege unul dintre ei în mod arbitrar.

- Pe figura dată, desenați primul compas de decizie și etichetați-l cu h_1 , indicând de asemenea cu $+/-$ cele două zone de decizie pe care le determină acest compas.
- Pe aceeași figură, scrieți în apropierea fiecărei instanțe probabilitatea asignată ei după prima iterare executată de algoritmul AdaBoost. De asemenea, încercuiți instanțele care au probabilitatea cea mai mare. Justificați răspunsul în mod riguros.

c. Cât este eroarea ponderată la antrenare produsă de primul compas de decizie după prima iterație, adică după ce probabilitățile asociate instanțelor au fost recalculate? Justificați.

d. Desenați pe figura dată cel de-al doilea compas de decizie și etichetați-l cu h_2 , indicând de asemenea cu $+$ / $-$ cele două zone de decizie pe care le determină acest compas.

s	0.5	2.5	3.5	4.5
$err_{D_2}(X_1 < s)$				
$err_{D_2}(X_1 \geq s)$				

s	1	1.75	2.5	3.5
$err_{D_2}(X_2 < s)$				
$err_{D_2}(X_2 \geq s)$				

e. Există oare instanțe de antrenament care sunt clasificate eronat de către H_2 , ipoteza combinată produsă de AdaBoost după două iterări? Justificați răspunsul în mod riguros.

	i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\alpha_1 = \dots$	$h_1(x_i)$									
$\alpha_2 = \dots$	$h_2(x_i)$									
	$H_2(x_i)$									

$$\Rightarrow err_S(H_2) = \dots$$

f. Dacă răspunsul pe care l-ați dat la punctul e este pozitiv, ce se poate spune dacă se execută încă o iterărie (adică, a treia)?

s	0.5	2.5	3.5	4.5
$err_{D_3}(X_1 < s)$				
$err_{D_3}(X_1 \geq s)$				

s	1	1.75	2.5	3.5
$err_{D_3}(X_2 < s)$				
$err_{D_3}(X_2 \geq s)$				

Desenați pe figura dată cel de-al treilea compas de decizie și etichetați-l cu h_3 , indicând de asemenea cu $+$ / $-$ cele două zone de decizie pe care le determină acest compas.

	i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
$\alpha_1 = \dots$	$h_1(x_i)$									
$\alpha_2 = \dots$	$h_2(x_i)$									
$\alpha_3 = \dots$	$h_3(x_i)$									
	$H_3(x_i)$									

$$\Rightarrow err_S(H_3) = \dots$$

Dacă eroarea la antrenare este acum 0, trasați separatorul decizional determinat de algoritmul AdaBoost.

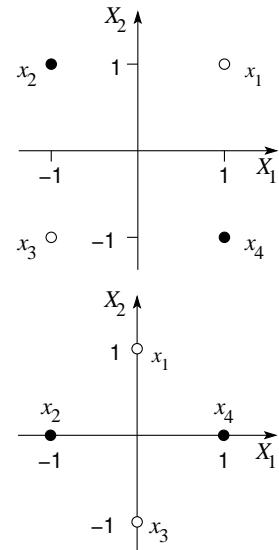
Care ar fi rezultatul dacă AdaBoost ar alege ca a treia ipoteză „slabă“ un alt (cel mai bun!) compas de decizie (h'_3) în locul lui h_3 ?

57.

(Algoritmul AdaBoost: aplicare pe dataset-uri de tip XOR)

*CMU, 2011 fall, T. Mitchell, A. Singh, HW6, pr. 3.1
 CMU, 2007 spring, Carlos Guestrin, HW2, pr. 2.2*

- a. Considerați setul de date XOR, reprezentat ca de obicei în planul bidimensional. Presupunând că algoritmul AdaBoost folosește compași de decizie pe post de ipoteze „slabe“, va fi el oare capabil să obțină o acuratețe mai bună de 50% pe acest set de date? Justificați răspunsul dumneavoastră în mod riguros. (Ca de obicei, clasa +1 a fost reprezentată prin simbolul •, iar clasa -1 prin simbolul ○.)



În cele ce urmează veți aplica același algoritm AdaBoost pe setul de date din figura alăturată (obținut prin rotirea dataset-ului XOR cu 45° la stânga și făcând apoi rescalarea).

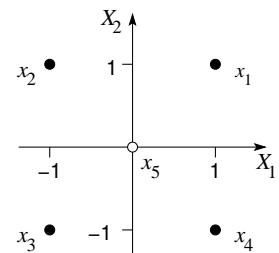
- b. Arătați cum lucrează AdaBoost pe acest [al doilea] set de date, considerând că numărul de iterații de efectuat este $T = 4$. Vă reamintim că pentru fiecare $t = 1, \dots, 4$ va trebui să calculați numerele $\varepsilon_t, \alpha_t, Z_t$, precum și distribuția probabilistă $D_t(i)$ pentru $i = 1, \dots, 4$. De asemenea, la fiecare iteratie t veți trasa separatorul decizional corespunzător ipotezei h_t alese de clasificatorul „slab“ (A) și veți indica printr-o mică săgeată desenată perpendicular pe separator zona de decizie corespunzătoare etichetei +1.
- c. Cât va fi la finalul celor $T = 4$ iterații eroarea la antrenare produsă de [combinăția liniară H livrată la ieșire de către] algoritmul AdaBoost?
- d. Este oare setul de date (cel folosit la punctele *b* și *c*) liniar separabil? Explicați de ce algoritmul AdaBoost se comportă mai bine decât un [singur] compas de decizie pe acest set de date.

58.

(Algoritmul AdaBoost: aplicare pe un set de date din \mathbb{R}^2)

CMU, 2010 fall, Aarti Singh, midterm, pr. 6.1-2

În acest exercițiu veți aplica algoritmul AdaBoost folosind drept clasificatori „slabi“ compași de decizie pe setul de date de antrenament prezentat în figura alăturată. (Atenție! Sunt patru instanțe pozitive și una negativă.)

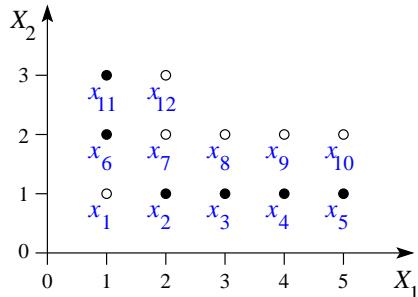


- a. Care dintre aceste exemple de antrenament vor avea probabilitățile ($D_t(i)$) mărite la sfârșitul primei iterații? Încercuiți-le pe desen.
- b. Cât de multe iterații vor fi necesare pentru a atinge eroare zero la antrenare? Justificați elaborând toate detaliile necesare.
- c. Puteți adăuga încă un exemplu (instantă de antrenament) la setul de date de mai sus astfel încât algoritmul AdaBoost să obțină în [doar] două iterații eroare la antrenare zero? Dacă nu, explicați de ce nu este posibil aşa ceva.
- d. Care credeți că este motivul (principal) pentru care se folosesc clasificatori „slabi” (și nu clasificatori mai puternici / „tari”) în conjuncție cu algoritmul AdaBoost?
59. (Algoritmul AdaBoost și învățabilitate empirică γ -slabă:
o margine superioară pentru numărul de iterații de efectuat
până la atingerea erorii empirice 0)
CMU, 2010 fall, Aarti Singh, midterm, pr. 6.3
CMU, 2011f, Eric Xing, HW5, pr. 3.2.ab
- Să presupunem că rulăm algoritmul AdaBoost pe m exemple de antrenament și, de asemenea, că la fiecare iterație (t) eroarea ponderată la antrenare (ε_t) produsă de ipoteza „slabă” h_t este de cel mult $1/2 - \gamma$, unde $\gamma > 0$ nu depinde de t . Determinați numărul de iterații (T) pe care trebuie să le execute algoritmul AdaBoost pentru ca ipoteza combinată H să fie consistentă cu cele m exemple de antrenament, adică să obțină eroare zero la antrenare. Trebuie să exprimați răspunsul doar în funcție de m și de γ .
- Sugestie:* Cât este eroarea la antrenare atunci când un singur exemplu este clasificat eronat?
60. (AdaBoost și non-învățabilitate γ -slabă:
exemplificare pe date din \mathbb{R}^2)
prelucrare de Liviu Ciortuz, după
CMU, 2006 spring, Carlos Guestrin, final, pr. 3

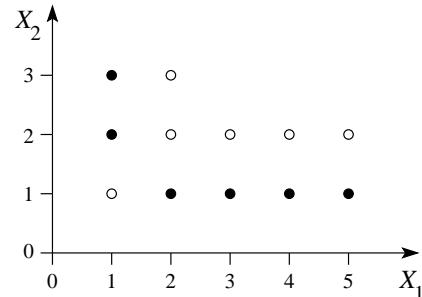
Considerând setul de date de antrenament din figura / figurile de mai jos, am dori să „învățăm” un clasificator care să separe instanțele pozitive de instanțele negative, folosind algoritmul AdaBoost. Fiecare instanță x_i are o etichetă $y_i \in \{+1, -1\}$; eticheta +1 corespunde simbolului •, iar eticheta -1 corespunde simbolului ○.

Vom folosi ipoteze „slabe”, ale căror granițe de decizie (engl., decision boundaries) sunt paralele cu una din axele de coordonate, adică separatorul este fie vertical, fie orizontal. Puteți gândi aceste ipoteze „slabe” ca fiind compași de decizie, pentru care pragurile de separare (engl., threshold splits) sunt situate fie pe axa X fie pe axa Y .

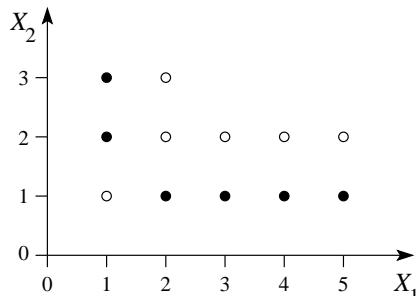
La fiecare iterație t , vom alege acea ipoteză „slabă” h_t care maximizează acuratețea ponderată la antrenare în raport cu ponderile / probabilitățile curente D_t , adică alegem h_t care maximizează $\sum_i D_t(i) \cdot 1_{\{h_t(x_i)=y_i\}}$. Ca de obicei, se consideră că $h_t(x)$ ia valori doar în multimea $\{+1, -1\}$, după cum ipoteza h_t clasifică instanța x ca fiind pozitivă ori negativă.



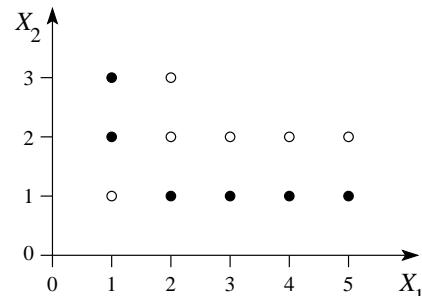
(a) Granița de decizie după prima iterare.



(b) Granița de decizie după a doua iterare.



(c) Exemplul / exemplele cu cea mai mică pondere.



(d) Exemplul / exemplele cu cea mai mare pondere.

- Desenați pe figura (a) granița de decizie determinată de algoritmul AdaBoost după prima iterare (adică, după ce a fost aleasă prima ipoteză „slabă“). Nu uitați să indicați ce parte a planului euclidian este clasificată cu '+' și respectiv '-'.
- Efectuați acum cea de-a doua iterare de boosting. Desenați granițele de decizie ale ambelor ipoteze „slabe“ din figura (b). AdaBoost combină deciziile celor două ipoteze „slabe“. Indicați care sunt regiunile din plan [în care sunt situate punctele] clasificate de către boosting-ul cu 2 ipoteze „slabe“ cu '+' și respectiv cu '-'.
- La problema 22.v am arătat că $\alpha_i > \alpha_j \Leftrightarrow \varepsilon_i < \varepsilon_j$. Verificați că această relație este într-adevăr satisfăcută pentru primele două iterări ale algoritmului AdaBoost pe setul de antrenament dat.
- Marcați în figura (c) exemplul / exemplele cu cea mai mică pondere / probabilitate (D_{t+1}) după primele două iterări de boosting ($t = 2$).
- Marcați în figura (d) exemplul / exemplele cu cea mai mare pondere / probabilitate (D_{t+1}) după primele două iterări de boosting ($t = 2$).
- Câte exemple de antrenament sunt clasificate eronat după primele două iterări de boosting?
- Folosind acest set de date și ipotezele „slabe“ menționate mai sus, va obține oare AdaBoost vreodată eroare la antrenare zero? Justificați în mod riguros, folosind eventual o *implementare*.

61. (Algoritmul AdaBoost: margini de votare; aplicație)
Liviu Ciortuz, 2020

În această problemă veți folosi setul de date de la problema 24.

- Pentru fiecare dintre iterațiile $t = 1, t = 2$ și $t = 3$, calculați $\text{Margin}_t(x_i)$, pentru $i = 1, \dots, 8$. (Pentru definiția noțiunii de margine de votare, vedeti enunțul problemei 27.)
- Verificați, utilizând distribuțiile probabiliste D_t care au fost calculate la problema 24, că într-adevăr are loc implicația $\text{Margin}_t(x_i) > \text{Margin}_t(x_j) \Rightarrow D_{t+1}(i) < D_{t+1}(j)$ — care a fost demonstrată pentru cazul general la problema 27.b — pentru cazul particular $t = 2$, $i = 1$ și $j = 4$, precum și pentru cazul $t = 2$, $i' = 4$ și $j' = 8$.
- Pentru fiecare dintre iterațiile $t = 1, t = 2$ și $t = 3$ în parte, reprezentați grafic: *i. distribuția empirică* a marginilor de votare care au fost calculate la punctul a , precum și *ii. distribuția empirică cumulativă* a acestor margini.²⁶³

62. (Clasificare de documente:
 selecție de trăsături folosind algoritmul AdaBoost
 cu compași de decizie pentru attribute booleene)
CMU, 2007 spring, midterm, pr. 3
MIT, 2003 fall, Tommi Jaakkola, final, pr. 3.2-4

Considerăm o problemă de clasificare de texte, în care un document X este reprezentat sub forma unui vector de trăsături binare relativ la cuvintele din dicționarul limbii în care este scris documentul respectiv. Din punct de vedere formal, putem scrie $X = [X_1, X_2, X_3, \dots, X_m]$, unde $X_j = 1$ în cazul în care cuvântul de pe poziția j din dicționar este prezent în documentul X , și 0 în caz contrar.

Vom considera acum că algoritmul AdaBoost folosește niște ipoteze „slabe“ desemnate formal ca perechi (j, y) unde j este indicele unui cuvânt de dicționar, iar y este clasa selectată, cu $y \in \{-1, +1\}$. Din punct de vedere *intuitiv*, acest lucru se poate exprima astfel: fiecare ipoteză slabă este [o pereche formată din] un cuvânt împreună cu o etichetă reprezentând clasa asociată. De exemplu, considerând cuvântul **football** și clasele {**sports**, **non-sports**}, atunci vom avea două ipoteze „slabe“ relative la acest cuvânt, și anume

- dacă documentul conține cuvântul **football**, [vom] prezice clasa **sports**; altfel, [vom] prezice clasa **non-sports**;
- dacă documentul conține cuvântul **football**, [vom] prezice clasa **non-sports**; altfel, [vom] prezice clasa **sports**.

- a. Cât de multe ipoteze „slabe“ există în acest model?

Acest algoritm de tip boosting poate fi folosit pentru a face *selecția de trăsături*. Rulăm algoritmul și selectăm trăsăturile în *ordinea în care au fost identificate* de către algoritm.

- b. Este oare posibil ca acest algoritm de boosting să selecteze o aceeași ipoteză „slabă“ mai mult decât de o singură dată? Justificați.

²⁶³Ca aplicație practică, vedeti problema CMU, 2007 spring, Carlos Guestrin, HW2, pr. 2.3, care în Companionul practic al culegerii de exerciții de învățare automată, <https://profs.info.uaic.ro/~ciortuz/ML.ex-book/implementation-exercises/ML.ex-book.Companion.pdf>, este problema 36.b.

c. Să zicem că vrem să facem o ordonare (engl, ranking) a trăsăturilor în funcție de informația mutuală [individuală] relativă la variabila care reprezintă clasa (y), adică $IG(y; X_j)$. Va fi oare această ordonare mai „informativă“ / bună decât ordonarea produsă de către algoritmul AdaBoost? Justificați.

63.

(O proprietate interesantă:
orice mulțime de instanțe din \mathbb{R} care sunt etichetate în mod consistent,
este γ -slab învățabilă cu ajutorul compașilor de decizie)

*prelucrare de Liviu Ciortuz, după
■ Stanford, 2016 fall, A. Ng, J. Duchi, HW2, pr. 6.abc*

Rezumat: În această problemă vom arăta că, în ipoteza că instanțele de antrenament cu care lucrăm au un singur atribut, x , care ia valori în \mathbb{R} , iar aceste instanțe (x_i , cu $i = 1, \dots, m$) sunt etichetate în mod „consistent“ (adică, necontradictoriu), există o constantă $\gamma > 0$ care reprezintă o garanție (engl., guarantee, or edge) pentru învățabilitate empirică „slabă“, folosind compași de decizie în conjuncție cu algoritmul AdaBoost.²⁶⁴

Din punct de vedere *analitic*, compașii de decizie determinăți de praguri de separare (engl., thresholding-based decision stumps) pot fi definiți ca niște funcții-prag (sau, funcții-treaptă), care sunt indexate pe de o parte de un prag $s \in \mathbb{R}$ și pe de altă parte de un semn $+/-$, în felul următor:

$$\phi_{s,+}(x) = \begin{cases} 1 & \text{dacă } x \geq s \\ -1 & \text{dacă } x < s \end{cases} \quad \text{iar} \quad \phi_{s,-}(x) = \begin{cases} -1 & \text{dacă } x \geq s \\ 1 & \text{dacă } x < s. \end{cases}$$

Prin urmare, $\phi_{s,+}(x) = -\phi_{s,-}(x)$ pentru orice $x \in \mathbb{R}$.

Dată fiind un set de date de antrenament consistent etichetate $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, cu $x_i \in \mathbb{R}$ și $y_i \in \{-1, +1\}$ pentru $i = 1, \dots, m$, obiectivul nostru este să investigăm existența unei *garanții* pentru învățabilitate „slabă“ pe setul S . Vom demonstra că există o valoare $\gamma > 0$, astfel încât pentru orice distribuție probabilistă p definită pe S , putem găsi un prag $s \in \mathbb{R}$ cu proprietatea următoare:

$$\text{error}_p(\phi_{s,+}) \leq \frac{1}{2} - \gamma \quad \text{sau} \quad \text{error}_p(\phi_{s,-}) \leq \frac{1}{2} - \gamma,$$

unde

$$\text{error}_p(\phi_{s,+}) \stackrel{\text{def.}}{=} \sum_{i=1}^m p_i \cdot 1_{\{y_i \neq \phi_{s,+}(x_i)\}} \quad \text{și} \quad \text{error}_p(\phi_{s,-}) \stackrel{\text{def.}}{=} \sum_{i=1}^m p_i \cdot 1_{\{y_i \neq \phi_{s,-}(x_i)\}}.$$

²⁶⁴Noțiunile de învățabilitate empirică „slabă“ și garanție (γ) pentru învățabilitate empirică „slabă“ au fost definite la problema 23.e. Folosind notațiile de acolo, vă reamintim că *garanția* $\gamma \in (0, 1/2)$ are proprietatea că la orice iterație t a algoritmului AdaBoost există o ipoteză „slabă“ h_t astfel încât

$$\text{error}_{D_t}(h_t) \stackrel{\text{def.}}{=} \sum_{i=1}^m D_t(i) \cdot 1_{\{h_t(x_i) \neq y_i\}} \leq \frac{1}{2} - \gamma.$$

În formula aceasta, simbolul $1_{\{\dots\}}$ desemnează binecunoscuta funcție indicator. Concret, funcția $1_{\{h_t(x_i) \neq y_i\}}$ ia valoarea 1 pentru acel i pentru care $h_t(x_i) \neq y_i$, și 0 în caz contrar. Vă reamintim că p este o distribuție probabilistă discretă dacă $p_i \geq 0$ pentru $i = 1, \dots, m$ și $\sum_{i=1}^m p_i = 1$.

Facem *presupunerea* că toate instanțele de antrenament sunt distincte, deci nu există $x_i = x_j$ cu $i \neq j$. De asemenea, vom presupune — fără a reduce generalitatea; însă acest fapt ne va permite să simplificăm notațiile din problemă — că

$$x_1 > x_2 > \dots > x_m.$$

- a. Arătați că, dat fiind S , pentru orice prag $s \in \mathbb{R}$ există un anumit indice $m_0(s) \in \{0, 1, \dots, m\}$ astfel încât să aibă loc egalitățile

$$\begin{aligned} \text{error}_p(\phi_{s,+}) &= \frac{1}{2} - \frac{1}{2} \left(\sum_{i=1}^{m_0(s)} y_i p_i - \sum_{i=m_0(s)+1}^m y_i p_i \right) \\ \text{error}_p(\phi_{s,-}) &= \frac{1}{2} - \frac{1}{2} \left(\sum_{i=m_0(s)+1}^m y_i p_i - \sum_{i=1}^{m_0(s)} y_i p_i \right). \end{aligned}$$

Convenție: În expresii precum cele de mai sus, veți trata sumele indexate pe mulțimi vide ca fiind egale cu 0. Așadar, $\sum_{i=1}^0 a_i = 0$, oricare ar fi termenii a_i și, similar, $\sum_{i=m+1}^m a_i = 0$.

- b. Arătați că, dat fiind setul de date de antrenament S , există o valoare $\gamma > 0$, care poate depinde de m , numărul de instanțe din S (dar nu și de distribuțiile probabiliste p care pot fi definite pe S), astfel încât pentru orice astfel de distribuție p , putem găsi un indice m_0 cu proprietatea următoare:

$$|f(m_0)| \geq 2\gamma, \text{ unde } f(m_0) \stackrel{\text{def.}}{=} \sum_{i=1}^{m_0} y_i p_i - \sum_{i=m_0+1}^m y_i p_i.$$

Sugestie: Analizați diferența $f(m_0 + 1) - f(m_0)$. Care este valoarea pe care ați găsit-o pentru γ ?

- c. Înănd cont de răspunsurile pe care le-ați dat la punctele a și b , cât anume este *garanția* [pentru învățabilitate empirică „slabă”] pe care compași de decizie o furnizează pentru orice set de date de antrenament $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, considerând că toate instanțele $x_i \in \mathbb{R}$, cu $i = 1, \dots, m$ sunt distincte?

- d. Puteți indica o margine superioară (engl., upper bound) pentru numărul de compași de decizie necesari pentru a obține eroare la antrenare 0 pe un astfel de set de date de antrenament (S)?

64.

(Ansambluri de clasificatori „slabi”: compași de decizie și funcții cu baza radială)
MIT, 2011 fall, L. P. Kaelbling, HW5, pr. 1.3
MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.3

- a. Cât de *complex* credeți că poate deveni un clasificator atunci când asamblăm / combinăm compași de decizie? Arătați că, atunci când exemplele de antrenament sunt puncte distincte din \mathbb{R} , avem nevoie de cel mult $2m$ compași de decizie pentru a clasifica în mod corect m exemple de antrenament.

MIT, 2006 fall, Tommi Jaakkola, HW4, pr. 3.c

b. La acest punct vom considera [ca ipoteze „slabe“ pentru algoritmul AdaBoost generalizat] funcții cu baza radială (engl., radial basis functions, RBF) centrate în anumite instanțe de antrenament, adică

$$h(x; \theta) = y_i \exp(-\beta \|x - x_i\|^2),$$

unde $\beta \in \mathbb{R}^+$, iar parametrul θ pur și simplu identifică instanța de antrenament x_i . Aceste ipoteze „slabe“ returnează (ca predictii) valori reale $h(x; \theta) \in [-1, 1]$. Demonstrați că un ansamblu format din m astfel de ipoteze „slabe“ poate în principiu să clasifice o mulțime de m instanțe de antrenament în toate modurile posibile.

65.

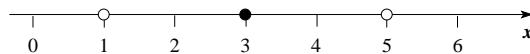
(Algoritmul AdaBoost generalizat: aplicare)

□ • · Liviu Ciortuz, 2020

La problema 29 am prezentat algoritmul AdaBoost generalizat, care poate fi parametrizat cu orice funcție de cost. La punctul a al acelei probleme am demonstrat (din punct de vedere *teoretic*) că atunci când se lucrează cu folosind funcția de cost [negativ] exponențială, respectiv „instanța“ a AdaBoost generalizat coincide cu algoritmul AdaBoost prezentat la problema 22.

Ne propunem — în cea mai mare parte a acestui exercițiu — să ilustrăm în mod *practic* acest lucru, pe date concrete.

Așadar, vom analiza ce se întâmplă atunci când aplicăm algoritmul AdaBoost generalizat cu funcție de cost [negativ] exponențială pe datele de la problema 25:



Atenție! Vom folosi compași de decizie, *inclusiv* compași cu prag *exterior* (engl., outside threshold) în raport cu datele de antrenament.

Vom limita analiza noastră pe de o parte la modul cum sunt selecționați compașii de decizie și pe de altă parte la probabilitățile asignate instanțelor de antrenament.

Prin urmare, ne vom uita mai întâi la *valorile* pe care le ia în cursul rulării acestui algoritm pe datele pe care tocmai le-am menționat *expresia* (96), adică derivata parțială a funcției $J_t(\alpha, \theta)$ calculată pentru $\alpha = 0$ și pentru θ (deci o funcție de parametrul θ):

$$\begin{aligned} \frac{\partial}{\partial \alpha} J_t(\alpha, \theta)_{|\alpha=0}, &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial \alpha} \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \theta))_{|\alpha=0} \\ &= \frac{1}{m} \sum_{i=1}^m dL(y_i f_{t-1}(x_i)) y_i h(x_i; \theta), \end{aligned}$$

unde, aşa cum ştim,

$$J_t(\alpha, \theta) = \frac{1}{m} \sum_{i=1}^m \text{Loss}(y_i f_{t-1}(x_i) + y_i \alpha h(x_i; \theta)),$$

iar Loss este funcția de cost / pierdere cu care lucrăm.

a. Arătați că $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta)_{|\alpha=0}$ are la iterată $t = 1$ expresia

$$\frac{\partial}{\partial \alpha} J_1(\alpha, \theta)_{|\alpha=0} = \begin{cases} -1/3 & \forall \theta \in (-\infty, 1] \cup (3, 5] \\ +1/3 & \forall \theta \in (1, 3] \cup (5, +\infty) \end{cases}$$

dacă semnele asociate compasului de decizie (h_1) sunt $+|-$;

$$\frac{\partial}{\partial \alpha} J_1(\alpha, \theta)_{|\alpha=0} = \begin{cases} +1/3 & \forall \theta \in (-\infty, 1] \cup (3, 5] \\ -1/3 & \forall \theta \in (1, 3] \cup (5, +\infty) \end{cases}$$

dacă semnele asociate compasului de decizie (h_1) sunt $-|+$.

Notă: Exact acestei expresii îi corespunde graficul care a fost prezentat la pagina 289.

Verificați apoi că $D_2(i) = \tilde{W}_1(i)$ pentru $i \in \{1, 2, 3\}$.

Observații:

1. Vă readucem aminte că parametrul θ desemează *pe lângă* pragul compasului de decizie și semnele asociate acestui compas de decizie. (Aceste semne sunt $+|-$ pentru graficul din stânga și respectiv $-|+$ pentru graficul din dreapta.)

2. Valorile lui θ pentru care se obține minimul lui $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta)_{|\alpha=0}$ sunt eligibile de către algoritmul AdaBoost la iterația curentă.²⁶⁵ Astfel, la iterația $t = 1$, după cum se observă în graficul de la pagina 289, putem alege $\theta \in (-\infty, 1] \cup (3, 5]$ cu semnele asociate $+|-$ sau, alternativ, $\theta \in (1, 3] \cup (5, +\infty)$ cu semnele asociate $-|+$. Observați că una dintre aceste posibilități, și anume, pragul $\theta = 0$ cu semnele asociate $+|-$, a fost ales la rezolvarea problemei 25.b (prima parte).

b. Procedând similar, calculați expresiile $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta)_{|\alpha=0}$ pentru iterațiile $t = 2$ și $t = 3$ și apoi obțineți graficele corespunzătoare.

Notă: În fiecare din cele două cazuri veți presupune că la iterația / iterațiile precedente s-au selectat compașii de decizie exact ca la rezolvarea problemei 25.b (prima parte).

c. Refaceți calculele de la punctul a, considerând că în locul funcției de cost [negativ] exponentiale se lucrează cu funcția de cost logistică.

66.

(Algoritmul AdaBoost: Adevărat sau Fals?)

CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final, pr. 8.f

MIT, 2002 fall, Tommi Jaakkola, midterm, pr. 5.3

MIT, 2001 fall, Tommi Jaakkola, midterm, pr. 4.2

CMU, 2014 spring, B. Poczos, A. Singh, midterm, pr. 1.8

MIT, 2001 fall, Tommi Jaakkola, final, pr. 1.4

CMU, 2010 fall, Aarti Singh, midterm, pr. 1.5

a. Eroarea la antrenare produsă de clasificatorul H_t obținut de AdaBoost — bazat pe combinația liniară determinată de ipotezele „slabe” — descrește monoton pe măsură ce crește numărul de iterații executate de algoritm.

b. Ponderile / „voturile” α_t asignate de către AdaBoost ipotezelor „slabe” h_t au tendința să scadă în timpul execuției algoritmului, pentru că eroarea ponderată produsă la antrenare (engl., weighted training error) de ipotezele „slabe” în general crește.

c. Ponderile / „voturile” α_t pe care algoritmul AdaBoost le asignează ipotezelor „slabe” h_t sunt optimale, în sensul că ele asigură o *acuratețe la antrenare* mai bună decât în cazul oricărora altor valori asignate acestor voturi / ponderi α_t .

²⁶⁵Mai general, se poate alege pentru θ o valoare pentru care expresia $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta)_{|\alpha=0}$ este negativă. Aici însă, ca și la problema 25, nu folosim această posibilitate.

- d. Folosind în algoritmul AdaBoost compași de decizie (cu rol de ipoteze „slabe“), este posibil să obținem granițe de decizie sub formă de parabolă (determinate, deci, de polinoame de ordinul al doilea).
- e. Unul dintre avantajele algoritmului AdaBoost este că nu produce niciodată supra-specializare (engl., overfitting).
- f. Învățăm un clasificator f folosind algoritmul AdaBoost cu ipoteze „slabe“ h . Forma funcțională a separatorului decizional determinat de f este aceeași cu forma funcțională a ipotezelor h , însă cu parametri diferiți. De exemplu, dacă ipotezele h au fost clasificatori liniari, atunci și f este un clasificator liniar.
67. (Adevărat sau Fals?
Acuratețe la antrenare vs. acuratețe la testare)
CMU, 2011 fall, T. Mitchell, A. Singh, midterm exam, pr. 1.2-3
- Presupunem că avem un set de date de imagini celulare de la pacienți cu și, respectiv, fără cancer.
- a. Să zicem că setul de date conține 900 de imagini de la pacienți fără cancer și 100 de imagini de la pacienți cu cancer. Dacă antrenăm un clasificator și obținem 85% acuratețe pe acest set de date, putem spune că acesta este un clasificator bun.
- b. Un clasificator care atinge 100% acuratețe pe setul de antrenament și 70% acuratețe pe setul de test este mai bun decât un clasificator care atinge 70% acuratețe pe setul de antrenament și 75% acuratețe pe setul de test.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

5 Mașini cu vectori-suport

Sumar

Noțiuni preliminare

- elemente [simple] de *calcul vectorial*; proprietăți elementare ale produsului scalar al vectorilor din \mathbb{R}^n , norma euclidiană (L_2) și norma L_1 în \mathbb{R}^n : ex. 1, ex. 34;
- elemente [simple] de *geometrie analitică*: ecuația unei drepte din planul euclidian, ecuația unui plan din \mathbb{R}^3 , ecuația unui hiper-plan din \mathbb{R}^n ; ecuația dreptei care trece prin două puncte date în planul euclidian: ex. 5.c; panta unei drepte perpendiculare pe o dreaptă dată: ex. 9.d, ex. 35, ex. 37;
- distanța (cu sau fără semn) de la un punct la o dreaptă (respectiv la un plan, sau mai general la un hiperplan): ex. 5, ex. 6.c, ex. 1;
- proprietăți de bază din *calculul matriceal*;
- calculul *derivatelor parțiale* [pentru funcții obținute prin compuneri de funcții elementare];
- *metoda lui Lagrange* pentru rezolvarea problemelor de *optimizare convexă cu restricții*: ex. 34, ex. din https://www.cs.helsinki.fi/u/jkivinen/teaching/sumale/Spring2014/kkt_example.pdf, ex. 74, ex. 75 de la capitolul de *Fundamente*;
- *separabilitate liniară*, separator optimal, *margine geometrică*: ex. 37, ex. 6.abc, ex. 9.

SVM cu margine “hard”: forma primală și forma duală

- (•) deducerea *formei primale* pentru problema SVM (cu margine “hard”), pornind de la principiul maximizării marginii geometrice: ex. 2;²⁶⁶
- (P0) o formă [simplă] echivalentă cu *forma primală* a problemei de optimizare SVM: ex. 7;
- *exemplificarea* identificării *separatorului optimal* și a *vectorilor-suport*, pornind de la condițiile din forma primală: ex. 3-5, ex. 35, ex. 37, ex. 38 și CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW4, ex. 4.1-4;
- calcularea *erorii* la cross-validation “leave-one-out” atunci când se folosește o SVM liniară cu margine “hard”: ex. 4.d, ex. 36;
- exemple de [funcții de] *mapare a atributelor*, cu scopul de a obține separabilitate liniară: ex. 6.d, ex. 8, ex. 9.b, ex. 9, ex. 39.bd, ex. 40.a; rezolvarea directă a problemei SVM primale în [noul] spațiu de trăsături; identificarea separatorului neliniar din spațiul inițial [de trăsături]: ex. 9.de, ex. 39.ce, ex. 40.b-e;
- (•) deducerea *formei duale* pentru problema SVM cu margine “hard”: ex. 10;
- *exemplificarea* a două modalități de găsire a formei duale a problemei SVM cu margine “hard” pentru învățarea unui concept [reprezentat de un set de date de antrenament neseparabil în spațiul „inițial“ de trăsături] folosind o funcție de mapare Φ dată: prin optimizare directă (ex. 11, ex. 41), respectiv prin folosirea relațiilor de legătură cu soluția problemei primale: ex. 12;
- (P1) efectul *multiplicării valorilor atributelor* cu o constantă pozitivă asupra separatorului obținut de SVM: ex. 27.a;²⁶⁷

²⁶⁶Vedeți și Andrew Ng (Stanford), Lecture Notes, part V, section 3.

²⁶⁷Rezultatul nu se menține și în cazul C-SVM: ex. 27.b.

- vedeti proprietatile (P4) și (P6) enunțate mai jos (la secțiunea despre C-SVM), care sunt valabile și în cazul SVM [cu margine “hard”];
- (P2) efectul unui *atribut irrelevant* — în sensul că nu afectează satisfacerea *restrictiilor* de separabilitate liniară a datelor de antrenament și, în plus, nu mărește marginea de separare — asupra rezultatelor clasificatorului SVM (și respectiv C-SVM): ex. 48, ex. 56;
- *comparații* între SVM [având, eventual, diferite funcții-nucleu] și alți clasificatori: ex. 56, ex. 44, ex. 45; vedeti și ex. 12 de la capitolul *Învățare bazată pe memorare*.

SVM cu margine “soft” (C-SVM):

forma primală și forma duală

- (•) deducerea *formei duale* pentru problema SVM cu margine “soft” (C-SVM): ex. 13;
 - (P3) exprimarea / calcularea distanței geometrice de la un vectori-suport x_i pentru care $\bar{\alpha}_i = C$ la hiperplanul-margine corespunzător etichetei y_i , cu ajutorul variabilei de „destindere“ ξ_i : ex. 14.a;
 - exemplificarea noțiunilor de bază: ex. 46, ex. 47 și CMU, 2008 fall, Eric Xing, final, ex. 2.2; un *exemplu* de calculare a valorii optime pentru funcția obiectiv a problemei de optimizare C-SVM: ex. 14.b;
 - exemplificarea poziționării separatorului optimal determinat de C-SVM (pentru diferite valori ale parametrului C), în prezența unui outlier: ex. 16;
 - exemplificarea efectului pe care îl are creșterea valorii parametrului de „destindere“ C [asupra marginii și asupra exceptiilor la clasificare]: ex. 17, CMU, 2010 fall, Aarti Singh, HW3, ex. 3.2;
 - un exemplu de situație în care forma duală a problemei de optimizare C-SVM are soluție unică, dar forma sa primală nu are soluție unică: ex. 18;
 - (P4) o proprietate pentru C-SVM (dar și pentru SVM): ex. 15.
- Dacă în setul de date de antrenament două trăsături (engl., features) sunt duplicate ($x_{ij} = x_{ik}$ pentru $i = 1, \dots, m$), atunci ele vor primi ponderi identice ($\bar{w}_j = \bar{w}_k$) în soluția optimală calculată de clasificatorul [C-]SVM;
- (P5) o *margine superioară* (engl., upper bound) pentru numărul de *erori* comise la *antrenare* de către C-SVM: ex. 19;

$$\text{err}_{\text{train}}(\text{C-SVM}) \leq \frac{1}{m} \sum_i \xi_i$$

- (P6) o proprietate pentru C-SVM (dar și pentru SVM):

La CVLOO numai vectorii-suport pot fi (eventual!) clasificați eronat: ex. 20;

ășadar avem [și] o *margine superioară* pentru numărul de *erori* comise la *CVLOO* de către C-SVM:

$$\text{err}_{\text{CVLOO}}(\text{C-SVM}) \leq \frac{\#SVs}{m}$$

- (P7) chestiuni legate de *complexitatea computațională* privind clasificatorul C-SVM: ex. 54;
 - (•) deducerea *formei duale* pentru problema SVM cu margine “soft” (C-SVM) de normă \mathcal{L}_2 : ex. 49;
 - (•) o formă echivalentă a problemei de optimizare C-SVM, în care nu apar deloc restricții asupra variabilelor, dar în care se folosește funcția de pierdere / cost *hinge*: ex. 21; exemplificare / aplicare (și comparare cu regresia logistică): ex. 50;
 - (•) algoritmul SMO (Sequential Minimal Optimization): deducerea relațiilor de actualizare a variabilelor Lagrange;
- exemple* de aplicare a algoritmului SMO simplificat: ex. 23, ex. 51;

- o *comparație* asupra efectului *atributelor irelevante* (aici, în sensul că odată eliminate / adăugate, n-ar trebui să afecteze rezultatele clasificării) asupra clasificatorilor 1-NN și C-SVM: ex. 56.

SVM / C-SVM și funcțiile-nucleu — câteva proprietăți

- *exemplificarea* corespondenței dintre forma (primală sau duală) a problemei C-SVM și alegerea valorii parametrului de „destindere” C și a *funcției-nucleu* pe de o parte și alura și poziționarea separatorului optimal pe de altă parte: ex. 24, ex. 52;
- *exemplificarea* efectului pe care îl are translatarea datelor în raport cu o axă (Oy) asupra poziției separatorului optimal (în raport cu *funcția-nucleu* folosită): ex. 42;
- C-SVM: condiții suficiente asupra parametrului de „destindere” C și asupra valorilor *funcției-nucleu* pentru ca toate instanțele de antrenament să fie vectori-suport: ex. 25;
- SVM cu nucleu RBF: câteva proprietăți remarcabile
 - pentru SVM pe un set de date [separabil liniar în spațiul de trăsături] instanțe foarte depărtate de separatorul optimal pot fi vectori-suport: ex. 43;
 - (P8) pentru orice set de instanțe distințe și pentru orice etichetare a acestora, există o valoare a hiper-parametrului nucleului RBF (σ) astfel încât SVM obține la antrenare eroare 0: ex. 26. Rezultatul *nu* este valabil și pentru C-SVM;
 - (P9) pentru orice set de instanțe distințe, pentru orice etichetare a acestora și pentru *orice* valoare a hiper-parametrului nucleului RBF (σ), problema de tip SVM care impune ca toate instanțele să fie corect clasificate și la distanță $1/\|w\|$ față de separatorul optimal are soluție: ex. 58;
- avantaje și dezavantaje ale folosirii metodelor de clasificare liniară de tipul SVM, Perceptron etc. și versiunile lor kernel-izate: ex. 55;
- chestiuni recapitulative: ex. 28, ex. 27, ex. 65, ex. 57.

Alte probleme [de optimizare] de tip SVM

- SVM pentru clasificare *n*-ară (SVM multiclass): ex. 29, ex. 59;
- deducerea *formei duale* pentru problema *one-class SVM*, versiunea *Max Margin*: ex. 30 (varianta cu margine “hard”), ex. 61 (varianta cu margine “soft”, folosind ν -SVM);
- legătura dintre soluțiile problemei *one-class SVM*, versiunea *Max Margin*, cu margine “hard” și respectiv cele ale problemei SVM (cu și respectiv fără termen liber (engl., bias), tot cu margine “hard”: ex. 60;
- deducerea *formei duale* pentru problema *one-class SVM*, versiunea *minimum enclosing ball* (MEB): ex. 31 (varianta cu margine “hard”), ex. 62 (varianta cu margine “soft”, folosind ν -SVM);
- o condiție suficientă pentru ca variantele cu margine “hard” pentru cele două tipuri de probleme de optimizare *one-class SVM*, și anume *Max Margin* și *minimum enclosing ball* (MEB), în forma kernel-izată, să fie echivalente: ex. 31;
- deducerea *formei duale* pentru problema ν -SVM: ex. 32;
- deducerea *formei duale* pentru problema SVR (*Support Vector Regression*), folosind funcție de cost / pierdere ε -senzitivă: ex. 33 (cu margine “hard”), ex. 64 (cu margine “soft” și (echivalent) cu funcție de cost ε -senzitivă);
- exemplificare / aplicare: ex. 63.

5.1 Probleme rezolvate

5.1.1 SVM cu margine “hard”

1. (O proprietate a vectorului w care apare în ecuația unui hiperplan; Distanța de la un hiperplan din \mathbb{R}^d la un punct oarecare din același spațiu: deducerea formulei)
prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Ziv Bar-Joseph, HW4, pr. 1.1

Fie un punct oarecare x_0 din \mathbb{R}^d și un hiperplan de ecuație $w \cdot x + b = 0$, unde w , x și b sunt de asemenea din \mathbb{R}^d , iar simbolul \cdot indică produsul scalar al vectorilor din \mathbb{R}^d .

a. Demonstrați că vectorul w este ortogonal pe acest hiperplan.

(*Sugestie:* Fie x_1 și x_2 două puncte situate pe acest hiperplan. Ce puteți spune despre valoarea produsului scalar $w \cdot (x_1 - x_2)$?)

b. Demonstrați că distanța [măsurată pe perpendiculara] de la x_0 la hiperplanul $w \cdot x + b = 0$ este

$$\frac{|w \cdot x_0 + b|}{\|w\|},$$

unde simbolul $\|\cdot\|$ desemnează norma euclidiană.²⁶⁸

Răspuns:

a. Fie H hiperplanul de ecuație $w \cdot x + b = 0$ (în scriere vectorială, $w^\top x + b = 0$, unde am considerat w și x vectori-colonă din \mathbb{R}^d). Considerăm x_1 și x_2 două puncte oarecare distincte situate pe hiperplanul H . Așadar, vom avea

$$w \cdot x_1 + b = 0 \text{ și } w \cdot x_2 + b = 0,$$

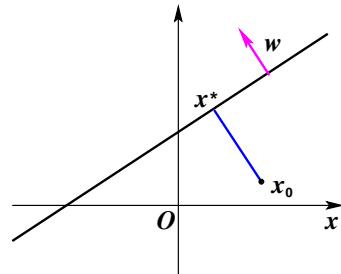
de unde, prin scădere rezultă $w \cdot (x_1 - x_2) = 0$, ceea ce înseamnă că vectorii w și $x_1 - x_2$ sunt perpendiculari. Întrucât vectorul $x_1 - x_2$ are direcția hiperplanului H , rezultă că w este perpendicular pe H .

b. Notăm cu x^* „picioară“ perpendicularăi coborâte din punctul x_0 pe hiperplanul H . Prin urmare, vectorul $x^* - x_0$ are aceeași direcție cu vectorul w . Stim că pentru oricare doi vectori paraleli u și u' este satisfăcută proprietatea următoare: există o constantă $\lambda \in \mathbb{R}$ astfel încât $u' = \lambda u$. Așadar, există $t \in \mathbb{R}$ cu proprietatea $x^* - x_0 = tw$. Altfel spus,

$$x^* = x_0 + tw. \quad (104)$$

Stim de asemenea că punctul x^* aparține hiperplanului H . Prin urmare,

$$w \cdot x^* + b = 0. \quad (105)$$



²⁶⁸ $\|w\|^2 = w \cdot w$ sau $\|w\|^2 = w^\top w$, unde simbolul \top indică operația de transpunere de vectori / matrice. Precizăm că $\|\cdot\|$ se scrie pur și simplu $\|\cdot\|$ atunci când în contextul respectiv nu există posibilitatea de a se face confuzie cu vreo altă normă).

Din relațiile (104) și (105) rezultă

$$\begin{aligned} w \cdot (x_0 + tw) + b = 0 &\Leftrightarrow w \cdot x_0 + tw^2 + b = 0 \Leftrightarrow w \cdot x_0 + t\|w\|^2 + b = 0 \Leftrightarrow \\ t\|w\|^2 &= -(w \cdot x_0 + b) \Leftrightarrow t = -\frac{w \cdot x_0 + b}{\|w\|^2}. \end{aligned}$$

În concluzie, distanța de la x^* la hiperplanul H este

$$\|x^* - x_0\| = \|tw\| = |t| \|w\| = \frac{|w \cdot x_0 + b|}{\|w\|^2} \|w\| = \frac{|w \cdot x_0 + b|}{\|w\|}.$$

2.

(Exercițiu teoretic: deducerea formei primale a problemei de optimizare SVM (cu margine "hard", pornind de la principiul maximizării marginii geometrice)
prelucrare de Liviu Ciortuz, după CMU, 2016 fall, N. Balcan. M. Gormley, HW4, pr. 1.1 CMU, 2006 spring, Carlos Guestrin, midterm, pr. 1.4

În acest exercițiu veți vedea cum anume se deduce *problema de optimizare SVM cu margine "hard"* pornind de la *principiul marginii [geometrice] maxime*.

Presupunem că avem setul de date de antrenament $D = (\mathbf{X}, \mathbf{y})$, unde $\mathbf{X} \in \mathbb{R}^{d \times m}$, iar $y \in \{-1, 1\}^m$. Coloana i a matricei \mathbf{X} este x_i , vectorul de trăsături al celui de-al i -lea exemplu de antrenament, iar y_i este eticheta acestui exemplu.

Pentru clasificare, vom folosi o funcție liniară, de forma

$$f(x) = w \cdot x, \text{ unde } w \in \mathbb{R}^d,$$

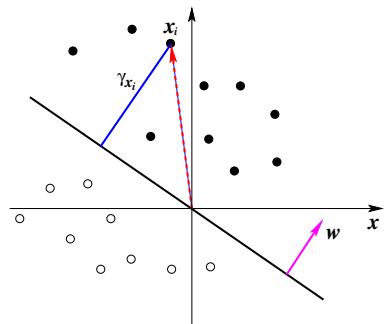
iar operatorul \cdot reprezintă produsul scalar al vectorilor.

Observăm că, pentru simplitate, în expresia lui f nu a fost adăugat termenul liber (notat îndeobște cu w_0 sau cu b , ultimul provenind de la termenul englezesc *bias*). Așadar, vom trata aici doar cazul separabilității liniare prin originea sistemului de coordonate.²⁶⁹

O instantă oarecare x va fi clasificată în clasa 1 dacă $f(x) > 0$, respectiv în clasa -1 în caz contrar. Hiperplanul de ecuație $f(x) = 0$ va funcționa ca *separator* al instanțelor de antrenament, sau *graniță de decizie* (engl., decision boundary). Așadar, f este funcția pe care vrem să-o învățăm (adică funcția "target").

Presupunem că datele de antrenament sunt liniar separabile. Atunci pentru orice exemplu de antrenament (x, y) vom avea $yf(x) > 0$. În consecință, distanța geometrică de la x la granița de decizie — distanță despre care, cf. pr. 1, știm că este egală cu $\frac{|f(x)|}{\|w\|}$ — poate fi scrisă ca

$$\gamma_x = \frac{yf(x)}{\|w\|}.$$



²⁶⁹Cazul separabilității liniare cu termenul liber w_0 luând o valoare oarecare se tratează extinzând în mod corespunzător demonstrația de la punctul a.

În contextul învățării automate, această distanță se numește *marginea geometrică* sau, pe scurt, *marginea* [dintre instanța x și separatorul determinat de ecuația $f(x) = 0$].

În vederea reducerii *riscului de clasificare eronată* la faza de *generalizare*, este natural să determinăm f (deci, de fapt, vectorul de ponderi w) astfel încât minimul acestor margini γ_x să fie cât mai mare. Altfel spus, vrem să maximizăm distanța [de la *separatorul optimal*] până la instanțele de antrenament cele mai apropiate. Așadar, *funcția noastră obiectiv* va fi:

$$\max_w \min_{i=1,\dots,m} \frac{y_i f(x_i)}{\|w\|}. \quad (106)$$

a. Arătați că problema de optimizare fără restricții (106) este *echivalentă* cu următoarea problemă de optimizare convexă cu restricții liniare (pe care o vom numi *problema SVM*):²⁷⁰

$$\min_w \frac{1}{2} \|w\|^2 \quad (107)$$

a. i. $y_i(w \cdot x_i) \geq 1$, pentru $i = 1, \dots, m$.

Sugestie: Are oare vreun efect scalarea vectorului de ponderi $w \rightarrow kw$, cu $k \in \mathbb{R}$, $k > 0$?

b. Presupunând că datele de antrenament sunt liniar separabile, precizați ce se va întâmpla cu separatorul obținut prin rezolvarea problemei SVM dacă se renunță la unul dintre exemplele de antrenament: se va deplasa oare separatorul *însprij* exemplul / punctul respectiv, se va retrage *dinspre* punctul respectiv, ori poziția lui va rămâne neschimbată? Justificați.

Răspuns:

a. Vom demonstra echivalența dintre problemele de optimizare date în enunț (106) și (107) construind o succesiune de câteva probleme de optimizare echivalente, obținute prin aplicarea unor transformări succesive, pornind de la problema (106) și ajungând în final la problema (107).

Fie, așadar, următoarele probleme de optimizare:

$$\max_w \min_{i=1,\dots,m} \frac{y_i w \cdot x_i}{\|w\|} \quad (108)$$

a. i. $y_i(w \cdot x_i) > 0$, pentru $i = 1, \dots, m$.

$$\max_w \frac{1}{\|w\|} \min_{i=1,\dots,m} y_i w \cdot x_i \quad (109)$$

a. i. $y_i(w \cdot x_i) > 0$, pentru $i = 1, \dots, m$.

$$\max_w \frac{1}{\|w\|} \min_{i=1,\dots,m} y_i w \cdot x_i \quad (110)$$

a. i. $y_i(w \cdot x_i) \geq 1$, pentru $i = 1, \dots, m$.

²⁷⁰LC: Este vorba de o echivalență *slabă* (engl., soft): optimul — adică, valoarea funcției obiectiv — pentru cele două probleme este același, iar orice soluție a primei probleme poate fi pusă în corespondență cu o soluție a celei de-a doua probleme și invers. (Echivalența ar fi *tare* (engl., hard) dacă valoarea funcției obiectiv pentru cele două probleme ar fi aceeași, iar orice soluție optimă a primei probleme ar fi soluție optimă și pentru cea de-a două problemă și invers.)

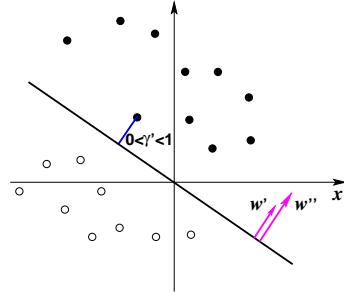
$$\max_w \frac{1}{\|w\|} \quad (111)$$

a. i. $y_i(w \cdot x_i) \geq 1$, pentru $i = 1, \dots, m$.

- Se observă că problema de optimizare (108) diferă de problema inițială (106) prin introducerea unor restricții liniare. Mai mult, aceste restricții corespund hiperplanelor care sunt separatori liniari pentru mulțimea de antrenament dată (care este, conform enunțului, separabilă liniar). Așadar, aceste restricții nu schimbă cu nimic soluția problemei date inițial.²⁷¹
- Problema de optimizare (109) a fost obținută din problema (108) modificând doar funcția obiectiv: $\frac{1}{\|w\|}$ a fost scos în fața operatorului $\min_{i=1,\dots,m}$ fiindcă acesta nu depinde de w . (Observați că expresia care urmează acestui operator depinde de w , dar acolo acest w este considerat fixat și variază doar i .)
- Problema de optimizare (110) a fost obținută din problema (109) schimbând în restricții relația > 0 cu ≥ 1 . În sine, această modificare restrâng mulțimea acelor valori ale lui w peste care se aplică operatorul \max_w .

Să considerăm însă un w' arbitrar care este eliminat la trecerea de la forma (109) la forma (110). Notăm $\gamma' = \min_{i=1,\dots,m} y_i \underbrace{w' \cdot x_i}_{f(x_i)}$; stim că $\gamma' \in (0, 1)$. Rezultă că

$$\frac{y_i w' \cdot x_i}{\gamma'} = y_i \frac{w'}{\gamma'} \cdot x_i \geq 1 \text{ pentru } 1, \dots, m.$$



Observați că egalitatea chiar se produce, și anume atunci când indexul i ia valoarea pentru care se atinge minimul expresiei $y_i w' \cdot x_i$.

Așadar, $w'' \stackrel{\text{not.}}{=} \frac{w'}{\gamma'}$ satisfac restricțiile problemei (110). În plus,

$$\frac{1}{\|w''\|} = \frac{1}{\|w'\|} \gamma' \text{ și, echivalent, } \frac{1}{\|w''\|} \underbrace{\min_{i=1,\dots,m} y_i w'' \cdot x_i}_{1} = \frac{1}{\|w'\|} \underbrace{\min_{i=1,\dots,m} y_i w' \cdot x_i}_{\gamma'}.$$

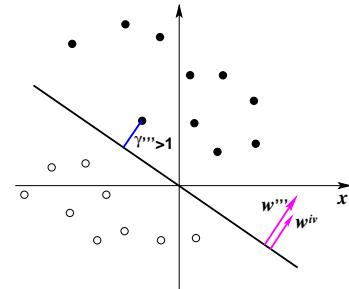
Prin urmare, putem spune că rolul lui w' în relația (109) este jucat de w'' în relația (110). În consecință, la trecerea de la o formă la cealaltă optimul rămâne același.²⁷²

- Problema de optimizare (111) a fost obținută din problema (110) renunțând în funcția obiectiv la componenta $\min_{i=1,\dots,m} y_i w \cdot x_i$.

²⁷¹ Pentru un separator liniar oarecare al mulțimii de antrenament, expresia $\min_{i=1,\dots,m} \frac{y_i f(x_i)}{\|w\|}$ are o valoare strict pozitivă. Pentru hiperplanele care nu separă mulțimea de antrenament, expresia $\min_{i=1,\dots,m} \frac{y_i f(x_i)}{\|w\|}$ are valoare negativă sau 0.

²⁷² Din punct de vedere geometric, trecerea de la (109) la (110) poate fi asociată cu înmulțirea cu o constantă pozitivă (chiar supra-unitară) a ecuațiilor acelor separatori liniari [ai mulțimii de antrenament] pentru care minimul distanțelor geometrice până la instanțele de antrenament este strict mai mic decât 1, astfel încât în urma înmulțirii această distanță minimă să devină 1.

Aceasta revine de fapt la a renunța la acele valori ale lui w pentru care $\min_{i=1,\dots,m} y_i w \cdot x_i$ este strict mai mare decât 1. (Am văzut mai sus că există valori ale lui w (și anume, w'') pentru care minimul respectiv este chiar 1.) Să considerăm un w''' cu proprietatea $\gamma''' \stackrel{\text{not.}}{=} \min_{i=1,\dots,m} y_i \underbrace{w''' \cdot x_i}_{f(x_i)} > 1$.



Rezultă că

$$\min_{i=1,\dots,m} y_i \frac{w'''}{\gamma'''} \cdot x_i = 1$$

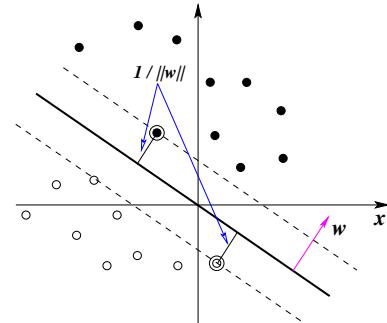
și, notând $w^{iv} = \frac{w'''}{\gamma'''}$, vom avea:

$$\frac{1}{\|w^{iv}\|} \underbrace{\min_{i=1,\dots,m} y_i w^{iv} \cdot x_i}_{1} = \frac{1}{\|w^{iv}\|} = \frac{\gamma'''}{\|w'''\|} = \frac{1}{\|w'''\|} \underbrace{\min_{i=1,\dots,m} y_i w''' \cdot x_i}_{\gamma'''}$$

Prin urmare, [ca și mai sus,] putem spune că rolul lui w''' în relația (110) este jucat de w^{iv} în relația (111). În consecință, [ca și mai sus,] nu se modifică valoarea optimă a funcției obiectiv dacă se renunță [șii] la acești w''' .²⁷³

- Se constată imediat că problema (111) este echivalentă cu problema de optimizare SVM care a fost dată în enunț (107). Forma problemei de optimizare SVM ne spune că separatorul optimal se alege dintre acei separatori $w \cdot x$ ai mulțimii de antrenament care au exact valoarea 1 pentru $\min_{i=1,\dots,m} y_i w \cdot x_i$.

Concret, alegerea se face maximizând $1/\|w\|$, adică distanța geometrică de la separator(i) până la cele mai apropiate instanțe de antrenament. Intuitiv, maximizarea aceasta va implica faptul că distanțele de la separatorul optimal până la *toate* instanțele de antrenament cele mai apropiate (adică, *vectorii-suport*, cei pozitivi și respectiv cei negativi) vor fi $1/\|w\|$. (În figura alăturată, în care am pus în evidență acest fapt, vectorii-suport sunt încercuiți.)



Altfel spus, valoarea funcției $w \cdot x$ pentru w optim va fi +1 pentru vectorii-suport pozitivi și -1 pentru vectorii-suport negativi.

- b. La eliminarea unei instanțe de antrenament x_j , marginea $\max_w \min_i \frac{y_i f(x_i)}{\|w\|}$ nu poate să scadă; poate doar să crească sau, eventual, să rămână neschimbătă. Într-adevăr, pentru

²⁷³Din punct de vedere geometric, trecerea de la (110) la (111) corespunde înmulțirii cu o constantă pozitivă (sub-unitară) a ecuațiilor acelor separatori liniari [ai mulțimii de antrenament] pentru care minimul distanțelor geometrice până la instanțele de antrenament este strict *mai mare* decât 1, astfel încât în urma înmulțirii această distanță minimă să devină 1.

un w oarecare (fixat), cantitatea $\min_i \frac{y_i f(x_i)}{\|w\|}$ fie crește fie rămâne aceeași la eliminarea unui x_j , fiindcă acum operatorul min este aplicat pe o mulțime mai puțin amplă. În consecință, lăsându-l apoi pe w să varieze, cantitatea $\max_w \min_i \frac{y_i f(x_i)}{\|w\|}$ crește și ea sau cel puțin rămâne aceeași.

Tinând cont de rezultatul de echivalentă de la punctul a , rezultă că la eliminarea unei instanțe de antrenament x_j marginea de separare optimală $(1/\|w\|)$ crește sau rămâne, eventual, neschimbătă. Așadar, poziția separatorului optimal (i) va fi deplasată înspre instanță eliminată sau (ii) va rămâne neschimbătă. Cazul (i) este posibil să apară doar dacă x_j este vector-suport, adică se află la distanță minimală față de separatorul optimal.²⁷⁴ Cazul (ii) se produce atunci când x_j este vector-suport și există mai mulți vectori-suport care au etichete identice cu eticheta lui x_j sau atunci când x_j nu este vector-suport.

Observație importantă:

Până la problema 10 — unde vom prezenta deducerea formei duale a problemei de optimizare SVM —, vom lucra cu o *definiție în sens larg* (geometric) pentru noțiunea de *vector-suport*: spunem că un vector-suport este o instanță x_i care se află la distanță geometrică de $1/\|w\|$ față de separatorul optimal (w, w_0) . Odată cu introducerea formei duale a problemei de optimizare SVM, vom folosi pentru noțiunea de vector-suport *definiția clasică* (în sens (analitic) restrâns în raport cu cel precedent): o instanță x_i este vector-suport dacă în soluția optimă a problemei duale, variabila / multiplicatorul Lagrange $\bar{\alpha}_i$ are valoare nenulă (deci strict pozitivă, pentru că toți multiplicatorii Lagrange sunt acolo nenegativi). În această situație, unele instanțe situate la distanță (geometrică) de $1/\|w\|$ față de separatorul optimal (w, w_0) pot să nu fie vectori-suport. Pentru SVM cu margine “soft” (C-SVM), va fi util să vedetă *Rezumatul* de la rezolvarea punctului e de la problema 13. Se va constata acolo că este posibil să avem vectori-suport aflați la distanță (geometrică) mai mică de $1/\|w\|$ față de separatorul optimal (w, w_0) ; aceștia sunt așa-numiții (în engl.) *non-bound support vectors*.

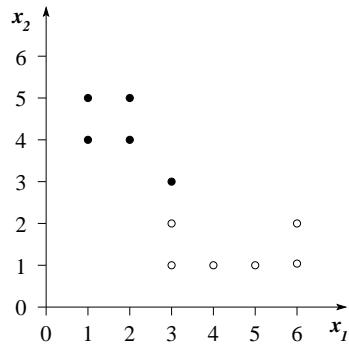
3.

(Separabilitate în \mathbb{R}^2 ; SVM liniară, forma primală)*CMU, 2003 fall, T. Mitchell, A. Moore, final exam, pr. 5*

Desenul de mai jos prezintă un set de date cu două atrbute de intrare x_1 și x_2 , și un atrbut de ieșire y , ale cărui valori sunt reprezentate prin culoarea punctului (alb/- și negru/+).

²⁷⁴Totuși, nu este *obligatoriu* ca în această situație poziția separatorului optimal să se modifice. Vedetă cazul (ii).

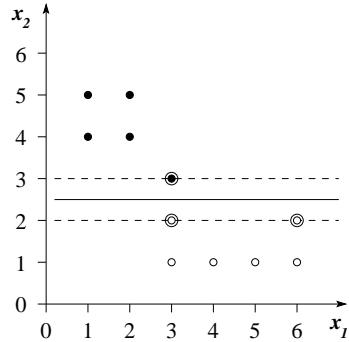
- a. Presupunând că aceste date sunt corecte (adică fără ‘zgomote’) și că folosim o SVM liniară, trasați pe acest desen trei drepte care să indice linia de *separare optimală* și cele două „margini“ (paralelele la separatorul optimal care trec prin vectorii-suport). Încercuți vectorii-suport.
- b. Plecând de la forma primală a unei SVM liniare (având regula de decizie $y = \text{sign}(w \cdot x + w_0)$), calculați valorile corespunzătoare parametrilor w și w_0 care determină hiperplanul de separare optimală de la punctul a .



Răspuns:

- a. În figura alăturată, linia continuă reprezintă dreapta de separare, iar cele două linii punctate reprezintă marginile maximale. Vectorii-suport sunt punctele $(3, 3)$, $(3, 2)$ și $(6, 2)$.
- b. În forma primală a problemei SVM liniare se cere maximizarea valorii $1/2 \cdot \|w\|^2$, simultan cu satisfacerea restricțiilor $y_i(w \cdot x + w_0) \geq 1$, unde $w \stackrel{\text{not.}}{=} (w_1, w_2)$ și $x \stackrel{\text{not.}}{=} (x_1, x_2)$. Înținând cont de faptul că aceste inegalități sunt îndeplinite cu egalitate [doar] în cazul vectorilor-suport, obținem următorul sistem de ecuații:

$$\begin{cases} -(w_1, w_2) \cdot (3, 2) + w_0 = 1 \\ -(w_1, w_2) \cdot (6, 2) + w_0 = 1 \\ (w_1, w_2) \cdot (3, 3) + w_0 = 1 \end{cases} \Rightarrow \begin{cases} 3w_1 + 2w_2 + w_0 = -1 \\ 6w_1 + 2w_2 + w_0 = -1 \\ 3w_1 + 3w_2 + w_0 = 1 \end{cases} \Rightarrow \begin{cases} w_1 = 0 \\ w_2 = 2 \\ w_0 = -5 \end{cases}$$



Am obținut deci $w = (w_1, w_2) = (0, 2)$ și $w_0 = -5$, ceea ce conduce la ecuația $2x_2 - 5 = 0 \Leftrightarrow x_2 - \frac{5}{2} = 0$ pentru separatorul optimal.

Alternativ, valorile parametrilor w_0 , w_1 și w_2 pot fi determinate folosind cunoștințe de geometrie analitică. Ecuația dreptei din figura de mai sus este

$$x_2 = \frac{5}{2} \Leftrightarrow x_2 - \frac{5}{2} = 0$$

sau, mai general, $\alpha \left(x_2 - \frac{5}{2} \right) = 0$, cu $\alpha \in \mathbb{R}$, $\alpha \neq 0$. Valoarea lui α corespunzătoare separatorului optimal se obține impunând restricțiile specifice vectorilor-suport:

$$\alpha \left(x_2 - \frac{5}{2} \right) = 1 \text{ pentru } x_2 = 3 \text{ sau / și, respectiv, } \alpha \left(x_2 - \frac{5}{2} \right) = -1 \text{ pentru } x_2 = 2.$$

Rezultă că $\alpha = 2$, deci ecuația separatorului optimal este $2\left(x_2 - \frac{5}{2}\right) = 0 \Leftrightarrow 0 \cdot x_1 + 2 \cdot x_2 - 5 = 0$, de unde, identificând pe componente, rezultă: $w_1 = 0$, $w_2 = 2$ și $w_0 = -5$.

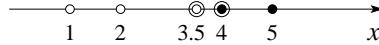
4. (Separabilitate în \mathbb{R} ; SVM liniară, forma primală; calculul erorilor la antrenare și CVLOO)
CMU, 2001 fall, Andrew Moore, final exam, pr. 11

Fie următorul set de date:

x_i	1	2	3.5	4	5
y_i	-1	-1	-1	1	1

- a. Care sunt valorile parametrilor w și w_0 învățate de o SVM liniară pornind de la aceste date? (Vă readucem aminte că regula de decizie a unei astfel de SVM este $y = \text{sign}(w \cdot x + w_0)$.)
- b. Care sunt vectorii-suport?
- c. Care este eroarea de clasificare pe mulțimea de date de antrenament de mai sus?
- d. Care este eroarea de clasificare pe același set de date, folosind metoda de cross-validation "Leave-One-Out"? (Veți exprima eroarea sub forma procentajului de instanțe clasificate eronat.)

Răspuns:



În mod *direct*, reprezentarea grafică ne arată că separatorul optimal va fi plasat în dreptul punctului 3.75, folosind vectorii-suport $x_3 = 3.5$ și $x_4 = 4$, și conducând la eroare de antrenare 0 (deoarece datele de antrenament sunt liniar separabile). Este imediat că instanțele $x_3 = 3.5$ și $x_4 = 4$ vor fi clasificate eronat la CVLOO.

- a. *Analitic*, din forma primală a problemei SVM rezultă că trebuie satisfăcute următoarele restricții:

$$\begin{cases} -1(w \cdot 1 + w_0) \geq 1 \\ -1(w \cdot 2 + w_0) \geq 1 \\ -1(w \cdot 3.5 + w_0) \geq 1 \\ 1(w \cdot 4 + w_0) \geq 1 \\ 1(w \cdot 5 + w_0) \geq 1 \end{cases} \Leftrightarrow \begin{cases} w + w_0 \leq -1 \\ 2w + w_0 \leq -1 \\ 3.5w + w_0 \leq -1 \\ 4w + w_0 \geq 1 \\ 5w + w_0 \geq 1 \end{cases}$$

Impunând ca restricțiile corespunzătoare vectorilor-suport ($x_3 = 3.5$ și $x_4 = 4$) să fie satisfăcute cu egalitate — ceea ce va conduce în mod implicit și la satisfacerea celorlalte inegalități din sistem —, rezultă:

$$\begin{cases} 3.5w + w_0 = -1 \\ 4w + w_0 = 1 \end{cases} \Rightarrow \begin{cases} -3.5w - w_0 = 1 \\ 4w + w_0 = 1 \end{cases} \Rightarrow \begin{cases} 0.5w = 2 \\ 4w + w_0 = 1 \end{cases} \Rightarrow \begin{cases} w = 4 \\ w_0 = -15 \end{cases}$$

Am obținut deci soluția $w = 4$, $w_0 = -15$.

- b. După cum am menționat mai sus, vectorii-suport sunt $x_3 = 3.5$ și $x_4 = 4$.

Alternativ, însă tot *analitic*, punctele a și b de mai sus se pot rezolva astfel:

Se observă în mod direct că inegalitatea $x \geq 3.75$ este satisfăcută pentru — deci, funcția $\text{sign}(x - 3.75)$ clasifică corect — toate instanțele, atât cele pozitive cât și cele negative. Mai mult, instanțele $x_3 = 3.5$ (negativă) și $x_4 = 4$ (pozitivă) sunt cele mai apropiate față de „pragul” $x = 3.75$ și sunt situate la distanțe egale față de acest prag, deci sunt vectorii-suport. Ecuatia $x = 3.75$ se scrie mai general $\alpha(x = 3.75)$, cu $\alpha \neq 0$. Impunând condiția $\alpha(x - 3.75) = -1$ pentru $x = 3.5$ și respectiv $\alpha(x - 3.75) = 1$ pentru $x = 4$, rezultă $\alpha = 4$ și deci, prin identificare pe componente, $w = 4$ și $w_0 = -15$.

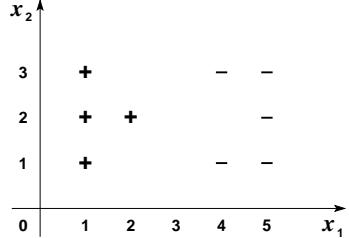
c. Judecând *calitativ*, eroarea de clasificare pe mulțimea datelor de antrenament este 0, fiindcă datele sunt liniar separabile. *Analitic*, am arătat deja că $\text{sign}(w \cdot x_i + w_0) = y_i$, pentru $y \in \{1, 2, 3, 4, 5\}$.

d. La cross-validation cu metoda “Leave-One-Out”, instanțele 3.5 și 4 vor fi clasificate eronat pentru că pragul de separare optimală devine $x = 3$ și respectiv $x = 4.5$. Eroarea este deci de 40%. De notat, ca un *principiu general*: atunci când se lucrează cu SVM, erorile la cross-validation se pot produce doar la eliminarea vectorilor-suport.

5.

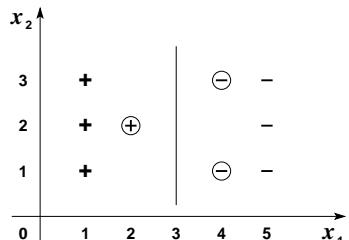
(SVM liniară: exemplificare pe date din \mathbb{R}^2 ;
modificarea eventuală a poziției separatorului optimal
la eliminarea unei instanțe de antrenament)

În imaginea alăturată, încercuiți fiecare punct care are proprietatea că odată ce este eliminat din setul de date de antrenament, la re-antrenarea mașinii cu vectorii-suport vom obține un alt separator optimal decât cel rezultat în cazul antrenării pe întreaga mulțime de date. Justificați răspunsul.



Răspuns:

În figura alăturată am reprezentat *grafic* separatorul liniar învățat de SVM din datele de antrenament și am încercuit vectorii-suport. După cum știm deja din rezolvarea problemei 4.d, eliminarea datelor care nu sunt vectorii-suport nu modifică rezultatul procesului de învățare. Așadar, rămâne de verificat comportamentul sistemului la eliminarea către unui vector suport.



Analitic, putem vedea că separatorul învățat este unic determinat de sistemul:

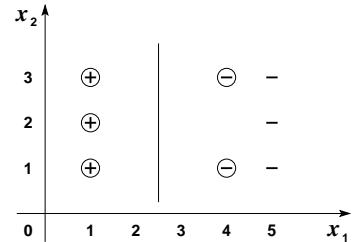
$$\begin{cases} (w_1, w_2) \cdot (2, 2) + w_0 = 1 \\ -1((w_1, w_2) \cdot (4, 1) + w_0) = 1 \\ -1((w_1, w_2) \cdot (4, 3) + w_0) = 1 \end{cases} \Rightarrow \begin{cases} 2w_1 + 2w_2 + w_0 = 1 \\ 4w_1 + w_2 + w_0 = -1 \\ 4w_1 + 3w_2 + w_0 = -1 \end{cases} \Rightarrow \begin{cases} w_1 = -1 \\ w_2 = 0 \\ w_0 = 3 \end{cases}$$

În consecință, ecuația separatorului optimal este $-x_1 + 3 = 0 \Leftrightarrow x_1 = 3$, iar marginea geometrică (distanța de la vectorii-suport până la hiperplanul de separare) este $d = \frac{1}{\|w\|} = \frac{1}{\sqrt{(-1)^2 + 0^2}} = 1$, unde după cum știm, $w = (w_1, w_2)$.

La eliminarea punctului (2,2), hiperplanul de separare optimal este altul, vectorii-suport devenind (1,1), (1,2), (1,3), (4,1) și (4,3).

Noile valori ale parametrilor (noteate w'_0, w'_1, w'_2) se calculează rezolvând sistemul:

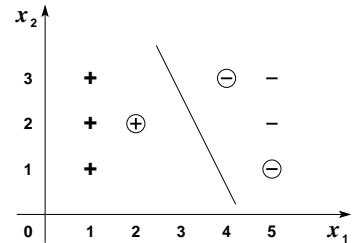
$$\begin{cases} w'_1 + w'_2 + w'_0 = 1 \\ w'_1 + 2w'_2 + w'_0 = 1 \\ w'_1 + 3w'_2 + w'_0 = 1 \\ 4w'_1 + w'_2 + w'_0 = -1 \\ 4w'_1 + 3w'_2 + w'_0 = -1 \end{cases} \Rightarrow \begin{cases} w'_1 = -\frac{2}{3} \\ w'_2 = 0 \\ w'_0 = \frac{5}{3} \end{cases}$$



Așadar, ecuația separatorului optimal este acum $-\frac{2}{3}x_1 + \frac{5}{3} = 0 \Leftrightarrow x_1 = \frac{5}{2}$, iar marginea geometrică este $d' = \frac{1}{\|w'\|} = \frac{1}{\sqrt{(-2/3)^2 + 0^2}} = \frac{3}{2}$.

La eliminarea punctului (4,1), vectorii-suport devin (2,2), (5,1) și (4,3). Determinarea noilor valori ale parametrilor se face prin rezolvarea sistemului:

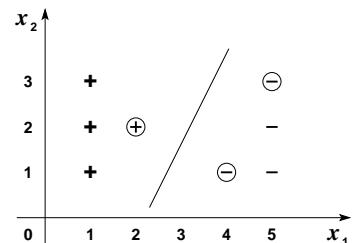
$$\begin{cases} 2w''_1 + 2w''_2 + w''_0 = 1 \\ 5w''_1 + w''_2 + w''_0 = -1 \\ 4w''_1 + 3w''_2 + w''_0 = -1 \end{cases} \Rightarrow \begin{cases} w''_1 = -\frac{4}{5} \\ w''_2 = -\frac{2}{5} \\ w''_0 = \frac{17}{5} \end{cases}$$



În consecință, în acest caz ecuația separatorului optimal este $-\frac{4}{5}x_1 - \frac{2}{5}x_2 + \frac{17}{5} = 0 \Leftrightarrow 4x_1 + 2x_2 - 17 = 0 \Leftrightarrow x_2 = -2x_1 + \frac{17}{2}$, iar marginea geometrică până la hiperplan este $d'' = \frac{1}{\|w''\|} = \frac{1}{\sqrt{(-4/5)^2 + (-2/5)^2}} = \frac{5}{2\sqrt{5}} = \frac{\sqrt{5}}{2}$.

La eliminarea punctului (4,3), vectorii-suport devin (2,2), (4,1) și (5,3). Vom obține următoarele valori ale parametrilor:

$$\begin{cases} 2w'''_1 + 2w'''_2 + w'''_0 = 1 \\ 4w'''_1 + w'''_2 + w'''_0 = -1 \\ 5w'''_1 + 3w'''_2 + w'''_0 = -1 \end{cases} \Rightarrow \begin{cases} w'''_1 = -\frac{4}{5} \\ w'''_2 = \frac{2}{5} \\ w'''_0 = \frac{9}{5} \end{cases}$$



Prin urmare, aici ecuația separatorului optimal este $-\frac{4}{5}x_1 + \frac{2}{5}x_2 + \frac{9}{5} = 0 \Leftrightarrow -4x_1 + 2x_2 + 9 = 0 \Leftrightarrow x_2 = 2x_1 - \frac{9}{2}$, iar marginea geometrică este $d''' = \frac{1}{\|w'''\|} = \frac{1}{\sqrt{(-4/5)^2 + (2/5)^2}} =$

$$\frac{5}{2\sqrt{5}} = \frac{\sqrt{5}}{2}.$$

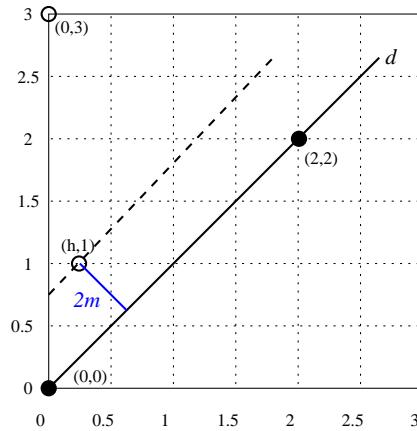
6.

(Separabilitate liniară în \mathbb{R}^2 : tratare în raport cu un parametru dat;Separabilitate neliniară în \mathbb{R} : folosirea unei funcții simple pentru maparea atributelor)*CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 4*

Presupunem că avem doar patru exemple de antrenament în spațiul euclidian bidimensional, și anume: $x_1 = (0, 0)$, $x_2 = (2, 2)$ sunt exemple pozitive, iar $x_3 = (h, 1)$, $x_4 = (0, 3)$ sunt exemple negative, h fiind un parametru cu proprietatea $0 \leq h \leq 3$.

- Cât de mare poate fi valoarea lui $h \geq 0$ cu condiția ca punctele de antrenament să rămână liniar separabile?
- Se schimbă direcția²⁷⁵ separatorului optimal ca funcție de h atunci când punctele sunt separabile? (Da/Nu; justificare.)
- Cât este *marginea* [geometrică] corespunzătoare separatorului optimal (adică, distanța de la separator până la vectorii-suport) ca funcție de h ?
- Presupunem că putem observa doar a două componente a instanțelor de antrenament. Fără cealaltă componentă, datele de antrenament etichetate se reduc la $(0, +)$, $(1, -)$, $(2, +)$ și $(3, -)$. Care este gradul minim p al unei funcții polinomiale care, aplicată acestor date, ne permite să le clasificăm corect?

Răspuns:



- Dreapta d , care este determinată de instanțele pozitive $x_1 = (0, 0)$ și $x_2 = (2, 2)$ are ecuația $y = x$. Pentru ca punctele de antrenament să rămână liniar separabile trebuie ca punctul x_3 să fie situat de aceeași parte a dreptei d ca instanța negativă $x_4 = (0, 3)$. Această condiție se traduce analitic prin inegalitatea $h < 1$.

- Hiperplanul de separare optimală va fi o dreaptă paralelă cu dreapta d (de ecuație $y = x$), situată la jumătatea distanței dintre aceasta și punctul $x_3 = (h, 1)$. Dreapta d este determinată de vectorii-suport $x_1 = (0, 0)$ și $x_2 = (2, 2)$. Panta dreptei d este 1 și nu

²⁷⁵ Direcția unei drepte din planul euclidian este dată de *panta ei*, adică tangenta unghiului format de dreaptă cu axa Ox . O definiție alternativă este următoarea: direcția unei drepte este reprezentată de mulțimea tuturor dreptelor paralele cu ea.

deinde de valoarea parametrului h . În concluzie, direcția separatorului optimal nu se schimbă în funcție de parametrul h , atât timp cât instanțele rămân separabile.

c. Marginea separatorului optimal este $m = \frac{1}{2} \text{dist}(x_3, d)$. Dreapta d are ecuația $-x + y = 0$, iar punctul x_3 este $(h, 1)$. Înținând cont de formula care ne dă distanța de la un punct oarecare la o dreaptă de ecuație cunoscută,²⁷⁶ rezultă că

$$m = \frac{1}{2} \cdot \frac{|(-1)h + 1|}{\sqrt{(-1)^2 + 1^2}} = \frac{1}{2} \cdot \frac{|1 - h|}{\sqrt{2}}$$

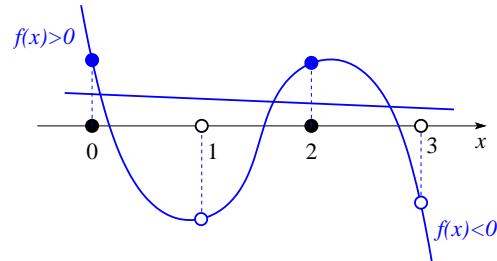
Ca urmare a combinării restricției $h \in [0, 3]$ din enunț cu restricția $h < 1$ rezultată la punctul a , vom avea $h \in [0, 1)$, deci expresia lui m de mai sus devine:

$$m = \frac{1-h}{2\sqrt{2}} = \frac{(1-h)\sqrt{2}}{4}$$

d. Punctele de antrenare $(0, +)$, $(1, -)$, $(2, +)$ și $(3, -)$ se reprezintă pe axa reală după cum urmează:



Pentru a clasifica corect aceste puncte este nevoie de o funcție polinomială de forma reprezentată în figura alăturată. Se observă că „proiecțiile“ instanțelor de antrenament pe curba polinomială sunt separabile liniar. Așadar, gradul minim p al unei funcții polinomiale care ne permite să clasificăm corect aceste puncte este 3. De pildă, putem considera funcția $-(x - 0.5)(x - 1.5)(x - 2.5)$.



7.

(O formă echivalentă cu forma primală a problemei de optimizare SVM)

University of Utah, 2008 spring, Hal Daumé III, HW1C, pr. 3

Formularea standard a problemei SVM cu margine “hard” este:

$$\begin{aligned} \min_{w, w_0} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i(w \cdot x_i + w_0) \geq 1 \quad (\forall 1 \leq i \leq m) \end{aligned}$$

Arătați că soluția acestei probleme rămâne efectiv neschimbăță atunci când numărul 1 din partea dreaptă a restricției este înlocuit cu o constantă oarecare pozitivă a .

Răspuns:

²⁷⁶Vedeți problema 1.

Setul de inegalități $y_i(w \cdot x_i + w_0) \geq 1$, cu $i = 1, \dots, m$ constituie o condiție suficientă pentru separabilitatea liniară a instanțelor de antrenament $x_i \in \mathbb{R}^d$. Această inegalitate devine egalitate exclusiv pentru punctele care sunt vectori-suport. Hiperplanul de separare optimală are ecuația $w \cdot x + w_0 = 0$. Distanța dintre acest hiperplan și oricare dintre vectorii-suport este deci $1/\|w\|$.

Dacă numărul 1 din partea dreaptă a restricției este înlocuit cu o constantă oarecare pozitivă a , vom avea următoarea echivalentă între probleme:

$$\begin{aligned} \min_{w, w_0} \frac{1}{2} \|w\|^2 &\Leftrightarrow \min_{w, w_0} \frac{1}{2} \|w\|^2 \\ y_i(w \cdot x_i + w_0) \geq a \quad (i = 1, \dots, m) &\Leftrightarrow y_i\left(\frac{1}{a}w \cdot x_i + \frac{1}{a}w_0\right) \geq 1 \quad (i = 1, \dots, m) \\ \min_{w, w_0} \frac{1}{2} \left\| \frac{1}{a}w \right\|^2 &\Leftrightarrow \min_{w', w'_0} \frac{1}{2} \|w'\|^2 \\ y_i\left(\frac{1}{a}w \cdot x_i + \frac{1}{a}w_0\right) \geq 1 \quad (i = 1, \dots, m) &\Leftrightarrow y_i(w' \cdot x_i + w'_0) \geq 1 \quad (i = 1, \dots, m) \end{aligned}$$

unde $w' \stackrel{\text{not.}}{=} \frac{1}{a}w$ și $w'_0 \stackrel{\text{not.}}{=} \frac{1}{a}w_0$. Echivalența se datorează pozitivității constantei a . Așadar, până la un factor pozitiv ($1/a$), soluția (w, w_0) rămâne neschimbată.

Evident, hiperplanul de separare optimală rămâne același. La fel, mulțimea vectorilor-suport este neschimbată. Distanța dintre hiperplan și vectorii-suport devine $a/\|w\|$, iar egalitatea $y_i(w \cdot x_i + w_0) = a$ va fi adevărată doar pentru punctele x_i care sunt vectori-suport.

8.

(O metodă particulară de mapare a atributelor, care asigură separare liniară)

University of Utah, 2008 fall, Hal Daumé III, HW3, pr. 2

Considerăm N puncte într-un spațiu D -dimensional, nu neapărat separabile liniar. În acest exercițiu vom arăta că aceste puncte se pot mapa într-un spațiu cu $D+N$ dimensiuni, încât să devină separabile liniar. Acest lucru se realizează transformând punctele $\vec{x}_n \in \{x_1, x_2, \dots, x_D\}$ în puncte de forma $(x_1, x_2, \dots, x_D, 1_{\{n=1\}}, 1_{\{n=2\}}, 1_{\{n=3\}}, \dots, 1_{\{n=N-1\}}, 1_{\{n=N\}})$, unde de exemplu prin notația $1_{\{n=3\}}$ se înțelege valoarea 1 dacă $n = 3$ și 0 în caz contrar. Arătați că punctele obținute în urma acestei mapări sunt într-adevăr separabile liniar.

Răspuns:

Considerând $\vec{x}_1, \dots, \vec{x}_N$ instanțele date și $\vec{X}_1, \dots, \vec{X}_N$ imaginile lor din spațiul $(D+N)$ -dimensional, vom arăta că există $\vec{w} \in \mathbb{R}^{D+N}$ astfel încât semnul expresiei $\vec{w} \cdot \vec{X}_i$ este pozitiv pentru instanțele etichetate pozitiv și negativ în rest.

Dacă luăm $\vec{w} = (0, 0, \dots, 0, y_1, y_2, \dots, y_N) \in \mathbb{R}^{D+N}$, unde

$$y_i = \begin{cases} 1 & \text{dacă punctul } X_i \text{ este etichetat cu +} \\ -1 & \text{dacă punctul } X_i \text{ este etichetat cu -}, \end{cases}$$

rezultă că

$$\vec{w} \cdot \vec{X}_i = \sum_{i=1}^N y_i \cdot X_i^{(D+i)} = y_i, \forall i \in \{1, \dots, N\}$$

adică $\vec{w} \cdot \vec{X}_i = +1$ pentru toate instanțele \vec{x}_i pozitive și $\vec{w} \cdot \vec{X}_i = -1$ pentru toate instanțele \vec{x}_i negative.

Așadar, hiperplanul $\vec{w} \cdot \vec{X} = 0$ este un separator liniar pentru cele N puncte din spațiul $(D+N)$ -dimensional.

Observație: Deși are o anumită importanță teoretică, maparea indicată în enunțul acestei probleme nu este eficientă din punct de vedere practic, în special atunci când N și / sau D sunt numere foarte mari.

9.

(Exemplu de folosire a unei funcții de mapare a atributelor cu scopul de a obține separabilitate liniară; rezolvarea directă a problemei SVM primale în spațiul [nou] de trăsături; identificarea separatorului neliniar din spațiul inițial [de trăsături])

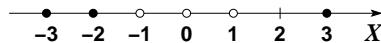
CMU, 2009 fall, Carlos Guestrin, HW3, pr. 2.1

Se dă un set de date de antrenament D , caracterizate de atributul X care ia valori în \mathbb{R} și de eticheta corespunzătoare $y \in \{+1, -1\}$. Acest set de date este constituit din trei exemple pozitive și anume pentru $X \in \{-3, -2, 3\}$ și trei exemple negative pentru $X \in \{-1, 0, 1\}$.

- Putem separa acest set de date (în spațiul de trăsături specificat mai sus) folosind un separator liniar? De ce da, sau de ce nu?
- Definim funcția $\Phi(u) = (u, u^2)$ care transformă / „mapează“ punctele din \mathbb{R} în \mathbb{R}^2 . Aplicați funcția Φ datelor din D și trasați imaginea lor în \mathbb{R}^2 , noul spațiu de trăsături. Poate un separator liniar să separe în mod perfect punctele din noul spațiu de trăsături, obținut prin funcția Φ ? De ce da, sau de ce nu?
- Găsiți forma analitică a funcției-nucleu $K(x, x')$ care corespunde transformării Φ . Vă reamintim că $K(x, x') \stackrel{\text{def.}}{=} \Phi(x) \cdot \Phi(x')$.
- Identificați un *hiperplan de separare optimală* pentru mulțimea $\Phi(D)$. Acest hiperplan este o dreaptă în planul euclidian, caracterizată de o ecuație de forma $w_1Y_1 + w_2Y_2 + w_0 = 0$, unde $\Phi(X) = (Y_1, Y_2)$. Găsiți valorile lui w_0, w_1 și w_2 ; conform problemei de optimizare SVM, pentru fiecare vector-suport X având eticheta y trebuie să avem satisfăcută restricția $w_1Y_1 + w_2Y_2 + w_0 = y$. Desenați hiperplanul de separare optimală și încercuiți vectorii-suport. De asemenea, calculați marginea hiperplanului, adică distanța de la hiperplan la vectorii-suport.
- Desenați în \mathbb{R} corespondentul hiperplanului de separare optimală din \mathbb{R}^2 .
- Dacă adăugăm încă un punct pozitiv ($y=+1$) la mulțimea de antrenament, și anume $X=5$, se va schimba hiperplanul de separare optimală sau marginea lui? De ce da, sau de ce nu?

Răspuns:

- Figura de mai jos reprezintă datele de antrenament în spațiul original, \mathbb{R} , folosind puncte negre pentru exemplele pozitive și puncte albe pentru exemplele negative.



Conform *definiției*, un separator liniar în mulțimea numerelor reale este un punct de pe axă care are de o parte a sa toate exemplele pozitive și de cealaltă parte toate exemplele negative. Se observă foarte ușor că nu există niciun punct pe axa reală care să satisfacă această condiție. Prin urmare, mulțimea datelor de antrenament nu este separabilă liniar în spațiul de trăsături inițial.

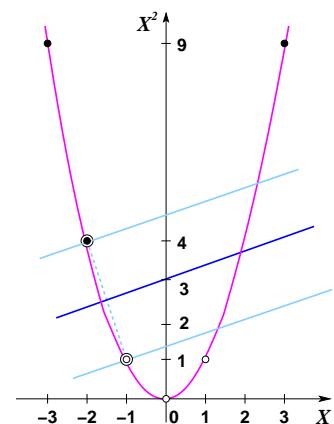
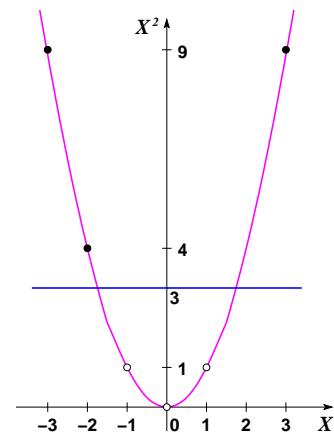
b. Figura alăturată reprezintă imaginea setului de antrenament în noul spațiu de trăsături, \mathbb{R}^2 . Se observă ușor că aici datele sunt separabile liniar. Putem lua, de exemplu, dreapta de ecuație $y - 3 = 0$, reprezentată în figura alăturată. Avem deci $w_1 = 0, w_2 = 1, w_0 = -3$. *Analitic*, se observă imediat că sunt îndeplinite inegalitățile de forma $\text{sign}(w_1 u_i + w_2 u_i^2 + w_0) = y_i$:

$$\begin{cases} 0 \cdot (-3) + 1 \cdot 9 - 3 &= 6 > 0 \\ 0 \cdot (-2) + 1 \cdot 4 - 3 &= 1 > 0 \\ 0 \cdot (-1) + 1 \cdot 1 - 3 &= -2 < 0 \\ 0 \cdot 0 + 1 \cdot 0 - 3 &= -3 < 0 \\ 0 \cdot 1 + 1 \cdot 1 - 3 &= -2 < 0 \\ 0 \cdot 3 + 1 \cdot 9 - 3 &= 6 > 0 \end{cases}$$

c. Conform *definiției* funcției-nucleu,

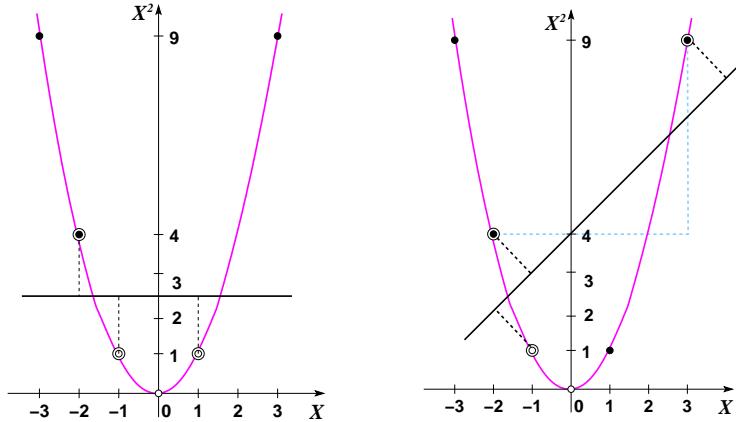
$$K(x, x') = \Phi(x) \cdot \Phi(x') = (x, x^2) \cdot (x', x'^2) \Rightarrow K(x, x') = xx' + x^2x'^2.$$

d. Hiperplanul de separare optimală pentru setul de date de antrenament D din enunț este determinat de punctele $(-2, 4)$ și $(-1, 1)$. Se observă că aceste două puncte constituie perechea de puncte de semne / etichete diferite aflate la distanță minimă, între toate perechile de puncte de semne contrare din D . Mediațoarea punctelor $(-2, 4)$ și $(-1, 1)$ este hiperplanul de separare maximală pentru mulțimea D , iar cele două puncte sunt vectorii-suport. Marginea, adică distanța dintre orice vector-suport și hiperplanul de separare optimală este $\frac{\sqrt{10}}{2} = \sqrt{\frac{5}{2}}$.



Observație: Se poate verifica [si] în mod *analitic* că, dintre toți separatorii liniari ai mulțimii D , dreapta de mai sus maximizează distanța până la cele mai apropiate instanțe pozitive și respectiv negative. În particular, comparând marginea corespunzătoare mediatoarei punctelor $(-2, 4)$ și $(-1, 1)$ cu marginile determinate respectiv de către cele două drepte din figurile de mai jos, vom obține: $\sqrt{\frac{5}{2}} > \frac{3}{2}$ pentru figura din partea stângă și $\sqrt{\frac{5}{2}} > \sqrt{2}$ pentru figura din partea dreaptă.²⁷⁷

²⁷⁷Mai mult, cele două drepte menționate au poziții extreme în raport cu toate dreptele care – separă instanțele din D ,



Ecuăția separatorului optimal se determină astfel:

- ecuația dreptei determinată de punctele \$(-2, 4)\$ și \$(-1, 1)\$ este

$$\frac{x - (-1)}{-2 - (-1)} = \frac{y - 1}{4 - 1} \Leftrightarrow 3(x + 1) = -(y - 1) \Leftrightarrow y = -(3x + 2),$$

așadar, panta acestei drepte este \$-3\$;

- orice dreaptă perpendiculară pe dreapta de mai sus are panta \$-\frac{1}{-3} = \frac{1}{3}\$ și deci este determinată de o ecuație de forma \$y = \frac{1}{3}x + c\$, unde \$c\$ este o constantă reală;
- mijlocul segmentului determinat de punctele \$(-2, 4)\$ și \$(-1, 1)\$ este \$\left(-\frac{3}{2}, \frac{5}{2}\right)\$;
- mediatoarea segmentului determinat de punctele \$(-2, 4)\$ și \$(-1, 1)\$ va determina valoarea corespunzătoare pentru constanta \$c\$, prin faptul că punctul \$\left(-\frac{3}{2}, \frac{5}{2}\right)\$ aparține acestei mediatoare, deci:

$$\frac{5}{2} = \frac{1}{3} \left(-\frac{3}{2}\right) + c \Leftrightarrow c = \frac{5}{2} + \frac{1}{2} = 3;$$

- așadar, ecuația separatorului optimal este:

$$y = \frac{1}{3}x + 3 \Leftrightarrow x - 3y + 9 = 0 \Leftrightarrow -\frac{1}{5}x + \frac{3}{5}y - \frac{9}{5} = 0.$$

În legătură cu ultima formă a ecuației de mai sus se poate observa că expresia \$-\frac{1}{5}x + \frac{3}{5}y - \frac{9}{5}\$ satisface condiția — exprimată în definiția problemei SVM, forma primală — de a produce

- trec prin punctul de la jumătatea segmentului determinat de punctele \$(-2, 4)\$ și \$(-1, 1)\$, care sunt cele mai apropiate puncte de semne / etichete contrare din \$D\$;
- nu se apropie de punctul \$(3, 9)\$ și respectiv \$(1, 1)\$ mai mult decât este distanța până la fiecare din punctele \$(-2, 4)\$ și \$(-1, 1)\$.

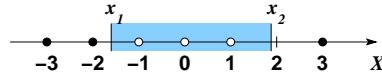
valorile 1 și respectiv -1 pentru punctele $(-2, 4)$ și $(-1, 1)$. Așadar, funcția care definește separatorul optimal este $-\frac{1}{5}x + \frac{3}{5}y - \frac{9}{5}$, iar soluțiile sunt $\bar{w} = \left(-\frac{1}{5}, \frac{3}{5}\right)$ și $\bar{w}_0 = -\frac{9}{5}$.

Rezultă că marginea este $\frac{1}{\|\bar{w}\|} = \frac{5}{\sqrt{1+9}} = \sqrt{\frac{5}{2}}$.

e. Clasificarea unui punct $x \in \mathbb{R}$ cu separatorul determinat la punctul d se face în felul următor:

$$y = \text{sign}(\bar{w} \cdot \Phi(x) + \bar{w}_0) = \text{sign}\left(\left(-\frac{1}{5}, \frac{3}{5}\right) \cdot (x, x^2) - \frac{9}{5}\right) = \text{sign}\left(\frac{3}{5}x^2 - \frac{1}{5}x - \frac{9}{5}\right)$$

Prin urmare, vom avea eticheta $y = -1$ dacă și numai dacă $3x^2 - x - 9 < 0 \Leftrightarrow x \in (x_1, x_2)$, unde $x_1 = \frac{1 - \sqrt{1 + 12 \cdot 9}}{6} \approx -1.57$ iar $x_2 = \frac{1 + \sqrt{1 + 12 \cdot 9}}{6} \approx 1.90$. Corespondentul hiperplanului de separare din \mathbb{R}^2 în spațiul inițial de trăsături este ilustrat în figura de mai jos, sub forma „tăieturilor“ reprezentate de x_1 și x_2 :



Observație: Rezultatul de mai sus corespunde (intuitiv) cu rezultatul de la punctul precedent (d): instanțele (x, x^2) situate de partea „pozitivă“ a dreptei de ecuație $-\frac{1}{5}x + \frac{3}{5}y - \frac{9}{5} = 0 \Leftrightarrow y = \frac{1}{3}x + 3$ satisfac condiția $x^2 > y \Leftrightarrow x^2 > \frac{1}{3}x + 3 \Leftrightarrow 3x^2 - x - 9 > 0$. Similar, instanțele (x, x^2) situate de partea „negativă“ a dreptei de ecuație $-\frac{1}{5}x + \frac{3}{5}y - \frac{9}{5} \Leftrightarrow y = \frac{1}{3}x + 3$ satisfac condiția $x^2 < y \Leftrightarrow x^2 < \frac{1}{3}x + 3 \Leftrightarrow 3x^2 - x - 9 < 0$.

f. Se observă ușor (vedeți prima figură de la punctul d) că imaginea noului exemplu de antrenament (5, 25) cade de partea „pozitivă“ a hiperplanului de separare și, mai mult, cade în afara marginii. Prin urmare, separatorul optimal rămâne același.

10.

(Exercițiu teoretic: deducerea formei duale pentru problema SVM cu margine “hard”)

*prelucrare de L. Ciortuz, după
■ CMU, 2010 fall, Ziv Bar-Joseph, HW4, pr. 1.3-5*

Se consideră vectorii de intrare $x_1, \dots, x_m \in \mathbb{R}^d$ și etichetele corespunzătoare $y_1, \dots, y_m \in \{-1, 1\}$. Problema SVM cu margine “hard” — termen care desemnează cazul în care instanțele $(x_1, y_1), \dots, (x_n, y_n)$ se presupune că sunt liniar separabile — este o problemă de optimizare convexă, exprimată sub forma primală astfel:

$$\min_{w, w_0} \frac{1}{2} \|w\|^2 \quad (P)$$

a. î. $(w \cdot x_i + w_0)y_i \geq 1$, pentru $i = 1, \dots, m$,

unde $w \in \mathbb{R}^d$ și $w_0 \in \mathbb{R}$. În urma rezolvării acestei probleme se va obține un *model* liniar, de forma $y(x) = w \cdot x + w_0$, ce va servi ulterior pentru clasificare, conform funcției de decizie $\text{sign}(y(x))$.²⁷⁸

La curs am prezentat *metoda dualității Lagrange*, care ne permite în anumite condiții să rezolvăm probleme de *optimizare convexă cu restricții de asemenea convexe*, transpunând aceste probleme într-o formă mai convenabilă numită *forma duală*. Pentru început, vom defini o altă funcție, numită *lagrangeanul generalizat*,²⁷⁹ care combină funcția obiectiv din (P) cu expresiile care intervin în partea stângă a restricțiilor:²⁸⁰

$$L_P(w, w_0, \alpha) \stackrel{\text{def.}}{=} \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i ((w \cdot x_i + w_0) y_i - 1),$$

unde $\alpha_i \geq 0$ pentru $i = \overline{1, m}$ sunt așa-numiții multiplicatori Lagrange sau *variabilele duale*. Variabilele *primale* sunt w și w_0 .²⁸¹

²⁷⁸ Este ușor de constatat faptul că problema (P) este un caz particular de problemă de optimizare convexă cu restricții (vedeți nota de subsol 763 de la pr. 73 de la capitolul de *Fundamente*) și, conform teoremei Karush-Kuhn-Tucker (vedeți nota de subsol 782 de la pr. 76 tot de la capitolul de *Fundamente*), ea are soluție atunci când *regiunea fezabilă* (engl., feasible region), adică mulțimea convexă formată din valorile lui w și w_0 pentru care *restricțiile* $(w \cdot x_i + w_0) y_i \geq 1$ cu $i = \overline{1, m}$ sunt satisfăcute este nevidă.

Din punct de vedere *computațional*, rezolvarea acestei probleme devine neficientă atunci când m , numărul de instanțe de antrenament, este foarte mare. Dificultatea ține de satisfacerea restricțiilor. Ca să depășim această dificultate, *ideea* de bază este să punem problema noastră sub o altă formă, cu restricții mai simple. „Prețul” pe care va trebui să-l plătim în schimb este „complicarea” funcției obiectiv; se va dovedi ulterior că acest schimb este convenabil.

²⁷⁹ Unii autori numesc funcția L_P *lagrangeanul primal*. Pentru problema de optimizare convexă (cazul general), el se definește astfel:

$$L_P(x, \alpha, \beta) \stackrel{\text{def.}}{=} f(x) + \sum_i \alpha_i g_i(x) + \sum_j \beta_j h_j(x).$$

Vedeți nota de subsol 764 de la pr. 73 de la capitolul de *Fundamente*.

²⁸⁰ Semnul minus ($-$) din fața simbolului de sumare (\sum) din funcția L_P corespunzătoare problemei de optimizare SVM în forma primală (P) apare din cauza faptului că restricțiile din cadrul acestei probleme sunt de tip \geq , în timp ce restricțiile din cadrul formulării generale a problemei de optimizare convexă (vedeți nota de subsol 763 de la problema 73 de la capitolul de *Fundamente*) sunt de tip \leq .

²⁸¹ Vă readucem aminte următoarele *puncte principale* ale *teoriei dualității Lagrange*, care justifică de ce anume procedăm aşa cum procedăm în rezolvarea problemei noastre.

- [P→P1] Este relativ ușor de arătat — vedeți documentul *Convex Optimization Overview (cont'd)*, de Chuong B. Do, 2009, pag. 4-5 sau documentul *Support Vector Machines* de Andrew Ng, (Stanford University, CS229 Lecture Notes, Part V), pag. 8-9 — că problema (P) este echivalentă cu următoarea problemă de optimizare

$$\begin{aligned} & \min_x \max_{\alpha, \beta} L_P(x, \alpha, \beta) \\ \text{a. i. } & \alpha_i \geq 0 \text{ pentru } i = 1, \dots, m. \end{aligned} \tag{P1}$$

Se observă că în această problemă restricțiile sunt mult mai simple decât în problema (P).

- [P1→D1] Inversând operatorii min și max în problema (P1), obținem problema de optimizare convexă

$$\begin{aligned} & \max_{\alpha, \beta} \min_x L_P(x, \alpha, \beta) \\ \text{a. i. } & \alpha_i \geq 0 \text{ pentru } i = 1, \dots, m. \end{aligned} \tag{D1}$$

Dacă notăm cu p^* optimul problemei (P1) (deci și a problemei (P)) și cu d^* optimul problemei (D1), este relativ ușor de demonstrație — vedeți problema 73 de la capitolul de *Fundamente* și / sau documentul *Convex Optimization Overview (cont'd)*, de Chuong B. Do, 2009, pag. 5-6 —

a. Calculând derivatele parțiale ale funcției L_P în raport cu variabilele primale w și w_0 , arătați că între valorile \bar{w} , \bar{w}_0 și $\bar{\alpha}$ pentru care aceste derivate parțiale se anulează există relațiile:

$$\bar{w} = \sum_{i=1}^m \bar{\alpha}_i x_i y_i, \quad (115)$$

$$\sum_{i=1}^m \bar{\alpha}_i y_i = 0. \quad (116)$$

proprietatea de *dualitate slabă*:

$$d^* \leq p^*. \quad (112)$$

Proprietatea de dualitate slabă este valabilă și în cazul general al problemelor de optimizare convexă [cu restricții].

- Este imediat că pentru problema (P1) este satisfăcută aşa-numita *condiție a lui Slater* (vedeți *Learning with Kernels*, B. Schölkopf, A. Smola, MIT Press, 2002, pag 167):

$$\exists w \in \mathbb{R}^d \text{ și } w_0 \in \mathbb{R} \text{ astfel încât } (w \cdot x_i + w_0)y_i - 1 > 0, \text{ pentru } i = 1, \dots, m. \quad (113)$$

Într-adevăr, faptul că instanțele $(x_1, y_1), \dots, (x_n, y_n)$ sunt liniar separabile implică imediat satisfacerea condiției de mai sus. Se demonstrează că în general, adică pentru orice problemă de optimizare convexă cu restricții, dacă este îndeplinită condiția lui Slater (sub forma $\exists x$ a. î. $g_i(x) < 0$ pentru $i = 1, \dots, n$ și $g_j(x) < 0$ pentru $j = 1, \dots, p$), atunci are loc egalitatea

$$d^* = p^*,$$

care reprezintă proprietatea de *dualitate tare*. Așadar, optimul problemei (D1) este optim și pentru problema (P1) și, corespunzător, pentru problema (P).

O teoremă importantă din teoria problemelor de optimizare convexă — vedeți documentul *Convex Optimization Overview (cont'd)*, de Chuong B. Do, 2009, pag. 6 sau *Learning with Kernels*, B. Schölkopf, A. Smola, MIT Press, 2002, pag 165, Teorema 6.21 — demonstrează următoarea *implicație*: în cazul în care proprietatea de dualitate tare este satisfăcută, notând cu $\bar{\alpha}, \bar{\beta}$ o soluție a problemei (D1), urmează că

$$\bar{\alpha}_i g_i(\bar{x}) = 0 \text{ pentru } i = 1, \dots, n.$$

Aceste egalități se numesc *condițiile de complementaritate duală Karush-Kuhn-Tucker*.

Pentru problema SVM cu margine “hard” din enunț, considerăd \bar{w} , \bar{w}_0 și $\bar{\alpha}$ soluțiile problemei (D1) (deci și ale problemei (P1)), condițiile de complementaritate duală Karush-Kuhn-Tucker se vor exprima astfel:

$$\bar{\alpha}_i ((\bar{w} \cdot x_i + \bar{w}_0)y_i - 1) = 0 \text{ pentru } i = 1, \dots, m, \quad (114)$$

adică, pentru fiecare valoare admisibilă a lui i , avem fie $\bar{\alpha}_i = 0$, fie $\bar{\alpha}_i > 0$ și, în consecință, $(\bar{w} \cdot x_i + \bar{w}_0)y_i - 1 = 0$.

- [D1→D] În fine, pentru că funcția L_P este derivabilă în raport cu w și respectiv w_0 , este posibil ca expresia lui $\min_{w, w_0} L_P(w, w_0, \alpha)$ din cadrul funcției obiectiv a problemei (D1) să se obțină calculând rădăcinile derivatelor parțiale ale lui L_P . (Se poate arăta — vedeți documentul *Convex Optimization Overview (cont'd)*, de Chuong B. Do, 2009, pag. 5 — că funcția $\min_{w, w_0} L_P(w, w_0, \alpha)$ este concavă.) Se va ajunge astfel la a asocia problemei (P) forma duală (D) din enunț; vedeți punctul b.

Raționamentul acesta este valabil în general pentru problemele de optimizare convexă în care toate funcțiile f , g_i și h_j sunt convexe și derivabile.

Condițiile ca derivatele parțiale ale lui $L_P(w, w_0, \alpha)$ în raport cu w și respectiv w_0 să se anuleze se numesc *condițiile de optimalitate* (sau: *staționaritate*) *Karush-Kuhn-Tucker*.

De asemenea, arătați că din *condiția de complementaritate Karush-Kuhn-Tucker*²⁸² se poate deduce relația:

$$\bar{w}_0 = y_i - \bar{w} \cdot x_i \text{ pentru orice } i \text{ astfel încât } \bar{\alpha}_i > 0. \quad (117)$$

Notă: Instanțele x_i pentru care $\bar{\alpha}_i > 0$ sunt numite vectori-suport [în sens *clasic*, analitic].²⁸³ Aceasta este definiția pe care o vom folosi de acum încolo pentru această noțiune.

b. Calculați funcția $L_D(\alpha)$ care se obține din expresia lagrangeanului generalizat $L_P(w, w_0, \alpha)$ substituind variabila w cu $\sum_{i=1}^m \alpha_i x_i y_i$ și folosind egalitatea $\sum_{i=1}^m \alpha_i y_i = 0$, conform relațiilor (115) și (116) de la punctul precedent.

Observație (1): Este evident acum că problemei (P) îi poate fi asociată următoarea formă (numită „duală“):

$$\begin{aligned} & \max_{\alpha} L_D(\alpha) \\ \text{a. i. } & \alpha_i \geq 0, \text{ pentru } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (\text{D})$$

în care *restricțiile* sunt mult mai *simple* decât erau în forma primală (P). Este de reținut faptul că relațiile (115) și (117) de la punctul precedent constituie legătura dintre soluția / soluțiile problemei (D) și soluția problemei (P).

c. Dacă se dă o instanță nouă (de test) x_{new} , cum veți decide clasa ei?

Răspuns:

a. Calculăm mai întâi derivatele parțiale ale funcției L_P în raport cu w și respectiv w_0 :²⁸⁴

$$\begin{aligned} \frac{\partial}{\partial w} L_P(w, w_0, \alpha) &= w - \sum_{i=1}^m \alpha_i x_i y_i \\ \frac{\partial}{\partial w_0} L_P(w, w_0, \alpha) &= - \sum_{i=1}^m \alpha_i y_i. \end{aligned}$$

Atunci când se atinge optimul funcției L_P (considerând argumentul său α fixat), aceste derivate parțiale devin egale cu 0. Din $\frac{\partial}{\partial w} L_P(\bar{w}, \bar{w}_0, \bar{\alpha}) = 0$ rezultă că

$$\bar{w} = \sum_{i=1}^m \bar{\alpha}_i x_i y_i.$$

²⁸²Vedeți nota de subsol 281.

²⁸³Vedeți *Observația importantă* de la pag. 337.

²⁸⁴În notația matriceală, considerând w și x_i pentru $i = 1, \dots, m$ vectori-colonă, putem scrie lagrangeanul L_P astfel:

$$L_P(w, w_0, \alpha) = \frac{1}{2} w^\top w - \sum_{i=1}^m \alpha_i ((w^\top x_i + w_0) y_i - 1).$$

Pentru derivarea lui L_P în raport cu vectorul w , se folosesc regulile (5a) și (5b) din documentul *Matrix Identities* de Sam Roweis (New York University, June 1999), pe care le-am folosit (de exemplu) și la problema 20 de la capitolul *Clusterizare*.

Revenind la formula inițială, care folosește notația vectorială și produsul scalar din \mathbb{R}^d , se poate constata că într-un astfel de caz (simplu!), regulile de derivare sunt similare cu cele din \mathbb{R} .

Similar, $\frac{\partial}{\partial \alpha_i} L_P(\bar{w}, \bar{w}_0, \bar{\alpha}) = 0$ implică relația $\sum_{i=1}^m \bar{\alpha}_i y_i = 0$.

În fine, din condiția de complementaritate Karush-Kuhn-Tucker, care se exprimă sub forma

$$\bar{\alpha}_i[(\bar{w} \cdot x_i + \bar{w}_0)y_i - 1] = 0 \text{ pentru } i = 1, \dots, m$$

rezultă că pentru orice $i \in \{1, \dots, m\}$ cu $\bar{\alpha}_i > 0$ avem $(\bar{w} \cdot x_i + \bar{w}_0)y_i - 1 = 0$.²⁸⁵ Această relație este echivalentă cu $\bar{w} \cdot x_i + \bar{w}_0 = y_i$ fiindcă $y_i \in \{-1, 1\}$. Din această ultimă egalitate rezultă:

$$\bar{w}_0 = y_i - \bar{w} \cdot x_i.$$

b. Substituind $w = \sum_{i=1}^m \alpha_i x_i y_i$ în expresia lui L_P și ținând cont că $\sum_{i=1}^m \alpha_i y_i = 0$, vom obține:

$$\begin{aligned} L_D(\alpha) &= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i=1}^m \alpha_i \sum_{j=1}^m [(\alpha_j y_j x_j \cdot x_i + w_0) y_i - 1] \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \end{aligned} \quad (118)$$

c. Mai întâi vom calcula

$$\begin{aligned} f(x_{new}) &= \bar{w} \cdot x_{new} + \bar{w}_0 \\ &= \left(\sum_{i=1}^m \bar{\alpha}_i y_i x_i \right) \cdot x_{new} + \bar{w}_0 = \sum_{i=1}^m \bar{\alpha}_i y_i x_i \cdot x_{new} + \bar{w}_0, \end{aligned} \quad (119)$$

unde $\bar{\alpha}$ este soluția problemei duale (D), iar \bar{w} și \bar{w}_0 , soluțiile problemei primale (P) sunt calculate conform relațiilor (115) și (117) de la punctul a.

După aceea, dacă $f(x_{new}) \geq 0$ atunci x_{new} va fi clasificat pozitiv, iar în caz contrar va fi clasificat negativ.

Observație (2): Remarcăm faptul că atât în funcția obiectiv a problemei de optimizare SVM în formă duală (D) cât și în funcția f care servește la clasificarea instanțelor noi, operațiile care se execută asupra instanțelor sunt doar de tip produs scalar: $x_i \cdot x_j$ și respectiv $x_i \cdot x_{new}$. Acest fapt face posibilă folosirea *funcțiilor-nucleu* în contextul SVM (așa cum vom exemplifica la problema 11), ceea ce este convenabil atât din punctul de vedere al obtinerii (eventuale) a separabilității, cât și din punctul de vedere al executării eficiente a calculelor.

²⁸⁵Dacă $\bar{\alpha}_i = 0$ pentru $i = \overline{1, m}$, din relația $\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i$ rezultă că $\bar{w} = 0$. În consecință, funcția $f(x) = \bar{w} \cdot x + \bar{w}_0$ care dă ecuația separatorului optimal ($f(x) = 0$) nu va avea putere de discriminare între instanțele pozitive și instanțele negative (indiferent de valoarea atribuită lui \bar{w}_0 , care este o constantă).

De fapt, atunci când s-a introdus problema SVM ca o problemă de optimizare a marginii / distanței $\frac{1}{\|w\|}$, s-a considerat în mod implicit că se caută soluții $w \neq 0$.

Așadar, în cazul în care în urma rezolvării problemei primale, respectiv a celei duale — urmată în ultimul caz de aplicarea relațiilor de legătură între cele două tipuri / seturi de soluții — obținem $\bar{w} = 0$, căutarea lui \bar{w}_0 (valoarea optimă pentru w_0) pur și simplu nu are sens.

11.

(Învățarea conceptului \neg XOR folosind forma duală a problemei SVM și o mapare particulară a trăsăturilor)

CMU, 2006 fall, E. Xing, T. Mitchell, midterm exam, pr. 5

Fie o problemă de învățare supervizată în care exemplele de antrenare se află în spațiul euclidian bidimensional. Exemplele pozitive sunt $x_1 = (1, 1)$ și $x_3 = (-1, -1)$ iar exemplele negative sunt $x_2 = (-1, 1)$ și $x_4 = (1, -1)$.

- Sunt exemplele pozitive separabile liniar față de exemplele negative?
- Considerăm transformarea $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^4$, definită astfel: $\Phi(u) = (1, u_1, u_2, u_1 u_2)$, unde $u = (u_1, u_2)$. Funcția de predicție pe care vrem să o obținem este de forma $y(x) = w \cdot \Phi(x) + w_0$, unde $x \in \mathbb{R}^4$, $w \in \mathbb{R}^4$, $w_0 \in \mathbb{R}$ (cu x variabil și w și w_0 fixați), iar \cdot reprezintă produsul scalar. Indicați coeficienții w și w_0 corespunzători unui hiperplan de separare cu margine maximă pentru imaginea punctelor de antrenament din enunț în „spațiul de trăsături“ (\mathbb{R}^4).

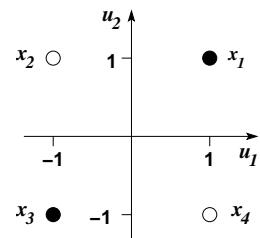
Indicație: Vă recomandăm să rezolvați problema SVM în *formă duală* și apoi să exprimați *soluția formei primale* folosind *relațiile* dintre cele două tipuri de soluții, care au fost date la curs.²⁸⁶ (*Observație:* Se poate constata în prealabil că mulțimea $\{\Phi(x_1), \Phi(x_2), \Phi(x_3), \Phi(x_4)\}$ este liniar separabilă și (ca atare) în *spațiul de trăsături* determinat de maparea Φ este satisfăcută condiția lui Slater.)

- Adăugați un nou exemplu la mulțimea de date de antrenament, astfel încât această nouă mulțime să nu mai fie separabilă liniar în spațiul corespunzător transformării Φ de la punctul anterior.
- Cărei funcții-nucleu (engl., kernel function) îi corespunde transformarea Φ ?

Răspuns:

- Reprezentând grafic datele într-un sistem de coordonate bidimensional, se poate observa imediat că ele nu sunt separabile liniar. *Analitic*, putem demonstra acest lucru după cum urmează:

Să presupunem prin reducere la absurd că exemplele negative sunt separabile liniar de cele pozitive în \mathbb{R}^2 . Atunci ar exista $\bar{w} = (w_1, w_2) \in \mathbb{R}^2$ și $w_0 \in \mathbb{R}$ astfel încât:



$$\left. \begin{array}{l} w_1 + w_2 + w_0 > 0 \\ -w_1 - w_2 + w_0 > 0 \\ -w_1 + w_2 + w_0 < 0 \\ w_1 - w_2 + w_0 < 0 \end{array} \right\} \Rightarrow \left. \begin{array}{l} 2w_0 > 0 \Rightarrow w_0 > 0 \\ 2w_0 < 0 \Rightarrow w_0 < 0 \end{array} \right\} \Rightarrow \text{contradicție!}$$

Așadar, presupunerea făcută este falsă. În consecință, cele 4 puncte nu sunt liniar separabile în \mathbb{R}^2 .

- Conform formei duale a problemei SVM,²⁸⁷ avem de maximizat funcția „lagrangeană“ L_D , unde:

$$L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \sum_{i=1}^4 \alpha_i - \frac{1}{2} \sum_{i,j=1}^4 \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) ,$$

²⁸⁶ Adică relațiile (115) și (117).

²⁸⁷ Vedeti problema de optimizare (D) de la pagina 351 și relația (118).

ținând de asemenea cont de restricțiile $\sum_{i=1}^4 y_i \alpha_i = \alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$ și $\alpha_i \geq 0$ pentru $i = \overline{1, 4}$.

Conform definiției funcției de mapare Φ , se observă că $\Phi(x_i) \cdot \Phi(x_i) = 4$ pentru $i = \overline{1, 4}$ și $\Phi(x_i) \cdot \Phi(x_j) = 0$ pentru orice $i, j = \overline{1, 4}$ cu $i \neq j$.

Înlocuind aceste produse în expresia lui L_D de mai sus, obținem:

$$\begin{aligned} L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(4\alpha_1^2 + 4\alpha_2^2 + 4\alpha_3^2 + 4\alpha_4^2) = \\ &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - 2\alpha_1^2 - 2\alpha_2^2 - 2\alpha_3^2 - 2\alpha_4^2. \end{aligned}$$

$\Phi(x_1)$	$\Phi(x_2)$	$\Phi(x_3)$	$\Phi(x_4)$
1	1	1	1
1	-1	-1	1
1	1	-1	-1
1	-1	1	-1

Vom încerca să rezolvăm problema duală căutând punctul de optim al lui L_D ,²⁸⁸ adică determinând $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ astfel încât derivatele parțiale ale lui L_D în funcție de aceste variabile să se anuleze:

$$\left. \begin{array}{l} \frac{\partial}{\partial \alpha_1} L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 1 - 4\alpha_1 = 0 \\ \frac{\partial}{\partial \alpha_2} L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 1 - 4\alpha_2 = 0 \\ \frac{\partial}{\partial \alpha_3} L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 1 - 4\alpha_3 = 0 \\ \frac{\partial}{\partial \alpha_4} L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 1 - 4\alpha_4 = 0 \end{array} \right\} \Rightarrow \bar{\alpha}_1 = \bar{\alpha}_2 = \bar{\alpha}_3 = \bar{\alpha}_4 = \frac{1}{4}.$$

Se observă că aceste soluții satisfac restricțiile din forma duală a problemei SVM: $\bar{\alpha}_i \geq 0$ pentru $i = \overline{1, 4}$ și $\sum_{i=1}^4 y_i \bar{\alpha}_i = \bar{\alpha}_1 - \bar{\alpha}_2 + \bar{\alpha}_3 - \bar{\alpha}_4 = 0$.

Pentru a determina vectorul \bar{w} din soluția formei primale a problemei SVM, vom folosi formula (115), care leagă soluția formei duale de soluția formei primale:

$$\begin{aligned} \bar{w} &= \sum_{i=1}^4 y_i \bar{\alpha}_i \Phi(x_i) \\ &= \frac{1}{4}(1, 1, 1, 1) - \frac{1}{4}(1, -1, 1, -1) + \frac{1}{4}(1, -1, -1, 1) - \frac{1}{4}(1, 1, -1, -1) = (0, 0, 0, 1). \end{aligned}$$

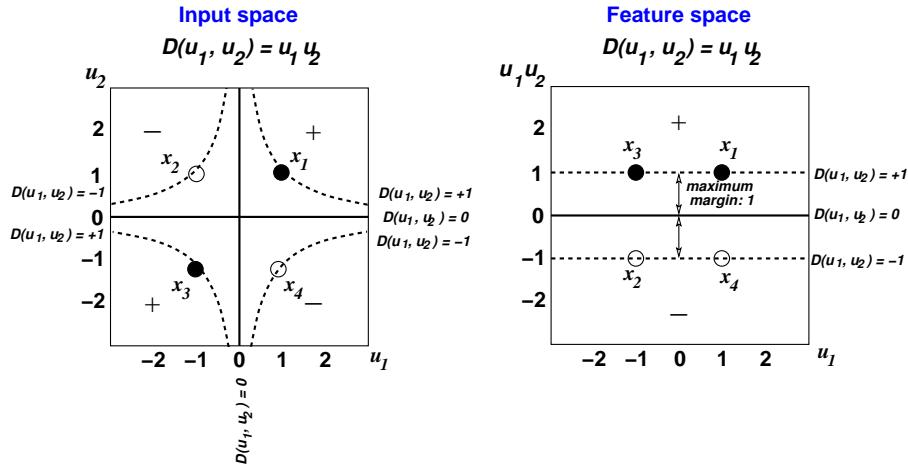
Aflarea valorii lui \bar{w}_0 se face folosind una dintre ecuațiile (117), adică $\alpha_i(y_i(\bar{w} \cdot \Phi(x_i) + \bar{w}_0) - 1) = 0$ cu $i = \overline{1, 4}$. Luând, spre exemplu, $i = 1$ obținem:

$$\begin{aligned} \frac{1}{4}(1((0, 0, 0, 1) \cdot \Phi((1, 1)) + \bar{w}_0) - 1) &= 0 \Leftrightarrow \\ (0, 0, 0, 1) \cdot (1, 1, 1, 1) + \bar{w}_0 - 1 &= 0 \Leftrightarrow 1 + \bar{w}_0 = 1 \Leftrightarrow \bar{w}_0 = 0. \end{aligned}$$

Așadar, ecuația separatorului optimal este

$$\bar{w} \cdot \Phi(u) + \bar{w}_0 = 0 \Leftrightarrow (0, 0, 0, 1) \cdot (1, u_1, u_2, u_1 u_2) + 0 = 0 \Leftrightarrow u_1 u_2 = 0,$$

²⁸⁸ Observație importantă: Se poate constata imediat că hessianul lui $L_D(\alpha)$ este o matrice negativ definită, prin urmare lagrangeanul $L_D(\alpha)$ este funcție concavă și, deci, admite un maxim. A priori, este posibil ca acest punct de maxim să fie în afara intervalului de fezabilitate, adică să nu satisfacă restricțiile $\alpha_i \geq 0$, și / sau să nu satisfacă condiția Karush-Kuhn-Tucker $\sum_i y_i \alpha_i = 0$. Vom vedea că, din fericire, în problema de față punctul de optim al lui $L_D(\alpha)$ satisfac aceste două restricții.



ceea ce corespunde mulțimii de puncte (u_1, u_2) formate din cele două axe de coordonate. Cele două „margini“ au ecuațiile $u_1 u_2 = -1$ și respectiv $u_1 u_2 = 1$ și sunt reprezentate grafic sub forma celor două hiperbole din figura de mai jos, partea stângă. Clasificarea unei instanțe oarecare de test $u = (u_1, u_2)$ se va face conform expresiei:

$$y = \text{sign}(\bar{w} \cdot \Phi(u) + \bar{w}_0) = \text{sign}(u_1 u_2).$$

Punctele din primul și din al treilea cadran vor fi clasificate pozitiv, iar punctele din al doilea și al patrulea cadran vor fi clasificate negativ.

Se constată ușor acum că în „spațiul de trăsături“ rezultat prin mapare avem separabilitate liniară, după cum indică figura următoare, partea dreaptă. (Pentru conveniență, am reținut doar coordonatele a două și a patra, adică u_1 și $u_1 u_2$. Deși nu era necesar, am fi putut adăuga și coordonata u_2 , dar atunci ar fi trebuit să facem graficul în 3D.) În acest spațiu, separatorul liniar ($u_1 u_2 = 0$) reprezintă o dreaptă; la fel și marginile ($u_1 u_2 = -1$ și $u_1 u_2 = 1$). Semiplanul superior corespunde punctelor clasificate pozitiv, iar semiplanul inferior corespunde punctelor clasificate negativ.

c. Se poate observa ușor că, dacă se consideră adițional instanța de antrenament $x_5 = (0.5, 4)$ clasificată negativ, mulțimea $\{\Phi(x_1), \Phi(x_2), \Phi(x_3), \Phi(x_4), \Phi(x_5)\}$ din „spațiul de trăsături“ este liniar neseparabilă.

d. Considerând $u = (u_1, u_2)$ și $u' = (u'_1, u'_2)$, urmează că

$$\Phi(u) \cdot \Phi(u') = (1, u_1, u_2, u_1 u_2) \cdot (1, u'_1, u'_2, u'_1 u'_2) = 1 + u_1 u'_1 + u_2 u'_2 + u_1 u_2 u'_1 u'_2 = K(u, u').$$

12.

(Un [alt] exemplu de rezolvare a problemei duale SVM)

CMU, 2009 fall, Carlos Guestrin, HW3, pr. 2.1.8

CMU, 2011 fall, T. Mitchell, A. Singh, HW6, pr. 2.1

Calculați soluția \bar{a} pentru forma duală a problemei de optimizare SVM pentru setul de date D și transformarea Φ (așadar, în spațiul de „trăsături“ determinat de Φ) de la problema 9.

Răspuns:

Putem rezolva această problemă fie (i.) în mod de sine stătător, adică rezolvând problema de optimizare în forma duală, făcând abstracție de rezolvarea problemei în forma primală (obținută la pr. 9), fie (ii.) folosind relațiile dintre soluțiile celor două forme ale problemei de optimizare SVM (fără însă a folosi efectiv lagrangeanul dual), fie (iii.) în mod combinat, adică rezolvând problema în forma duală după ce în prealabil am simplificat expresia lagrangeanului dual ținând cont de soluția problemei în forma primală.

i. Doar schițând calculele, veți vedea că această variantă de rezolvare nu este „fezabilă“ în mod analitic pe aceste date (deși, de exemplu pe datele de la problema 11 ea a funcționat) din cauza restricțiilor care însotesc multiplicatorii Lagrange.

Lagrangeanul dual este

$$\begin{aligned} L_D(\alpha) &= \sum_{i=1}^6 \alpha_i - \frac{1}{2} \sum_{i=1}^6 \sum_{j=1}^6 \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \\ &= \sum_{i=1}^6 \alpha_i - \frac{1}{2} (90\alpha_1^2 + 20\alpha_2^2 + 2\alpha_3^2 + 2\alpha_5^2 + 90\alpha_6^2 + \\ &\quad 2(42\alpha_1\alpha_2 - 12\alpha_1\alpha_3 - 6\alpha_1\alpha_5 + 72\alpha_1\alpha_6 \\ &\quad - 6\alpha_2\alpha_3 - 2\alpha_2\alpha_5 + 30\alpha_2\alpha_6 - 6\alpha_3\alpha_6 - 12\alpha_5\alpha_6)), \end{aligned}$$

iar condiția de optimalitate Karush-Kuhn-Tucker asociată este $\sum_{i=1}^6 y_i \alpha_i = 0$,²⁸⁹ adică $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 - \alpha_5 + \alpha_6 = 0$.

Pentru a găsi soluția problemei duale am putea încerca să rezolvăm sistemul obținut prin egalarea derivatelor parțiale ale lui $L_D(\alpha)$ cu 0, după care să verificăm dacă soluția $\bar{\alpha}$ satisfac restricțiile $\alpha_i \geq 0$ pentru $i = 1, \dots, 6$, precum și condiția de optimalitate Karush-Kuhn-Tucker.²⁹⁰

Vă puteți convinge, făcând calculele, că soluția acestui sistem $\bar{\alpha} = (\bar{\alpha}_1, \dots, \bar{\alpha}_6)$ nu satisfac restricțiile $\alpha_i \geq 0$. Prin urmare, soluția problemei duale ar trebui căutată altfel (folosind eventual o metodă numerică). Din fericire, vom vedea mai jos că metodele ii și iii funcționează (pe aceste date) foarte bine.²⁹¹

ii. Vom arăta că soluția $\bar{\alpha} = (\bar{\alpha}_1, \dots, \bar{\alpha}_6)$ a problemei duale se poate determina folosind relațiile dintre soluțiile celor două forme ale problemei SVM:

$$\bar{w} = \sum_{i=1}^6 \bar{\alpha}_i y_i \Phi(x_i) \quad \text{și} \quad \bar{\alpha}_i (y_i (\bar{w} \cdot \Phi(x_i) + \bar{w}_0) - 1) = 0 \text{ pentru } i = \overline{1, 6}.$$

De la rezolvarea problemei în forma primală (vedeti pr. 9) știm că $\bar{\alpha}_1 = \bar{\alpha}_4 = \bar{\alpha}_5 = \bar{\alpha}_6 = 0$, fiindcă vectorii-suport sunt (doar) instanțele x_2 și x_3 . În consecință, relația $\bar{w} = \sum_{i=1}^6 \bar{\alpha}_i y_i \Phi(x_i)$ devine:

²⁸⁹Vedeți relația (116) de la problema 10.

²⁹⁰Se poate observa că $\frac{\partial L}{\partial \alpha_4} = 1 \neq 0!$ De aceea este convenabil ca mai întâi să înlocuim în L_D variabila α_4 cu $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_5 + \alpha_6$ și apoi să rescriem sistemul de ecuații formate cu ajutorul derivatelor parțiale ale noului L_D .

²⁹¹De fapt, în general / practică, alternativa cea mai bună este aplicarea unei metode „numerice“ de optimizare, care să ia în calcul restricțiile de tip $\alpha_i \geq 0$ pentru $i = 1, \dots, m$. O astfel de metodă este algoritmul SMO (pentru rezolvarea problemei de optimizare C-SVM), a cărui aplicare o vom exemplifica la problema 23.

$$\begin{pmatrix} -1/5 \\ 3/5 \end{pmatrix} = \bar{\alpha}_2 \begin{pmatrix} -2 \\ 4 \end{pmatrix} - \alpha_3 \begin{pmatrix} -1 \\ 1 \end{pmatrix} \Leftrightarrow \begin{cases} -2\bar{\alpha}_2 + \bar{\alpha}_3 = -\frac{1}{5} \\ 4\bar{\alpha}_2 - \bar{\alpha}_3 = \frac{3}{5} \end{cases} \Leftrightarrow \bar{\alpha}_2 = \bar{\alpha}_3 = \frac{1}{5}.$$

Soluția astfel obținută, $\bar{\alpha} = \left(0, \frac{1}{5}, \frac{1}{5}, 0, 0, 0\right)$, verifică și restrictia Karush-Kuhn-Tucker $\sum_{i=1}^6 \bar{\alpha}_i y_i = 0$ din forma duală a problemei SVM.

În consecință, funcția de decizie obținută de SVM se poate exprima ca

$$\begin{aligned} y(x) &= \text{sign} \left(\sum_i \bar{\alpha}_i y_i \Phi(x_i) \cdot \Phi(x) + \bar{w}_0 \right) = \\ &= \text{sign} \left(\frac{1}{5} K(x, -2) - \frac{1}{5} K(x, -1) - \frac{9}{5} \right) = \\ &= \text{sign} (-2x + 4x^2 + x - x^2 - 9) = \text{sign} (3x^2 - x - 9). \end{aligned}$$

Așadar, regăsim rezultatul de la punctul *e* de la problema 9.

iii. Tinând cont de relațiile $\alpha_2 = \alpha_3 \stackrel{\text{not.}}{=} \alpha > 0$ și $\alpha_1 = \alpha_4 = \alpha_5 = \alpha_6 = 0$ care rezultă (ca o consecință) din soluția problemei în forma primală (așa cum am arătat la *ii*), urmează că putem scrie lagrangeanul dual astfel:

$$L_D(\alpha) = 2\alpha - \frac{1}{2}(20\alpha^2 + 2\alpha^2 - 12\alpha^2) = 2\alpha - 5\alpha^2 = \alpha(2 - 5\alpha).$$

Maximul funcției $L_D(\alpha)$ se obține pentru $\alpha = -\frac{2}{2 \cdot (-5)} = \frac{1}{5} > 0$. Așadar, regăsim rezultatul de la *ii*.

5.1.2 SVM cu margine “soft”

13. (Exercițiu teoretic: deducerea formei duale pentru problema de optimizare SVM cu margine “soft” (C-SVM))
prelucrare de Liviu Ciortuz, după CMU, 2012 spring, Ziv Bar-Joseph, HW3, pr. 3.2

Antrenăm o SVM pe un set de date de intrare $\{x_i\}$ cu $i = 1, \dots, n$, considerate împreună cu setul de etichete asociate $\{y_i\}$, cu $y_i \in \{-1, 1\}$. Se presupune că aceste date sunt neseparabile liniar. *Obiectivul* pe care ni-l propunem aici este să maximizăm *marginea* dar în același timp să permitem ca, la finalul antrenării, unele (în genere puține) dintre datele de antrenament să fie clasificate eronat.

Așadar, vom considera problema de optimizare

$$\begin{aligned} \min_{w, w_0, \xi} & \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \right) \\ \text{a. i. } & (w \cdot x_i + w_0) y_i \geq 1 - \xi_i, \text{ pentru } i = 1, \dots, m \\ & \xi_i \geq 0, \text{ pentru } i = 1, \dots, m, \end{aligned} \tag{P'}$$

unde $\xi \stackrel{not.}{=} (\xi_1, \dots, \xi_m)$, iar $C > 0$ este un parametru („cost“) care controlează compromisul (engl., trade-off) pe care urmărim să-l facem între mărimea *marginii* pe de o parte,²⁹² și *penalizările pentru „destindere“* (engl., slack penalty) reprezentate prin variabilele $\xi_i \geq 0$ pe de altă parte.

- Verificați faptul că pentru problema (P') este satisfăcută *condiția lui Slater*.²⁹³
- Folosind variabile duale (adică, multiplicatori Lagrange), scrieți expresia *lagrangeanului generalizat* care corespunde problemei (P') . Specificați pentru fiecare multiplicator în parte care este restricția care-i corespunde lui în problema (P') .
- Scrieți *condițiile de complementaritate Karush-Kuhn-Tucker* corespunzătoare problemei (P') .
- Calculând derivatele parțiale ale lagrangeanului generalizat $L_P(w, w_0, \xi, \alpha, \beta)$ în raport cu variabilele w, w_0 și respectiv ξ , arătați că forma duală a problemei (P') este:

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq C \text{ pentru } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned} \tag{D'}$$

Faceți toate calculele în mod detaliat. Analizați deosebirile dintre (D') și forma duală a problemei SVM cu margine “hard” (a se vedea definiția (D) de la problema 10, pag. 351).

- Dată fiind o soluție a problemei duale (D') , cum identificăm vectorii-suport?
- Cum va fi clasificată o instanță nouă x' ?

Răspuns:

- Condiția lui Slater pentru problema (P') se formulează astfel:

$$\exists w \in \mathbb{R}^d, w_0 \in \mathbb{R} \text{ și } \xi \in \mathbb{R}^m \text{ a. i. } (w \cdot x_i + w_0)y_i - 1 + \xi_i > 0 \text{ și } \xi_i > 0, \text{ pentru } i = 1, \dots, m.$$

Luând $w = 0$, $w_0 = 0$ și $\xi_i = 2$ pentru $i = 1, \dots, m$,²⁹⁴ se constată că această condiție se verifică imediat. În consecință, optimul problemei primale (P') va coincide cu optimul problemei duale (de la punctul d).

- Lagrangeanul generalizat care corespunde problemei de optimizare (P') este:

$$L_P(w, w_0, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i ((w \cdot x_i + w_0)y_i - 1 + \xi_i) - \sum_{i=1}^m \beta_i \xi_i.$$

Multiplicatorii $\alpha_i \geq 0$ corespund restricțiilor $(w \cdot x_i + w_0)y_i \geq 1 - \xi_i$, în vreme ce multiplicatorii $\beta_i \geq 0$ corespund restricțiilor $\xi_i \geq 0$.

- $\alpha_i((w \cdot x_i + w_0)y_i - 1 + \xi_i) = 0$ și $\beta_i \xi_i = 0$, pentru $i = 1, \dots, m$.

²⁹²Prin *margine* aici vom înțelege distanța dintre hiperplanul de separare optimală definit de soluția (w, w_0) a problemei (P') și oricare dintre instanțele x_i pentru care $y_i(w \cdot x_i + w_0) = 1$. Această distanță este egală cu $1/\|w\|$.

²⁹³Pentru formalizarea acestei condiții — ca și pentru diverse noțiuni implicate la punctele următoare —, a se vedea nota de subsolu 281 de la problema 10 (pag. 349), care tratează cazul formei duale a problemei SVM cu margine “hard”.

²⁹⁴De fapt, este suficient ca — pe lângă $w = 0$, $w_0 = 0$ — să considerăm $\xi_i > 1$ pentru $i = 1, \dots, m$.

d. Calculând derivatele parțiale indicate în enunț, vom avea:

$$\begin{aligned}\frac{\partial}{\partial w} L_P(w, w_0, \xi, \alpha, \beta) &= 0 \Leftrightarrow w - \sum_{i=1}^m \alpha_i y_i x_i = 0 \Leftrightarrow w = \sum_{i=1}^m \alpha_i y_i x_i \\ \frac{\partial}{\partial w_0} L_P(w, w_0, \xi, \alpha, \beta) &= 0 \Leftrightarrow -\sum_{i=1}^m \alpha_i y_i = 0 \Leftrightarrow \sum_{i=1}^m \alpha_i y_i = 0 \\ \frac{\partial}{\partial \xi_i} L_P(w, w_0, \xi, \alpha, \beta) &= 0 \Leftrightarrow C - \alpha_i - \beta_i = 0 \Leftrightarrow \alpha_i + \beta_i = C \text{ pentru } i = 1, \dots, m.\end{aligned}$$

Substituind primele două dintre aceste rezultate în expresia de definiție a lui L_P vom obține:

$$\begin{aligned}L_D(\alpha, \beta) &\stackrel{\text{def.}}{=} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + C \sum_{i=1}^m \xi_i - \\ &\quad - \sum_{i=1}^m \alpha_i \left(\left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right) \cdot x_i + w_0 \right) y_i - 1 + \xi_i \right) - \sum_{i=1}^m \beta_i \xi_i \\ &= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + C \sum_{i=1}^m \xi_i - \\ &\quad - \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - w_0 \underbrace{\sum_{i=1}^m \alpha_i y_i}_{0} + \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \alpha_i \xi_i - \sum_{i=1}^m \beta_i \xi_i \\ &= -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \alpha_i - \sum_{i=1}^m \underbrace{(\alpha_i + \beta_i)}_C \xi_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \stackrel{\text{not.}}{=} L_D(\alpha).\end{aligned}$$

Observați că la scrierea argumentelor lagrangeanului L_D , am renunțat în final la β , întrucât β_i (pentru $i = 1, \dots, m$) a fost eliminat în timpul efectuării calculului.

Tinând cont că $\beta_i = C - \alpha_i \geq 0$ implică $\alpha_i \leq C$, rezultă imediat că forma duală asociată problemei (P') este cea indicată în enunț.

De remarcat faptul că forma duală (D') pentru problema C-SVM (cu margine “soft”) diferă de forma duală (D) pentru problema SVM cu margine “hard” doar prin restricția suplimentară $\alpha_i \leq C$. Aceasta înseamnă că „importanță“ care revine fiecărei instanțe etichetate (x_i, y_i) cu privire la determinarea separatorului optimal devine limitată.

Facem aici mențiunea că legătura dintre soluțiile problemelor (P') și (D') este dată de relațiile care au fost obținute la punctele c și d : mai întâi

$$\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i, \tag{120}$$

iar apoi \bar{w}_0 se obține din relația $(\bar{w} \cdot x_i + \bar{w}_0)y_i = 1 - \bar{\xi}_i$ pentru un $i \in \{1, \dots, m\}$ astfel încât $\bar{\alpha}_i > 0$.²⁹⁵ Așadar,

²⁹⁵Vedeți condițiile de complementaritate Karush-Kuhn-Tucker: $\bar{\alpha}_i[(\bar{w} \cdot x_i + \bar{w}_0)y_i - 1 + \bar{\xi}_i] = 0$ și $\bar{\beta}_i \bar{\xi}_i = 0$ pentru $i = 1, \dots, m$. Dacă $\bar{\alpha}_i > 0$, atunci $(\bar{w} \cdot x_i + \bar{w}_0)y_i - 1 + \bar{\xi}_i = 0$, deci $(\bar{w} \cdot x_i + \bar{w}_0)y_i = 1 - \bar{\xi}_i$. Dacă $\bar{\alpha}_i < C$, atunci $\bar{\beta}_i = C - \bar{\alpha}_i > 0$ și deci $\bar{\xi}_i = 0$.

$$\bar{w}_0 = -\bar{w} \cdot x_i + y_i(1 - \bar{\xi}_i), \text{ cu } \bar{\xi}_i = 0 \text{ dacă } \bar{\alpha}_i < C. \quad (121)$$

Această ultimă egalitate este implicată de relația $\alpha_i + \beta_i = C$ dedusă mai sus și de condiția de complementaritate Karush-Kuhn-Tucker $\beta_i \xi_i = 0$.²⁹⁶

e. Fie $\bar{\alpha}$ o soluție a problemei (D'), iar \bar{w} și \bar{w}_0 stabiliți ca mai sus (vedeți relațiile (120) și (121)). Vectorii-suport sunt instanțele x_i pentru care $\bar{\alpha}_i > 0$. Mai sus (vedeți rezolvarea de la punctul d, la final) am arătat că $\bar{\alpha}_i \in (0, C) \Rightarrow \bar{\xi}_i = 0$, deci $y_i(\bar{w} \cdot x_i + \bar{w}_0) = 1$. Alternativ, pentru $\bar{\alpha}_i = C$ rezultă $\bar{\beta}_i = C - \bar{\alpha}_i = 0$, deci $\xi_i \geq 0$ și $y_i(\bar{w} \cdot x_i + \bar{w}_0) \leq 1$.²⁹⁷ Similar, pentru instanțele care nu sunt vectori-suport, $\bar{\alpha}_i = 0 \Rightarrow \bar{\beta}_i = C \Rightarrow \bar{\xi}_i = 0$, deci $y_i(\bar{w} \cdot x_i + \bar{w}_0) \geq 1$.

Observație: Cele trei relații deduse mai sus, rescrise sintetizat ca

$$\begin{cases} \bar{\alpha}_i \in (0, C) \Rightarrow y_i(\bar{w} \cdot x_i + \bar{w}_0) = 1 \\ \bar{\alpha}_i = C \Rightarrow y_i(\bar{w} \cdot x_i + \bar{w}_0) \leq 1 \\ \bar{\alpha}_i = 0 \Rightarrow y_i(\bar{w} \cdot x_i + \bar{w}_0) \geq 1 \end{cases} \quad (122)$$

se întâlnesc uneori în literatura de specialitate sub numele de *condiții Karush-Kuhn-Tucker*, dar acesta este impropriu. Ele sunt *condiții necesare* implicate de *condiții de optimalitate și complementaritate Karush-Kuhn-Tucker*. Însă ceea ce este important de reținut este faptul că relațiile de mai sus sunt folosite ca o *condiție de oprire* pentru algoritmul SMO,²⁹⁸ care rezolvă în practică problema de optimizare cu margine “soft” (desemnată în continuare prin C-SVM).

²⁹⁶Dacă $\bar{\alpha}_i = 0$ pentru $i = \overline{1, m}$, obținem $\bar{w} = 0$, ceea ce reprezintă o soluție inadmisibilă.

Rămâne de tratat cazul în care $\exists \bar{\alpha}_i = C$ și pentru orice alt $\bar{\alpha}_j$ avem fie $\bar{\alpha}_j = 0$ fie $\bar{\alpha}_j = C$. Este de notat mai întâi faptul că din relația $\sum_{i=1}^m \alpha_i y_i = 0$ va rezulta că jumătate din vectorii-suport sunt instanțe pozitive, iar restul (cealaltă jumătate) sunt instanțe negative. Apoi, $\bar{\alpha}_j = C > 0$ implică $y_j(\bar{w} \cdot x_j + w_0) = 1 - \bar{\xi}_j$ și $\bar{\beta}_j = 0$ și deci $\bar{\xi}_j \geq 0$. Pe de altă parte, $\bar{\alpha}_j = 0$ implică $\bar{\beta}_j = C$ și deci $\bar{\xi}_j = 0$, și $y_j(\bar{w} \cdot x_j + w_0) \geq 1$.

În consecință,

$$\begin{aligned} \sum_i \xi_i &= \sum_{i:\bar{\alpha}_i=C} \xi_i = \sum_{i:\bar{\alpha}_i=C} (1 - y_i(\bar{w} \cdot x_i + \bar{w}_0)) = \sum_{i:\bar{\alpha}_i=C} (1 - y_i \bar{w} \cdot x_i) \\ &= |\{i : \bar{\alpha}_i = C\}| - \sum_{i:\bar{\alpha}_i=C} y_i \bar{w} \cdot x_i = |\{i : \bar{\alpha}_i = C\}| - \bar{w} \cdot \sum_{i:\bar{\alpha}_i=C} y_i x_i \\ &= |\{i : \bar{\alpha}_i = C\}| - \frac{1}{C} \bar{w} \cdot \sum_{i:\bar{\alpha}_i=C} \bar{\alpha}_i y_i x_i = |\{i : \bar{\alpha}_i = C\}| - \frac{1}{C} \bar{w}^2 \end{aligned}$$

Aceasta ne arată că optimul problemei primale (P') depinde doar de \bar{w} (nu și de \bar{w}_0).

Notând în mod generic prin x_+ instanțele pozitive (și cu $\bar{\xi}_+$ valoarea variabilei de „destindere“ corespunzătoare), iar prin x_- instanțele negative (și, corespunzător, $\bar{\xi}_-$), vom avea de satisfăcut restricțiile $\bar{w} \cdot x_+ + \bar{w}_0 \geq 1 - \bar{\xi}_+$ și $\bar{w} \cdot x_- + \bar{w}_0 \leq -1 + \bar{\xi}_-$. Tinând cont că $\bar{\xi}_+ \geq 0$ și $\bar{\xi}_- \geq 0$, considerăm că din punct de vedere practic se poate lua pentru \bar{w}_0 valoarea $\frac{1}{2}(\max_{x_+}\{\bar{w} \cdot x_+\} - \min_{x_-}\{\bar{w} \cdot x_-\})$. De remarcat că într-o astfel de situație, marginea va reprezenta distanța dintre separatorul optimal și cele mai departăte instanțe pozitive / negative clasificate corect. Evident, aceasta este o situație (destul de) extremă în raport cu ideea cu care s-a plecat la drum în formalizarea clasificării cu margine maximală.

²⁹⁷Dacă $0 < \bar{\xi}_i \leq 1$, atunci instanța x_i este situată în interiorul „marginii“ de separare, iar dacă $\bar{\xi}_i > 1$, instanța x_i este clasificată eronat.

²⁹⁸Vedeți problemele 22 și 23.

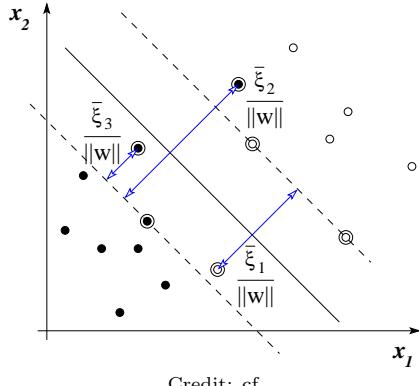
Rezumat: Pentru SVM cu margine “soft” (C-SVM), *vectorii-suport* sunt instanțele pentru care $\bar{\alpha}_i > 0$ (pentru acestea, avem $y_i(\bar{w} \cdot x_i + \bar{w}_0) \leq 1$), aşadar inclusiv instanțele pentru care $y_i(\bar{w} \cdot x_i + \bar{w}_0) < 1$ (deci cu $\bar{\alpha}_i = C$).

f. Păstrând notațiile de mai sus, instanța de test x' va fi clasificată pozitiv dacă $\bar{w} \cdot x' + \bar{w}_0 \geq 0$, și negativ în caz contrar.

14. (C-SVM: calculul distanței față de hiperplanul-margine corespunzător, pentru acei vectori-suport x_i pentru care $\bar{\alpha}_i = C$; calculul valorii optime pentru funcția obiectiv)

UAIC Iași, 2018 spring, Sebastian Ciobanu

- a. Folosind notațiile pentru forma primală și forma duală de la problema de optimizare C-SVM, — vedeti problema 13 — demonstrați următoarea afirmație: pentru toate acele instanțe de antrenament x_i pentru care $\bar{\alpha}_i = C$ (adică, pentru acei x_i pentru care, conform rezolvării problemei 13.e, $y_i(\bar{w} \cdot x_i + \bar{w}_0) \leq 1$ și, deci $\xi_i \geq 0$), *distanța geometrică* de la x_i la *hiperplanul-margine*, care este paralel cu hiperplanul de separare optimală și are ecuația $\bar{w} \cdot x + \bar{w}_0 = y_i$, este $\frac{\xi_i}{\|\bar{w}\|}$.



Credit: cf.

<http://efavdb.com/svm-classification/>.

Altfel spus,

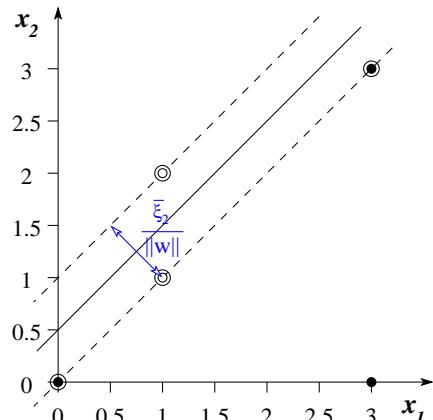
$$\bar{\alpha}_i = C \Rightarrow \frac{\xi_i}{\|\bar{w}\|} = d(x_i, \bar{w} \cdot x + \bar{w}_0 = y_i), \forall i \in \{1, \dots, m\}.$$

- b. Fie următorul set de date de antrenament:

i	1	2	3	4	5
x_i	(0,0)	(1,1)	(3,3)	(1,2)	(3,0)
y_i	+1	-1	+1	-1	+1

Valorile multiplicatorilor Lagrange învățate de către o C-SVM liniară pentru $C = 10$, pornind de la aceste date sunt:

i	1	2	3	4	5
α_i	8.66	10	5.33	3.99	0



- i. Verificați numeric relația $\frac{\xi_2}{\|\bar{w}\|} = d(x_2, \bar{w} \cdot x + \bar{w}_0 = y_2)$.

ii. Calculați valoarea funcției obiectiv de la problema primală C-SVM pentru același $C = 10$.

Sugestii:

1. Folosiți condițiile [de complementaritate] Karush-Kuhn-Tucker pentru problema de optimizare C-SVM, precum și relațiile de legătură dintre soluțiile formei primale și soluțiile formei duale ale problemei C-SVM. (Vedeți rezolvarea problemei 13, subpunctele c și d.)
2. Pentru ușurința calculelor, se pot face aproximări. (De exemplu, 1.99 poate fi aproxiimat cu 2.)

Răspuns:

a. Întrucât $\bar{\alpha}_i = C > 0$, din condiția de complementaritate Karush-Kuhn-Tucker $\bar{\alpha}_i[\bar{w} \cdot x_i + \bar{w}_0 - (1 - \bar{\xi}_i)]$ (vedeți rezolvarea problemei 13.c), rezultă

$$y_i(\bar{w} \cdot x_i + \bar{w}_0) = 1 - \bar{\xi}_i \Rightarrow \bar{w} \cdot x_i + \bar{w}_0 = y_i(1 - \bar{\xi}_i) \Rightarrow \bar{w} \cdot x_i + \bar{w}_0 - y_i = -y_i \bar{\xi}_i.$$

Prin urmare, ținând cont și de formula dată la problema 1, distanța de la punctul x_i la hiperplanul de ecuație $\bar{w} \cdot x_i + \bar{w}_0 - y_i = 0$ este

$$\frac{|-y_i \bar{\xi}_i|}{\|\bar{w}\|} \stackrel{\bar{\xi}_i \geq 0}{=} \frac{\bar{\xi}_i}{\|\bar{w}\|}.$$

b.i. Conform problemei 13.d, mai precis conform relației (120),

$$\bar{w} = \sum_{i=1}^5 \bar{\alpha}_i y_i x_i.$$

Deci, în cazul nostru,

$$\bar{w} = 8.66 \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 10 \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 5.33 \begin{pmatrix} 3 \\ 3 \end{pmatrix} - 3.99 \begin{pmatrix} 1 \\ 2 \end{pmatrix} + 0 = \begin{pmatrix} -10 + 15.99 - 3.99 \\ -10 + 15.99 - 7.98 \end{pmatrix} = \begin{pmatrix} 2 \\ -1.99 \end{pmatrix}.$$

Pentru ușurința calculelor vom aproxima:

$$\bar{w} \approx \begin{pmatrix} 2 \\ -2 \end{pmatrix}.$$

Așadar,

$$\|\bar{w}\| = \sqrt{2^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2\sqrt{2}.$$

Conform problemei 13.d, putem aproxima termenul liber \bar{w}_0 folosind ecuațiile / proprietățile corespunzătoare unui vector-suport x_i pentru care $\bar{\xi}_i = 0$:

$$\bar{\alpha}_1 = 8.66 \in (0, C) \Rightarrow \bar{\beta}_1 > 0 \stackrel{\text{KKT}}{\Rightarrow} \bar{\xi}_1 = 0 \Rightarrow x_1 \text{ este vector-suport.}$$

$$y_1(\bar{w} \cdot x_1 + \bar{w}_0) = 1 - \bar{\xi}_1 \Rightarrow (+1) \left(\begin{pmatrix} 2 \\ -2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \bar{w}_0 \right) = 1 - 0 \Rightarrow \bar{w}_0 = 1.$$

Acum putem calcula $\bar{\xi}_2$ folosind prima parte a raționamentului de la punctul a:

$$\begin{aligned} \bar{\alpha}_2 = 10 = C \Rightarrow y_2(\bar{w} \cdot x_2 + \bar{w}_0) &= 1 - \bar{\xi}_2 \\ \Rightarrow \bar{\xi}_2 = 1 - y_2(\bar{w} \cdot x_2 + \bar{w}_0) &= 1 - (-1) \left(\begin{pmatrix} 2 \\ -2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 1 \right) = 2. \end{aligned}$$

În consecință,

$$\begin{aligned} d(x_2, \bar{w} \cdot x + \bar{w}_0 = y_2) &= d\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix} \cdot x + 1 = -1\right) \\ &= \frac{\left| \begin{pmatrix} 2 \\ -2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 1 - (-1) \right|}{\|\bar{w}\|} = \frac{|2|}{2\sqrt{2}} = \frac{1}{\sqrt{2}}. \end{aligned}$$

Așadar, se verifică egalitatea

$$\frac{\bar{\xi}_2}{\|\bar{w}\|} = d(x_2, \bar{w} \cdot x + \bar{w}_0 = y_2).$$

Observație: Se poate verifica și din punct de vedere pur geometric, lucrând pe figura din enunț, că distanța dintre punctul x_2 și hiperplanul $\bar{w} \cdot x + \bar{w}_0 = y_2$ este de (aproximativ²⁹⁹) $1/\sqrt{2}$.

b.ii. Trebuie să calculăm valoarea

$$\frac{1}{2}\|\bar{w}\|^2 + C \sum_{i=1}^5 \bar{\xi}_i.$$

De la subpunctul precedent știm că $\|\bar{w}\| \approx 2\sqrt{2}$ și $\bar{\xi}_2 \approx 2$. Pentru a determina celelalte valori ale variabilelor de „destindere” $\bar{\xi}_i$, vom face din nou apel la *condițiile Karush-Kuhn-Tucker*:

$$0 \leq \bar{\alpha}_1, \bar{\alpha}_3, \bar{\alpha}_4, \bar{\alpha}_5 < C \xrightarrow{\alpha_i + \beta_i = C} 0 < \bar{\beta}_1, \bar{\beta}_3, \bar{\beta}_4, \bar{\beta}_5 \leq C \xrightarrow{\beta_i \xi_i = 0} \bar{\xi}_1, \bar{\xi}_3, \bar{\xi}_4, \bar{\xi}_5 = 0.$$

În consecință,

$$\frac{1}{2}\|\bar{w}\|^2 + C \sum_{i=1}^5 \bar{\xi}_i \approx \frac{1}{2}(2\sqrt{2})^2 + 10 \cdot (0 + 2 + 0 + 0 + 0) = \frac{1}{2} \cdot 8 + 10 \cdot 2 = 24.$$

15. (SVM și C-SVM: o proprietate simplă a soluției (\bar{w}) , indusă de o caracteristică particulară a datelor de antrenament)

*prelucrare de Liviu Ciortuz, după
MIT, 2008 fall, Tommi Jaakkola, midterm exam, pr. 1.1
CMU, 2010 fall, Aarti Singh, HW3, pr. 3.3.a*

Considerăm instanțele de antrenament $x_1 = (1, 1)$, $x_2 = (2, 2)$, $x_3 = (-1.5, -1.5)$ și $x_4 = (4, 4)$. Antrenăm o mașină cu vectori-suport pe aceste date.

Arătați că

- indiferent care este etichetarea celor patru instanțe de antrenament, și
- indiferent dacă mașina cu vectori-suport lucrează cu margine “hard” ori cu margine “soft”,

²⁹⁹Din cauza „rotunjirii“ aplicate mai sus asupra lui \bar{w} .

vectorul $\bar{w} = (\bar{w}_1, \bar{w}_2)$ care constituie soluția problemei de optimizare [C]-SVM pe aceste date are următoarea proprietate: $\bar{w}_1 = \bar{w}_2$.

Răspuns:

Stim că soluția problemei [C]-SVM în forma primală este $\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i$, unde $\bar{\alpha}_i$, cu $i = 1, \dots, m$, constituie soluția problemei în forma duală.³⁰⁰ Întrucât toate instanțele de antrenament au proprietatea $x_{i1} = x_{i2}$, rezultă imediat că $\bar{w}_1 = \bar{w}_2$.

Observație: Exercițiul acesta a ilustrat de fapt următoarea proprietate importantă: dacă în setul de date de antrenament două trăsături (engl., features) sunt duplicate ($x_{ij} = x_{ik}$ pentru $i = 1, \dots, m$), atunci ele vor primi ponderi identice ($\bar{w}_j = \bar{w}_k$) în soluția optimală calculată de clasificatorul [C]-SVM.

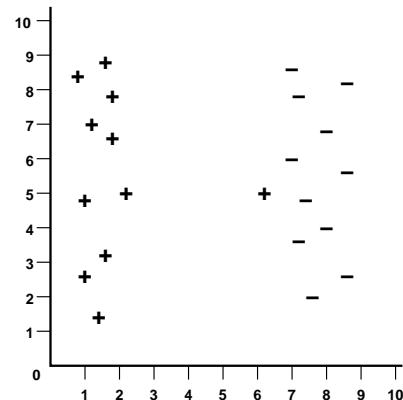
16.

(C-SVM, cazul [datelor separabile] liniar:
aplicare în prezența unui “outlier”)

CMU, 2005 spring, C. Guestrin, T. Mitchell, HW3, pr. 2.2

Se dau datele prezentate în figura alăturată. În cadrul cu clasificatorul C-SVM, parametrul C — pentru penalizarea aferentă „destinderii“; engl., slack penalty — va determina poziția hiperplanului de separare. Presupunem că se lucrează în varianta liniară, adică fără funcții-nucleu. Răspundeți succint, în manieră *calitativă*, la întrebările de mai jos.

- Unde va fi situat separatorul optimal în cazul în care C ia valori foarte mari (adică, $C \rightarrow \infty$)? Indicați pe figura dată.
- Pentru $C \approx 0$, indicați pe figura dată unde va fi situat separatorul optimal.
- Care din cele două situații de mai sus credeți că este mai adecvată pentru clasificare? De ce?



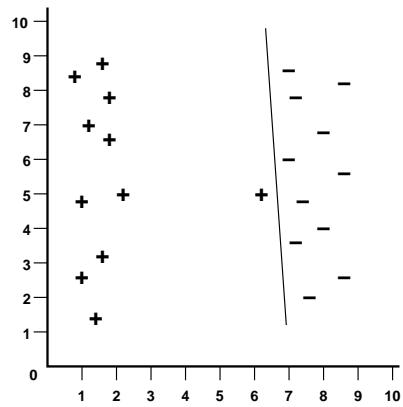
Răspuns:

Putem face mai întâi două observații în legătură cu acest set de date de antrenament:

- grupul de date clasificate negativ, precum și grupul de date clasificate pozitiv (din partea stângă) sunt foarte compacte;
- există un singur exemplu pozitiv care este foarte apropiat de grupul exemplelor negative. Acest exemplu-excepție constituie o „anomalie“ (engl., outlier).

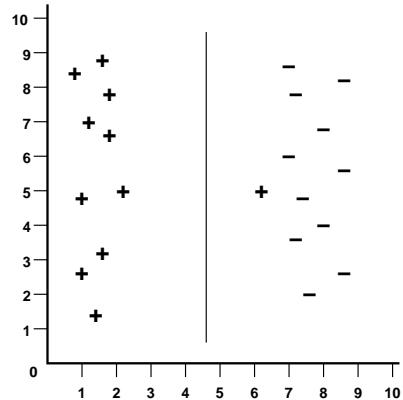
³⁰⁰Vedeți relațiile (115) și (120) deduse la problemele 10 și respectiv 13.

- a. O valoare mare a parametrului C semnifică o penalizare mai mare a erorilor de clasificare ξ_i . Așadar, pentru valori ale lui C tînzând la $+\infty$ hiperplanul de separare optimală va corespunde celei mai mici valori posibile pentru suma $\sum_i \xi_i$, care reprezintă eroarea totală în raport cu marginile. Întrucât exemplele date sunt separabile liniar, separatorul optimal învățat este cel reprezentat în figura alăturată. Poziția sa este foarte mult influențată de poziția outlierului (+), care este situat în imediata proximitate a instanțelor negative.



Observație: În general, dacă avem încredere în corectitudinea datelor de antrenament, putem alege $C \rightarrow \infty$ pentru a obține o separare cât mai precisă a celor două clase.

- b. În cazul $C \approx 0$, erorile ξ_i sunt penalizate foarte puțin. În consecință, separatorul optimal tinde să maximizeze marginea — adică distanța de la hiperplanul de separare $w \cdot x + w_0 = 0$ la vectorii-suport x_i pentru care $y_i(w \cdot x_i + w_0) = 1$ —, chiar dacă asta înseamnă că unele instanțe vor fi clasificate eronat. Separatorul determinat în acest caz va fi poziționat ca în figura alăturată.



Observație: În general, este indicat să alegem o valoare mică pentru parametrul C atunci când datele de antrenament pe care le avem sunt afectate de „zgomote“ / perturbații / anomalii. Într-o astfel de situație am putea avea de-a face cu un set de date neseparabile liniar, în care câteva exemple pozitive să fie amestecate printre cel negative și invers.

- c. Separatorul determinat la punctul b este alegerea cea mai bună pentru generalizare / testare în prezența outlier-elor.

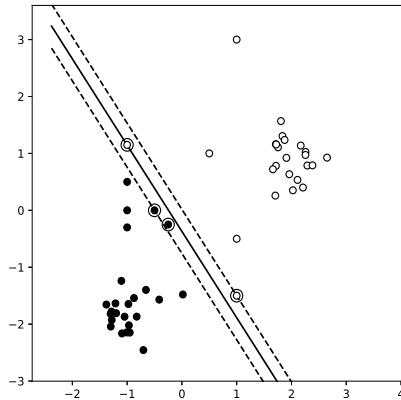
17.

(C-SVM, cazul [datelor separabile] liniar:
efectul alegerii diverselor valori pentru parametrul C)
CMU, 2010 fall, Ziv Bar-Joseph, midterm exam, pr. 7.c

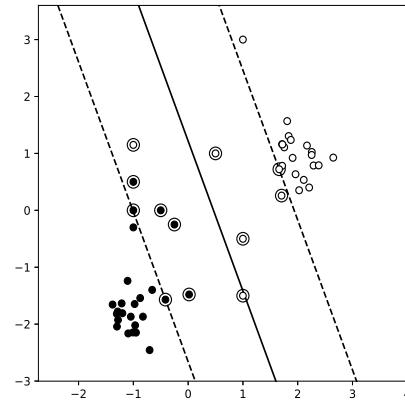
În următoarele patru figuri, se consideră că s-a aplicat o C-SVM liniară — adică o mașină cu vectori-suport cu margine “soft” și cu parametru de „destindere“ C , fără funcție-nucleu — pe un set de date din \mathbb{R}^2 , folosind de fiecare dată una din următoarele valori pentru parametrul C : 0.1, 1, 10 sau 100.

Sub fiecare figură scrieți valoarea corespunzătoare pentru parametrul C .

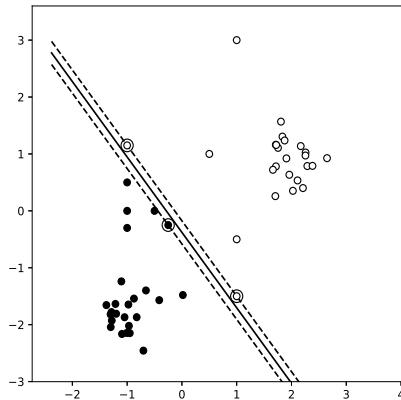
Justificați alegerea pe care ati făcut-o.



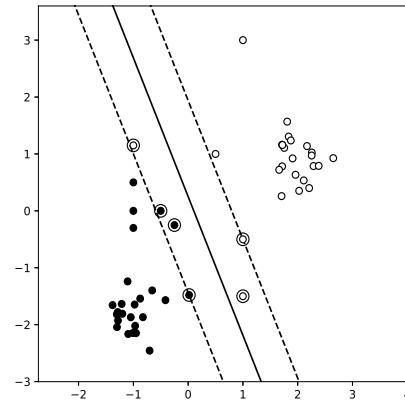
A.



B.



C.



D.

Răspuns:

Considerând datele de antrenament (x_i, y_i) cu $i = \overline{1, m}$, funcția obiectiv pentru forma primală a problemei C-SVM este

$$\min_{w \in \mathbb{R}^d, \xi} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \right),$$

unde $\xi_i \geq 0$ sunt variabile de „destindere” ($y_i(w \cdot x_i + w_0) \geq 1 - \xi_i$ pentru $i = 1, \dots, m$). Din expresia funcției obiectiv decurge imediat că la valori mari ale parametrului C — vă reamintim, $C > 0$ — sunt permise eventuale erori ξ_i mici, și invers: la valori mici ale parametrului C sunt permise erori ξ_i mari. De asemenea, ne așteptăm ca, pe măsură ce valoarea parametrului C crește, marginea³⁰¹ să scadă.

³⁰¹Definită ca $1/\|w\|$, adică distanța de la hiperplanul de separare optimală la instanțele (vectorii-suport) x_i pentru care $(w \cdot x_i + w_0)y_i = 1$.

Dintre cele patru grafice din enunț, în cazul celui de-al treilea (C) avem eroare la antrenare 0; în toate celelalte trei cazuri avem în mod evident $\sum_i \xi_i > 0$.³⁰² Așadar, graficul C corespunde celei mai mari valori dintre cele patru valori propuse pentru parametrul C : 100.

În graficele A și D există câte o singură eroare de clasificare. În cazul graficului B sunt mult mai mulți vectori-suport decât în cazul graficelor A sau D, iar suma $\sum_i \xi_i$ este cea mai mare dintre toate aceste trei cazuri rămase în discuție (A, B și D). Așadar, graficul B îi corespunde cea mai mică dintre valorile lui C rămase: 0.1. În fine, marginea este mai mare în cazul D decât în cazul A. Prin urmare, lui A îi corespunde $C = 10$, iar lui D valoarea rămasă: $C = 1$.

18. (C-SVM: un set simplu de date pentru care forma duală a problemei de optimizare C-SVM are soluție unică, dar forma primală nu are soluție unică)

S. Ciobanu, L. Ciortuz, 2019, după “Uniqueness of the SVM solution”, C. Burges, D. Crisp, 1998

Fie instanțele de antrenament $(x_1 = -1, y_1 = -1)$ și $(x_2 = 1, y_2 = 1)$. Vrem să antrenăm o mașină cu vectori-suport cu margine “soft” (deci, C-SVM) pe acest set de date. Să se arate că



- în cazul $C \geq \frac{1}{2}$, soluția problemei de optimizare C-SVM este [unică, și anume]: $\bar{w} = 1$, $\bar{\xi}_1 = \bar{\xi}_2 = \bar{w}_0 = 0$.
- în cazul $C < \frac{1}{2}$, rezultă că $\bar{w} = 2C$, $\bar{\xi}_1 = 1 - 2C + \bar{w}_0$, $\bar{\xi}_2 = 1 - 2C - \bar{w}_0$ și orice(!) $\bar{w}_0 \in [2C - 1, 1 - 2C]$ este soluție a formei primale pentru problema de optimizare C-SVM. Așadar, în acest caz soluția problemei de optimizare C-SVM (în forma primală) nu este unică!

Observație: Pentru ambele cazuri, soluția problemei de optimizare C-SVM în forma duală este unică.

Răspuns: Conform restricției $\sum_i y_i \bar{\alpha}_i = 0$ din problema duală C-SVM (vedeți ex. 13.d), rezultă $\bar{\alpha}_1 = \bar{\alpha}_2$ și, în consecință, $\bar{w} = \sum_i y_i \alpha_i x_i = 2\bar{\alpha}_1$ (conform relației (120)).

Datorită egalității $\alpha_1 = \alpha_2$, funcția obiectiv din problema duală, $L_D(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j$, devine:

$$\begin{aligned} L_D(\alpha_1, \alpha_2) &= \alpha_1 + \alpha_2 - \frac{1}{2} [\alpha_1^2 (-1)^2 (-1)^2 + \alpha_2^2 \cdot 1^2 \cdot 1^2 + 2\alpha_1 \alpha_2 (-1) \cdot 1 \cdot (-1) \cdot 1] \\ &= 2\alpha_1 - \frac{1}{2} (2\alpha_1^2 + 2\alpha_1^2) = 2\alpha_1 - 2\alpha_1^2 = 2\alpha_1(1 - \alpha_1). \end{aligned}$$

Evident, maximul acestei funcții de gradul al doilea se atinge pentru $\alpha_1 = 1/2$.

Soluțiile optime pentru problema duală C-SVM sunt supuse restricțiilor $0 \leq \bar{\alpha}_i \leq C$. Așadar, vom avea două cazuri: $\bar{\alpha}_1 = \bar{\alpha}_2 = 1/2$ atunci când $C \geq 1/2$ și, respectiv, $\bar{\alpha}_1 = \bar{\alpha}_2 = C$ atunci când $C < 1/2$.

³⁰²Inegalitatea $\sum_i \xi_i > 0$ se datorează (cel puțin!) faptului că avem un “outlier” în fiecare dintre aceste ultime trei cazuri.

Cazul 1 ($C \geq 1/2$): $\bar{\alpha}_1 = \bar{\alpha}_2 = 1/2$, $\bar{w} = 2\bar{\alpha}_1 = 1$.

Va trebui să mai calculăm $\bar{\xi}_1$, $\bar{\xi}_2$ și \bar{w}_0 . Analizând condițiile de tip Karush-Kuhn-Tucker (122), constatăm că pentru cazul acesta ($C \geq 1/2$) este convenabil să considerăm două subcazuri: $C > 1/2$ și, respectiv, $C = 1/2$, întrucât calculul valorilor $\bar{\xi}_i$ se va face diferit pentru cele două subcazuri.

Cazul 1.1 ($C > 1/2$): $\bar{\xi}_1 = \bar{\xi}_2 = 0$, fiindcă *i.* valorile variabilelor lagrangeene $\bar{\beta}_1 = C - \bar{\alpha}_1$ și $\bar{\beta}_2 = C - \bar{\alpha}_2$ sunt strict pozitive (vedeți relația $\alpha_i + \beta_i = C$ din ex. 13.d) și *ii.* avem condiția de complementaritate Karush-Kuhn-Tucker, $\bar{\beta}_i \bar{\xi}_i = 0$ pentru $i \in \{1, 2\}$. Putem calcula acum \bar{w}_0 :

$$y_1(\bar{w} \cdot x_1 + \bar{w}_0) = 1 \Rightarrow -(-1 + \bar{w}_0) = 1 \Rightarrow \bar{w}_0 = 0.$$

Cazul 1.2 ($C = 1/2$): Calculăm $\bar{\xi}_i$ folosind celelalte relații de complementaritate Karush-Kuhn-Tucker:

$$\begin{cases} \bar{\alpha}_1[y_1(\bar{w} \cdot x_1 + \bar{w}_0) - (1 - \bar{\xi}_1)] = 0 & \bar{\alpha}_i \neq 0 \\ \bar{\alpha}_2[y_2(\bar{w} \cdot x_2 + \bar{w}_0) - (1 - \bar{\xi}_2)] = 0 & \end{cases} \Rightarrow \begin{cases} y_1(\bar{w} \cdot x_1 + \bar{w}_0) = 1 - \bar{\xi}_1 \\ y_2(\bar{w} \cdot x_2 + \bar{w}_0) = 1 - \bar{\xi}_2 \end{cases}$$

$$\Rightarrow \begin{cases} -1(-1 + \bar{w}_0) = 1 - \bar{\xi}_1 \\ 1 + \bar{w}_0 = 1 - \bar{\xi}_2 \end{cases} \Rightarrow \bar{w}_0 = \bar{\xi}_1 = -\bar{\xi}_2.$$

Tinând cont de faptul că $\bar{\xi}_1 \geq 0$ și $\bar{\xi}_2 \geq 0$, rezultă $\bar{w}_0 = \bar{\xi}_1 = \bar{\xi}_2 = 0$.

Așadar, în ambele subcazuri a rezultat aceeași soluție ($\bar{w} = 1$, $\bar{w}_0 = 0$), iar separatorul optimal are ecuația $\bar{w} \cdot x + \bar{w}_0 = 0$, deci $x = 0$.

Cazul 2 ($C < 1/2$): $\bar{\alpha}_1 = \bar{\alpha}_2 = C$, $\bar{w} = 2\bar{\alpha}_1 = 2C$.

Aplicând (ca și mai sus) relațiile de complementaritate Karush-Kuhn-Tucker, vom obține:

$$\begin{cases} \bar{\alpha}_1[y_1(\bar{w} \cdot x_1 + \bar{w}_0) - (1 - \bar{\xi}_1)] = 0 & \bar{\alpha}_i \neq 0 \\ \bar{\alpha}_2[y_2(\bar{w} \cdot x_2 + \bar{w}_0) - (1 - \bar{\xi}_2)] = 0 & \end{cases} \Rightarrow \begin{cases} y_1(\bar{w} \cdot x_1 + \bar{w}_0) = 1 - \bar{\xi}_1 \\ y_2(\bar{w} \cdot x_2 + \bar{w}_0) = 1 - \bar{\xi}_2 \end{cases}$$

$$\Rightarrow \begin{cases} -(-2C + \bar{w}_0) = 1 - \bar{\xi}_1 \\ 2C + \bar{w}_0 = 1 - \bar{\xi}_2 \end{cases} \Rightarrow \begin{cases} \bar{\xi}_1 = 1 - 2C + \bar{w}_0 \\ \bar{\xi}_2 = 1 - 2C - \bar{w}_0 \end{cases}$$

Ultimul sistem obținut mai sus este un sistem liniar de două ecuații cu trei necunoscute. Soluțiile $\bar{\xi}_1$ și $\bar{\xi}_2$ se scriu în funcție de \bar{w}_0 . În final, tinând cont de restricțiile $\bar{\xi}_1 \geq 0$ și $\bar{\xi}_2 \geq 0$, rezultă $\bar{w}_0 \geq 2C - 1$ și $\bar{w}_0 \leq 1 - 2C$, deci în concluzie $\bar{w}_0 \in [2C - 1, 1 - 2C]$.³⁰³

Prin urmare, în acest caz, forma primală a problemei C-SVM are o infinitate de soluții — deși forma duală are o singură soluție —, și anume câte una pentru fiecare valoare posibilă a lui \bar{w}_0 în intervalul $[2C - 1, 1 - 2C]$. Ecuația separatorului optimal este $\bar{w} \cdot x + \bar{w}_0 = 0 \Rightarrow 2Cx + \bar{w}_0 = 0 \Rightarrow x = -\bar{w}_0/2C$.³⁰⁴

³⁰³Observați că $2C - 1 < 0$, iar $1 - 2C > 0$.

³⁰⁴Observați că

$$\bar{w}_0 \in [2C - 1, 1 - 2C] \Rightarrow -\bar{w}_0 \in [2C - 1, 1 - 2C] \Rightarrow -\bar{w}_0/2C \in \left[-\frac{1}{2C} + 1, \frac{1}{2C} - 1 \right],$$

$$\text{iar } 0 < C < 1/2 \Rightarrow 0 < 2C < 1 \Rightarrow \frac{1}{2C} > 1 \Rightarrow -\frac{1}{2C} < -1 \Rightarrow -\frac{1}{2C} + 1 < 0, \text{ iar } \frac{1}{2C} - 1 > 0.$$

Remarcăți faptul că separatorul optimal $(-\bar{w}_0/2C)$ se poate afla oriunde pe axa reală dacă parametrul C este lăsat să varieze în intervalul $(0, 1/2)$!

19.

(C-SVM: o margine superioară pentru numărul de erori la antrenare)

CMU, 2017 fall, Nina Balcan, HW4, pr. 4.Q12

Fie $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ un set de m instanțe etichetate din \mathbb{R}^d , cu etichete din mulțimea $\{1, -1\}$. Vă readucem aminte că forma primală a problemei C-SVM este

$$\min_{w, w_0, \xi} \left(\|w\|^2 + C \sum_{i=1}^n \xi_i \right),$$

a. i. $\forall i, y_i(w \cdot x_i + w_0) \geq 1 - \xi_i$, cu $\xi_i \geq 0$.

Fie $\bar{w}, \bar{w}_0, \bar{\xi}$ soluția acestei probleme. Stabilită o margine superioară (engl., upper bound) pentru numărul de instanțe din S care sunt clasificate în mod eronat la antrenare, folosind ξ_i pentru $i = 1, \dots, m$.

Răspuns:

Tinând cont de semnificația [analitică a] variabilelor de destindere ξ_i , rezultă că o instanță x_i este clasificată eronat la antrenare dacă $\bar{\xi}_i > 1$. Așadar, numărul total de erori comise de C-SVM la antrenare poate fi scris ca $\sum_i 1_{\{\bar{\xi}_i > 1\}}$, unde am folosit variabila-indicator

$$1_{\{\bar{\xi}_i > 1\}} = \begin{cases} 1 & \text{dacă } \bar{\xi}_i > 1, \\ 0 & \text{în caz contrar.} \end{cases}$$

Prin urmare,

$$\sum_i 1_{\{\bar{\xi}_i > 1\}} < \sum_{i: \bar{\xi}_i > 1} \bar{\xi}_i \leq \sum_i \bar{\xi}_i.$$

Dacă definim eroarea comisă de C-SVM la antrenare ca raportul dintre numărul de erori și m , numărul total de instanțe de antrenament, atunci marginea superioară cerută este $\frac{1}{m} \sum_i \bar{\xi}_i$.

20.

(C-SVM: o margine superioară pentru eroarea de tip CVLOO)

prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Aarti Singh, midterm exam, pr. 5.2

Considerăm un C-SVM, adică o mașină cu vectori-suport cu margine "soft", care folosește parametrul de „destindere” C . Arătați că — pentru orice valoare a lui C fixată (în mod arbitrar) — eroarea produsă de acest clasificator la cross-validation cu metoda "Leave-One-Out" este mai mică sau egală cu

$$\frac{\#\text{SVs}}{m},$$

unde m este numărul exemplelor de antrenament, iar $\#\text{SVs}$ este numărul de vectori-suport obținuți (pentru respectiva valoare a lui C) la antrenarea acestui clasificator pe întreg setul de exemple.

Observații:

1. Vom demonstra în mod riguros (i.e., analitic) următoarea proprietate importantă: La

CVLOO cu C-SVM, numai vectorii-suport pot fi (eventual!) clasificați eronat. Altfel spus, orice instanță x_k care nu este vector-suport nu produce eroare la CVLOO.

2. Vă readucem aminte că în contextul clasificatorului C-SVM, vectorii-suport sunt acei x_i pentru care $\bar{\alpha}_i > 0$. Așadar, ei sunt fie acei x_i pentru care $y_i(\bar{\alpha}_i \cdot x_i + \bar{w}_0) = 1$ (pentru aceștia, $\bar{\alpha}_i \in (0, C)$), fie acei x_i pentru care $y_i(\bar{\alpha}_i \cdot x_i + \bar{w}_0) < 1$ (pentru aceștia, $\bar{\alpha}_i = C$; ei constituie erori la antrenare dacă $\xi_i > 1$). Vedeți *Rezumatul* de la rezolvarea problemei 13.e, pag. 361.

3. Se poate demonstra că rezultatul din enunț este valabil și pentru SVM (adică mașina cu vectori-suport cu margine “hard”).

4. Eroarea de tip CVLOO este un bun indicator pentru capacitatea de generalizare a unui clasificator automat. Numărul vectorilor-suport fiind în general relativ mic în raport cu numărul total de instanțe de antrenament, marginea superioară indicată mai sus pentru eroarea CVLOO produsă de C-SVM „atestă“ într-un anumit sens calitatea acestui clasificator.

Răspuns:

Folosind notațiile de la problema 13, vom considera problemele de optimizare

$$\begin{aligned} \min_{w, w_0, \xi} & \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \right) \\ \text{a. i. } & (w \cdot x_i + w_0) y_i \geq 1 - \xi_i, \text{ pentru } i = 1, \dots, m \\ & \xi_i \geq 0, \text{ pentru } i = 1, \dots, m \end{aligned} \tag{P'}$$

și

$$\begin{aligned} \min_{w, w_0, \xi} & \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1, i \neq k}^m \xi_i \right) \\ \text{a. i. } & (w \cdot x_i + w_0) y_i \geq 1 - \xi_i, \text{ pentru } i = 1, \dots, m, i \neq k \\ & \xi_i \geq 0, \text{ pentru } i = 1, \dots, m, i \neq k. \end{aligned} \tag{P'_k}$$

unde $k \in \{1, \dots, m\}$.

Stim (tot de la problema 13) că problemele (P') și (P'_k) sunt în relație de *dualitate tare* cu cu dualele (D') și respectiv (D'_k) :³⁰⁵

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq C \text{ pentru } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \tag{D'}$$

și

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1, i \neq k}^m \alpha_i - \frac{1}{2} \sum_{i,j, i \neq k} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq C \text{ pentru } i = 1, \dots, m, i \neq k \\ & \sum_{i=1, i \neq k}^m \alpha_i y_i = 0. \end{aligned} \tag{D'_k}$$

În plus, relația dintre soluțiile problemelor (P') și (D') este

$$\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i,$$

³⁰⁵Pentru definiția dualității tari, vedeți nota de subsol 281, pag. 349.

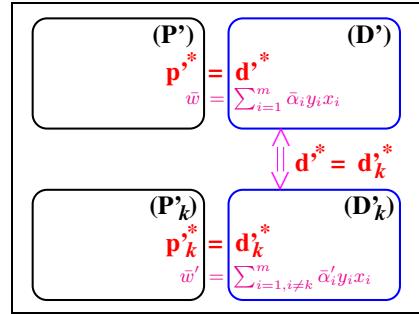
iar \bar{w}_0 se obține din relația $(\bar{w} \cdot x_i + \bar{w}_0)y_i = 1 - \bar{\xi}_i$ pentru un $i \in \{1, \dots, m\}$ cu proprietatea $\bar{\alpha}_i > 0$. Relația dintre soluțiile problemelor (P'_k) și (D'_k) se exprimă în mod similar.

Dacă în urma rezolvării problemei (P') , instanța x_k nu este vector-suport, adică $\bar{\alpha}_k = 0$, atunci urmează că $\bar{\beta}_k = C - \bar{\alpha}_k = C \neq 0$, deci din condiția de complementaritate Karush-Kuhn-Tucker $\bar{\beta}_k \bar{\xi}_k = 0$ rezultă $\bar{\xi}_k = 0$ și, prin urmare

$$y_k(\bar{w} \cdot x_k + \bar{w}_0) \geq 1 - \bar{\xi}_k = 1. \quad (123)$$

Lucrând încă odată cu presupoziția că x_k nu este vector-suport, se poate arăta — vedetă mai jos — că soluțiile optime (\bar{w}, \bar{w}_0) pentru problemele (P') și (P'_k) sunt aceleași. În consecință, folosind relația (123), va rezulta că x_k este corect clasificat la CVLOO.

Demonstrația faptului că soluțiile optime ale problemelor (P') și (P'_k) sunt aceleași se poate face astfel:



- se ține cont de relațiile de dualitate tare menționate mai sus, desemnate prin notațiile $(P') \equiv (D')$ și $(P'_k) \equiv (D'_k)$. Referindu-ne la valorile optime ale funcțiilor-obiectiv respective, putem scrie — folosind notații inspirate de proprietatea de dualitate tare — $p'^* = d'^*$ și $p'_k^* = d'_k^*$;
- se arată că soluția optimă pentru problema (D') coincide cu soluția optimă pentru problema (D'_k) , cu singura „diferență“ că $\bar{\alpha}_k = 0$. (Așadar, $d'^* = d'_k^*$.) Pe scurt, putem justifica aceasta în modul următor:

Valoarea optimă pentru funcția obiectiv a problemei (D') este mai mare sau cel puțin egală cu valoarea optimă pentru funcția obiectiv a problemei (D'_k) , pentru că spațiul de *valori admisibile* pentru variabilele α_i (i.e., valorile care satisfac restricțiile) este mai amplu. Mai departe, știind că $\bar{\alpha}_k = 0$, rezultă că valorile optime ale celor două probleme coincid, iar soluția optimă $\bar{\alpha}$ pentru problema (D') este (abstracție făcând de $\bar{\alpha}_k$) și soluție optimă a problemei (D'_k) ;³⁰⁶

³⁰⁶În manieră riguroasă, considerând (în principal de dragul simplificării exprimării) problema

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq C \text{ pentru } i = 1, \dots, m \text{ cu } i \neq k \text{ și } \alpha_k = 0 \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (D'')$$

veți observa mai întâi că problemele (D') și (D'') sunt *echivalente* — adică orice soluție optimă a lui (D') este și soluție optimă a lui (D'') — și apoi că problemele (D'') și (D'_k) , deși diferă din punct de vedere „sintactic“, sunt practic identice, întrucât pe de o parte α_k (componenta k a vectorului generic α) este 0 în problema (D') și lipsește în problema (D'_k) , iar pe de altă parte substituind $\alpha_k = 0$ în funcția obiectiv a problemei (D') obținem funcția obiectiv a problemei (D'_k) .

Pentru a demonstra că problemele (D') și (D'') sunt echivalente, observați mai întâi că orice soluție admisibilă a problemei (D') este și soluție admisibilă pentru problema (D'') . Așadar, mulțimea de soluții admisibile pentru problema (D') este mai amplă decât mulțimea de soluții admisibile pentru problema (D'') . Notăm cu F funcția obiectiv a problemei (D') , adică $F(\alpha) =$

- se utilizează în final relațiile de legătură de forma $\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i$ (cu \bar{w}_0 determinat corespunzător).

Observație: Demonstrația proprietății din enunț pentru SVM [cu margine “hard”] se poate face fie urmând „tipicul“ demonstrației de mai sus — adică, făcând apel la legătura dintre forma primală și forma duală a problemei de optimizare SVM —, fie (considerabil mai simplu!) folosind o formă ușor generalizată a proprietății pe care am demonstrat-o la problema 2, care face legătura dintre problema de optimizare SVM și *interpretarea ei geometrică*.

21.

(Problema de optimizare C-SVM:
o formulare echivalentă, dar fără restricții,
folosind în schimb funcția de cost / pierdere hinge)

*prelucrare de Liviu Ciortuz, după
CMU, 2008 fall, Eric Xing, HW2, pr. 1.2
CMU, 2017 fall, Nina Balcan, HW4. pr. 4.1.1*

Considerăm m instanțe de antrenament $\{x_i, y_i\}_{i=1}^m$. Vă readucem aminte că problema SVM cu margine “soft” și parametru de „destindere” $C > 0$ poate fi formulată ca o problemă de optimizare (pătratică) cu restricții:

$$\begin{aligned} & \min_{w, w_0, \xi} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \right) \\ \text{a. i. } & (w \cdot x_i + w_0) y_i \geq 1 - \xi_i, \text{ pentru } i = 1, \dots, m \\ & \xi_i \geq 0, \text{ pentru } i = 1, \dots, m \end{aligned} \tag{P'}$$

a. Demonstrați că formularea de mai sus este *echivalentă* cu o problemă de optimizare (tot pătratică) *fără* restricții, de forma:

$$\min_{w, w_0} \left(\|w\|^2 + \lambda \sum_{i=1}^m \max(1 - y_i(w \cdot x_i + w_0), 0) \right), \tag{L}$$

unde λ este un parametru real pozitiv fixat.

b. Exprimăți valoarea noului parametru λ în funcție de parametrul de „destindere” C .

c. Cum apreciați din punct de vedere „calitativ“ această nouă formulare a problemei de optimizare C-SVM?

$\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$. Fie $\bar{\alpha}$ soluție optimă a problemei (D') , cu componenta k nulă, adică $\bar{\alpha}_k = 0$. Urmează că $F(\bar{\alpha}) \geq F(\alpha)$ pentru orice soluție admisibilă α a problemei (D') , deci

$$F(\bar{\alpha}) = \max\{F(\alpha) \mid \alpha \text{ este soluție admisibilă a problemei } (D')\}.$$

Întrucât multimea de soluții admisibile pentru problema (D') este mai amplă decât multimea de soluții admisibile pentru problema (D'') , rezultă că

$$F(\bar{\alpha}) \geq \max\{F(\alpha) \mid \alpha \text{ este soluție admisibilă a problemei } (D'')\}.$$

Însă ținând cont pe de o parte că $\bar{\alpha}$ este și soluție admisibilă pentru problema (D'') , iar pe de altă parte că $\bar{\alpha}_k = 0$, rezultă că inegalitatea precedentă devine

$$F(\bar{\alpha}) = \max\{F(\alpha) \mid \alpha \text{ este soluție admisibilă a problemei } (D'')\}.$$

Așadar, $\bar{\alpha}$ este soluție optimă și pentru problema (D'') .

Răspuns:

a. Din restricțiile din forma primală a problemei C-SVM, avem

$$\xi_i \geq 1 - y_i(w \cdot x_i + w_0) \text{ și } \xi_i \geq 0,$$

ceea ce echivalează cu $\xi_i \geq \max(1 - y_i(w \cdot x_i + w_0), 0)$ pentru $i = 1, \dots, m$.

Operatorul min din formularea obiectivului problemei de optimizare C-SVM (P') implică faptul că $\bar{\xi}_i$ din soluția optimă a acestei probleme $(\bar{w}, \bar{w}_0, \bar{\xi})$ va fi setată chiar la valoarea $\max(1 - y_i(\bar{w} \cdot x_i + \bar{w}_0), 0)$.

Prin urmare, este naturală transformarea funcției obiectiv a problemei (P') în funcția obiectiv a problemei (L). Ceea ce nu apare a fi la fel de ușor de explicat este renunțarea la restricțiile formulate în cadrul problemei (P') atunci când se face trecerea la problema (L). De aceea, în cele ce urmează este necesar să arătam că aceste două probleme sunt *echivalente*, dovedind că orice soluție optimă a problemei (P') corespunde unei soluții optime a problemei (L) și invers, orice soluție optimă a problemei (L) corespunde unei soluții optime a problemei (P').

Considerăm deci mai întâi $\bar{w}, \bar{w}_0, \bar{\xi}$ o soluție optimă a problemei C-SVM (P'). Luând $\lambda = 2C$, vom arăta că \bar{w}, \bar{w}_0 este și soluție optimă a noii probleme de optimizare (L). Presupunem prin *prin reducere la absurd* că \bar{w}, \bar{w}_0 ne este soluție optimă a noii probleme de optimizare (L). Rezultă că există w^*, w_0^* o soluție optimă a problemei (L), care este mai bună decât soluția \bar{w}, \bar{w}_0 , adică

$$L(w^*, w_0^*) < L(\bar{w}, \bar{w}_0),$$

unde prin L am notat funcția obiectiv a problemei de optimizare (L). Notând $\xi_i^* \stackrel{not.}{=} \max(1 - y_i(w^* \cdot x_i + w_0^*), 0)$, rezultă că w^*, w_0^*, ξ^* este pentru problema C-SVM (P') o soluție fezabilă, adică satisfac sistemul de restricții $(w \cdot x_i + w_0)y_i \geq 1 - \xi_i$ și $\xi_i \geq 0$, pentru $i = 1, \dots, m$. Pe lângă aceasta, notând cu P' funcția obiectiv a problemei de optimizare C-SVM (P'), rezultă

$$\underbrace{P'(w^*, w_0^*, \xi^*)}_{=2L(w^*, w_0^*)} < \underbrace{P'(\bar{w}, \bar{w}_0, \bar{\xi})}_{=2L(\bar{w}, \bar{w}_0)}.$$

În această relație, egalitatea $P'(w^*, w_0^*, \xi^*) = 2L(w^*, w_0^*)$ are loc datorită modului în care am definit L , P' și ξ^* , iar egalitatea $P'(\bar{w}, \bar{w}_0, \bar{\xi}) = 2L(\bar{w}, \bar{w}_0)$ are loc datorită operatorului min din cadrul obiectivului problemei (P'). Inegalitatea $P'(w^*, w_0^*, \xi^*) < P'(\bar{w}, \bar{w}_0, \bar{\xi})$ implică faptul că soluția w^*, w_0^*, ξ^* constituie pentru problema (P') o soluție mai bună decât soluția optimă $\bar{w}, \bar{w}_0, \bar{\xi}$, ceea ce este absurd. Prin urmare, \bar{w}, \bar{w}_0 este soluție optimă a problemei (L).

Invers, trebuie să arătăm acum că luând $\lambda = 2C$, orice soluție optimă \bar{w}, \bar{w}_0 a problemei (L) este – printr-o extensie naturală – soluție optimă a problemei C-SVM (P'). Este imediat (vedeți raționamentul de mai sus) că orice soluție optimă \bar{w}, \bar{w}_0 a problemei (L), augmentată cu $\bar{\xi}_i \stackrel{not.}{=} \max(1 - y_i(\bar{w} \cdot x_i + \bar{w}_0), 0)$ satisfac restricțiile problemei (P'). Presupunem, ca și mai sus, prin reducere la absurd că $\bar{w}, \bar{w}_0, \bar{\xi}$ nu este soluție optimă a problemei (P'). Considerând w^*, w_0^*, ξ^* o soluție optimă a problemei (P'), rezultă:³⁰⁷

$$\underbrace{P'(w^*, w_0^*, \xi^*)}_{=2L(w^*, w_0^*)} < \underbrace{P'(\bar{w}, \bar{w}_0, \bar{\xi})}_{=2L(\bar{w}, \bar{w}_0)}.$$

³⁰⁷Egalitățile $P'(w^*, w_0^*, \xi^*) = 2L(w^*, w_0^*)$ și $P'(\bar{w}, \bar{w}_0, \bar{\xi}) = 2L(\bar{w}, \bar{w}_0)$ se mențin (cu aceeași justificare ca mai sus).

Așadar, $L(w^*, w_0^*) < L(\bar{w}, \bar{w}_0)$, ceea ce contrazice faptul că \bar{w}, \bar{w}_0 este soluție optimă pentru problema (L). Conchidem că presupunerea făcută mai sus este falsă, deci $\bar{w}, \bar{w}_0, \bar{\xi}$ este soluție optimă pentru problema (P').

b. $\lambda = 2C$, conform rezolvării de la punctul a.

c. Forma nou-obținută pentru problema de optimizare C-SVM este caracterizată de *i.* lipsa restricțiilor asupra variabilelor și *ii.* de realizarea unui echilibru (engl., trade-off) între

- simplitate,³⁰⁸ reflectată de termenul $\|w\|^2$;
- o bună capacitate de predicție / generalizare în cazul neseparabilității liniare a datelor de antrenament, grație termenului $\lambda \sum_{i=1}^m \max(1 - y_i(w \cdot x_i + w_0), 0)$.

Observație importantă: [Relația cu regresia logistică]

Este imediat că problema de optimizare (L) este echivalentă cu problema

$$\min_{w, w_0} \left(\theta \|w\|^2 + \sum_{i=1}^m \max(1 - y_i(w \cdot x_i + w_0), 0) \right), \quad (124)$$

dacă se consideră parametrul (fixat) $\theta = \frac{1}{\lambda} > 0$. Comparând problema de optimizare (124) cu problema de optimizare din definiția regresiei logistice cu termen de regularizare L_2 ,³⁰⁹ putem observa o mare similaritate. Diferența constă (doar!) în funcția de cost / pierdere folosită: SVM folosește *funcția de cost hinge*, definită prin $f(z) = \max(1 - z, 0)$, pe când regresia logistică folosește *funcția de cost logistică* $\ln(1 + e^{-z})$.³¹⁰

Observație: Pentru detalii de implementare (și aplicare) a acestei forme a problemei de optimizare SVM folosind *metoda subgradientului* (pentru că funcția *hinge* nu este derivabilă pe tot domeniul de definiție), vedeți articolul *Pegasos: Primal estimated sub-gradient solver for SVM*, de Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, Andrew Cotter, publicat în revista *Mathematical programming*, 127(1):3–30, 2011.³¹¹

22. (C-SVM: deducerea relațiilor folosite în algoritmul SMO)

Liviu Ciortuz, 2018, după

■ Nello Cristianini, John Shawe-Taylor,
An Introduction to Support Vector Machines,
Cambridge University Press, 2000, pp. 139-140

În această problemă veți face deducerea relațiilor (pentru *actualizare* și respectiv pentru *oprire*) folosite în algoritmul SMO,³¹² care rezolvă forma duală a problemei de optimizare

³⁰⁸În raport cu alți clasificatori, de exemplu rețelele neuronale artificiale.

³⁰⁹Puteți vedea problemele 12.c, 17, 14 și 15 de la Capitolul *Metode de regresie*.

³¹⁰În mod similar, regresia liniară folosește ca funcție de cost suma pătratelor erorilor, iar algoritmul AdaBoost folosește funcția de cost [negativ]-exponențială e^{-z} . Pentru un cadru unitar de prezentare a acestor metode de învățare (foarte diferite!), bazat pe minimizarea costurilor / pierderilor, vă recomandăm să citiți documentul *Supplemental lecture notes*, de John Duchi, de la Universitatea Stanford.

³¹¹Vedeți și CMU, 2017 fall, Nina Balcan, HW4, pr. 4.1.

³¹²John C. Platt, *Sequential Minimal Optimization: A fast algorithm for training Support Vector Machines*. Microsoft Research, Technical report MSR-TR-98-14, 1998. O variantă simplificată a algoritmului SMO este prezentată de Andrew Ng în *Machine Learning, Lecture Notes*, Part V, section 9.

C-SVM. Acest algoritm folosește ca strategie de optimizare *metoda creșterii pe coordonate* (engl., *coordinate ascent*).³¹³ Aceasta este o metodă iterativă. În *cazul clasic* al aplicării acestei metode, la fiecare iterație se alege câte o variabilă care este lăsată liberă, iar celalte variabile sunt fixate; la iterarea respectivă, optimizarea funcției obiectiv se face în raport cu variabila liberă. Însă, spre deosebire de cazul clasic, la aplicarea metodei creșterii pe coordonate pentru rezolvarea problemei duale C-SVM, datorită restricției $\sum_{i=1}^m y_i \alpha_i = 0$ va trebui ca la fiecare iterație să alegem două variabile pe care să le lăsăm libere, iar pe restul să le fixăm.³¹⁴

a. Folosind notațiile standard de la problema de optimizare C-SVM (în forma primală și respectiv forma duală; vedeți pr. 13) și presupunând că *variabilele libere* la iterarea curentă sunt α_1 și α_2 , demonstrați că lagrangeanul dual L_D are la această iterație soluțiile optime

$$\begin{aligned}\alpha_2^{new, unclipped} &= \alpha_2 + \frac{y_2(E_2 - E_1)}{\eta} \\ \alpha_1^{new, unclipped} &= \alpha_1 + y_1 y_2 (\alpha_2 - \alpha_2^{new, unclipped}),\end{aligned}$$

unde

$$\begin{aligned}E_k &= \underbrace{w \cdot x_k + w_0}_{not. : f(x_k)} - y_k \text{ pentru } k \in \{1, 2\}, \\ w &= \sum_{i=1}^m y_i \alpha_i x_i, \\ \eta &= -\|x_1 - x_2\|^2.\end{aligned}$$

b. Vă readucem aminte că $0 \leq \alpha_j \leq C$ pentru $j \in \{1, 2\}$ (vedeți problema 13.d). La fiecare iterație a algoritmului SMO trebuie calculate două *margini* (engl., bounds), L și H , astfel încât să restricționăm și mai mult variabila α_2 : $0 \leq L \leq \alpha_2 \leq H \leq C$.

Demonstrați că (în contextul problemei noastre) aceste margini sunt conform următoarelor relații:

- dacă $y_1 \neq y_2$, atunci $L = \max(0, \alpha_2 - \alpha_1)$, $H = \min(C, C + \alpha_2 - \alpha_1)$;
- dacă $y_1 = y_2$, atunci $L = \max(0, \alpha_1 + \alpha_2 - C)$, $H = \min(\alpha_1 + \alpha_2, C)$.

În consecință, *regulile de actualizare* vor fi:

$$\begin{aligned}\alpha_2^{new, clipped} &= \begin{cases} H & \text{dacă } \alpha_2^{new, unclipped} > H \\ \alpha_2^{new, unclipped} & \text{dacă } L \leq \alpha_2^{new, unclipped} \leq H \\ L & \text{dacă } \alpha_2^{new, unclipped} < L \end{cases} \quad (125) \\ \alpha_1^{new, clipped} &= \gamma - s\alpha_2^{new, clipped} = \alpha_1^{old} + s\alpha_2^{old} - s\alpha_2^{new, clipped} \\ &= \alpha_1^{old} + y_1 y_2 (\alpha_2^{old} - \alpha_2^{new, clipped}),\end{aligned}$$

unde calificativele *new* și *old* desemnează noile și respectiv vechile valori ale variabilelor α_1 și α_2 , iar calificativul *clipped* (spre deosebire de *unclipped*) corespunde „ajustării“ valorii lui α_2 ca urmare a aplicării restricției $L \leq \alpha_2 \leq H$.

c. Algoritmul SMO folosește următoarea *condiție de oprire*:

³¹³Metoda aceasta este folosită în mod indirect la algoritmul *K-means* (văzut ca algoritm de minimizare a criteriului sumei celor mai mici pătrate (J)) și la algoritmul EM; vedeți problema 12.a de la capitolul de *Clusterizare* și respectiv problema 1 de la capitolul *Algoritmul EM*.

³¹⁴În prezentă problemă nu includem detalii despre modul în care se aleg aceste două variabile.

NOT $[(\alpha_i < C \text{ AND } y_i E_i < -tol) \text{ OR } (\alpha_i > 0 \text{ AND } y_i E_i > tol)]$, $i = 1, \dots, m$, unde tol este un număr pozitiv (mic, fixat în prealabil), numit *parametru de toleranță*. Demonstrați că această condiție este implicată în mod natural de relațiile de tip Karush-Kuhn-Tucker (122) care au fost deduse (la ex. 13) pentru problema de optimizare C-SVM.

Răspuns:

a. Știm — conform problemei 13.d — că forma duală a problemei de optimizare C-SVM are funcția obiectiv $L_D(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j x_i \cdot x_j$, cu restricțiile $\sum_{i=1}^m y_i \alpha_i = 0$ (atât pentru noile cât și pentru vechile valori ale variabilelor) și $0 \leq \alpha_i \leq C$ pentru $i = 1, \dots, m$. Notând

$$v_i = \left(\underbrace{\sum_{j=3}^m y_j \alpha_j x_j}_{w - (y_1 \alpha_1 x_1 + y_2 \alpha_2 x_2)} \right) \cdot x_i \text{ pentru } i = 1, 2,$$

va rezulta că

$$v_i = w \cdot x_i - (y_1 \alpha_1 x_1 \cdot x_i + y_2 \alpha_2 x_2 \cdot x_i) \text{ pentru } i \in \{1, 2\} \quad (126)$$

și

$$L_D(\alpha_1, \alpha_2) = \alpha_1 + \alpha_2 - \frac{1}{2} \alpha_1^2 x_1^2 - \frac{1}{2} \alpha_2^2 x_2^2 - \underbrace{y_1 y_2}_{not.: s} \alpha_1 \alpha_2 x_1 \cdot x_2 - y_1 \alpha_1 v_1 - y_2 \alpha_2 v_2 + const_1.$$

Întrucât $y_1 \alpha_1 + y_2 \alpha_2 = -\sum_{j>2} y_j \alpha_j$, înmulțind această egalitate cu y_1 — despre care știm că aparține multimii $\{-1, 1\}$ —, vom obține

$$\alpha_1 + \underbrace{y_1 y_2}_{s} \alpha_2 = -y_1 \underbrace{\sum_{j>2} y_j \alpha_j}_{const_2}.$$

Așadar,

$$\alpha_1^{old} + s \alpha_2^{old} = \underbrace{const_2}_{not.: \gamma} = \alpha_1^{new, unclipped} + s \alpha_2^{new, unclipped}. \quad (127)$$

Substituind $\alpha_1 = \gamma - s \alpha_2$ în expresia lagrangeanului $L_D(\alpha_1, \alpha_2)$, vom obține — renunțând totodată la argumentul α_1 — următoarea expresie:

$$\begin{aligned} L_D(\alpha_2) &= \gamma - s \alpha_2 + \alpha_2 - \frac{1}{2}(\gamma - s \alpha_2)^2 x_1^2 - \frac{1}{2} \alpha_2^2 x_2^2 - s(\gamma - s \alpha_2) \alpha_2 x_1 \cdot x_2 \\ &\quad - y_1(\gamma - s \alpha_2) v_1 - y_2 \alpha_2 v_2 + const_1. \end{aligned}$$

Pentru a maximiza $L_D(\alpha_2)$, mai întâi vom calcula derivata sa în raport cu α_2 , după care vom egala această derivată cu 0:

$$\begin{aligned} \frac{\partial L_D(\alpha_2)}{\partial \alpha_2} &= -s + 1 + \frac{1}{2} 2s(\gamma - s \alpha_2) x_1^2 - \frac{1}{2} 2 \alpha_2 x_2^2 - s \gamma x_1 \cdot x_2 + 2 \alpha_2 x_1 \cdot x_2 + y_1 s v_1 - y_2 v_2 \\ &= -s + 1 + (s\gamma - \alpha_2) x_1^2 - \alpha_2 x_2^2 - s \gamma x_1 \cdot x_2 + 2 \alpha_2 x_1 \cdot x_2 + y_1 s v_1 - y_2 v_2 \\ &= -\alpha_2(x_1^2 + x_2^2 - 2x_1 \cdot x_2) - s + 1 + s\gamma x_1^2 - s \gamma x_1 \cdot x_2 + y_1 \underbrace{s v_1 - y_2 v_2}_{y_1 y_2} \\ &= -\alpha_2(x_1 - x_2)^2 - s + 1 + s\gamma x_1^2 - s \gamma x_1 \cdot x_2 + y_2 v_1 - y_2 v_2 = 0. \end{aligned}$$

Întrucât $(x_1 - x_2)^2 > 0$ pentru orice $x_1 \neq x_2$, rezultă că semnele acestei derivate corespund existenței maximului pentru $L_D(\alpha_2)$. Așadar, soluția este:

$$\begin{aligned}\alpha_2^{new, unclipped} &= \frac{-s + 1 + s\gamma(x_1^2 - x_1 \cdot x_2) + y_2(v_1 - v_2)}{x_1^2 + x_2^2 - 2x_1 \cdot x_2} \\ &= \frac{y_2(-y_1 + y_2 + y_1\gamma(x_1^2 - x_1 \cdot x_2) + v_1 - v_2)}{\|x_1 - x_2\|^2} \\ &\stackrel{(*)}{=} \frac{y_2(f(x_1) - y_1 - f(x_2) + y_2) + \alpha_2\|x_1 - x_2\|^2}{\|x_1 - x_2\|^2} \\ &= \alpha_2 + \frac{y_2(E_1 - E_2)}{\|x_1 - x_2\|^2} = \alpha_2 + \frac{y_2(E_2 - E_1)}{-\|x_1 - x_2\|^2}.\end{aligned}$$

Observație: Egalitatea (*) are loc întrucât folosind notația $f(x) = w \cdot x + w_0 = \sum_{j=1}^m y_j \alpha_j x_j \cdot x + w_0$ putem scrie

$$\begin{aligned}v_1 - v_2 &\stackrel{(126)}{=} f(x_1) - y_1 \alpha_1 x_1^2 - y_2 \alpha_2 x_1 \cdot x_2 - (f(x_2) - y_1 \alpha_1 x_1 \cdot x_2 - y_2 \alpha_2 x_2^2) \\ &= f(x_1) - y_1 \underbrace{\alpha_1 x_1^2}_{\gamma - s\alpha_2} - y_2 \alpha_2 x_1 \cdot x_2 - f(x_2) + y_1 \underbrace{\alpha_1 x_1 \cdot x_2}_{\gamma - s\alpha_2} + y_2 \alpha_2 x_2^2 \\ &= f(x_1) - f(x_2) - y_1 \gamma x_1^2 + \underbrace{sy_1 \alpha_2 x_1^2}_{y_2} - y_2 \alpha_2 x_1 \cdot x_2 + y_1 \gamma x_1 \cdot x_2 - \underbrace{sy_1 \alpha_2 x_1 \cdot x_2}_{y_2} + \\ &\quad y_2 \alpha_2 x_2^2 \\ &= f(x_1) - f(x_2) - y_1 \gamma x_1^2 + y_2 \alpha_2 x_1^2 - y_2 \alpha_2 x_1 \cdot x_2 + y_1 \gamma x_1 \cdot x_2 - y_2 \alpha_2 x_1 \cdot x_2 + y_2 \alpha_2 x_2^2 \\ &= f(x_1) - f(x_2) - y_1 \gamma (x_1^2 - x_1 \cdot x_2) + y_2 \alpha_2 (x_1^2 - 2x_1 \cdot x_2 + x_2^2) \\ &= f(x_1) - f(x_2) - y_1 \gamma (x_1^2 - x_1 \cdot x_2) + y_2 \alpha_2 \|x_1 - x_2\|^2.\end{aligned}$$

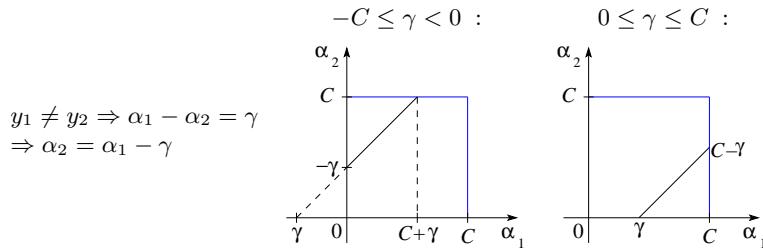
În final, pentru a deduce $\alpha_1^{new, unclipped}$, vom folosi egalitatea dublă

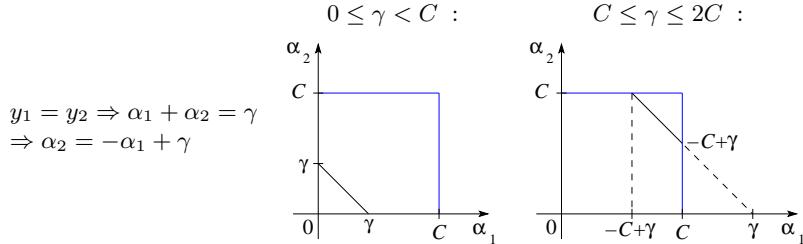
$$\alpha_1^{old} + s\alpha_2^{old} = \gamma = \alpha_1^{new, unclipped} + s\alpha_2^{new, unclipped}.$$

Așadar,

$$\begin{aligned}\alpha_1^{new, unclipped} &= \gamma - s\alpha_2^{new, unclipped} = \alpha_1^{old} + s\alpha_2^{old} - s\alpha_2^{new, unclipped} \\ &= \alpha_1^{old} + y_1 y_2 (\alpha_2^{old} - \alpha_2^{new, unclipped}).\end{aligned}$$

b. Vom analiza pe rând cele două cazuri: $y_1 = y_2$ (care implică $\gamma = \alpha_1 - \alpha_2$, conform relației (127)) și respectiv $y_1 \neq y_2$ (care implică $\gamma = \alpha_1 + \alpha_2$). Pentru fiecare din aceste două cazuri, în funcție de semnul lui $\gamma \stackrel{not.}{=} y_1 + sy_2$ și respectiv în funcție de semnul lui $\gamma - C$, vom avea câte două subcazuri, reprezentate grafic în partea dreaptă.





Pentru cazul $y_1 = y_2$, se observă din cele două grafice că $\alpha_2 \geq -\gamma$ și respectiv $\alpha_2 \geq 0$, deci $\alpha_2 \geq L = \max(-\gamma, 0)$. De asemenea, $\alpha_2 \leq C$ și respectiv $\alpha_2 \leq C - \gamma$, de unde rezultă $H = \min(C, C - \gamma)$. Similar, pentru cazul $y_1 = y_2$ vom obține $L = \max(0, -C + \gamma)$ și $H = \min(\gamma, C)$.

În fine, regula de actualizare (125) decurge în mod natural din proprietățile funcției de gradul al doilea aplicate pentru $L_D(\alpha_2)$, care trebuie maximizată ținând cont de restricțiile $L \leq \alpha_2 \leq H$.³¹⁵

c. Vom folosi egalitatea $y_i E_i = y_i(f(x_i) - y_i) = y_i f(x_i) - 1$. Conform relațiilor (122), cazurile $\alpha_i < C$ și $\alpha_i > 0$ vor avea câte două subcazuri:

$$\alpha_i < C : \left\{ \begin{array}{ll} \alpha_i = 0 & \Rightarrow y_i \underbrace{(w \cdot x_i + w_0)}_{f(x_i)} \geq 1 \Rightarrow y_i E_i \geq 0 \\ \alpha_i \in (0, C) & \Rightarrow y_i f(x_i) = 1 \Rightarrow y_i E_i = 0 \end{array} \right\} \Rightarrow y_i E_i \geq 0 \quad (128)$$

$$\alpha_i > 0 : \left\{ \begin{array}{ll} \alpha_i \in (0, C) & \Rightarrow y_i f(x_i) = 1 \Rightarrow y_i E_i = 0 \\ \alpha_i = C & \Rightarrow y_i f(x_i) \leq 1 \Rightarrow y_i E_i \leq 0 \end{array} \right\} \Rightarrow y_i E_i \leq 0. \quad (129)$$

Notând premisele și respectiv consecințele relațiilor (128) și (129) cu niște variabile din logica propozițiilor, și anume $p_1 \stackrel{\text{not.}}{=} (\alpha_i < C)$, $q_1 \stackrel{\text{not.}}{=} (y_i E_i \geq 0)$, $p_2 \stackrel{\text{not.}}{=} (\alpha_i > 0)$, și $q_2 \stackrel{\text{not.}}{=} (y_i E_i \leq 0)$, vom putea scrie următoarele echivalențe, folosind mai întâi legea dublei negații și apoi legile lui De Morgan:

$$(p_1 \rightarrow q_1) \wedge (p_2 \rightarrow q_2) \equiv \neg\neg[(p_1 \rightarrow q_1) \wedge (p_2 \rightarrow q_2)] \equiv \neg[\neg(p_1 \rightarrow q_1) \vee \neg(p_2 \rightarrow q_2)] \equiv \neg[\neg(\neg p_1 \vee q_1) \vee \neg(\neg p_2 \vee q_2)] \equiv \neg[(p_1 \wedge \neg q_1) \vee (p_2 \wedge \neg q_2)].$$

Expresia finală — $\neg[(p_1 \wedge \neg q_1) \vee (p_2 \wedge \neg q_2)]$ — coincide cu *condiția de oprire* din enunț, doar că $\neg q_1$ și $\neg q_2$ au fost rescrise (din motive de „stabilitate“ numerică la efectuarea calculelor) sub forma $y_i E_i < -tol$, și respectiv $y_i E_i > tol$, unde tol este constanta pozitivă considerată în enunț. Atunci când această condiție este satisfăcută pentru $i = 1, \dots, m$, algoritmul SMO poate fi oprit, întrucât [se consideră că] sunt satisfăcute condițiile de tip Karush-Kuhn-Tucker (122).

³¹⁵Pentru a vă convinge, vă recomandăm să faceți câte un grafic generic pentru funcția $L_d(\alpha_2)$ pentru fiecare din cele trei cazuri ale regulei (125). (Pentru două exemple concrete, puteți vedea graficele de la rezolvarea problemei 23.b.)

23. (C-SVM: exemplu de aplicare a algoritmului SMO)
 ■ CMU, 2008 fall, Eric Xing, HW2, pr. 1.3

Se dau patru instanțe în spațiul euclidian bidimensional, împreună cu etichetele asignate lor: $x_1 = (0, 1)$, $y_1 = -1$; $x_2 = (2, 0)$, $y_2 = +1$; $x_3 = (1, 0)$, $y_3 = +1$ și $x_4 = (0, 2)$, $y_4 = -1$. Vom folosi aceste exemple pentru a antrena un C-SVM (SVM liniar cu margine „soft” și parametru de „destindere” C). Fie $\alpha_1, \alpha_2, \alpha_3$ și α_4 multiplicatorii Lagrange asociati instanțelor x_1, x_2, x_3 și respectiv x_4 . Fixăm valoarea parametrului C la 100.

- Scrieți forma duală a problemei de optimizare C-SVM în acest caz.
- Vă cerem să executați două iterații ale algoritmului SMO (Sequential Minimal Optimization) pe acest set de date.³¹⁶ Presupunem că facem inițializările $\alpha_1 = 5$, $\alpha_2 = 4$, $\alpha_3 = 8$, $\alpha_4 = 7$.
 - La prima iterație veți actualiza multiplicatorii α_1 și α_4 (păstrând α_2 și α_3 fixați). Stabiliți relațiile de actualizare a valorilor pentru α_1 și α_4 în funcție de valorile multiplicatorilor α_2 și α_3 . Ce valori vor avea α_1 și α_4 după actualizare?
 - La a doua iterație veți fixa α_1 și α_4 și veți stabili relațiile de actualizare pentru α_2 și α_3 în funcție de α_1 și α_4 . Ce valori vor avea α_2 și α_3 după actualizare?

Răspuns:

- Particularizând forma duală a problemei de optimizare C-SVM (vedeți problema 13) pentru acest set de date, după efectuarea tuturor produselor scalare $x_i \cdot x_j$ vom obține:³¹⁷

$$\max_{\alpha_1, \alpha_2, \alpha_3, \alpha_4} [\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(\alpha_1^2 + 4\alpha_2^2 + \alpha_3^2 + 4\alpha_4^2 + 4\alpha_1\alpha_4 + 4\alpha_2\alpha_3)]$$

- $0 \leq \alpha_i \leq 100$, pentru $i = 1, \dots, 4$
 $-\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0$

- Facem *observația* că, la acest punct, exercițiul cere să se execute două iterații ale algoritmului SMO pe datele furnizate, fără a aplica criteriul de selecție a variabilelor libere și condițiile de oprire formulate de John Platt, autorul acestui algoritm.

Vom da mai jos două rezolvări. Prima va fi mai simplă. Ea va urma *ideea* algoritmului SMO — care aplică o metodă de optimizare numită *creștere pe coordonate* (engl., coordinate ascent) — fără a recurge efectiv la formulele stabilite de John Platt. În schimb, vom proceda direct la optimizarea funcțiilor obiectiv determinate de alegerea (impusă, conform enunțului) a celor două perechi de variabile duale specificate. La a doua rezolvare, vom aplica direct formulele generale pentru „actualizarea“ valorilor libere din algoritmul SMO. Facem *observația* că și această rezolvare va fi utilă cititorului, pentru că vom scoate în evidență anumite detalii / modalități de calcul care nu sunt chiar „immediate“ pentru cineva care nu este încă obisnuit cu algoritmul SMO.

Prima soluție:

- La prima iterație, inițial avem $\alpha_1 = 5$, $\alpha_2 = 4$, $\alpha_3 = 8$, $\alpha_4 = 7$, iar apoi se lasă „libere“ variabilele α_1 și α_4 . Datorită restricției $\sum_{i=1}^4 y_i \alpha_i = 0$ din forma duală a problemei de optimizare C-SVM (a se vedea (D') la problema 13), vom avea următoarea relație de legătură dintre valorile variabilelor libere: $\alpha_1^{new} + \alpha_4^{new} = \alpha_2 + \alpha_3 = 12$.

³¹⁶Vedeți problema 22 din prezentul capitol.

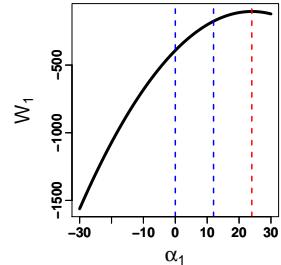
³¹⁷Concret, $x_1^2 = x_3^2 = 1$, $x_2^2 = x_4^2 = 4$, $x_1 \cdot x_4 = x_2 \cdot x_3 = 2$, iar restul produselor $x_i \cdot x_j$ cu $i \neq j$ au valoarea 0.

Din restricția $\alpha_i \geq 0$ — dar și datorită faptului că y_1 și y_4 au același semn — va rezulta că valorile posibile („fezabile“) pentru α_1 și α_4 vor fi limitate la intervalul $[0, 12]$, inclus în intervalul $[0, C] = [0, 100]$.

Înlocuind $\alpha_2 = 4$, $\alpha_3 = 8$ și $\alpha_4 = 12 - \alpha_1$ în funcția obiectiv a problemei de optimizare de la punctul a , vom obține expresia funcției pe care va trebui să o maximizăm la această iterație:

$$\begin{aligned} W_1(\alpha_1) &\stackrel{\text{not.}}{=} 24 - \frac{1}{2}(\alpha_1^2 + 4 \cdot 4^2 + 8^2 + 4(12 - \alpha_1)^2 + 4\alpha_1(12 - \alpha_1) + 4 \cdot 4 \cdot 8) \\ &= 24 - \frac{1}{2}(\alpha_1^2 - 48\alpha_1 + 832). \end{aligned}$$

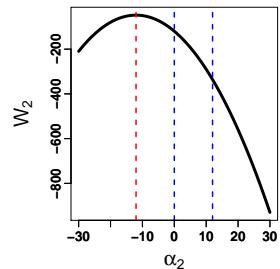
Valoarea lui α_1 pentru care se atinge optimul acestei funcții este notată cu $\alpha_1^{new, unclipped}$ și, evident, este $\frac{48}{2} = 24$. Această valoare se află în afara intervalului $[0, 12]$. Este imediat că maximul funcției $W_1(\alpha_1)$ pe intervalul $[0, 12]$ se atinge în punctul $\alpha_1^{new, clipped} = 12$. În consecință, α_4 va primi valoarea $\alpha_4^{new, unclipped} = 12 - \alpha_1^{new, clipped} = 0$.



ii. La a doua iterație, vom avea $\alpha_1 = 12$ și $\alpha_4 = 0$ fixați, iar $\alpha_2 = 4$ și $\alpha_3 = 8$ liberi. Din relația $\sum_{i=1}^4 y_i \alpha_i = 0$ rezultă $\alpha_2^{new} + \alpha_3^{new} = \alpha_1 + \alpha_4 = 12$. Ca și mai sus, intervalul în care vor fi permise noile valori ale variabilelor α_2 și α_3 este $[0, 12]$. Funcția obiectiv din problema de optimizare convexă devine:

$$\begin{aligned} W_2(\alpha_2) &\stackrel{\text{not.}}{=} 24 - \frac{1}{2}(12^2 + 4\alpha_2^2 + (12 - \alpha_2)^2 + 4 \cdot 0^2 + 4 \cdot 12 \cdot 0 + 4\alpha_2(12 - \alpha_2)) \\ &= -\frac{1}{2}\alpha_2^2 - 12\alpha_2 - 120 = -\frac{1}{2}(\alpha_2^2 + 24\alpha_2 + 240) \end{aligned}$$

Maximul global al acestei funcții se atinge în punctul $\alpha_2^{new, unclipped} = -\frac{24}{2} = -12$. Acest punct se situează în exteriorul intervalului de fezabilitate $[0, 12]$. Maximul funcției $W_2(\alpha_2)$ pe intervalul $[0, 12]$ se atinge în punctul $\alpha_2^{new, clipped} = 0$. În consecință, $\alpha_3^{new, unclipped} = 12 - \alpha_2^{new, clipped} = 12$.



A doua soluție:

Formulele date de John Platt pentru actualizarea variabilei libere α_i (la o iterare oarecare a algoritmului SMO) sunt:

$$\begin{aligned} \alpha_i^{new, unclipped} &= \alpha_i + \frac{y_i(E_i - E_j)}{\eta} \\ \alpha_i^{new, clipped} &= \begin{cases} H & \text{dacă } \alpha_i^{new, unclipped} > H \\ \alpha_i^{new, unclipped} & \text{dacă } L \leq \alpha_i^{new, unclipped} \leq H \\ L & \text{dacă } \alpha_i^{new, unclipped} < L, \end{cases} \end{aligned}$$

unde

$$\begin{aligned}
 E_k &= w \cdot x_k + w_0 - y_k \\
 w &= \sum_{i=1}^4 y_i \alpha_i x_i, \\
 \eta &= -\|x_i - x_j\|^2 \\
 L &= \max(0, \alpha_i - \alpha_j) \text{ și } H = \min(C, C + \alpha_i - \alpha_j) \text{ dacă } y_i \neq y_j \\
 L &= \max(0, \alpha_i + \alpha_j - C) \text{ și } H = \min(C, \alpha_i + \alpha_j) \text{ dacă } y_i = y_j.
 \end{aligned}$$

În consecință, vom avea:

i. La prima iterare, $\alpha_1 = 5$, $\alpha_2 = 4$, $\alpha_3 = 8$, $\alpha_4 = 7$, iar $\eta = -\|x_1 - x_4\|^2 = -1$. Fără a face deocamdată calculele, explicităm erorile $E_1 = f(x_1) - y_1 = w \cdot x_1 + w_0 - y_1$ și $E_4 = f(x_4) - y_4 = w \cdot x_4 + w_0 - y_4$, deci rezultă $E_1 - E_4 = w \cdot (x_1 - x_4)$. Din relația $w = \sum_{i=1}^4 y_i \alpha_i x_i$, urmează:

$$\begin{aligned}
 w &= -\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 - \alpha_4 x_4 \\
 &= -5 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 4 \begin{bmatrix} 2 \\ 0 \end{bmatrix} + 8 \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 7 \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 16 \\ -19 \end{bmatrix},
 \end{aligned}$$

de unde rezultă că $E_1 - E_4 = w \cdot (x_1 - x_4) = (16, -19) \cdot (0, -1) = 19$ și vom putea calcula $\alpha_1^{new, unclipped} = \alpha_1 + \frac{y_1(E_1 - E_4)}{\eta} = 5 + 19 = 24$.

Acum verificăm dacă $\alpha_1^{new, unclipped}$ este în intervalul de „fezabilitate“: deoarece $y_1 = y_4$, vom avea $L = \max(0, \alpha_1 + \alpha_4 - 100) = 0$, pentru că $\alpha_1 + \alpha_4 = 12$. Similar, $H = \min(100, 12) = 12$.

Întrucât $\alpha_1^{new, unclipped} > H = 12$, vom avea $\alpha_1^{new, clipped} = H = 12$ și, în consecință, $\alpha_4^{new, clipped} = 12 - \alpha_1^{new, clipped} = 0$.

ii. La a doua iterare, $\alpha_1 = 12$, $\alpha_2 = 4$, $\alpha_3 = 8$, $\alpha_4 = 0$ și $\alpha_2^{new} + \alpha_3^{new} = \alpha_2 + \alpha_3 = \alpha_1 + \alpha_4 = 12$. De asemenea, $\eta = -\|x_2 - x_3\|^2 = -1$ și $E_2 = f(x_2) - y_2 = w \cdot x_2 + w_0 - y_2$, iar $E_3 = f(x_3) - y_3 = w \cdot x_3 + w_0 - y_3$, deci $E_2 - E_3 = w \cdot (x_2 - x_3)$. Calculăm w astfel:

$$\begin{aligned}
 w &= -\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 - \alpha_4 x_4 \\
 &= -12 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 4 \begin{bmatrix} 2 \\ 0 \end{bmatrix} + 8 \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0 \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 16 \\ -12 \end{bmatrix},
 \end{aligned}$$

deci $E_2 - E_3 = (16, -12) \cdot (1, 0) = 16$. Așadar, $\alpha_2^{new, unclipped} = \alpha_2 + \frac{y_2(E_2 - E_3)}{\eta} = 4 - 16 = -12$. Deoarece $y_2 = y_3$, vom avea $L = \max(0, \alpha_2 + \alpha_3 - C) = 0$ și $H = \min(C, \alpha_2 + \alpha_3) = 12$. În consecință, $\alpha_2^{new, unclipped} = -12 < L = 0$, deci în final vom avea $\alpha_2^{new, clipped} = L = 0$ și $\alpha_3^{new, clipped} = 12 - \alpha_2^{new, clipped} = 12$. Se constată imediat că am regăsit rezultatele de la prima soluție.

Observație: Calculând valoarea funcției obiectiv $W(\alpha)$, folosind mai întâi valorile inițiale ale parametrilor α_i , apoi valorile rezultate la fiecare din cele două iterării, obținem valorile: -268.5 , -258 , -129 . Așa cum era de așteptat, aceste numere sunt în ordine crescătoare. Dacă am fi calculat și valorile parametrului w_0 care apare în forma primală a problemei de optimizare date, precum și valorile variabilelor de „destindere“ ξ_i , am fi putut calcula și valorile funcției obiectiv $\frac{1}{2} \|w\|^2 + C \sum_i \xi_i$. Aceste valori trebuie să fie

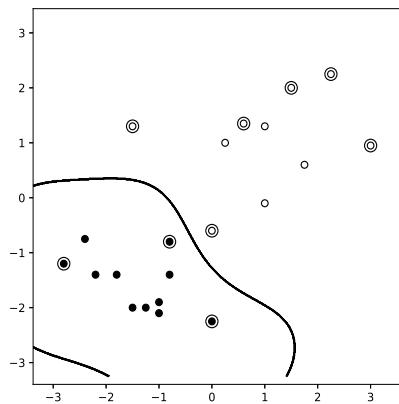
în ordine descrescătoare și mai mari decât valorile determinate pentru funcția $W(\alpha)$ mai sus (conform relației de *dualitate slabă* $d^* \leq p^*$; vedeți relația (112)).

24. (SVM și C-SVM, în forma primală sau forma duală: alegerea funcției-nucleu și a valorii parametrului C)
CMU, 2010 fall, Aarti Singh, HW3, pr. 3

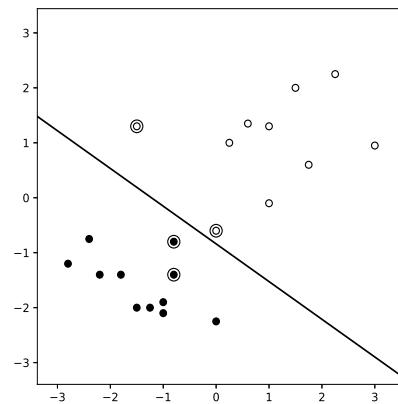
Figurile date mai jos indică suprafețele de decizie obținute de clasificatorul SVM pe un același set de date de antrenament fie în varianta marginii "hard" (folosind diferite funcții-nucleu), fie în varianta marginii "soft" (cu diferite valori pentru parametrul C). Exemplul pozitive sunt reprezentate prin simbolul \circ , iar cele negative prin simbolul \bullet . Acele care sunt încercuite desemnează vectori-suporți.

Indicați care dintre cele 6 figuri a fost generată de următoarele probleme de optimizare de tip SVM sau C-SVM (atenție, sunt 6 figuri și doar 5 probleme, deci o figură nu corespunde niciunei probleme):

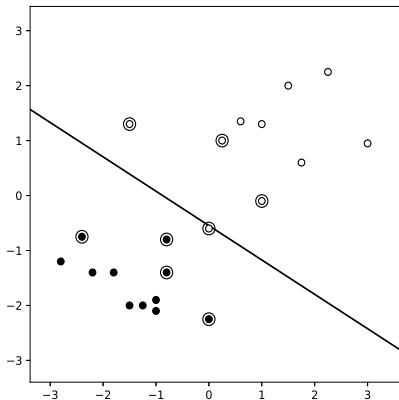
- a. $\min\left(\frac{1}{2}w \cdot w + C \sum_{i=1}^n \xi_i\right)$
 a.î. pentru $\forall i = 1, \dots, n$:
 $\xi_i \geq 0$
 $(w \cdot x_i + w_0)y_i \geq 1 - \xi_i$
 și $C = 0.1$.
- b. $\min\left(\frac{1}{2}w \cdot w + C \sum_{i=1}^n \xi_i\right)$
 a.î. pentru $\forall i = 1, \dots, n$:
 $\xi_i \geq 0$
 $(w \cdot x_i + w_0)y_i \geq 1 - \xi_i$
 și $C = 1$.
- c. $\max\left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\right)$
 a.î. $\sum_{i=1}^n \alpha_i y_i = 0$;
 $\alpha_i \geq 0, \forall i = 1, \dots, n$;
 unde $K(u, v) = u \cdot v + (u \cdot v)^2$.
- d. $\max\left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\right)$
 a.î. $\sum_{i=1}^n \alpha_i y_i = 0$;
 $\alpha_i \geq 0, \forall i = 1, \dots, n$;
- e. $\max\left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\right)$
 a.î. $\sum_{i=1}^n \alpha_i y_i = 0$;
 $\alpha_i \geq 0, \forall i = 1, \dots, n$;
 unde $K(u, v) = \exp\left(-\frac{\|u - v\|^2}{2}\right)$.



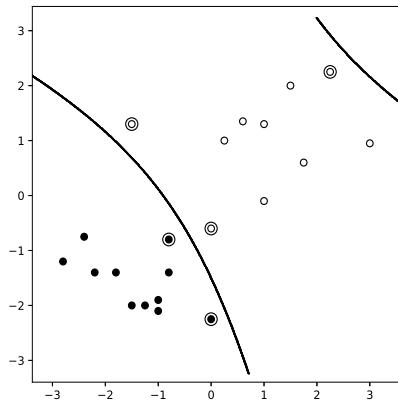
A.



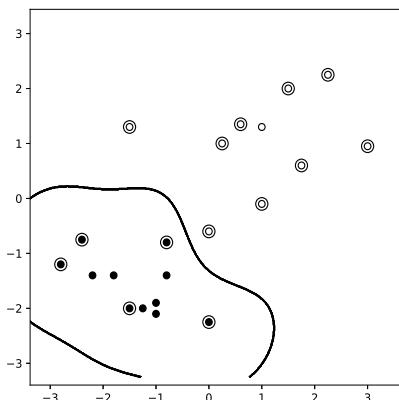
B.



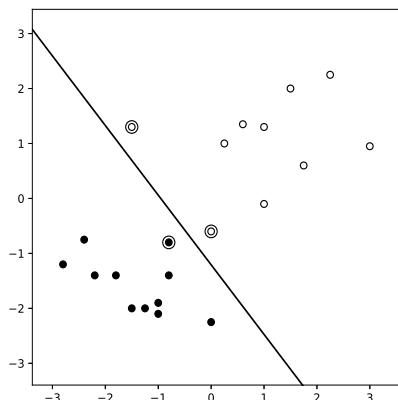
C.



D.



E.



F.

Răspuns:

Mai întâi vom „indica” în dreptul fiecăreia dintre cele cinci SVM-uri date (a-e) care este tipul mașinii respective (adică, SVM cu margine “hard”, respectiv SVM cu margine “soft”, deci, mai precis, C-SVM), iar în primul caz ce nucleu (liniar ori neliniar, de un anumit tip) îi corespunde.

- C-SVM liniar (i.e., fără funcție nucleu), cu erori mari;
- C-SVM liniar (i.e., fără funcție nucleu), cu erori [mai] mici [decât în cazul a];
- SVM cu margine “hard” (forma duală) cu funcție nucleu pătratică;
- SVM cu margine “hard” (forma duală) cu nucleu RBF (Radial Basis Function) de parametru $\sigma^2 = 1$;
- SVM cu margine “hard” (forma duală) cu nucleu RBF de parametru $\sigma^2 = \frac{1}{2}$.

Evident, este mai ușor de tratat cazul marginii “hard”. Avem trei astfel de probleme / SVM-uri: c, d și e. Dintre acestea, prima (c) lucrează cu nucleu pătratic, iar celelalte două cu nucleu de tip RBF. Din grafice se observă că doar separatorul din figura D are alură / formă parabolică. Așadar, graficul D corespunde problemei c. Dintre celelalte două

grafice (A și E), cel care se „mulează“ mai mult pe datele de antrenament este graficul E, deci lui îi va corespunde funcția RBF cu varianță mai mică, $\sigma^2 = \frac{1}{2}$. Așadar, mașinii e îi corespunde graficul E, iar mașinii d îi corespunde graficul A.

În sfârșit, pentru mașinile a și b — ambele folosind separator liniar, fără funcție nucleu —, cea care are valoarea parametrului C mai mică (0.1) permite erori mai multe / mari în raport cu marginea. Între cele trei grafice cu separator liniar (B, C și F), se observă că B și F nu au erori la antrenare.³¹⁸ Dintre acestea două, se observă că în graficul F există cei mai puțini (doar trei) vectori-suport și, aparent, eroarea totală în raport cu marginea, $\sum_i \xi_i$ este 0. Așadar, lui F îi corespunde $C = \infty$, pentru care nu avem corespondent (apropiat) între cele cinci mașini din enunț. Relativ la cele două grafice rămase în discuție (B și C), se observă că marginea — și, la fel, suma totală a erorilor în raport cu marginea — este mai mare în cazul lui C. Așadar, graficului C îi corespunde cea mai mică dintre valorile parametrului C rămase (0.1), deci mașina a, iar graficului B îi corespunde valoarea $C = 1$, deci mașina b.

25.

(C-SVM cu funcție-nucleu:
două condiții asupra parametrului C
și respectiv asupra funcției-nucleu, suficiente ca
toate instanțele de antrenament să fie vectori-suport)

*prelucrare de Liviu Ciortuz, după
MIT, 2008 fall, Tommi Jaakkola, midterm exam, pr. 1.3
CMU, 2010 fall, Aarti Singh, HW3, pr. 3.3.b*

Considerăm că antrenăm o mașină cu vectori-suport cu variabile de „destindere“ (engl., slack variables), care nu folosește variabila liberă w_0 (engl., bias variable). Se utilizează un nucleu $K(x, z)$ care are proprietatea $|K(x_i, x_j)| < 1$ dacă x_i și x_j (din setul de antrenament) sunt în relația $x_i \neq x_j$ și $|K(x_i, x_i)| \leq 1$ în caz contrar.³¹⁹ Sunt în total $m \geq 2$ puncte în setul de date de antrenament, dintre care cel puțin două sunt distințe.

Arătați că alegând pentru parametrul de „destindere“ C o valoare astfel încât $C < \frac{1}{m-1}$, toate variabilele α_i din soluția problemei duale vor fi nenule. (Așadar, toate instanțele de antrenament devin vectori-suport.)

Răspuns:

Fie $i \in \{1, \dots, m\}$, fixat. Prin definiție, instanța de antrenament x_i este vector-suport dacă $\alpha_i > 0$.

Dacă prin reducere la absurd am avea un $\alpha_i = 0$, ar rezulta

$$\begin{aligned} y_i w \cdot \Phi(x_i) &= y_i \left(\sum_{j=1}^m \alpha_j y_j \Phi(x_j) \right) \cdot \Phi(x_i) = y_i \left(\sum_{j=1}^m \alpha_j y_j K(x_j, x_i) \right) \\ &= y_i \left(\sum_{j \neq i}^m \alpha_j y_j K(x_j, x_i) \right) \leq \sum_{j \neq i}^m \alpha_j |K(x_j, x_i)| \end{aligned}$$

³¹⁸Atenție: noțiunea de *eroare în raport cu marginea* diferă de noțiunea de *eroare la clasificare*. Instanța x_i constituie eroare în raport cu marginea dacă $\xi_i > 0$.

³¹⁹Nucleul RBF îndeplinește aceste condiții.

$$< \sum_{j \neq i}^m \alpha_j \leq (m - 1)C$$

Întrucât $C < 1/(m - 1)$, vom avea $y_i w \cdot \Phi(x_i) < 1$.

Însă știm că din *condițiile de complementaritate duală Karush-Kuhn-Tucker* rezultă cu necesitate că dacă $\alpha_i = 0$ atunci $y_i w \cdot \Phi(x_i) \geq 1$, pentru $i = 1, \dots, m$. (A se vedea problema 13, *observația* de la rezolvarea punctului e.)

Am obținut deci o contradicție!

Prin urmare, singura posibilitate este ca α_i să fie nenul, ceea ce înseamnă că x_i este vector-suport.

26. (Condiții suficiente pentru ca o SVM cu nucleu RBF să producă eroare la antrenare 0)

■ Stanford, 2007 fall, Andrew Ng, HW2, pr. 3

Considerăm o mașină cu vectori-suport care folosește nucleul de tip gaussian (RBF) $K(x, z) = \exp(-\|x - z\|^2 / \tau^2)$, unde am notat cu $\exp()$ funcția exponențială. Parametrul τ determină mărimea deschiderii „clopotului“ gaussian.

La punctele a și b de mai jos vă vom ghida pas cu pas ca să demonstrați următoarea proprietate: în ipoteza că oricare două instanțe din setul de date de antrenament sunt distincte, se poate fixa o valoare a parametrului τ astfel încât eroarea la antrenare produsă de această SVM să fie zero. La punctul c se va arăta că această proprietate nu este valabilă și în cazul folosirii clasificatorului C-SVM.

a. Presupunem că setul de date de antrenament $\{(x_1, y_1), \dots, (x_m, y_m)\}$ este alcătuit din puncte care sunt separate unele de altele de o distanță de cel puțin ε , adică $\|x_j - x_i\| \geq \varepsilon$ pentru orice $i \neq j$.³²⁰

Vă readucem aminte că funcția de decizie învățată de către SVM cu funcție-nucleu K se poate scrie sub forma:³²¹

$$f(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + w_0, \quad (130)$$

unde $\alpha_1, \dots, \alpha_m \in \mathbb{R}_+$, w_0 este termenul liber / bias-ul din forma primală a problemei de optimizare [C-]SVM, iar $K(x_i, x) \stackrel{\text{def.}}{=} \Phi(x_i) \cdot \Phi(x)$, unde Φ este „maparea“ corespunzătoare funcției-nucleu K .

³²⁰Evident, dacă $x_i \neq x_j$ pentru orice $i \neq j$ și, bineînțeles, dacă m este finit, atunci putem lua $\varepsilon = \min_{i \neq j} \|x_j - x_i\|$.

³²¹Vedeți relația (119) de la problema 10.

Observație importantă: Această formulă presupunea *acolo* că α_i sunt soluțiile problemei SVM în forma duală. Însă *aici nu* lucrăm cu această presupozitie. Practic, aici vom alege α_i , w_0 și τ astfel încât să rezulte că f (de această formă) determină *separabilitate liniară* în spațiul de „trăsături“ în care sunt $\Phi(x_1), \dots, \Phi(x_m)$, deci și *separabilitate* (simplă, deci în general neliniară) în spațiul original (în care sunt instanțele x_1, \dots, x_m). Ulterior (la punctul b), folosind același f , vom arăta că problema SVM admite cel puțin o soluție strict „fezabilă“ — deci, conform condiției lui Slater și o *soluție optimă* — pentru forma primală (și, de fapt, și pentru cea duală, dar asta pur și simplu *nu* are relevanță aici) și, în consecință, soluția optimă va satisface și ea această proprietate de separabilitate, care ne asigură că eroarea la antrenare este 0.

Găsiți valori pentru $\alpha_1, \dots, \alpha_m, w_0$, precum și pentru parametrul gaussian τ , astfel încât toate punctele x_i să fie clasificate corect de către clasificatorul $sign(f(x))$.

Sugestii:

1. Verificați faptul că, lucrând cu $y_i \in \{-1, +1\}$, predicția făcută pentru x_i de către $sign(f(x))$ va fi corectă dacă $|f(x_i) - y_i| < 1$. Altfel spus, verificați că are loc implicația $|f(x_i) - y_i| < 1 \Rightarrow y_i f(x_i) > 0$.³²²
2. Fixând $\alpha_i = 1$ pentru $i = 1, \dots, m$ și $w_0 = 0$, găsiți o valoare a lui τ pentru care inegalitatea $|f(x_i) - y_i| < 1$ să fie satisfăcută pentru $i = 1, \dots, m$.
- b. Presupunem că rulăm o SVM fără variabile de „destindere“ (engl., slack variables) folosind pentru parametrul τ valoarea pe care ati găsit-o la punctul precedent. Va obține oare acest clasificator (în mod necesar) eroare de antrenare zero? De ce da, sau de ce nu?
- c. Presupunem că antrenăm un C-SVM (adică o SVM cu variabile de „destindere“) pe datele specificate mai sus, folosind pentru parametrul τ valoarea pe care ati ales-o la punctul a, iar pentru parametrul C o valoare fixată în mod arbitrar, dar pe care nu o cunoaștem dinainte. Va obține oare acest clasificator (în mod necesar) eroare de antrenare zero? De ce da, sau de ce nu?

Răspuns:

- a. Sunt imediate următoarele echivalențe:

$$|f(x_i) - y_i| < 1 \Leftrightarrow -1 < f(x_i) - y_i < 1 \Leftrightarrow -1 + y_i < f(x_i) < 1 + y_i.$$

Pentru $y_i = -1$, partea dreaptă a ultimei inegalități duble de mai sus devine $f(x_i) < 0$. Pentru $y_i = 1$, partea stângă a aceleiași inegalități duble devine $f(x_i) > 0$. Așadar, dacă inegalitatea $|f(x_i) - y_i| < 1$ este adevărată, atunci instanța x_i este corect clasificată de către funcția $sign(f(x))$.

Conform *sugestiei* din enunț, vom considera $\alpha_i = 1$ pentru $i = 1, \dots, m$ și $w_0 = 0$. Pentru un exemplu de antrenament oarecare (x_i, y_i) , vom avea:

$$\begin{aligned} |f(x_i) - y_i| &= \left| \sum_{j=1}^m y_j K(x_j, x_i) - y_i \right| = \left| \sum_{j=1}^m y_j \exp(-\|x_j - x_i\|^2 / \tau^2) - y_i \right| \\ &= \left| y_i + \sum_{j \neq i} y_j \exp(-\|x_j - x_i\|^2 / \tau^2) - y_i \right| \\ &= \left| \sum_{j \neq i} y_j \exp(-\|x_j - x_i\|^2 / \tau^2) \right| \\ &\leq \sum_{j \neq i} |y_j \exp(-\|x_j - x_i\|^2 / \tau^2)| = \sum_{j \neq i} |y_j| \exp(-\|x_j - x_i\|^2 / \tau^2) \\ &= \sum_{j \neq i} \exp(-\|x_j - x_i\|^2 / \tau^2) \\ &\leq \sum_{j \neq i} \exp(-\varepsilon^2 / \tau^2) = (m - 1) \exp(-\varepsilon^2 / \tau^2). \end{aligned}$$

³²²Observație: Aici f poate fi gândit la cazul general ($f : \mathbb{R}^d \rightarrow \mathbb{R}$), nu doar ca separator produs de clasificatorul SVM cu nucleu RBF (conform expresiei (130)). Totuși, este util să observăm că atunci când vom alege pentru parametrul τ o valoare suficient de mică, din expresia (130) va rezulta că $f(x_i) \approx y_i$ (vedeți problema 68 de la capitolul *Fundamente*) și, prin urmare, $f(x_i) - y_i \approx 0$. Așadar, este natural să examinăm relația $|f(x_i) - y_i| < 1$.

Prima dintre inegalitățile de mai sus este datorată aplicării repetate a inegalității triunghiului ($|a+b| \leq |a|+|b|$), iar a doua inegalitate decurge din presupunerea că $\|x_j - x_i\| \geq \varepsilon$ pentru orice $i \neq j$. Așadar, pentru a avea $|f(x_i) - y_i| < 1$ pentru $i = 1, \dots, m$ este suficient să-l alegem pe τ astfel încât

$$(m-1) \exp(-\varepsilon^2/\tau^2) < 1,$$

sau, echivalent,³²³

$$\tau < \frac{\varepsilon}{\sqrt{\log(m-1)}}.$$

De exemplu, putem lua $\tau = \varepsilon/\sqrt{\log m}$.

Rezumând, am arătat până acum că există o instanțiere pentru variabilele duale ($\alpha_i = 1$) și pentru variabila primală w_0 (și anume, $w_0 = 0$), pentru care funcția $f(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + w_0 = (\sum_{i=1}^m \alpha_i y_i \Phi(x_i)) \cdot \Phi(x) + w_0$ separă perfect exemplele de antrenament date. Punctele b și c de mai jos vor analiza dacă *soluțiile optime* produse de SVM și respectiv C-SVM pe același set de antrenament vor avea și ele această proprietate. Vă reamintim că soluțiile optime pentru problema de optimizare [C-]SVM există, atât pentru forma primală cât și pentru forma duală — și ele sunt în relația $\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i K(x_i, x)$ — dacă, spre exemplu, este satisfăcută condiția lui Slater (a se vedea nota de subsol 281 de la pr. 10).

b. Datorită faptului că $f(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + w_0 = (\sum_{i=1}^m \alpha_i y_i \Phi(x_i)) \cdot \Phi(x) + w_0$, raționamentul prin care am făcut alegerea valorii parametrului τ de la punctul a este în sine o demonstrație a faptului că multimea $\{\Phi(x_i)\}_{i=1}^m$, unde Φ este maparea corespunzătoare nucleului RBF este *liniar separabilă*.

Acum vom arăta că putem pune în corespondență funcția f din proprietatea de separabilitate liniară obținută în spațiul de „trăsături“ (în care sunt $\Phi(x_1), \dots, \Phi(x_m)$) cu o anumită pereche w' , w'_0 care satisface condiția lui Slater în spațiul de trăsături (în care sunt $\Phi(x_1), \dots, \Phi(x_m)$).

Condiția lui Slater, relativă la problema de optimizare SVM este următoarea: există o soluție strict „fezabilă“, adică o asignare pentru w și w_0 , astfel încât restricțiile din problema primală SVM sunt satisfăcute cu inegalitate strictă: $y_i(w \cdot \Phi(x_i) + w_0) > 1$ pentru $i = 1, \dots, m$.

Fie $i \in \{1, \dots, m\}$, fixat. Luând $\alpha_1 = 1, \dots, \alpha_m = 1, w_0 = 0$, $f(x) = \sum_{j=1}^m \alpha_j y_j K(x_j, x) + w_0$ și τ ca la punctul a, vom avea $|f(x_i) - y_i| < 1$, deci $y_i f(x_i) > 0$. Notăm $w = \sum_{j=1}^m \alpha_j y_j \Phi(x_j)$. În consecință,

$$\begin{aligned} y_i(w \cdot \Phi(x_i) + w_0) &= y_i w \cdot \Phi(x_i) = y_i \sum_{j=1}^m \alpha_j y_j \Phi(x_j) \cdot \Phi(x_i) = y_i \sum_{j=1}^m \alpha_j y_j K(x_j, x_i) \\ &= y_i f(x_i) > 0. \end{aligned}$$

Așadar,

$$y_i(w \cdot \Phi(x_i) + w_0) = y_i \sum_{j=1}^m \alpha_j y_j K(x_j, x_i) > 0 \text{ pentru } i = 1, \dots, m.$$

³²³Presupunând $m > 1$.

Evident, putem multiplica toți α_i cu o constantă pozitivă astfel încât relația precedentă să devină

$$y_i \sum_{j=1}^m \alpha_j y_j K(x_j, x_i) > 1 \text{ pentru } i = 1, \dots, m.$$

Așadar, am găsit o soluție strict „fezabilă“ (w', w'_0) pentru problema noastră de optimizare. Prin urmare, condiția lui Slater este îndeplinită. În concluzie, soluția optimă (\bar{w}, \bar{w}_0) a acestei probleme va fi într-adevăr găsită de către SVM cu nucleu RBF, în vreme ce separabilitatea liniară a mulțimii $\{\Phi(x_i)\}_{i=1}^m$ din „spațiul de trăsături“ garantează că această soluție optimă produce eroare nulă la antrenare.

c. Clasificatorul C-SVM cu nucleu RBF nu va obține în mod neapărat eroare nulă la antrenare pe setul de date considerat, chiar dacă asignăm parametrului τ valoarea găsită la punctul a .

[În general, pentru problema de optimizare convexă C-SVM soluția optimă obținută $(\bar{w}, \bar{w}_0, \bar{\xi})$ pentru o valoare fixată a parametrului de „destindere“ C nu produce în mod necesar eroare la antrenare 0, chiar dacă datele de antrenament sunt liniar separabile. Se poate să existe un alt triplet (w', w'_0, ξ') , pentru care eroarea la antrenare rezultată să fie 0, dar pentru care $\frac{1}{2} \|w'\|^2 + C \sum_i \xi'_i > \frac{1}{2} \|\bar{w}\|^2 + C \sum_i \bar{\xi}_i$.]

De exemplu, putem considera cazul extrem în care $C = 0$. În acest caz, funcția obiectiv este $\frac{1}{2} \|w\|^2$ și, evident, $w = 0$ este soluție [optimă] a problemei de optimizare C-SVM, indiferent de alegerea valorii parametrului τ . Însă în acest caz eroarea la antrenare este 0 doar dacă toate instanțele au etichetele de același semn, și anume semnul lui w_0 .

27.

(SVM și C-SVM, o proprietate: Adevărat sau Fals?)

CMU, 2017 fall, Nina Balcan, midterm, pr. 1.2.bc

Observație importantă: În contextul formulărilor legate de [C-]SVM din această problemă, nu vom folosi termen liber (engl., bias term).

Fie $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ o mulțime de m instanțe care sunt separabile liniar [cu ajutorul unui separator care trece] prin originea sistemului de coordonate din \mathbb{R}^d . Considerăm de asemenea mulțimea S' , obținută din mulțimea S astfel: $S' = \{(cx_1, y_1), \dots, (cx_m, y_m)\}$, unde $c > 0$ este o constantă.³²⁴

Care dintre afirmațiile următoare sunt adevărate?

a. Atunci când SVM (se subîntelege, cu margine “hard”) ia ca input S și respectiv S' , se obține același separator decizional, modulo un factor constant. Adică, dacă \bar{w} și \bar{w}' sunt soluțiile produse de către SVM pentru inputul S și respectiv S' , atunci $\bar{w} = c_1 \bar{w}'$, unde c_1 este o anumită constantă.

b. Atunci când C-SVM (adică, SVM cu margine “soft”) ia ca input S și respectiv S' , se obține același separator decizional, modulo un factor constant.

Răspuns:

³²⁴LC: În textul de la CMU apărea $c > 1$.

a. Fie \bar{w} soluția [optimă a] problemei

$$\begin{aligned} \min_w \quad & \frac{1}{2} \|w\|^2 \\ \text{a. i. } \quad & y_i(w \cdot x_i) \geq 1, \text{ pentru } i = 1, \dots, m, \end{aligned} \tag{P}$$

și \bar{w}' soluția [optimă a] problemei

$$\begin{aligned} \min_{w'} \quad & \frac{1}{2} \|w'\|^2 \\ \text{a. i. } \quad & y_i(w' \cdot cx_i) \geq 1, \text{ pentru } i = 1, \dots, m. \end{aligned} \tag{P'}$$

Remarcați faptul că în formularea problemelor (P) și (P') — contrar formulării generale a problemei de optimizare SVM de la ex. 10 — nu am folosit termen liber.

Întrucât \bar{w}' este soluție a problemei (P'), rezultă că $y_i \bar{w}' \cdot (cx_i) \geq 1 \Leftrightarrow y_i(c\bar{w}') \cdot x_i \geq 1$ pentru $i = 1, \dots, m$. Prin urmare, $c\bar{w}'$ satisfac restricțiile problemei (P), pentru care soluția optimă este \bar{w} , ceea ce implică mai departe faptul că

$$\frac{1}{2} \|\bar{w}\|^2 \leq \frac{1}{2} \|c\bar{w}'\|^2 \Leftrightarrow \|\bar{w}\|^2 \leq \|c\bar{w}'\|^2 \Leftrightarrow \|\bar{w}\|^2 \leq c^2 \|\bar{w}'\|^2. \tag{131}$$

Invers, \bar{w} fiind soluție a problemei (P), implică $y_i \bar{w} \cdot x_i \geq 1 \Leftrightarrow y_i \left(\frac{1}{c} \bar{w} \right) \cdot (cx_i) \geq 1$ pentru $i = 1, \dots, m$. Prin urmare, $\frac{1}{c} \bar{w}$ satisfac restricțiile problemei (P'), pentru care soluția optimă este \bar{w}' , ceea ce implică mai departe faptul că

$$\frac{1}{2} \left| \frac{1}{c} \bar{w} \right|^2 \geq \frac{1}{2} \|\bar{w}'\|^2 \Leftrightarrow \frac{1}{c^2} \|\bar{w}\|^2 \geq \|\bar{w}'\|^2 \Leftrightarrow \|\bar{w}\|^2 \geq c^2 \|\bar{w}'\|^2. \tag{132}$$

Din relațiile (131) și (132) rezultă că $\|\bar{w}\|^2 = c^2 \|\bar{w}'\|^2$, deci $\|\bar{w}\| = c \|\bar{w}'\|$. În continuare vom arăta că $\bar{w} = c\bar{w}'$ (o egalitate mai „tare“ decât egalitatea pe care tocmai am demonstrat-o).

Am precizat mai sus că din faptul că \bar{w}' este soluție a problemei (P') rezultă că $c\bar{w}'$ satisfac restricțiile problemei (P), despre căreia funcție obiectiv știm acum că are valoarea optimă $\frac{1}{2} \|\bar{w}\|^2 = \frac{1}{2} c^2 \|\bar{w}'\|^2 = \frac{1}{2} \|c\bar{w}'\|^2$. Așadar, $c\bar{w}'$ este soluție optimă a problemei (P). Înținând cont de faptul că soluția problemei (P) este unică³²⁵ — ea existând în cazul în care setul de date de antrenament este separabil și negenerat³²⁶ —, rezultă că are loc egalitatea $c\bar{w}' = \bar{w}$. Așadar, afirmația din enunț este *adevărată*.

Observație: Separatorul ca atare, este unul și același: $\bar{w} \cdot x = 0 \Leftrightarrow c\bar{w}' \cdot x = 0 \Leftrightarrow \bar{w}' \cdot (cx) = 0$.

b. Răspunsul în acest caz este *negativ*, din cauza faptului că separatorii depind de valoarea parametrului C . Vom ilustra acest fapt folosind un *exemplu* simplu.

Vom considera setul de date de antrenament din enunțul problemei 18, și anume $S = \{(x_1 = -1, y_1 = -1), (x_2 = 1, y_2 = 1)\}$; el este separabil prin originea sistemului de coordinate.

³²⁵Motivația ține de faptul că funcția obiectiv a problemei (P) este strict convexă. Vedeti teorema 1 din articolul *Uniqueness of the SVM solution*, de Christopher Burges și David Crisp, 1998.

³²⁶Aceasta se traduce prin: $\exists w$ a. i. $w \cdot x_i = y_i$ (în condițiile acestei probleme) pentru $i = 1, \dots, m$ și respectiv $\exists i, j$ a. i. $y_i = 1$ și $y_j = -1$.

Observație importantă: Se poate arăta (făcând un raționament similar cu cel de la problema 13) că forma duală a problemei C-SVM în cazul în care nu se folosește termen liber w_0 este similară cu forma duală (D') de la problema 13.d, cu singura diferență că acum nu vom mai avea restricția $\sum_i \alpha_i y_i = 0$.

Procedând într-o manieră similară cu cea din prima parte a rezolvării problemei 18, se constată că

- pentru S , avem $L_D(\alpha) = \alpha_1 + \alpha_2 - \frac{1}{2}(\alpha_1 + \alpha_2)^2 = s - \frac{1}{2}s^2 = \frac{1}{2}s(2-s)$, unde $s \stackrel{\text{not.}}{=} \alpha_1 + \alpha_2$. Această funcție de gradul al doilea își atinge maximul pentru $s = 1$. Așadar, $\max_{0 \leq \alpha_i \leq C} L_D(\alpha)$ se obține pentru $s = 1$ dacă $C \geq 1$ (și, de asemenea, pentru $C = 1/2$ dacă $\alpha_1 = \alpha_2 = 1/2$), și respectiv pentru $s = C$ dacă $C \in (0, 1) \setminus \{1/2\}$;
- pentru $S' = \{(x_1 = -2, y_1 = -1), (x_2 = 2, y_2 = 1)\}$, avem $L_D(\alpha') = \alpha'_1 + \alpha'_2 - \frac{1}{2}4(\alpha'_1 + \alpha'_2)^2 = s' - 2s'^2 = s'(1-2s')$, unde $s' \stackrel{\text{not.}}{=} \alpha'_1 + \alpha'_2$. Această funcție de gradul al doilea își atinge maximul pentru $s' = 1/4$. Așadar, $\max_{0 \leq \alpha'_i \leq C} L_D(\alpha')$ se obține pentru $s = 1/4$ dacă $C \geq 1/4$ (și, de asemenea, pentru $C = 1/8$ dacă $\alpha_1 = \alpha_2 = 1/8$), respectiv pentru $s = C$ dacă $C \in (0, 1/4) \setminus \{1/8\}$.

Notând cu $\bar{\alpha}_i$, $i \in \{1, 2\}$ soluțiile optime pentru problema duală în cazul S și cu $\bar{\alpha}'_i$, $i \in \{1, 2\}$ soluțiile optime pentru problema duală în cazul S' , urmează că în funcție de valorile parametrului C avem următoarele corespondențe între \bar{w} și \bar{w}' :

- $C \geq 1$: pentru S avem $\bar{\alpha}_1 + \bar{\alpha}_2 = 1 \Rightarrow \bar{w} = \bar{\alpha}_1 y_1 x_1 + \bar{\alpha}_2 y_2 x_2 = \bar{\alpha}_1 + \bar{\alpha}_2 = 1$, iar pentru S' avem $\bar{\alpha}'_1 + \bar{\alpha}'_2 = 1/4 \Rightarrow \bar{w}' = \bar{\alpha}'_1 y_1 x_1 + \bar{\alpha}'_2 y_2 x_2 = 2(\bar{\alpha}'_1 + \bar{\alpha}'_2) = \frac{1}{2} = \frac{1}{2}\bar{w}$;
- $C = 1/2$: pentru S avem $\bar{\alpha}_1 = \bar{\alpha}_2 = 1/2 \Rightarrow \bar{w} = \bar{\alpha}_1 + \bar{\alpha}_2 = 1$, iar pentru S' avem $\bar{\alpha}'_1 + \bar{\alpha}'_2 = 1/4 \Rightarrow \bar{w}' = 2(\bar{\alpha}'_1 + \bar{\alpha}'_2) = 1/2 = \frac{1}{2}\bar{w}$;
- $C \in [1/4, 1] \setminus \{1/2\}$: pentru S avem $\bar{\alpha}_1 + \bar{\alpha}_2 = C \Rightarrow \bar{w} = \bar{\alpha}_1 + \bar{\alpha}_2 = C$, iar pentru S' avem $\bar{\alpha}'_1 + \bar{\alpha}'_2 = 1/4 \Rightarrow \bar{w}' = 2(\bar{\alpha}'_1 + \bar{\alpha}'_2) = 1/2$;
- $C \in (0, 1/4) \setminus \{1/8\}$: pentru S avem $\bar{\alpha}_1 + \bar{\alpha}_2 = C \Rightarrow \bar{w} = \bar{\alpha}_1 + \bar{\alpha}_2 = C$, iar pentru S' avem $\bar{\alpha}'_1 + \bar{\alpha}'_2 = C \Rightarrow \bar{w}' = 2(\bar{\alpha}'_1 + \bar{\alpha}'_2) = 2C = 2\bar{w}$;
- $C = 1/8$: pentru S avem $\bar{\alpha}_1 + \bar{\alpha}_2 = C = 1/8 \Rightarrow \bar{w} = \bar{\alpha}_1 + \bar{\alpha}_2 = 1/8$, iar pentru S' avem $\bar{\alpha}'_1 = \bar{\alpha}'_2 = 1/8 \Rightarrow \bar{w}' = 2(\bar{\alpha}'_1 + \bar{\alpha}'_2) = 1/2 = 4\bar{w}$.

În concluzie,

- nu avem o relație de tip $\bar{w}' = \frac{1}{2}\bar{w}$ (ca la punctul a) decât pentru cazurile $C \geq 1$ și $C = 1/2$;
- pentru $C \in (0, 1/4) \setminus \{1/8\}$ avem relația „inversată”, $\bar{w}' = 2\bar{w}$ [iar pentru $C = 1/8$ avem chiar $\bar{w}' = 4\bar{w}$];
- pentru $C \in [1/4, 1] \setminus \{1/2\}$ nu avem practic o legătură de genul celei sugerate în enunț între \bar{w} și \bar{w}' , întrucât $\bar{w} = C$ și $\bar{w}' = 1/2$.

CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, final exam, pr. 3.b

- b. La clasificare cu ajutorul SVM, atunci când transformăm instanțele de antrenament folosind mapări (Φ) ce corespund unor funcții-nucleu polinomiale (K) având gradul 1, 2, 3, ..., ne aşteptăm ca vectorii-suport să rămână în general aceiași.

Stanford, 2007 fall, Andrew Ng, practice midterm exam, pr. 6.f

- c. Presupunem că lucrăm cu o mașină cu vectori-suport de normă L_1 folosind parametrul de „destindere” $C > 0$ și că folosim un set de date de antrenament liniar separabile. Urmărind să minimizeze funcția obiectiv, SVM-ul va asigna variabilelor de „destindere” ξ_i valoarea 0, întrucât datele sunt liniar separabile. În consecință, soluția obținută (w, w_0) va fi aceeași, indiferent de valoarea folosită pentru parametrul C (presupunând, bineînteleas, că această valoare este strict pozitivă). Adevărat sau fals?

Răspuns:

- a. Adevărat. Învățarea unei funcții de clasificare prin intermediul unei mașini cu vectori-suport constă în rezolvarea unei probleme de optimizare convexă, cu funcția obiectiv de ordin pătratic și restricții liniare. Algoritmii / metodele de rezolvare computațională a acestui tip de probleme permit găsirea soluției optime, cu o eroare situată sub un prag oarecare, fixat $\varepsilon > 0$.
- b. Fals. Vectorii de trăsături ($\Phi(x_i)$) corespunzători nucleelor polinomiale sunt rezultatul aplicării unor funcții neliniare asupra vectorilor de intrare (x_i). Din această cauză, vectorii-suport pentru hiperplanul de separare optimală în spațiul de trăsături pot fi foarte diferiți de la o funcție-nucleu la alta.
- c. Fals. Chiar dacă datele de antrenament sunt liniar separabile, totuși poziția separatorului optimal poate fi afectată de outlier-e. Astfel, în funcție de valoarea parametrului C , mașina cu vectori-suport cu margine “soft” (C-SVM) poate decide să clasifice [chiar] în mod eronat unu sau mai multe exemple, dacă în acest fel se obține o margine mai mare. Valoarea parametrului C va influența modul în care se face acest compromis (engl., trade-off) între mărimea marginii ($1/\|w\|$) pe de o parte și suma variabilelor ξ_i pe de altă parte.

5.1.3 Alte probleme de optimizare de tip SVM

29.

(Clasificare n -ară cu SVM: SVM multiclass, cu margine “hard”)

*prelucrare de Liviu Ciortuz, după
MIT, 2009 fall, Tommi Jaakkola, lecture notes 5*

În practică, majoritatea problemelor de clasificare lucrează cu mai mult de două clase (de exemplu, identificarea persoanelor dintr-o imagine, precizarea fonemelor în procesarea vorbirii, asocierea genelor la anumite procese biologice, ori precizarea tipului de cancer pe baza analizei datelor preluate la nivel celular). În acest capitol, până aici ne-am ocupat doar de [task-uri de] clasificare binară. Totuși, metodele de tip SVM cu care ati făcut cunoștință — dar de asemenea și alte metode, precum Perceptronul —, pot fi folosite și pentru clasificare n -ară (engl., multiclass).

Una dintre modalitățile de a face clasificare n -ară, și anume metoda “one versus all”, constă în a reduce problema de clasificare multiclass la un set de task-uri de clasificare binară.

În această abordare, fiecare dintre clasificatorii binari este antrenat în mod independent față de ceilalți.³²⁷ O altă modalitate — cea care ne va interesa pe noi aici — este să încercăm să formulăm o *problemă de optimizare* per ansamblu, în aşa fel încât să putem antrena [în mod direct] un clasificator n -ar.

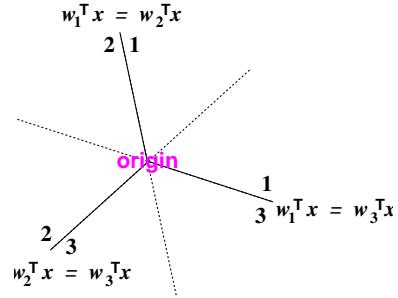
Vom folosi drept *componente* constitutive ale clasificatorului nostru n -ar mai mulți clasificatori liniari simpli care trec prin originea sistemului de coordinate,³²⁸ adică având forma analitică $f_j(x) = w_j \cdot x$, cu $x \in \mathbb{R}^d$ și $w_j \in \mathbb{R}^d$, pentru $j = 1, \dots, K$.³²⁹ Apoi vom impune cerința ca [la clasificarea fiecărei instanțe x] să „câștige“ acel clasificator care corespunde etichetei corecte, în sensul că valoarea *funcției* asociate aceluia clasificator [pentru x] să fie cea mai mare dintre toate valorile funcțiilor asociate diversilor clasificatori liniari [pentru același x], lăsând în plus un mic „spațiu de manevră“. Așadar, problema pe care vom încerca să o rezolvăm este următoarea:

$$\begin{aligned} & \min_{w^{(1)}, \dots, w^{(K)}} \frac{1}{2} \sum_{k=1}^K \|w^{(k)}\|^2 \\ \text{a. i. } & w^{(y_i)} \cdot x_i \geq w^{(y)} \cdot x_i + 1, \quad \text{pentru } i = 1, \dots, m \\ & \text{și orice } y \in \{1, \dots, K\}, \text{ cu } y \neq y_i, \end{aligned} \tag{133}$$

unde $w^{(1)}, \dots, w^{(K)} \in \mathbb{R}^d$, $x_i \in \mathbb{R}^d$, iar $y_i \in \{1, \dots, K\}$ pentru $i = 1, \dots, m$.

Observați că în această abordare componentele clasificatorului n -ar sunt antrenate împreună, iar predictia pentru o instanță oarecare x este determinată de cât de mult „sustine“ fiecare componentă-clasificator etichetarea corespunzătoare: $\hat{y} = \arg \max_y \{w^{(y)} \cdot x\}$.

Remarcați faptul că *zonele de decizie* care rezultă au o *interpretare geometrică* simplă. Putem obține fiecare dintre aceste zone ca *intersectie* a două *regiuni*, corespunzător mulțimilor de exemple pentru care o anumită etichetă este preferată în devoarea celorlalte etichete. Aceste regiuni sunt determinate în mod simplu prin linii (în general, hiperplane) care trec prin originea sistemului de coordinate: $w^{(y)} \cdot x = w^{(y')} \cdot x$, ca în figura alăturată, unde $y, y' \in \{1, 2, 3\}$. Linii punctate arată cum anume se extind aceste hiperplane în regiunile cărora le sunt asociate alte etichete. Regulile de decizie sunt desemnate prin linii continue.



Înainte de a trece la rezolvarea acestei probleme de optimizare în forma duală, ar fi util (pentru mai târziu) să scriem problema într-o formă ușor mai generală. Pentru aceasta, vom nota

$$w = (w^{(1)}, \dots, w^{(y)} \dots, w^{(K)})^\top, \quad \phi(x, y) = (0, \dots, x, \dots, 0)^\top,$$

³²⁷Vedeți MIT, 2009 fall, Tommi Jaakkola, lecture notes 5, pag. 7–11, precum și articolul *Reducing multiclass to binary* de Erin Allwein et al, din revista Journal of Machine Learning Research, 2000.

³²⁸La problema 59, care introduce versiunea SVM multiclass cu margine “soft”, această cerință va fi relaxată, în sensul că vom permite ca acești clasificatori liniari să aibă termeni liberi (b_k , pentru $k = 1, \dots, K$) eventual diferiți între ei. Pe lângă aceasta, vom cere ca ei să se intersecteze într-un același punct (ca și în cazul de față), dar nu neapărat în originea sistemului de coordinate.

³²⁹Pentru conveniență, deși am folosit mai sus termenul de clasificare n -ară, în continuare în formalismul matematic vom desemna numărul de clase cu simbolul K . (Așadar, se poate considera $K = n$.)

unde locația lui x din cadrul vectorului $\phi(x, y)$ garantează faptul că $w \cdot \phi(x, y) = w^{(y)} \cdot x$. În consecință, forma primală a problemei de optimizare *SVM multiclass* se poate scrie în mod echivalent (în raport cu forma anterioară (133)) astfel:

$$\min_w \frac{1}{2} \|w\|^2 \quad (134)$$

a. i. $w \cdot \phi(x_i, y_i) \geq w \cdot \phi(x_i, y) + 1_{\{y \neq y_i\}}$, pentru $i = 1, \dots, m$ și $y \in \{1, \dots, K\}$.

a. Deducreți forma duală a problemei de optimizare (134). Veți proceda similar cu modul în care s-a lucrat pentru clasificatorul binar SVM (vedeți pr. 10).

Sugestie: Introduceti multiplicatorii Lagrange $\alpha_{y,i} \geq 0$ pentru restricțiile de tip inegalitate, obțineți expresia lagrangeanului generalizat, găsiți valoarea lui w (ca funcție de multiplicatorii Lagrange $\alpha_{y,i}$) care minimizează acest lagrangean și apoi substituiți această valoare (pe care o veți nota cu $\hat{w}(\alpha)$) în expresia lagrangeanului, pentru a obține în sfârșit varianta duală.

b. Folosind \hat{w} , expresia vectorului de ponderi care reprezintă soluția optimă a problemei SVM multiclass, care a fost dedusă la punctul precedent în funcție de multiplicatorii Lagrange $\alpha_{y,i}$, scrieți *regula de predicție* pentru eticheta unei instanțe noi x , ținând cont că

$$\hat{y} = \arg \max_y \{\hat{w}^{(y)} \cdot x\}.$$

Răspuns:

a. Scriem mai întâi expresia lagrangeanului generalizat:

$$L_P(w, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \sum_{y=1}^K \alpha_{y,i} [w \cdot \phi(x_i, y_i) - w \cdot \phi(x_i, y) - 1_{\{y \neq y_i\}}].$$

Apoi, calculăm derivata parțială a acestei funcții în raport cu vectorul de ponderi w :

$$\frac{\partial}{\partial w} L_P(w, \alpha) = w - \sum_{i=1}^m \sum_{y=1}^K \alpha_{y,i} [\phi(x_i, y_i) - \phi(x_i, y)].$$

Conform condiției de optimalitate Karush-Kuhn-Tucker care se referă la atingerea minimului lui L_P în raport cu w , vom egala această derivată parțială cu vectorul 0 și vom obține soluția

$$\hat{w}(\alpha) = \sum_{i=1}^m \sum_{y=1}^K \alpha_{y,i} [\phi(x_i, y_i) - \phi(x_i, y)].$$

Acum vom substitui $\hat{w}(\alpha)$ în expresia lagrangeanului de mai sus, iar după ce vom simplifica termenii asemenea, vom obține imediat următoarea expresie pentru lagrangeanul dual:

$$\begin{aligned} L_D(\hat{w}, \alpha) &= \sum_{i=1}^m \sum_{y=1}^K \alpha_{y,i} 1_{\{y \neq y_i\}} - \\ &\quad \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \sum_{y=1}^K \sum_{y'=1}^K \alpha_{y,i} \alpha_{y',j} [\phi(x_i, y_i) - \phi(x_i, y)] \cdot [\phi(x_j, y_j) - \phi(x_j, y')]. \end{aligned}$$

Observați că această expresie poate fi simplificată și mai mult, dacă ținem cont că notația introdusă mai sus pentru $\phi(x, y)$ implică $\phi(x, y) \cdot \phi(x', y') = 1_{\{y=y'\}} (x \cdot x')$. Forma duală

a problemei de optimizare SVM multiclass are ca funcție obiectiv lagrangeanul L_D , iar ca restricții $\alpha_{y,i} \geq 0$ pentru $y = 1, \dots, K$ și $i = 1, \dots, m$.

b. Regula de decizie pentru clasificatorul SVM multiclass se poate scrie astfel:

$$\begin{aligned}\hat{y} &= \arg \max_y \{\hat{w}^{(y)} \cdot x\} \\ &= \arg \max_y \left\{ \sum_{i=1}^m \sum_{y'=1}^K \hat{\alpha}_{y';i} [\phi(x_i, y_i) - \phi(x_i, y')] \cdot \phi(x, y) \right\},\end{aligned}$$

unde $\hat{\alpha}_{y';i}$ sunt soluțiile problemei duale SVM multiclass. (Remarcați faptul că și în acest caz se pot face simplificări de genul celor indicate la punctul precedent.)

30.

(Problema SVM *one-class* (varianta “Max Margin”),
cazul marginii “hard”)

■ Stanford, 2007 fall, Andrew Ng, practice midterm exam, pr. 4

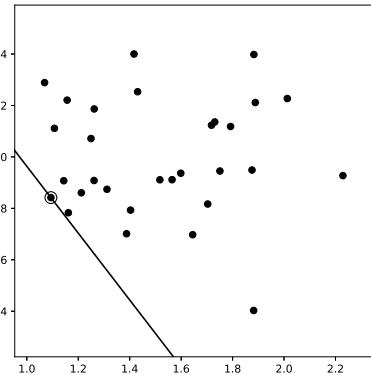
Considerăm un set de instanțe neetichetate $\{x_1, \dots, x_m\} \subset \mathbb{R}^d$.

Un algoritm SVM de tip *one-class* caută să identifice (dacă este posibil) o direcție $w \in \mathbb{R}^d$ care separă în sens maximal datele de originea sistemului de coordinate, ca în figura alăturată.³³⁰ Mai precis, acest algoritm rezolvă problema de optimizare (dată în formă primală):³³¹

$$\min_w \frac{1}{2} \|w\|^2$$

a. i. $w \cdot x_i \geq 1$, pentru $i = 1, \dots, m$.

O instanță nouă (de test) va fi etichetată cu + dacă $w \cdot x \geq 1$, și cu – în caz contrar.



Observație (1): Un astfel de algoritm de tip SVM este util pentru detectia anomalilor (engl., anomaly detection). În astfel de situații, ni se dă mai întâi un set de date care se consideră „normale“. Apoi ni se cere să decidem pentru alte instanțe dacă sunt (sau nu sunt) „anomalii“ (engl., outliers).

³³⁰Este de remarcat faptul că *one-class SVM* este o tehnică de învățare nesupervizată (așadar, nu de clasificare), întrucât ea nu necesită ca instanțele să fie etichetate.

În altă ordine de idei, putem formula problema *one-class SVM* într-o formă *mai generală*:

$$\min_{w, w_0} \frac{1}{2} \|w\|^2$$

a. i. $w \cdot x_i + w_0 \geq 1$, pentru $i = 1, \dots, m$.

Însă, dacă extindem (cum se face de obicei la rețele neuronale artificiale) orice instanță x_i cu o componentă $x_{i,0} = 1$, ajungem la forma (mai simplă!) din enunț.

³³¹La problema 62 vom da o altă variantă a problemei *one-class SVM*, definită cu ajutorul *sferelor de incluziune minimală* (engl., minimum enclosing ball, MEB). Pentru a putea distinge mai ușor cele două versiuni (una de cealaltă), vom numi varianta de aici *Max Margin*.

- a. Scrieți forma duală corespunzătoare problemei de optimizare *SVM one-class* (de tip *Max Margin*) de mai sus. Simplificați răspunsul cât mai mult posibil; evident, vectorul w nu trebuie să apară în rezultatul final. Verificați dacă forma primală satisfac condiția lui Slater.
- b. Am putea „kerneliza“ algoritmul *SVM one-class* (de tip *Max Margin*) atât la antrenare cât și la testare? Adică: dată fiind o funcție-nucleu K , este posibil ca după maparea corespunzătoare, variabilele x să apară
- atât în expresia lagrangeanului (L_D) care reprezintă funcția obiectiv a formei duale ale problemei SVM one-class,
 - cât și în funcția de decizie / clasificare pentru o instanță nouă x' , doar ca argumentele ale funcției-nucleu K ?
- c. Concepți un algoritm de tip SMO³³² care să rezolve problema duală obținută la punctul a. Ideea de bază a unui astfel de algoritm este ca la fiecare pas să se obțină soluția optimă pe cea mai mică dintre toate submulțimile posibile de variabile. Dați formulele analitice (engl., closed form formulas) pentru actualizarea variabilelor din această submulțime. Trebuie să justificați / explicați de ce este suficient să considerăm simultan respectivul număr de variabile la fiecare pas.

Răspuns:

- a. Putem verifica de la bun început faptul că problema de optimizare care a fost dată în enunț în formă primală satisfac condiția lui Slater.³³³ Dacă există un hiperplan care trece prin originea sistemului de coordonate și „lasă“ toate instanțele x_i cu $i = 1, \dots, m$ de o același parte a sa, aceasta înseamnă că există $w \in \mathbb{R}^d$ astfel încât $w \cdot x_i > 0$ pentru $i = 1, \dots, m$. Întrucât m este finit, înmulțind acest w cu o anumită constantă pozitivă obținem $w \cdot x_i > 1$ pentru $i = 1, \dots, m$, deci condiția lui Slater este satisfăcută.

Lagrangeanul generalizat care corespunde problemei primale date este:

$$L_P(w, \alpha) = \frac{1}{2}w \cdot w + \sum_{i=1}^m \alpha_i(1 - w \cdot x_i)$$

cu $\alpha_i \geq 0$ pentru $i = 1, \dots, m$.

Egalând cu 0 derivata parțială a lui L_P în raport cu w , obținem $w = \sum_{i=1}^m \alpha_i x_i$. Substițuind această egalitate în expresia lui L_P , vom avea:

$$\begin{aligned} L_D(\alpha) &= \frac{1}{2} \left(\sum_{i=1}^m \alpha_i x_i \right) \cdot \left(\sum_{j=1}^m \alpha_j x_j \right) + \sum_{i=1}^m \alpha_i \left(1 - \left(\sum_{j=1}^m \alpha_j x_j \right) \cdot x_i \right) \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j x_i \cdot x_j \end{aligned}$$

Așadar, forma lui L_D coincide cu cea de la problema SVM cu margine “hard” (vedeți problema 10.c), dar problema duală este în cazul de față mai simplă: $\max_{\alpha \geq 0} L_D(\alpha)$.

b. În ce privește kernel-izarea, este imediat că relativ la antrenare răspunsul este afirmativ, deoarece în expresia lagrangeanului L_D care a fost dedusă la punctul precedent x_i și x_j

³³²Vedeți problemele 23 și 22, precum și referințele bibliografice indicate acolo.

³³³Pentru formalizarea acestei condiții, vedeți nota de subsol 281 de la problema 10 (pag. 349).

apar doar ca [perechi de] factori în produsele scalare. Concret, după „mapare“ folosind funcția Φ corespunzătoare nucleului K , vom avea:

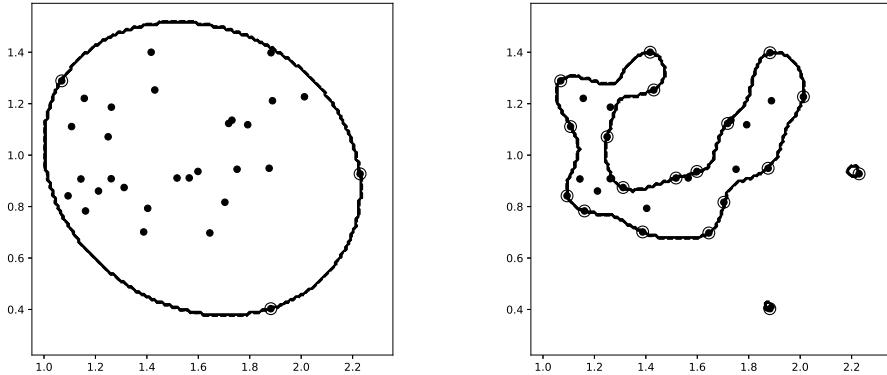
$$\begin{aligned} L_D(\alpha) &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j) \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j K(x_i, x_j). \end{aligned}$$

Relativ la testare, răspunsul este de asemenea afirmativ: dată fiind o instanță de test x' , ea va fi clasificată în funcție de valoarea expresiei

- $w \cdot x' = (\sum_{i=1}^m \alpha_i x_i) \cdot x' = \sum_{i=1}^m \alpha_i x_i \cdot x'$, în cazul în care nu se face mapare
- $w \cdot x' = \sum_{i=1}^m \alpha_i \Phi(x_i) \cdot \Phi(x') = \sum_{i=1}^m \alpha_i K(x_i, x')$, când se face mapare.

Observație (2): Imaginele de mai jos ilustrează rezultatul folosirii a două funcții-nucleu de tip RBF în conjuncție cu SVM one-class de tip Max Margin pe setul de date din enunț. Pentru rezultatul ilustrat în partea dreaptă, s-a lucrat cu o valoare [mai] mică pentru parametrul σ^2 din definiția nucleului RBF. Facem mențiunea că aici s-a folosit o formă ușor mai generală pentru problema SVM one-class (de tip Max Margin):³³⁴

$$\begin{aligned} \min_{w, \rho} & \left(\frac{1}{2} \|w\|^2 - \rho \right) \\ \text{a. î. } & w \cdot x_i \geq \rho, \text{ pentru } i = 1, \dots, m. \end{aligned}$$



c. Întrucât în formularea dată în enunț pentru problema one-class SVM nu se folosește termenul liber (engl., “bias”) w_0 , nu avem în forma duală a problemei de optimizare o restricție de tipul $\sum_{i=1}^n y_i \alpha_i = 0$ (cum este cazul la problemele 23 și 22). Așadar, vom folosi tot metoda creșterii pe coordonate (engl., coordinate ascent) ca în algoritmul clasic SMO, însă vom alege la fiecare iterație doar (câte) o variabilă Lagrange (α_i).

Din expresia lagrangeanului dual $L_D(\alpha)$ pe care l-am calculat la punctul a, obținem funcția pe care trebuie să-o optimizăm la iterația curentă:

$$L(\alpha_i) = \alpha_i + \sum_{j \neq i} \alpha_j \alpha_i x_i \cdot x_j - \frac{1}{2} \alpha_i^2 x_i \cdot x_i + const.$$

³³⁴LC: Această formă este foarte utilă pentru a înțelege cum anume s-a definit problema de optimizare ν -SVM de la pr. 32. Vedeți și pr. 61, unde se elaborează varianta “soft margin” pentru problema one-class de tip Max Margin.

Punctul de maxim este dat de soluția derivatei de ordinul întâi a acestei funcții:

$$\frac{\partial L(\alpha_i)}{\partial \alpha_i} = 0 \Leftrightarrow 1 - \sum_{j \neq i} \alpha_j x_j \cdot x_i - \alpha_i x_i \cdot x_i = 0 \Leftrightarrow \alpha_i^{new, unclipped} = \frac{1 - \sum_{j \neq i} \alpha_j x_j \cdot x_i}{x_i \cdot x_i}.$$

Tinând cont de restricția $\alpha_i \geq 0$, rezultă că noua valoare pe care o atribuim variabilei alese este $\alpha_i^{new, clipped} = \max \left\{ 0, \frac{1 - \sum_{j \neq i} \alpha_j x_j \cdot x_i}{x_i \cdot x_i} \right\}$.

Mai rămân de specificat:

- criteriul de selecție a variabilei „libere“; de preferință aceasta se va face în așa fel încât să se obțină o creștere cât mai mare a valorii funcției obiectiv de la o iterată la alta;
- criteriul de oprire a algoritmului; spre exemplu, atunci când creșterea funcției obiectiv de la o iterată la alta devine nesemnificativă este inutil să mai executăm noi iterării.

31. (O condiție suficientă pentru ca cele două tipuri de probleme de optimizare *one-class SVM*, și anume *Max Margin* și *minimum enclosing ball* (MEB), în varianta kernel-izată, să fie echivalente)
MIT, 2009 fall, Tommi Jaakkola, HW2, pr. 2.a

La exercițiul 30 am prezentat o metodă de detecție a „anomalilor“ dintr-un set de instanțe numită *one-class SVM*. Metoda respectivă se bazează pe separarea vectorilor de „trăsături“ (engl., feature vectors) față de originea sistemului de coordonate. Concret, am pornit de la un separator liniar care trece prin origine și are „margine“ (adică distanță) maximă în raport cu instanțele date. De aceea am numit această metodă problema *one-class Max Margin SVM*. Din punct de vedere matematic, varianta kernel-izată a acestei probleme a fost formulată — ca problemă de optimizare convexă cu restricții — astfel:³³⁵

$$\min_{w, \rho} \left(\frac{1}{2} \|w\|^2 - \rho \right) \quad (\text{Max Margin})$$

- a. i. $w \cdot \phi(x_i) \geq \rho$, pentru $i = 1, \dots, m$.

unde x_1, \dots, x_m sunt instanțe de antrenament, iar ϕ este o așa-numită funcție de „mapare“ a atributelor.

O altă modalitate prin care putem detecta „anomalii“ este să identificăm o hiper-sferă care să includă în sens minimal (engl., minimum enclosing ball, MEB) instanțele date sau, dacă folosim o funcție de „mapare“, imaginile instanțelor noastre în spațiul de „trăsături“ corespunzător. Matematic, putem formula această metodă în maniera următoare: date fiind instanțele x_1, \dots, x_m și o „mapare“ a trăsăturilor $\phi(x)$, considerăm problema de optimizare

$$\min_{R, w} R^2 \quad (\text{MEB})$$

- a. i. $\|w - \phi(x_i)\|^2 \leq R^2$, pentru $i = 1, \dots, m$.

Vă cerem să demonstrați că aceste două probleme devin identice în ipoteza că

$$\|\phi(x_i)\| = c \text{ pentru } i = 1, \dots, m,$$

³³⁵Vedeți *Observația* (2) de la pr. 30. Veți constata că varianta dată inițial în enunțul acelei probleme este mai simplă; este suficient de simplă pentru introducerea modelului *Max Margin*, însă este prea simplă pentru a o putea folosi în formularea proprietății pe care o vom da aici.

unde c este o constantă oarecare (reală, pozitivă).³³⁶ Cu alte cuvinte, soluția optimă \hat{w} obținută de una (oricare) dintre cele două probleme este soluție optimă și pentru celalătă problemă.

Sugestie: Deducreți forma duală pentru fiecare dintre cele două probleme de optimizare de mai sus (*Max Margin* și *MEB*). Apoi comparați cele două forme duale obținute, ținând cont de ipoteza $\|\phi(x_i)\| = c$ pentru c fixat și $i = 1, \dots, m$.

Observație: Ca o consecință directă a proprietății demonstrează în această problemă, putem afirma că graficele care au fost obținute la rezolvarea problemei 30.b (*Max Margin*) sunt identice cu cele care corespund soluțiilor problemei de optimizare *MEB* pe datele respective.

Răspuns:

Concret, strategia de rezolvare este următoarea: Se va constata ușor că atât în cazul problemei *Max Margin* (pentru detecția anomalilor) cât și în cazul problemei sferei de incluziune minimală (*MEB*), relația dintre soluția formei primale (\hat{w}) și soluția formei duale ($\hat{\alpha}$) este aceeași: $\hat{w} = \sum_{i=1}^m \hat{\alpha}_i \phi(x_i)$. Prin urmare, dacă presupunând adevărată relația $\|\phi(x_i)\| = c$ pentru $i = 1, \dots, m$ va rezulta că pentru ambele probleme de optimizare forma duală are exact aceeași soluție $\hat{\alpha}$, atunci este imediat că și soluțiile formelor primale ale celor două probleme coincid.³³⁷

Făcând calculele în maniera în care deja ne-am obișnuit, vom obține pentru problema *Max Margin* următoarea formă duală:³³⁸

$$\begin{aligned} & \max_{\alpha} \left(- \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j) \right) \\ & \text{a. i. } \alpha_i \geq 0 \text{ pentru } i = 1, \dots, m \text{ și } \sum_{i=1}^m \alpha_i = 1. \end{aligned}$$

³³⁶ *Observație importantă:* Această condiție este satisfăcută în cazul nucleului RBF, fiindcă $\phi(x)^2 = K(x, x) = \exp\left(-\frac{\|x - x\|^2}{2\sigma^2}\right) = 1$, pentru orice x și orice σ .

³³⁷ Ca și în cazul problemei *Max Margin* — pentru varianta ei [mai] simplă, am văzut deja justificarea la ex. 30.a —, se constată ușor că pentru problema *MEB* este satisfăcută condiția lui Slater.

³³⁸ Pe scurt, se procedează astfel:

$$\begin{aligned} L_P(w, \rho, \alpha) &= \frac{1}{2} w^2 - \rho - \sum_{i=1}^m \alpha_i (w \cdot \phi(x_i) - \rho) \\ \frac{\partial}{\partial w} L_P(w, \rho, \alpha) = 0 &\Leftrightarrow w - \sum_{i=1}^m \alpha_i \phi(x_i) = 0 \Leftrightarrow w = \sum_{i=1}^m \alpha_i \phi(x_i) \\ \frac{\partial}{\partial \rho} L_P(w, \rho, \alpha) = 0 &\Leftrightarrow -1 + \sum_{i=1}^m \alpha_i = 0 \Leftrightarrow \sum_{i=1}^m \alpha_i = 1 \\ L_D(\alpha) &= \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j \phi(x_i) \cdot \phi(x_j) - \rho - \sum_{i=1}^m \alpha_i \left(\left(\sum_{j=1}^m \alpha_j \phi(x_j) \right) \cdot \phi(x_i) - \rho \right) = \\ &= -\frac{1}{2} \sum_{i,j=1}^m \underbrace{\alpha_i \alpha_j \phi(x_i) \cdot \phi(x_j)}_1 - \rho + \rho \sum_{i=1}^m \alpha_i = -\frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j \underbrace{\phi(x_i) \cdot \phi(x_j)}_{K(x_i, x_j)} = -\frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j). \end{aligned}$$

În mod similar, pentru problema de optimizare MEB, forma duală va fi:³³⁹

$$\begin{aligned} \max_{\alpha} & \left(\sum_{i=1}^m \alpha_i K(x_i, x_i) - \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j) \right) \\ \text{a. i. } & \alpha_i \geq 0 \text{ pentru } i = 1, \dots, m \text{ și } \sum_{i=1}^m \alpha_i = 1. \end{aligned}$$

Întrucât $K(x_i, x_i) = \phi(x_i)^2 = \|\phi(x_i)\|^2 = c^2$ și $\sum_{i=1}^m \alpha_i = 1$, rezultă că prima sumă ($\sum_{i=1}^m \alpha_i K(x_i, x_i) = c^2$) din cadrul funcției obiectiv a problemei duale MEB nu are niciun efect asupra soluției optime a acestei probleme. În consecință, forma duală a problemei MEB este — în cazul $\|\phi(x_i)\| = c$ pentru $i = 1, \dots, m$ — echivalentă cu forma duală a problemei *Max Margin*. Ambele probleme vor produce același $\hat{\alpha}$ (pentru forma duală), și deci aceeași soluție optimă \hat{w} (pentru forma primală).

32. (Problema ν -SVM)

*prelucrare de Liviu Ciortuz, după
■ B. Schölkopf, A. Smola, “Learning with Kernels”,
MIT Press, 2002, pag. 206-209*

Parametrul de „destindere” $C > 0$, din forma problemei SVM cu margine “soft” (introdus de catre Cortes și Vapnik, în *Support Vector Networks*, 1995),³⁴⁰ permite realizarea unui compromis între două obiective antagoniste: maximizarea marginii³⁴¹ și minimizarea ero-rii la antrenare. La valori mari ale lui C rezultă un nivel scăzut al sumei erorilor în raport

³³⁹Din nou, pe scurt,

$$\begin{aligned} L_P(w, R, \alpha) &= R^2 + \sum_{i=1}^m \alpha_i [(w - \phi(x_i))^2 - R^2] \\ \frac{\partial}{\partial w} L_P(w, R, \alpha) = 0 &\Leftrightarrow \sum_{i=1}^m \alpha_i 2(w - \phi(x_i)) = 0 \Leftrightarrow \sum_{i=1}^m \alpha_i w = \sum_{i=1}^m \alpha_i \phi(x_i) \Leftrightarrow w = \frac{\sum_{i=1}^m \alpha_i \phi(x_i)}{\sum_{i=1}^m \alpha_i} \\ \frac{\partial}{\partial R} L_P(w, \rho, \alpha) = 0 &\Leftrightarrow 2R - \sum_{i=1}^m \alpha_i 2R = 0 \stackrel{R \neq 0}{\Leftrightarrow} \sum_{i=1}^m \alpha_i = 1 \\ &\Rightarrow w = \sum_{i=1}^m \alpha_i \phi(x_i) \text{ și} \\ L_D(\alpha) &= R^2 + \sum_{i=1}^m \alpha_i \left[\underbrace{\left(\sum_{j=1}^m \alpha_j \phi(x_j) - \phi(x_i) \right)^2}_{w} - R^2 \right] \\ &= R^2 - R^2 \underbrace{\sum_{i=1}^m \alpha_i}_{1} + w^2 \underbrace{\sum_{i=1}^m \alpha_i}_{1} - 2 \sum_{i,j=1}^m \alpha_i \alpha_j \underbrace{\phi(x_i) \cdot \phi(x_j)}_{K(x_i, x_j)} + \sum_{i=1}^m \alpha_i \underbrace{\phi(x_i)^2}_{K(x_i, x_i)} \\ &= \sum_{i,j=1}^m \alpha_i \alpha_j \underbrace{\phi(x_i) \cdot \phi(x_j)}_{K(x_i, x_j)} - 2 \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j) + \sum_{i=1}^m \alpha_i K(x_i, x_i) \\ &= - \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j) + \sum_{i=1}^m \alpha_i K(x_i, x_i). \end{aligned}$$

³⁴⁰Vedeți problema 13.

³⁴¹Definită ca distanța de la hiperplanul de separare optimală $w \cdot x + w_0 = 0$ până la vectorii-suport x_i pentru care $(w \cdot x_i + w_0)y_i = 1$.

cu marginea (și anume, $\sum_{i=1}^m \xi_i$, în notația ușuală). Invers, la valori mici ale lui C rezultă un nivel ridicat al sumei erorilor. Totuși, semnificația parametrului C este prea puțin intuitivă, iar valoarea sa nu poate fi determinată a priori.

De aceea, în locul acestei variante de SVM cu margine “soft”, B. Schölkopf, A. Smola, R. Williamson și P. Bartlett au propus în articolul *New Support Vector Machines*³⁴² o abordare diferită, în care se folosește un alt parametru numeric, ν , astfel încât dacă m reprezintă numărul instanțelor de antrenament, atunci νm va limita superior numărul de erori produse la antrenare.³⁴³ Se poate demonstra că νm este totodată o margine inferioară pentru numărul de vectori-suport.³⁴⁴

Varianta aceasta este cunoscută sub numele de ν -SVM și este caracterizată de forma primală următoare:

$$\begin{aligned} & \min_{w, w_0, \xi, \rho} \left(\frac{1}{2} \|w\|^2 - \nu \rho + \frac{1}{m} \sum_{i=1}^m \xi_i \right) \\ \text{a. i. } & y_i(w \cdot x_i + w_0) \geq \rho - \xi_i, \text{ pentru } i = 1, \dots, m \\ & \xi_i \geq 0 \text{ pentru } i = 1, \dots, m \\ & \rho \geq 0. \end{aligned} \tag{P''}$$

De remarcat prezența variabilei suplimentare ρ , care va trebui să fie supusă procesului de optimizare la fel ca și variabilele w , w_0 și $\xi \stackrel{\text{not.}}{=} (\xi_1, \dots, \xi_m)$.³⁴⁵

- a. Derivați forma duală corespunzătoare problemei ν -SVM. Simplificați rezultatul cât mai mult posibil.
- b. Stabiliți relațiile de legătură între \bar{w} , \bar{w}_0 , $\bar{\rho}$, prin care am notat soluțiile problemei (P''), și soluțiile problemei duale de la punctul a.
- c. Care este regula de clasificare a unei instanțe de test oarecare x' în acest model?

Răspuns:

- a. Vom urma liniile „metodologice“ care au fost folosite la problemele 10 și 13. Mai întâi, este ușor de verificat faptul că problema de optimizare (P'') satisfacă condiția lui Slater:³⁴⁶ luând $w = 0$, $w_0 = 0$, $\rho = 0$ și $\xi_i = 1$, restricțiile $y_i(w \cdot x_i + w_0) > \rho - \xi_i$ sunt îndeplinite,

³⁴²Publicat în revista *Neural Computation*, 12:1207-1245, 2000. (Vedeți și și *A Tutorial on ν -Support Vector Machines* de Pai-Hsuen Chen, Chih-Jen Lin, și Bernhard Schoelkopf, în vol. *Applied Stochastic Models in Business and Industry*, ed. Wiley InterScience, 2005.)

³⁴³LC: La problema 19 am arătat că în contextul problemei de optimizare C-SVM numărul de erori comise la antrenare este mărginit superior de către suma variabilelor de „destindere“ ($\sum_i \xi_i$). Similar, pentru problema (P'') de mai jos se poate arăta imediat, analizând doar(!) forma restricțiilor, că numărul de erori comise la antrenare este aici mărginit superior de $\frac{1}{\rho} \sum_i \xi_i$.

Prin urmare, dacă impunem condiția $\nu m \leq \frac{1}{\rho} \sum_i \xi_i$ rezultă $\nu \rho \leq \frac{1}{m} \sum_i \xi_i$. Aceasta explică forma funcției obiectiv din problema (P'') de mai jos.

³⁴⁴Vedeți *Observația* de la finalul rezolvării punctului a de mai jos.

³⁴⁵Din forma restricțiilor liniare rezultă că distanța de la separatorul optimal la vectorii-suport de pe margine (engl., bound support vectors) — adică acei vectori-suport pentru care multiplicatorii Lagrange corespunzători vor apartine intervalului $(0, \frac{1}{m})$, conform problemei duale (D'') de mai jos — va fi $\frac{\rho}{\|w\|}$. (Vedeți relația (122) de la problema 13.)

³⁴⁶Pentru formalizarea acestei condiții, vedeți nota de subsol 281 de la problema 10 (pag. 349).

pentru $i = 1, \dots, m$. Prin urmare, vom putea opta ca în loc să rezolvăm problema (P'') să rezolvăm duala ei.

Apoi, lagrangeanul generalizat pentru problema (P'') este

$$\begin{aligned} L_P(w, w_0, \xi, \rho, \alpha, \beta, \delta) &= \frac{1}{2} \|w\|^2 - \nu\rho + \frac{1}{m} \sum_{i=1}^m \xi_i \\ &\quad - \sum_{i=1}^m \alpha_i (y_i(w \cdot x_i + w_0) - \rho + \xi_i) - \sum_{i=1}^m \beta_i \xi_i - \delta\rho, \end{aligned}$$

unde α_i, β_i și δ sunt multiplicatori Lagrange, toți trebuind să satisfacă restricția de neneativitate.

Calculând derivatele parțiale ale lui L_P în raport cu variabilele primale w, w_0, ξ și ρ și apoi egalându-le cu 0, vom obține imediat:

$$\begin{aligned} w &= \sum_{i=1}^m \alpha_i y_i x_i \\ \sum_{i=1}^m \alpha_i y_i &= 0 \\ \alpha_i + \beta_i &= \frac{1}{m} \text{ pentru } i = 1, \dots, m \\ \sum_{i=1}^m \alpha_i - \delta &= \nu. \end{aligned}$$

Din relația $\alpha_i + \beta_i = \frac{1}{m}$, ținând cont că $\alpha_i \geq 0$ și $\beta_i \geq 0$, rezultă $\alpha_i \in \left[0, \frac{1}{m}\right]$ și $\beta_i \in \left[0, \frac{1}{m}\right]$ pentru $1, \dots, m$. De asemenea, știind că $\delta \geq 0$, din relația $\sum_{i=1}^m \alpha_i - \delta = \nu$ rezultă că $\sum_{i=1}^m \alpha_i \geq \nu$.

Condițiile de complementaritate Karush-Kuhn-Tucker sunt: $\alpha_i(y_i(w \cdot x_i + w_0) - \rho + \xi_i) = 0$, $\beta_i \xi_i = 0$ și $\delta\rho = 0$.

Substituind $w = \sum_{i=1}^m \alpha_i y_i x_i$ în expresia lui L_P și apoi făcând diversele simplificări posibile, va rezulta că forma duală corespunzătoare problemei (P'') este:

$$\begin{aligned} \max_{\alpha} & \left(-\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right) \\ \text{a. î.} & \quad 0 \leq \alpha_i \leq \frac{1}{m} \text{ pentru } i = 1, \dots, m \\ & \quad \sum_{i=1}^m \alpha_i y_i = 0 \\ & \quad \sum_{i=1}^m \alpha_i \geq \nu. \end{aligned} \tag{D''}$$

De remarcat că

- multiplicatorii Lagrange β_i și δ nu apar în (D'') ;
- în funcția obiectiv a problemei duale (D'') nu apare termenul $\sum_{i=1}^m \alpha_i$, care era prezent în funcția obiectiv a problemei duale atât pentru SVM cu margine “hard” cât și pentru C-SVM;
- în schimb, în partea de restricții apare condiția suplimentară $\sum_{i=1}^m \alpha_i \geq \nu$.

Observație: Din relația $0 \leq \alpha_i \leq \frac{1}{m}$ rezultă imediat că $\frac{\#SV}{m} \geq \sum_{i=1}^m \alpha_i \geq \nu$, deci $\#SV \geq \nu m$, unde prin $\#SV$ am notat numărul vectorilor-suport.

b. În ce privește legătura dintre soluțiile formei primale (P'') și cele ale formei duale (D''), avem mai întâi $\bar{w} = \sum_{i=1}^m \bar{\alpha}_i y_i x_i$. Apoi, din condițiile de complementaritate Karush-Kuhn-Tucker deducem că dacă există un $\bar{\alpha}_i$ astfel încât $0 < \bar{\alpha}_i < \frac{1}{m}$, atunci $\bar{\beta}_i > 0$ și deci $\bar{\xi}_i = 0$. De asemenea, $\bar{\alpha}_i > 0$ implică $y_i(\bar{w} \cdot x_i + \bar{w}_0) - \bar{\rho} + \bar{\xi}_i = 0$, de unde rezultă $y_i(\bar{w} \cdot x_i + \bar{w}_0) = \bar{\rho}$.

Folosind această ultimă relație, valorile optime \bar{w}_0 și $\bar{\rho}$ se vor determina astfel: dacă avem x_+ o instanță pozitivă și x_- o instanță negativă astfel încât multiplicatorii Lagrange corespunzători sunt în intervalul $(0, \frac{1}{m})$,³⁴⁷ atunci $\bar{w} \cdot x_+ + \bar{w}_0 = \bar{\rho}$ și $\bar{w} \cdot x_- + \bar{w}_0 = -\bar{\rho}$. Aceste ultime două ecuații formează un sistem din care se obțin imediat \bar{w}_0 și $\bar{\rho}$:³⁴⁸

$$\begin{aligned}\bar{w}_0 &= -\frac{1}{2}\bar{w} \cdot (x_+ + x_-) \\ \bar{\rho} &= \frac{1}{2}\bar{w} \cdot (x_+ - x_-)\end{aligned}$$

Observație: În articolul despre ν -SVM citat în enunț, autorii extind acest procedeu de calcul pentru valorile \bar{w}_0 și $\bar{\rho}$ la mai multe perechi de instanțe pozitive și respectiv negative, pentru a obține un rezultat cât mai robust.

c. Dată fiind o instanță nouă (de test) x' , ea va fi clasificată conform expresiei

$$\text{sign}\left(\sum_{i=1}^m \bar{\alpha}_i y_i x_i \cdot x' + \bar{w}_0\right).$$

33. (SVR — Regresie cu vectori-suport, varianta “hard-margin”, adică, fără variabile de „destindere“)
 ■ prelucrare de Liviu Ciortuz, după
 ■ Stanford, 2014 fall, Andrew Ng, midterm, pr. 4

Până acum, am văzut cum anume putem face clasificare cu algoritmul [C-]SVM. În acest exercițiu vom studia o variantă a acestui algoritm, care servește pentru a face regresie. Așadar, etichetele asociate aici instanțelor de antrenament vor avea valori continue, $y \in \mathbb{R}$. În mod natural, noul algoritm poartă numele de Regresie cu Vectori-Suport (engl., Support Vector Regression, SVR).³⁴⁹

Presupunem că avem un set de date de antrenament $\{(x_1, y_1), \dots, (x_m, y_m)\}$, cu $x_i \in \mathbb{R}^{n+1}$ și $y_i \in \mathbb{R}$ pentru $i = 1, \dots, m$. Urmărind să găsim o ipoteză [de regresie] de forma $h_{w,b}(x) = w \cdot x + b$, vom scrie următoarea problemă de optimizare (convexă):

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad (135)$$

$$\text{s. t. } y_i - (w \cdot x_i + b) \leq \varepsilon \text{ pentru } i = 1, \dots, m \quad (135)$$

$$(w \cdot x_i + b) - y_i \leq \varepsilon \text{ pentru } i = 1, \dots, m, \quad (136)$$

³⁴⁷De remarcat că dacă există una din cele două instanțe, atunci în mod necesar există și cea de-a doua, fiindcă $\sum_i \bar{\alpha}_i y_i = 0$.

³⁴⁸Dacă toți $\bar{\alpha}_i$ sunt 0, atunci $\bar{w} = 0$, ceea ce este în afara discuției, după cum am justificat la problema 10. Rămâne de tratat cazul în care pentru orice $i \in \{1, \dots, m\}$ avem fie $\bar{\alpha}_i = 0$ fie $\bar{\alpha}_i = \frac{1}{m}$, știind că există cel puțin un $\bar{\alpha}_i$ care are valoarea $\frac{1}{m}$. Acest caz se tratează similar cu cazul corespunzător de la problema 13.

³⁴⁹Puteți consulta articolul *A tutorial on support vector regression* de Alex Smola și Bernhard Schoelkopf, publicat în revista Statistics and Computing, 14:199–222, 2004.

unde $\varepsilon > 0$ are o valoare dată, fixată.

Observații:

1. Inegalitățile (135) și (136) se pot scrie în mod combinat sub forma

$$-\varepsilon \leq y_i - (w \cdot x_i + b) \leq \varepsilon \Leftrightarrow |y_i - (w \cdot x_i + b)| \leq \varepsilon.$$

Semnificația analitică a acestei inegalități este imediată. Așadar, restricția asupra *marginal functională* de la problema de optimizare SVM a fost modificată aici în aşa fel încât să se refere la diferența dintre valorile reale ale lui y și outputul produs de ipoteza pe care urmărim să o învățăm folosind SVR.

2. În mod similar cu problema [C]-SVM, în noua problemă de optimizare se cere ca valoarea lui $\|w\|$ să fie [cât mai] mică. Minimizarea lui $\|w\|$ corespunde maximizării distanței $(\varepsilon/\|w\|)$ dintre hiperplanul reprezentat de ipoteza $w \cdot x + b$ și marginile $w \cdot x + b = \varepsilon$ și $w \cdot x + b = -\varepsilon$ între care trebuie să se situeze toate valorile y_i .³⁵⁰

a. Scrieți lagrangeanul corespunzător problemei de optimizare (SVR) de mai sus. Vă recomandăm să folosiți două seturi de variabile / multiplicatori Lagrange, α_i și α_i^* , cu $i = 1, \dots, m$, corespunzător restricțiilor reprezentate de cele două inegalități (care au fost etichetate cu (135) și (136)). Prin urmare, funcția lagrangeană va fi de forma $L(w, b, \alpha, \alpha^*)$.

b. Obțineți forma duală a problemei de optimizare. (Va trebui să calculați derivatele parțiale ale lagrangeanului în raport cu w și respectiv cu b .)

c. Demonstrați că acest algoritm poate fi kernel-izat. Pentru aceasta, va trebui să arătați că (i) funcția obiectiv pentru forma duală a problemei de optimizare poate fi scrisă folosind (doar) produse scalare între instanțele de antrenament, și (ii) la faza de testare, date fiind o instanță nouă x , ipoteza $h_{w,b}(x)$ poate fi calculată folosind de asemenea (doar) produse scalare.

Răspuns:

a. Fie $\alpha_i, \alpha_i^* \geq 0$ ($i = 1, \dots, m$) multiplicatorii Lagrange pentru restricțiile (135) și respectiv (136). Atunci, lagrangeanul poate fi scris astfel:

$$L(w, b, \alpha, \alpha^*) = \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (y_i - w \cdot x_i - b - \varepsilon) + \sum_{i=1}^m \alpha_i^* (-y_i + w \cdot x_i + b - \varepsilon).$$

b. Mai întâi, precizăm că funcția obiectiv pentru forma duală a problemei de optimizare SVR este definită astfel:

$$L_D(\alpha, \alpha^*) = \min_{w, b} L(w, b, \alpha, \alpha^*).$$

Acum, calculând derivatele lagrangeanului în raport cu variabilele primale și egalându-le apoi cu 0, vom avea:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^m (\alpha_i - \alpha_i^*) x_i = 0 \Rightarrow w = \sum_{i=1}^m (\alpha_i - \alpha_i^*) x_i \quad (137)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^m (\alpha_i^* - \alpha_i) = 0. \quad (138)$$

³⁵⁰De asemenea, minimizarea lui $\|w\|$ corespunde obiectivului de a reduce overfitting-ul, așa cum știm și de la celelalte metode de regresie studiate.

Substituind aceste două relații în $L(w, b, \alpha, \alpha^*)$, vom obține:

$$\begin{aligned}
L_D(\alpha, \alpha^*) &= \frac{1}{2} \|w\|^2 + \sum_{i=1}^m \alpha_i (y_i - w \cdot x_i - b - \varepsilon) + \sum_{i=1}^m \alpha_i^* (-y_i + w \cdot x_i + b - \varepsilon) \\
&= \frac{1}{2} \|w\|^2 - \varepsilon \sum_{i=1}^m (\alpha_i + \alpha_i^*) + \sum_{i=1}^m y_i (\alpha_i - \alpha_i^*) \\
&\quad - \sum_{i=1}^m (\alpha_i - \alpha_i^*) w \cdot x_i - b \underbrace{\sum_{i=1}^m (\alpha_i - \alpha_i^*)}_{0} \\
&\stackrel{(137)}{=} \frac{1}{2} \left\| \sum_{i=1}^m (\alpha_i - \alpha_i^*) x_i \right\|^2 - \varepsilon \sum_{i=1}^m (\alpha_i + \alpha_i^*) + \sum_{i=1}^m y_i (\alpha_i - \alpha_i^*) \\
&\quad - \sum_{i=1}^m (\alpha_i - \alpha_i^*) \sum_{j=1}^m (\alpha_j - \alpha_j^*) x_j \cdot x_i \\
&= -\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i \cdot x_j - \varepsilon \sum_{i=1}^m (\alpha_i + \alpha_i^*) + \sum_{i=1}^m y_i (\alpha_i - \alpha_i^*).
\end{aligned}$$

Așadar, problema duală SVR poate fi formulată astfel:

$$\begin{aligned}
&\max_{\alpha, \alpha^*} \left(-\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i \cdot x_j - \varepsilon \sum_{i=1}^m (\alpha_i + \alpha_i^*) + \sum_{i=1}^m y_i (\alpha_i - \alpha_i^*) \right) \\
&\text{a. i. } \sum_{i=1}^m (\alpha_i - \alpha_i^*) = 0 \\
&\quad \alpha_i, \alpha_i^* \geq 0 \text{ pentru } i = 1, \dots, m.
\end{aligned}$$

c. Se observă că în expresia lagrangeanului dual L_D instanțele de antrenament apar întotdeauna în produse scalare de forma $x_i \cdot x_j$. În mod similar, atunci când vrem să facem predicție pentru o instanță oarecare x , vom avea:

$$w \cdot x + b \stackrel{(137)}{=} \sum_{i=1}^m (\alpha_i - \alpha_i^*) x_i \cdot x + b.$$

Așadar, algoritmul SVR poate fi kernel-izat.

5.2 Probleme propuse

5.2.1 SVM cu margine “hard”

34. (Calcularea distanței de la un hiperplan la originea sistemului de coordonate; rezolvare cu metoda vectorială și cu metoda multiplicatorilor lui Lagrange)
prelucrare de Liviu Ciortuz, după CMU, 2018 spring, Nina Balcan, HW0, pr. “Geometry”

Considerăm hiperplanul de ecuație $w \cdot x + b = 0$, unde w și x sunt din \mathbb{R}^d , iar $b \in \mathbb{R}$.

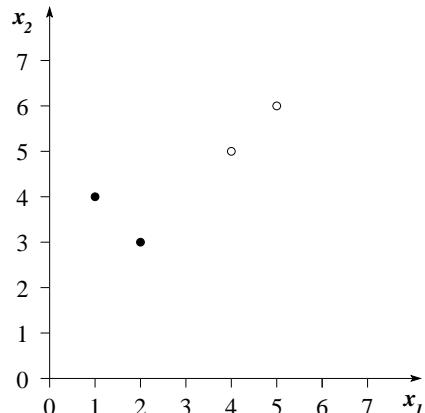
Demonstrați că distanța geometrică [deci fără semn] de la originea sistemului de coordonate până la [cel mai apropiat punct de pe] hiperplanul de ecuație $w \cdot x + b = 0$ este $\frac{|b|}{\|w\|}$.

Veți face demonstrația atât în manieră vectorială³⁵¹ cât și cu ajutorul metodei multiplicatorilor lui Lagrange.³⁵²

35. (SVM liniară: aplicare pe date din \mathbb{R}^2)
CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final exam, pr. 7.b

Presupunem că antrenați o mașină cu vectori-suport pe setul de date din figura alăturată. Acest set de date constă din două exemple cu eticheta +1 (aceste exemple sunt marcate cu semnul •) și două exemple cu eticheta -1 (marcate cu ○).

- Care este ecuația corespunzătoare separatorului optimal?
- Trasați separatorul optimal și încercuiți vectorii-suport.

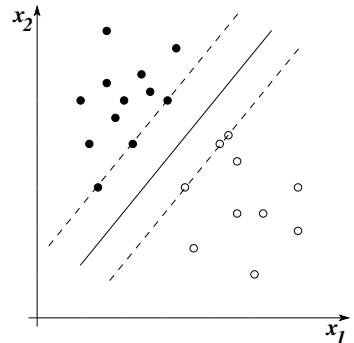


³⁵¹Puteți prelua ideea rezolvării de la problema 1, însă vă cerem să nu preluati pur și simplu formula care a fost obținută acolo și s-o aplicați la cazul de față. Dacă doriți să lucrați în *cazul particular* $d = 2$, puteți folosi mijloace obișnuite de analiză matematică, în sprijinul proprietățile funcției polynomiale de gradul al doilea.

³⁵²Pentru o scurtă introducere la metoda multiplicatorilor lui Lagrange, veДЕti notele de sub-sol 763, 764 de la problema 73 și problemele 153 și 76 din secțiunea *Metode de optimizare* de la capitolul de *Fundamente*.

Observație: În problema de optimizare care corespunde formulării din enunț nu avem restricții de tip inegalitate, deci veți putea rezolva ușor problema de optimizare în *forma sa primală*, făcând apel [nu neapărat în mod explicit] la condițiile Karush-Kuhn-Tucker.

36. (Separabilitate în \mathbb{R}^2 ; SVM liniară; calculul erorii la CVLOO)
CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, final exam, pr. 3.f



Care este eroarea la cross-validation cu metoda "Leave-One-Out" atunci când folosim o mașină cu vectori-suport ca în figura alăturată?

37. (SVM liniară în \mathbb{R}^2 , forma primală; determinarea hiperplanului de separare optimală)
CMU, 2009 spring, Ziv Bar-Joseph, HW3, pr. 4.1

Considerăm că în \mathbb{R}^2 sunt date două puncte: (x_1, y_1) cu eticheta +1 și (x_2, y_2) cu eticheta -1. Care este separatorul decizional pe care îl vom obține rulând o SVM liniară pe setul de date de antrenament compus din aceste două puncte? Calculați expresia analitică a acestui separator.

38. (SVM — problema de optimizare în forma primală: familiarizare cu noțiunile de margine și separator optimal în spațiul de „trăsături“)
*CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW4, pr. 4.5
MIT, 2004 fall, Tommi Jaakkola, HW3, pr. 1.3*

Considerăm un set de date din \mathbb{R} , foarte simplu, constând din numai două exemple de antrenament:

$$(x_1 = 0, y_1 = -1) \text{ și } (x_2 = \sqrt{2}, y_2 = 1).$$

Vom folosi o funcție-nucleu polinomială de ordinul al doilea, mai precis vom pune în corespondență fiecare instanță x cu vectorul

$$\Phi(x) = (1, \sqrt{2}x, x^2).$$

Vrem să găsim soluția $\hat{w} = (\hat{w}_1, \hat{w}_2, \hat{w}_3)$ și \hat{w}_0 a problemei de optimizare SVM [în aşa-numitul spațiu de „trăsături“ determinat de transformarea Φ]:

$$\begin{aligned} & \min_{w, w_0} \frac{1}{2} \|w\|^2 \\ & \text{astfel încât } y_1(w \cdot \Phi(x_1) + w_0) \geq 1 \\ & \quad y_2(w \cdot \Phi(x_2) + w_0) \geq 1. \end{aligned}$$

- a. Folosind ceea ce știți despre [granița de] separare cu margine maximală, indicați un vector care are aceeași direcție cu vectorul-soluție \hat{w} .³⁵³
- b. Cât este valoarea *marginii de separare* pe care o obținem pentru aceste date (evident, în spațiul de „trăsături“ indicat mai sus)?
- c. Făcând legătura dintre marginea de separare și $\|\hat{w}\|$, indicați soluția \hat{w} și deduceți valoarea lui \hat{w}_0 .

39.

(Neseparabilitate liniară:
 două exemple de mapare a trăsăturilor;
 rezolvarea — în manieră directă — a problemei SVM
 în spațiul / spațiile de trăsături)
CMU, 2011 fall, Eric Xing, HW4, pr. 3

Se dau 6 puncte din \mathbb{R} : $x_1 = -1$, $x_2 = 0$, $x_3 = 1$ au etichete negative, iar $x_4 = -2$, $x_5 = 2$, $x_6 = 3$ au etichete pozitive.

- a. Desenați cele 6 puncte pe axa reală, folosind simbolul \circ pentru a reprezenta etichetele negative și simbolul \bullet pentru etichetele pozitive.
- b. Aceste 6 puncte nu sunt liniar-separabile. Definiți $f : \mathbb{R} \rightarrow \mathbb{R}$, o funcție de transformare a trăsăturilor astfel încât punctele $f(x_1), f(x_2), \dots, f(x_6)$ să fie liniar-separabile. Desenați pe axa reală cele 6 instanțe astfel obținute. Apoi indicați poziția separatorului optimal calculat de mașina cu vectori-suport liniară cu margine “hard”, marcând instanțele care sunt vectori-suport.
- c. Separatorul de la punctul precedent are forma analitică $w_0 + w_1 f(x) = 0$. Indicați valorile lui w_0 și w_1 .
- d. Să presupunem acum că mapăm cele 6 puncte în spațiul de trăsături $(x, f(x))$, unde $f(x)$ este funcția de transformare a trăsăturilor de la punctul b . Cu alte cuvinte, acum vom avea 6 puncte în plan, $(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_6, f(x_6))$. Desenați aceste 6 puncte în planul bidimensional, împreună cu separatorul optimal definit de către mașina cu vectori-suport liniară cu margine “hard”. Indicați apoi vectorii-suport.
- e. Separatorul de la punctul precedent are forma analitică $w_0 + w_1 x + w_2 f(x) = 0$. Calculați valorile parametrilor w_0, w_1 și w_2 .
- f. Funcția de mapare a trăsăturilor $x \rightarrow (x, f(x))$ de la punctele d și e este asociată cu o funcție-nucleu $K(x, x')$, unde x și x' sunt puncte din spațiul original (de trăsături) unidimensional. Scrieți expresia acestei funcții-nucleu.

³⁵³Veți ține cont că

1. într-un spațiu de tip \mathbb{R}^d , egalitatea $w \cdot x = 0$ implică faptul că vectorii w și x sunt ortogonali / perpendiculari;
 2. ecuația unui hiperplan din \mathbb{R}^d este de forma $w \cdot x + w_0 = 0$, ceea ce implică faptul că vectorul w este perpendicular pe acest hiperplan (întrucât este perpendicular pe orice vector din hiperplan, văzând un astfel de vector ca diferență de doi vectori de poziție $x_1 - x_2$, unde punctele x_1 și x_2 aparțin hiperplanului).

Așadar, vectorul \hat{w} din problema noastră este perpendicular pe hiperplanul de separare optimă.

40.

(Găsirea separatorului liniar optimal,
după maparea într-un „spațiu de trăsături“,
conform unei funcții-nucleu date)

*CMU, 2008 spring, Eric Xing, midterm exam, pr. 5
MIT, 2006 fall, Tommi Jaakkola, midterm exam, pr. 4*

Presupunem că avem instanțele pozitive $x_1 = (0.4, 0.2)$, $x_2 = (0.8, 0.4)$, $x_3 = (0.2, 0.4)$, $x_4 = (0.4, 0.8)$ și instanțele negative $x_5 = (0.4, 0.4)$ și $x_6 = (0.8, 0.8)$. Se poate constata ușor că aceste date nu sunt separabile liniar.

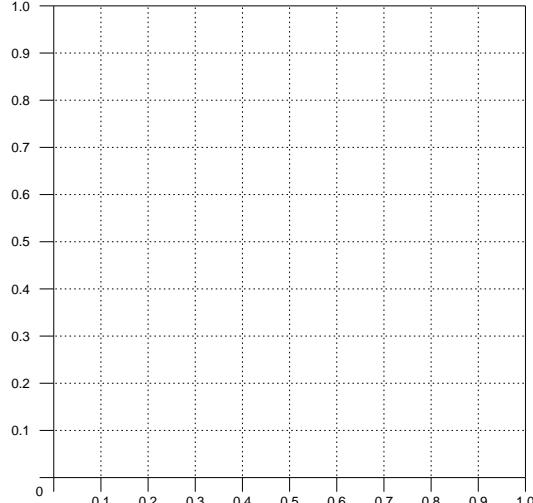
Vom folosi o mapare Φ astfel încât

$$\Phi(x) \cdot \Phi(x') \stackrel{not.}{=} K(x, x') = \frac{x \cdot x'}{\|x\| \|x'\|}.$$

a. Care este vectorul $\Phi(x)$ corespunzător acestei funcții-nucleu?

b. Pe gridul alăturat, reprezentați atât instanțele de antrenament x_i cât și „imaginile“ $\Phi(x_i)$, pentru $i = 1, \dots, 6$. Fiecare „imagină“ $\Phi(x_i)$ va avea asociată aceeași etichetă ca și x_i . Veți folosi *convenția* noastră de notare: simbolul \bullet desemnează instanțe pozitive, iar simbolul \circ instanțe negative.

Sugestie: În loc să faceți efectiv calcularea coordonatelor pentru „imaginile“ $\Phi(x_i)$, ar fi mai util / interesant să vedeti cât este $\|\Phi(x_i)\|$ pentru fiecare x_i , iar apoi să trageți o *concluzie generală* cu privire la poziția relativă a lui $\phi(x_i)$ în raport cu x_i , pentru orice i .



c. Veți observa că instanțele $\Phi(x_i)$ sunt liniar-separabile, deci există o dreaptă determinată de parametrii w_1, w_2 și w_0 , care le separă în mod optimal. Cât este raportul w_1/w_2 ?

Indicație: Nu este necesar să calculați efectiv w_1 și w_2 . Este suficient să exploațați simetriile.

d. Pe desenul de la punctul b, încercuiți vectorii-suport $\Phi(x_i)$.

e. Indicați în spațiul de origine zonele de decizie [și separatorul decizional] care corespund separatorului optimal găsit la punctul c.

41.

(Învățarea conceputului \neg XOR,
folosind forma duală a problemei SVM
și o funcție-nucleu polinomială de ordin 2)
*prelucrare de Liviu Ciortuz, după
CMU, 2006 fall, E. Xing, T. Mitchell, midterm exam, pr. 5*

Fie o problemă de învățare supervizată în care exemplele de antrenament se află în spațiul euclidian bidimensional. Exemplele pozitive sunt $x_1 = (1, 1)$ și $x_3 = (-1, -1)$, iar exemplele negative sunt $x_2 = (-1, 1)$ și $x_4 = (1, -1)$.

Considerăm transformarea $\Phi(x)$ corespunzătoare funcției-nucleu $K(x, x') = (x \cdot x' + 1)^2$, unde x și x' sunt din \mathbb{R}^2 .

Funcția de predicție pe care vrem să o obținem este de forma $y(x) = w \cdot \Phi(x) + w_0$, unde $\Phi(x), w \in \mathbb{R}^n$ (cu n ales convenabil), $w_0 \in \mathbb{R}$, iar \cdot reprezintă produsul scalar.

Indicați coeficienții w, w_0 corespunzători suprafeței de separare cu margine maximă pentru punctele de antrenament date. Cum va fi clasificată o instanță oarecare de test $x \in \mathbb{R}^2$?

Sugestie de lucru:

- Mai întâi reprezentați datele. Determinați $\Phi(x), n$, și $\Phi(x_1), \Phi(x_2), \Phi(x_3), \Phi(x_4)$.
- Calculați apoi lagrangeanul dual

$$L_D(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \Phi(x_i) \cdot \Phi(x_j)$$

c. Determinați soluția problemei duale $\bar{\alpha} = \operatorname{argmax}_{\alpha} L_D(\alpha)$ cu restricțiile $\alpha_i \geq 0$ pentru $i = 1, \dots, 4$ și $\sum_i \alpha_i y_i = 0$ și, în final, găsiți soluția problemei primale $\bar{w} = \sum_i \bar{\alpha}_i y_i \Phi(x_i)$ și \bar{w}_0 , știind că \bar{w}_0 poate fi obținut dintr-una din restricțiile $y_i(\bar{w} \cdot \Phi(x_i) + \bar{w}_0) = 1$.

42.

(SVM cu diferite funcții-nucleu:
efectul unei translatări a datelor de antrenament
asupra poziției separatorului optimal)
CMU, 2009 fall, Carlos Guestrin, HW3, pr. 2.10

Trei mașini cu vectori-suport au fost antrenate pe un set de date din \mathbb{R}^2 folosind

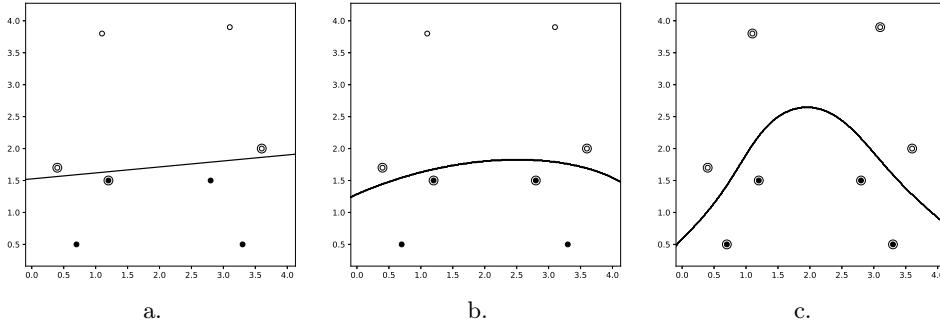
- o funcție-nucleu liniară $K(x, y) = x \cdot y$ (a se vedea figura a);
- o funcție-nucleu polinomială de gradul al doilea $K(x, y) = (x \cdot y + 1)^2$ (a se vedea figura b);
- o funcție-nucleu cu baza radială $K(x, y) = e^{-\frac{\|x - y\|^2}{2\sigma^2}}$ (a se vedea figura c).

Presupunem că translatăm datele adăugând o constantă mare (de exemplu 10) la coordonata de pe axa verticală, pentru fiecare din datele de antrenament, adică (x, y) devine $(x, y + 10)$.

Dacă reantrenăm SVM-urile de mai sus pe noile date de antrenament, se schimbă și poziția separatorului optimal în raport cu datele? Tratați pe rând cazurile a, b și c.

Explicați pe scurt de ce se schimbă (sau de ce nu se schimbă) poziția separatorului optimal în raport cu datele, pentru fiecare din cazurile a, b și c. Trasați pe desen poziția noului separator optimal, acolo unde este cazul.³⁵⁴

³⁵⁴ Sugestie: Puteți consulta articolul *On Invariance of Support Vector Machines*, de Shigeo Abe, <http://www2.kobe-u.ac.jp/~abe/pdf/ideal2003.pdf> accesat la data 12.06.2018. LC: Mulțumesc lui Sebastian Ciobanu pentru această sugestie.



43. (SVM cu nucleu RBF — o caracteristică surprinzătoare:
instante foarte distanțate față de separatorul optimal
pot fi vectori-suport)

*MIT, 2002 fall, Tommi Jaakkola, midterm exam, pr. 3.1-3
CMU, 2011 spring, Tom Mitchell, HW6, pr. 1.2*

[Remember:] Folosind o funcție-nucleu oarecare, SVM caută într-un anumit „spațiu de trăsături” Q un hiperplan care să maximizeze distanța dintre cele două clase. Clasificarea unei instanțe oarecare de test x se face determinând semnul expresiei

$$\bar{w} \cdot \Phi(x) + \bar{w}_0 = \left(\sum_{i \in SV} y_i \bar{\alpha}_i \Phi(x_i) \right) \cdot \Phi(x) + \bar{w}_0 = \sum_{i \in SV} y_i \bar{\alpha}_i K(x_i, x) + \bar{w}_0 \stackrel{not.}{=} f(x; \bar{\alpha}, \bar{w}_0),$$

unde

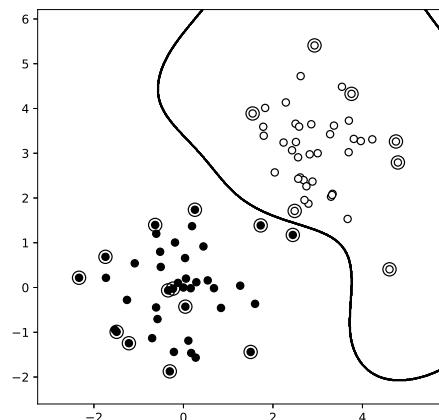
\bar{w} și \bar{w}_0 sunt parametrii pentru hiperplanul de clasificare în spațiul Q ,

SV este setul de vectori-suport,

$\bar{\alpha}_i$ este coeficientul (multiplicatorul Lagrange) corespunzător vectorului-suport i .

În acest exercițiu vom folosi drept nucleu funcția cu baza radială $K(x_i, x_j) = e^{-\frac{1}{2}||x_i - x_j||^2}$.
Vom presupune că instanțele de antrenament sunt liniar separabile în spațiul Q .

Observație: Figura alăturată prezintă grafic rezultatul obținut de SVM folosind nucleu RBF pe un set de date din planul euclidian. Vectorii-suport sunt încercuiți (îngroșați). Ceea ce este curios la această clasificare este că unii vectori-suport sunt destul de departe de separatorul optimal. Să, totuși, ei sunt vectori-suport! Întrebările de mai jos vor încerca să contribuie la elucidarea acestei chestiuni.



- a. Arătați că ori de câte ori alegem un punct de test $x_{\text{far}} \in \mathbb{R}^d$ foarte distanțat de orice instanță de antrenament x_i , rezultă că $f(x_{\text{far}}; \bar{\alpha}, \bar{w}_0) \approx \bar{w}_0$.

În continuare, vom presupune, pentru simplitate, că $\bar{w}_0 = 0$.

- b. Din ipoteză, întrucât setul de date de antrenament este liniar separabil, rezultă că orice instanță de antrenament x_i satisfac inegalitatea $y_i \bar{w} \cdot \Phi(x_i) \geq 1$, unde Φ este funcția de mapare corespunzătoare nucleului RBF specificat mai sus. Satisfac și x_{far} această inegalitate?

Observație: Pentru RBF, $\Phi(x)$ este un vector infinit, de aceea de drept întrebarea aceasta nu are sens aici. De fapt,³⁵⁵ puteți răspunde totuși la această întrebare raportându-vă la $\Phi(x)$ ca și când ar fi un vector finit.

- c. Dacă am include punctul x_{far} în setul de date de antrenament, ar deveni el oare un vector-suport?

44.

(Comparație între SVM și alți clasificatori)
CMU, 2010 fall, Aarti Singh, midterm, pr. 1.2.1

X_1	X_2	Y
0	0	+
1	0	-
0	1	-
1	1	+

Fie setul de date din \mathbb{R}^2 din tabelul alăturat. Identificați dintre clasificatorii următori pe aceia care obțin eroare de antrenare 0 pe acest set de date.

- SVM cu nucleu polinomial de ordin 2, adică $(c + x \cdot x')^2$, în care meta-parametrul c este la libera alegere;
- regresia logistică
- arbori ID3 de adâncime 2, adică având două nivele de test, dintre care unul este nivelul-rădăcină;
- 3-NN.

45.

(O legătură (simplă) între rețele neuronale și SVM cu funcție-nucleu polinomială)
prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Aarti Singh, HW5, pr. 4.2

Se consideră datele de antrenament $\bar{x}_1, \dots, \bar{x}_m \in \mathbb{R}^d$, etichetate respectiv cu $y_1, \dots, y_m \in \{-1, +1\}$. Se consideră de asemenea o funcție-nucleu polinomială de gradul $p \in \mathbb{N}$

$$K(\bar{x}, \bar{x}') = (\gamma \bar{x} \cdot \bar{x}' + r)^p = \Phi(\bar{x}) \cdot \Phi(\bar{x}')$$

unde γ și r sunt numere reale.

Stim că o SVM care folosește această funcție-nucleu și care a fost antrenată pe datele de mai sus va clasifica o instanță de test $\bar{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$ conform expresiei:

$$\text{sign}(w \cdot \Phi(\bar{x}) + w_0) = \text{sign}\left(\sum_{i=1}^m y_i \alpha_i \Phi(\bar{x}) \cdot \Phi(\bar{x}_i) + w_0\right) = \text{sign}\left(\sum_{i=1}^m y_i \alpha_i K(\bar{x}, \bar{x}_i) + w_0\right),$$

³⁵⁵LC: Puteți face abstracție de procesul de trecere la limită care ar trebui avut în vedere la operația dintre w și $\phi(x)$.

unde $w \in \mathbb{R}^n$ (n fiind dimensiunea spațiului de trăsături asociat cu „maparea“ Φ) și $w_0 \in \mathbb{R}$ desemnează ponderile învățate de SVM, iar α_i sunt valorile multiplicatorilor Lagrange (corespunzători instanțelor de antrenament) care constituie soluțiile formei duale a problemei SVM.

Se consideră o rețea neuronală artificială de tip feed-forward cu un singur nivel ascuns, descrisă astfel:

- intrările în rețea sunt x_1, \dots, x_d (a se vedea \bar{x} de mai sus);
- rețeaua are m unități ascunse, și anume câte una pentru fiecare instanță de antrenament $\bar{x}_i = (x_{i1}, \dots, x_{id})$, cu $i \in \{1, \dots, m\}$;
- ponderile de pe conexiunile dintre intrări și unitățile ascunse sunt fixate astfel:
 - pentru unitatea ascunsă i , ponderile de la intrările x_1, \dots, x_d sunt respectiv x_{i1}, \dots, x_{id} (a se vedea \bar{x}_i de mai sus); aceasta are loc pentru fiecare $i \in \{1, \dots, m\}$;
 - nu se folosește termen liber / constant ($x_0 = 1$) pentru aceste unități ascunse;
- ieșirea unității ascunse i (cu $i \in \{1, \dots, m\}$) este definită într-o manieră nestandard, și anume:

$$o_i = (\gamma \text{ net}_i + r)^p, \text{ unde } \text{net}_i \text{ este } \sum_{j=1}^d x_j x_{ij}$$

- rețeaua are o singură unitate pe stratul de ieșire, pe care o vom nota cu $m+1$; ea are funcția de activare de tip *sign*;
 - ponderile hidden-output sunt de forma $y_i \alpha_i$ pentru fiecare $i \in \{1, \dots, m\}$;
 - pentru unitatea de ieșire, ponderea intrării libere ($x_0 = 1$) este w_0 .
- a. Să se deseneze această rețea neuronală sub forma unui graf în care intrările și unitățile neuronale sunt noduri, iar ponderile sunt marcate corespunzător pe arce.
- b. Arătați că acești doi clasificatori — rețeaua neuronală și SVM-ul de mai sus — sunt echivalenți. Adică, pentru fiecare instanță de test $\bar{x} \in \mathbb{R}^d$ rezultatele produse de către cei doi clasificatori (văzuți simplu ca funcții de clasificare) sunt identice.
- Indicație:* Calculați funcția reprezentată de outputul rețelei neuronale.
- c. O rețea neuronală similară cu cea de mai sus poate chiar să învețe parametrii γ și r . Indicați ce modificări trebuie făcute în rețea și, corespunzător, în algoritm (general) de retro-propagare, pentru a învăța acești doi parametri.

5.2.2 SVM cu margine “soft”

46.

(C-SVM: exemplificarea noțiunilor de bază)
CMU, 2017 fall, Nina Balcan, midterm, pr. 1.3

Atunci când *nu* folosim termen liber (engl., bias), problema de optimizare C-SVM are

forma primală:

$$\min_{w, \xi} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \right)$$

a. î. $y_i w \cdot x_i \geq 1 - \xi_i$ și $\xi_i \geq 0$ ($i = 1, \dots, m$)

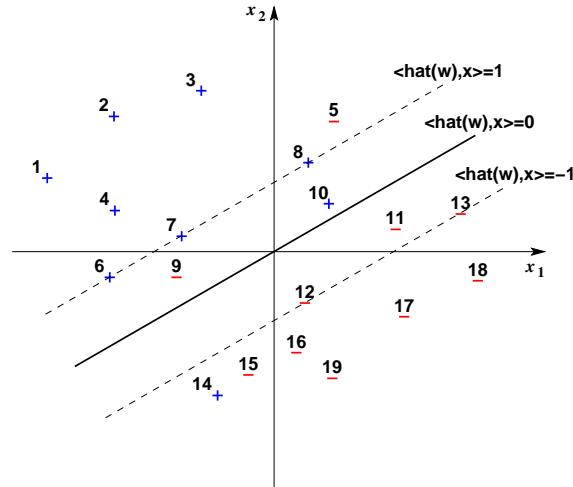
forma duală:

$$\max_{\alpha \in \mathbb{R}^m} \left(\frac{1}{2} \sum_{i=1}^m \alpha_i + \sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j x_i \cdot x_j \right)$$

a. î. $\sum_{i=1}^m y_i \alpha_i = 0$ și $0 \leq \alpha_i \leq C$ ($i = 1, \dots, m$).

Notăm cu $\hat{w}, \hat{\xi}$ soluția [optimă a] problemei primale și cu $\hat{\alpha}, \hat{\beta}$ soluția [optimă a] problemei duale. (Vă readucem aminte că $\beta_i \geq 0$ este multiplicatorul Lagrange corespunzător inegalității $\xi_i \geq 0$.)

În figura alăturată, vi se dă un set de date de antrenament din \mathbb{R}^2 , precum și separatorul optimal învățat de către clasificatorul C-SVM pe acest set de date. În această figură, notația \hat{w} corespunde lui \hat{w} , iar $\langle \hat{w}, x \rangle$ desemnează produsul scalar $\hat{w} \cdot x$. Numărul marcat în dreptul fiecărei instanțe reprezintă indexul (sau, ID-ul) respectivului punct.



- Indicați toate instanțele pentru care $\xi_i \neq 0$.
- Indicați toți vectorii-suport. Vă readucem aminte că instanța x_i este vector-suport dacă $\hat{\alpha}_i \neq 0$.
- Considerăm următoarea propoziție: $\hat{\alpha}_i = C$ pentru punctele cu indicii 10 și 14. Adevarat sau Fals? Justificați.

47.

(SVM vs. C-SVM: comparație între soluțiile celor două probleme pe seturi de date liniar separabile)

CMU, 2017 fall, Nina Balcan, HW4, pr. 4.Q11

Fie $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ o mulțime de instanțe din \mathbb{R}^d , cu etichetele luând valori în mulțimea $\{1, -1\}$. Presupunem că exemplele din S sunt liniar separabile și că rulăm C-SVM cu $C > 0$ pe setul de date S .

- Este oare separatorul decizional obținut de către clasificatorul C-SVM identic cu separatorul de margine maximă (adică, separatorul obținut de către SVM)? Justificați.
- Este oare adevărată afirmația că separatorul decizional obținut de către clasificatorul C-SVM separă întotdeauna în mod corect cele două clase (adică, obține eroare la antrenare 0)? Justificați.

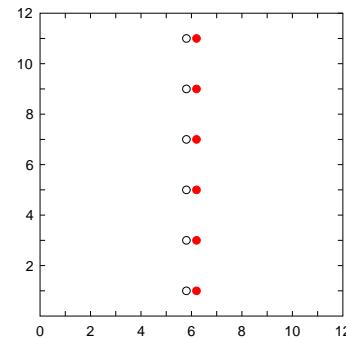
48. (SVM și C-SVM: efectul adăugării unui atribut irelevant
 (i.e., care nu mărește marginea de separare)
 asupra vectorului de ponderi w)

*CMU, 2017 fall, Nina Balcan, midterm, pr. 1.4
 CMU, 2007 spring, Carlos Guestrin, midterm exam, pr. 6.3*

a. Cazul SVM:³⁵⁶



Fie $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ un set de date din \mathbb{R}^d care sunt liniar separabile.³⁵⁷ Presupunem că adăugăm la fiecare instanță x_i un atribut $\tilde{x}_i \in \mathbb{R}$,³⁵⁸ care nu conduce la mărirea marginii [geometrice, de separare a] lui S . În acest context, spunem că atributul \tilde{x} este *irrelevant*. Va ignora oare SVM în mod automat acest atribut (adică, în vectorul soluție \hat{w} , componenta corespunzătoare noului atribut va fi 0)? Justificați răspunsul în mod riguros.



Observație: Pentru demonstrație, veți putea folosi următoarea proprietate: dacă multimea S este liniar separabilă, atunci problema de optimizare SVM în formă primală are soluție unică, întrucât funcția obiectiv a acestei probleme este strict convexă.

b. Cazul C-SVM:³⁵⁹

Presupunem că rulăm o C-SVM pe [un dataset S cu] atributele X_1, \dots, X_d , iar apoi adăugăm [la instanțele din S] un atribut X_{d+1} care este *irrelevant*, în sensul că [presupunând că păstrăm aceeași valoare pentru parametrul C] el nu poate crește marginea de separare. Va ignora oare C-SVM în mod automat acest atribut? Justificați răspunsul în mod riguros.

49. (C-SVM cu termeni de „destindere” în norma L_2)
*University of Utah, 2008 fall, Hal Daumé III, HW5, pr. 3
 CMU, 2010 fall, Aarti Singh, midterm exam, pr. 5.1*

La curs am arătat că atunci când datele noastre de antrenament nu sunt liniar separabile, putem modifica forma problemei SVM introducând în fiecare restricție câte o variabilă de

³⁵⁶Se subînțelege, SVM liniară, cu margine “hard”.

³⁵⁷LC: În enunțul original se precizează — în paranteze! — că separarea se [poate] face cu un separator care trece prin originea sistemului de coordonate. Restricția aceasta nu este neapărat necesară, deoarece [veți constata că] demonstrația poate fi făcută în cazul general al separabilității liniare.

³⁵⁸Se poate observa imediat că adăugarea acestui atribut nu afectează proprietatea de separabilitate liniară a datelor de antrenament S .

³⁵⁹Adică, SVM liniară cu margine “soft”.

„destindere“ (engl., slack variable) în raport cu marginea.³⁶⁰ Mai precis, formularea pe care am dat-o este cunoscută sub denumirea *C-SVM de normă L₁*.

Acum vom considera o variantă cunoscută sub numele de *C-SVM de normă L₂*. Se pleacă de la următoarea problemă de optimizare:

$$\min_{w, w_0, \xi} \left(\frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^m \xi_i^2 \right)$$

a.î. $y_i(w \cdot x_i + w_0) \geq 1 - \xi_i, i = 1, \dots, m.$

Remarcați faptul că variabilele de „destindere“ ξ_i apar la puterea a două în funcția obiectiv, cu scopul de a limita posibilitatea ca valorile lor să fie excesiv de mari.³⁶¹

- a. În forma problemei de mai sus, nu apare condiția $\xi_i \geq 0$. Arătați că aceste restricții de nenegativitate pot fi într-adevăr eliminate. Așadar, vă cerem să demonstrați că valoarea optimă a funcției obiectiv va fi aceeași, indiferent dacă aceste restricții ($\xi_i \geq 0$) sunt sau nu incluse în formularea problemei.
- b. Care este *funcția lagrangeană generalizată* (notație: $L_P(w, w_0, \xi, \alpha)$) asociată problemei de optimizare C-SVM de normă L_2 ?
- c. Minimizați funcția L_P în raport cu w , w_0 și ξ , calculând mai întâi derivatele parțiale $\frac{\partial L_P}{\partial w}$, $\frac{\partial L_P}{\partial w_0}$ și $\frac{\partial L_P}{\partial \xi}$, și egalându-le apoi cu 0. Am folosit notația $\xi = (\xi_1, \xi_2, \dots, \xi_m)$.
- d. Care este *forma duală* a problemei de optimizare C-SVM de normă L_2 ? Prin ce diferă această formă duală de forma duală a problemei C-SVM (vedeți pr. 13)?

50. (C-SVM folosind funcție de cost *hinge*: exemplu de aplicare (adică, rezolvarea problemei de optimizare corespunzătoare); comparație cu regresia logistică, relativ la efectul outlier-elor asupra poziției separatorului optimal)

CMU, 2012 fall, E. Xing, A. Singh, HW2, pr. 4.3

La problema 21 am arătat că putem reformula problema de optimizare C-SVM liniară (adică, nekernel-izată) astfel încât să minimizăm suma costurilor *hinge* pe setul de date de antrenament, cu termen de regularizare $\|w\|^2$:

$$\min_{w, b} \left(\|w\|^2 + C \sum_i Loss_{SVM}(f(x_i), y_i) \right),$$

unde (x_i, y_i) sunt instanțe etichetate, cu $y_i \in \{-1, +1\}$, iar costurile *hinge* sunt definite în modul următor:

$$Loss_{SVM}(f(x_i), y_i) \stackrel{\text{def.}}{=} \max(1 - (w \cdot x_i + w_0)y_i, 0).$$

Vom considera setul de date $(x_1, y_1), \dots, (x_n, y_n)$ cu $n = 2000000$, unde pentru $i = 1, \dots, n/2$ avem $x_i = 0$ și $y_i = -1$, iar pentru $i = n/2 + 1, \dots, 2n$ avem $x_i = 2$ și $y_i = +1$.

³⁶⁰Am reluat această chestiune la problema 13.

³⁶¹Pentru conveniență calculelor, în locul lui C — factorul care precede suma $\sum_i \xi_i$ în problema 13 —, aici lucrăm cu $\frac{C}{2}$ ca factor pentru $\sum_i \xi_i^2$. A se vedea calculele de la punctele c și d .

Cu alte cuvinte, se dă un milion de copii ale instanței 0 (în spațiul unidimensional, desigur), toate etichetate cu -1 , și un milion de copii ale instanței 2, toate etichetate cu $+1$.

a. Găsiți valorile lui w și w_0 care minimizează funcția obiectiv a problemei de optimizare C-SVM de mai sus pe aceste date, atunci când se lucrează cu $C = 1$.

Care este granița de decizie și cât este „marginea“ corespunzătoare? Cum va difera răspunsul dacă vom considera valori din ce în ce mai mici pentru C , tînzând la 0?

b. Acum vom presupune că pe lângă cele $n = 2000000$ de instanțe de mai sus ni se mai dă încă o instanță:

$$x_{n+1} = 100, \quad y_{n+1} = -1.$$

Care sunt noile valori optimale pentru w și w_0 (folosind din nou $C = 1$)?

c. Considerând că la punctul b în locul algoritmului C-SVM folosim regresia logistică, precizați (raționând în mod intuitiv) care vor fi noile rezultate.

51.

(Algoritmul SMO pentru C-SVM:
o restricție la alegerea variabilelor libere, suficientă ca
algoritmul să nu poată îmbunătăți soluția (α) fixată inițial)
MIT, 2008 fall, Tommi Jaakkola, midterm exam, pr. 1.2

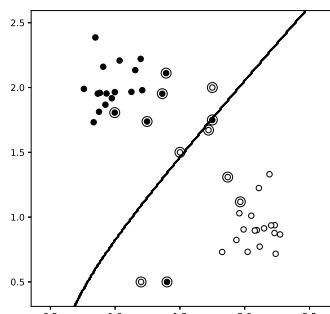
Considerăm algoritmul SMO pentru antrenarea unui C-SVM (adică, SVM cu variabile de „destindere“). Inițial, toate variabilele duale α_i ($i = 1, \dots, m$) sunt setate la valoarea 0. Apoi, la fiecare iterație a algoritmului alegem două variabile y_i și y_j impunând spre deosebire de [sau: pe lângă criteriu de selecție din] forma clasică a algoritmului SMO un criteriu simplu: cele două variabile trebuie să satisfacă restricția $y_i = y_j$. După selectare, optimizarea celor două variabile se face ca în algoritmul classic SMO.

Ce soluție va calcula această versiune a algoritmului SMO? Justificați răspunsul.

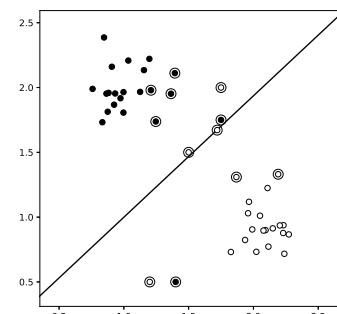
52.

(C-SVM: efectul alegерii
valorii parametrului C și / sau a funcției-nucleu)
CMU, 2012 spring, Ziv Bar-Joseph, HW3, pr. 3.1

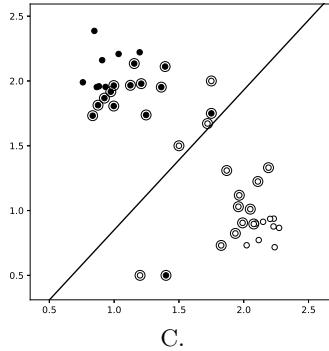
În figura de mai jos sunt ilustrate suprafețele de decizie pentru patru mașini cu vectori-suport cu margine „soft“ (adică, C-SVM-uri), care folosesc diferite funcții-nucleu și diferite valori pentru parametrul C pentru „destindere“.



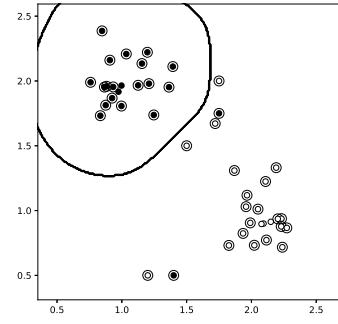
A.



B.



C.



D.

Pentru fiecare dintre aceste exemple, specificați ce setare credeți că a fost folosită.

- $C = 1$ și nicio funcție-nucleu;
- $C = 0.1$ și nicio funcție-nucleu;
- $C = 0.1$ și funcția-nucleu $K(x_i, x_j) = e^{-10\|x_i - x_j\|^2}$;
- $C = 0.1$ și funcția-nucleu $K(x_i, x_j) = x_i \cdot x_j + (x_i \cdot x_j)^2$.

53.

(C-SVM cu funcție-nucleu pătratică: determinarea graniței de decizie în funcție de valoarea parametrului pentru „destindere“ C)

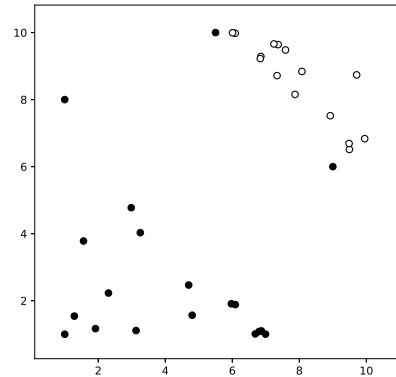
CMU, 2007 spring, Carlos Guestrin, midterm exam, pr. 2

Obiectivul acestei probleme este să clasificăm corect date de test, pornind de la un anumit set de date de antrenament.

Vom presupune că antrenăm o C-SVM, folosind ca funcție-nucleu o funcție polinomială de gradul al doilea. Vi se dă setul de date de antrenament din figura alăturată.

Avertisment: Se consideră că datele de antrenament provin de la niște senzori care pot fi afectați de „zgomote“ / perturbații, deci ar trebui să evitați să vă încredeți prea mult în vreun punct anume.

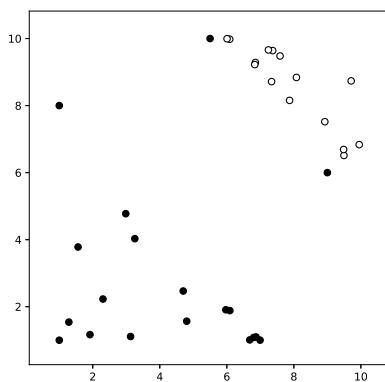
Penalizarea de „destindere“ (engl., slack penalty) C va determina situația hiperplanului de separare optimală.



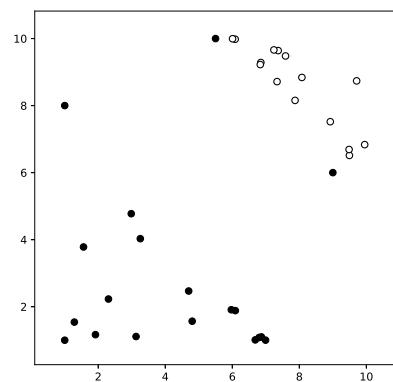
Vi se cere să răspundeți la întrebările de mai jos în manieră *calitativă*. Pentru fiecare dintre aceste puncte, formulați răspunsul sub formă unei singure fraze. Desenați soluția în partea corespunzătoare din figurile de la sfârșitul acestui enunț.

- Unde va fi situată granița de separare atunci când parametrul C ia valori mari (adică $C \rightarrow \infty$)? Desenați răspunsul în figura a.
- Pentru $C \approx 0$, indicați în figura b unde anume credeți că va fi situată granița de decizie. Justificați răspunsul.

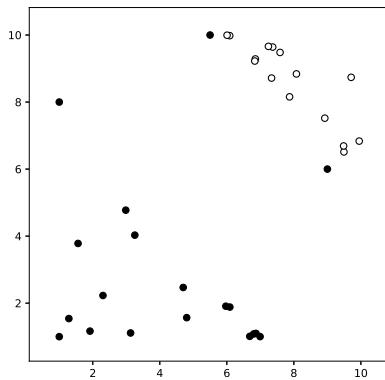
- c. Care dintre cele două cazuri de mai sus credeți că este mai adekvat pentru task-ul de generalizare / predicție? De ce?
- d. Desenați în figura *d* un punct care nu va schimba granița de decizie care a fost învățată pentru valori foarte mari ale lui C . Justificați răspunsul.
- e. Desenați în figura *e* un punct care va schimba în mod considerabil granița de decizie care a fost învățată pentru valori foarte mari ale lui C . Justificați răspunsul.



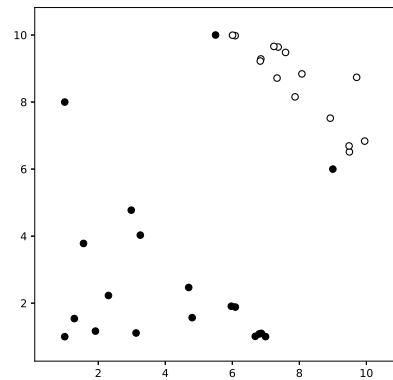
a.



b.



c.



d.

54. (C-SVM cu funcții-nucleu: *complexitatea* computațională la antrenare și respectiv la testare)
CMU, 2014 fall, E. Xing, B. Poczos, HW2, pr. 3.6

Presupunem că avem m exemple de antrenament $(x_i, y_i)_{i=1}^m$ din $\mathbb{R}^d \times \{-1, +1\}$ și că dorim să antrenăm un C-SVM (adică SVM cu margine “soft”) care folosește funcție-nucleu. În multe cazuri practice, numărul m este foarte mare (de ordinul sutelor de milioane).

- a. Cât este *complexitatea de spațiu* dacă implementăm C-SVM cu funcție nucleu în *măieră naivă*? Veți da răspunsul presupunând că $m \gg d$.

- b. Cât este costul calculării *funcției de decizie* pentru o instanță oarecare x , dacă presupunem că există $\mathcal{O}(m)$ vectori-support, iar complexitatea de timp pentru calculul valorii funcției-nucleu $k(x, x')$ este $\mathcal{O}(d)$?
- c. Există mai multe modalități de a aproxima *funcția de decizie*. Citiți secțiunea de introducere din articolul *Fastfood – approximating kernel expansions in loglinear time*, de Quoc Le, Tamás Sarlós și Alex Smola³⁶² și scrieți cât este complexitatea computațională a funcției de decizie în respectiva abordare.
55. (Clasificatori liniari — [C-]SVM, Perceptronul etc. — și versiunile lor kernel-izate: avantaje și dezavantaje)
CMU, 2017 fall, Nina Balcan, HW3, pr. 3.3.2/Q17
 Care credeți că sunt avantajele și dezavantajele (engl., *pros* and *cons*) folosirii modelelor liniare precum SVM, Perceptronul etc., precum și versiunile lor kernel-izate?
56. (O comparație între algoritmii 1-NN și C-SVM: efectul atributelor irelevante pentru clasificare)
CMU, 2007 spring, Carlos Guestrin, midterm exam, pr. 6.1-2
Convenție: În tot acest exercițiu, prin SVM se va înțelege o mașină cu vectori-suport cu margine “soft” (adică, C-SVM), fără funcție-nucleu (ceea ce înseamnă că lucrăm cu separator liniar).
- a. Alcătuiți un set de date din \mathbb{R}^2 astfel încât pe acest set algoritmul 1-NN să producă eroare la cross-validation de tip “leave one out” (CVLOO) mai mică decât SVM.
- b. Alcătuiți un alt set de date din \mathbb{R}^2 astfel încât pe noul dataset eroarea de tip CVLOO produsă de algoritmul 1-NN să fie mai mare decât cea produsă de SVM.
- c. Acum veți genera două dataset-uri care să pună în evidență caracterul robust al algoritmului SVM în raport cu *atributele irelevante*.³⁶³ Veți crea un set de date din \mathbb{R}^2 (câte unul pentru fiecare dintre cele două probleme date mai jos) cu atributul X_1 și X_2 — dintre care X_2 va fi atributul irrelevant —, astfel încât:
- dacă se folosește doar atributul X_1 , eroarea la CVLOO produsă de algoritmul 1-NN este mai mică decât cea produsă de SVM,
 - dacă se folosesc atrbutele X_1 și X_2 , eroarea CVLOO produsă de SVM nu se schimbă (în raport cu cazul de mai sus), însă eroarea CVLOO produsă de 1-NN crește în mod semnificativ.

57. (C-SVM: Adevărat sau Fals?)
CMU, 2010 fall, Aarti Singh, HW3, pr. 3.2

Presupunem că se lucrează cu o mașină cu vectori-suport cu margine “soft” (C-SVM), pe un anumit set de exemple, fără a folosi vreo funcție de „mapare“ a trăsăturilor. Pe măsură ce valoarea parametrului de „destindere“ C crește (pornind de la o anumită valoare de start),

³⁶²Proceedings of the 30th International Conference on Machine Learning (ICML), 2013, pp 244–252.

³⁶³Adică, acele atrbute care, atunci când sunt eliminate / adăugate, n-ar trebui să schimbe rezultatul clasificării.

- a. mai multe instanțe de antrenament vor fi clasificate eronat;
 b. marginea — adică $\frac{1}{\|w\|}$, distanța de la hiperplanul de separare optimală la instanțele (vectorii-suport) x_i pentru care $(w \cdot x_i + w_0)y_i = 1$, unde y_i este eticheta asociată instanței x_i — nu va crește (adică fie descrește fie rămâne aceeași).

5.2.3 Alte probleme de optimizare de tip SVM

58. (Funcția-nucleu RBF — o proprietate remarcabilă:
 pentru orice set de instanțe de antrenament distincte și
 pentru orice etichetare a acestora și, de asemenea,
 pentru orice valoare a parametrului funcției-nucleu RBF,
 problema de optimizare de tip SVM care impune ca
 toate instanțele de antrenament să fie corect clasificate
 și la distanța $1/\|w\|$ de separatorul optimal are soluție)
 ■ MIT, 2009 fall, Tommi Jaakkola, HW2, pr. 1

Expresia de definiție a funcției-nucleu cu bază radială (RBF) poate fi scrisă sub forma următoare:

$$K(x, x') = \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right), \quad (139)$$

unde x și x' sunt elemente din \mathbb{R}^d , iar σ este parametrul care arată cât de repede se micșorează valoarea funcției-nucleu pe măsură ce punctele x și x' se situează la distanțe din ce în ce mai mari unul față de celălalt.

În acest exercițiu ne propunem să arătăm că funcția-nucleu RBF are câteva proprietăți remarcabile. În primul rând, ea poate separa în mod perfect orice multime finită de instanțe de antrenament *distincte*,³⁶⁴ iar acest rezultat este valabil pentru orice valoare pozitivă finită a parametrului σ .³⁶⁵ (Trebuie să menționăm totuși că valoarea lui σ afectează calitatea generalizării / predicției pe instanțe de test.)

- a. Vom începe să demonstrăm că problema de optimizare

$$\min_w \frac{1}{2}\|w\|^2 \text{ cu restricțiile } y_i w \cdot \phi(x_i) = 1 \text{ pentru } i = 1, \dots, n \quad (140)$$

admete soluție, indiferent de ce valori (± 1) ar avea etichetele y_i asignate instanțelor x_i , cu $i = 1, \dots, n$. Am notat cu $\phi(x_i)$ vectorul de trăsături (engl., feature vector) determinat de funcția de „mapare” care corespunde funcției K (vedeți definiția dată mai sus).

Comentariu: Formularea problemei de optimizare (140) diferă de formularea problemei [SVM], din două motive. Mai întâi, se observă că încercăm să găsim o soluție a acestei probleme [SVM de un tip particular] astfel încât toate instanțele etichetate date să fie

³⁶⁴La problema 26 am demonstrat un astfel de rezultat folosind SVM și alegând în mod convenabil o anumită valoare pentru σ . Aici vom formula o problema de optimizare convexă similară, însă sensibil diferită de problema SVM.

³⁶⁵O altă variantă a acestui rezultat poate fi obținută folosind regresia liniară kernel-izată, cu funcție-nucleu de tip RBF. A se vedea problema 10 de la capitolul *Metode de regresie*.

vectori-suport. Evident, această proprietate nu poate fi satisfăcută pentru orice funcție-nucleu validă, însă vom păstra pentru început o generalitate mai mare decât este nevoie — la punctele a și b vom considera K funcție-nucleu oarecare și doar începând cu punctul c vom restricționa K la funcția-nucleu RBF, definită în relația (139) — și vom vedea că astfel ne va fi mai ușor să atingem obiectivul pe care ni l-am propus inițial. În al doilea rând, observăm că formularea noastră omite termenul liber (bias-ul, notat în general cu b sau cu w_0), din cauză că el nu este necesar pentru demonstrarea proprietății enunțate la acest punct (a).

Introduceți multiplicatori Lagrange pentru restricțiile problemei de optimizare date — similar cu cele de la problema SVM duală, veți problema 10 — și indicați forma generală a soluției, w^* . Pentru simplificarea calculelor / raționamentului, vă recomandăm / permiteți să lucrați ca și cum vectorii w și $\phi(x_i)$ ar avea întotdeauna dimensiune finită (deși nu acesta este cazul funcțiilor-nucleu RBF; veți problema 67 de la capitolul *Fundamente*).

Observație: Întrucât încercăm să satisfacem restricții de tip egalitate, multiplicatorii Lagrange pot lua orice valori reale. Așadar, spre deosebire de problema de optimizare SVM standard, aici multiplicatorii Lagrange nu [mai] sunt constrânsi să ia valori pozitive.

Vă cerem să exprimați w^* , soluția problemei de optimizare (140), în funcție de acești multiplicatori Lagrange. (Aceasta n-ar trebui să implice calcule prea elaborate.)

b. Introduceți soluția w^* pe care ati obținut-o la punctul precedent în *restricțiile* de clasificare (corespunzătoare „marginilor“) din problema de optimizare în formă primală (140) și exprimați rezultatul sub forma unei combinații liniare de aplicări ale funcției-nucleu K .

c. Arătați în manieră succintă cum anume se poate folosi *teorema lui Michelli* — redată mai jos — pentru a demonstra că pentru orice funcție-nucleu RBF K , matricea pătratică definită prin $K_{ij} = K(x_i, x_j)$ conform relației (139) pentru i și $j \in \{1, \dots, n\}$ este inversabilă.

Teoremă (Michelli, 1986): Dacă $\rho : [0, \infty) \rightarrow \mathbb{R}$ este o funcție monotonă pe intervalul de definiție, atunci pentru orice set de puncte distincte x_i dintr-un spațiu \mathbb{R}^l , cu $i = 1, \dots, n$, matricea pătratică — la care ne vom referi, prin abuz de notație, tot cu ρ — ale cărei elemente sunt $\rho_{ij} \stackrel{\text{def.}}{=} \rho(\|x_i - x_j\|)$, este inversabilă.

d. Coroborând rezultatele obținute la punctele precedente, demonstrați că într-adevăr se poate găsi o soluție a problemei de optimizare date în forma primală (140) [și, în consecință, toate instanțele x_i vor fi vectori-suport].

e. Desigur, faptul că putem separa, în principiu, orice multime de exemple de antrenament nu implică în mod neapărat că un clasificator oarecare antrenat pe aceste date se comportă bine pe date de test. (Din contra!) Așadar, cum se justifică faptul că folosim [pentru clasificare, în particular cu SVM] nucleu RBF? Răspunsul ține de „marginea“ maximă [LC: de separare] pe care o putem atinge atunci când variem parametrul σ .

Observație: Variind valoarea lui σ , efectul asupra marginii [LC: geometrică] nu este pur și simplu [dat de] scalarea vectorilor de trăsături $\phi(x_i)$. Într-adevăr, este ușor să observăm că pentru [orice] nucleu RBF, adică pentru orice valoare a lui σ , avem

$$\|\phi(x)\|^2 = \phi(x) \cdot \phi(x) = K(x, x) = 1 \quad (\text{valoarea maximă posibilă pentru orice RBF}).$$

Să începem prin a atribui lui σ o valoare pozitivă foarte mică. Cât este — în acest caz — valoarea marginii $1/\|w^*\|$ pe care o obținem ca răspuns [la rezolvarea problemei de optimizare (140)] pentru orice set de n instanțe de antrenament distincte?

f. Alegeți un set de instanțe din spațiul unidimensional (\mathbb{R}) în aşa fel încât să arătați că marginea [LC: geometrică, văzută ca distanța până la hiperplanul $w^* \cdot x = 0$] poate fi mai mare decât cea obținută ca răspuns la punctul e (pentru $\sigma \rightarrow 0$). Puteți da orice valori pentru σ și puteți seta instanțele cum doriti, astfel încât să puneti în evidență modul în care ele pot determina „marginea” / distanța maximă dintre ele.

59.

(SVM multiclass cu margine “soft”: verificarea echivalenței cu C-SVM în cazul clasificării binare)
MIT, 2016 spring, David Sontag, HW2, pr. 4

Clasificatorul *SVM multiclass* constituie o generalizare a clasificatorului binar SVM de la două clase la un număr oarecare de clase, $K \geq 2$. După cum am arătat la problema 29 (unde am prezentat varianta SVM-multiclass cu margine “hard”), aceasta implică introducerea unui vector de ponderi $w^{(k)}$ și a unui termen liber $b^{(k)}$ pentru fiecare clasă $k \in \{1, \dots, K\}$.³⁶⁶

Ca și la alte variante ale clasificatorului [C-]SVM, și în cazul de față învățarea / antrenarea constă în rezolvarea unei *probleme de optimizare*:

$$\begin{aligned} & \min_{w^{(k)}, b^{(k)}, \xi} \left(\frac{1}{2} \|w^{(k)}\|^2 + C \sum_i \xi_i \right) \\ \text{a. i. } & w^{(y_i)} \cdot x_i + b^{(y_i)} \geq w^{(k)} \cdot x_i + b^{(k)} + 1 - \xi_i, \text{ pentru } i = 1, \dots, m \text{ și } k \neq y_i \\ & \xi_i \geq 0, \forall i = 1, \dots, m. \end{aligned}$$

Observați că în această problemă, pentru fiecare instanță de antrenament x_i apare — ca și în cazul problemei C-SVM, veДЕti pr. 13 — câte o variabilă de destindere (engl., slack variable) ξ_i , însă aici asociem la fiecare instanță [un număr de] $K - 1$ restricții.

Pentru o instanță nouă x , predicția se va face folosind regula următoare:

$$y = \arg \max_k (w^{(k)} \cdot x + b^{(k)}). \quad (141)$$

În acest exercițiu veți compara această regulă de predicție (de tip multiclass) în cazul particular în care se lucrează cu $K = 2$ cu regula de predicție pe care am folosit-o pentru clasificatorul [binar] C-SVM (veDEti pr. 13.f):

$$\text{sign}(w \cdot x + b). \quad (142)$$

Concret, veți arăta că fiecare dintre cele două reguli se reduce la [adică, este echivalentă cu] cealaltă.

a. Calculați w și w_0 în funcție de $w^{(1)}, b^{(1)}, w^{(2)}$ și $b^{(2)}$ astfel încât pentru fiecare instanță x rezultatul obținut folosind noua regulă de predicție binară (142) să fie același cu rezultatul care se obține dacă se aplică regula de predicție multiclass (141) care folosește $w^{(1)}, b^{(1)}, w^{(2)}$ și $b^{(2)}$.

b. La acest punct veți proceda exact invers față de punctul precedent: Considerând w și w_0 ca fiind date, calculați $w^{(1)}, b^{(1)}, w^{(2)}$ și $b^{(2)}$ (în funcție de w și w_0) astfel încât pentru fiecare instanță x predicția făcută cu ajutorul reguli de predicție multiclass (cu $K = 2$) să fie același cu rezultatul care s-ar obține dacă am folosi regula de predicție binară cu respectivele valori (date) pentru w și w_0 .

³⁶⁶La problema 29, din considerente care ţin (doar) de simplitatea formulării, nu s-au folosit termeni liberi. Adăugarea lor se poate face în mod natural.

60. (O legătură între *one-class SVM* (versiunea *Max Margin*) și SVM (cu și respectiv fără termen liber (engl., bias))
prelucrare de Liviu Ciortuz, după MIT, 2009 fall, Tommi Jaakkola, midterm, pr. 3

Un prieten ne-a spus că el poate să emuleze clasificatori de tip SVM folosind doar cod care a fost dezvoltat pentru detecția „anomalilor“, mai precis algoritmul *one-class SVM*, versiunea *Max Margin*, pe care l-am prezentat la pr. 30. Rutinele de antrenare și respectiv testare despre care el afirmă că sunt suficiente pentru acest scop sunt următoarele:³⁶⁷

$$(\hat{w}, \hat{\rho}) = \text{train}(\phi_1, \dots, \phi_m) : \begin{aligned} & \text{Minimize}_{w, \rho} \left(\frac{1}{2} \|w\|^2 - \rho \right) \\ & \text{subject to } w \cdot \phi_i \geq \rho, \quad i = 1, \dots, m \\ & \text{Return } \hat{w}, \hat{\rho} \end{aligned}$$

$$y = \text{test}(w, \rho, \phi) : \text{Return } +1 \text{ if } w \cdot \phi \geq \rho \text{ else return } -1$$

unde ϕ_1, \dots, ϕ_m și ϕ sunt vectori de trăsături.

Noi n-am fost siguri dacă el are dreptate sau nu, aşa că ne-am decis să verificăm.
Să începem prin a găsi clasificatorul SVM fără termen liber (engl., bias) de forma

$$\hat{y} = \text{sign}(w \cdot \phi(x))$$

corespunzător unui set de m instanțe de antrenament $\phi(x_1), \dots, \phi(x_m)$ având etichetele $y_1, \dots, y_m \in \{+1, -1\}$. Presupunem că setul de date de antrenament este liniar separabil.

- a. Precizați care sunt vectorii de trăsături pe care ar trebui să-i transmitem [pentru antrenare] rutinei **train**?
- b. Fie \hat{w} și $\hat{\rho}$ parametrii returnați de către rutina **train** în urma alegerii făcute la punctul precedent pentru vectorii de trăsături [pentru antrenare]. Care sunt argumentele pe care ar trebui să le transmitem rutinei **test** astfel încât ea să clasifice instanța de test $\phi(x)$ identic cu clasificatorul de margine maximă?
- c. Cât este — în funcție de \hat{w} și $\hat{\rho}$ — marginea geometrică pe care o obține clasificatorul SVM pe setul de date de antrenament?
- d. Încurajați de aceste rezultate, ne punem întrebarea dacă nu cumva ar fi posibil să antrenăm și clasificatorul SVM care include și termenul liber, adică $\hat{y} = \text{sign}(w \cdot \phi(x) + w_0)$, folosind [doar] cele două rutine. Este într-adevăr posibil? Justificați riguros.

61. (Rezolvarea problemei *one-class*, varianta *Max Margin*, cu margine “soft”, folosind abordarea de la ν -SVM)
prelucrare de Liviu Ciortuz, după MIT, 2009 fall, Tommi Jaakkola, ML course, lecture notes 5

Vom reveni aici asupra problemei de optimizare *SVM one-class* varianta *Max Margin* (vedeți pr. 30). Mai întâi precizăm că, în ceea ce privește versiunea cu margine “hard”,

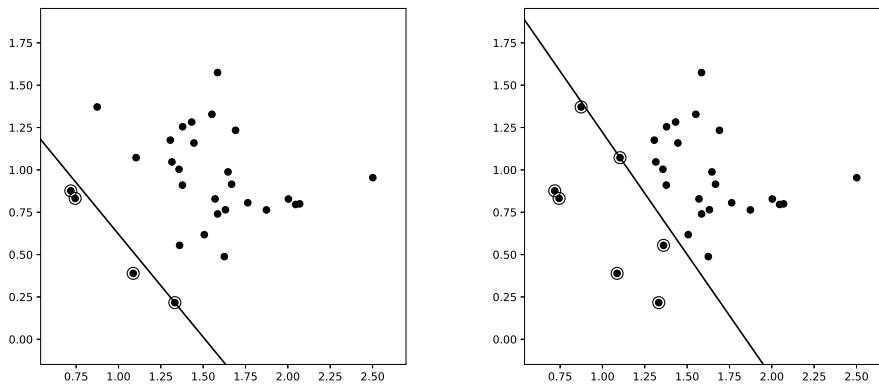
³⁶⁷Vedeți *Observația* (2) de la pr. 30.

față de forma primală pe care am considerat-o în problema menționată, acum vom lucra cu o versiune mai generală:³⁶⁸

$$\min_{w,\rho} \left(\frac{1}{2} \|w\|^2 - \rho \right)$$

a. î. $w \cdot x_i \geq \rho$, pentru $i = 1, \dots, m$.

Forma aceasta este mai convenabilă pentru *obiectivul* pe care ni-l fixăm aici, acela de a elabora cazul marginii “soft” pentru problema *one-class SVM* (Max Margin) urmând abordarea de tip ν -SVM. (A se vedea problema 32.)



Vă reamintim că ν -SVM folosește un parametru numeric (ν) care va funcționa ca margine superioară pentru proporția de erori la antrenare din totalul instanțelor de antrenament.

Forma primală pe care o vom considera aici pentru problema ν -SVM *one-class* (Max Margin) este de asemenea ușor schimbăț în raport cu formularea originală a problemei ν -SVM (B. Schölkopf, A. Smola, R. Williamson și P. Bartlett, *New Support Vector Machines*, 2000):³⁶⁹

$$\min_{w,\xi,\rho} \left(\frac{1}{2} \|w\|^2 - \rho + \frac{1}{\nu m} \sum_{i=1}^m \xi_i \right)$$

a. î. $w \cdot x_i \geq \rho - \xi_i$, pentru $i = 1, \dots, m$ (P'')

$\xi_i \geq 0$ pentru $i = 1, \dots, m$.

Remarcați faptul că în ambele probleme de optimizare date mai sus n-au fost impuse restricții asupra variabilei ρ .³⁷⁰

a. Demonstrați că forma duală corespunzătoare formei primale (P'') a problemei ν -SVM *one-class* (Max Margin) este următoarea:

³⁶⁸Distanța de la hiperplanul de separare optimală la vectorii-suport care nu produc erori în raport cu marginea va fi $\frac{\rho}{\|w\|}$. Din punctul de vedere al „marginii“ geometrice, noi am vrea ca [și] ρ să fie maximizat. Aceasta echivalează cu a minimiza $-\rho$. Așa se justifică (în mod intuitiv) expresia funcției obiectiv din problema de optimizare pe care urmează să o formulăm.

³⁶⁹A se vedea și problema 32.

³⁷⁰La problema 32 aveam $\rho \geq 0$.

$$\begin{aligned} \max_{\alpha} & \left(-\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j x_i \cdot x_j \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq \frac{1}{\nu m} \text{ pentru } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i = 1. \end{aligned} \tag{D''}$$

b. Scrieți condițiile de complementaritate Karush-Kuhn-Tucker pentru problema primală (P''), apoi arătați cum anume se poate calcula $\bar{\rho}$, valoarea rezultată pentru variabila ρ la rezolvarea problemei duale (D''), în funcție de valorile celorlalte variabile ($\bar{w}, \bar{\alpha}_i, \bar{\xi}_i$, etc).

62. (Problema [one-class SVM, varianta] *sferei de inclusiune minimală*
 (engl., minimum enclosing ball, MEB)
 în varianta cu margine “soft”, folosind ν -SVM
prelucrare de Liviu Ciortuz, după
 ■ MIT, 2009 fall, Tommi Jaakkola, ML course, lecture notes 5

Date fiind instanțele $x_1, \dots, x_m \in \mathbb{R}^d$, ne propunem să găsim o sferă care să includă toate punctele x_i și să aibă cea mai mică rază posibilă. Pentru aceasta, formulăm următoarea problemă de optimizare convexă:

$$\min_{R,w} R^2 \text{ astfel încât } \|w - x_i\|^2 \leq R^2 \text{ pentru } i = 1, \dots, m.$$

De remarcat faptul că restricțiile din formularea acestei probleme sunt de ordin pătratic (spre deosebire de cazul problemei SVM clasice, în care restricțiile sunt liniare). La finalul rezolvării acestei probleme,

- valoarea găsită pentru w va reprezenta centrul sferei;
- vectorii-suport vor fi punctele x_i de pe suprafața sferei; restricțiile corespunzătoare lor vor fi satisfăcute cu egalitate ($\|w - x_i\|^2 = R^2$).

Ca și în cazul problemei 61, putem impune condiția ca maximum νm puncte (din totalul celor m) să fie lăsate în afara sferei. Corespunzător, problema de optimizare convexă de tip ν -SVM va fi:

$$\begin{aligned} \min_{R,w,\xi} & \left(R^2 + \frac{1}{\nu m} \sum_{i=1}^m \xi_i \right) \\ \text{astfel încât } & \|w - x_i\|^2 \leq R^2 + \xi_i \text{ pentru } i = 1, \dots, m \\ & \xi_i \geq 0 \text{ pentru } i = 1, \dots, m. \end{aligned} \tag{P^{iv}}$$

- a. Demonstrați că forma duală a problemei de tip ν -SVM de mai sus este:

$$\begin{aligned} \max_{\alpha} & \left(-\sum_i \sum_j \alpha_i \alpha_j x_i \cdot x_j + \sum_{i=1}^m \alpha_i x_i \cdot x_i \right) \\ \text{a. i. } & 0 \leq \alpha_i \leq \frac{1}{\nu m} \text{ pentru } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i = 1 \end{aligned} \tag{D^{iv}}$$

unde α_i este multiplicatorul Lagrange corespunzător restricției $i \in \{1, \dots, m\}$.

b. Indicați relația de legătură dintre soluția \bar{w} a problemei primale și soluția $\bar{\alpha}$ a problemei duale. (Remarcați semnificația geometrică a rezultatului!)

c. Cum se poate calcula valoarea optimă \bar{R} (pentru variabila R din forma primală (P^{iv})) pornind de la soluția problemei duale?

Sugestie: Pentru aceasta, este util ca, pentru problema de tip ν -SVM de mai sus (D^{iv}), să enunțați condițiile de complementaritate Karush-Kuhn-Tucker.

63. (SVR cu margine “hard”, cazul liniar: exemplu de aplicare)
CMU, 2008 fall, Eric Xing, midterm, pr. 3

Pentru *regresie*, ni se dau m exemple de antrenament (x_1, y_1) , (x_2, y_2) , ..., (x_m, y_m) , unde fiecare x_i este un vector de trăsături, iar $y_i \in \mathbb{R}$ este outputul pentru exemplul de pe poziția i ($i = 1, 2, \dots, m$).

Vă readucem aminte că în cazul *regresiei liniare cu vectori-suport* (engl., support vector regression, SVR),³⁷¹ obiectivul este ca, pornind de la setul de exemple de antrenament, să învățăm o funcție liniară de forma $f(x) = w \cdot x + b$, unde w și b sunt parametrii care trebuie „învățați“, iar w este vectorul de ponderi. Problema de *optimizare convexă* pentru SVR liniară poate fi formulată în felul următor:

$$\min_w \frac{1}{2} \|w\|^2 \quad (143)$$

$$\text{a. i. } y_i - (w^\top x_i + b) \leq \varepsilon \text{ și } (w \cdot x_i + b) - y_i \leq \varepsilon \quad (i = 1, \dots, m). \quad (144)$$

Ideea din spatele relațiilor (143) și (144) este următoarea: dorim să „învățăm“ parametrii w și b astfel încât (i) funcția f obținută să fie cât mai „netedă“ (engl, smooth) cu puțință (așadar, urmărim ca $\|w\|$ să fie cât mai mic), iar (ii) pentru fiecare exemplu de antrenament, eroarea în raport cu predicția făcută de către funcția f să fie de cel mult ε , unde $\varepsilon \geq 0$ este un *parametru* dat / fixat în acest algoritm.³⁷²

Acum, date fiind 3 exemple de antrenament, (1, 1), (2, 2) și (3, 3), vrem să folosim relațiile (143) și (144) pentru a învăța un model liniar SVR.

- a. Cât este w atunci când $\varepsilon = 0$?
- b. Cât este w atunci când $\varepsilon = 0.5$?
- c. Cât este w atunci când $\varepsilon = 1$?

64. (SVR cu margine “soft”: varianta care folosește funcție de cost / pierdere ε -sensitivă)

■ *CMU, 2014 spring, B. Poczos, A. Singh, HW2, pr. 1*
CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW3, pr. 2.1-4

În acest exercițiu veți deduce forma duală a problemei de optimizare pentru regresie cu vectori-suport (SVR) cu margine “soft”, care folosește o funcție de cost ε -sensibilă (engl., epsilon-sensitive loss), dată de expresia

$$L_\varepsilon(x, y, f) = |y - f(x)|_\varepsilon \stackrel{\text{not.}}{=} \max(0, |y - f(x)| - \varepsilon), \quad (145)$$

³⁷¹Vedeți problema 33.

³⁷²Remarcați faptul că pe măsură ce valorile lui ε cresc, funcția f pe care o căutăm devine tot mai „netedă“.

unde x este inputul, y este outputul, iar f este funcția folosită pentru a predicție.³⁷³ Setul de date de antrenament este $(x_1, y_1), \dots, (x_n, y_n)$, unde $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$.³⁷⁴

Folosind această notație, funcția obiectiv pentru problema SVR este definită astfel:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L_\varepsilon(x_i, y_i, f),$$

unde $f(x) = w \cdot x$, iar $C > 0$ și $\varepsilon > 0$ sunt parametri.

a. Scrieți această problemă ca o problemă de optimizare pătratică cu restricții liniare, folosind variabile de „destindere“ (engl., slack variables). Această formă este numită *forma primală* a problemei de optimizare SVR cu margine “soft”.

Sugestie: Expresia funcției de cost (145) nu trebuie să vă jeneze. Practic va trebui să procedați în manieră inversă față de cum s-a procedat la problema 21, unde pornind de la forma primală a problemei C-SVM se obține o formă echivalentă care folosește funcția de cost *hinge*. Alternativ, puteți pleca de la forma primală a problemei SVR cu margine “hard” (vedeți pr. 33) și introduceți variabile de destindere ξ_i similar modului în care am procedat pentru a obține clasificatorul C-SVM din clasificatorul SVM (vedeți pr. 13).

b. Scrieți *funcția lagrangeană* pentru *forma primală* a problemei SVR pe care ați obținut-o la punctul precedent.

c. Folosind *condițiile Karush-Kuhn-Tucker*, deduceți *forma duală* a acestei probleme SVR.

d. Am putea folosi solver-e [LC: adică programe de analiză numerică] de probleme de *optimizare pătratică* pentru a rezolva problema duală SVR dedusă mai sus?

e. Cum ați putea defini *vectorii-suport* pentru această problemă SVR?

f. Scrieți expresia care poate fi folosită pentru a face *predicția* etichetei pentru o instanță nouă, x .

g. Este oare posibil să kernel-izăm acest algoritm?

h. Formulați motivul pentru care în general rezolvăm problemele SVR și SVM în varianta duală, nu în cea primală.

i. Ce se întâmplă atunci când modificăm valoarea parametrului ε ?

j. Ce se întâmplă atunci când modificăm valoarea parametrului C ?

65.

(Exercițiu recapitulativ: corespondența dintre diverse funcții de cost (respectiv diverse tipuri de granițe de decizie) și diferite metode de învățare automată)

CMU, 2014 spring, B. Poczos, A. Singh, midterm, pr. 2.1-2

a. Puneți în corespondență fiecare dintre metodele de învățare automată de mai jos (din coloana din stânga) cu una dintre funcțiile de cost / pierdere (engl., loss functions) din coloana din dreapta, și anume acea funcție de cost pe care respectiva metodă de învățare o minimizează în cazul cel mai frecvent:

³⁷³Notând $z = y - f(x)$, este foarte util — pentru a înțelege mai bine ceea ce urmează — să faceti graficul funcției $\max(0, |z| - \varepsilon)$.

³⁷⁴Remarcați faptul că funcția de cost / pierdere *hinge* pe care am folosit-o la problema 21 (în legătură cu C-SVM) este adecvată doar pentru clasificare, nu și pentru regresie.

regresia liniară	funcția de cost pătratică
regresia logistică	funcția de cost logistică
AdaBoost	funcția de cost exponentială
k -NN	funcția de cost $0 - 1$
SVM	funcția de cost <i>hinge</i>

b. Pentru fiecare dintre metodele de învățare automată de mai jos (din coloana din stânga), încercuiți tipul / tipurile de separator decizional (graniță de decizie) pe care respectiva metodă îl poate produce la efectuarea unui task de clasificare binară. În unele cazuri, este posibil ca mai multe opțiuni să fie corecte. Veți încercui toate opțiunile pe care le considerați corecte.

regresia logistică:	liniar
ID3 (cu atrbute numerice continue):	liniar, combinăție de separatori liniari ³⁷⁵
AdaBoost:	liniar, combinăție de separatori liniari, pătratic
clasificatorul Bayes Naiv gaussian:	liniar, pătratic
SVM (fără funcție-nucleu):	liniar

³⁷⁵ Engl., piecewise linear.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

6 Rețele neuronale artificiale

Sumar

Noțiuni preliminare

- funcție matematică; compunere de funcții reale;
calculul valorii unei funcții pentru anumite valori specificate pentru argumentele / variabilele ei;
- funcție prag (sau, treaptă), funcție liniară, funcție sigmoidală (sau, logistică), funcție sigmoidală generalizată;
separabilitate liniară pentru o mulțime de puncte din \mathbb{R}^d ;
- ecuații asociate dreptelor în plan / planelor în spațiu / hiper-planelor în spațiul \mathbb{R}^d ;
ecuația dreptei în plan care trece prin două puncte date;
semnele asociate punctelor din semiplanele determinate de o dreaptă dată în plan;
- derivate ale funcțiilor elementare de variabilă reală; derivate parțiale
- vectori; operații cu vectori, în particular produsul scalar al vectorilor;
- metoda gradientului descendente (ca metoda de optimizare); avantaje și dezavantaje;
ex. 72, ex. 149 și ex. 150 de la cap. *Fundamente*; ex. 23.

Câteva noțiuni specifice

- *unități* neuronale artificiale (sau, *neuroni artificiali, perceptroni*);
tipuri de neuroni artificiali: neuroni-prag, liniari, sigmoidali;
componente ale unui neuron artificial: input, componenta de sumare, componenta / funcția de activare, output;
funcția matematică reprezentată / calculată de un neuron artificial;
- *rețea* neuronală artificială; rețele de tip feed-forward;
niveluri / straturi de neuroni, niveluri ascunse, niveluri de ieșire;
ponderi asociate conexiunilor dintr-o rețea neuronală artificială;
funcția matematică reprezentată / calculată de o rețea neuronală artificială;
granițe și zone de decizie determinate de o rețea neuronală artificială;
funcția de eroare / cost (engl., loss function).

Câteva proprietăți relative la expresivitatea rețelelor neuronale artificiale

- (P0) Toate cele trei tipuri de neuroni artificiali (prag, liniar, sigmoidal) produc *separatori liniari*.
Consecință: Conceptul XOR nu poate fi reprezentat / învățat cu astfel de „dispozitive“ simple de clasificare.
- (P0') Rețelele neuronale artificiale pot determina granițe de decizie neliniare (și, în consecință, pot reprezenta concepte precum XOR).
Observație: Rețele de unități sigmoidale pot determina granițe de decizie curbilinii: ex. 8.

- (P1) Rețele de neuroni diferite (ca structură și / sau tipuri de unități) pot să calculeze o aceeași funcție: ex. 3 și ex. 1.c vs. ex. 2.
- (P1') Dată o topologie de rețea neuronală (i.e., graf de unități neuronale al căror tip este lăsat nespecificat), este posibil ca plasând în noduri unități de un anumit tip să putem reprezenta / calcula o anumită funcție, iar schimbând tipul unora dintre unități (sau al tuturor unităților), funcția respectivă să nu mai potă fi calculată: ex. 4 vs. ex. 3.³⁷⁶
- (P2) Orice unitate liniară situată pe un nivel ascuns poate fi „absorbită“ pe nivelul următor: ex. 32.
- (P3) Orice funcție booleană poate fi reprezentată cu ajutorul unei rețele neuronale artificiale având doar două niveluri de perceptriони-prag: ex. 5.
- (P4) Orice funcție definită pe un interval mărginit din \mathbb{R} , care este continuă în sens Lipschitz, poate fi aproximată oricără de bine cu ajutorul unei rețele neuronale care are un singur nivel ascuns: ex. 7.

Algoritmi de antrenare a neuronilor artificiali folosind metoda gradientului descendente

- algoritmul de antrenare a unității liniare: ex. 34; vedeti T. Mitchell, *Machine Learning*, p. 93, justificare: p. 91-92; convergență: p. 95; exemplu de aplicare: ex. 10; varianta incrementală a algoritmului de antrenare a unității liniare: cartea ML, p. 93-94; despre convergență acestei variante (ca aproximare a variantei precedente (“batch”)): cartea ML, p. 93 jos;
- algoritmul de antrenare a perceptronului-prag și convergență: cartea ML, p. 88-89; exemplu de aplicare: ex. 11;
- algoritmul de antrenare a perceptronului sigmoidal și justificarea sa teoretică: cartea ML, p. 95-97;
- algoritmul *Perceptron* al lui Rosenblatt; exemplu de aplicare: ex. 16, ex. 36;
- deducerea regulii de actualizare a ponderilor pentru tipuri particulare de perceptriони: ex. 12, ex. 25.a, ex. 35, ex. 13.a;
- o justificare probabilistă (gen ipoteză de tip *maximum likelihood*) pentru minimizarea sumei pătratelor erorilor [la deducerea regulii de antrenare] pentru perceptronul liniar: ex. 13.b;
- exemple de [folosire a unei] alte funcții de cost / pierdere / penalizare (engl., loss function) decât semisuma pătratelor erorilor: suma costurilor de tip log-sigmoidal, ex. 14 (pentru perceptronul liniar), o funcție de tip cross-entropie, ex. 15 (pentru perceptronul sigmoidal).

Perceptronul Rosenblatt și rezultate de convergență

- exemplu de aplicare [adică, învățare cu perceptronul Rosenblatt]: ex. 16.
- câteva *proprietăți* simple ale perceptronului Rosenblatt: ex. 17.
- rezultate de convergență de tip “mistake bound” pentru [algoritmul de antrenare pentru] perceptronul-prag [în varianta] Rosenblatt: ex. 18, ex. 37; pentru perceptronul-prag (clasic): ex. 39; învățare online cu perceptronul-prag de tip Rosenblatt: ex. 38;

³⁷⁶Problemele 1.d și ex. 31 au în vedere o chestiune similară, însă pentru rețele cu topologii diferite: o anumită extensie a funcției XOR nu poate fi reprezentată pe rețele de neuroni-prag care au un singur nivel ascuns.

- *Perceptronul kernel-izat [dual]*: ex. 19; particularizare pentru cazul nucleului RBF: ex. 48.

Antrenarea rețelelor neuronale artificiale:

algoritmul de *retro-propagare* pentru rețele feed-forward

- T. Mitchell, *Machine Learning*, p. 98: pseudo-cod pentru rețele cu unități de tip sigmoidal, cu 2 niveluri, dintre care unul ascuns; pentru deducerea regulilor de actualizare a ponderilor; în cazul mai general al rețelelor feed-forward (de unități sigmoidale) cu oricâte niveluri, vedeti p. 101-103;
ex. 20: deducerea regulilor de actualizare a ponderilor în cazul rețelelor cu 2 niveluri, având însă unități cu funcție de activare oarecare (derivabilă);
- aplicare: ex. 21, ex. 41, ex. 42;
- prevenirea overfitting-ului:
folosirea unei componente de tip „moment“ în expresia regulilor de actualizare a ponderilor: ex. 44;
regularizare: introducerea unei componente suplimentare în funcția de optimizat: ex. 22;
- cazul folosirii unei funcții de activare de tip tangentă hiperbolică: ex. 43;
- cazul folosirii unei funcții de cost / penalizare / eroare de tip cross-entropie: ex. 46;
- execuția manuală a unei iterații a algoritmului de retro-propagare în cazul unei rețele neuronale simple, având un singur nivel ascuns, cu unități ce folosesc funcția de activare ReL: ex. 47.

Rețele neuronale profunde — câteva chestiuni introductive

- analiza convexității unor funcții de cost folosite în învățarea profundă: ex. 52;
- fenomenul de „dispariție“ a gradientului [în cazul aplicării algoritmului de retro-propagare] pentru rețele neuronale profunde (engl., deep neural networks) care folosesc funcția de activare sigmoidală: ex. 26;
- determinarea numărului de parametri și de conexiuni din rețea neuronală convolutivă LeNet: ex. 27;
- determinarea mărimii hărții de trăsături de pe un anumit nivel, precum și a numărului de operații în virgulă mobilă (FLOPs) executate la procesarea forward într-o rețea neuronală convolutivă: ex. 53.

6.1 Probleme rezolvate

6.1.1 Chestiuni introductive

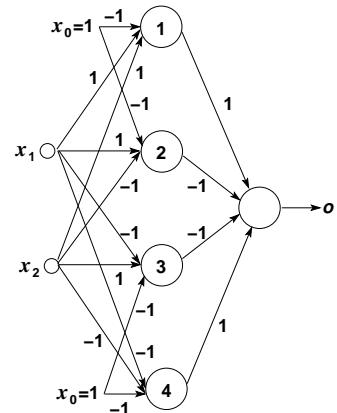
1. (Rețele de perceptroni-prag, exemplificare: calculul outputului)
prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Aarti Singh, HW5, pr. 4.1.1

Considerăm rețeaua neuronală din figura următoare.

Toate unitățile acestei rețele neuronale sunt de tip prag, adică folosesc pentru activare funcția $sign$ definită prin $sign(z) = 1$ dacă $z \geq 0$ și -1 în rest. Pentru unitatea de pe nivelul de ieșire (lăsată nenumerotată), ponderea corespunzătoare termenului liber ($x_0 = 1$) este 0.

a. Scrieți funcția matematică calculată de fiecare dintre unitățile rețelei, în raport cu intrările x_1 și x_2 . Veți nota cu o_i (unde $i = 1, \dots, 4$) ieșirile unităților de pe nivelul ascuns și cu o ieșirea rețelei, adică valoarea produsă de către unitatea de pe nivelul de ieșire.

b. Calculați outputul rețelei atunci când intrările x_1 și x_2 iau valori în mulțimea $\{-1, 1\}$.



c. Indicați funcțiile booleene reprezentate de către uitățile de pe nivelul ascuns (1, 2, 3 și 4) atunci când $x_1, x_2 \in \{-1, 1\}$. Procedați similar pentru ieșirea o .

d. Specificați cum anume ar putea fi modificată rețea dată astfel încât noua variantă să calculeze funcția de variabile reale (nu booleene ca mai înainte!) x_1 și x_2 , a cărei relație de definiție este: $f(x_1, x_2) = 1$ dacă $x_1, x_2 \geq 0$ sau $x_1, x_2 < 0$, și -1 în caz contrar.

Răspuns:

a. Sunt imediate următoarele relații:

$$\begin{aligned} o_1(x_1, x_2) &= sign(x_1 + x_2 - 1), \\ o_2(x_1, x_2) &= sign(x_1 - x_2 - 1), \\ o_3(x_1, x_2) &= sign(-x_1 + x_2 - 1), \\ o_4(x_1, x_2) &= sign(-x_1 - x_2 - 1), \\ o(x_1, x_2) &= sign(o_1(x_1, x_2) - o_2(x_1, x_2) - o_3(x_1, x_2) + o_4(x_1, x_2)). \end{aligned}$$

b. Date fiind formulele de la punctul precedent, calculele cerute sunt simple; centralizăm rezultatele sub forma tabelului următor:

x_1	x_2	o_1	o_2	o_3	o_4	o
1	1	1	-1	-1	-1	1
1	-1	-1	1	-1	-1	-1
-1	1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	1	1

c. Din tabelul obținut la punctul precedent este imediat că, atunci când $x_1, x_2 \in \{-1, 1\}$, ieșirile calculate de către uitățile 1, 2, 3 și 4 corespund conceptelor / funcțiilor $x_1 \wedge x_2$, $x_1 \wedge \neg x_2$, $\neg x_1 \wedge x_2$ și respectiv $\neg x_1 \wedge \neg x_2$. Ieșirea rețelei, o , corespunde conceptului $\neg(x_1 \text{ XOR } x_2)$.

Observație: Se poate remarcă faptul că în această rețea un neuron (și doar unul!) de pe nivelul ascuns este activat la fiecare combinație de valori (din cele 4 posibile) pentru $x_1, x_2 \in \{-1, 1\}$.

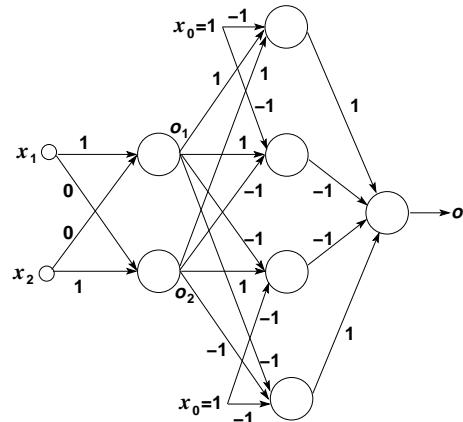
d. Este imediat că funcția indicată în enunț se poate scrie sub forma

$$f(x_1, x_2) = \begin{cases} 1 & \text{dacă } \text{sign}(x_1) \cdot \text{sign}(x_2) \geq 0 \\ -1 & \text{dacă } \text{sign}(x_1) \cdot \text{sign}(x_2) < 0. \end{cases}$$

Se observă imediat că aceasta coincide cu funcția $\neg(\text{sign}(x_1) \text{ XOR } \text{sign}(x_2))$.

Prin urmare, este suficient să adăugăm la rețeaua dată în enunț un nivel ascuns suplimentar, format din două unități cu funcție de activare de tip prag, care să transforme intrările x_1 și x_2 în $\text{sign}(x_1)$ și respectiv $\text{sign}(x_2)$. Vom obține ca rezultat rețeaua din figura alăturată.

Observație: La problema 31 vi se va cere să demonstrezi că această funcție de variabile reale nu poate fi calculată de nicio rețea neuronală care are un singur nivel ascuns și este compusă doar din unități [cu funcție de activare] de tip prag.



2.

(Reprezentarea unor funcții booleene cu ajutorul perceptronilor-prag sau al rețelelor de perceptri-prag)

Tom Mitchell, "Machine Learning", 1997, pr. 4.2
CMU, 1995 fall, Tom Mitchell, HW4, pr. 6

a. Concepți un perceptron care are

- funcție de activare de tip prag, cu valori -1 și $+1$,
- două intrări

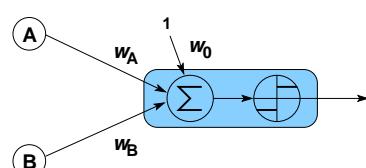
și implementează funcția booleană $A \wedge (\neg B)$.

b. Concepți o rețea neuronală formată din perceptri cu funcție de activare de tip prag dispusi pe două niveluri, care implementează funcția $A \text{ XOR } B$.

Răspuns:

Observație: Deoarece mulțimea de valori de ieșire a perceptronului este $\{-1, 1\}$, vom alege această codificare a valorilor fals / adevarat în locul celei clasice 0/1.

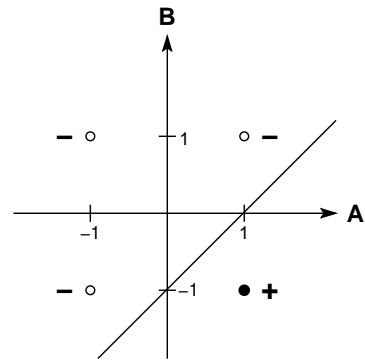
a. Perceptronul-prag are structura din figura alăturată. A îndeplini cerința din enunț revine la a alege în mod convenabil valori pentru ponderile w_0 , w_A și w_B .³⁷⁷



³⁷⁷Am putea satisface direct această cerință dacă ne raportăm la problema 1.c, în care $o_2(x_1, x_2) = x_1 \wedge \neg x_2$, deci am putea seta $w_0 = -1$, $w_A = 1$ și $w_B = -1$. Totuși, aici vom proceda independent de problema 1, arătând cum anume se rezolvă [în general] un exercițiu de acest tip.

Un perceptron consistent cu funcția booleană $A \wedge (\neg B)$ poate fi reprezentat ca o dreaptă ce separă instanțele pozitive de cele negative, ca în desenul alăturat. O astfel de dreaptă are ecuația $d(A, B) = 0$, unde $d(A, B) = w_0 + w_A A + w_B B$.

Este evident că sunt o infinitate de funcții care îndeplinesc proprietatea de separator liniar pentru cele două mulțimi, însă noi avem nevoie doar de una singură. Putem alege în mod convenabil / preferențial două puncte prin care să treacă dreapta, de pildă ca în figura alăturată. Așadar, *analytic*, considerăm astfel încât $(1, 0) \in d$ și $(0, -1) \in d$.



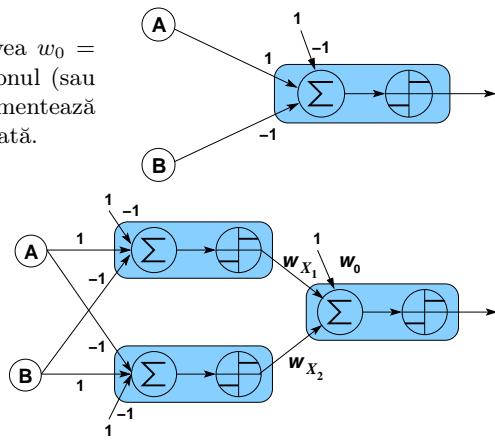
O astfel de alegere va impune anumite *restricții* asupra valorilor w_0, w_A și w_B . Într-adevăr, ținând cont de ecuația dreptei d scrisă mai sus (în sens generic), va rezulta următorul sistem:

$$\begin{cases} (1, 0) \in d \\ (0, -1) \in d \end{cases} \Rightarrow \begin{cases} w_0 + w_A = 0 \\ w_0 - w_B = 0 \end{cases} \Rightarrow \begin{cases} w_0 = -w_A \\ w_0 = w_B \end{cases}$$

Avem deci $w_0 = w_B \stackrel{\text{not.}}{=} \alpha, w_A = -\alpha$, unde $\alpha \in \mathbb{R}^*$. Prin urmare, $d(A, B) = \alpha - \alpha A + \alpha B$. Rămâne să mai analizăm în ce condiții dreapta d clasifică pozitiv instanța $(1, -1)$ și negativ instanțele $(1, 1)$, $(-1, 1)$ și $(-1, -1)$. De fapt, ținând cont de o proprietate din geometria analitică,³⁷⁸ este suficient să impunem una (de exemplu, prima) dintre aceste condiții. Aceasta revine la *restricția* ca expresia $\alpha - \alpha A + \alpha B$ să fie pozitivă pentru $A = 1$ și $B = -1$:

$$d(1, -1) > 0 \Leftrightarrow \alpha - \alpha - \alpha > 0 \Leftrightarrow \alpha < 0$$

Pentru fixare, alegem $\alpha = -1$, și vom avea $w_0 = -1, w_A = 1, w_B = -1$. Așadar, perceptronul (sau mai bine spus: un perceptron) care implementează funcția $A \wedge (\neg B)$ arată ca în figura alăturată.



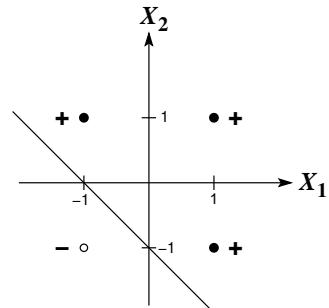
b.³⁷⁹ Știm că $A \text{ XOR } B = (A \wedge (\neg B)) \vee (B \wedge (\neg A))$. Această formulă poate fi reprezentată printr-o rețea de perceptriuni cu un nivel ascuns, ca în figura alăturată.

³⁷⁸ Este vorba despre următoarea proprietate din geometria plană: Toate punctele situate de o parte a dreptei de ecuație $w_0 + w_A A + w_B B$ au același semn, în vreme ce punctele situate de cealaltă parte a dreptei au semn contrar.

³⁷⁹ *Observație:* Am putea obține o soluție imediată pentru acest punct al problemei noastre dacă preluăm rețeaua neuronală dată în enunțul problemei 1 — despre care știm (vedeți rezolvarea respectivei probleme) că are outputul $\neg(x_1 \text{ XOR } x_2)$ — și schimbăm semnele tuturor ponderilor de pe arcele hidden-to-output din rețea. Vom arăta însă aici cum se poate construi „de la zero“ o astfel de rețea, pornind de la cerințele specificate în enunț. În plus, se va vedea la final că nouă

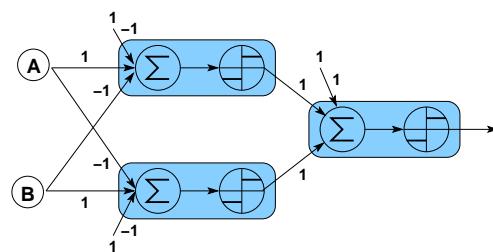
Perceptronii de pe nivelul de ascuns codifică funcția $A \wedge (\neg B)$ (vedeți punctul anterior) și respectiv $(\neg A) \wedge B$, după cum se poate vedea după ponderile de pe muchii, iar perceptronul de ieșire trebuie să descrie funcția logică \vee . Suprafața de decizie pentru aceasta din urmă este precum cea din figura alăturată.

Avem $d'(X_1, X_2) = w_0 + w_{X_1}X_1 + w_{X_2}X_2 = 0$. Fie d' dreapta care trece prin punctele $(0, -1)$ și $(-1, 0)$. Atunci:



$$\begin{cases} (0, -1) \in d' \\ (-1, 0) \in d' \end{cases} \Rightarrow \begin{cases} w_0 - w_{X_2} = 0 \\ w_0 - w_{X_1} = 0 \end{cases} \Rightarrow w_0 = w_{X_1} = w_{X_2} \stackrel{\text{not.}}{=} \alpha' \in \mathbb{R}^* \Rightarrow d'(X_1, X_2) = \alpha' + \alpha'X_1 + \alpha'X_2 = 0.$$

Împunând condiția referitoare la semne, vom avea $d'(-1, -1) < 0 \Leftrightarrow \alpha' - \alpha' - \alpha' < 0 \Leftrightarrow \alpha' > 0$. Pentru fixare, alegem $\alpha' = 1$, ceea ce duce la $w_0 = w_{X_1} = w_{X_2} = 1$, iar rețeaua neuronală va arăta astfel:



3.

(Rețele neuronale: exemplificare)

■ CMU, 2011 spring, Roni Rosenfeld, HW4, pr. 1.c

Să se reprezinte expresia booleană $(A \vee \neg B) \text{ XOR } (\neg C \vee D)$ printr-o rețea neuronală cu două niveluri care este formată din neuroni cu funcție de activare de tip prag.

Răspuns:

Observație: Este imediat — vedeti rezolvările problemelor 1 și / sau 2 — că se poate defini câte un perceptron cu funcție de activare de tip prag (engl., threshold perceptron) care să reprezinte funcțiile $(A \vee \neg B)$ și respectiv $(\neg C \vee D)$. Dacă ieșirile acestor doi perceptoni vor fi cuplate la intrările rețelei care reprezintă funcția $A \text{ XOR } B$ (a se vedea exercițiul 2, punctul b), atunci se va obține o rețea cu *trei* niveluri care reprezintă funcția $(A \vee \neg B) \text{ XOR } (\neg C \vee D)$. Exercițiul nostru cere însă să identificăm o rețea cu (doar!) *două* niveluri care să reprezinte această funcție.

rețea este mai simplă decât cea de la problema 1. (Așadar, rețele neuronale diferite pot codifica / reprezenta o același funcție reală. Evident, este de dorit ca, pentru o funcție dată, rețeaua neuronală care o codifică să fie cât mai simplă.)

Explicitând definiția funcției logice XOR și apoi aplicând regulile lui DeMorgan, expresia booleană din enunț devine:

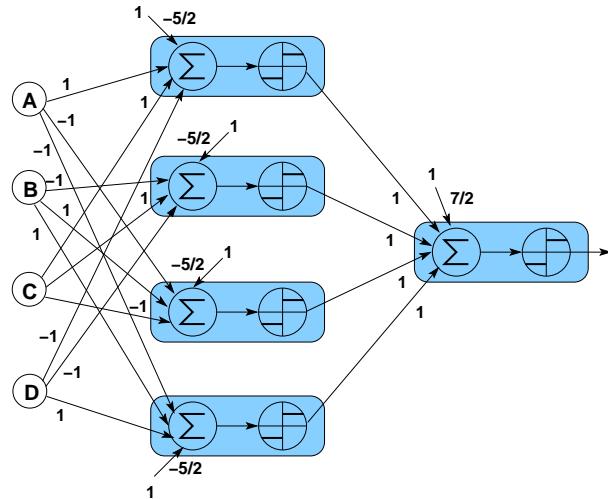
$$\begin{aligned}(A \vee \neg B) \text{ XOR } (\neg C \vee D) &= [(A \vee \neg B) \wedge \neg(\neg C \vee D)] \vee [\neg(A \vee \neg B) \wedge (\neg C \vee D)] \\ &= [(A \vee \neg B) \wedge (C \wedge \neg D)] \vee [(\neg A \wedge B) \wedge (\neg C \vee D)]\end{aligned}$$

Întrucât operatorii logici \vee și \wedge sunt distributivi unul față de celălalt, rezultă că:

$$(A \vee \neg B) \text{ XOR } (\neg C \vee D) = (A \wedge C \wedge \neg D) \vee (\neg B \wedge C \wedge \neg D) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge D)$$

Fiecare din cele patru paranteze din partea dreaptă a egalității de mai sus poate fi reprezentată cu ajutorul unui perceptron-prag cu patru intrări, și anume câte o intrare pentru fiecare din cele trei variabile booleene din paranteză, plus o intrare pentru termenul liber. (De exemplu, conjuncției $A \wedge C \wedge \neg D$ îi putem asocia inegalitatea $x + z - t > 5/2$, deci ponderile intrărilor perceptronului corespunzător vor fi $-5/2$, 1, 1 și -1 .) Cei patru perceptroni vor fi plasați pe primul nivel ascuns al rețelei pe care o construim.

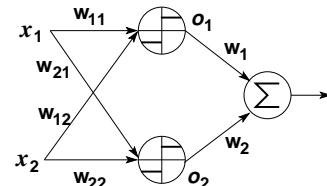
Apoi, ieșirile acestor patru perceptroni vor constitui intrări pentru un alt perceptron-prag, care să reprezinte disjuncția a patru variabile booleene. (Acestui perceptron, situat pe nivelul de ieșire, îi putem asocia, de exemplu, inegalitatea $x + y + z + t > -7/2$.) Rețeaua neuronală rezultată este cea din figura alăturată.



4.

(Verificarea (im)posibilității de a reprezenta funcția booleană XOR cu o rețea de structură / componiție specificată)
CMU, 2001 fall, Andrew Moore, final exam, pr. 10.a

Presupunem că lucrăm cu rețeaua neuronală din figura alăturată,³⁸⁰ care are un nivel ascuns format din doi perceptroni cu funcția de activare de tip prag: $sign(z) = 1$ dacă $z \geq 0$ și $sign(z) = -1$ dacă $z < 0$.



³⁸⁰Observație importantă: Pentru comoditate, vom opta aici — dar și în [multe dintre] problemele care urmează — pentru o reprezentare simplificată a unităților neuronale (comparativ cu reprezentările din problemele precedente). În figura dată, se va subîntâlege că neuroni de pe nivelul ascuns au (ca întotdeauna) o componentă de sumare (care aici nu este pusă în evidență), iar unitatea de pe nivelul de ieșire nu are o componentă de activare (deci este unitate liniară).

Folosim setul de date de antrenament din tabelul alăturat, care reprezintă funcția booleană *sau-exclusiv* dacă interpretăm -1 ca desemnând valoarea logică *fals*, și 1 ca *adevărat*.

X_1	X_2	Y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

Vă cerem să indicați ce valori putem atribui ponderilor rețelei date, astfel încât să obținem eroare nulă la antrenare. Dacă este imposibil să se găsească astfel de valori pentru ponderile rețelei, dați răspunsul *Imposibil*.

Observație: În raport cu problema 2.b, remarcăți faptul că aici s-a eliminat termenul liber $x_0 = 1$ de la toți neuronii, precum și funcția de activare (de tip prag) de la neuronul de pe nivelul de ieșire.

Răspuns:

Funcțiile calculate de către cele trei unități neuronale sunt:

$$\begin{aligned} o_1(x_1, x_2) &= \text{sign}(w_{11}x_1 + w_{12}x_2) \\ o_2(x_1, x_2) &= \text{sign}(w_{21}x_1 + w_{22}x_2) \\ o(x_1, x_2) &= w_1 o_1(x_1, x_2) + w_2 o_2(x_1, x_2) \\ &= w_1 \text{sign}(w_{11}x_1 + w_{12}x_2) + w_2 \text{sign}(w_{21}x_1 + w_{22}x_2) \end{aligned}$$

Așadar,

$$o(1, 1) = w_1 \text{sign}(w_{11} + w_{12}) + w_2 \text{sign}(w_{21} + w_{22}) = -1 \quad (146)$$

$$o(1, -1) = w_1 \text{sign}(w_{11} - w_{12}) + w_2 \text{sign}(w_{21} - w_{22}) = 1 \quad (147)$$

$$o(-1, 1) = w_1 \text{sign}(-w_{11} + w_{12}) + w_2 \text{sign}(-w_{21} + w_{22}) = 1 \quad (148)$$

$$o(-1, -1) = w_1 \text{sign}(-w_{11} - w_{12}) + w_2 \text{sign}(-w_{21} - w_{22}) = -1 \quad (149)$$

Se poate arăta ușor că $\text{sign}(x) = -\text{sign}(-x)$ pentru orice $x \in \mathbb{R} \setminus \{0\}$. Așadar, dacă $-w_{11} + w_{12} \neq 0$ (ceea ce este echivalent cu $w_{11} \neq w_{12}$) și $-w_{21} + w_{22} \neq 0$ (echivalent cu $w_{21} \neq w_{22}$), relația (148) implică

$$w_1 \text{sign}(w_{11} - w_{12}) + w_2 \text{sign}(w_{21} - w_{22}) = -1,$$

ceea ce intră în contradicție evidentă cu relația (147) scrisă mai sus.

Similar, dacă $w_{11} \neq -w_{12}$ și $w_{21} \neq -w_{22}$, relația (149) implică

$$w_1 \text{sign}(w_{11} + w_{12}) + w_2 \text{sign}(w_{21} + w_{22}) = 1,$$

ceea ce contravine relației (146).

Sumarizând cele scrise mai sus, dacă ponderile w_{ij} , cu $i, j \in \{1, 2\}$ satisfac condiția multiplă

$$w_{11} \neq w_{12} \text{ și } w_{21} \neq w_{22},$$

sau

$$w_{11} \neq -w_{12} \text{ și } w_{21} \neq -w_{22},$$

răspunsul la problema noastră este: *Imposibil*.

Rămân de analizat următoarele cazuri, derivate din negarea condiției multiple de mai sus:

- *Cazul particular 1:* $w_{11} = w_{12}$ și $w_{11} = -w_{12}$ (de unde rezultă $w_{11} = w_{12} = 0$);
- *Cazul particular 2:* $w_{11} = w_{12}$ și $w_{21} = -w_{22}$;
- *Cazul particular 3:* $w_{21} = w_{22}$ și $w_{11} = -w_{12}$;
- *Cazul particular 4:* $w_{21} = w_{22}$ și $w_{21} = -w_{22}$ (de unde rezultă $w_{21} = w_{22} = 0$).

Cazul 4 este similar cazului 1, iar cazul 3 este similar cazului 2. Se poate arăta relativ ușor (deși nu este chiar imediat) că în fiecare dintre aceste cazuri particolare obținem același răspuns: *Imposibil*.

5.

(Expresivitatea rețelelor neuronale:
funcții booleene)

■ CMU, 2010 fall, Aarti Singh, HW5, pr. 4.3
CMU, 2011 spring, Roni Rosenfeld, HW4, pr. 1.d

Să se arate că orice funcție booleană (cu n argumente / variabile booleene și câte o singură valoare booleană pentru fiecare combinație posibilă de valori pentru cele n argumente), poate fi reprezentată printr-o rețea neuronală cu doar două niveluri, folosind neuroni care au componenta de activare de tip funcție-prag.

Răspuns:

Stim că orice funcție booleană poate fi reprezentată în mod echivalent ca o expresie din logica propozițiilor. La rândul ei, aceasta se poate scrie ca o conjuncție de disjuncții de literalii (sau invers, ca o disjuncție de conjuncții de literalii), un literal fiind fie o variabilă fie negația ei.³⁸¹ Pentru exemplificare, vedeti rezolvarea problemei 3. (Demonstrația dată aici poate fi văzută ca o generalizare naturală a exemplului particular de acolo.) Pentru a prezenta o astfel de expresie cu ajutorul rețelelor de perceptriони-prag vom proceda astfel:

- Fiecare disjuncție de literali va fi reprezentată prin câte un perceptron-prag plasat pe nivelul ascuns. Ponderile pentru acest perceptron vor fi:

1 pentru literalii pozitivi,
 -1 pentru literalii negativi,
 $k - \frac{1}{2}$ pentru termenul liber ($x_0 = 1$), unde k este numărul de literali din disjuncție.

- Pentru conjuncția de disjuncții, vom pune un perceptron-prag pe nivelul exterior și-i vom asigna următoarele ponderi:

1 pentru fiecare conexiune hidden-to-output,
 $-(l - \frac{1}{2})$ pentru termenul liber, unde l este numărul de disjuncții, același cu numărul de perceptriони-prag de pe nivelul ascuns.

³⁸¹Pentru fiecare combinație de valori l_1, \dots, l_n ale variabilelor X_1, \dots, X_n pentru care funcția booleană dată (f) ia valoarea de adevăr T , se va considera conjuncția $L_1 \wedge \dots \wedge L_n$, unde L_i este X_i dacă $l_i = T$, și L_i este $\neg X_i$ dacă $l_i = F$. Expresia booleană care reprezintă funcția f este disjuncția de conjuncții astfel calculate.

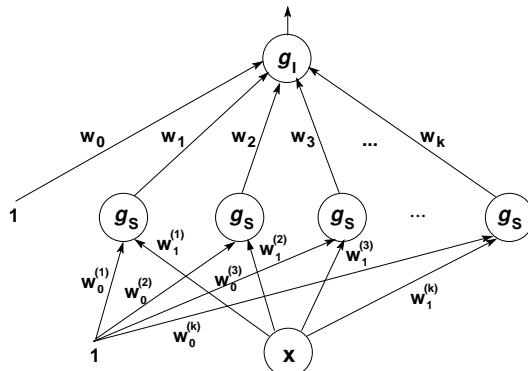
Observații:

1. În mod absolut similar se poate proceda pornind de la rezultatul, cunoscut din logica propozițiilor, că orice expresie booleană se poate scrie ca o conjuncție de disjuncții.
2. Este posibil ca în rețea obținută, numărul de unități de pe nivelul ascuns să fie exponențial în raport cu numărul de variabile de intrare (n). Așadar, în astfel de cazuri, pentru valori mari ale lui n , rezultatul care a fost demonstrat la acest exercițiu, deși are o semnificație teoretică importantă, nu are aplicabilitate practică.

6. (Reprezentarea unor funcții de tip treaptă cu ajutorul unor rețele cu unități liniare sau de tip prag)
CMU, 2007 fall, Carlos Guestrin, HW2, pr. 4

În această problemă presupunem că dispunem (doar) de unități neuronale ale căror funcții de activare pot fi de două tipuri:

- funcția identitate: $g_I(x) = x$, și
- funcția treaptă: $g_S(x) = 1$ dacă $x \geq 0$ și 0 în rest.



De exemplu, rețeaua neuronală din figura alăturată are o intrare $x \in \mathbb{R}$, un singur nivel ascuns pe care sunt k unități având funcția de activare de tip treaptă, iar pe nivelul de ieșire un singur perceptron având funcția de activare de tip identitate.

Ieșirea acestei rețele poate fi descrisă astfel:

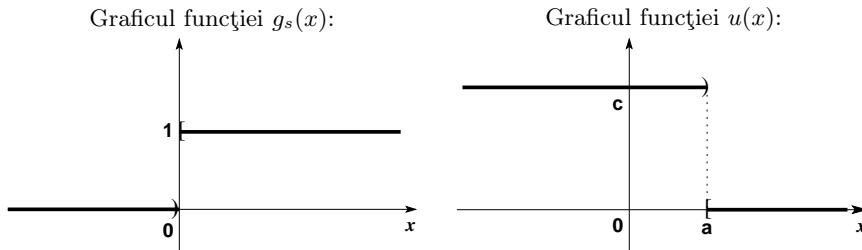
$$out(x) = g_I \left(w_0 + \sum_{i=1}^k w_i g_S \left(w_0^{(i)} + w_1^{(i)} x \right) \right) = w_0 + \sum_{i=1}^k w_i g_S \left(w_0^{(i)} + w_1^{(i)} x \right)$$

a. Considerăm funcția de tip treaptă care este definită prin expresia $u(x) = c$ când $x < a$ și 0 în rest (unde a și c sunt constante reale fixate). Construji o rețea neuronală având un singur nivel ascuns, intrarea x și outputul $u(x)$. Desenați structura rețelei neuronale, indicând funcția de activare pentru fiecare unitate (fie g_I fie g_S), și specificați valorile tuturor ponderilor (ca expresii în funcție de a și c).

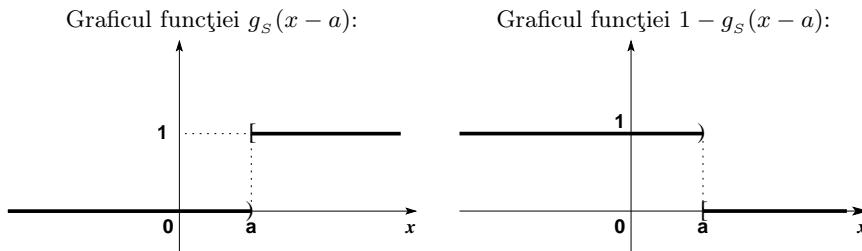
b. Se consideră constantele reale (fixate) a , b și c . Construji o rețea neuronală având o intrare x și un nivel ascuns, al cărei output este c dacă $x \in [a, b]$ și 0 în rest. Desenați structura rețelei neuronale, indicând funcția de activare pentru fiecare unitate (fie g_I fie g_S), și specificați valorile tuturor ponderilor (ca expresii în funcție de a , b și c).

Răspuns:

a. Pentru conveniență, vom reprezenta mai întâi graficele funcțiilor treaptă g_s și u din enunțul problemei.



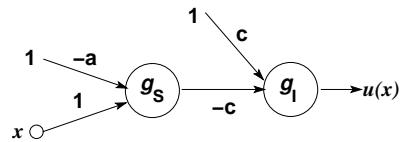
Pornind de la aceste reprezentări grafice, putem să obținem în câteva pași funcția treaptă $u(x)$ ca funcție compusă, exprimată cu ajutorul lui g_s , după cum urmează:



Comparând ultimul grafic de mai sus cu graficul funcției u , rezultă că $u(x) = c(1 - g_s(x - a)) \Rightarrow u(x) = c - c \cdot g_s(x - a)$.

Pe de altă parte, putem observa că o rețea neuronală care are o singură unitate pe nivelul ascuns având funcția de activare de tip treaptă și o singură unitate pe nivelul de ieșire având funcția de activare de tip identitate va avea ieșirea $out(x) = g_I(w_0 + w_1 \cdot g_S(w_0^{(1)} + w_1^{(1)}x))$, adică $out(x) = w_0 + w_1 \cdot g_S(w_0^{(1)} + w_1^{(1)}x)$.

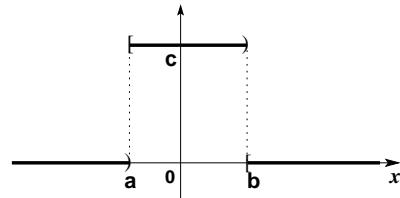
Prin urmare, ponderile rețelei neuronale care reprezintă funcția u vor fi: $w_0 = c$, $w_1 = -c$, $w_0^{(1)} = -a$ și $w_1^{(1)} = 1$, iar rețeaua însăși va arăta ca în figura alăturată.



b. Funcția

$$v(x) = \begin{cases} c & \text{dacă } x \in [a, b] \\ 0 & \text{în rest} \end{cases}$$

este reprezentată grafic în figura alăturată. Se poate observa din acest grafic că funcția v se obține ușor din funcția u de la punctul anterior.



Într-adevăr, dacă notăm

$$u_a(x) = \begin{cases} c & \text{dacă } x < a \\ 0 & \text{în rest,} \end{cases}$$

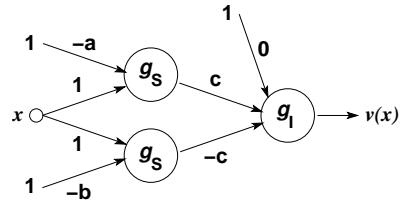
atunci

$$v(x) = u_b(x) - u_a(x) = c(1 - g_s(x - b)) - c(1 - g_s(x - a)).$$

Calculând, obținem $v(x) = c - c \cdot g_S(x - b) - c + c \cdot g_S(x - a)$

$$\Rightarrow v(x) = c \cdot g_S(x - a) - c \cdot g_S(x - b)$$

Așadar, putem construi o rețea neuronală cu două unități pe nivelul ascuns (fiecare având funcția de activare de tip treaptă). Ieșirile acestor două unități vor constitui intrările unei unități [cu funcție de activare] de tip liniar, aflată pe nivelul de ieșire. Setând corespunzător ponderile corespunzătoare acestui nivel, outputul rețelei va fi chiar $v(x)$. Reprezentarea acestei rețele este cea din figura alăturată.



7.

(Orice funcție definită pe un interval mărginit din \mathbb{R} , care este continuă în sens Lipschitz, poate fi aproximată oricără de bine cu ajutorul unei rețele neuronale care are un singur nivel ascuns)

■ CMU, 2011 fall, T. Mitchell, A. Singh, HW5, pr. 2.3

Considerăm o funcție oarecare $f(x)$, al cărei domeniu este de forma $[C, D] \subset \mathbb{R}$. Supunem că f este continuă în sens Lipschitz,³⁸² adică există o constantă $L \geq 0$ astfel încât

$$|f(x') - f(x)| \leq L|x' - x|, \forall x, x' \in [C, D].$$

Folosiți rezultatele obținute la problema 6 pentru a construi o rețea neuronală cu un singur nivel ascuns, care aproximează funcția f cu o eroare de cel mult $\varepsilon > 0$, adică

$$\forall x \in [C, D], |f(x) - out(x)| \leq \varepsilon,$$

unde $out(x)$ desemnează ieșirea acestei rețele neuronale pentru intrarea x .

Va trebui ca rețeaua să folosească doar funcțiile de activare de tip identitate și respectiv prag (notate cu g_I și g_S în problema 6). Indicați

- numărul K de unități ascunse,
- funcția de activare pentru fiecare unitate,
- o formulă pentru calcularea fiecărei ponderi de pe conexiunile input-to-hidden (notate cu $w_0^{(k)}$ și $w_1^{(k)}$, unde $k \in \{1, 2, \dots, K\}$), precum și pentru conexiunile hidden-to-output (și anume w_0, w_k , cu $k \in \{1, 2, \dots, K\}$).

ACESTE PONDERI POT FI SPECIFICATE ÎN FUNCȚIE DE C, D, L SI ε , PRECUM SI ÎN FUNCȚIE DE VALORILE $f(x)$ OBȚINUTE PENTRU UN NUMĂR FINIT DE VALORI x , LA LIBERA ALEGERE. (VA TREBUI SĂ SPECIFICAȚI ÎN MOD EXPLICIT CARE SUNT VALORILE x PE CARE LE VETI FOLOSI.)

Nu vi se cere să scrieți în mod explicit funcția $out(x)$.

Cum justificați faptul că rețeaua concepută de dumneavoastră atinge acuratețea indicată?

Răspuns:

³⁸²Continuitatea în sens Lipschitz este o formă [mai] tare de *uniform-continuitate*: se poate demonstra că orice funcție continuă în sens Lipschitz este uniform continuă (și, deci, continuă).

Este imediat că din ipoteza $|f(x) - f(x')| \leq L|x - x'|$, în cazul în care $|x - x'| \leq \frac{\varepsilon}{L}$, rezultă

$$|f(x) - f(x')| \leq \varepsilon \quad (150)$$

Așadar, este de dorit să „acoperim“ intervalul $[C, D)$ cu intervale de lungime $\frac{2\varepsilon}{L}$ (sau mai mică):

$$[C, C + \frac{2\varepsilon}{L}), [C + \frac{2\varepsilon}{L}, C + 2\frac{2\varepsilon}{L}), \dots, [C + (K-1)\frac{2\varepsilon}{L}, D))$$

și să luăm punctele x' de forma $\xi_i = (\alpha_i + \beta_i)/2$, pentru cu $i = 1, \dots, K$, unde

$$\begin{aligned} K &\stackrel{\text{not.}}{=} \left\lceil (D - C) \frac{L}{2\varepsilon} \right\rceil, \\ \alpha_1 &= C, \beta_1 = \alpha_2 = C + \frac{2\varepsilon}{L}, \dots, \beta_{K-1} = \alpha_K = C + (K-1)\frac{2\varepsilon}{L}, \beta_K = D. \end{aligned}$$

Așadar, prin ξ_i am notat mijlocul intervalului i de mai sus, $[\alpha_i, \beta_i]$.

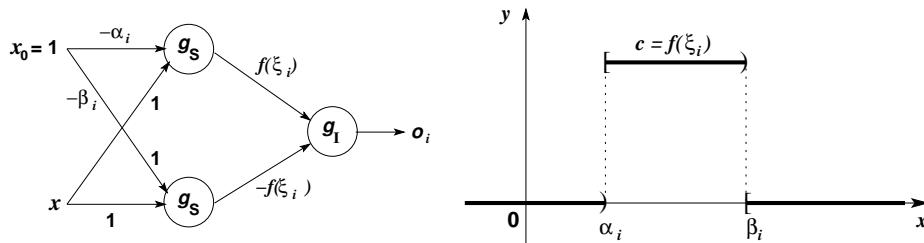
În continuare, vom arăta că putem construi o rețea neuronală cu un singur nivel ascuns și care folosește doar unități de tip liniar sau de tip prag, astfel încât

$$\text{pentru } \forall x \in [C, D), \text{ dacă } x \in [\alpha_i, \beta_i), \text{ atunci } \text{out}(x) \stackrel{\text{def.}}{=} f(\xi_i). \quad (151)$$

În consecință, dacă în relația (150) vom înlocui x' cu ξ_i , va rezulta imediat că $|f(x) - \text{out}(x)| \leq \varepsilon$.

Revenind acum la proprietatea (151), vom arăta că există o rețea având K unități pe nivelul ascuns (și o singură unitate de ieșire), astfel încât, dacă $x \in [\alpha_i, \beta_i)$, atunci unitatea i de pe nivelul ascuns se activează (producând valoarea $f(\xi_i)$, iar toate celelalte unități de pe nivelul ascuns rămân neactive (adică produc output 0). Prin urmare, rezultatul va fi exact cel dorit.

Conform problemei 6 punctul b, există o rețea neuronală care produce valoarea $c = f(\xi_i)$ pentru orice input $x \in [\alpha_i, \beta_i)$ și 0 în rest, și care folosește funcții de activare g_S pe nivelul ascuns și g_I pe nivelul de ieșire:



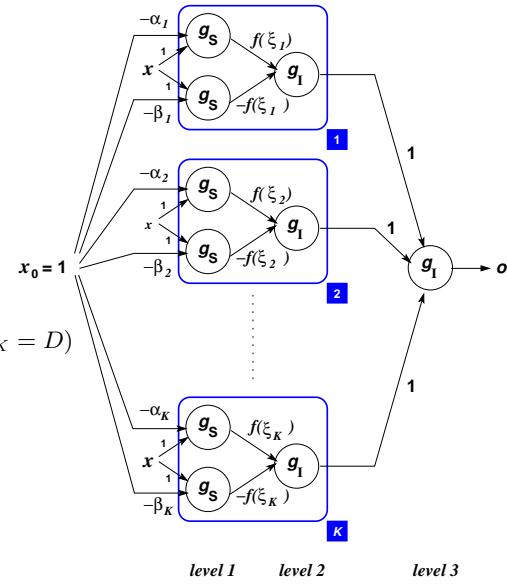
„Ansamblând“ K astfel de rețele, vom obține rețeaua de mai jos (partea dreaptă).

Outputul acestei rețele este exact cel dorit (deci satisfacă condiția din enunț):

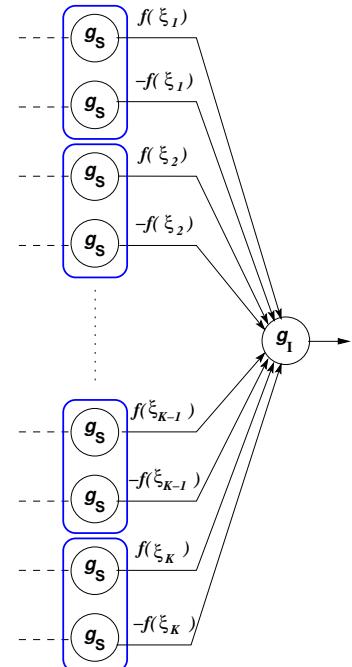
$$out(x) = \begin{cases} f(\xi_1) & \text{pt. } x \in [\alpha_1 = C, \beta_1] \\ f(\xi_2) & \text{pt. } x \in [\alpha_2 = \beta_1, \beta_2] \\ \dots & \dots \\ f(\xi_K) & \text{pt. } x \in [\alpha_K = \beta_{K-1}, \beta_K = D] \end{cases}$$

\Rightarrow

$$|f(x) - out(x)| \leq \varepsilon, \forall x \in [C, D].$$

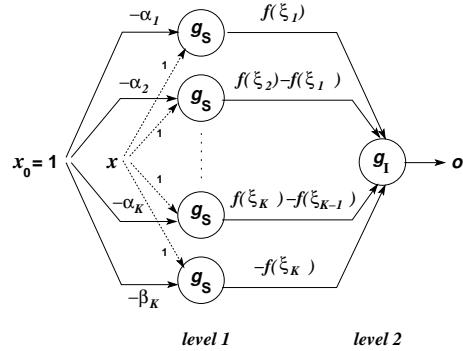


Neajunsul este că rețeaua aceasta are două niveuri ascunse, în loc de unul singur, aşa cum se cere în enunț. Totuși, el poate fi „corectat“ imediat, comasând nivelurile 2 și 3 — formate doar din unități liniare — într-o singură unitate liniară (care va constitui nivelul de ieșire al noii rețele), ca în figura alăturată.³⁸³



³⁸³Vedeți proprietatea enunțată la problema 32.b.

De asemenea, datorită faptului că $\beta_1 = \alpha_2, \beta_2 = \alpha_3, \dots, \beta_{K-1} = \alpha_K$, nivelul ascuns (al acestei noi rețele), format din cele $2K$ unități de tip prag, poate fi echivalent cu un altul, compus din doar $K+1$ unități de tip prag. La final, obținem rețeaua din figura alăturată.

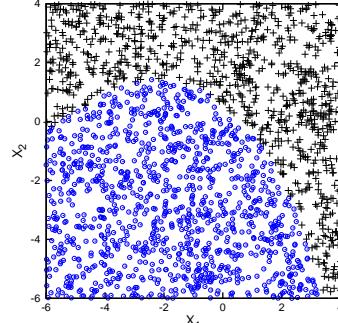
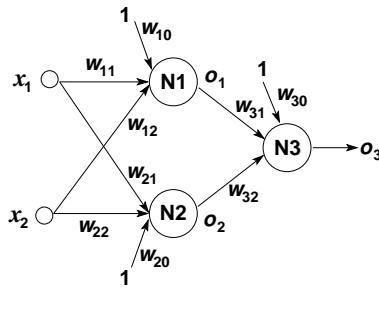


8.

(Rețele neuronale cu unități de tip sigmoidal:
granițe / suprafete de decizie)

*CMU, 2008 fall, Eric Xing, HW2, pr. 2.1
CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW3, pr. 2.1
CMU, 2011 fall, Eric Xing, HW1, pr. 3.2*

Una dintre chestiunile importante pe care trebuie să ni le punem atunci când facem cunoștință cu un nou clasificator este *ce tip de suprafete de decizie* (sau: granițe de decizie; engl., decision boundaries) *poate să învețe respectivul clasificator*.



Fie rețeaua neuronală din figura de mai sus, partea stângă. Cele trei unități din componenta această rețele sunt de tip sigmoidal, adică au funcția de activare $\sigma(x) = \frac{1}{1 + e^{-x}}$. Imaginea din partea dreaptă a fost obținută calculând valorile produse de rețea pentru o serie de puncte generate în mod aleatoriu în dreptunghiul $(-6, 4) \times (-6, 4)$ din planul euclidian, după ce ponderile au fost instantiată cu următoarele valori: $w_{10} = -0.8, w_{11} = 0.8, w_{12} = 0.1, w_{20} = 0.3, w_{21} = 0.3, w_{22} = -0.4, w_{30} = 0.2, w_{31} = 1$ și $w_{32} = -1$. Clasificarea instanțelor a fost desemnată folosind semnele $+$ și respectiv \circ .

- Expremați ieșirile o_1 și o_2 în funcție de intrările x_1 și x_2 și ponderile $w_{10}, w_{11}, w_{12}, w_{20}, w_{21}$ și w_{22} .
- Scrieti *regula de decizie* corespunzătoare acestei rețele neuronale.

c. Presupunem că eliminăm funcția de activare sigmoidală din perceptronii de pe nivelul ascuns (așadar, N1 și N2 devin unități liniare), însă păstrăm aceleași valori pentru ponderi. Scrieți regula de decizie pentru noua rețea.

d. Presupunem că lăsăm valorile ponderilor din rețeaua de la punctul precedent să varieze liber. Poate această rețea să învețe conceptul reprezentat de rețeaua inițială?

Răspuns:

$$\begin{aligned} \text{a. } o_1(x_1, x_2) &= \sigma(w_{10} + w_{11}x_1 + w_{12}x_2) = \frac{1}{1 + \exp(-(w_{10} + w_{11}x_1 + w_{12}x_2))} \\ o_2(x_1, x_2) &= \sigma(w_{20} + w_{21}x_1 + w_{22}x_2) = \frac{1}{1 + \exp(-(w_{20} + w_{21}x_1 + w_{22}x_2))}, \end{aligned}$$

unde simbolul \exp desemnează funcția exponențială cu baza e .

$$\begin{aligned} \text{b. } o(x_1, x_2) &= \sigma(w_{30} + w_{31}o_1(x_1, x_2) + w_{32}o_2(x_1, x_2)) \\ &= \frac{1}{1 + \exp(-(w_{30} + w_{31}o_1(x_1, x_2) + w_{32}o_2(x_1, x_2)))}. \end{aligned}$$

Regula de decizie corespunzătoare rețelei neuronale din enunț este:

if $o(x_1, x_2) \geq 1/2$ then assign the instance (x_1, x_2) the label +;
else assign it the label -;

sau, echivalent:

if $w_{30} + w_{31}o_1(x_1, x_2) + w_{32}o_2(x_1, x_2) \geq 0$
then assign the instance (x_1, x_2) the label +;
else, assign it the label - .

Este de remarcat faptul că granița de decizie a rețelei neuronale, și anume curba de ecuație $w_{30} + w_{31}o_1(x_1, x_2) + w_{32}o_2(x_1, x_2) = 0$, este una *neliniară*, după cum se poate observa și din figura din enunț (partea dreaptă).

$$\begin{aligned} \text{c. } o'_1(x_1, x_2) &= w_{10} + w_{11}x_1 + w_{12}x_2 \\ o'_2(x_1, x_2) &= w_{20} + w_{21}x_1 + w_{22}x_2 \\ o'(x_1, x_2) &= \sigma(w_{30} + w_{31}o'_1(x_1, x_2) + w_{32}o'_2(x_1, x_2)) \\ &= \sigma(w_{30} + w_{31}(w_{10} + w_{11}x_1 + w_{12}x_2) + w_{32}(w_{20} + w_{21}x_1 + w_{22}x_2)) \\ &= \sigma(w_{30} + w_{31}w_{10} + w_{32}w_{20} + (w_{31}w_{11} + w_{32}w_{21})x_1 + \\ &\quad (w_{31}w_{12} + w_{32}w_{22})x_2). \end{aligned}$$

Regula de decizie este:

if $w_{30} + w_{31}w_{10} + w_{32}w_{20} + (w_{31}w_{11} + w_{32}w_{21})x_1 + (w_{31}w_{12} + w_{32}w_{22})x_2 \geq 0$
then assign the instance (x_1, x_2) the label +;
else, assign it the label - .

d. Se observă că la punctul *c* granița (suprafața) de decizie este o dreaptă. Așadar, noua rețea nu poate învăța în mod convenabil conceptul reprezentat de rețeaua inițială.

Observație: Este util de remarcat faptul că o rețea exact de același tip, având însă alte valori pentru ponderile w_{ij} , poate reprezenta funcția XOR.³⁸⁴

³⁸⁴Vedeti CMU, 2008 fall, Eric Xing, midterm, pr. 4.2. Ar fi [un exercițiu] instructiv să vizualizați cum arată zonele de decizie în acel caz.

9.

(Unitatea sigmoidală: exemplificare)

■ CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 6.a

Considerăm o unitate neuronală sigmoidală care are intrările x_1, x_2 și x_3 . Așadar, ieșirea ei este de forma $y = \sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$ unde $\sigma(z) = \frac{1}{1 + e^{-z}}$. Valorile fiecărei dintre aceste trei intrări sunt 0 sau 1.

Asignați valori ponderilor w_0, w_1, w_2 și w_3 astfel încât outputul unității neuronale să fie strict mai mare decât 0.5 dacă și numai dacă valoarea logică a expresiei $(x_1 \wedge x_2) \vee x_3$ este *adevărat*.

Răspuns:

Vom scrie mai întâi tabela de valori a funcției / expresiei booleene date. Vom adăuga în această tabelă o coloană care va conține forma particulară a sumei ponderate $w_0 + w_1x_1 + w_2x_2 + w_3x_3$ —adică, *argumentul* pe care-l va lua funcția sigmoidală (σ) — pentru fiecare combinație de valori ale variabilelor x_1, x_2, x_3 . În plus, la fiecare linie vom preciza și *condiția* care trebuie să fie satisfăcută de către suma ponderată de pe linia respectivă, în aşa fel încât perceptronul să realizeze codificarea cerută în enunț.³⁸⁵

x_1	x_2	x_3	$(x_1 \wedge x_2) \vee x_3$	$w_0 + w_1x_1 + w_2x_2 + w_3x_3$	$\vdots 0$
0	0	0	0	w_0	< 0
0	0	1	1	$w_0 + w_3$	> 0
0	1	0	0	$w_0 + w_2$	< 0
0	1	1	1	$w_0 + w_2 + w_3$	> 0
1	0	0	0	$w_0 + w_1$	< 0
1	0	1	1	$w_0 + w_1 + w_3$	> 0
1	1	0	1	$w_0 + w_1 + w_2$	> 0
1	1	1	1	$w_0 + w_1 + w_2 + w_3$	> 0

Așadar, rezumând, pentru ca unitatea sigmoidală să codifice expresia booleană dată, trebuie ca ponderile w_i să satisfacă sistemul de inecuații scrise în ultima coloană a tabelului de mai sus.

În vederea găsirii unei soluții pentru acest sistem, putem observa că în expresia $(x_1 \wedge x_2) \vee x_3$ variabilele logice x_1 și x_2 joacă rol simetric în raport cu operatorul \wedge . Deci este natural să considerăm că ponderile alocate lui x_1 și x_2 [ca inputuri] în perceptronul sigmoidal pot fi egale. Introducând deci restricția $w_1 = w_2$, sistemul (152) va deveni:

$$\left\{ \begin{array}{l} w_0 < 0 \\ w_0 + w_3 > 0 \\ w_0 + w_1 < 0 \\ w_0 + w_1 + w_3 > 0 \\ w_0 + 2w_1 > 0 \\ w_0 + 2w_1 + w_3 > 0 \end{array} \right. \quad (153)$$

Mai departe, analizând din nou expresia $(x_1 \wedge x_2) \vee x_3$, este natural să gândim că, în raport cu operatorul \vee , ponderea variabilei x_3 ar trebui să fie aceeași cu ponderile „cumulate“

³⁸⁵Stim că la clasificare cu unitatea sigmoidală se folosește echivalența $\sigma(z) > 1/2 \Leftrightarrow z > 0$ și, complementar, $\sigma(z) < 1/2 \Leftrightarrow z < 0$.

ale variabilelor x_1 și x_2 . Prin urmare, „injectând“ în sistemul de inecuații (153) restricția $w_3 = w_1 + w_2 = 2w_1$, el va deveni:

$$\begin{cases} w_0 < 0 \\ w_0 + 2w_1 > 0 \\ w_0 + w_1 < 0 \\ w_0 + 3w_1 > 0 \\ w_0 + 4w_1 > 0 \end{cases} \quad (154)$$

Din forma acestui sistem, apare natural să explorăm ce anume se întâmplă dacă restricționăm spațiul de soluții impunând condiția $w_1 > 0$. În această ipoteză, sistemul (154) va avea aceeași soluție ca și inecuația dublă

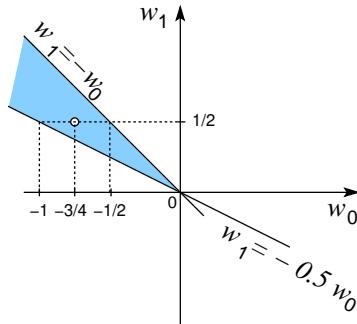
$$w_0 + w_1 < 0 < w_0 + 2w_1,$$

care este echivalentă cu

$$w_1 < -w_0 < 2w_1. \quad (155)$$

Pentru această ultimă inecuație dublă este foarte ușor să indicăm soluții. Iată una dintre ele: $w_1 = \frac{1}{2}$ și $w_0 = -\frac{3}{4}$. Prin urmare, $w_2 = \frac{1}{2}$ și $w_3 = 1$.

Observație (1): De fapt, dacă lucrăm într-un reper de coordonate w_0Ow_1 , orice punct (w_0, w_1) din cadrul al doilea, și care este situat între dreptele de ecuații $w_1 = -w_0$ și respectiv $w_1 = -\frac{1}{2}w_0$ este o soluție a inecuației duble (155), deci și a sistemului de inecuații (152), la care, aşa cum am văzut, au fost adăugate restricțiile $w_1 = w_2 = \frac{1}{2}w_3$. (Invers, adică a preciza care este întreg spațiul de soluții al sistemului (152) este mult în afara obiectivului acestui exercițiu.)



Observație (2): Rostul acestui [tip de] exercițiu este să arate că pentru a găsi valori corespunzătoare ponderilor unui perceptron în așa fel încât el să reprezinte / codifice o anumită funcție, putem apela la [rezolvarea de] sisteme de restricții / inecuații. Mai general, aşa cum precizează și Tom Mitchell în cartea *Machine Learning* la pag. 95, putem folosi metode de programare liniară sau neliniară. Pe de altă parte, acest exercițiu pune în evidență în mod indirect faptul că ar fi de dorit ca (alternativ) să dispunem de o procedură de calcul generală, cât mai simplă, prin care să atingem obiectivul menționat anterior. Modul în care, pentru cazul perceptronului liniar, se fundamentează din punct de vedere teoretic o astfel de procedură — bazată pe metoda gradientului descendente — va face obiectul problemei propuse 34, iar aplicarea ei va fi exemplificată la problemele 10 și 11. În raport cu programarea liniară sau cea neliniară, această procedură are avantajul că se scalează în mod elegant / convenabil la nivel de rețea neuronală.

6.1.2 Unități neuronale — algoritmi de antrenare

10. (Unitatea liniară;
deducerea regului de actualizare a ponderilor (în manieră “batch”),
în cazul particular al învățării unei anumite funcții booleane)

Liviu Ciortuz, 2011

Vom considera din nou funcția booleană $A \wedge \neg B$ (vedeți problema 2.a), dar nu vom mai folosi perceptronul-prag ci unitatea liniară (perceptronul liniar). Ne propunem aici să deducem forma specifică a *regulilor* de actualizare a ponderilor acestui perceptron, cu *obiectivul* de a „învăța“ ulterior valorile concrete ale acestor ponderi, astfel încât perceptronul rezultat să reprezinte funcția booleană $A \wedge \neg B$ (așa cum se va vedea la problema 11). Considerând că antrenarea se va face în regim de lucru “batch” (folosind, desigur, ca instanțe de antrenament, liniile din tabela de adevăr a acestei funcții booleene, cu codificarea 1 pentru *true*, și -1 pentru *false*), vă cerem să particularizați forma pe care o ia regula generală de actualizare a ponderilor ($\bar{w} \leftarrow \bar{w} + \Delta\bar{w}$) la învățarea acestei funcției booleene, procedând în două moduri diferite:

- a. calculând efectiv vectorul $\Delta\bar{w}$ (sau, pe componente, Δw_j) folosind formula dată la curs:

$$\Delta\bar{w} = \eta \sum_{i=1}^4 (t_i - o_i) \bar{x}_i$$

unde \bar{x}_i sunt instanțele din tabela de valori ale funcției $A \wedge \neg B$, $o_i = \bar{w} \cdot \bar{x}_i$ desemnează valoarea calculată de perceptron pentru $i = 1, \dots, 4$, iar t_i este valoarea target a funcției booleene date, pentru intrarea \bar{x}_i , cu $i = 1, \dots, 4$,³⁸⁶

- b. calculând mai întâi expresia funcției $E(\bar{w})$ care desemnează *semisuma pătratelor eroilor* corespunzătoare instanțelor \bar{x}_i ($i = 1, \dots, 4$):

$$E(\bar{w}) = \frac{1}{2} \sum_{i=1}^4 (t_i - o_i)^2 = \frac{1}{2} \sum_{i=1}^4 (t_i - \bar{w} \cdot \bar{x}_i)^2 = \dots,$$

precum și cele trei derivate parțiale din expresia vectorului gradient

$$\nabla E(\bar{w}) = \left(\frac{\partial E(\bar{w})}{\partial w_0}, \frac{\partial E(\bar{w})}{\partial w_1}, \frac{\partial E(\bar{w})}{\partial w_2} \right),$$

folosind expresia funcției $E(\bar{w})$ care a fost calculată anterior, pentru ca la final, în spiritul *metodei gradientului descendente*, să obținem $\Delta\bar{w} = -\eta \nabla E(\bar{w})$.

Atenție! Dacă la final nu ați obținut același rezultat la punctele *a* și *b*, înseamnă că ați greșit la calcule.

Răspuns:

- a. Știm că $o_i = \bar{w} \cdot \bar{x}_i$ și $\Delta\bar{w} = \eta \sum_{i=1}^4 (t_i - o_i) \bar{x}_i$. În cazul nostru, $o_i = (w_0, w_1, w_2) \cdot (1, x_{i,1}, x_{i,2})$, deci $o_1 = w_0 - w_1 - w_2$, $o_2 = w_0 - w_1 + w_2$, $o_3 = w_0 + w_1 - w_2$, $o_4 =$

³⁸⁶Atenție! Aceste instanțe trebuie extinse cu componenta 1 corespunzătoare termenului liber din expresia separatorului liniar.

$w_0 + w_1 + w_2$ și, ținând cont că $t_1 = t_2 = t_4 = -1$ și $t_3 = 1$, rezultă (în scriere matriceală):

$$\Delta \bar{w} = \eta \left[(-1 - o_1) \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + (-1 - o_2) \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} + (1 - o_3) \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} + (-1 - o_4) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right].$$

Scriind această egalitate pe componente, vom avea mai întâi:

$$\begin{aligned} \Delta w_0 &= \eta(-1 - o_1 - 1 - o_2 + 1 - o_3 - 1 - o_4) = \eta(-2 - (o_1 + o_2 + o_3 + o_4)) \\ &= \eta(-2 - (w_0 - w_1 - w_2 + w_0 - w_1 + w_2 + w_0 + w_1 - w_2 + w_0 + w_1 + w_2)) \\ &= \eta(-2 - 4w_0) = -2\eta(2w_0 + 1) \end{aligned}$$

Așadar, regula de actualizare a ponderii w_0 va fi $w_0 \leftarrow w_0 - 2\eta(2w_0 + 1)$.

În mod similar, facând calculele, vom obține $w_1 \leftarrow w_1 - 2\eta(2w_1 - 1)$ și $w_2 \leftarrow w_2 - 2\eta(2w_2 + 1)$.

b. Conform definiției, $E(w_0, w_1, w_2) = \frac{1}{2} \sum_i (t_i - o_i)^2$, cu $o_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$. Prin urmare,

$$\begin{aligned} E(w_0, w_1, w_2) &= \frac{1}{2} [(-1 - (w_0 - w_1 - w_2))^2 + (-1 - (w_0 - w_1 + w_2))^2 + \\ &\quad (1 - (w_0 + w_1 - w_2))^2 + (-1 - (w_0 + w_1 + w_2))^2] \\ &= \frac{1}{2} [(1 + w_0 - w_1 - w_2)^2 + (1 + w_0 - w_1 + w_2)^2 + \\ &\quad (1 - w_0 - w_1 + w_2)^2 + (1 + w_0 + w_1 + w_2)^2] \\ &= \frac{1}{2} [2(1 + w_0 - w_1)^2 + 2w_2^2 + 2(1 + w_2)^2 + 2(w_0 + w_1)^2] \\ &= (1 + w_0 - w_1)^2 + w_2^2 + (1 + w_2)^2 + (w_0 + w_1)^2 \\ &= 2(1 + w_0^2 + w_1^2 + w_2^2 + w_0 - w_1 + w_2) \end{aligned}$$

Derivatele parțiale ale lui E în raport cu variabilele w_0 , w_1 și w_2 sunt:

$$\frac{\partial E}{\partial w_0} = 2(2w_0 + 1), \quad \frac{\partial E}{\partial w_1} = 2(2w_1 - 1), \quad \frac{\partial E}{\partial w_2} = 2(2w_2 + 1)$$

În concluzie, regulile de actualizare pentru ponderile unității liniare vor fi:

$$w_0 \leftarrow w_0 - 2\eta(2w_0 + 1), \quad w_1 \leftarrow w_1 - 2\eta(2w_1 - 1), \quad w_2 \leftarrow w_2 - 2\eta(2w_2 + 1)$$

Se observă că rezultatele obținute la punctele a și b coincid, după cum era de așteptat.

11.

(Perceptronul-prag;
aplicarea algoritmului de actualizare a ponderilor
în manieră incrementală / stochastică,
pentru a învăța o anumită funcție booleană)

Liviu Ciortuz, 2011

La problema 2.a am arătat că funcția booleană $A \wedge (\neg B)$ poate fi reprezentată cu ajutorul unui perceptron-prag — adică, un perceptron cu funcție de activare de tip prag — ale cărui ponderi pe intrări sunt date de ecuația dreptei $y = x - 1 \Leftrightarrow x - y - 1 = 0 \Leftrightarrow w_0 = -1, w_1 = 1, w_2 = -1$. (A se vedea linia punctată din graficul de mai jos.) Evident, există o infinitate de drepte care separă în mod corect cele patru instanțe de antrenament.

În exercițiul de față, spre deosebire de cum am procedat la problema 2.a, nu vom mai „ghici”, ci vom învăța un separator liniar care să reprezinte / codifice aceeași funcție booleană de mai sus. Învățarea se va face aplicând algoritmul de antrenare a perceptronului (adică de actualizare a ponderilor) din carte Machine Learning a profesorului T. Mitchell, pag. 93.³⁸⁷

Convenție: Vom considera că valorile logice Adevărat / Fals sunt codificate de numerele $+1/-1$. De asemenea, variabila numerică x (rescrisă uneori, pentru conveniență, ca x_1) corespunde variabilei A , iar y (rescrisă ca x_2) corespunde variabilei B .

Vom considera că valorile inițiale ale ponderilor perceptronului sunt cele corespunzătoare dreptei $y = \frac{1}{3}x - \frac{1}{3} \Leftrightarrow \frac{1}{3}x - y - \frac{1}{3} = 0 \Leftrightarrow w_0 = -\frac{1}{3}, w_1 = \frac{1}{3}, w_2 = -1$. Se poate constata imediat că funcția $\frac{1}{3}x - y - \frac{1}{3}$ clasifică în mod eronat punctul $(-1, -1)$. Prin urmare, separatorul liniar reprezentat de ecuația $\frac{1}{3}x - y - \frac{1}{3} = 0$ — vedeti dreapta continuă din graficul de mai jos — nu este consistent cu conceptul logic $A \wedge \neg B$.

- a. Folosind rata de învățare $\eta = 0.1$, să se aplice atât-tea iterării câte sunt necesare pentru actualizarea ponderilor perceptronul-prag — în manieră incrementală / stochastică — astfel încât, la final, toate instanțele din tabloul funcției booleene $A \wedge \neg B$ (vedeti tabelul alăturat) să fie clasificate corect. Scrieți ecuația separatorului liniar rezultat.

A	B	$A \wedge \neg B$
-1	-1	-1
-1	+1	-1
+1	-1	+1
+1	+1	-1

Vă reamintim că regula de actualizare a ponderilor pentru perceptronul-prag, în varianta stochastică / incrementală este de formă:

$$\bar{w} \leftarrow \bar{w} + \Delta \bar{w}, \text{ cu } \Delta \bar{w} = \eta(t - o)\bar{x}$$

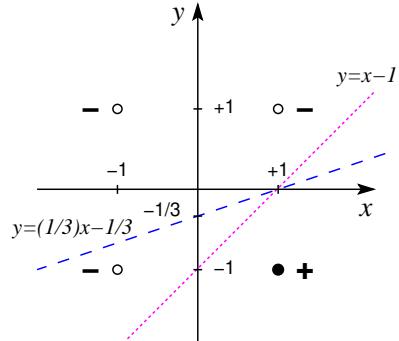
unde $\bar{x} = (1, x_1, x_2)$ desemnează o instanță de antrenament oarecare, o este valoarea produsă de către perceptron pentru intrarea \bar{x} , iar t este valoarea țintă (engl., target) pentru aceeași intrare \bar{x} .

- b. Ce s-ar fi întâmplat dacă am fi folosit o rată de învățare mult mai mică, de exemplu $\eta = 0.01$?

Răspuns:

³⁸⁷Atenție: La problema 10 s-a lucrat cu același concept — funcția booleană $A \wedge (\neg B)$ — însă acolo am folosit perceptronul liniar în varianta “batch”. Aici folosim perceptronul-prag în varianta incrementală / stochastică. Din punct de vedere analitic, forma regulii de actualizare este $\bar{w} \leftarrow \bar{w} + \Delta \bar{w}$, cu $\Delta w_i = -\eta \sum_n (t_n - o_n) \frac{\partial E}{\partial w_i} \cdot x_{n,i}$ pentru prima variantă și, respectiv, $\Delta w_i = -\eta (t_n - o_n) \frac{\partial E}{\partial w_i} \cdot x_{n,i}$ pentru a doua variantă. Rețineți că outputurile (o_n) produse de cei doi perceptriuni pentru un același input x_n diferă în general, întrucât perceptronul-prag aplică funcția de activare *sign* la combinația liniară $\bar{w} \cdot \bar{x}$, în vreme ce perceptronul liniar nu o aplică.

Reprezentarea grafică a datelor de antrenament și a poziției inițiale a separatorului liniar ($\frac{1}{3}x - y - \frac{1}{3} = 0$) asociat perceptronului din enunț este cea din figura alăturată. Acest separator liniar greșește la clasificarea instanței logice $A = F, B = F$ (repräsentată numeric de către $x = -1$ și $y = -1$), fiindcă $\frac{1}{3}(-1) - (-1) - \frac{1}{3} = 1 - \frac{2}{3} = \frac{1}{3} > 0$. În schimb, toate celelalte instanțe de antrenament sunt clasificate corect (verificarea este imediată).



În vederea aplicării regulii de actualizare a ponderilor, vom calcula mai întâi $\Delta\bar{w} = \eta(t_1 - o_1) \cdot \bar{x}_1 = \eta(-1 - 1) \cdot \bar{x}_1$, unde $\bar{x}_1 = (1, -1, -1)$ este punctul incorrect clasificat. Așadar,

$$\Delta\bar{w} = -2\eta(1, -1, -1) = -0.2 \cdot (1, -1, -1) = (-0.2, 0.2, 0.2)$$

sau, rescris pe componente, $\Delta w_0 = -0.2$, $\Delta w_1 = 0.2$, $\Delta w_2 = 0.2$.³⁸⁸ Aplicând regula de actualizare a ponderilor, după efectuarea calculelor vom obține:

$$w_0 = -\frac{1}{3} - 0.2 = -\frac{8}{15}, \quad w_1 = \frac{1}{3} + 0.2 = \frac{8}{15}, \quad w_2 = -1 + 0.2 = -\frac{4}{5}$$

În consecință, perceptronul învață o nouă poziție pentru separatorul liniar:

$$\frac{8}{15}x - \frac{4}{5}y - \frac{8}{15} = 0 \Leftrightarrow \frac{2}{3}x - y - \frac{2}{3} = 0 \Leftrightarrow y = \frac{2}{3}x - \frac{2}{3}$$

³⁸⁸Dacă în locul perceptronului-prag am fi lucrat cu perceptronul liniar în varianta stochastică, aici am fi obținut

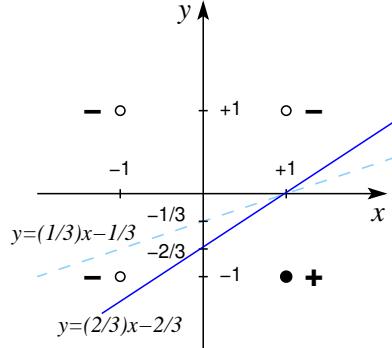
$$\begin{aligned} \Delta\bar{w} &= \eta(-1 - o(-1, -1))\bar{x}_1 = \eta(-1 - (-\frac{1}{3} + 1 - \frac{1}{3}))\bar{x}_1 = \eta(-1 - \frac{1}{3})\bar{x}_1 = -\eta\frac{4}{3}\bar{x}_1 = -\frac{1}{10}\frac{4}{3}\bar{x}_1 \\ &= -\frac{2}{15}(1, -1, -1) = \left(-\frac{2}{15}, \frac{2}{15}, \frac{2}{15}\right). \end{aligned}$$

După actualizare, aceasta ar fi dus la valorile

$$w_0 = -\frac{1}{3} - \frac{2}{15} = -\frac{7}{15}, \quad w_1 = \frac{1}{3} + \frac{2}{15} = \frac{7}{15}, \quad w_2 = -1 + \frac{2}{15} = -\frac{13}{15}.$$

Dreapta determinată de aceste ponderi are ecuația $\frac{17}{15}x_1 - \frac{13}{15}x_2 - \frac{7}{15} = 0 \Leftrightarrow x_2 = \frac{7}{13}x_1 - \frac{7}{13} = 0$. Pentru instanța \bar{x}_1 , expresia $\frac{17}{15}x_1 - \frac{13}{15}x_2 - \frac{7}{15}$ are valoarea $-\frac{17}{15} + \frac{13}{15} - \frac{7}{15} = -\frac{1}{15} < 0$.

Valoarea funcției $\frac{2}{3}x - y - \frac{2}{3}$ pentru $x = -1$ și $y = -1$ este $-\frac{2}{3} + 1 - \frac{2}{3} = -\frac{1}{3} < 0$. Așadar, această instanță de antrenament este acum corect clasificată. Mai mult, se verifică imediat că toate instanțele de antrenament sunt clasificate corect de către noul separator liniar (vedeți graficul alăturat), deci algoritmul de actualizare a ponderilor perceptronului-prag se oprește după ce s-a efectuat o singură iterație.



Observație:

Aplicarea variantei "batch" pentru algoritmul de învățare a ponderilor perceptronului-prag (bazată pe regulile deduse prin folosirea metodei gradientului descendente pentru perceptronul liniar) nu ar fi diferit de modul de lucru de mai sus (iar soluția obținută ar fi fost aceeași), fiindcă $t_n = o_n$ pentru orice $n \neq 1$. Doar scrierea / redactarea calculelor ar fi diferit ușor.

b. Dacă s-ar fi lucrat folosind o valoare mai mică pentru rata de învățare η , este foarte posibil să fi fost necesar să efectuăm mai multe iterări. Cele patru instanțe de antrenament fiind separabile liniar, convergența algoritmului (la valoarea minimă, adică 0, pentru eroarea totală) este asigurată pentru valori mici ale lui η . (Vedeți T. Mitchell, *Machine Learning*, pag. 89.)

12.

(Deducerea regulii de antrenare pentru un tip particular de unitate liniară)
CMU, 1995 fall, Tom Mitchell, HW4, pr. 5

Considerăm un nou tip de unitate neuronală liniară care are două intrări x_1 și x_2 și o ieșire definită astfel:

$$\text{output} = w_0 + w_1 w_0 x_1 + w_2 w_0 x_2.$$

Derivați regula de actualizare a ponderilor cu metoda gradientului descendente, folosind următoarea definiție pentru *funcția de cost / pierdere* (engl., loss function):

$$E = \frac{1}{2} \sum_{d \in D} [\text{target}_d - \text{output}_d]^2,$$

unde D este setul de instanțe de antrenament.

Răspuns:

Conform metodei gradientului descendente, modificarea ponderilor se va face după formula $w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$. Calculăm deci derivata funcției de eroare:

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 = \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) = \sum_{d \in D} -(t_d - o_d) \frac{\partial o_d}{\partial w_i} \end{aligned}$$

$$\begin{aligned}
&= - \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (w_0 + w_1 w_0 x_{1,d} + w_2 w_0 x_{2,d}) \\
&= \begin{cases} - \sum_{d \in D} (t_d - o_d) (1 + w_1 x_{1,d} + w_2 x_{2,d}) & \text{dacă } i = 0 \\ - \sum_{d \in D} (t_d - o_d) w_0 x_{1,d} & \text{dacă } i = 1 \\ - \sum_{d \in D} (t_d - o_d) w_0 x_{2,d} & \text{dacă } i = 2. \end{cases}
\end{aligned}$$

Așadar, actualizarea pondererilor o vom face folosind formulele:

$$\begin{aligned}
w_0 &\leftarrow w_0 + \eta \sum_{d \in D} (t_d - o_d) (1 + w_1 x_{1,d} + w_2 x_{2,d}) \\
w_1 &\leftarrow w_1 + \eta \sum_{d \in D} (t_d - o_d) w_0 x_{1,d} \\
w_2 &\leftarrow w_2 + \eta \sum_{d \in D} (t_d - o_d) w_0 x_{2,d}
\end{aligned}$$

unde η este o constantă pozitivă (rata de învățare).

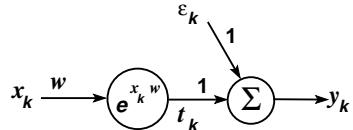
13.

(Aplicarea metodei gradientului descendente
pentru modelarea unui dataset particular, cu „zgomot“ de tip gaussian)
*prelucrare de Liviu Ciortuz, după
CMU, 2001 fall, Andrew Moore, final exam, pr. 10.2*

Presupunem că avem la dispoziție un set de date, în care fiecare instanță are un atribut de intrare $x \in \mathbb{R}$ și un atribut de ieșire y . Bănuim că relația dintre valorile celor două attribute este de forma

$$y_k = e^{wx_k} + \varepsilon_k,$$

unde w este o constantă reală, iar $\varepsilon_k \in \mathbb{R}$ este un „zgomot“, care este independent de x_k . Putem să ne imaginăm y_k ca fiind produs de o rețea neuronală [având pe nivelul ascuns un perceptron de un tip particular], care are o singură pondere nespecificată w , ca în figura alăturată. Se va considera că $t_k \stackrel{\text{not.}}{=} e^{wx_k}$, este target-ul de învățat.



a. Considerând că instanțele x_1, \dots, x_n (fixate) au fost generate în mod independent și că „zgomotul“ ε urmează o distribuție de tip gaussian cu media 0 și varianță σ^2 (pentru orice $k \in \{1, \dots, n\}$), demonstrați că valoarea lui w pentru care se maximizează verosimilitatea condițională a datelor de antrenament, adică

$$\operatorname{argmax}_w P(y_1, \dots, y_n | x_1, \dots, x_n, w) \stackrel{\text{indep.}}{=} \operatorname{argmax}_w \prod_{k=1}^n P(y_k | x_k, w),$$

este exact aceeași cu valoarea (lui w) pentru care se minimizează suma pătratelor erorilor:

$$\operatorname{argmin}_w \sum_{k=1}^n (e^{wx_k} - y_k)^2.$$

Acet rezultat oferă o explicație suplimentară, probabilistă pentru utilizarea criteriului sumei pătratelor erorilor — a cărui justificare ar rămâne altminteri doar intuitivă — drept criteriu de optimizat la aplicarea metodei gradientului descendente pentru perceptri și rețele neuronale.

Indicație: Puteti face demonstrația nu doar pentru cazul de față, ci mai general, și anume pentru toate situațiile în care [ca target] vrem să învățăm o funcție oarecare f (în cazul nostru e^{wx}), datele de antrenament sunt de forma (x_k, y_k) , iar diferența dintre y_k și $f(x_k)$ (în cazul nostru ε_k) umează o distribuție gaussiană de medie 0.

b. Aplicați metoda gradientului descendente — adică, scrieți regula de actualizare a ponderii w — pentru a afla valoarea ponderii w care minimizează suma pătratelor erorilor.³⁸⁹

Răspuns:

a. Considerând că deviația standard a distribuției următoare de ε_k este σ și ținând cont că

$$P(\varepsilon_k) \stackrel{\text{def.}}{=} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\varepsilon_k - \mu)^2}{2\sigma^2}}, \text{ cu } \mu = 0, \text{ vom avea:}$$

$$\begin{aligned} \operatorname{argmax}_w \prod_{k=1}^n P(y_k | x_k, w) &= \operatorname{argmax}_w \prod_{k=1}^n P(y_k - f(x_k) | x_k, w) \\ &= \operatorname{argmax}_w \prod_{k=1}^n P(\varepsilon_k | x_k, w) = \operatorname{argmax}_w \prod_{k=1}^n P(\varepsilon_k | w) \\ &= \operatorname{argmax}_w \prod_{k=1}^n P(y_k - t_k | w) = \operatorname{argmax}_w \prod_{k=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_k - t_k)^2}{2\sigma^2}} \\ &= \operatorname{argmax}_w \ln \prod_{k=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_k - t_k)^2}{2\sigma^2}} = \operatorname{argmax}_w \sum_{k=1}^n \ln \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_k - t_k)^2}{2\sigma^2}} \\ &= \operatorname{argmax}_w \sum_{k=1}^n \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{(y_k - t_k)^2}{2\sigma^2} \right) = \operatorname{argmax}_w \sum_{k=1}^n \left(-\frac{(y_k - t_k)^2}{2\sigma^2} \right) \\ &= \operatorname{argmax}_w \left(-\sum_{k=1}^n \frac{(y_k - t_k)^2}{2\sigma^2} \right) = \operatorname{argmin}_w \sum_{k=1}^n \left(\frac{(y_k - t_k)^2}{2\sigma^2} \right) \\ &= \operatorname{argmin}_w \sum_{k=1}^n (y_k - t_k)^2. \end{aligned}$$

b. Ca și la exercițiile precedente, în locul sumei pătratelor erorilor vom considera drept criteriu de optimizat pentru metoda gradientului descendente semisuma pătratelor erorilor, $E(w) = \frac{1}{2} \sum_k (y_k - t_k)^2$, unde $t_k = e^{wx_k}$, întrucât aceasta este ușor mai convenabilă la calcule și nu modifică rezultatul problemei. (A se vedea echivalența $(cf)'(x) = 0 \Leftrightarrow cf'(x) = 0$, unde c este constantă iar f este funcție derivabilă.)

³⁸⁹În enunțul original, în locul *sumei* este specificată *media* (engl., mean) *pătratelor erorilor*. Este vorba de medie în sensul de medie aritmetică. Cele două formulări sunt echivalente din punctul de vedere al optimizării (fiindcă diferă doar printr-un factor pozitiv), deci ambele sunt corecte. Am preferat să folosim termenul *sumă*, întrucât la punctul *a* se lucrează probabilist, deci în contextul respectiv termenul *medie* (ambiguu în românește!) ar avea [și] o altă semnificație.

Conform metodei gradientului descendente, regula de actualizare a ponderii w va fi (în varianta “batch”) de forma

$$w \leftarrow w - \eta \frac{\partial E(w)}{\partial w}.$$

unde $\eta > 0$ este rata de învățare.

Calculăm derivata lui E în raport cu w :

$$\begin{aligned} \frac{\partial E(w)}{\partial w} &= \frac{\partial}{\partial w} \frac{1}{2} \sum_k (y_k - t_k)^2 = -\frac{1}{2} \sum_k 2(y_k - t_k) \frac{\partial t_k}{\partial w} \\ &= -\sum_k (y_k - t_k) \frac{\partial}{\partial w} (e^{wx_k}) = -\sum_k (y_k - t_k) e^{wx_k} x_k = \sum_k (e^{wx_k} k - y_k) e^{wx_k} x_k. \end{aligned}$$

Așadar, regula de actualizare a ponderii w va fi:

$$w \leftarrow w + \eta \sum_k (y_k - o_k) e^{wx_k} x_k.$$

14.

(O variantă a regulii de actualizare a ponderilor

pentru perceptronul liniar,
obținută prin aplicarea metodei gradientului descendente

pe o altă funcție de pierdere / cost (engl., loss function)

în locul sumei pătratelor erorilor:

funcția log-sigmoidală)

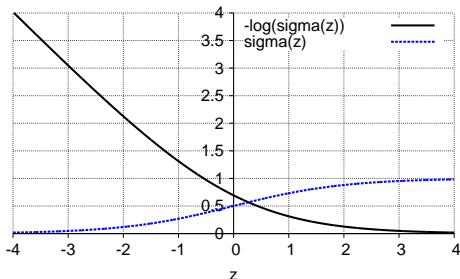
prelucrare de Liviu Ciortuz, după

■ University of Utah, 2008 fall, Hal Daumé III, HW4, pr. 2-4

În acest exercițiu, vom construi o nouă variantă de unitate liniară (perceptron fără prag) care va optimiza în locul semisumei pătratelor erorilor, o *funcție de cost* — a se vedea *explicația suplimentară* dată mai jos — de tip log-sigmoidal:³⁹⁰

$$l(y, h(x)) = -\ln \sigma(yh(x)),$$

unde $\sigma(z) = 1/(1 + \exp(-z))$.



Convenție: Vom considera că o ipoteză oarecare h produce *valori reale* și că valorile pozitive desemnează clasa $+1$, iar cele negative clasa -1 .

a. Verificați că funcția log-sigmoidală de mai sus este într-adevăr o *funcție de cost*, adică: atunci când clasa desemnată de ipoteza $h(x)$ este corectă ($y = h(x)$), valoarea $l(y, h(x))$ este mică și, invers, atunci când clasa desemnată de $h(x)$ este incorectă ($y \neq h(x)$), valoarea $l(y, h(x))$ este mare.

b. Folosind o unitate liniară, dorim să învățăm o funcție liniară minimizând în locul semisumei pătratelor erorilor — cum am procedat la problemele 10, 11, 12 și 13 —, o funcție de cost de tip log-sigmoidal, și anume $f(b, \bar{w}) = \sum_n -\ln \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))$, unde

³⁹⁰La capitolul *Metode de regresie* am numit această funcție *funcția de pierdere logistică*.

n este indicele folosit pentru indexarea instanțelor de antrenament $(\bar{x}_1, y_1), (\bar{x}_2, y_2), \dots$. Calculați gradientul acestei funcții în raport cu \bar{w} și în raport cu b .

c. Arătați că funcția log-sigmoidală de la punctul precedent este convexă atât în \bar{w} cât și în b . Așadar, ulterior veți [putea] elabora algoritmul gradientului descendente pentru această variantă de perceptron.

Explicație suplimentară:

În clasificarea automată, noțiunea de *funcție de cost* sau *funcție de pierdere* (engl., loss function) are rolul de a evalua calitatea unei ipoteze oarecare h produse de un algoritm de clasificare automată. Cele mai des folosite tipuri de funcții de pierdere sunt:

- costul mediu (cauzat de clasificările eronate)
- rata erorii (engl., missclassification error rate),
- log-verosimilitatea (engl., log-likelihood).

Un exemplu clasic de funcție de pierdere de tip cost mediu³⁹¹ este suma pătratelor erorilor.

În cele ce urmează vom arăta cum se poate defini un alt gen de funcție de pierdere de tip cost mediu, aplicând pentru instanțele care au fost clasificate greșit *costuri diferențiate*. (Funcția log-sigmoidală folosită în acest exercițiu este tocmai o astfel de funcție.) Facem observația că aici ne vom limita la clasificarea binară.

Fie c_0 costul clasificării eronate a unei instanțe negative și c_1 costul clasificării eronate a unei instanțe pozitive. Considerăm următoarea *funcție-indicator*, definită ca o funcție de tip treaptă 0/1, prin expresia

$$I_g(y, x) = \begin{cases} 0 & \text{dacă } y = g(x), \\ 1 & \text{altfel.} \end{cases}$$

Costul mediu al clasificării făcute de o ipoteză h se poate calcula cu ajutorul acestei funcții-indicator astfel:³⁹²

$$\begin{aligned} E_x E_{y|x} [c_0 I_g(y_x, x) h(x) + c_1 I_g(y_x, x) (1 - h(x))] \text{ sau} \\ E_x [c_0 P(y_x = 0 | x) h(x) + c_1 P(y_x = 1 | x) (1 - h(x))] \end{aligned}$$

unde y_x este eticheta asignată instanței x , iar g este conceptul (funcția) de învățat. Evident, este de dorit ca un algoritm de învățare automată să producă o ipoteză cu un cost minim. Așadar, chestiunea pe care urmează să ne-o punem este cum anume putem calcula o ipoteză de cost minim.

Funcția-indicator fiind o funcție de tip treaptă, rezultă că folosirea metodei gradientului pentru aflarea minimului erorii medii definite mai sus nu este posibilă. Din această cauză se apeleză de obicei la funcții cu proprietăți matematice mai bune, aşa cum este funcția log-sigmoidală.³⁹³

³⁹¹Dacă se face înmulțirea nu cu 1/2 cum am procedat la problema 20, ci cu 1/ n , unde n este numărul de exemple. Constanta aceasta nu influențează procesul / calculul de optimizare.

³⁹²Cf. *The Handbook of Data Mining*, Nong Ye (ed.), 2003, Lawrence Erlbaum Assoc. inc.

³⁹³Pentru clarificare, facem următoarea precizare: La problemele precedente, funcția σ se aplică la calcularea outputului perceptronului sigmoidal: $o_n = \sigma(\bar{w} \cdot \bar{x}_n)$. Mai departe, o_n contribuia la calcularea funcției de eroare $E(\bar{w}) = 1/2 \sum_n (t_n - o_n)^2$ folosită la aplicarea metodei gradientului descendente. În schimb, la prezenta problemă funcția σ se aplică ieșirii produse de perceptronul liniar, sub forma $-\ln \sigma(t_n o_n)$, ceea ce corespunde termenului $(t_n - o_n)^2$ din $E(\bar{w})$.

Răspuns:

a. Funcția l se poate scrie sub forma:

$$l(y, h(x)) = -\ln \sigma(y h(x)) = -\ln \frac{1}{1 + e^{-y h(x)}} = \ln(1 + e^{-y h(x)}).$$

Din proprietățile logaritmului știm că $\ln 1 = 0$, $\ln a < 0$ dacă $0 < a < 1$, și $\ln a > 0$ dacă $a > 1$.

Dacă ipoteza $h(x)$ este corectă, adică are același semn cu y , atunci $e^{-y h(x)}$ este o valoare mică (și cu atât mai mică cu cât $|h(x)|$ este mai mare), deci și valoarea funcției $l(y, h(x))$ este mică.

Dacă ipoteza $h(x)$ este incorectă, adică are semn contrar semnului lui y , atunci $e^{-y h(x)}$ este o valoare mare (și cu atât mai mare cu cât $|h(x)|$ este mai mare), deci și valoarea funcției $l(y, h(x))$ este mare.

De exemplu, dacă $y = 1$ și $h(x) = -1$ (sau $y = -1$ și $h(x) = 1$) atunci pierdere / costul este $-\ln \frac{1}{1+e} \approx 1.31$, iar dacă $y = h(x) = 1$ (sau, ambele, -1) atunci costul este $-\ln \frac{1}{1+e^{-1}} = -\ln \frac{e}{1+e} \approx 0.31$.

b. Calculăm derivatele parțiale ale funcției $f(b, \bar{w}) = \sum_n -\ln \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))$ în raport cu w_i și respectiv b . Vom folosi următoarele proprietăți:

- derivata sumei de funcții derivabile este suma derivelor, adică $(f + g)' = f' + g'$;
- derivata unei compunerii de funcții derivabile este $(f \circ g)' = (f' \circ g)g'$;
- derivata logaritmului este $\ln'(x) = \frac{1}{x}$;
- derivata funcției sigmoidale este $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

$$\begin{aligned} \frac{\partial f}{\partial w_i} &= \frac{\partial}{\partial w_i} \sum_n -\ln \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) = -\sum_n \frac{\partial}{\partial w_i} \ln \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) \\ &= -\sum_n \frac{1}{\sigma(y_n(\bar{w} \cdot \bar{x}_n + b))} \frac{\partial}{\partial w_i} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) \\ &= -\sum_n \frac{1}{\sigma(y_n(\bar{w} \cdot \bar{x}_n + b))} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))(1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n x_{n,i} \\ &= -\sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n x_{n,i}. \end{aligned}$$

Similar, vom obține

$$\frac{\partial f}{\partial b} = -\sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n.$$

c. Funcția $f(b, \bar{w})$ este convexă în raport cu variabila w_i dacă $\frac{\partial^2 f}{\partial w_i^2} \geq 0$. Similar, $f(b, \bar{w})$ este convexă în raport cu b dacă $\frac{\partial^2 f}{\partial b^2} \geq 0$. Așadar, trebuie să calculăm aceste derivate parțiale de ordinul 2. Pentru aceasta, vom folosi derivatele parțiale de ordinul întâi care au fost calculate la punctul precedent.

$$\begin{aligned}
\frac{\partial^2 f}{\partial w_i^2} &= \frac{\partial}{\partial w_i} \left(\frac{\partial f}{\partial w_i} \right) = \frac{\partial}{\partial w_i} \left(- \sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n x_{n,i} \right) \\
&= - \sum_n \frac{\partial}{\partial w_i} ((1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n x_{n,i}) \\
&= - \sum_n y_n x_{n,i} (0 - \frac{\partial}{\partial w_i} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) \\
&= \sum_n y_n x_{n,i} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) \frac{\partial}{\partial w_i} (y_n(\bar{w} \cdot \bar{x}_n + b)) \\
&= \sum_n y_n x_{n,i} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n x_{n,i} \\
&= \sum_n \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n^2 x_{n,i}^2
\end{aligned}$$

Cum $\sigma(x) \in (0, 1)$ pentru orice $x \in \mathbb{R}$, rezultă că și $1 - \sigma(x) \in (0, 1) \forall x$. Așadar, $\sigma(x) > 0$, $1 - \sigma(x) > 0$ și, evident, $y_n^2 \geq 0$, deci $\frac{\partial^2 f}{\partial b^2} \geq 0$, adică f este convexă în w_i , pentru orice i .

Analog, calculăm derivata parțială de ordin secund a lui f în raport cu b :

$$\begin{aligned}
\frac{\partial^2 f}{\partial b^2} &= \frac{\partial}{\partial b} \left(\frac{\partial f}{\partial b} \right) = \frac{\partial}{\partial b} \left(- \sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n \right) \\
&= - \sum_n \frac{\partial}{\partial b} ((1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n) = - \sum_n y_n (0 - \frac{\partial}{\partial b} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) \\
&= \sum_n y_n \frac{\partial}{\partial b} \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) \\
&= \sum_n y_n \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) \frac{\partial}{\partial b} (y_n(\bar{w} \cdot \bar{x}_n + b)) \\
&= \sum_n y_n \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n \\
&= \sum_n \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)) (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n^2 \geq 0
\end{aligned}$$

Așadar, f este convexă și în raport cu variabila b .

Observație: La curs, b a fost asimilat cu w_0 , iar $x_{n,0} = 1$ prin definiție. Dacă s-ar fi procedat la fel și aici, atunci nu am fi avut de calculat decât o derivată la punctul a și una la punctul b .

În *concluzie*, valoarea optimă a funcției f este un minim, iar algoritmul de învățare automată bazat pe metoda gradientului descendente va găsi (la limită, eventual) acest minim. Antrenarea acestui perceptron se face similar cu antrenarea unității liniare care a fost prezentată la curs, folosind reguli de actualizare de forma:

$$w_i \leftarrow w_i + \Delta w_i \text{ și } b \leftarrow w_i + \Delta b.$$

În cazul de față, Δw_i și Δb sunt definite de expresiile

$$\Delta w_i = -\eta \frac{\partial f}{\partial w_i} = \eta \sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b))) y_n \bar{x}_{n,i}$$

și respectiv

$$\Delta b = -\eta \frac{\partial f}{\partial b} = \eta \sum_n (1 - \sigma(y_n(\bar{w} \cdot \bar{x}_n + b)))y_n.$$

15.

(Unități neuronale sigmoidale
[și rețele neuronale de unități sigmoidale]:
deducerea expresiei funcției de eroare de tip cross-entropie;
două variante: cu și, respectiv, fără „zgomot“)
*prelucrare de Liviu Ciortuz, după
■ CMU, 2011 fall, Eric Xing, HW1, pr. 3.3*

Stim că pentru a antrena o unitate neuronală — și vom vedea mai târziu că este la fel și în cazul rețelelor neuronale — avem nevoie să definim o funcție de eroare / pierdere / cost, care poate fi minimizată cu ajutorul algoritmului de „învățare“ a ponderilor unității / rețelei. Până acum,³⁹⁴ am folosit [ca funcție de eroare] suma pătratelor erorilor. Se poate arăta că obiectivul de a minimiza această funcție este implicat de principiul verosimilității maxime (engl., MLE), în contextul „procesării“ unui „semnal“ însotit de un „zgomot“ care urmează o distribuție de tip Gaussian.³⁹⁵ În acest exercițiu vom arăta că pentru antrenarea unităților sigmoidale există încă un tip de funcție de cost care poate fi interpretată [probabilist] în sensul maximizării verosimilității.

Fie datele de antrenament $D = (X, t) = \{(x_i, t_i)\}, i = 1, 2, \dots, N$. De exemplu, x_i (element dintr-un spațiu \mathbb{R}^d) poate fi imaginea unei fețe, în vreme ce t_i este o etichetă binară care are valoarea 0 dacă fața respectivă este a unui bărbat, respectiv 1 în cazul unei femei.

Vom considera o unitate neuronală de tip sigmoidal.³⁹⁶ Notând cu y valoarea reală produsă de această unitate, rezultă că $y \in (0, 1)$. Este natural să interpretăm acest output ca fiind probabilitatea ca eticheta booleană t să fie 1. Formal, putem scrie $y \stackrel{\text{def.}}{=} P(t = 1 | x; w)$. În consecință, este natural să căutăm valori convenabile pentru ponderile w folosind principiul verosimilității maxime (MLE), sub forma următoare:

$$w_{MLE} = \underset{w}{\operatorname{argmax}} \prod_{i=1}^N P(t_i | x_i; w).$$

a. Arătați că a maximiza expresia $\prod_{i=1}^N P(t_i | x_i; w)$ revine la a minimiza expresia următoare, despre care putem spune, după o observare atentă, că este o cross-entropie:³⁹⁷

$$E(w) = - \sum_{i=1}^N [t_i \ln y_i + (1 - t_i) \ln(1 - y_i)],$$

unde y_i este outputul produs de rețeaua neuronală pentru exemplul x_i .

³⁹⁴Adică: la toate problemele de până acum, cu excepția problemei 14.

³⁹⁵Pentru a vedea cum se face deducerea expresiei funcției de eroare ca sumă a pătratelor erorilor într-un context particular (însă sugestiv), cititorul poate consulta problema 13.

³⁹⁶Mai general, se poate considera o rețea formată din unități sigmoidale. (De fapt, aşa a fost formulată problema originală de la CMU.) Într-un astfel de caz este însă suficient să presupunem că tipul unității / unităților de pe nivelul de ieșire din rețea este cel sigmoidal.

³⁹⁷A se vedea problema 58 de la capitolul de *Fundamente*.

- b. Să presupunem că este posibil ca etichetele datelor de antrenament să fi fost puse eronat, și anume cu probabilitatea ε . Considerând că datele de antrenament sunt distribuite în mod identic și independent unele de altele, scrieți expresia funcției de eroare / cost $E^*(w, \varepsilon)$ care corespunde log-verosimilității cu semn schimbă (engl., the negative log likelihood).
- c. Verificați că funcția de eroare $E(w)$ se obține din $E^*(w, \varepsilon)$ pentru cazul particular $\varepsilon = 0$.
- d. Explicați de ce anume funcția de eroare $E^*(w, \varepsilon)$ va face ca modelul obținut să fie [mai] robust în raport cu date incorect etichetate, spre deosebire de funcția uzuală $E(w)$.

Răspuns:

- a. Conform enunțului, $y_i \stackrel{def.}{=} P(t_i = 1|x_i; w)$. Așadar, putem scrie

$$P(t_i|x_i; w) = \begin{cases} y_i & \text{dacă } t_i = 1, \\ 1 - y_i & \text{dacă } t_i = 0. \end{cases}$$

Se verifică imediat că, din punct de vedere matematic, relația de mai sus se poate exprima în mod echivalent sub forma

$$P(t_i|x_i; w) = (y_i)^{t_i} (1 - y_i)^{1-t_i}.$$

Această expresie, în ciuda faptului că este mai puțin intuitivă, este mult mai convenabilă pentru calculele pe care trebuie să le facem în vederea aplicării principiului verosimilității maxime:³⁹⁸

$$\begin{aligned} w_{MLE} &\stackrel{def.}{=} \underset{w}{\operatorname{argmax}} P(t_1, \dots, t_n | x_1, \dots, x_n; w) \stackrel{i.i.d.}{=} \underset{w}{\operatorname{argmax}} \prod_{i=1}^N P(t_i | x_i; w) \\ &= \underset{w}{\operatorname{argmax}} \sum_{i=1}^N \ln P(t_i | x_i; w) = \underset{w}{\operatorname{argmax}} \sum_{i=1}^N \ln(y_i)^{t_i} (1 - y_i)^{1-t_i} \\ &= \underset{w}{\operatorname{argmax}} \sum_{i=1}^N (t_i \ln y_i + (1 - t_i) \ln(1 - y_i)) = \underset{w}{\operatorname{argmax}} (-E(w)) \\ &= \underset{w}{\operatorname{argmin}} E(w). \end{aligned}$$

Pentru penultima egalitate de mai sus, vedeți formula de definiție a funcției E din enunț.

- b. În cazul în care etichetarea datelor de antrenament s-a făcut în mod eronat, cu probabilitatea ε , rezultă imediat că

$$x_i \text{ a fost etichetat cu } \begin{cases} 1 & \text{cu probabilitatea } 1 - \varepsilon, \text{ dacă } t_i = 1, \\ 0 & \text{cu probabilitatea } \varepsilon, \text{ dacă } t_i = 1, \\ 0 & \text{cu probabilitatea } 1 - \varepsilon, \text{ dacă } t_i = 0, \\ 1 & \text{cu probabilitatea } \varepsilon, \text{ dacă } t_i = 0. \end{cases}$$

Prin urmare, putem scrie în manieră condensată

$$P(t_i|x_i; w) = \begin{cases} y_i(1 - \varepsilon) + (1 - y_i)\varepsilon & \text{dacă } t_i = 1, \\ (1 - y_i)(1 - \varepsilon) + y_i\varepsilon & \text{dacă } t_i = 0. \end{cases}$$

³⁹⁸De fapt, în cazul de față vom lucra cu verosimilitate condițională.

Ba chiar și mai mult, această expresie este echivalentă cu următoarea:

$$P(t_i|x_i; w) = [y_i(1 - \varepsilon) + (1 - y_i)\varepsilon]^{t_i} [(1 - y_i)(1 - \varepsilon) + y_i\varepsilon]^{1-t_i}.$$

Făcând un calcul similar cu cel de la punctul a , va rezulta:³⁹⁹

$$E^*(w, \varepsilon) = - \sum_{i=1}^N \{t_i \ln[(y_i(1 - \varepsilon) + (1 - y_i)\varepsilon)] + (1 - t_i) \ln[(1 - y_i)(1 - \varepsilon) + y_i\varepsilon]\}.$$

c. Înlocuind parametrul ε cu valoarea 0 în expresia pe care tocmai am obținut-o mai sus, vom avea:

$$\begin{aligned} E^*(w, 0) &= - \sum_{i=1}^N \{[t_i \ln y_i + (1 - y_i) \times 0] + (1 - t_i) \ln[(1 - y_i) + y_i \times 0]\} \\ &= - \sum_{i=1}^N [t_i \ln y_i + (1 - t_i) \ln(1 - y_i)] = E(w) \end{aligned}$$

d. Vom considera ca *exemplu* cazul extrem (însă instructiv) în care toate etichetele au fost puse incorrect. Dacă am folosi funcția de eroare $E(w)$, atunci modelul produs de algoritmul de retro-propagare ar fi complet eronat. Dacă în schimb vom folosi funcția de eroare $E^*(w, \varepsilon)$ cu $\varepsilon = 1$, atunci algoritmul de retro-propagare va încerca să învețe un model care clasifică datele exact invers față de etichetele originale, ceea ce corespunde obiectivului nostru.

Notă: Problema 46 cere să se adapteze algoritmul de retro-propagare folosit pentru antrenarea rețelelor neuronale de tip “feed-forward” (vedeți pr. 20) pentru cazul în care funcția de optimizat este de tip cross-entropie, să cum a fost definită în acest exercițiu.

16.

(Algoritmul de antrenare a perceptronului-prag,
versiunea Rosenblatt: aplicare)

*prelucrare de Liviu Ciortuz, după
CMU, 2013 fall, W. Cohen, E. Xing, Sample questions, pr. 2
CMU, 2013 fall, A. Smola, G. Gordon, midterm ex. practice, pr. 10
CMU, 2013 spring, A. Smola, B. Poczos, HW2, pr. 2*

Se consideră o secvență de exemple $\bar{x}_i \in \mathbb{R}^d$, pentru $i = 1, \dots, n$, împreună cu etichetele corespunzătoare $y_i \in \{-1, 1\}$. Vom antrena perceptronul-prag folosind următorul algoritm, cunoscut în literatura de specialitate sub numele de *perceptronul Rosenblatt*:

```
initialize  $\bar{w} \leftarrow \bar{0}$ 
for  $i = 1, \dots, n$  do
    if  $y_i (\bar{w} \cdot \bar{x}_i) \leq 0$  then
         $\bar{w} \leftarrow \bar{w} + y_i \bar{x}_i$ 
    end if
end for
```

³⁹⁹Se verifică imediat că $[y_i(1 - \varepsilon) + (1 - y_i)\varepsilon] + [(1 - y_i)(1 - \varepsilon) + y_i\varepsilon] = 1$.

unde operatorul \cdot reprezintă produsul scalar, vectorii $\bar{x}_i \in \mathbb{R}^d$ desemnează instanțele de antrenament, iar etichetele asignate lor sunt notate (ca de obicei) cu y_i .

Observație: În pseudo-codul folosit mai sus se consideră că termenul liber w_0 fie este absent fie este inclus în \bar{w} și, în acest ultim caz, prima componentă a vectorului \bar{x}_i (notată cu $x_{0,i}$) este prin convenție 1.

a. Calculați actualizările vectorului de parametri \bar{w} folosind algoritmul de mai sus pe următorul set de exemple (considerând că nu se folosește termenul liber w_0):

$$\begin{aligned}\bar{x}_1 &= (0, 0, 0, 1, 0, 0, 1), \quad y_1 = 1 \\ \bar{x}_2 &= (1, 1, 0, 0, 0, 1, 0), \quad y_2 = -1 \\ \bar{x}_3 &= (0, 0, 1, 1, 0, 0, 0), \quad y_3 = 1 \\ \bar{x}_4 &= (1, 0, 0, 0, 1, 1, 0), \quad y_4 = -1 \\ \bar{x}_5 &= (1, 0, 0, 0, 0, 1, 0), \quad y_5 = -1\end{aligned}$$

b. Verificați dacă separatorul obținut la finalul execuției algoritmului va clasifica perfect toate datele de antrenament.

c. Care sunt diferențele ce caracterizează acest algoritm în raport cu algoritmul prezentat în cartea *Machine Learning* a lui Tom Mitchell (pag. 88-94), care este bazat pe aşa-numita *regulă delta*?⁴⁰⁰ Vă puteți referi de exemplu la modul de lucru “batch” vs. modul incremental / stochastic, inițializarea ponderilor, rata de învățare, și criteriul de oprire.

Răspuns:

a. Conform algoritmului dat, avem:

$$\begin{aligned}i = 1 : \quad \bar{w} &= \bar{0} \stackrel{\text{not.}}{=} (0, 0, 0, 0, 0, 0, 0) \Rightarrow y_1(\bar{w} \cdot \bar{x}_1) = 0 \leq 0 \\ &\Rightarrow \bar{w} \leftarrow y_1 \bar{x}_1 = \bar{x}_1 = (0, 0, 0, 1, 0, 0, 1) \\ i = 2 : \quad y_2(\bar{w} \cdot \bar{x}_2) &= -(0, 0, 0, 1, 0, 0, 1) \cdot (1, 1, 0, 0, 0, 1, 0) = 0 \leq 0 \\ &\Rightarrow \bar{w} \leftarrow \bar{w} + y_2 \bar{x}_2 = \bar{x}_1 - \bar{x}_2 = (-1, -1, 0, 1, 0, -1, 1) \\ i = 3 : \quad y_3(\bar{w} \cdot \bar{x}_3) &= (-1, -1, 0, 1, 0, -1, 1) \cdot (0, 0, 1, 1, 0, 0, 0) = 1 > 0 \\ i = 4 : \quad y_4(\bar{w} \cdot \bar{x}_4) &= -(-1, -1, 0, 1, 0, -1, 1) \cdot (1, 0, 0, 0, 1, 1, 0) = 2 > 0 \\ i = 5 : \quad y_5(\bar{w} \cdot \bar{x}_5) &= -(-1, -1, 0, 1, 0, -1, 1) \cdot (1, 0, 0, 0, 0, 1, 0) = 2 > 0.\end{aligned}$$

b. Știm deja de la punctul precedent că $y_i(\bar{w} \cdot \bar{x}_i) > 0$ pentru $i \in \{3, 4, 5\}$. Așadar, va trebui să mai verificăm doar dacă are loc această relație și pentru $i \in \{1, 2\}$:

$$\begin{aligned}i = 1 : \quad y_1(\bar{w} \cdot \bar{x}_1) &= (-1, -1, 0, 1, 0, -1, 1) \cdot (0, 0, 0, 1, 0, 0, 1) = 2 > 0 \\ i = 2 : \quad y_2(\bar{w} \cdot \bar{x}_2) &= -(-1, -1, 0, 1, 0, -1, 1) \cdot (1, 1, 0, 0, 0, 1, 0) = 3 > 0.\end{aligned}$$

În concluzie, separatorul $\bar{w} = (-1, -1, 0, 1, 0, -1, 1)$ clasifică perfect datele de antrenament.

c. Este imediat că perceptronul Rosenblatt corespunde modului de lucru incremental: el face actualizarea ponderilor după fiecare instanță de antrenament \bar{x}_i pentru care $\text{sign}(\bar{w} \cdot \bar{x}_i)$, i.e., clasificarea produsă de actualul \bar{w} , nu coincide cu eticheta y_i .

⁴⁰⁰Vedeți formula (4.10) din cartea citată, pag. 93.

Initializarea ponderilor se face în acest algoritm cu 0. În algoritmul clasic (cel din cartea lui Tom Mitchell), initializarea ponderilor w_j , pentru $j = 1, \dots, d$, se face cu numere mici (în modul), alese în mod aleatoriu.

Forma regulii de actualizare a ponderilor perceptronului Rosenblatt nu coincide cu *regula delta*:

$$\bar{w} \leftarrow \bar{w} + \eta(y_i - o_i)\bar{x}_i \quad (156)$$

unde $\eta > 0$ este rata de învățare. Legătura dintre cele două forme se poate face în felul următor:

Este evident că regula (156) are efect doar în cazul în care $y_i \neq o_i$, ceea ce echivalează cu $y_i(\bar{w} \cdot \bar{x}_i) < 0$. Perceptronul-prag folosește funcția de activare *sign*, aşadar faptul că y_i este diferit de o_i implică fie $y_i = 1$ și $o_i = -1$, fie $y_i = -1$ și $o_i = 1$. În ambele situații rezultă $y_i - o_i = 2y_i$. Prin urmare, folosind pre-condiția $y_i(\bar{w} \cdot \bar{x}_i) < 0$, regula (156) devine

$$\bar{w} \leftarrow \bar{w} + 2\eta y_i x_i$$

Luând $\eta = 1/2$, obținem forma regulii din algoritmul Rosenblatt.

Algoritmul Rosenblatt face o singură trecere prin setul de instanțe $\{x_i\}_i$. Evident, se poate extinde în mod natural acest algoritm pentru a proceda ca în cazul clasic, permitând trecerea de mai multe ori prin setul de instanțe, eventual până la convergență, adică până când în cursul unei același iterări nu se mai fac deloc ajustări ale ponderilor w (presupunând că că setul de instanțe de antrenament este liniar separabil).

17.

(Perceptronul Rosenblatt, câteva proprietăți simple:
atât numărul de greșeli cât și poziția finală a separatorului
depind de ordinea de furnizare a exemplelor)

■ CMU, 2015 spring, T. Mitchell, N. Balcan, HW6, pr. 3.ab
CMU, 2017 spring, Barnabas Poczos, HW3, pr. 1.bc

Considerăm că rulăm algoritmul *Perceptron* al lui Rosenblatt,⁴⁰¹ folosind un anumit sir de exemple S . (Vă reamintim că un *exemplu* este o *instanță* — punct din \mathbb{R}^d — împreună cu *eticheta* sa). Fie S' același set de exemple ca și S , însă prezentate într-o altă ordine.

- a. Face oare algoritmul *Perceptron* același număr de greșeli pe sirul / succesiunea de exemple S ca și pe sirul S' ? Dacă da, de ce? Dacă nu, indicați două astfel de siruri (S și S') pe care algoritmul *Perceptron* face un număr diferit de greșeli.
- b. Din punct de vedere geometric, algoritmul *Perceptron* clasifică instanțele date conform poziției lor relativ la un hiperplan de separare. Vectorul de ponderi (w) învățat de către *Perceptron* va fi normal [adică, perpendicular] pe acest hiperplan? Răspundeti prin *Adevărat* sau *Fals* și justificați în mod riguros.

Răspuns:

- a. Considerăm următorul set de exemple (S), în ordinea indicată:

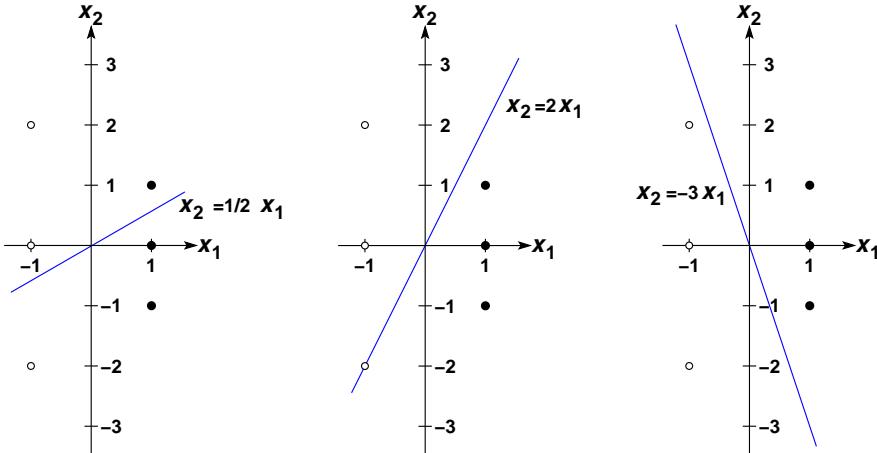
exemplul	1	2	3	4	5	6
instanța (x_1, x_2)	(−1, 2)	(1, 0)	(1, 1)	(−1, 0)	(−1, −2)	(1, −1)
eticheta y	−1	+1	+1	−1	−1	+1

⁴⁰¹Pentru pseudo-cod, vedeti enunțul problemei 16.

Sintetizăm execuția algoritmului *Perceptron* pe aceste date, fără a folosi termenul “bias” (w_0), alcătuind tabelul următor:

iterația(i)	\bar{x}_i	y_i	$y_i \bar{w} \cdot \bar{x}_i$	\bar{w}	greșală	ec. separatorului
initializare	—	—	(0, 0)	—	—	—
1	(-1, 2)	-1	$0 \leq 0$	(1, -2)	da	$x_2 = 0.5x_1$
2	(1, 0)	+1	$1 > 0$	(1, -2)	nu	$x_2 = 0.5x_1$
3	(1, 1)	+1	$-1 \leq 0$	(2, -1)	da	$x_2 = 2x_1$
4	(-1, 0)	-1	$2 > 0$	(2, -1)	nu	$x_2 = 2x_1$
5	(-1, -2)	-1	$0 \leq 0$	(3, 1)	da	$x_2 = -3x_1$
6	(1, -1)	+1	$2 > 0$	(3, 1)	nu	$x_2 = -3x_1$

Așadar, algoritmul a comis trei greșeli. Cei trei separatori obținuți în cursul aplicării algoritmului *Perceptron* pe sirul de exemple S sunt prezentați în figura următoare.



Este imediat că modul în care se construiește separatorul — a cărui expresie analitică este o combinație liniară de instanțe greșit catalogate — este dependent de exemplele prezentate. Întrebarea care a fost pusă în enunț este dacă pentru două succesiuni / ordini diferite, rezultatele finale (i.e., expresiile separatorilor obținuți) coincid (întotdeauna) sau nu. În continuare vom arăta că în mod obișnuit rezultatele finale (i.e., pozițiile finale ale separatorilor obținuți) nu coincid.

Considerăm S' secvența de exemple obținută din S prin inversarea [ordinii] exemplelor \bar{x}_1 și \bar{x}_2 . Este imediat că la prima iterație perceptronul greșește și apoi calculează $\bar{w} = (1, 0)$, ceea ce conduce la ecuația separatorului $x_1 = 0$, adică axa Ox_1 . Evident, aceasta dreaptă este un separator perfect pentru întregul set de exemple, deci algoritmul nu va mai comite până la final nicio [altă] greșală. Concluzia este că, pe un același set de exemple de antrenament, algoritmul *Perceptron* poate comite un număr diferit de greșeli (și, de asemenea, un alt separator) dacă exemplele îi sunt prezentate în două secvențe diferite.

Observație importantă: Deși în ambele cazuri de mai sus (S și S') algoritmul *Perceptron* găsește un separator liniar pentru datele de intrare, acest fapt nu este garantat în general.

general, chiar dacă datele sunt liniar separabile. Vedeți de exemplu problema 36. (Motivele sunt: rata de învățare (implicită) prea mare și / sau neparcurgerea setului de date decât o singură dată.)

b. Cunoaștem din geometria analitică faptul că orice doi vectori \bar{w} și \bar{x} pentru care are loc relația $\bar{w} \cdot \bar{x} = 0$ sunt ortogonalii (adică, perpendiculari unul pe celălalt).

În cazul perceptronilor (de tip liniar, prag sau sigmoidal), ecuația separatorului este tocmai de forma $\bar{w} \cdot \bar{x} = 0$, unde \bar{x} este un punct arbitrar de pe dreapta [sau, în general, hiperplanul] separator. Considerând \bar{x}_1 și \bar{x}_2 două astfel de puncte, diferența lor, $\bar{x}_1 - \bar{x}_2$, va fi un vector care are direcția dreptei-separator. Din relațiile $\bar{w} \cdot \bar{x}_1 = 0$ și $\bar{w} \cdot \bar{x}_2 = 0$ rezultă $\bar{w} \cdot (\bar{x}_1 - \bar{x}_2) = 0$. Prin urmare, vectorul de ponderi \bar{w} este perpendicular pe direcția dreptei-separator.

Proprietatea aceasta se poate verifica în mod particular pe datele de la punctele precedente. De exemplu, la punctul a , pe sirul de exemple S , la iterația 1 avem $\bar{w} = (1, -2)$ și se poate observa direct pe grafic că acest vector este perpendicular pe direcția dreptei $y = \frac{1}{2}x$.

Așadar, răspunsul la întrebarea din enunț este *Adevărat*.

18.

(Convergența algoritmului de antrenare a perceptronului-prag, varianta Rosenblatt)

prelucrare de Liviu Ciortuz, 2014, după

■ *Tommy Jaakkola, MIT, ML course, 2009 fall, lecture notes 2, cf. Block and Novikoff's results, 1962 (see also CMU, 2016 fall, E. Xing, Z. Bar-Joseph, HW2, pr. 4 CMU, 2014 fall, E. Xing, B. Poczos, HW1, pr. 4.3)*

Presupunem că folosim perceptronul de tip prag în varianta Rosenblatt (vedeți problema 16) și vrem să învățăm un concept, folosind instanțele de antrenament $x_1, \dots, x_n, \dots \in \mathbb{R}^d$ împreună cu etichetele corespunzătoare $y_1, \dots, y_n, \dots \in \{-1, 1\}$.

Demonstrați că în cazul în care sunt îndeplinite condițiile *i-iv* de mai jos, algoritmul de actualizare a ponderilor perceptronului „converge”, adică termină într-un număr finit de pași. Formal, exprimăm acest fapt astfel: $\exists m \in \mathbb{N}$ astfel încât $y_t w^{(m)} \cdot x_t \geq 0$ pentru orice $t \in \{1, \dots, n, \dots\}$, unde $w^{(m)}$ este vectorul de ponderi obținut de perceptron la iterația m .

Iată acum *condițiile* menționate mai sus:

- i. Instanțele x_1, \dots, x_n, \dots sunt separabile liniar prin originea sistemului de coordonate, cu o margine finită $\gamma > 0$. Din punct de vedere formal, aceasta înseamnă că există $w^* \in \mathbb{R}^d$ astfel încât $y_t w^* \cdot x_t \geq \gamma$ pentru $t = 1, \dots, n, \dots$
- ii. Toate instanțele x_1, \dots, x_n, \dots sunt conținute într-o sferă din \mathbb{R}^d cu centrul în origine, adică $\exists R > 0$ astfel încât $\|x_t\| \stackrel{\text{def}}{=} \sqrt{x_t \cdot x_t} \leq R$ pentru orice t .
- iii. Învățarea se face în manieră *incrementală*, folosind următoarea regulă de actualizare a ponderilor:

$$w^{(k+1)} = w^{(k)} + y_{t_k} x_{t_k} \text{ pentru un } t_k \in \{1, \dots, n, \dots\} \text{ a.î. } y_{t_k} w^{(k)} \cdot x_{t_k} \leq 0, \quad (157)$$

ceea ce înseamnă că instanța x_{t_k} este clasificată eronat de către perceptron la iterația k .

iv. Startarea procesului de învățare se face cu $w^{(0)} = 0 \in \mathbb{R}^d$.

Indicație: Arătați că la fiecare iterare (k) a algoritmului, sunt satisfăcute următoarele proprietăți:

- a. $w^* \cdot w^{(k)} \geq k\gamma$;
- b. $\|w^{(k)}\|^2 \leq kR^2$;
- c. $k \leq \left(\frac{\|w^*\|}{\gamma} R\right)^2$.

Ultima inegalitate indică [o margine superioară pentru] numărul maxim de iterări execuție de către perceptron.

Observația 1: Notând cu θ_t unghiul format de vectorii x_t și w^* în \mathbb{R}^d și ținând cont de faptul că

$$\cos(\theta_t) = \cos(x_t, w^*) = \frac{x_t \cdot w^*}{\|x_t\| \|w^*\|},$$

deci

$$y_t w^* \cdot x_t = y_t \|w^*\| \|x_t\| \cos(\theta_t),$$

ceea ce, coroborat cu condiția *i*, implică $y_t \|x_t\| \cos(\theta_t) \|w^*\| \geq \gamma$, adică $y_t \|x_t\| \cos(\theta_t) \geq \frac{\gamma}{\|w^*\|}$. Din punct de vedere geometric, aceasta înseamnă că în spațiul \mathbb{R}^d distanța de la orice instanță x_t la vectorul w^* (care trece prin originea sistemului de coordonate) este mai mare sau egală cu $\frac{\gamma}{\|w^*\|}$.

Observația 2: Separatorul $w^{(k)}$ este o combinație liniară de instanțele x_i , întrucât regula de actualizare este $w^{(k+1)} = w^{(k)} + y_i x_i$. Observația aceasta este valabilă și la antrenarea perceptronului-prag clasic, bazat pe regula delta.

Răspuns:

Algoritmul de actualizare a ponderilor perceptronului determină schimbarea poziției separatorului la fiecare iterare (k) la care avem de a face cu un exemplu clasificat greșit. Intuitiv, ne așteptăm ca poziția separatorului la iterare $k+1$ (adică hiperplanul de ecuație $w^{(k+1)} \cdot x = 0$) să se apropie de poziția separatorului w^* care definește conceptul de învățat. În consecință, cosinusul unghiului dintre $w^{(k)}$ și w^* ar trebui să crească de la o iterare la alta. Pentru a dovedi / verifica în mod riguros aceasta, vom ține cont că, prin definiție,

$$\cos(w^{(k)}, w^*) = \frac{w^{(k)} \cdot w^*}{\|w^{(k)}\| \|w^*\|}.$$

Mai întâi vom compara valorile produsului scalar de la numărătorul fracției de mai sus, la două iterări successive. Folosind regula (157), putem scrie:

$$w^{(k+1)} \cdot w^* = (w^{(k)} + y_{t_k} x_{t_k}) \cdot w^* = w^{(k)} \cdot w^* + y_{t_k} x_{t_k} \cdot w^*.$$

Întrucât $y_{t_k} x_{t_k} \cdot w^* \geq \gamma$ (vedeți condiția *i* din enunț), rezultă că valoarea produsului scalar $w^{(k)} \cdot w^*$ crește la fiecare iterare cu o cantitate cel puțin egală cu γ . Cum $w^{(0)} = 0$ conform restricției *iv*, este imediat că $w^{(k)} \cdot w^* \geq k\gamma$ la iterare k . Așadar, numărătorul fracției care definește $\cos(w^{(k)}, w^*)$ crește cel puțin liniar în raport cu k . Cu aceasta, tocmai am demonstrat relația a.

A analiza evoluția numitorului fracției care definește $\cos(w^{(k)}, w^*)$ revine la a compara $\|w^{(k+1)}\|$ cu $\|w^{(k)}\|$, întrucât $\|w^*\|$ este constant în raport cu k .

$$\begin{aligned} \|w^{(k+1)}\|^2 &\stackrel{\text{def.}}{=} w^{(k+1)} \cdot w^{(k+1)} \stackrel{(157)}{=} (w^{(k)} + y_{t_k} x_{t_k}) \cdot (w^{(k)} + y_{t_k} x_{t_k}) \\ &= w^{(k)} \cdot w^{(k)} + 2y_{t_k} w^{(k)} \cdot x_{t_k} + y_{t_k}^2 x_{t_k} \cdot x_{t_k} \\ &= \|w^{(k)}\|^2 + 2y_{t_k} w^{(k)} \cdot x_{t_k} + y_{t_k}^2 \|x_{t_k}\|^2 \\ &= \|w^{(k)}\|^2 + 2y_{t_k} w^{(k)} \cdot x_{t_k} + \|x_{t_k}\|^2 \\ &\leq \|w^{(k)}\|^2 + R^2. \end{aligned}$$

Inegalitatea de mai sus derivă din faptul că $y_{t_k} w^{(k)} \cdot x_{t_k} \leq 0$ (i.e., exemplul t_k este clasificat greșit la iterată k , conform condiției *iii* din enunț) și din ipoteza că orice instanță de antrenament este conținută în sferă de rază R și având centrul în originea spațiului \mathbb{R}^d , conform condiției *ii*. Cum $w^{(0)} = 0$, este imediat că $\|w^{(k)}\|^2 \leq k R^2$, deci $\|w^{(k)}\| \leq \sqrt{k} R$ la iterată k . Cu aceasta, am demonstrat relația b.

Acum, combinând rezultatele a și b, obținem următoarea inegalitate:

$$\cos(w^{(k)}, w^*) = \frac{w^{(k)} \cdot w^*}{\|w^{(k)}\| \|w^*\|} \geq \frac{k\gamma}{\sqrt{k} R \|w^*\|} = \sqrt{k} \frac{\gamma}{R \|w^*\|}.$$

Stim că valoarea funcției cos este întotdeauna cel mult egală cu 1. În consecință,

$$1 \geq \cos(w^{(k)}, w^*) \geq \sqrt{k} \frac{\gamma}{R \|w^*\|} \Rightarrow k \leq \left(R \frac{\|w^*\|}{\gamma} \right)^2,$$

deci am demonstrat relația c. Aceasta înseamnă că în condițiile specificate în enunț, antrenarea perceptronului-prag se va face în cel mult $\left\lfloor \left(R \frac{\|w^*\|}{\gamma} \right)^2 \right\rfloor$ iterării, unde perechea de simboluri $\lfloor \rfloor$ desemnează funcția parte întreagă inferioară.

Observația 3: Marginea superioară calculată mai sus este remarcabilă, întrucât ea nu depinde nici de instanțele x_i și nici de dimensiunea (d) a spațiului din care sunt selectate aceste instanțe.

Observația 4: Restricția referitoare la separabilitatea prin origine a instanțelor de antrenament (vedeți condiția *i*, prima parte) — și anume, termenul „liber” w_0 asociat separătorului w^* trebuie să fie 0 — nu este de fapt limitativă.⁴⁰² Cazul general al separabilității liniare în \mathbb{R}^d , adică $y_t(w_0^* + w^* \cdot x_t) \geq 0$ pentru orice $t \in \{1, 2, \dots, n\}$, poate fi redus la cazul particular al separabilității prin origine în \mathbb{R}^{d+1} dacă pe de o parte se consideră $w' = (w_0^0, w_1^1, \dots, w_d^d)$, cu $w^* = (w_1^1, \dots, w_d^d)$, iar pe de altă parte se mapează toate instanțele de antrenament $x_t = (x_t^1, \dots, x_t^d)$ din \mathbb{R}^d în \mathbb{R}^{d+1} astfel: $x'_t = (x_t^0, x_t^1, \dots, x_t^d)$, cu $x_t^0 = 1$ pentru orice t . Cu această mapare, rezultă $y_t w' \cdot x'_t \geq 0$ pentru orice t (respectiv $y_t w' \cdot x'_t \geq \gamma$ când se lucrează cu margine finită, ca în enunțul acestei probleme). Nici restricția *iv* ($w^{(0)} = 0$) nu este cu adevărat limitativă; în general algoritmii de antrenare a unităților / rețelelor neuronale inițializează ponderile w la valori mici (în modul). Similar, este imediat că restricția potrivit căreia sfera în care sunt conținute instanțele x_1, \dots, x_n trebuie să aibă centrul în originea lui \mathbb{R}^d (vedeți condiția *ii*, partea a doua) poate fi eliminată fără ca rezultatul de convergență să fie afectat.

⁴⁰²În esență, ea ne ușurează calculele pe parcursul demonstrației, fiindcă ne plasează într-o situație particulară.

Observația 5: Deducerea marginii superioare de la punctul c de mai sus în cazul în care se folosește o rată de învățare oarecare $\eta > 0$ se face extinzând în mod natural demonstrația de mai sus (vedeți pr. 39). În concluzie, mai rămân de examinat două condiții: separabilitatea liniară cu margine $\gamma > 0$ și conținerea instanțelor de antrenament într-o sferă de rază finită. Problema 37 va cere să se demonstreze că ambele condiții sunt esențiale pentru convergența perceptronului Rosenblatt în regim de învățare online.

19.

(Perceptronul kernel-izat [dual])

*prelucrare de Liviu Ciortuz, după
CMU, 2013 spring, A. Smola, B. Poczos, HW2, pr. 2.d
Stanford, 2016 fall, A. Ng, J. Duchi, HW2, pr. 2
MIT, 2006 fall, Tommy Jaakkola, HW2, pr. 3*

Majoritatea clasificatorilor liniari pot fi kernel-izați, în vederea clasificării de date care sunt neseparabile liniar. În acest exercițiu vă cerem să elaborați varianta kernel-izată [duală] a algoritmului *Perceptron*. (Pentru pseudo-codul variantei simple, nekernel-izate a algoritmului *Perceptron*, vedeți pr. 16.)

Ca input, acest algoritm va lua secvența de instanțe etichetate $\{(x_i, y_i)\}_{i=1}^n$ cu $x_i \in \mathbb{R}^d$ și $y_i \in \{-1, 1\}$, precum și funcția-nucleu $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^m$, cu proprietatea că există $m > d$ și o funcție („mapare“) $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ astfel încât $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ pentru orice x_i, x_j .⁴⁰³ Perceptronul kernel-izat [dual] va lucra nu cu instanțele x_i , ci cu imaginile lor, $\phi(x_i)$, și va încerca să găsească pentru acestea un separator liniar în spațiul \mathbb{R}^m .

Comentariu: Analizând algoritmul *Perceptron*, veți constata că vectorul de ponderi $w \in \mathbb{R}^d$ este o combinație liniară de instanțe x_i , desemnată prin expresia $w = \sum_{i=1}^n \alpha_i x_i$. Din acest motiv este suficient să găsim valorile coeficienților α_i care asigură separabilitatea datelor. Această proprietate se dovedește deosebit de utilă în cazul funcțiilor de mapare (ϕ) cu vectori de „trăsături“ ($\phi(x)$) de dimensiuni foarte mari sau chiar infiniti, aşa cum este cazul funcțiilor-nucleu gaussiene, numite și *funcții cu bază radială* (engl., Radial Basis Functions, RBF). Așadar, în loc să actualizeze vectorul de ponderi w (cum face Perceptronul simplu), Perceptronul kernel-izat [dual] actualizează direct coeficienții $\alpha_1, \dots, \alpha_n$. Se va vedea că în timpul actualizărilor, operațiile care se fac asupra instanțelor de antrenament date (x_i) sunt doar produse scalare de forma $\phi(x_i) \cdot \phi(x_j)$, care, aşa cum am precizat mai sus, pot fi văzute ca valori ale funcției-nucleu, $K(x_i, x_j)$. Această proprietate este necesară pentru asigurarea eficienței calculelor.

Arătați cum anume se scrie regula de actualizare a coeficienților $\alpha_1, \dots, \alpha_n$ la procesarea unui nou exemplu (x_i, y_i) de către algoritmul *Perceptron* kernel-izat [dual], și cum se face predicția ($y = 1$ sau -1) pentru o instanță nouă x .

Răspuns:

În spațiul \mathbb{R}^m (care se mai numește, în contextul kernel-izării, *spațiul de trăsături*), regula de actualizare a ponderilor w se scrie astfel:

$$w^{(i)} \leftarrow w^{(i-1)} + y_i \phi(x_i) \text{ dacă } y_i w^{(i-1)} \cdot \phi(x_i) \leq 0,$$

unde operatorul \cdot desemnează produsul scalar.

⁴⁰³Din punct de vedere practic, este necesar ca funcția K să poată fi calculată în mod eficient.

În consecință, întrucât facem initializarea $w^{(0)} = \bar{0}$, vectorul $w \in \mathbb{R}^m$ va fi întotdeauna o combinație liniară de vectori de trăsături, $\phi(x_i)$. Aceasta înseamnă că există coeficienții $\alpha_i \in \mathbb{R}$ astfel încât $w^{(i)} = \sum_{l=1}^i \alpha_l \phi(x_l)$ după procesarea primelor i instanțe de antrenament. Așadar, vectorul $w^{(i)}$ poate fi reprezentat în mod compact prin coeficienții α_l (cu $l = 1, \dots, i$) din această combinație liniară. În particular, valoarea initială $w^{(0)}$ corespunde cazului când suma nu conține niciun termen (adică avem o listă vidă de coeficienți α_l).

Arătăm acum că putem calcula în mod eficient coeficienții α_i .

La iterația i trebuie să calculăm produsul scalar $w^{(i-1)} \cdot \phi(x_i)$. Întrucât produsul scalar în spațiul de trăsături \mathbb{R}^m este o operație costisitoare atunci când m este mare, vom ține cont că

$$w^{(i-1)} \cdot x_i = \left(\sum_{l=1}^{i-1} \alpha_l \phi(x_l) \right) \cdot \phi(x_i) = \sum_{l=1}^{i-1} \alpha_l (\phi(x_l) \cdot \phi(x_i)) = \sum_{l=1}^{i-1} \alpha_l K(x_l, x_i),$$

ceea ce înseamnă că acești coeficienți se pot calcula într-adevăr în mod eficient.

În mod similar, se poate face în mod eficient predicția clasei / etichetei pentru o instanță nouă (de test) $x \in \mathbb{R}^d$:

$$w^{(i)} \cdot \phi(x) = \sum_{l=1}^i \alpha_l \phi(x_l) \cdot \phi(x) = \sum_{l=1}^i \alpha_l K(x_l, x).$$

Sumarizând, algoritmul pentru antrenarea Perceptronului kernel-izat [dual] se poate scrie în pseudo-cod astfel:

```

initialize  $\alpha_i = 0$  for  $i = 1, \dots, n$ ;
for  $i = 1, \dots, n$  do
    if  $y_i \sum_{l=1}^{i-1} \alpha_l K(x_l, x_i) \leq 0$  then
         $\alpha_i \leftarrow y_i$ 
    end if
end for

```

Observație importantă: Se poate arăta relativ ușor că raționamentele (și rezultatele) din acest exercițiu se pot extinde și la cazul perceptronului care folosește funcție de activare de tip prag (în particular, funcția *sign*), rată de învățare oarecare $\eta > 0$ — dar face initializarea vectorului de ponderi $w \in \mathbb{R}^d$ cu $\bar{0}$, ceea ce este echivalent cu $\alpha_i = 0$, pentru $i = 1, \dots, n$ — și care eventual parurge de mai multe ori setul de date de antrenament.

În acest caz, regula de actualizare a perceptronului kernel-izat [dual] este de forma

$$\alpha_i \leftarrow \alpha_i + \eta(y_i - \text{sign}(w^{(i-1)} \cdot \phi(x_i))) \text{ dacă } y_i \text{sign}(w^{(i-1)} \cdot \phi(x_i)) \leq 0$$

unde, exact ca mai sus,

$$w^{(i-1)} \cdot \phi(x_i) = \left(\sum_{l=1}^{i-1} \alpha_l \phi(x_l) \right) \cdot \phi(x_i) = \sum_{l=1}^{i-1} \alpha_l K(x_l, x_i).$$

6.1.3 Rețele “feed-forward” — algoritmul de retropropagare

20.

(Rețele feed-forward, algoritmul de retro-propagare: o generalizare simplă în raport cu *Machine Learning* de Tom Mitchell, pag. 101-103)

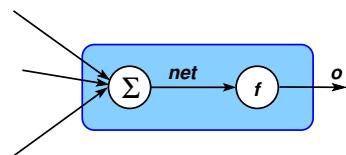
■ prelucrare de Liviu Ciortuz, după
CMU, 2008(?) spring, HW2, pr. 2.2-4

La acest exercițiu veți deriva regula de actualizare pentru algoritmul de retro-propagare (varianta stochastică) pentru o rețea neuronală artificială de tip “feed-forward”, cu două niveluri de unități sigmoidale. Se consideră d unități de intrare, H unități ascunse și K unități de ieșire.

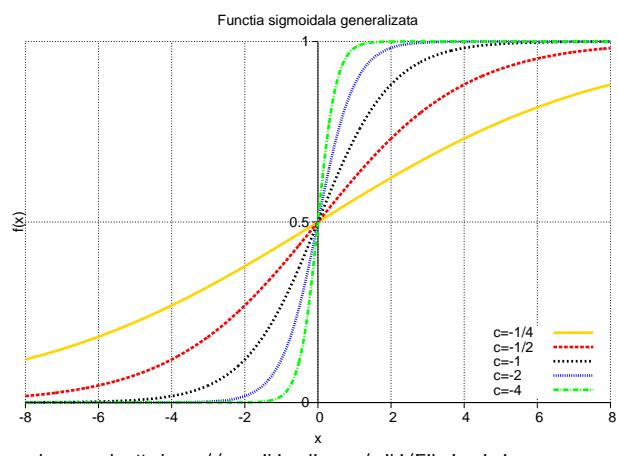
Funcția de eroare cu care se lucrează este

$$E(\bar{w}) = \frac{1}{2} \sum_k (t_k - o_k)^2,$$

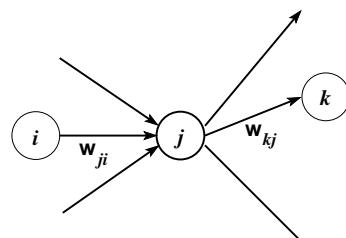
unde $o_k = f(\text{net}_k) = \frac{1}{1 + e^c \text{net}_k}$, iar c este o constantă.



Observație: Este de remarcat faptul că se poate folosi constanta c pentru a „controla“ panta cu care se face smoothing-ul (netezirea) funcției treaptă 0/1. Valori negative mici (adică, mari în modul) ale lui c implică o trecere foarte rapidă dinspre valori apropiate de 0 înspre valori apropiate de 1. Similar, valori negative mari (adică, apropiate de 0) pentru c implică o trecere lentă de la valori ≈ 0 la valori ≈ 1 .



- a. Calculați derivata funcției sigmoidale generalizate $f(z) = \frac{1}{1 + e^{cz}}$ și exprimați rezultatul în funcție de însuși $f(z)$, adică fără a-l implica direct pe z .
- b. Fie w_{kj} ponderea conexiunii de la unitatea ascunsă j către unitatea de ieșire k . Arătați că regula de actualizare pentru w_{kj} este de forma $w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$, unde y_j este ieșirea unității ascunse j , iar $\delta_k = f'(\text{net}_k)(t_k - o_k)$, cu $\text{net}_k = \sum_{j'} w_{kj'} y_{j'}$.
- c. Arătați că regulile de actualizare pentru ponderile care corespund conexiunilor input-to-hidden sunt de forma $w_{ji} \leftarrow w_{ji} + \eta \tilde{\delta}_j x_i$, unde x_i este intrarea i , iar $\tilde{\delta}_j = f'(\text{net}_j)[\sum_{k=1}^K w_{kj} \delta_k]$, cu $\text{net}_j = \sum_{i'} w_{ji'} x_{i'}$.



Răspuns:

Demonstrația de mai jos urmează, în esență, aceeași linie de gândire ca și cea din carteau lui Tom Mitchell, *Machine Learning* (1997), pag. 101–103, unde s-a lucrat cu $c = -1$. Noi am preferat însă o redactare care să urmeze mai îndeaproape aplicarea formulelor de analiză matematică.

a. Mai întâi calculăm derivata lui f în raport cu argumentul z :

$$\begin{aligned}\frac{\partial f(z)}{\partial z} &= \frac{\partial}{\partial z} \left(\frac{1}{1 + e^{cz}} \right) \\ &= -\frac{1}{(1 + e^{cz})^2} \cdot \frac{\partial}{\partial z} (1 + e^{cz}) = -\frac{1}{(1 + e^{cz})^2} \cdot c \cdot e^{cz}.\end{aligned}$$

Apoi, în expresia pe care tocmai am obținut-o, fortăm apariția lui $f(z)$:

$$\begin{aligned}\frac{\partial f(z)}{\partial z} &= -\frac{1}{(1 + e^{cz})^2} \cdot c \cdot e^{cz} = -c \cdot \frac{1}{1 + e^{cz}} \cdot \frac{e^{cz}}{1 + e^{cz}} = -c \cdot \frac{1}{1 + e^{cz}} \cdot \frac{1 + e^{cz} - 1}{1 + e^{cz}} \\ &= -c \cdot \frac{1}{1 + e^{cz}} \cdot \left(1 - \frac{1}{1 + e^{cz}} \right) = -cf(z)(1 - f(z)).\end{aligned}$$

Observație importantă: Punctele b și c vor fi rezolvate de fapt în raport cu o funcție oarecare f derivabilă, lăsată nespecificată. Forma particulară dată în enunț pentru funcția f (folosită la punctul a) nu are nicio consecință particulară asupra raționamentului dezvoltat în continuare.

b. Conform metodei gradientului descendente, actualizarea ponderilor w_{kj} de pe conexiunile care intră într-o unitate de ieșire k se va face conform formulei

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E}{\partial w_{kj}},$$

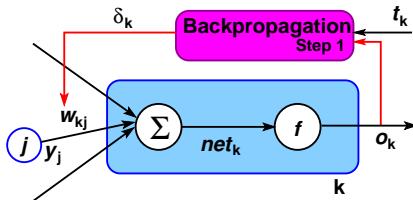
unde η este o constantă pozitivă (rata de învățare).

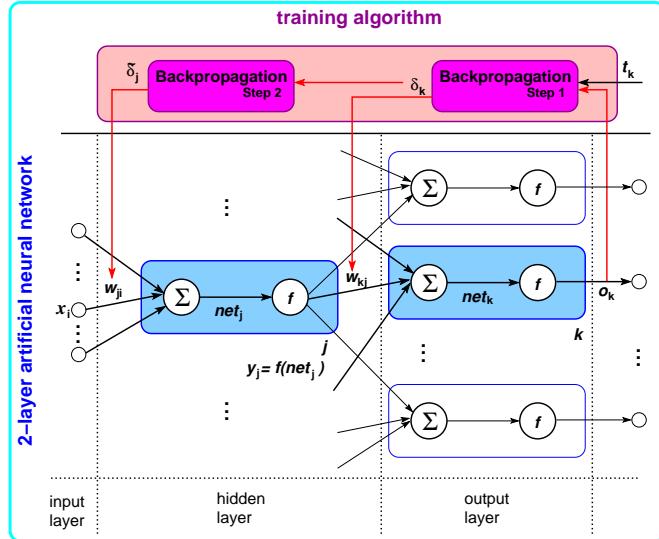
Intuitiv, ponderea w_{kj} influențează valoarea funcției de eroare E (doar) prin intermediul lui net_k , valoarea care este transmisă componentei de activare a unității k de pe nivelul de ieșire. (S-a notat cu net_k suma $\sum_{j'} w_{kj'} y_{j'}$.)

Așadar, este necesară aplicarea formulei pentru derivarea unei compunerii de funcții derivabile:

$$\begin{aligned}\frac{\partial E}{\partial w_{kj}} &= \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = \left(\frac{\partial}{\partial net_k} \frac{1}{2} \sum_{k'=1}^K (t_{k'} - o_{k'})^2 \right) \left(\frac{\partial}{\partial w_{kj}} \sum_{j'=0}^{n_k} w_{kj'} y_{j'} \right) \\ &= \left(\frac{\partial}{\partial net_k} \frac{1}{2} (t_k - f(net_k))^2 \right) y_j \\ &= y_j \left(\frac{1}{2} 2(t_k - f(net_k)) \frac{\partial}{\partial net_k} (t_k - f(net_k)) \right) = -y_j (t_k - f(net_k)) f'(net_k),\end{aligned}$$

unde n_k este numărul de intrări ale unității k . În particular, $n_k = H$ dacă se consideră toate conexiunile posibile (hidden-to-output).





Dacă notăm $\delta_k = (t_k - f(\text{net}_k))f'(\text{net}_k)$, atunci avem $\frac{\partial E}{\partial w_{kj}} = -\delta_k y_j$, iar regula de actualizare a ponderilor unității ascunse j devine:

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E}{\partial w_{kj}} = w_{kj} + \eta \delta_k y_j$$

c. Ca și la punctul precedent, actualizarea ponderilor w_{ji} de pe conexiunile care intră într-o unitate ascunsă j se va face după formula $w_{ji} \leftarrow w_{ji} - \eta \frac{\partial E}{\partial w_{ji}}$. Ponderea w_{ji} influențează valoarea funcției de eroare E (doar) prin intermediul lui net_j , valoarea care intră în componenta de activare a unității ascunse j .

Folosind din nou formula pentru derivarea unei compunerii de funcții derivabile și ținând cont că $\text{net}_j = \sum_{i'} w_{ji'} x_{i'}$, vom avea:

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ji}} = \left(\frac{\partial}{\partial \text{net}_j} \frac{1}{2} \sum_{k=1}^K (t_k - o_k)^2 \right) \left(\frac{\partial}{\partial w_{ji}} \sum_{i'=0}^d w_{ji'} x_{i'} \right) \\ &= \left(\frac{1}{2} \sum_{k=1}^K 2(t_k - o_k) \frac{\partial}{\partial \text{net}_j} (t_k - o_k) \right) x_i = -x_i \sum_{k=1}^K (t_k - o_k) \frac{\partial o_k}{\partial \text{net}_j} \end{aligned}$$

Pe de altă parte, valoarea net_j influențează valoarea o_k (doar) prin intermediul lui net_k . Așadar, urmează că

$$\begin{aligned} \frac{\partial o_k}{\partial \text{net}_j} &= \frac{\partial o_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial \text{net}_j} = f'(\text{net}_k) \frac{\partial}{\partial \text{net}_j} \sum_{j'=0}^{n_k} w_{kj'} y_{j'} \\ &= f'(\text{net}_k) \frac{\partial}{\partial \text{net}_j} \sum_{j'=0}^{n_k} w_{kj'} f(\text{net}_{j'}) = f'(\text{net}_k) f'(\text{net}_j) w_{kj} \end{aligned}$$

Înlocuind acest rezultat în egalitatea precedentă, vom avea:

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}} &= -x_i \sum_{k=1}^K (t_k - o_k) f'(net_k) f'(net_j) w_{kj} \\ &= -x_i f'(net_j) \sum_{k=1}^K (t_k - o_k) f'(net_k) w_{kj} \\ &= -x_i f'(net_j) \sum_{k=1}^K \delta_k w_{kj}\end{aligned}$$

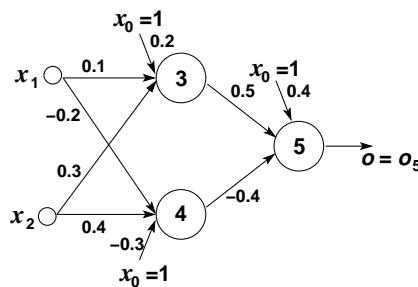
Folosind notația $\tilde{\delta}_j = f'(net_j) \sum_{k=1}^K \delta_k w_{kj}$, egalitatea de mai sus devine $\frac{\partial E}{\partial w_{ji}} = -x_i \tilde{\delta}_j$, iar regula de actualizare a ponderii se transformă în:

$$w_{ji} \leftarrow w_{ji} + \eta \tilde{\delta}_j x_i$$

Observație: În mod similar cu demonstrația din cartea *Machine Learning* de Tom Mitchell, pag. 101-103, demonstrația de aici poate fi extinsă în mod facil la rețele neuronale “feed-forward” cu un număr oarecare de niveluri ascunse. De asemenea, mai este posibilă *încă o generalizare*: funcția de activare f poate fi diferită de la o unitate la alta (sigmoidală, generalizat-sigmoidală (cu o anumită constantă c fixată), liniară etc). În demonstrație va fi suficient să atașăm simbolului f indicele unității neuronale respective.

21. (Aplicarea algoritmului de retro-propagare a erorilor pe o rețea feed-forward formată din unități sigmoidale dispuse pe 2 niveluri)
*prelucrare de Liviu Ciortuz,
după un exemplu din „Apprentissage artificiel“,
■ A. Cornuéjols, L. Miclet, 2010, pag. 332, 341*

Rețeaua neuronală din figura următoare este formată din unități sigmoidale.



- a. Care este rezultatul produs de această rețea pentru intrarea $x \stackrel{\text{not.}}{=} (x_1, x_2) = (1, 1)$?

Recomandare: Pentru ușurința urmăririi calculelor, vă cerem să completați un tabel de forma următoare:

i	$net_i = \sum_j w_{ij} x_j$	$o_i = \sigma(net_i)$
3		
4		
5		

b. La acest punct, veți executa manual prima iterare a algoritmului de retro-propagare pe rețeaua dată. Presupunem că în cazul intrării $x = (1, 1)$ ieșirea produsă de rețea ar trebui să fie $t = 0$. Luând rata de învățare $\eta = 1$, precizați care vor fi

- valorile ponderilor $w_{30}, w_{31}, w_{32}, w_{40}, w_{41}, w_{42}, w_{50}, w_{51}, w_{52}$, după aplicarea acestei prime iterării în antrenarea rețelei;⁴⁰⁴
- noul output produs de rețea (după actualizarea ponderilor) pe aceeași intrare $x = (1, 1)$.

c. Comparați rezultatele de la punctele *a* și *b*. Ce constatăți?

Răspuns:

a. Completăm tabelul dat, ținând cont că ieșirile neuronilor de pe nivelul ascuns (unitățile sigmoidale 3 și 4) constituie intrările neuronului de pe nivelul exterior (unitatea sigmoidală 5).

i	$net_i = \sum_j w_{ij}x_j$	$o_i = \sigma(net_i)$
3	$0.2 + 0.1 + 0.3 = 0.6$	$1/(1 + e^{-0.6}) \simeq 0.646$
4	$-0.3 - 0.2 + 0.4 = -0.1$	$1/(1 + e^{0.1}) \simeq 0.475$
5	$0.4 + 0.5 \cdot 0.646 - 0.4 \cdot 0.475 = 0.533$	$1/(1 + e^{-0.533}) \simeq 0.630$

Așadar, outputul produs de rețea este 0.63.

b. Pentru a vedea detaliile algoritmului de retro-propagare a erorii, cititorul poate consulta cartea *Machine Learning* de Tom Mitchell, pag. 98 sau problema 20 din prezentul capitol, punctele *b* și *c* (unde parametrul *c* va fi considerat ca având valoarea 1).

Precizare: Ca să facilităm înțelegerea modului în care se aplică acest algoritm, în schema de mai jos am arătat cum anume vor fi calculate noile valori ale ponderilor w , precum și cantitățile δ implicate în execuția unei iterării a algoritmului. Ilustrarea se rezumă la parcurgerea (în ordinea output-to-input) unuia dintre drumurile din această rețea, și anume drumul 5, 4, 2. Extensia la celelalte căi este facilă. În această imagine (dar numai aici) am notat cu w' noile valori pentru ponderi (calculate în ultimul pas al iterării), pentru a nu fi confundate cu vechile valori, care intervin în calculul cantităților δ .

Conform algoritmului de retro-propagare,

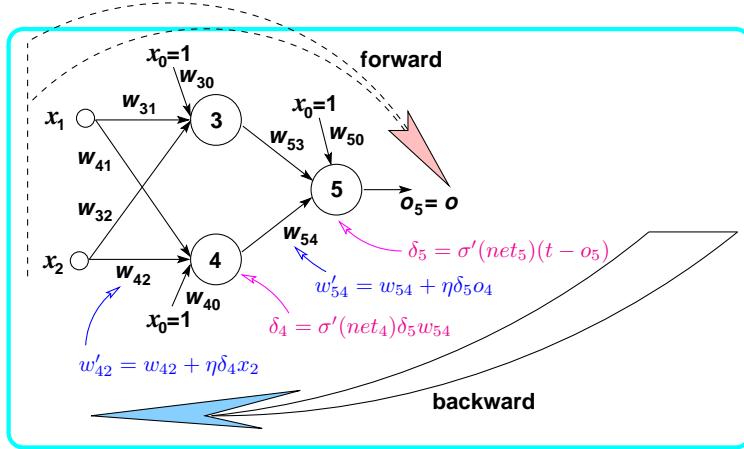
$$\begin{aligned} \delta_5 &\stackrel{\text{def.}}{=} -\frac{\partial E}{\partial net_5} = \sigma'(net_5)(t - o) = \sigma(net_5)(1 - \sigma(net_5))(t - o) \\ &= o(1 - o)(t - o) = 0.63(1 - 0.63)(0 - 0.63) = -0.147 \end{aligned}$$

Outputul neuronului 3 constituie unul din inputurile neuronului 5. În notația folosită de Tom Mitchell, vom scrie $Downstream(3) = \{5\}$. Așadar,

$$\delta_3 \stackrel{\text{def.}}{=} -\frac{\partial E}{\partial net_3} = o_3 \cdot (1 - o_3) \cdot \delta_5 \cdot w_{53} = 0.646 \cdot (1 - 0.646) \cdot (-0.147) \cdot 0.5 \simeq -0.017$$

În mod similar, $Downstream(4) = \{5\}$, și

$$\delta_4 \stackrel{\text{def.}}{=} -\frac{\partial E}{\partial net_4} = o_4 \cdot (1 - o_4) \cdot \delta_5 \cdot w_{54} = 0.475 \cdot (1 - 0.475) \cdot (-0.147) \cdot (-0.4) \simeq 0.015$$



Întrucât am terminat de calculat toate cantitățile de tip δ_j , vom trece acum la actualizarea ponderilor rețelei. Mai întâi vom calcula noile valori pentru ponderile de pe conexiunile hidden-to-output, deci vom avea:

$$w_{5j} \leftarrow w_{5j} + \Delta w_{5j} \text{ cu } \Delta w_{5j} = \eta \delta_5 o_j = \delta_5 o_j \text{ pentru } j \in \{0, 3, 4\}.$$

Așadar,

$$\begin{cases} \Delta w_{50} = -0.147 \cdot 1 = -0.147 \\ \Delta w_{53} = -0.147 \cdot 0.646 \simeq -0.095 \\ \Delta w_{54} = -0.147 \cdot 0.475 \simeq -0.070 \end{cases} \Rightarrow \begin{cases} w_{50} \leftarrow 0.4 - 0.147 \simeq 0.253 \\ w_{53} \leftarrow 0.5 - 0.095 = 0.405 \\ w_{54} \leftarrow -0.4 - 0.070 = -0.470 \end{cases}$$

Apoi vom actualiza ponderile care corespund conexiunilor de tip input-to-hidden. Știm că $\Delta w_{3i} = \eta \delta_3 x_i$ pentru $i \in \{0, 1, 2\}$, așadar inputul $x_0 = x_1 = x_2 = 1$ va implica $\Delta w_{30} = \Delta w_{31} = \Delta w_{32} = 1 \cdot (-0.017) \cdot 1 = -0.017$.

În consecință,

$$\begin{cases} w_{30} \leftarrow 0.2 - 0.017 = 0.183 \\ w_{31} \leftarrow 0.1 - 0.017 = 0.083 \\ w_{32} \leftarrow 0.3 - 0.017 = 0.283 \end{cases}$$

Similar, fiindcă $\Delta w_{4i} = \eta \delta_4 x_i$ pentru $i \in \{0, 1, 2\}$, rezultă că $\Delta w_{40} = \Delta w_{41} = \Delta w_{42} = 1 \cdot 0.015 \cdot 1 = 0.015$ și, prin urmare:

$$\begin{cases} w_{40} \leftarrow -0.3 + 0.015 = -0.285 \\ w_{41} \leftarrow -0.2 + 0.015 = -0.185 \\ w_{42} \leftarrow 0.4 + 0.015 = 0.415 \end{cases}$$

⁴⁰⁴Semnificația pentru w_{ji} este cea din cartea *Machine Learning* lui Tom Mitchell, la pag. 98: w_{ji} este ponderea de pe arcul / legătura de la unitatea i către unitatea j . A se vedea și reprezentarea grafică a rețelei din rezolvarea de mai jos.

Acum putem calcula noua valoare produsă de rețea neuronală:

i	$net_i = \sum_j w_{ij} x_j$	$o_i = \sigma(net_i)$
3	$0.183 + 0.083 + 0.283 = 0.549$	$1/(1 + e^{-0.549}) \simeq 0.634$
4	$-0.285 - 0.185 + 0.415 = -0.055$	$1/(1 + e^{0.055}) \simeq 0.486$
5	$0.253 + 0.405 \cdot 0.634 - 0.47 \cdot 0.486 = 0.281$	$1/(1 + e^{-0.281}) \simeq 0.569$

Așadar, outputul produs de rețea după efectuarea primei iterări din cadrul algoritmului de retro-propagare a erorii este 0.569.

c. Valoarea produsă de rețea după ce a fost aplicată o iterăre din algoritmul de retro-propagare (0.569) este mai aproape de valoarea-target (0) decât fusese outputul produs de rețea înainte de aplicarea acestei prime iterări (0.63). Este de așteptat ca rezultatul să se îmbunătățească la execuția următoarelor iterări ale algoritmului.

22.

(Regularizare: prevenirea overfitting-ului; cazul perceptronului liniar și al rețelelor feed-forward formate din astfel de perceptroni)
prelucrare de Liviu Ciortuz, după Tom Mitchell, "Machine Learning", 1997, pr. 4.10

În vederea reducerii riscului de overfitting, pentru o unitate liniară (engl., un-thresholded perceptron) se poate considera funcția de eroare

$$E(\bar{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 + \gamma \sum_i w_i^2,$$

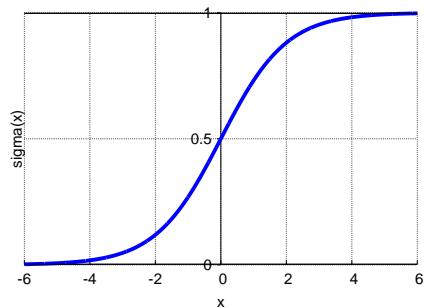
unde:

d indexează instanțele de antrenament din multimea $D \subset \mathbb{R}^n$,
 $\bar{w} = (w_0, w_1, \dots, w_n)$ sunt ponderile pentru intrările perceptronului,
 t_d este target-ul de învățat pentru instanță d ,
 o_d este outputul obținut de perceptron pentru aceeași instanță d ,
 γ este o constantă pozitivă.

Observație:

Suma $\sum_i w_i^2$ se poate scrie ca $\|w\|^2$, unde $\|w\|$ notează norma vectorului w . Intuitiv, minimizarea funcției obiectiv $E(\bar{w})$ va implica menținerea lui $\|w\|^2$ la o valoare destul de redusă.

Efectul practic, în cazul folosirii de unități sigmoidale este următorul: granița (suprafața) de decizie calculată de către rețea pentru ponderi w mici (în modul) este aproape liniară — a se vedea graficul funcției σ în jurul originii —, ceea ce o împiedică să se „muleze” în jurul neregularităților din datele de antrenament.



a. Folosind metoda gradientului descendente, derivați regula de actualizare a ponderilor w_i pentru unitatea liniară care utilizează funcția de eroare dată mai sus.

b. La acest punct vom considera că avem o rețea de tip feed-forward cu două niveluri de unități liniare și că forma funcției de eroare pentru întreaga rețea este similară cu cea indicată la punctul precedent. Pentru acest tip de rețea, vă cerem să deduceți regulile de actualizare a ponderilor. Pentru conveniență, veți considera varianta stochastică / incrementală, adică veți lucra cu o singură instanță de antrenament pe ciclu / „epocă“ de antrenare.

Observație: Deși orice rețea neuronală formată doar din unități liniare este echivalentă cu o unitate liniară, am propus punctul de mai sus pentru a servi drept model pentru rezolvarea problemei propuse 45.

c. Arătați că minimizarea funcției $E(\bar{w})$ într-adevăr va conduce la găsirea unor valori mai apropiate de 0 pentru w_i decât în varianta clasică.

Răspuns:

Se poate arăta imediat că expresia

$$E(\bar{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 + \gamma \sum_i w_i^2 = \frac{1}{2} \sum_{d \in D} (t_d - \sum_i w_i x_{i,d})^2 + \gamma \sum_i w_i^2$$

este o funcție convexă în raport cu toate argumentele w_i . Așadar, în vederea minimizării erorii, se justifică folosirea metodei gradientului descendente.

a. Avem:

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial w_i} \left(\frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 + \gamma \sum_{i'} w_{i'}^2 \right) = 2 \cdot \frac{1}{2} \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) + 2\gamma w_i \\ &= - \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} \sum_{i'} w_{i'} x_{i',d} + 2\gamma w_i = - \sum_d (t_d - o_d) x_{i,d} + 2\gamma w_i \end{aligned}$$

Așadar,

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i} = w_i + \eta \left[\sum_d (t_d - o_d) x_{i,d} - 2\gamma w_i \right] = (1 - 2\eta\gamma) w_i + \eta \sum_d (t_d - o_d) x_{i,d}$$

b. Înainte de a proceda la rezolvarea propriu-zisă a acestui punct al problemei, vom face următoarea *observație importantă*:

Cititorul atent va remarcă, desigur, analogia cu problema 20 (și problema 43). Această analogie este justificată, însă doar într-o anumită măsură. Va trebui să ținem cont că:

i. Aici f corespunde funcției identitate, fiindcă lucrăm cu unitatea liniară. Așadar, $f'(net) = 1$ pentru orice valoare a expresiei net .

ii. Nu vom mai putea lucra cu expresiile

$$\delta_k \stackrel{\text{def.}}{=} \frac{\partial E}{\partial net_k} \text{ și } \tilde{\delta}_j \stackrel{\text{def.}}{=} \frac{\partial E}{\partial net_j}$$

unde k este indicele unei unități neuronale de pe nivelul de ieșire iar j este indicele unei unități de pe nivelul ascuns. Explicația ține de faptul că expresia E nu se poate scrie

ca o compunere de funcții $E(\text{net}_-(w_-))$, ci doar ca $E(\text{net}_-(w_-), w_-)$, deci pur și simplu $E(\bar{w})$.⁴⁰⁵ Așadar, nu vom putea scrie, ca în rezolvarea problemei 20 (și a problemei 43), că $\frac{\partial E}{\partial w_-} = \frac{\partial E}{\partial \text{net}_-} \frac{\partial \text{net}_-}{\partial w_-}$. În consecință, va trebui să calculăm direct derivatele parțiale $\frac{\partial E}{\partial w_-}$. Similar cu demonstrația din cartea *Machine Learning* de Tom Mitchell de la pag. 101-103 și cu rezolvarea problemei 20 punctele b și c , vom calcula aceste derivate parțiale mai întâi pentru variabilele care reprezintă ponderile de pe conexiunile hidden-to-output ale rețelei și apoi pentru variabilele care corespund ponderilor de pe conexiunile input-to-hidden.

Pentru conveniență, la acest punct al problemei vom urma convenția de scriere a variabilelor w folosind doi indici: w_{kj} va corespunde unei conexiuni hidden-to-output (unde j indică o unitate de pe nivelul ascuns, iar k o unitate de pe nivelul de ieșire), iar w_{ji} va corespunde unei conexiuni input-to-hidden (i – input, j – hidden). Notațiile t_k și o_k vor corespunde valorii-target și respectiv outputului unității liniare k de pe nivelul de ieșire al rețelei.

Avem:

$$E(\bar{w}) = \frac{1}{2} \sum_{k \in \text{Outputs}} (t_k - o_k)^2 + \gamma \|\bar{w}\|^2$$

și, în consecință:

$$\begin{aligned} \frac{\partial E}{\partial w_{kj}} &= \frac{\partial}{\partial w_{kj}} \left[\frac{1}{2}(t_k - o_k)^2 + \gamma w_{kj}^2 \right] + \sum_{k' \neq k} \frac{\partial}{\partial w_{kj}} \left[\frac{1}{2}(t_{k'} - o_{k'})^2 + \gamma w_{k'j}^2 \right] \\ &= -\frac{1}{2}2(t_k - o_k) \frac{\partial}{\partial w_{kj}} o_k + 2\gamma w_{kj} \\ &= -(t_k - o_k) \frac{\partial}{\partial w_{kj}} \sum_{j'} w_{kj'} y_{j'} + 2\gamma w_{kj} = -(t_k - o_k) y_j + 2\gamma w_{kj} \end{aligned}$$

Așadar,

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E}{\partial w_{kj}} = w_{kj} + \eta[(t_k - o_k)y_j - 2\gamma w_{kj}] = (1 - 2\eta\gamma)w_{kj} + \eta(t_k - o_k)y_j$$

Fie acum indicii j și i fixați ca mai sus, corespunzător unei conexiuni input-to-hidden. Notând ca și Tom Mitchell $\text{Downstream}(j)$ mulțimea tuturor indicilor k cu proprietatea că există conexiune de la unitatea j către unitatea k , vom avea:

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial}{\partial w_{ji}} \left\{ \left[\sum_{k \in \text{Downstream}(j)} \frac{1}{2}(t_k - o_k)^2 \right] + \gamma w_{ji}^2 \right\} \\ &= 2\gamma w_{ji} - \sum_{k \in \text{Downstream}(j)} (t_k - o_k) \frac{\partial}{\partial w_{ji}} o_k \\ &= 2\gamma w_{ji} - \sum_{k \in \text{Downstream}(j)} \left[(t_k - o_k) \frac{\partial}{\partial w_{ji}} \left(\sum_{j'} w_{kj'} y_{j'} \right) \right] \\ &= 2\gamma w_{ji} - \sum_{k \in \text{Downstream}(j)} \left\{ (t_k - o_k) \frac{\partial}{\partial w_{ji}} \left[\sum_{j'} w_{kj'} \left(\sum_{i'} w_{j'i'} x_{i'} \right) \right] \right\} \end{aligned}$$

⁴⁰⁵În această notație, caracterul ‘_’ ține loc de indice, lăsându-i valoarea nespecificată.

$$\begin{aligned}
&= 2\gamma w_{ji} - \sum_{k \in \text{Downstream}(j)} \left[(t_k - o_k) \frac{\partial}{\partial w_{ji}} \left(\sum_{j'} \sum_{i'} w_{kj'} w_{j'i'} x_{i'} \right) \right] \\
&= 2\gamma w_{ji} - x_i \left(\sum_{k \in \text{Downstream}(j)} (t_k - o_k) w_{kj} \right)
\end{aligned}$$

Așadar,

$$\begin{aligned}
w_{ji} \leftarrow w_{ji} - \eta \frac{\partial E}{\partial w_{ji}} &= w_{ji} + \eta \left[x_i \left(\sum_{k \in \text{Downstream}(j)} w_{kj} (t_k - o_k) \right) - 2\gamma w_{ji} \right] \\
&= (1 - 2\eta\gamma) w_{ji} + \eta x_i \left(\sum_{k \in \text{Downstream}(j)} w_{kj} (t_k - o_k) \right)
\end{aligned}$$

c. Analizând regulile de actualizare a ponderilor deduse la punctul b, se observă că [în ambele cazuri, input-to-hidden și hidden-to-output] ele sunt de forma $w \leftarrow (1 - 2\eta\gamma)w + \Delta w$. Deosebirea față de cazul clasic ($w \leftarrow w + \Delta w$), este evidentă. Întrucât algoritmul de retro-propagare inițializează ponderile w cu valori apropiate de 0 — iar în practică se folosesc valori mici pentru constantele η și γ —, lucrând cu varianta funcției de eroare propusă în această problemă vom obține valori ale ponderilor w mai aproape de 0 (decât în varianta clasică).

23. (De ce nu sunt convenabile funcțiile-prag pentru antrenarea perceptronilor și a rețelelor neuronale artificiale?)
CMU, 2011 spring, Tom Mitchell, HW5, pr. 3.4

Care sunt motivele pentru care în general funcțiile-prag nu convin pentru antrenare perceptronilor și a rețelelor neuronale artificiale [folosind metoda gradientului]?

Răspuns:

Răspunsul cel mai des citat, și anume că funcția-prag nu este derivabilă nu este în totalitate corect. Funcția modul ($|x|$) este folosită adeseori la aplicarea metodei gradientului, deși este derivabilă peste tot în afară de un singur punct (și anume, în 0), ca și funcția-prag. Un avantaj important al funcției modul este faptul că este continuă. Din contră, cel mai mare dezavantaj al funcției-treaptă este faptul că în orice punct în care este derivabilă, derivata sa este 0. Se știe că la aplicarea metodei gradientului descendente actualizarea parametrilor w se face după o regulă de forma $w \leftarrow w \pm \eta \nabla_w E(w)$. Prin urmare, în cazul în care derivatele parțiale ale funcției $E(w)$ sunt constant nule, atunci regula gradientului va lăsa parametrul w neschimbat. În consecință, funcția-prag nu poate servi la actualizarea valorilor parametrilor atunci când se folosește metoda gradientului.

24. (Rețele cu unități neuronale având funcții de activare liniare și / sau sigmoidale: corespondența cu suprafete de decizie liniare / neliniare)
CMU, 2008 fall, Eric Xing, midterm, pr. 4.1

Fie o rețea neuronală având două niveluri, cu două unități pe nivelul ascuns și o singură unitate pe nivelul de ieșire (output). Vom lucra cu *funcția de activare liniară* $y = C \cdot a \stackrel{\text{not.}}{=}$

$C \cdot \sum_i w_i x_i$, unde C este o constantă, iar a este suma ponderată a intrărilor și, de asemenea, cu funcția sigmoidală (sau, logistică): $y = \sigma(a) \stackrel{\text{not.}}{=} \frac{1}{1 + e^{-a}}$.

- Presupunem că toate unitățile sunt liniare. Poate oare rețeaua aceasta să genereze / determine granițe de decizie (engl., decision boundaries) pe care un *model de regresie* standard de forma $y = b_0 + b_1 x_1 + b_2 x_2 + \varepsilon$ nu le poate genera?
- Presupunem că unitățile ascunse ale rețelei noastre sunt de tip *sigmoidal*, iar unitatea de [pe nivelul de] ieșire este de tip *liniar*. Este oare posibil ca în acest caz rețeaua să genereze granițe de decizie neliniare?
- Folosind funcția de activare *sigmoidală* pentru toate unitățile (atât cele de pe nivelul ascuns cât și cea de pe nivelul de ieșire), este oare posibil să aproximăm suprafețe de decizie oricăr de complicate prin combinarea multor [porțiuni de] granițe de decizie neliniare? Ce schimbări ar trebui să operați asupra rețelei de mai sus pentru a putea approxima orice graniță de decizie?

Răspuns:

- Nu. Orică rețea formată doar din unități liniare poate fi redusă la un model liniar simplu.
- Da. Răspunsul se justifică imediat din punct de vedere matematic.
- Da; avem nevoie de unități ascunse adiționale. Atunci când folosim mai multe unități ascunse putem obține suprafețe de decizie mai complicate.

25.

(Adevărat sau Fals?)

a. *CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 6.b.5*

Regula gradientului descendente aplicată în regim “batch” pentru o unitate neuronală care are intrările x_1 și x_2 și ieșirea $w_0 + w_1(x_1 + 1) + w_2(x_2^2)$ produce:

$$\begin{aligned}\Delta w_0 &= \eta \sum_i (t_i - o_i) \\ \Delta w_1 &= \eta \sum_i [(t_i - o_i)x_{i,1} + (t_i - o_i)] \\ \Delta w_2 &= \eta \sum_i [(t_i - o_i)2x_{i,2}]\end{aligned}$$

unde:

- t_i este valoarea dorită (engl., target) pentru outputul unității neuronale pentru exemplul i ;
- o_i este outputul unității neuronale pentru exemplul i ;
- $x_{i,1}$ este valoarea intrării / atributului x_1 din exemplul i ;
- $x_{i,2}$ este valoarea intrării / atributului x_2 din exemplul i .

b. *CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 6.b.3*

Varianta incrementală (sau, stochastică) a metodei gradientului descendente se comportă întotdeauna mai bine decât varianta “batch”.

c. *CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 6.b.2*

Suprafața de eroare urmată (conform metodei gradientului descendente) de către algoritmul de retro-propagare se modifică dacă modificăm datele de antrenament.

d. *CMU, 2003 fall, T. Mitchell, A. Moore, final exam, pr. 7.e*

Perceptronul poate fi (dar nu neapărat este) capabil să obțină o mai bună performanță de clasificare dacă (anterior antrenării) intrările sale sunt mapate într-un „spațiu de trăsături” folosind o funcție-nucleu cu baza radială.

Răspuns:

a. Fals.

Valoarea funcției de ieșire pentru exemplul i este $o_i(w_0, w_1, w_2) = w_0 + w_1(x_{i,1} + 1) + w_2(x_{i,2}^2)$, iar funcția de eroare este $E(w_0, w_1, w_2) = \frac{1}{2} \sum_i (t_i - o_i)^2$. Stim că $w_j = w_j + \Delta w_j$, unde $\Delta w_j = -\eta \frac{\partial E}{\partial w_j}$. Este imediat că $\frac{\partial E}{\partial w_j} = \sum_i (t_i - o_i) \frac{\partial (t_i - o_i)}{\partial w_j}$. Calculând derivatele parțiale din această expresie, vom obține:

$$\frac{\partial (t_i - o_i)}{\partial w_0} = -1 \Rightarrow \frac{\partial E}{\partial w_0} = -\sum_i (t_i - o_i) \Rightarrow \Delta w_0 = \eta \sum_i (t_i - o_i);$$

$$\begin{aligned} \frac{\partial (t_i - o_i)}{\partial w_1} &= -(x_{i,1} + 1) \Rightarrow \frac{\partial E}{\partial w_1} = -\sum_i (t_i - o_i)(x_{i,1} + 1) \\ &\Rightarrow \Delta w_1 = \eta \sum_i (t_i - o_i)(x_{i,1} + 1); \end{aligned}$$

$$\frac{\partial (t_i - o_i)}{\partial w_2} = -x_{i,2}^2 \Rightarrow \frac{\partial E}{\partial w_2} = -\sum_i (t_i - o_i)x_{i,2}^2 \Rightarrow \Delta w_2 = \eta \sum_i (t_i - o_i)x_{i,2}^2.$$

Se observă că Δw_2 pe care tocmai l-am calculat mai sus diferă de cel dat în enunț, care este astăzi greșit.

b. Fals. Comparativ cu varianta “batch”, varianta incrementală a metodei gradientului descendente are *avantajul* de a evita unele optime (în spăță, minime) locale. Însă varianta incrementală este doar o aproximare a metodei “batch” (care, pentru perceptronul liniar, obține în mod garantat optimul (global)). În plus, atunci când cele două variante folosesc rate ale învățării egale, varianta incrementală este mai lentă decât varianta “batch”.

c. Adevărat. Algoritmul de retro-propagare aplică metoda gradientului descendente funcției de eroare, care poate fi de exemplu $\frac{1}{2} \sum_d (t_i - o_i)^2$. În calculul funcției de eroare intervin atât target-ul (eticheta) t_i cât și atributele instanțelor de antrenament (acestea din urmă prin intermediul outputului o_i). În consecință, orice schimbare în datele de antrenament ca atare — cât și extinderea sau reducerea setului de date de antrenament — poate afecta funcția de eroare și, în consecință, rezultatul algoritmului de retro-propagare.

d. Adevărat. Chestiunea aceasta se poate pune pentru orice clasificator care învăță un separator liniar. Maparea mulțimii de instanțe antrenament $S = \{x_1, x_2, \dots, x_l\} \subset \mathbb{R}^n$ folosind o funcție-nucleu cu baza radială sau, mai general, o funcție oarecare $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ cu $m > n$ poate conduce la găsirea unui separator liniar pentru mulțimea $\{\Phi(x_1), \Phi(x_2), \dots, \Phi(x_l)\}$ în \mathbb{R}^m , chiar dacă mulțimea S nu este liniar separabilă. Totuși, găsirea unui astfel de separator (în urma aplicării funcției Φ) nu este garantată.

În general, chiar dacă nu se obține separabilitate liniară în urma mapării cu funcția Φ , este posibil (dar nu obligatoriu) ca în \mathbb{R}^m să se obțină un optim mai convenabil pentru funcția de optimizat. În particular, pentru perceptron, ca funcție de optimizat se ia în mod obișnuit semisuma pătratelor erorilor. „Un optim mai convenabil“ se va traduce în practică în faptul că funcția învățată în \mathbb{R}^m se comportă mai bine pe date de test (prin comparație, eventual, cu funcția învățată în \mathbb{R}^n).

Observație: Funcțiile cu baza radială sunt un caz particular de funcții-nucleu. (A se vedea capitolul *Mașini cu vectori-suport.*) Funcțiile-nucleu prezintă avantajul că produsele scalare $\Phi(x) \cdot \Phi(y)$ pot fi calculate în mod eficient, chiar dacă m este mult mai mare decât n . Această proprietate este foarte convenabilă pentru implementarea mașinilor cu vectori-suport.

26.

(Rețele neuronale profunde:
dispariția – ori, dimpotrivă, „explozia“ – vectorului gradient)
 ■ CMU, 2015 fall, E. Xing, Z. Bar-Joseph, HW3, pr. 1.3

În acest exercițiu veți studia anumite dificultăți care apar la aplicarea algoritmului de retro-propagare pentru antrenarea rețelelor neuronale profunde (engl., deep neural networks).⁴⁰⁶ Pentru conveniență, vom considera cea mai simplă rețea neuronală profundă, și anume una în care există câte un singur neuron pe fiecare nivel, iar outputul produs de neuronul de pe nivelul j este

$$z_j = f(\text{net}_j) \text{ cu } \text{net}_j \stackrel{\text{not.}}{=} \begin{cases} b_1 + w_{11}x_1 + \dots + w_{1d}x_d & \text{pentru } j = 1; \\ b_j + w_j z_{j-1} & \text{pentru } j = 2, \dots, m, \end{cases} \quad (158)$$

unde

- f este o anumită funcție de activare, a cărei derivată în punctul x este $f'(x)$,
- m este numărul de niveluri din această rețea neuronală,
- b_j este bias-ul (adică, termenul liber) al unității neuronale de pe nivelul j .

Vom nota cu L funcția de eroare / cost / pierdere (engl., loss function) care este folosită ca funcție obiectiv la antrenare.

a. Calculați derivata parțială a funcției L în raport cu b_1 , bias-ul (adică, termenul liber) pentru neuronul de pe primul nivel.

Indicație: Pentru acest scop, nu este necesar să precizăm cine anume este funcția L și, la fel, cine este funcția f . Presupunând doar că L se exprimă în funcție de f și este derivabilă, veți exprima $\frac{\partial L}{\partial b_1}$ în funcție de $\text{net}_1, \dots, \text{net}_m$ și w_2, \dots, w_m . Veți folosi regula pentru derivarea compunerilor de funcții: $(f_1(f_2(x)))' = f'_1(f_2(x)) \cdot f'_2(x)$.

b. Presupunem că funcția de activare f este obișnuita funcție sigmoidală, $\sigma(x) = 1/(1 + \exp(-x))$ și că ponderile w sunt inițializate astfel încât $|w_j| < 1$ pentru $j = 1, \dots, m$.

Explicați de ce derivata parțială de la punctul precedent, $\frac{\partial L}{\partial b_1}$, tinde la 0 atunci când m are valori mari. *Comentariu:* Din acest motiv, se spune că — în astfel de condiții — avem de a face cu „dispariția“ gradientului (engl., *gradient vanishing*).

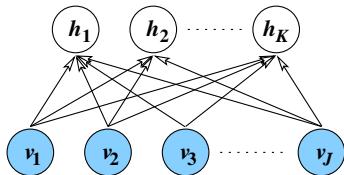
⁴⁰⁶Se spune că o rețea neuronală este profundă dacă are cel puțin două niveluri ascunse.

Explicați de ce chiar și atunci când valoarea lui $|w|$ este mare, derivata parțială menționată mai sus tinde la 0 (deci, tinde să dispară), nu tinde la infinit (adică, să „explodeze“).

c. Una dintre modalitățile propuse pentru rezolvarea (parțială) a problemei „dispariției“ gradientului este ca în locul funcției sigmoidale să se folosească pentru activare *funcția liniar-rectificată* (engl., rectified linear function), notată cu ReL. Funcția de activare ReL este definită prin $\max\{0, x\}$. Explicați de ce funcția ReL ne poate ajuta să evităm fenomenul de „dispariție“ a gradientului.

d. O altă modalitate de rezolvare (parțială) a problemei „dispariției“ ori „exploziei“ gradientului este *pre-antrenarea nivel-cu-nivel* (engl., layer-wise pre-training).

Modelul *mașinii Boltzmann restricționate* (engl., Restricted Boltzmann machine), abreviat RBM, este unul dintre modelele cele mai des folosite pentru pre-antrenarea [unei rețele neuronale] nivel-cu-nivel. Figura alăturată prezintă un exemplu de RBM conținând K unități ascunse (h_1, \dots, h_K) și J intrări (v_1, \dots, v_J).



În vederea antrenării unei RBM, definim [ca funcție obiectiv] *distribuția probabilistă comună* — peste vectorii de valori posibile $\bar{v} \stackrel{\text{not.}}{=} (v_1, \dots, v_J)$ și $\bar{h} \stackrel{\text{not.}}{=} (h_1, \dots, h_K)$ — având forma [generală] următoare:

$$P(\bar{v}, \bar{h}) = \frac{1}{Z} \exp \left(\sum_i \theta_i \phi_i(\bar{v}, \bar{h}) \right),$$

unde

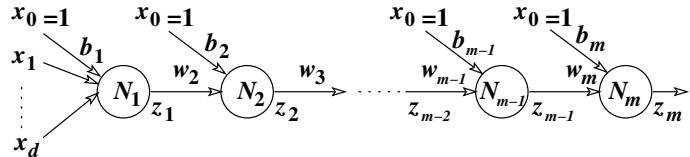
- $Z = \sum_{\bar{v}', \bar{h}'} \exp \left(\sum_i \theta_i \phi_i(\bar{v}', \bar{h}') \right)$ este constanta de normalizare;
- $\phi_i(\bar{v}, \bar{h})$ sunt anumite trăsături / attribute;
- θ_i sunt parametrii care corespund ponderilor din RBM.

Considerând că pentru antrenarea unei RBM se folosește metoda gradientului descent, arătați că expresia derivatei patiale a logaritmului funcției de probabilitate marginală $P(\bar{v})$ în raport cu parametrul / ponderea θ_i este:

$$\frac{\partial \ln P(\bar{v})}{\partial \theta_i} = \sum_{\bar{h}} \phi_i(\bar{v}, \bar{h}) P(\bar{h}|\bar{v}) - \sum_{\bar{v}', \bar{h}'} \phi_i(\bar{v}', \bar{h}') P(\bar{v}', \bar{h}').$$

Răspuns:

a. Ilustrăm mai jos rețeaua neuronală profundă definită în enunț.



Apoi vom scrie în mod desfășurat relațiile (158):

$$\begin{aligned} z_1 &= f(\text{net}_1) = f(b_1 + w_{11}x_1 + \dots + w_{1d}x_d) \\ z_2 &= f(\text{net}_2) = f(b_2 + w_2 z_1) \end{aligned}$$

...

$$o = z_m = f(\text{net}_m) = f(b_m + w_m z_{m-1})$$

Funcția de pierdere L se va exprima astfel:

$$\begin{aligned}
 & L(w_{11}, \dots, w_{1d}, w_2, \dots, w_m, b_1, \dots, b_m) \\
 & \stackrel{\text{not.}}{=} L(f(\underbrace{\text{net}_m}_{b_m + w_m z_{m-1}})) \\
 & = L(f(b_m + w_m f(\underbrace{\text{net}_{m-1}}_{b_{m-1} + w_{m-1} z_{m-2}}))) \\
 & \quad \dots \\
 & = L(f(b_m + w_m f(b_{m-1} + w_{m-1} f(b_{m-2} + \dots + w_3 f(\underbrace{\text{net}_2}_{b_2 + w_2 z_1} \dots)))) \\
 & = L(f(b_m + w_m f(b_{m-1} + w_{m-1} f(b_{m-2} + \dots + w_3 f(b_2 + w_2 z_1) \dots))) \\
 & = L(f(b_m + w_m f(b_{m-1} + w_{m-1} f(b_{m-2} + \dots + \\
 & \quad w_3 f(b_2 + w_2 f(\underbrace{\text{net}_1}_{b_1 + w_{11} x_1 + \dots + w_{1d} x_d} \dots))))
 \end{aligned}$$

Pentru a deriva funcția L în raport cu b_1 , ne vom aminti mai întâi regula de derivare a unei compuneri multiple de funcții $f_1(f_2(\dots f_n(x) \dots))$. Pe lângă forma clasică pe care o stim din liceu, $f'_1(f_2(\dots f_n(x) \dots)) \cdot f'_2(\dots f_n(x) \dots) \cdot f'_n(x)$, ea se poate scrie sub forma aşa-numitei *reguli de înlățuire*: $\frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_3} \cdots \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial x}$.

Adoptând / aplicând această regulă pentru a calcula derivata parțială cerută în enunț, vom avea:

$$\begin{aligned}
 \frac{\partial L}{\partial b_1} &= \frac{\partial L}{\partial \text{net}_m} \cdot \frac{\partial \text{net}_m}{\partial \text{net}_{m-1}} \cdots \frac{\partial \text{net}_2}{\partial \text{net}_1} \cdot \frac{\partial \text{net}_1}{\partial b_1} \\
 &= L'(f(\text{net}_m)) \cdot f'(\text{net}_m) \cdot \\
 &\quad w_m \cdot f'(\text{net}_{m-1}) \cdot \\
 &\quad w_{m-1} \cdot f'(\text{net}_{m-2}) \cdot \\
 &\quad \dots \\
 &\quad w_2 \cdot f'(\text{net}_1) \cdot \\
 &\quad 1 \\
 &= L'(f(\text{net}_m)) \cdot f'(\text{net}_1) \cdot \prod_{k=2}^m (f'(\text{net}_k) \cdot w_k)
 \end{aligned}$$

b. Stim că $\sigma(x) \in (0, 1)$ și $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ pentru orice $x \in \mathbb{R}$, iar maximul funcției $z(1 - z)$ este $1/4$ și se atinge în punctul $z = 1/2$. Prin urmare, $|\prod_{k=2}^m \sigma'(\text{net}_k)| \leq \left(\frac{1}{4}\right)^{m-1}$. Întrucât $|w_k| < 1$ pentru orice k (conform ipotezei din enunț), rezultă imediat că $\left|\frac{\partial L}{\partial b_1}\right| = |L'(\sigma(\text{net}_m)) \cdot \sigma'(\text{net}_1)| \cdot \prod_{k=2}^m |\sigma'(\text{net}_k) \cdot w_k|$ tinde la 0 pentru $k \rightarrow +\infty$ și pentru orice input \bar{x} fixat.

Pentru a contracara acest fenomen de „dispariție“ a gradientului, ar trebui să impunem condiții de forma $|w_k \sigma'(\text{net}_k)| > 1$. Trebuie însă să remarcăm faptul că $\sigma(\text{net}_k)$ depinde de w_k : $\sigma(\text{net}_k) = \sigma(b_k + w_k z_{k-1})$. În consecință, dacă îi vom da lui w_k posibilitatea să ia valori mari în modul, atunci $z_k \stackrel{\text{not.}}{=} b_k + w_k z_{k-1}$ va lua de asemenea valori mari în modul, iar $\sigma(z_k)$ va tinde fie la 0 fie la 1, deci $\sigma'(z_k) = \sigma(z_k)(1 - \sigma(z_k))$ va tinde la 0. În sfârșit,

se poate verifica în mod riguros că $\lim_{z \rightarrow \pm\infty} z\sigma'(z) = 0$ (lăsăm această demonstrație ca exercițiu cititorului).

c. Dacă f este funcția de activare ReL, atunci $f'(x) = 0$ pentru $x < 0$ și $f'(x) = 1$ pentru $x > 0$. Prin urmare, folosind această funcție putem împiedica „dispariția“ gradientului.

d. Pornind de la expresia distribuției probabiliste $P(\bar{v}, \bar{h})$ care a fost dată în enunț, explicitând mai întâi constanta de normalizare Z , vom putea scrie apoi expresia distribuției marginale $P(\bar{v})$:

$$\begin{aligned} P(\bar{v}, \bar{h}) &= \frac{1}{\sum_{\bar{v}', \bar{h}'} \exp(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}'))} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right) \\ \Rightarrow P(\bar{v}) &= \frac{1}{\sum_{\bar{v}', \bar{h}'} \exp(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}'))} \sum_{\bar{h}} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right) \end{aligned}$$

Prin urmare,

$$\ln P(\bar{v}) = \ln \sum_{\bar{h}} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right) - \ln \sum_{\bar{v}', \bar{h}'} \exp(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}')).$$

În consecință, derivata parțială a funcției $\ln P(\bar{v})$ în raport cu parametrul θ_i se poate calcula astfel:

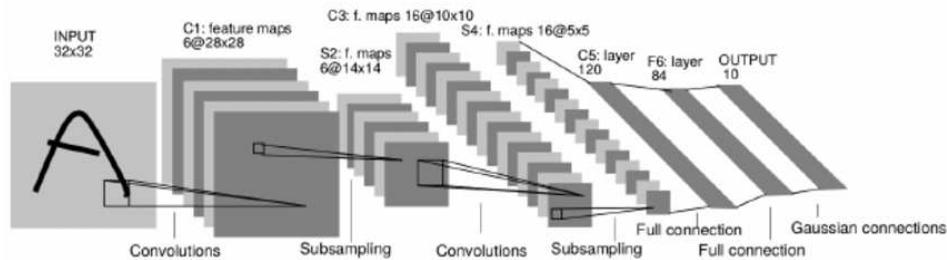
$$\begin{aligned} \frac{\partial \ln P(\bar{v})}{\partial \theta_i} &= \\ &= \frac{\sum_{\bar{h}} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right) \cdot \phi_i(\bar{v}, \bar{h})}{\sum_{\bar{h}} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right)} - \underbrace{\frac{\sum_{\bar{v}', \bar{h}'} \exp\left(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}')\right) \cdot \phi_i(\bar{v}, \bar{h})}{\sum_{\bar{v}', \bar{h}'} \exp\left(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}')\right)}}_Z \\ &= \frac{\sum_{\bar{h}} \frac{1}{Z} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right) \cdot \phi_i(\bar{v}, \bar{h})}{\sum_{\bar{h}} \frac{1}{Z} \exp\left(\sum_k \theta_k \phi_k(\bar{v}, \bar{h})\right)} - \sum_{\bar{v}', \bar{h}'} \frac{1}{Z} \exp\left(\sum_k \theta_k \phi_k(\bar{v}', \bar{h}')\right) \cdot \phi_i(\bar{v}, \bar{h}) \\ &= \frac{\sum_{\bar{h}} P(\bar{v}, \bar{h}) \cdot \phi_i(\bar{v}, \bar{h})}{P(\bar{v})} - \sum_{\bar{v}', \bar{h}'} P(\bar{v}', \bar{h}') \phi_i(\bar{v}', \bar{h}') \\ &= \sum_{\bar{h}} \frac{P(\bar{v}, \bar{h})}{P(\bar{v})} \cdot \phi_i(\bar{v}, \bar{h}) - \sum_{\bar{v}', \bar{h}'} P(\bar{v}', \bar{h}') \phi_i(\bar{v}', \bar{h}') \\ &= \sum_{\bar{h}} P(\bar{h}|\bar{v}) \cdot \phi_i(\bar{v}, \bar{h}) - \sum_{\bar{v}', \bar{h}'} P(\bar{v}', \bar{h}') \phi_i(\bar{v}', \bar{h}') \end{aligned}$$

27.

(Determinarea numărului de parametri și de conexiuni din rețea neuronală convolutivă LeNet)

prelucrare de Liviu Ciortuz, după CMU, 2015 fall, E. Xing, Z. Bar-Joseph, HW3, pr. 1.2.1

Ca să puteți rezolva acest exercițiu, trebuie să fiți deja familiari cu modelul de rețea neuronală convolutivă LeNet, a cărei reprezentare grafică o reproducem mai jos.⁴⁰⁷



Vă cerem să determinați numărul total de parametri și numărul total de conexiuni din această rețea. Câtă parametri și câte conexiuni sunt în fiecare dintre nivelurile convective și de sub-selectie, C1, S2, C3 și S4? Câtă parametri sunt în fiecare dintre nivelurile total-conectate (engl., fully-connected layers), C5, F6 și OUTPUT?

Precizări:

i. Mărimea *filtrului* pentru fiecare dintre nivelurile convective și nivelurile de sub-selectie (engl., sub-sampling layers):

- C1: 5×5 , adică, fiecare unitate neuronală din C1 are un *spațiu de recepție* (engl., receptive field) de 5×5 în nivelul precedent lui;
- S2: 2×2 ;
- C3: 5×5 ;
- S4: 2×2 .

ii. Spre deosebire de nivelul C1, care este total conectat cu nivelul precedent (INPUT), nivelul C3 nu este total conectat cu nivelul S2. Tabelul de mai jos arată care dintre hărțile de trăsături (sau, *planele*) de pe nivelul S2 sunt conectate cu (toate!) planele de pe nivelul C3.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3	X	X	X				X	X	X	X			X		X	X
4		X	X	X				X	X	X	X		X	X		X
5			X	X	X			X	X	X	X		X	X	X	X

Răspuns:

Vom prezenta o sinteză a calculelor, precum și răspunsurile finale, în tabelul următor:

⁴⁰⁷Vă recomandăm să citiți secțiunile A și B din articolul *Gradient-based learning applied to document recognition*, de Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Proceedings of the IEEE, November 1998, 86 (11), p. 2278-2324. De asemenea, puteți vedea următorul exercițiu de tip implementare: CMU, 2016 spring, W. Cohen, N. Balcan, HW7 (sau CMU, 2016 fall, N. Balcan, M. Gormley, HW6).

	nr. parametri	nr. conexiuni
ConvLayers:		
C1	$(5 * 5 + 1) * 6 = 156$	$28 * 28 * 26 * 6 = 122304$
S2	$2 * 6 = 12$	$14 * 14 * 6 * 5 = 5880$
C3	$(5 * 5 * 3 + 1) * 6 +$ $(5 * 5 * 4 + 1) * 9 +$ $(5 * 5 * 6 + 1) = 1516$	$1516 * 10 * 10 = 151600$
S4	$2 * 16 = 32$	$5 * 5 * 16 * 16 = 2000$
FCLayers:		
C5	$(5 * 5 * 16 + 1) * 120 = 48120$	48120
F6	$(120 + 1) * 84 = 10164$	10164
OUTPUT	$(84 + 1) * 10 = 850$	850
Total	60850	340918

6.2 Probleme propuse

6.2.1 Chestiuni introductive

28. (Perceptronul-prag; funcția calculată; zone de decizie)
CMU, 1997 fall, T. Mitchell, A. Moore, midterm exam, pr. 3.1

Considerăm un perceptron-prag cu intrările x_1 și x_2 și având ponderile $w_0 = 0$, $w_1 = 1$ și $w_2 = -3$. Desenați granița de decizie a acestui perceptron și indicați zonele din planul bidimensional în care perceptronul produce rezultatele +1 și -1.

29. (Perceptronul-prag; exemplificare pentru disjuncții / conjuncții generalizate)
prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Ziv Bar-Joseph, HW3, pr. 1.1

Imaginează-ți că ești responsabil cu design-ul unor preceptroni de tip prag și că trebuie să implementezi câteva funcții logice.

- a. Concepă un perceptron-prag care implementează operatorul logic AND. Se vor considera două intrări, x_1 și $x_2 \in \{-1, +1\}$, și ieșirea $y \in \{-1, +1\}$.

Scrie mai întâi *tabela de adevăr* a operatorului AND.

În planul bidimensional al intrărilor x_1 și x_2 , desenează *separatorul* (adică granița de decizie) a perceptronului. Determină *ecuația* $f(x_1, x_2) = 0$ corespunzătoare acestei separator. Verifică semenele funcției f pe *zonele de decizie* determinate de separator.

Desenează perceptronul-prag AND specificând ponderile w_1 , w_2 și w_0 .

Extinde rezultatul precedent la

- conjuncții de n variabile (A_1 AND A_2 AND ... AND A_n);

- conjuncții de n literali ($l_1 \text{ AND } l_2 \text{ AND } \dots \text{ AND } l_n$) unde l_i este fie A_i fie $\neg A_i$.
- b. Procedează similar pentru funcția logică OR.
30. (Perceptroni-prag și rețele de astfel de perceptroni:
exemplificare)
- TU Dresden, 2006 summer, S. Hölldobler, A. Grossmann, HW3, pr. 2*

Fie următoarele instanțe de antrenament din \mathbb{R}^2 :

Exemplu	X	Y	Clasa
1	1	6	–
2	1	2	+
3	2	4	+
4	5	2	–
5	6	5	–
6	7	8	–

- a. Indicați fie un perceptron cu funcție de activare de tip prag fie o rețea formată din astfel de perceptroni — alegeti varianta cea mai simplă! — care să fie consistentă cu aceste exemple. (Nu trebuie să apelați la niciun algoritm de învățare automată!)
- b. Îndepliniți aceleași cerințe ca mai sus după ce în prealabil ati adăugat la tabelul dat și exemplul următor:

Exemplu	X	Y	Clasa
7	7	1	+

31. (O proprietate:
extensia funcției XOR pe \mathbb{R}^2 (vedeți pr. 1.d)
nu poate fi calculată de rețele de perceptroni-prag
care au un singur nivel ascuns)
- prelucrare de Liviu Ciortuz, după
CMU, 2010 fall, Aarti Singh, HW5, pr. 4.1.2*

În rezolvarea problemei 1.d s-a arătat că funcția $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ definită prin $f(x_1, x_2) = 1$ pentru $x_1, x_2 \geq 0$ sau $x_1, x_2 < 0$, și -1 în rest este calculabilă de către o rețea neuronală cu *două niveluri ascunse*, folosind doar unități de tip prag.

Demonstrați că nu există nicio rețea cu *un singur nivel ascuns*, constituită doar din astfel de unități, care să calculeze funcția f . Veți presupune că numărul de unități de pe nivelul ascuns poate fi oricât de mare, însă finit.

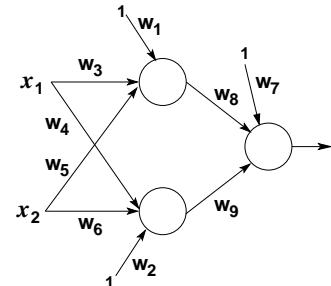
Indicație: Întrucât rețeaua are un singur nivel ascuns, fiecare unitate de pe acest nivel poate fi pusă în corespondență cu o dreaptă din planul bidimensional, iar outputul calculat de către respectiva unitate este +1 pentru punctele (x_1, x_2) situate de o parte a dreptei și -1 pentru punctele de pe celălaltă parte. Considerăm o vecinătate [de formă circulară] a originii reperului de coordinate, astfel încât dacă dreapta (adică, separatorul, sau granița de separare) asociată unei unități ascunse (oricare dintre acestea!) intersectează această vecinătate atunci ea trece (în mod necesar) prin origine. Este oare posibil ca o rețea neuronală [alcătuită doar din unități-prag și] având un singur nivel ascuns să reprezinte funcția f în această vecinătate?

32.

(Rețele neuronale; o proprietate de bază:
unitățile liniare de pe nivelurile ascunse pot fi „absorbite“
pe nivelul următor)

CMU, ? spring, ML course 10-701, midterm, pr. 4

Considerăm rețeaua neuronală pentru clasificare binară din figura alăturată. Pentru unitățile ascunse folosim o funcție liniară de activare $h(z) = cz$, iar pentru unitatea de pe nivelul de ieșire funcția de activare sigmoidală $\sigma(z) = \frac{1}{1 + e^{-z}}$.



- a. Desenați o rețea neuronală care este echivalentă cu rețeaua dată, dar care nu are niciun nivel ascuns. Scrieți ponderile \tilde{w} ale acestei noi rețele în funcție de c și de w_i .
- b. Este adevărat că orice rețea neuronală de tip feed-forward, cu mai multe niveluri și având toți neuronii situați pe nivelurile ascunse de tip unități liniare poate fi reprezentată ca o rețea neuronală fără niciun nivel ascuns? Explicați succint răspunsul dat.

33.

(Alegerea ponderilor pentru
o rețea cu structură / compozitie specificată,
în aşa fel încât să codifice funcția XOR)

CMU, 2011 spring, Tom Mitchell, HW5, pr. 3.3

Fie o rețea neuronală, constând dintr-un nivel ascuns format din două unități sigmoidale, și un nivel de ieșire pe care se află o singură unitate, de tip prag. La niciuna dintre aceste trei unități nu se folosește termenul liber $x_0 = 1$. Găsiți valori pentru ponderile din rețea astfel încât ea să codifice conceptul logic XOR.

Indicație: Întrucât unitățile de pe nivelul ascuns sunt de tip sigmoidal, se va considera că intrările x_1 și x_2 iau valorile 0 și / sau 1.

Observație: În raport cu problema 2.b, aici la toți neuronii s-a eliminat termenul liber $x_0 = 1$ (ca și la problema 4), iar tipul funcției de activare, și anume prag, a fost înlocuit cu cel sigmoidal. Din cele trei probleme se observă că astfel de modificări pot afecta capacitatea de reprezentare a unei rețele neuronale (chiar dacă topologia de ansamblu este aceeași în toate cele trei cazuri).

6.2.2 Unități neuronale — algoritmi de antrenare

34.

(Antrenarea pereceptronului liniar: deducerea regulii de actualizare)

Liviu Ciortuz, 2017, urmând

■ Tom Mitchell, Machine Learning, 1997, p. 89-93

În acest exercițiu vi se cere mai întâi să elaborați / redați partea de fundamentare teoretică pentru antrenarea perceptronului liniar, date fiind instanțele $(\bar{x}_1, t_1), \dots, (\bar{x}_n, t_n)$, cu $\bar{x}_i \in \mathbb{R}^d$ și $t_i \in \mathbb{R}$ pentru $i = 1, \dots, n$.

Așadar, veți considera perceptronul liniar având intrările $x_0 = 1, x_1, \dots, x_d \in \mathbb{R}$ și ponderile $w_0, w_1, \dots, w_d \in \mathbb{R}$.

a. În linie cu *metoda gradientului descendente*, deduceți care este forma *regulii de actualizare* a vectorului de ponderi $\bar{w} \in \mathbb{R}^{d+1}$ (sau, echivalent, forma regulilor de actualizare a ponderilor w_i , cu $i = 0, \dots, d$) pentru găsirea acelei valori care minimizează semisuma pătratelor erorilor comise de perceptron,⁴⁰⁸ adică

$$E(\bar{w}) \stackrel{\text{def.}}{=} \frac{1}{2} \sum_{i=1}^n (t_i - o_i)^2.$$

În această ultimă expresie, o_i este outputul perceptronului liniar pentru intrarea $\bar{x}_i \stackrel{\text{renot.}}{=} (x_0 = 1, x_{i,1}, \dots, x_{i,d})$, adică $o_i = w_0 + \sum_{j=1}^d w_j x_{i,j}$.

Vă reamintim că la aplicarea metodei gradientului descendente pentru găsirea unui punct de minim al unei funcții derivabile $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}$, regula de actualizare a parametrilor \bar{w} ai funcției f are forma $\bar{w} \leftarrow \bar{w} + \Delta \bar{w}$, cu $\Delta \bar{w} \stackrel{\text{def.}}{=} -\eta \nabla_{\bar{w}} f(\bar{w})$, unde $\eta > 0$ este un număr real mic, numit *rata de învățare*, iar $\nabla_{\bar{w}} f(\bar{w})$ este vectorul *gradient*, adică $\left(\frac{\partial}{\partial w_0} f(\bar{w}), \frac{\partial}{\partial w_1} f(\bar{w}), \dots, \frac{\partial}{\partial w_d} f(\bar{w}) \right)$.

b. Dacă în locul perceptronului liniar vom considera perceptronul-prag, respectiv perceptronul sigmoidal, prin ce va difera forma regulii de actualizare a ponderilor față de rezultatul de la punctul a? (Pentru perceptronul-prag, se va presupune că $t_i \in \{-1, +1\}$, în vreme ce pentru perceptronul sigmoidal vom considera $t_i \in \{0, 1\}$, pentru $i = 1, \dots, n$.)

c. Elaborați algoritmul de antrenare a perceptronului liniar (și apoi, dacă doriti, și a celui sigmoidal, dar veți preciza doar diferențele!). La *initializare*, ponderile w_i vor primi ca valori numere reale mici. În ce privește *condiția de oprire* a algoritmului, se poate opta pentru una din următoarele variante (eventual combinate):

- toate instanțele de antrenament sunt corect clasificate (în ipoteza că datele \bar{x}_i , cu $i = 1, \dots, n$, sunt liniar separabile);
- efectuarea unui număr prestabilit de iterații;
- verificarea condiției $|E(\bar{w}^{(t+1)}) - E(\bar{w}^{(t)})| < \varepsilon$, unde pragul numeric $\varepsilon > 0$ este stabilit inițial, iar $E(\bar{w}^{(t)})$ este valoarea *funcției de eroare* E (i.e., ceea ce mai sus am numit semisuma pătratelor erorilor) pe setul de date de antrenament, la finalul iterației t .

d. Ce cunoașteți (de la curs) despre convergența algoritmului de la punctul c?

⁴⁰⁸Conform problemei 13.b, am putea scrie — în ipoteza că datele au fost generate probabilist, cu o componentă „zgomot“ urmând o distribuție gaussiană unidimensională de medie 0, aşa cum s-a precizat acolo —,

$$\bar{w}_{\text{ML}} = \arg \min_{\bar{w}} E(\bar{w}) = \arg \min_{\bar{w}} \frac{1}{2} \sum_{i=1}^n (t_i - \bar{w} \cdot \bar{x}_i)^2.$$

35. (Deducerea regulii de actualizare a ponderilor pentru un tip particular de perceptron)
CMU, 1997 fall, Tom Mitchell, midterm, pr. 3.d

Se consideră o variantă particulară a perceptronului, [fără funcție de activare,] la care ieșirea o depinde de intrările x_i astfel:

$$o = w_0 + w_1x_1 + w_1x_1^3 + w_2x_2 + w_2x_2^3 + \dots + w_nx_n + w_nx_n^3$$

Derivați un algoritm de antrenare bazat pe metoda descreșterii gradientului, care să minimizeze suma pătratelor erorilor pentru acest tip de perceptron. Dați răspunsul sub forma

$$w_i \leftarrow w_i + \dots \quad \text{pentru } 1 \leq i \leq n.$$

36. (Algoritmul *Perceptron* (al lui Rosenblatt): aplicare pe date din \mathbb{R}^2 , folosind termen liber / bias)
CMU, 2015 spring, Alex Smola, midterm, pr. 4

Considerăm următoarele exemple de antrenament $(x, y) \in \mathbb{R}^2 \times \{\pm 1\}$, în ordinea indicată:

exemplul	1	2	3	4	5	6	7	8
instanța (x_1, x_2)	(10, 10)	(0, 0)	(8, 4)	(3, 3)	(4, 8)	(0.5, 0.5)	(4, 3)	(2, 5)
eticheta y	+1	-1	+1	-1	+1	-1	+1	+1

Aplicați algoritmul *Perceptron* (al lui Rosenblatt) pe această succesiune de exemple. La initializare, vectorul de ponderi va primi valoarea $w = (1, 1)$, iar bias-ul (adică, termenul liber) va fi $b = 0$.⁴⁰⁹

37. (Învățare online cu perceptronul Rosenblatt: exemplificare pentru clasificarea de texte)
CMU, 2008 fall, Eric Xing, midterm exam, pr. 3.2

Vă reamintim că la învățarea de tip online (vedeți problema 38), la fiecare iterație

- supervisorul (desemnat în continuare cu A) îi transmite sistemului de învățare (B) o instanță x_t , fără a-i indica și eticheta y_t ;
- B prezice clasa \hat{y}_t pentru x_t ;
- în cazul în care $\hat{y}_t \neq y_t$, supervisorul A semnalează „eroare“ și îi transmite lui B eticheta corectă y_t .

Considerăm un set de date de antrenament notat cu $D = \{(x_1, y_1), \dots, (x_T, y_T)\}$, unde

- i. fiecare variabilă x_i reprezintă un document (d_i) în limba engleză, în care nu există mai mult de 100 de cuvinte (pentru această problemă, vom considera că nu există mai mult de 100 000 de cuvinte engleză);

⁴⁰⁹[LC:] Initializarea cu 0 a bias-ului nu înseamnă că el va păstra această valoare pe parcursul întregii execuții a algoritmului. Atenție: faptul că se folosește bias înseamnă că va trebui să considerați în mod implicit (sau, să adăugați explicit) componenta $x_0 = 1$ la fiecare instanță de antrenament!

- ii.* fiecăruia x_i îi este asociat un vector „rar“ (engl., sparse) $b_1, \dots, b_{100\,000}$, unde b_w este 1 în cazul în care cuvântul w — mai exact, cuvântul cu indicele w din dicționarul / lista cuvintelor din limba engleză — este prezent în documentul d_i , și 0 în caz contrar;
- iii.* $y_i \in \{+1, -1\}$ este eticheta care desemnează clasa documentului d_i ;
- iv.* există un vector w^* astfel încât $y_i w^* \cdot x_i > 1$ pentru orice document d_i .

Indicați căte o *margine superioară* pentru

- a. R — distanța maximă dintre originea sistemului de coordonate din $\mathbb{R}^{100\,000}$ și orice exemplu x_i ;
- b. m — numărul de erori făcute de un perceptron-prag înainte de a converge la o ipoteză corectă atunci când se face antrenarea pe datele din mulțimea D în regim de învățare online.

Observație: Se poate considera $\|w^*\| = 1$.

38.

(Învățare online cu perceptronul Rosenblatt)
prelucrare de Liviu Ciortuz, 2014, după CMU, 2008 spring, T. Mitchell, W. Cohen, HW2, pr. 4

Problema 18 poate fi ușor reformulată în termenii învățării automate online.⁴¹⁰ În această situație, instanțele de antrenament nu sunt furnizate în avans perceptronului-prag. În schimb — după ce vectorul de ponderi w a fost inițializat cu valori arbitrară, sau cu 0 ca în problema 18 —, la fiecare iterație (k) al algoritmului de antrenare, supervisorul prezintă sistemului de învățare automată căte o instanță de antrenament x_{t_k} , fără a divulga și eticheta y_{t_k} . După ce a primit vectorul x_{t_k} , perceptronul calculează ieșirea o_{t_k} (în funcție de $w^{(k)}$), iar supervisorul îi comunică eticheta y_{t_k} . Dacă $y_{t_k} \neq o_{t_k}$, perceptronul își actualizează / modifică vectorul de ponderi w .

Dacă presupunem că instanțele de antrenament respectă condițiile specificate în problema 18 și ne punem întrebarea *care este numărul maxim de greșeli pe care le poate face perceptronul în regim online înainte de a „converge“*, răspunsul va fi: $\left\lfloor \left(R \frac{\|w^*\|}{\gamma} \right)^2 \right\rfloor$.

Demonstrația este practic aceeași cu cea de la problema 18.⁴¹¹

Acum vă cerem să concepeți un algoritm care produce o secvență de exemple care forțează perceptronul să producă un sir de lungime arbitrară (m) de greșeli, dacă oricare dintre cele două presupuneri de mai jos este eliminată, dar datele de antrenament rămân în continuare separabile:⁴¹²

- instanțele de antrenament sunt liniar-separabile (de către un vector w^*), cu marginea $\gamma > 0$;⁴¹³
- toate instanțele de antrenament sunt situate într-o sferă cu raza R , cu centrul în originea sistemului de coordonate din \mathbb{R}^d .⁴¹⁴

⁴¹⁰Toate metodele de învățare prezentate până acum sunt modele off-line, adică nu permit / prevăd nicio interacțiune a supervisorului cu algoritmul de antrenare.

⁴¹¹Acesta este motivul pentru care am specificat acolo un număr posibil infinit de instanțe de antrenament x_1, \dots, x_n, \dots

⁴¹²Adică, $y_i w^* \cdot x_i > 0$.

⁴¹³Negarea acestei condiții înseamnă că pentru orice $\gamma > 0$ există o valoare a lui i astfel încât $0 < y_i w^* \cdot x_i < \gamma$.

⁴¹⁴Negarea acestei condiții înseamnă că pentru orice $R > 0$ există o valoare a lui i astfel încât $\|x_i\| > R$.

39. (Convergența algoritmului de antrenare a perceptronului-prag
comparație cu varianta Rosenblatt;
perceptronul kernel-izat dual)

CMU, 2013 spring, A. Smola, B. Poczos, HW2, pr. 2.a-c

Fie o secvență formată din n instanțe de antrenament $x_i \in \mathbb{R}^d$, împreună cu etichetele corespunzătoare $y_i \in \{-1, 1\}$. Vă readucem aminte⁴¹⁵ că algoritmul lui Rosenblatt pentru antrenarea unui perceptron-prag — în cele ce urmează vom folosi pentru acest algoritm numele de *Perceptron* — procedează astfel:

```

initialize  $w \leftarrow 0$ 
for  $i = 1, \dots, n$  do
    if  $y_i w \cdot x_i \leq 0$  then
         $w \leftarrow w + y_i x_i$ 
    end if
end for

```

Pentru simplitate, am inclus termenul liber (sau, bias-ul, de la engl., “bias”) w_0 în vectorul w , adică am renosat cu w vectorul $[w_0, w]$ și, similar, cu x_i vectorul $[1, x_i]$. Așadar, în această problemă nu este necesar să tratăm în mod explicit termenul liber.

Presupunem că $\|x_i\| \leq R$ pentru orice i . La problema 18 am demonstrat că atunci când există un vector de ponderi w^* astfel încât $\|w^*\| = 1$ și pentru orice i are loc inegalitatea

$$y_i w^* \cdot x_i \geq \gamma,$$

numărul de actualizări ale lui w este mărginit superior de

$$R^2 / \gamma^2. \quad (159)$$

a. Stim că regula [mai] generală pentru actualizarea perceptronului-prag este $w \leftarrow w + 2\eta y_i x_i$, unde $\eta > 0$ este rata de învățare.⁴¹⁶ Așadar, algoritmul *Perceptron* al lui Rosenblatt corespunde cazului special $\eta = 1/2$.

Găsiți o margine superioară (engl., upper bound) pentru numărul de actualizări ale perceptronului-prag, similar cu modul în care a fost obținută marginea (159). Cum variază această margine în funcție de rata η ?

b. Din relația (159) stim că pentru valori mici ale lui γ problema antrenării *Perceptronului* poate fi dificil de rezolvat. De fapt, complexitatea ei poate fi exponențială! Ca să ilustrăm acest fapt, vom folosi următorul exemplu:

$$y_i = (-1)^{i+1} \text{ și } x_i = (\underbrace{(-1)^i, \dots, (-1)^i, (-1)^{i+1}}_{i \text{ elemente}}, 0, \dots, 0) \text{ pentru } i = 1, \dots, m.$$

Demonstrați că sunt necesare $O(2^m)$ actualizări pentru ca algoritmul *Perceptron* să găsească o soluție optimă w^* , satisfăcând inegalitatea $y_i x_i \cdot w^* > 0$ pentru orice i , indiferent de modul în care sunt selectate instanțele la fiecare iterație.

⁴¹⁵Vedeți problema 16.

⁴¹⁶În enunțul original se dă pentru regula de actualizare forma $w \leftarrow w + \eta y_i x_i$. Noi însă am folosit definiția clasică: $w \leftarrow w - \eta \Delta w$, cu $\Delta w = \frac{\partial E_i}{\partial w} = -(y_i - o_i)x_i$, unde o_i este outputul perceptronului pentru inputul x_i . De remarcat că $y_i \neq o_i$ implică $y_i = 1$ și $o_i = -1$ sau $y_i = -1$ și $o_i = 1$, deci $y_i - o_i = 2y_i$ în ambele situații. Prin urmare, $\Delta w = -2y_i x_i$, iar forma regulii de actualizare este, în această abordare, $w \leftarrow w + 2\eta y_i x_i$.

c. Acum vom renunța la presupunerea că instanțele (x_i, y_i) sunt liniar separabile.⁴¹⁷ (Viața nu este aşa de simplă! Însă, din fericire, nu este nici din cauza afara de complicată.) Fie u un vector oarecare din \mathbb{R}^d , cu $\|u\| = 1$, și $\gamma > 0$. Definim *deviația* lui x_i prin expresia $d_i = \max(0, \gamma - y_i u \cdot x_i)$, iar $\delta = (\sum_{i=1}^n d_i^2)^{-1/2}$. Arătați că numărul de actualizări făcute de către algoritmul *Perceptron* este mărginit superior de $(R + \delta)^2/\gamma^2$.

Sugestie: Ați putea încerca ca, pornind de la instanțele x_i să construiți în mod convenabil instanțe separabile x'_i și apoi să refolosiți / adaptați demonstrația de la punctul precedent. De exemplu, puteți defini $x'_i = [x_i, 0, \dots, 0, c, 0, \dots, 0]$, vector în spațiul \mathbb{R}^{d+n} , având pe poziția $d+i$ constanta $c \in \mathbb{R}$. Cum se poate oare construi în mod corespunzător un vector de ponderi u' astfel încât u' să separe instanțele x'_i ? Apoi — odată ce ați obținut o margine superioară similară cu (159) —, cum ar trebui să fie aceeași constantă c astfel încât să minimizăm această margine superioară? Este oare această margine valabilă / bună [șil] pentru instanțele originale x_i ?

Observație: Ideea principală din spatele demonstrației de la punctul precedent este maparea / „proiecțarea” instanțelor x_i într-un spațiu de dimensiune mai mare, în care separarea liniară să fie posibilă. Aceasta coincide cu ideea folosirii *funcțiilor-nucleu* (engl., kernel functions), care mapează / transformă x_i în $\phi(x_i)$. Pentru varianta kernel-izată [duală] a algoritmului *Perceptron*, vedeți problema 19.

40.

(Antrenarea perceptronului:
elaborare cod C / pseudo-cod)

Liviu Ciortuz, 2012

Elaborați fie în pseudo-cod fie în limbajul de programare C funcții pentru

- construirea (adică reprezentarea ca structură de date) a unui perceptron. Veți lucra cu n intrări (numere reale), iar funcția de activare va fi de *tip* prag, liniar sau sigmoidal. Într-o versiune ulterioară veți putea adăuga tipurile: generalizat-sigmoidal (a se vedea problema 20) sau tangentă hiperbolică (*tanh*, ca în problema 43).
- calculul outputului unui perceptron (construit la punctul precedent), pentru un input dat;
- algoritmul de antrenare (adică algoritmul de actualizare a ponderilor corespunzătoare intrărilor perceptronului), conform tipului de funcție de activare indicat mai sus. Veți elabora atât varianta “batch” cât și varianta stochastică / incrementală (a se vedea cartea lui Tom Mitchell, *Machine Learning*, pag. 92-94).

Indicație: Acest algoritm se va opri atunci cind una din următoarele condiții sunt satisfăcute:

- toate instanțele de antrenament sunt corect clasificate, sau
- s-au executat deja un număr *maxim* de iterări dat ca parametru. Alternativ, se poate testa dacă funcția de optimizat (de exemplu, semisuma pătratelor erorilor, sau suma costurilor / pierderilor de tip log-sigmoidal (vedeți problema 14)) a scăzut sub un anumit prag ε (dat ca parametru) la ultima iterație executată, în comparație cu iterarea precedentă. Vă sugerăm să testați implementarea dumneavoastră pe datele de la problemele 2.a, 29, 30.a și 9 (pentru punctele *a* și *b*), și respectiv problema 11 (pentru punctul *c*). Ulterior, veți putea implementa varianta în care funcția de activare este de tip log-sigmoidal (ca la problema 14), sau de tip tangentă hiperbolică (ca la problema 43).

⁴¹⁷Deci nu există w^* astfel încât $y_i w^* \cdot x_i > 0$ pentru $i = 1, \dots, n$.

6.2.3 Rețele “feed-forward” — algoritmul de retropropagare

41.

(Rețele “feed-forward”:

algoritmul de retro-propagare, aplicare)

CMU, 2014 spring, Seyoung Kim, HW2, pr. 1.3

Considerăm că toate unitățile rețelei neuronale din figura de mai jos sunt de tip sigmoidal. Vă readucem aminte că acest fapt înseamnă că pentru inputul x_1, \dots, x_d , o astfel de unitate neuronală calculează outputul

$$\frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}},$$

unde w_i este ponderea corespunzătoare intrării x_i , pentru fiecare $i = 1, \dots, d$.

În cele ce urmează vom considera că $w_0 = 0$ pentru fiecare unitate din această rețea.

a. Presupunem că inputul rețelei este $i_1 = 1$.

Cât va fi valoarea outputului calculat de către nodul o_1 ?

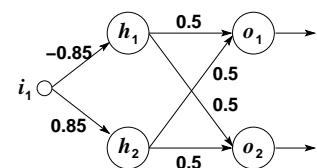
Dar valoarea outputului calculat de către nodul o_2 ?

b. De data aceasta presupunem că outputurile corecte pentru inputul $i_1 = 1$ sunt $t_1 = t_2 = 1$. Folosind formulele date pentru cantitățile δ_i în algoritmul de retro-propagare — a se vedea pr. 20 și / sau cartea *Machine Learning* de Tom Mitchell, pag. 98 —, calculați:

- valoarea lui δ_{o_1} ;
- valoarea lui δ_{o_2} .

c. În final, folosind aceste valori pentru δ și considerând că rata de învățare este $\eta = 0.1$, calculați:

- valoarea lui δ_{h_1} ;
- noua valoare a ponderii de pe conexiunea care leagă unitatea h_1 de unitatea o_1 ;
- noua valoare a ponderii de pe conexiunea care leagă unitatea h_1 de unitatea o_2 ;
- noua valoare a ponderii de pe conexiunea care leagă intrarea i_1 de unitatea h_1 .



42.

(Algoritmul de retro-propagare: Adevărat sau Fals?)

CMU, 1994 fall, A. Blum, T. Mitchell, HW4, pr. 1

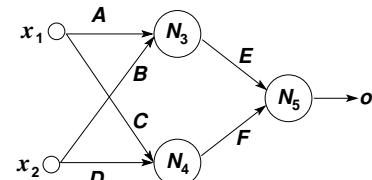
Rețeaua neuronală din figura alăturată este formată din unități de același tip, de exemplu (pentru fixarea ideilor) sigmoidal.

Presupunem că executăm algoritmul de retro-propagare pe această rețea, folosind un set oarecare de date de antrenament și dând inițial valori egale tuturor ponderilor din rețea.

a. Precizați care dintre afirmațiile de mai jos vor fi întotdeauna adevărate:

- i. Ponderile A și B nu vor dифeri niciodată.
- ii. Ponderile A și C nu vor dифeri niciodată.
- iii. Ponderile E și F nu vor dифeri niciodată.

b. Justificați de ce NU este bine să se inițializeze toate ponderile dintr-o rețea neuronală cu o aceeași valoare atunci când se rulează algoritmul de retro-propagare.



43.

(Rețele neuronale — algoritmul de retro-propagare pentru rețele de unități neuronale având funcția de activare \tanh)

prelucrare de Liviu Ciortuz, după T. Mitchell, "Machine Learning", 1997, pr. 4.8 și CMU, 2011 spring, Roni Rosenfeld, HW4, pr. 2.a

Rescrieți algoritmul de retro-propagare pentru rețele neuronale aciclice în care unitățile neuronale folosesc funcția de activare tangentă hiperbolică (\tanh) în locul funcției sigmoidale.

Recomandare: Pentru a reduce din generalitatea problemei, vă veți limita la a elabora varianta stochastică / incrementală a acestui algoritm pentru rețele de tip feed-forward cu două niveluri.

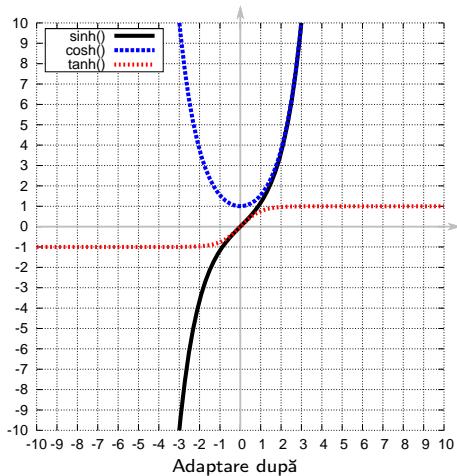
Vă reamintim că

$$\tanh(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}} = \frac{e^{2y} - 1}{e^{2y} + 1} = \frac{\sinh(y)}{\cosh(y)},$$

iar

$$\sinh(y) = \frac{e^y - e^{-y}}{2} \text{ și } \cosh(y) = \frac{e^y + e^{-y}}{2}$$

Din definiția de mai sus rezultă imediat că funcția \tanh este derivabilă și deci continuă pe întreg domeniul de definiție (axa reală), iar din imaginea alăturată se observă că funcția \tanh este o formă „netezită“ (engl., smoothed) a funcției treaptă $sign$.



http://en.wikipedia.org/wiki/File:Sinh_cosh_tanh.svg

Așadar, vom considera că outputul unităților neuronale din rețea este de forma $o(\bar{x}) = \tanh(\bar{w} \cdot \bar{x})$.

- Demonstrați că $\tanh'(y) = 1 - \tanh^2(y)$ pentru orice $y \in \mathbb{R}$.
- Arătați că pentru regula de actualizare a ponderilor unităților neuronale de pe nivelul de ieșire, $w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$, avem

$$\Delta w_{kj} \stackrel{\text{def.}}{=} -\eta \frac{\partial E}{\partial w_{kj}} = \eta(t_k - o_k)(1 - \tanh^2(\text{net}_k))y_j$$

unde notațiile sunt similare cu cele din carteaua *Machine Learning* de Tom Mitchell.⁴¹⁸

- Similar, pentru ponderile unităților neuronale de pe nivelul ascuns:

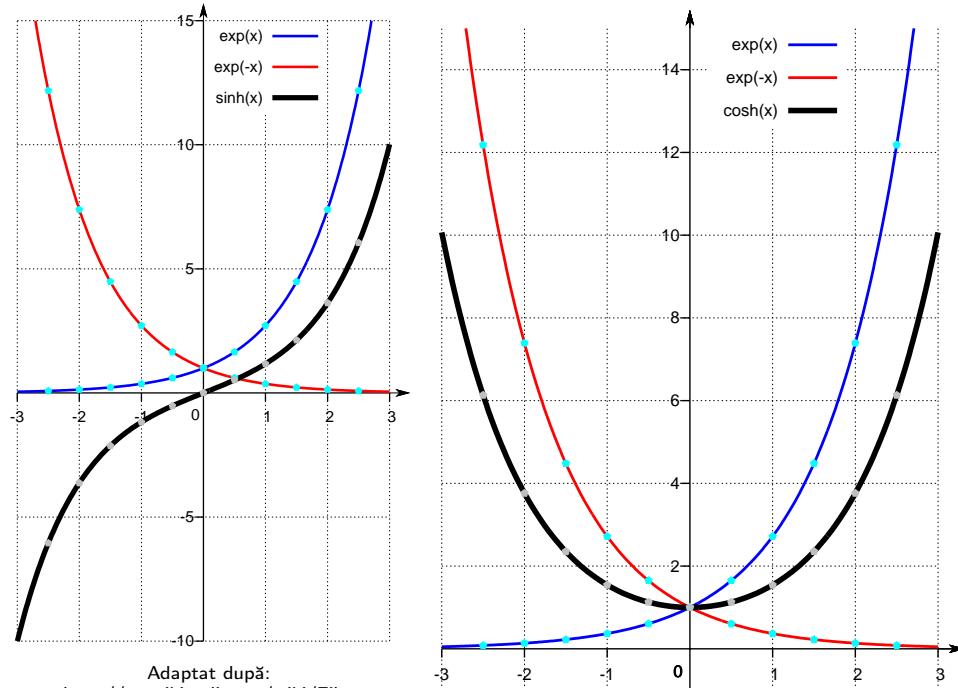
$$\Delta w_{ji} \stackrel{\text{def.}}{=} -\eta \frac{\partial E}{\partial w_{ji}} = \eta(1 - \tanh^2(\text{net}_j))x_i \left(\sum_{k \in \text{Downstream}(j)} \delta_k w_{kj} \right),$$

⁴¹⁸Aceleasi notații au fost folosite și în problema 20.

unde

$$\delta_k = -\frac{\partial E_d(\bar{w})}{\partial net_k} \text{ iar } E_d(\bar{w}) = \sum_{k \in outputs} \frac{1}{2}(t_{k,d} - o_{k,d})^2,$$

d fiind o instanță de antrenament oarecare sau (echivalent) indicele asociat ei.



Adăugăm câteva **observații** cu caracter instructiv / informativ în legătură cu funcția $tanh$:

(1) Pentru a vedea relația dintre funcția \sinh (și respectiv \cosh) pe de o parte și funcțiile exponentiale (e^x) și invers-exponentiale (e^{-x}) pe de altă parte, cititorul poate observa graficele de mai jos.

(2) Legătura dintre funcția sigmoidală $\sigma(x) = \frac{1}{1 + e^{-x}}$ (care mai este numită și *funcția logistică*) și funcția $tanh$ este dată de relația:

$$\sigma(x) = \frac{1}{2} \left(1 + \tanh \frac{x}{2} \right)$$

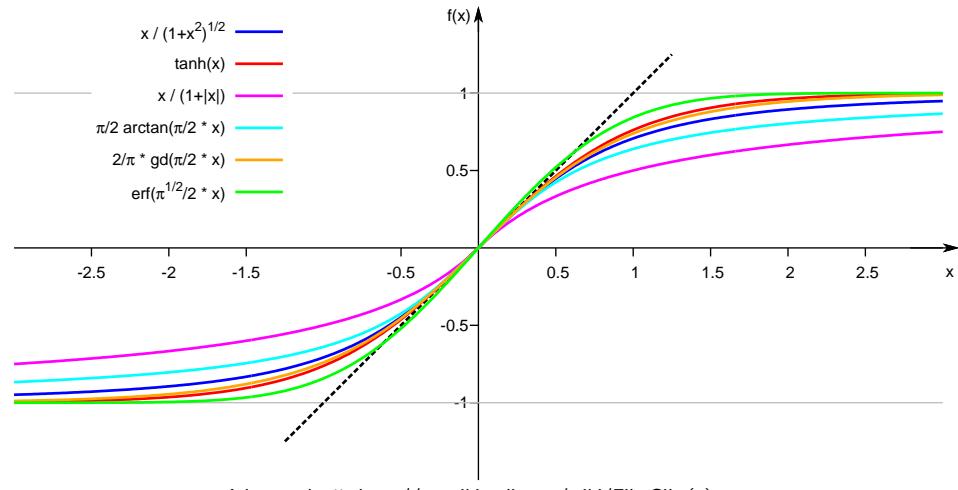
(3) Spre deosebire de funcția sigmoidală (care este o variantă „netezită“ a funcției treaptă 0/1), funcția $tanh$ este simetrică față de originea sistemului de coordonate. Din această cauză, în literatura de specialitate se apreciază că funcția $tanh$ favorizează o convergență mai rapidă a algoritmului de antrenare a rețelei neuronale.

(4) Există și alte funcții care „netezesc“ funcțiile-treaptă, de exemplu:

$\frac{x}{1+|x|}$, $\frac{x}{\sqrt{1+x^2}}$, funcția arctg (inversa funcției tangentă trigonometrică), funcția de eroare gaussiană (notată erf , prin abreviere de la engl. error function) și funcția Gudermannian (gd), unde

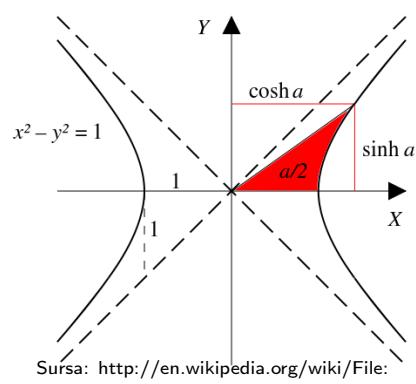
$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \text{ iar } gd(x) = \int_0^x \frac{dt}{cosh}(t) = 2 \operatorname{arctg}(e^x) - \frac{\pi}{2}$$

Pentru aceste funcții se pot defini versiuni „normalize“, în aşa fel încât panta tangentei lor pentru $x = 0$ să fie 1. A se vedea graficul următor.



(5) Legătura dintre funcțiile trigonometrice hiperbolice și funcțiile trigonometrice clasice („circulare“) poate fi ilustrată astfel:

Pentru funcțiile trigonometrice circulare avem proprietatea fundamentală: $\sin^2 \alpha + \cos^2 \alpha = 1$. În consecință, dacă notăm $x = \cos \alpha$ și $y = \sin \alpha$, rezultă că punctul (x, y) este situat pe cercul de ecuație $x^2 + y^2 = 1$ (având centru O și raza 1). Pentru funcțiile trigonometrice hiperbolice avem o altă proprietate, care se verifică imediat: $\cosh^2 a - \sinh^2 a = 1$. Așadar, dacă notăm $x' = \cosh a$ și $y' = \sinh a$, rezultă că punctul (x', y') este situat pe hiperbola de ecuație $(x')^2 - (y')^2 = 1$, ale cărei asimptote la $+\infty$ și $-\infty$ sunt prima și cea de-a doua bisectoare a sistemului de coordinate în plan. (De remarcat proprietatea: pentru $\cosh a$ și $\sinh a$, aria zonei hașurate din imagine este $a/2$.)



44.

(Aplicarea algoritmului de retro-propagare pe o rețea feed-forward formată din unități sigmoidale dispuse pe 2 niveluri, utilizând și componenta „moment“)
adaptare făcută de către Liviu Ciortuz, după T. Mitchell, "Machine Learning", 1997, pr. 4.7

Se consideră o rețea neuronală de tip feed-forward cu două niveluri, având două intrări a și b , o unitate ascunsă c și o unitate de ieșire d . Ambele unități sunt de tip sigmoidal. Această rețea are 5 ponderi: $w_{ca}, w_{cb}, w_{c0}, w_{dc}, w_{d0}$, unde w_{c0} reprezintă ponderea asociată termenului liber ($x_0 = 1$) pentru unitatea c și similar pentru unitatea d .

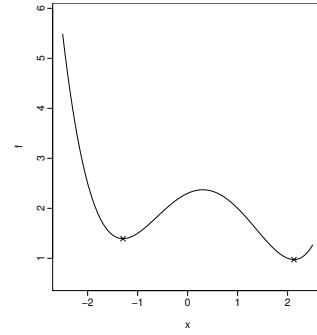
Inițializați ponderile cu valorile $(0.1, 0.1, 0.1, 0.1, 0.1)$, apoi calculați valorile lor după fiecare dintre primele două iterații ale algoritmului de retro-propagare în varianta "batch". Considerăm rata de învățare $\eta = 0.3$, momentul $\alpha = 0.9$ și datele de antrenament din tabelul alăturat.⁴¹⁹

a	b	t
1	0	1
0	1	0

Comentariu:

Vă reducem aminte următoarele:

1. La aplicarea metodei gradientului descendente, pentru evitarea „căderii“ în unele minime locale, regula de actualizare a parametrilor w_{ji} , și anume $w_{ji}^{(n)} \leftarrow w_{ji}^{(n-1)} + \Delta w_{ji}^{(n)}$, se poate extinde cu încă un termen, $\alpha \Delta w_{ji}^{(n-1)}$, numit *termenul moment*, datorită analogiei cu mișcarea unui punct material din fizică. (Constanta $\alpha > 0$ desemnează cât anume dorim să păstrăm din „inerția“ de mișcare a particulei considerate.) Veți lua $\Delta w_{(ji)}^{(0)} = 0$ pentru orice j și orice i .



2. Folosind notațiile de la pag. 101 din cartea *Machine Learning* de Tom Mitchell, avem $\Delta w_{ji}^{(n)} = -\eta \frac{\partial E}{\partial w_{ji}}(w^{(n-1)})$. Făcând abstracție de n , această cantitate Δw_{ji} este $\eta (\sum_l \delta_{j,l} x_{j,i,l})$, cu $\delta_{j,l} = o_{j,l}(1 - o_{j,l})(t_{j,l} - o_{j,l})$ atunci când j este indicele unei unități de pe nivelul de ieșire și, respectiv, $\delta_{j,l} = o_{j,l}(1 - o_{j,l}) (\sum_{k \in \text{Downstream}(j)} \delta_{k,l} w_{kj})$ atunci când j este indicele unei unități de pe un nivel ascuns. Indicele l parcurge setul de instanțe de antrenament.

⁴¹⁹Dacă limităm exercițiul acesta la aplicarea manuală a primei iterații, atunci nu avem de-a face cu componenta *moment*, însă el rămâne un exercițiu instructiv prin faptul că se solicită aplicarea algoritmului de retro-propagare în regim "batch", adică non-incremental (ceea ce nu mai apare, deocamdată, în niciun alt exercițiu din acest capitol). Aceasta implică folosirea formulelor de actualizare a ponderilor în forma din *Comentariul* al doilea de mai jos.

45. (Regularizare:
prevenirea overfitting-ului pentru o rețea feed-forward
cu unități de tip (generalizat-)sigmoidale;
extensie pentru problemele 20 și 43)
*prelucrare de Liviu Ciortuz, după
Tom Mitchell, "Machine Learning", 1997, pr. 4.10
CMU, 2011 spring, Roni Rosenfeld, HW4, pr. 2.b*

Considerăm o rețea neuronală de tip feed-forward care folosește o funcție de eroare adecvată pentru prevenirea overfitting-ului (a se vedea problema 22):

$$E_D(w) = \sum_{d \in D} \sum_{k \in \text{outputs}} \frac{1}{2} (t_{kd} - o_{kd})^2 + \gamma \sum_{i,j} w_{ji}^2,$$

unde D este mulțimea instanțelor de antrenament.

- Deduceți regulile de actualizare a ponderilor de pe conexiunile din rețea, corespunzător acestei definiții a erorii presupunând că rețeaua folosește
- doar unități (generalizat-)sigmoidale (ca la problema 20);
 - doar unități cu funcția de activare de tip \tanh (ca la problema 43).

46. (Algoritmul de retro-propagare pentru rețele feed-forward
cu funcția obiectiv de tip cross-entropie)
*CMU, 2008 fall, E. Xing, HW2, pr. 2.2
CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, HW3, pr. 2.2*

Fie o rețea neuronală de tip feed-forward având d intrări, H unități pe nivelul ascuns și K unități pe nivelul de ieșire. Toate unitățile sunt de tip sigmoidal. Vom considera N exemple de antrenament, iar funcția de eroare cu care vom lucra este de tip cross-entropie:

$$E(w) = - \sum_{n=1}^N \sum_{k=1}^K [t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk})]$$

unde

t_{nk} este target-ul pentru al n -lea exemplu și pentru ieșirea k a rețelei,
 y_{nk} este răspunsul rețelei pentru ieșirea k atunci când cel de-al n -lea exemplu este furnizat rețelei.

În această problemă veți calcula regula de actualizare pentru algoritmul de tip retro-propagare (folosind metoda gradientului descedent) în varianta stochastică. După cum știți, regula de actualizare a ponderii de pe conexiunea care leagă intrarea i de unitatea j este:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \eta \frac{\partial E_n}{\partial w_{ji}^{(t-1)}},$$

unde E_n este funcția de eroare pentru exemplul n .

Vom nota cu net_j intrarea în componenta de activare a unității j :

$$net_j = \sum_i w_{ji} z_i,$$

unde z_i este outputul unității ascunse i (dacă unitatea j este pe nivelul de ieșire) sau intrarea i (dacă unitatea j este pe nivelul ascuns).

Observații: (i) Întrucât veți calcula regula de actualizare pentru un singur exemplu, puteți elimina indicele n din notațiile de mai sus. (ii) În mod *implicit* vom considera că nu se folosesc termeni liberi (engl., bias) pentru neuronii rețelei.

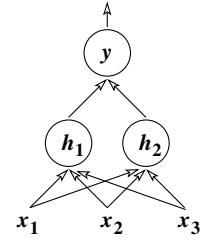
- Câți parametri (i.e., câte ponderi) are rețeaua aceasta? (Veți presupune, la acest punct, că se folosesc termeni liberi.)
- Calculați derivata funcției de activare, $\sigma'(x)$, în funcție de însuși $\sigma(x)$.
- Definim $\delta_j = \frac{\partial E_n}{\partial net_j}$. Arătați că $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$.⁴²⁰
- Arătați că pentru orice unitate de ieșire k , avem $\delta_k = y_k - t_k$.
- Considerăm intrarea (x_1, x_2, \dots, x_d) . Deducreți regula de actualizare pentru $w_{kj}^{(t)}$, valoarea ponderii de pe conexiunea dintre unitatea ascunsă j și unitatea de ieșire k la iterată t , în funcție de x_i și de valoarea ponderilor din rețea la iterată $t - 1$.
- Pentru o unitate ascunsă j oarecare, avem $\delta_j = \frac{\partial E_n}{\partial net_j} = \sum_k \frac{\partial E_n}{\partial net_k} \frac{\partial net_k}{\partial net_j}$. Arătați că $\delta_j = \sigma(net_j)(1 - \sigma(net_j)) \sum_k \delta_k w_{kj}$.
- Considerăm intrarea (x_1, x_2, \dots, x_d) . Deducreți regula de actualizare pentru $w_{ji}^{(t)}$, valoarea ponderii de pe conexiunea dintre intrarea i și unitatea ascunsă j la iterată t , în funcție de x_i și de valoarea ponderilor din rețea la iterată $t - 1$.

47.

(Aplicarea metodei gradientului descentant pentru o rețea formată din unități cu funcția de activare ReL)

CMU, 2015 spring, Alex Smola, final, pr. 2.2

În figura alăturată vi se dă structura unei rețele neuronale simple, având un singur nivel ascuns. Inputul este 3-dimensional: $x = (x_1, x_2, x_3)$. Nivelul ascuns este constituit din două unități, deci vom putea scrie outputul acestui nivel sub forma $h = (h_1, h_2)$. Nivelul de ieșire este constituit dintr-o singură unitate, y . Pentru conveniență, nu vom folosi termeni liber (adică, bias-uri). Pentru toate unitățile neuronale din această rețea vom folosi funcția de activare liniar-rectificată (engl., linear rectified activation function, ReL), definită prin expresia $f(z) = \max(0, z)$ pentru orice $z \in \mathbb{R}$.



Considerăm funcția de eroare / cost / pierdere (engl., loss function) $l(y, t) = \frac{1}{2}(y - t)^2$, unde t este valoarea target pentru inputul x , iar y este valoarea produsă de rețea pentru același input. De asemenea, vom nota cu W și V matricele de ponderi care corespund

⁴²⁰Egalitatea $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$ are loc nu doar pentru funcția E din enunț (o cross-entropie), ci și, de exemplu, pentru semisuma pătratelor erorilor (vedeți pr. 20). Mai general, ea are loc pentru acele funcții de cost E în care ponderile w sunt folosite doar în interiorul termenilor de forma $net_j \stackrel{\text{def.}}{=} \sum_i w_{ji} x_i$. În rest — vedeți, de exemplu, cazul folosirii termenilor pentru regularizare, pr. 22 —, această manieră de lucru (centrată pe calculul cantităților δ) nu mai este adecvată / utilizabilă.

conexiunilor dintre nivelul de intrare și nivelul ascuns, și respectiv dintre nivelul ascuns și nivelul de ieșire. Aceste matrice sunt inițializate astfel:

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \text{ și } V = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

- a. Folosind simbolii f , W and V , scrieți [în manieră matriceală] expresia funcției $x \rightarrow y$ calculate de către rețea neuronală dată. (Nu este nevoie să folosiți aici valorile care au fost specificate mai sus pentru matricele W și V .)
- b. Presupunem că inputul rețelei este $x = (1, 2, 1)$, iar valoarea target este $t = 1$. Calculați valoarea *numerică* pentru outputul y , elaborând în mod clar toți pașii intermediari. Puteți refolosi rezultatele de la punctul precedent. Folosirea formei matriceale este recomandată, dar nu este obligatorie.
- c. Calculați vectorul gradient pentru funcția de eroare l , în raport cu ponderile rețelei. În mod specific, calculați mai întâi [în manieră *simbolică*] expresiile gradientului
 - relativ la matricea de ponderi V , adică $\frac{\partial l}{\partial V}$;
 - relativ la matricea de ponderi W , adică $\frac{\partial l}{\partial W}$.

Calculați apoi valoarea *numerică* a gradientului funcției l , pentru valorile specificate mai sus pentru W , V , x și y .

48. (Proprietăți ale unei variante a algoritmului *Perceptron kernel-izat*,
în cazul când nucleul este de tip RBF)
*prelucrare de L. Ciortuz, după
MIT, 2009 fall, Tommi Jaakkola, midterm, pr. 2*

Algoritmul *Perceptron* — cu care am făcut cunoștință la problema 16 — constituie probabil una dintre cele mai simple modalități de a rezolva probleme de clasificare. La problema 19 am prezentat varianta kernel-izată a acestui algoritm și am menționat funcția-nucleu de tip Gaussian (care se mai numește și *funcție cu bază radială*, RBF). În pseudo-codul următor redăm algoritmul *Perceptron kernel-izat* într-o variantă ușor extinsă în raport cu pseudo-codul din problema 19, în sensul că aici se permite parcurgerea de mai multe ori a setului de date de antrenament.

Date de intrare: $x_1, \dots, x_n \in \mathbb{R}^d$ și $y_1, \dots, y_n \in \{-1, +1\}$;

Inițializare: $\alpha_1 = \dots = \alpha_n = 0$;

Procedură:

Parcurge ciclic $i = 1, \dots, n$ până când nu se mai produce nicio greșeală
dacă $y_i(\sum_{j=1}^n \alpha_j y_j K(x_j, x_i)) \leq 0$, atunci $\alpha_i \leftarrow \alpha_i + 1$;

În cele de mai jos, funcția K va fi considerată funcție-nucleu de tip RBF, deci $K(x, x') = \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$ și $\sigma > 0$.

- a. Converge oare întotdeauna acest algoritm? (În acest context, a converge înseamnă că la un moment dat algoritmul nu mai actualizează variabilele α_i .) Trebuie oare să adăugăm vreo condiție suplimentară pentru a ne asigura că algoritmul nostru se oprește?
- b. Indicați — și apoi justificați în mod riguros — care dintre afirmațiile următoare sunt *adevărate*.

- i. Dacă algoritmul converge, atunci el găsește o soluție⁴²¹ pentru care putem calcula „marginea“ de separare.⁴²² Această „margină“ depinde de ordinea în care sunt parcuse exemplele de antrenament.
- ii. Presupunem că toate instanțele de antrenament x_i , cu $i = 1, \dots, n$, sunt distințe. În acest caz, dacă parametrul nucleului RBF (și anume, σ) are o valoare suficient de mică, algoritmul converge în mod cert după ce va fi făcut maximum n clasificări eronate (engl., mistakes).
- iii. Numărul de clasificări eronate făcute de acest algoritm (în cazul în care converge) depinde de valoarea parametrului σ .

Indicație: Pentru a putea răspunde corect la întrebările de mai sus, vă recomandăm să faceți referire la rezultatele teoretice demonstrează problema 58 de la capitolul *Mașini cu vectori-support* din prezența culegere, precum și la problema 18 din capitolul de față (*Rețele neuronale artificiale*).

49.

(Expresivitate: rețele neuronale vs. arbori de decizie)

CMU, 2009 spring, Ziv Bar-Joseph, HW2, pr. 3

La acest exercițiu veți compara puterea de reprezentare a arborilor de decizie pe de o parte și a rețelelor neuronale pe de altă parte, privitor la capacitatea de a codifica două funcții, și anume *funcția majoritate* (engl., majority function) și *funcția paritate* (engl., parity function). Ambele funcții primesc ca input n -uple $x_1, \dots, x_n \in \{0, 1\}^n$, iar outputul lor este în multimea $\{0, 1\}$. Funcția *majoritate* returnează valoarea 1 dacă mai mult de jumătate dintre componente din intrare sunt 1. De exemplu, tuplul $(1, 1, 1, 1, 0)$ conține mai multe cifre de 1 decât de 0, deci va produce outputul 1. Funcția *paritate* returnează 1 dacă și numai dacă un număr par de componente din intrare sunt 1. De exemplu, tuplul $(1, 0, 1, 0, 1, 0)$ conține 3 cifre de 1, deci produce outputul 0.

- a. Poate oare perceptronul[-prag] să fie folosit ca să codifice
 - funcția majoritate?
 - funcția paritate?

Pentru fiecare dintre aceste două funcții, în cazul în care ați răspuns afirmativ, prezentați perceptronul corespunzător, iar dacă ați răspuns negativ, justificați.

- b. Poate oare un arbore de decizie să codifice
 - funcția majoritate?
 - funcția paritate?

Pentru fiecare dintre aceste două funcții, în cazul în care ați răspuns afirmativ, prezentați arborele de decizie corespunzător, iar dacă ați răspuns negativ, justificați.

- c. Pentru fiecare dintre cele două funcții de la punctul a la care ați răspuns negativ, este oare posibil să obținem codificarea dorită folosind o rețea neuronală cu două niveluri (un nivel ascuns și un nivel de ieșire)? Dacă este așa, desenați rețea care rezolvă problema. Altminteri, precizați câte niveluri ascunse sunt necesare pentru a putea rezolva problema.

⁴²¹ Adică un *separator liniar* în spațiul de trăsături \mathbb{R}^m , conform „mapării“ $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ specifice nucleului RBF.

⁴²² Adică distanța de la [iperplanul] separator până la cele mai apropiate „imagini“ $\phi(x_i)$.

50. (Adevărat / Fals?)

*CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm, pr. 1.5
 CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, midterm, pr. 1.a*

- a. Granița de decizie învățată de către o rețea neuronală este întotdeauna neliniară.
- b. Este posibil ca la antrenarea perceptronului liniar — așadar, urmărind să minimizăm suma pătratelor erorilor, folosind metoda gradientului descendente — să obținem puncte [multiple] de optim local.
- c. Indiferent de mărimea rețelei neuronale, algoritmul de retro-propagare poate întotdeauna să găsească valorile optime globale pentru ponderile rețelei.

51. (Antrenarea rețelelor feed-forward
 în diverse variante: elaborare cod C / pseudo-cod)

Liviu Ciortuz, 2012

Elaborați fie în pseudo-cod fie în limbajul de programare C funcții pentru

- a. configurarea (adică reprezentarea ca structură de date) a unei rețele neuronale artificiale de tip feed-forward;

Indicație: Într-o primă versiune, veți lucra cu n intrări (numere reale), un singur nivel ascuns și o singură unitate pe nivelul de ieșire, iar toate unitățile neuronale vor fi de același tip (de exemplu, sigmoidal). Ulterior, veți elabora o variantă extinsă a acestei funcții, în așa fel încât să se permită lucrul cu K niveluri ascunse, iar pentru fiecare unitate din rețea să se poată alege unul din tipurile de perceptri indicate la problema 40.

- b. calculul outputului unei rețele de tipul indicat la punctul precedent, pentru un input dat.

Indicații:

1. La punctele a și b veți putea folosi funcțiile corespunzătoare de la implementarea perceptronului (vedeți problema 40).
2. Vă sugerăm să testați implementarea dumneavoastră (la acest stadiu) pe datele de la problemele 1.ab, 2.b, 3, 6, 8, [7.] 30.b și 33.
- c. algoritmul de antrenare (adică algoritmul de retro-propagare) pentru rețele neuronale, de asemenea de tipul indicat mai sus. Veți elabora atât varianta "batch" cât și varianta stochastică / incrementală (a se vedea cartea lui Tom Mitchell, *Machine Learning*, pag. 92-94).

Indicație: Vă sugerăm să elaborați mai întâi o versiune de bază a acestei funcții, pe care să o testați pe datele de la problema rezolvată 21 (verificând la final rezultatele) și de la problemele propuse 41 și 42. Ulterior puteți extinde această versiune (fie în manieră parametrizată, fie orientată pe obiecte) în așa fel încât să poată opera cu diverse variante ale *funcției obiectiv* de optimizat: fie semisuma pătratelor erorilor, fie varianta cu *regularizare*, pentru prevenirea overfitting-ului (ca în problemele 22 și 45), fie o funcție de tip cross-entropie (ca în problema 46). În sfârșit, încă o variantă posibilă este cea în care regulile de actualizare a ponderilor conțin și o componentă de tip „moment“, pentru evitarea unor optime locale ale funcției de eroare (a se vedea problema 44).

52. (Funcții de cost în învățarea profundă și, mai general, învățarea automată)
CMU, 2015 fall, A. Smola, B. Poczos, HW1, pr. 3.2

În acest exercițiu vom lucra cu câteva dintre cele mai des folosite funcții obiectiv din învățarea profundă [și învățarea automată]. Stabilită, pe bază de demonstrație, dacă ele sunt (sau nu) convexe.

Sugestie: Puteți face apel la *proprietățile* prezentate la problema 70 de la capitolul de *Fundamente*.

Vom introduce acum anumite *notații* pe care le vom folosi în continuare:

- $w \in \mathbb{R}^d$ reprezintă o pondere;
 - $x_i \in \mathbb{R}^d$ este o instanță (vector de trăsături); vom considera că primul element din vectorul x_i are întotdeauna valoarea 1, deci în cele ce urmează nu vom folosi [niciun] termen liber (engl., bias term);
 - $r_i = w \cdot x_i \in \mathbb{R}$;
 - $y_i \in \mathbb{R}$ este eticheta instanței x_i .
- a. Funcția de cost *logistică* (engl., logistic loss function): $L(r_i) = -\ln(1 + e^{-r_i})$.
 - b. Funcția *ReLU* (engl., Rectified Linear function): $R(r_i) = \max(0, r_i)$.
 - c. Funcția de cost *hinge* (engl., hinge loss): $H([r_1, r_2, \dots, r_N]) = \sum_{i=1}^N \max(0, 1 - y_i r_i)$.
 - d. Funcția de cost *softmax* (engl., softmax loss) pentru straturi cu conexiuni totale (engl., fully connected layers): considerăm matricea $W \in \mathbb{R}^{d \times k}$, pentru care vom nota coloana i cu w_i ; $W^\top x_i = [w_1 \cdot x_i, \dots, w_k \cdot x_i]^\top \stackrel{\text{not.}}{=} [r_i(1), r_i(2), \dots, r_i(k)]^\top$; $S_s(W) = \ln \left(\sum_{j=1}^k e^{r_i(j)} \right) - r_i(s)$.
 - e. Bazându-vă pe punctele precedente, precizați motivele din cauza cărora se poate „rupe“ convexitatea funcțiilor obiectiv folosite din învățarea profundă. Justificați pe scurt de ce unele dintre funcțiile obiectiv folosite în învățarea profundă nu sunt convexe.

53. (Rețele neuronale convoluțive: determinarea mărimii hărții de trăsături de pe un anumit nivel)

CMU, 2015 fall, E. Xing, Z. Bar-Joseph, midterm, pr. 4.1

Considerăm un nivel convoluțiv C urmat de un nivel de tip selecție a maximului (engl., max pooling layer) P . Inputul nivelului C are 50 de *canale* / planuri, fiecare dintre ele având dimensiunea 12×12 . Nivelul C are 20 de *filtre*, fiecare dintre acestea fiind de mărime 4×4 . Adausul / *bordarea* convoluțivă (engl., the convolution padding) este de mărime 1, iar lungimea *pasului*⁴²³ (engl., the stride) este 2.

Nivelul P efectuează o selecție a maximului din fiecare *hartă de trăsături* (engl., feature maps) din outputul nivelului C , cu *ferestre* (sau, câmpuri locale receptive; engl., local receptive fields) de dimensiune 3×3 și lungimea *pasului* este 1.

Cât este mărimea fiecărei *hărți de trăsături* (engl., feature map) din outputul nivelului P ?

⁴²³Pasul se referă la deplasarea *ferestrei* care corespunde filtrului de pe nivelul respectiv.

Această pagină a fost lăsată liberă în mod intenționat.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

7 Clusterizare

Sumar

Noțiuni de bază

- instanță neetichetată vs. instanță etichetată (exemplu de antrenament);
- învățare nesupervizată (clusterizare) vs. învățare supervizată (clasificare);
- [funcție / măsură de] distanță definită pe $\mathbb{R}^d \times \mathbb{R}^d$: ex. 2 de la capitolul *Învățare bazată pe memorare*;
- cluster / grup / grupare / bin (engl.) vs. clasă;
- tipuri de clusterizare: ierarhică vs. neierarhică;
- tipuri de ierarhii: ierarhii (arbori de clusterizare, dendrograme) obișnuite vs. ierarhii plate (engl., flat hierarchies);
exemple: ex. 1.a și respectiv ex. 1.b, ex. 6.a;
- tipuri de apartenență a unei instanțe la un cluster: hard vs. soft (ultima numai pt. clusterizare neierarhică).

7.1. Clusterizare ierarhică

7.1.1. Noțiuni specifice

- [funcție de] similaritate între clustere, definită pe baza [extinderii] noțiunii de distanță la $\mathcal{P}(X) \times \mathcal{P}(X)$, unde $X \subset \mathbb{R}^d$ este mulțimea de instanțe, iar $\mathcal{P}(X)$ este mulțimea părților lui X ;

tipuri de [funcții de] similaritate:

“single-linkage”:⁴²⁴ $d(A, B) = \min\{d(x, y) | x \in A, y \in B\}$

“complete-linkage”:⁴²⁵ $d(A, B) = \max\{d(x, y) | x \in A, y \in B\}$

“average-linkage”: $d(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$

metrica lui Ward: ex. 31, ex. 32, ex. 33.

În general, putem considera $sim(A, B) = 1/(1+d(A, B))$ sau chiar $sim(A, B) = 1/d(A, B)$ dacă lucrăm doar cu clustere non-singleton;

proprietate / restricție: $sim(A \cup B, C) \leq \min\{sim(A, C), sim(B, C)\}$ pentru orice clustere A, B selectate de algoritmul de clusterizare ierarhică la un pas oarecare [al algoritmului de clusterizare ierarhică] și orice alt cluster C ;

- [funcție de] coeziune internă a unui cluster (sau: între elementele / instanțele dintr-un cluster);

exemplu (pentru clustere non-singleton):

$$coh(A) = \left(\frac{1}{C_{|A|}^2} \sum_{x, y \in A} d(x, y) \right)^{-1} = \frac{C_{|A|}^2}{\sum_{x, y \in A} d(x, y)}.$$

⁴²⁴Sau: nearest-neighbour.

⁴²⁵Sau: furthest-neighbour.

7.1.2. Algoritmi de clusterizare ierarhică

- tipuri de algoritmi de clusterizare ierarhică:
bottom-up (*clusterizare aglomerativă*) vs. top-down (*clusterizare divizivă*);
- pseudo-cod: Manning & Schütze, *Foundations of Statistical Natural Language Processing*, 2002, pag. 502;
- analiza (ca algoritmi *per se*): ambii algoritmi sunt iterativi și “greedy”; rezultatele (ierarhiile) obținute nu sunt determinate neapărat în mod unic: ex. 3.b;
- exemple de aplicare: ex. 1-5, ex. 26-30 (pentru bottom-up), respectiv ex. 6 (pentru top-down);
- implementări: ex. 35, ex. 33, ex. 36.

7.1.3 Proprietăți

- (P0) clusterizarea folosind similaritate de tip “single-linkage” are tendința să creeze clustere alungite; invers, folosind similaritate “complete-linkage” sau “average-linkage”, se formează clustere de formă mai degrabă sferică: ex. 5 și ex. 29;
- (P1) dacă atunci când folosim “single-linkage” și “complete-linkage” se obțin dendograme identice, *nu* rezultă în mod neapărat că folosind “average-linkage” vom obține aceeași dendrogramă: ex. 3.b;
- (P2) numărul maxim de niveluri dintr-o dendrogramă (văzută ca arbore în sensul teoriei grafurilor) este $n - 1$, unde n este numărul de instanțe de clusterizat: ex. 4.a; numărul minim de niveluri: $\lceil \log_2 n \rceil$; ex. 4.b;
- (P3) există o anumită corespondență între clusterizare ierarhică cu similaritate de tip
 - “single-linkage” și aflarea *arborelui [de acoperire] de cost minim* dintr-un graf: ex. 6;
 - “complete-linkage” și aflarea unei *clici* (subgraf maximal complet) dintr-un graf (vedeți Manning & Schütze, *op. cit.*, pag. 506-507);
- (P4) algoritmul de clusterizare aglomerativă la al cărui pseudo-cod am făcut referire mai sus are complexitate $\mathcal{O}(n^3)$: ex. 26; atunci când se folosește single-linkage sau complete-linkage, există însă versiuni / algoritmi de complexitate $\mathcal{O}(n^2)$: SLINK (1973) și respectiv CLINK (1976);
- la clusterizare ierarhică aglomerativă cu similaritate “average-linkage”:
- (P5) dacă se folosește ca măsură de similaritate între 2 instanțe cosinusul unghiului dintre vectorii care reprezintă instanțele și se „normalizează“ acești vectori (i.e., se lucrează cu 2 vectori coliniari cu ei, dar de normă egală cu 1), atunci calculul coeziunii [interne a] unui cluster nou format, precum și calculul „distanței“ dintre două clustere se pot face în timp constant: ex. 34.

7.2. Clusterizare partitională

7.2.1 Noțiuni specifice

- centroid (centru de greutate) al unui cluster,
- K -partiție, K -configurație [initială] a centroizilor: ex. 11;
- o funcție de evaluare a „calității“ clusterelor (sau: funcție de „coeziune“ / „distorsiune“ / „eroare“ totală):
 - „suma celor mai mici pătrate“: $J_K(C, \mu) = \sum \|x_i - \mu_{C(x_i)}\|^2$, unde C este K -partiție, μ este K -configurație de centroizi, iar $\mu_{C(x_i)}$ este centroidul cel mai apropiat de x_i : ex. 12.

7.2.2 Algoritmul K -means

- pseudo-cod (o versiune [mai] generală): Manning & Schütze, *op. cit.*, pag. 516; alternativ, vedeti enunțul ex. 12 (sau, echivalent, folosind variabile-indicator: ex. 42); exemple de aplicare: ex. 7-11, ex. 17.a, ex.21.a, ex. 22.a, ex. 37, ex. 38.
- exemple de *euristici pentru inițializarea centroizilor*: inițializare arbitrară / random în \mathbb{R}^d sau în $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{R}^d$ (setul de date de clusterizat); aplicare în prealabil a unui algoritm de clusterizare ierarhică; folosind o anumită distribuție probabilistă definită pe X : K -means++ (David Arthur, Sergei Vassilvitskii, 2007): ex. 46.
- exemple de *criterii de oprire*: după efectuarea unui număr maxim de iterații (fixat inițial); când compoziția clusterelor nu se mai modifică de la o iteratie la alta; când pozițiile centroizilor nu se mai modifică de la o iteratie la alta; când descreșterea valorii criteriului J_K de la o iteratie la alta nu mai este strictă sau nu mai este peste un anumit prag ϵ fixat în prealabil.
- ca algoritm *per se*:
 - K*-means este un algoritm de *căutare*: spațiul de căutare este mulțimea tuturor K -partițiilor care se pot forma pe dataset-ul de intrare;
 - (P0) întrucât acest spațiu de căutare (deși este finit) este exponențial (K^n), K -means explorează doar partiaj spațiul de căutare, procedând *iterativ*: el pleacă de la o „soluție“ (K -partiție) aleasă eventual în mod arbitrar / aleatoriu și o „îmbunătățește“ la fiecare iteratie;
 - (P1) soluția găsită este dependentă de inițializarea centroizilor: ex. 10;
 - (P1') mai mult, chiar la o aceeași inițializare, rezultatele pot diferi(!) dacă avem instanțe multiple / redundante, situate la egală distanță de 2 centroizi la o iteratie oarecare: ex. 12.b;
 - (P1'') rezultatele lui K -means sunt dependente [și] de măsura de distanță folosită: ex. 45;
 - K -means poate fi văzut și ca *algoritm de optimizare* — vedeti criteriul J_K de mai sus;
 - (P2) strategia de căutare / optimizare folosită de K -means este de tipul *descreștere pe coordinate* (engl., coordinate descent), i.e. descreștere iterativă, mergând alternativ pe fiecare din cele două coordonate ale criteriului $J_K(C^t, \mu^t)$: ex. 12.a;
 - (P2') algoritmul K -means nu garantează atingerea optimului global (i.e., minimul) criteriului J_K : ex. 12.b, ex. 43.b.
- ca algoritm de *învățare automată*:
 - [următor de] „generalizare“: o instanță nouă x se asociază clusterului având centroidul cel mai apropiat de x ;
 - (P3) „granițele“ de separare dintre [perechile de] clustere produse de K -means sunt [doar] liniare, [cel puțin] atunci când se folosește distanța euclidiană: ex. 11.b;
 - (P3') este însă posibil să se obțină separatori neliniari dacă se folosește o versiune „kerneлизată“ a algoritmului K -means: ex. 47;
 - (P4) rezultatele lui K -means pot fi influențate de prezența outlier-elor: ex. 10.
- chestiunea alegerii unei valori convenabile / „naturale“ pentru K (pentru un dataset dat): ex. 41 (și CMU, 2012f, E. Xing, A. Singh, HW3, ex. 1.de);
- adaptarea algoritmului K -means pentru cazul în care în locul distanței euclidiene se folosește distanța Manhattan: ex. 45;

- implementare: ex. 50.

7.2.3 Alte proprietăți ale algoritmului K -means

- în legătură cu criteriul definit mai sus, $J_K : \mathcal{P}_K \times (\mathbb{R}^d)^K \leftarrow [0, +\infty)$, unde \mathcal{P}_K este multimea tuturor K -partițiilor peste multimea de instanțe, $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{R}^d$:
 - (P5) pentru $K > 0$ fixat, $|\mathcal{P}_K| = K^n$, deci este finit, și există $\underline{J}_K \stackrel{\text{not.}}{=} \min_C J_K(C, \mu_C)$; acest minimum (\underline{J}_K) se poate obține prin explorarea exhaustivă a spațiului \mathcal{P}_K , însă consumul de timp este prohibitiv în practică: ex. 12.b;
 - (P6) valoarea 0 pentru \underline{J} este atinsă, și anume atunci când $K = n$, C este K -partiția de clustere singleton $C_i = \{x_i\}$, iar $\mu_i = x_i$, pentru $i = 1, \dots, n$ (ex. 41);
 - (P7) $\underline{J}_1 \geq \underline{J}_2 \geq \dots \geq \underline{J}_{n-1} \geq \underline{J}_n = 0$: ex. 13.
- (P8) dacă $d = 1$, deci $x_1, x_2, \dots, x_n \in \mathbb{R}$,
 - orice K -partiție (C_1, \dots, C_K) pentru care se atinge \underline{J}_K este de forma unei colecții de „intervale”: $C_1 = \{x_1, \dots, x_{i_1}\}$, $C_2 = \{x_{i_1+1}, \dots, x_{i_2}\}$, ..., $C_K = \{x_{i_{K-1}+1}, \dots, x_n\}$, cu $i_1 < i_2 < \dots < i_{K-1} < i_K = n$;
 - există un algoritm [de programare dinamică] de complexitate $\mathcal{O}(Kn^2)$ care calculează \underline{J}_K : ex. 43.
- în legătură cu J_K și algoritmul K -means:
 - (P9) $J_K(C^{t-1}, \mu^{t-1}) \geq J_K(C^t, \mu^t)$ la orice iterație ($t > 0$) a algoritmului K -means: ex. 12.a;
 - (P9') în consecință, dacă se impune restricția ca la fiecare iterare inegalitatea de mai sus să fie satisfăcută în varianta strictă ($J_K(C^{t-1}, \mu^{t-1}) > J_K(C^t, \mu^t)$), atunci algoritmul K -means termină într-un număr finit de pași;
 - (P10) în vreme ce minimizează *coezinea intra-clustere*, i.e. o variantă ponderată a „sumelor celor mai mici pătrate” calculate pe clustere,

$$\sum_{k=1}^K \frac{\sum_{i=1}^n \gamma_{ik} \|x_i - \mu_k\|^2}{\sum_{i=1}^n \gamma_{ik}},$$

unde $\gamma_{ik} = 1$ dacă x_i aparține clusterului de centroid μ_k și $\gamma_{ik} = 0$ în caz contrar, algoritmul K -means maximizează (în mod aproximativ!) o sumă ponderată a distanțelor dintre clustere:

$$\sum_{k=1}^K \left(\frac{\sum_{i=1}^n \gamma_{ik}}{n} \right) \|\mu_k - \bar{x}\|^2,$$

unde \bar{x} este media instanțelor x_1, x_2, \dots, x_n (ex. 42).

7.3. Clusterizare prin modelare probabilistă

7.3.1 Noțiuni preliminare

- variabile aleatoare (discrete, resp. continue);
 - media, varianța și covarianța variabilelor aleatoare;
- vector de variabile aleatoare; matrice de covarianță pentru un astfel de vector;
 - proprietăți: matricea de covarianță trebuie să fie în mod necesar simetrică și pozitiv definită: ex. 20 de la capitolul de *Fundamente*;

- distribuție (funcție de densitate) de probabilitate (p.d.f.); parametri ai unei distribuții de probabilitate; distribuția gaussiană: cazurile uni- și multidimensional;
- mixtură de distribuții probabiliste:
văzută ca o formă particulară de *combinație liniară* de distribuții de probabilitate $\pi_1\Psi_1 + \pi_2\Psi_2 + \dots + \pi_k\Psi_k$ (cu $\pi_i \geq 0$ și $\sum_{i=1}^k \pi_i = 1$), definită [și mai specific] scriind distribuția $P(X)$ ca o sumă ponderată de probabilități condiționate: $\sum_z P(X|Z)P(Z)$, unde X sunt variabilele „observabile“, iar variabila Z (eventual multiplă) poate fi „neobservabilă“ / „latentă“ / „ascunsă“; exemple: o mixtură de distribuții categoriale, respectiv o mixtură de distribuții Bernoulli: ex. 26 și ex. 103 de la capitolul de *Fundamente*; o mixtură de distribuții gaussiene multidimensionale: ex. 107 de la capitolul de *Fundamente*; o mixtură de distribuții oarecare: ex. 108 de la capitolul de *Fundamente*;
- funcție de *verosimilitate* a unui set de date (D), în raport cu o distribuție probabilistă dată: $L(\theta) = P(D|\theta)$, unde prin θ se notează parametrii respectivei distribuții. Exemplificare: ex. 40.abd, ex. 39 de la capitolul de *Fundamente*;
- MLE (Maximum Likelihood Estimation): estimarea [valorilor] parametrilor unei distribuții probabiliste în sensul maximizării verosimilității datelor disponibile. Exemplificare: capitolul de *Fundamente*, ex. 40-49, ex. 113-123. Aplicare în cazul distribuției gaussiene unidimensionale: ex. 15.ab de la capitolul *Clasificare bayesiană*;
- analiza discriminativă gaussiană: ex. 39 de la capitolul *Clasificare bayesiană*;
- Observație: Algoritmul EM este [sau, mai degrabă, poate fi folosit ca] o metodă de estimare a parametrilor unei mixturi de distribuții probabiliste. *Alternativ*, pentru același obiectiv pot fi folosite alte metode, de exemplu *metoda gradientului ascendent*: ex. 63.

7.3.2 Algoritmul EM pentru clusterizare prin estimarea parametrilor unui model de mixturi de distribuții gaussiene (EM/GMM)

- pseudo-cod:⁴²⁶
cazul unidimensional, varianta când doar parametrul μ este lăsat liber: ex. 15 (cf. *Machine Learning*, Tom Mitchell, 1997, pag. 193); aplicare: ex. 17.b, ex. 16, ex. 51;
cazul unidimensional, varianta când toți parametrii (π , μ și σ) sunt lăsați liberi: ex. 18 (aplicare: ex. 17.c);
alte variante: [ex. 19] ex. 53, ex. 52;
cazul multidimensional, varianta când toți parametrii (π , μ și Σ) sunt lăsați liberi: ex. 20;
alte variante: ex. 56, ex. 58;
aplicarea algoritmului EM/GMM, cazul bidimensional: ex. 21.b, ex. 22.b, ex. 23, ex. 24, ex. 59, ex. 60, ex. 61;
- schema algoritmică EM: vedeti Tom Mitchell, *Machine Learning* book, 1997, pag. 194-195; ex. 19;
- ca algoritm de *învățare statistică*:
algoritmul EM poate fi văzut ca o metodă de estimare a parametrilor (engl., parameter fitting);

⁴²⁶Nu doar pentru pseudo-cod, ci și (sau, mai ales) pentru o privire de ansamblu unitară, atât pentru cazul unidimensional cât și pentru cazul multidimensional (ambele urmând a fi sistematizate mai jos), puteți consulta documentul *An Introduction to Expectation-Maximization*, de Dahua Lin, MIT, ML course 6768, 2012 fall.

- ca algoritm *per se*:
 - *algoritm iterativ*: pleacă de la o soluție (instantiere pentru parametri) aleasă eventual în mod arbitrar / aleatoriu și o „îmbunătățește“ la fiecare iterație;
 - *algoritm de optimizare*:

în *esență / rezumat*, metoda de maximizare a funcției de *log-verosimilitate a datelor observabile* $\log P(X|\theta)$ este maximizarea la fiecare iterație t a unei funcții auxiliare Q_t , care constituie o margine inferioară a lui $\log P(X|\theta)$, și anume media funcției de *log-verosimilitate a datelor complete* în raport cu distribuția de probabilitate a *variabilelor neobservabile* la iterarea t ;

mai precis, la fiecare iterare t se calculează funcția „auxiliară“ $Q_t(\theta|\theta^{(t)})$, care reprezintă media funcției de log-verosimilitate a datelor „complete“ (cele „observabile“ plus cele „ne-observabile“), unde $\theta^{(0)}$, constând din valorile inițiale ale parametrilor mixturii (θ), se alege în mod arbitrar, iar apoi $\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q_t(\theta|\theta^{(t)})$;

media reprezentată de funcția Q_t se calculează în funcție de distribuțiile condiționale ale variabilelor „neobservabile“ Z în raport cu datele observabile X și cu $\theta^{(t)}$;

(P0) Se poate demonstra că funcția Q_t constituie o *margine inferioară* pentru funcția de log-verosimilitate a variabilelor „observabile“, $\log P(X|\theta)$: ex. 1 de la capitolul *Algoritmul EM*;

(P1) *Teorema de corectitudine* (vedeți ex. 1 și în special ex. 2 de la capitolul *Algoritmul EM*) pe de o parte garantează faptul că la fiecare iterare a algoritmului EM, log-verosimilitatea datelor „observabile“, $\log P(X|\theta^{(t)})$ nu descrește (ci fie crește, fie rămâne neschimbăță),
dar pe de altă parte nu garantează găsirea optimului global al funcției de log-verosimilitate a datelor „observabile“, $\log P(X|\theta)$, ci eventual a unui optim local.
- ca algoritm de *învățare automată*:
 - algoritmul EM este o metodă de identificare / învățare de ipoteze ML (Maximum Likelihood); vedeți capitolul / secțiunea 6.4 din carte *Machine Learning*;
 - învățare în prezența unor variabile aleatoare neobservabile(!);
 - [urmată eventual de] „generalizare“: o instantă nouă x se asociază clusterului (i.e., distribuției) j pentru care se atinge $\max_{j'} P(X = x|h_{j'})P(h_{j'})$;

(P2) Rezultatele algoritmului EM depind (ca și la *K-means*) de valorile atribuite parametrilor la inițializare (ex. 17.c);

(P3) Anumite valori atribuite inițial parametrilor algoritmului EM pot provoca rularea la infinit a algoritmului, fără ca [la pasul M] valorile parametrilor să se modifice de la o iterare la alta: ex. 19.c;

(P4) Spre deosebire de cazul algoritmului *K-means*, suprafețele / granițele de separare create de algoritmul EM/GMM nu sunt în mod neapărat liniare (vedeți de exemplu situațiile întâlnite la rezolvarea ex. 17.c, pag. 567, sau la ex. 61.c și ex. 62.c).
- Comparativ cu algoritmul *K-means*,
 - (P5) algoritmul EM/GMM este în general mai lent — mișcarea centroizilor poate explora într-o manieră mai fină spațiul (vedeți de exemplu ex. 21) —, iar din acest motiv el poate să obțină uneori rezultate mai bune / convenabile (vedeți spre exemplu ex. 22);
 - (P6) Apare un fenomen de “atracție” reciprocă a mediilor gaussianelor (aceste medii fiind echivalentul centroizilor din algoritmul *K-means*), datorită faptului că fiecare instantă aparține (cu o anumită probabilitate) la fiecare cluster. Atracția mediilor este cu atât

mai puternică cu cât varianțele sunt mai mari. (Vedeți spre exemplu ex. 17.b);

(P7) EM/GMM este mai robust la influența outlier-elor.

7.3.3 Alte proprietăți ale algoritmului EM/GMM

- Pentru distribuții gaussiene multidimensionale:
 - (P8) în cazul cel mai general (deci când matricea Σ nu este neapărat diagonală), datele generate de acest tip de distribuție se grupează în elipse (corpuri elipsoidale) cu axe de simetrie [desigur, perpendiculare, dar altfel] nerestricționate;
 - (P8') dacă matricea de covarianță Σ este diagonală, atunci
 - i. distribuția gaussiană respectivă este echivalentă cu un set / vector de variabile gaussiene unidimensionale independente: ex. 31 de la capitolul de *Fundamente*;
 - ii. datele generate se grupează în elipse (sau: corpuri elipsoidale) având axe de simetrie paralele cu axele sistemului de coordinate;
 - (P8'') dacă matricea Σ este de forma $\sigma^2 I$, unde I este matricea identitate, datele generate de respectiva distribuție tind să se grupeze în sfere;
- (P9) Legătura dintre algoritmul K -means și algoritmul EM/GMM (cazul multidimensional):
 - atunci când $\Sigma = \sigma^2 I$, iar $\sigma^2 \rightarrow 0$ (și sunt satisfăcute încă două restricții), algoritmul EM/GMM tinde să se comporte ca și algoritmul K -means: ex. 58;
- O legătură interesantă între algoritmul EM/GMM și metoda gradientului ascendent, în cazul în care matricele de covarianță sunt de forma $\sigma_k^2 I$: ex. 63;
- O legătură interesantă între clasificatorul Bayes Naïv gaussian și algoritmul EM/GMM în cazul în care matricele de covarianță sunt de forma $\sigma_k^2 I$:
 - variantă *semisupervizată* a algoritmului EM/GMM: ex. 64.

7.1 Probleme rezolvate

7.1.1 Clusterizare ierarhică

1. (Clusterizare ierarhică aglomerativă: exemplificare pe date din \mathbb{R} , folosind similaritate de tip “single-linkage”)

CMU, 2006 spring, Carlos Guestrin, HW5, pr. 1

a. Desenați *dendrograma* (adică arborele de clusterizare ierarhică) pentru următoarea mulțime de 10 puncte pe axa reală:

$$S = \{-2.2, -2.0, -0.3, 0.1, 0.2, 0.4, 1.6, 1.7, 1.9, 2.0\}$$

Folosiți *similaritate* de tip “single-linkage”, adică $d(C_i, C_j) = \min_{x \in C_i, x' \in C_j} |x - x'|$.

Observație: În dendrogramă, înălțimea (h) corespunzătoare nodului rădăcină al unui cluster va fi considerată direct proporțională cu media aritmetică a distanțelor dintre punctele

din clusterul respectiv,⁴²⁷ iar *coeziunea* (c) clusterului va fi definită aici ca inversul acestei medii.⁴²⁸

b. Bazat pe arborele obținut, justificați — în manieră informală — faptul că 3 este numărul natural de clustere.

Răspuns:

a. Numerotăm cele 10 puncte date în ordinea crescătoare a valorilor lor: $x_1 = -2.2, x_2 = -2, x_3 = -0.3, x_4 = 0.1, x_5 = 0.2, x_6 = 0.4, x_7 = 1.6, x_8 = 1.7, x_9 = 1.9, x_{10} = 2.0$.

Dendrograma mulțimii de instanțe date se construiește în manieră *bottom-up* astfel:

- $d(x_4, x_5) = d(x_7, x_8) = d(x_9, x_{10}) = 0.1 = \min_{1 \leq i < j \leq 10} d(x_j, x_i)$, prin urmare primele 3 clustere care se vor forma sunt $C_1 = \{x_4, x_5\}, C_2 = \{x_7, x_8\}, C_3 = \{x_9, x_{10}\}$. În mod evident, $h(C_1) = h(C_2) = h(C_3) = 0.1$ iar $c(C_1) = c(C_2) = c(C_3) = 1/0.1 = 10$.

- Avem:

$$\begin{aligned} d(x_1, x_2) &= 0.2 \\ d(C_1, x_6) &= d(x_5, x_6) = 0.2 \\ d(C_2, C_3) &= d(x_8, x_9) = 0.2 \end{aligned}$$

Celelalte distanțe sunt toate mai mari decât 0.2, aşadar următorul nivel al arborelui va fi alcătuit din clusterele $C_4 = \{x_1, x_2\}, C_5 = \{x_4, x_5, x_6\}$ și $C_6 = \{x_7, x_8, x_9, x_{10}\}$.

Făcând calculele, obținem: $h(C_4) = h(C_5) = 0.2$ și $h(C_6) = \frac{1.4}{6} = 0.2(3)$, iar $c(C_4) = c(C_5) = 1/0.2 = 5$ și $c(C_6) = \frac{6}{1.4} \approx 4.285$.

- Avem:

$$\begin{array}{ll} d(C_4, x_3) = d(x_2, x_3) = 1.7 & d(C_4, C_5) = d(x_2, x_4) = 2.1 \\ d(C_5, x_3) = d(x_4, x_3) = 0.4 & d(C_4, C_6) = d(x_2, x_7) = 3.6 \\ d(C_6, x_3) = d(x_7, x_3) = 1.9 & d(C_5, C_6) = d(x_6, x_7) = 1.2 \end{array}$$

Așadar, pe următorul nivel al ierarhiei se va afla clusterul $C_7 = C_5 \cup \{x_3\} = \bigcup_{3 \leq i \leq 6} \{x_i\}$. Înălțimea și respectiv coeziunea acestui cluster sunt: $h(C_7) = \frac{2.2}{6} = 0.3(6)$ și $c(C_7) = \frac{6}{2.2} = 2.(72)$.

- Avem:

$$\begin{aligned} d(C_4, C_6) &= d(x_2, x_7) = 3.6 \\ d(C_4, C_7) &= d(x_2, x_3) = 1.7 \\ d(C_6, C_7) &= d(x_6, x_7) = 1.2 \end{aligned}$$

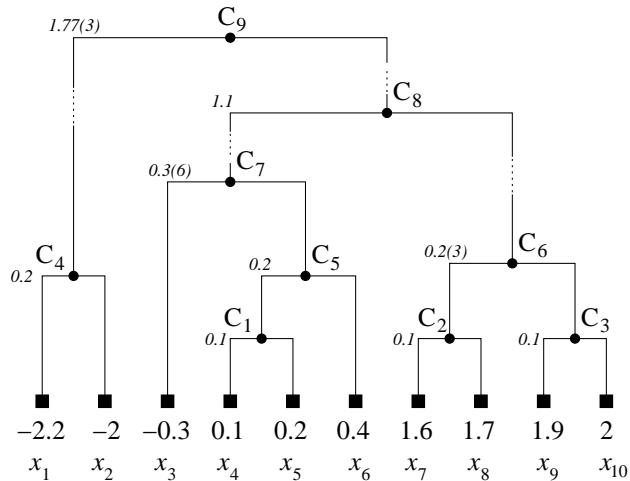
În consecință, pe următorul nivel al ierarhiei se va afla clusterul $C_8 = C_6 \cup C_7 = \bigcup_{3 \leq i \leq 10} \{x_i\}$. Făcând calculele, obținem $h(C_8) = \frac{30.8}{28} = 1.1$ și $c(C_8) = 1/1.1 = 0.9(09)$.

- Pentru clusterul $C_9 = C_4 \cup C_8$, avem $h(C_9) = \frac{79.8}{45} = 1.77(3)$ și $c(C_9) \approx 0.564$.

⁴²⁷În rezolvarea acestei probleme, factorul de proporționalitate va fi considerat 1.

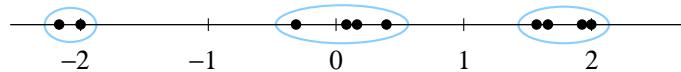
⁴²⁸Evident, această definiție pentru coeziune are sens doar în cazul clusterelor non-singleton.

Dendrograma multimii date va arăta deci astfel:



b. După alcătuirea dendrogramei, împărțirea multimii de instanțe date în clustere cu coeziune comparabilă se face „tăind“ dendrogramea cu o linie paralelă cu baza. Împărțirea în clustere fine corespunde liniilor paralele apropiate de bază, iar împărțirea în clustere ample corespunde liniilor paralele apropiate de vârful / rădăcina dendrogramei.

În cazul nostru, se observă că pentru o mare parte a înălțimii dendrogramei (și anume între 0.3(6) și 1.1), rezultatul operației descrise mai sus rămâne neschimbăt (multimea dată descompunându-se corespunzător în clusterele C_4, C_7 și C_8), ceea ce sugerează că „numărul natural“ de clustere este 3. Acest lucru se poate vedea și dacă reprezentăm cele 10 puncte pe axa reală:



2. (Clusterizare ierarhică aglomerativă: exemplificare pe date din \mathbb{R}^2 , folosind tipurile de similaritate “single-linkage” și “complete-linkage”)

Edinburgh, 2009 fall, C. Williams, V. Lavrenko, HW4, pr. 1

Considerăm următorul set de date, în care fiecare instanță (x, y) este un punct în plan. Pentru conveniență, în cele ce urmează, vom identifica aceste instanțe prin literele asociate punctelor respective.

$$\begin{aligned} A &: (-4, -2), B : (-3, -2), C : (-2, -2), D : (-1, -2), E : (+1, -1) \\ F &: (+1, +1), G : (+2, +3), H : (+3, +2), I : (+3, +4), J : (+4, +3) \end{aligned}$$

a. Reprezentați grafic datele din enunț.

b. Realizați clusterizarea ierarhică a datelor în maniera bottom-up, folosind similaritate de tip “single-linkage” și distanță euclidiană între puncte.

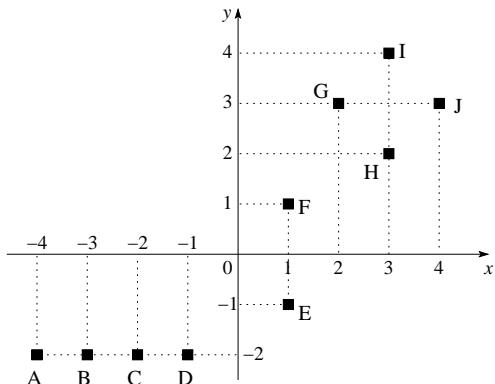
Observație: Dacă la o iterație a algoritmului de clusterizare distanțele (adică similaritățile) dintre două perechi de clustere au aceeași valoare, prioritarea la alcătuirea noului cluster este dictată de ordinea alfabetică.

c. Realizați clusterizarea datelor, folosind de această dată similaritate de tip “complete-linkage”.

d. Discutați diferența dintre clusterizările obținute în urma folosirii celor două tipuri de [măsuri de] similaritate.

Răspuns:

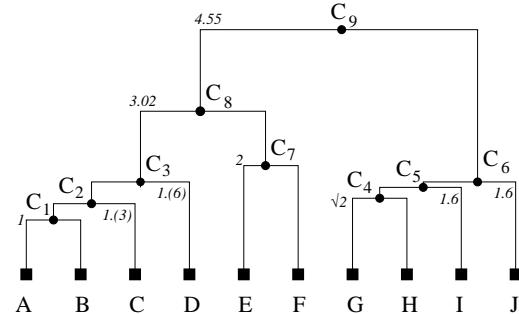
- a. Reprezentarea în plan a celor 8 puncte date în enunț este cea din figura alăturată.



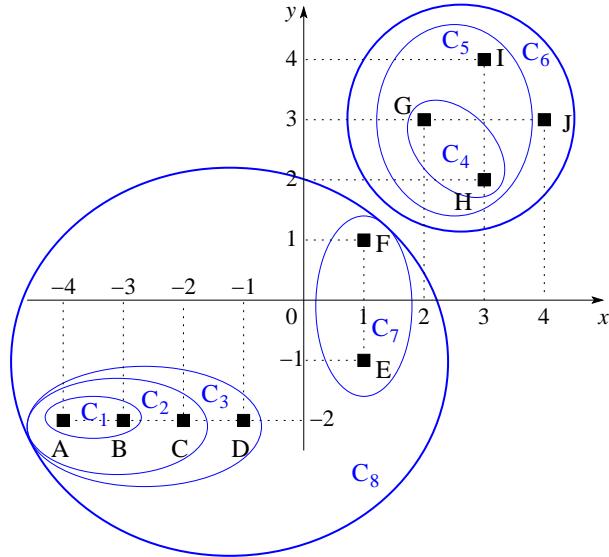
b. Clusterizarea folosind similaritate de tip “single-linkage” (pentru care distanța) dintre două clustere este definită ca fiind distanța dintre cele mai apropiate două puncte, câte unul din fiecare cluster) se realizează astfel:

- Minimul distanțelor mutuale dintre punctele date este 1. Acest minim este obținut pentru mai multe perechi de puncte, deci se va recurge la ordinea alfabetică pentru a stabili prioritarea. Prin urmare, primul cluster format va fi $C_1 = \{A, B\}$. Evident, $h(C_1) = c(C_1) = 1$, unde h și c sunt înăltîmirea asociată clusterului C_1 , respectiv coeziunea lui, calculate după aceeași metodă ca la problema 1.
- Acum, distanța minimă este obținută (ținând cont și de regula alfabetică) între clusterul C_1 și punctul C , deci $C_2 = C_1 \cup \{C\} = \{A, B, C\}$. Prin urmare, $h(C_2) = 4/3 = 1.(3)$, iar $c(C_2) = 3/4 = 0.75$.
- În mod similar, tot la distanță 1, vom avea $C_3 = C_2 \cup \{D\} = \{A, B, C, D\}$, iar $h(C_3) = 10/6 = 1.(6)$ și $c(C_3) = 6/10 = 0.6$.
- La acest nou pas se consideră clusterul C_3 și punctele E, F, \dots, J . Distanța minimă este $\sqrt{2}$, și anume — având din nou în vedere ordinea alfabetică — între punctele G și H , deci $C_4 = \{G, H\}$, cu $h(C_4) = \sqrt{2} \approx 1.414$ și $c(C_4) = 1/\sqrt{2} \approx 0.707$.
- Apoi $C_5 = C_4 \cup \{I\} = \{G, H, I\}$ și $C_6 = C_5 \cup \{J\} = \{G, H, I, J\}$, clustere care au înăltîmile (egale!) $h(C_5) = h(C_6) = \frac{2}{3}(\sqrt{2} + 1) \approx 1.609$ și, deci, și coeziunile egale: $c(C_5) = c(C_6) = \frac{3}{2} \frac{1}{\sqrt{2} + 1} \approx 0.621$.
- Acum se consideră clusterele C_3 și C_6 și punctele E și F . Distanța minimă este 2, și anume cea dintre punctele E și F , deci $C_7 = \{E, F\}$, cu $h(C_7) = 2$ și $c(C_7) = 0.5$.

- Între clusterele C_3, C_6 și C_7 , distanța minimă este $\sqrt{5}$, deci $C_8 = C_3 \cup C_7 = \{A, B, C, D, E, F\}$, cu $h(C_8) \approx 3.02$ și $c(C_8) \approx 0.331$.
- Rămâne în final $C_9 = C_8 \cup C_6$, cu $h(C_9) \approx 4.552$ și $c(C_9) \approx 0.219$.



Dendrograma obținută este redată în figura alăturată.



Varianta „aplatizată“ a acestei ierarhii (engl., flat hierarchy) este prezentată în desenul alăturat.

c. Clusterizarea ierarhică de tip “complete-linkage” (pentru care similaritatea dintre două clustere este dată de distanța dintre cele mai depărtate două puncte, câte unul din fiecare cluster) se realizează astfel:

- Minimul distanțelor dintre oricare două puncte este 1, iar acest minim este obținut pentru mai multe perechi de puncte. Luând în considerare ordinea alfabetică, primul cluster format va fi $C_1 = \{A, B\}$, cu $h(C_1) = c(C_1) = 1$.
- La acest nou pas, distanța minimă este $d(C, D) = 1$. Deci, spre deosebire de cazul “single-linkage”, vom avea $C_2 = \{C, D\}$, cu $h(C_2) = c(C_2) = 1$.
- În continuare, cea mai mică distanță este $\sqrt{2}$, între G și H , deci $C_3 = \{G, H\}$, cu $h(C_3) = \sqrt{2} \approx 1.414$ și $c(C_3) = 1/\sqrt{2} \approx 0.707$.
- Tot $\sqrt{2}$ este și distanța dintre I și J , deci $C_4 = \{I, J\}$, cu $h(C_4) = \sqrt{2}$ și $c(C_4) = 1/\sqrt{2}$.

- Apoi, $C_5 = \{E, F\}$, între aceste două puncte distanța fiind 2, deci $h(C_5) = 2$ și $c(C_5) = 0.5$.
- În acest moment se consideră clusterele C_1, C_2, C_3, C_4 și C_5 . Distanțele dintre ele sunt:

$$\begin{aligned} d(C_1, C_2) &= d(A, D) = 3 \\ d(C_1, C_3) &= d(A, H) = 8.06 \\ d(C_1, C_4) &= d(A, J) = 9.43 \\ d(C_1, C_5) &= d(A, F) = 5.83 \\ d(C_2, C_3) &= d(C, H) = 6.4 \end{aligned}$$

$$\begin{aligned} d(C_2, C_4) &= d(C, I) = d(C, J) = 7.81 \\ d(C_2, C_5) &= d(C, F) = 4.24 \\ d(C_3, C_4) &= d(G, J) = d(H, I) = 2 \\ d(C_3, C_5) &= d(G, E) = 4.12 \\ d(C_4, C_5) &= d(E, I) = 5.39 \end{aligned}$$

Luând minimul acestor distanțe, și anume $d(C_3, C_4) = 2$, se obține $C_6 = C_3 \cup C_4 = \{G, H, I, J\}$, cu $h(C_6) = \frac{2}{3}(\sqrt{2} + 1) \approx 1.609$ și $c(C_6) \approx 0.621$.

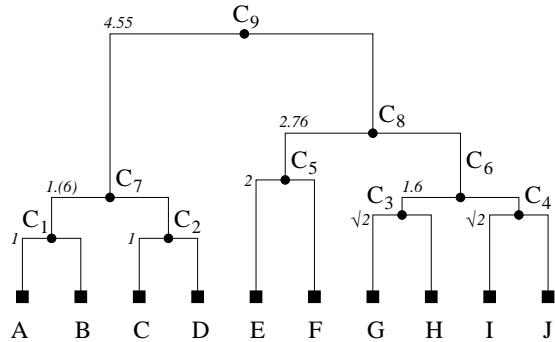
- Urmează $C_7 = C_1 \cup C_2 = \{A, B, C, D\}$, cu $h(C_7) = 10/6 = 1.(6)$ și $c(C_7) = 0.6$.
- Între C_5, C_6 și C_7 , distanțele sunt:

$$\begin{aligned} d(C_5, C_6) &= d(E, I) = 5.39 \\ d(C_5, C_7) &= d(F, A) = 5.83 \\ d(C_6, C_7) &= d(I, A) = 9.43 \end{aligned}$$

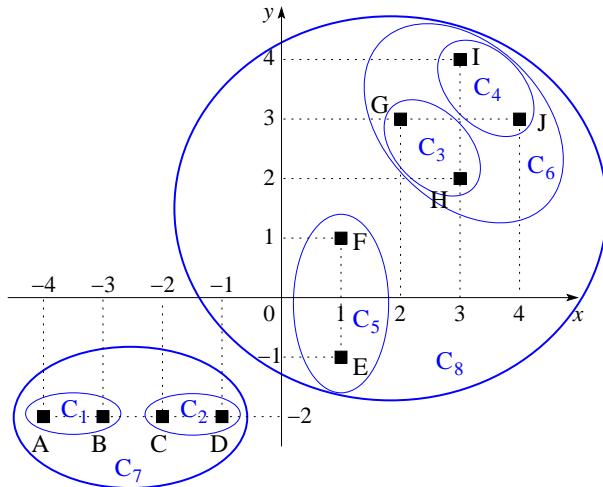
Întrucât cea mai mică dintre aceste distanțe este 5.39, rezultă $C_8 = C_5 \cup C_6 = \{E, F, G, H, I, J\}$, cu $h(C_8) \approx 2.76$ și $c(C_8) \approx 0.36$.

- Rămâne în final $C_9 = C_7 \cup C_8$, distanța dintre cele două clustere constituente fiind de 9.43. Înălțimea și coeziunea clusterului C_9 sunt respectiv 4.55 și 0.21.

Dendrograma rezultată este cea din figura alăturată.



Varianta „aplatizată“ a acestei ierarhii (engl., flat hierarchy) este prezentată în desenul alăturat.



- d. Clusterizarea ierarhică folosind similaritate de tip “single-linkage” are tendința de a înlăntui / alungi clusterele, spre deosebire de cea “complete-linkage” care grupează clusterele mai degrabă sub formă sferică.

3. (Clusterizare ierarhică aglomerativă: exemplificare în \mathbb{R} , folosind tipurile de similaritate “single-linkage” și “complete-linkage”, comparativ cu “average-linkage”)

prelucrare făcută de Liviu Ciortuz, după CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 9.1

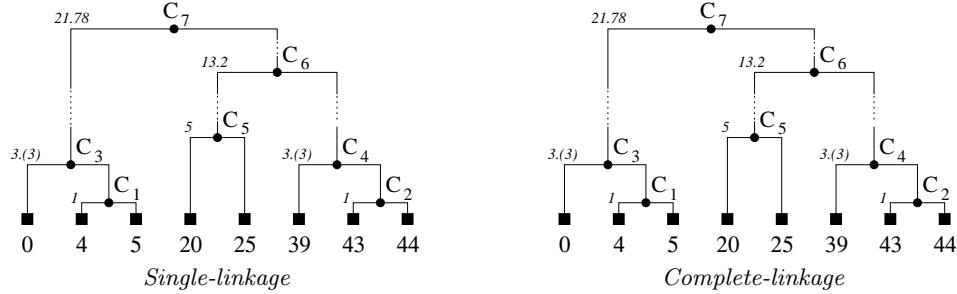
Considerăm următorul set de date: $\{0, 4, 5, 20, 25, 39, 43, 44\}$. Presupunem că vrem să obținem cele mai importante 2 clustere din dendrogramă (cele situate imediat sub nodul rădăcină), numite în continuare *top-clustere*.

- a. Care sunt cele două top-clustere atunci când se folosesc similaritate *single-linkage*, *complete-linkage*, și respectiv *average-linkage*?
b. Dacă *single-linkage* și *complete-linkage* produc rezultate identice (relativ la compoziția [top]-clusterelor), rezultă că *average-linkage* va conduce și el la același rezultat?

Răspuns:

- a. Dendrogramele corespunzătoare similarităților *single-linkage* și *complete-linkage* sunt următoarele:⁴²⁹

⁴²⁹Calculele sunt absolut similare celor de la rezolvarea problemei 2.



- b. Se observă că atât în cazul *single-linkage* cât și în cazul *complete-linkage*, cele două top-clustere sunt $\{0, 4, 5\}$ și $\{20, 25, 39, 43, 44\}$. Mai mult, cele două dendrograme sunt identice.

Clusterizarea ierarhică cu similaritate *average-linkage* folosește ca „distanță” între două clustere oarecare media aritmetică a distanțelor calculate pentru toate perechile de puncte ce se pot forma luând un punct dintr-un cluster și un punct din celălalt cluster. Aplicarea acestui algoritm se realizează astfel:

- $C_1 = \{4, 5\}$ și $C_2 = \{43, 44\}$, ambele constituite din câte o pereche de clustere “singleton” aflate la distanță 1. Înălțimile clusterelor rezultate sunt $h(C_1) = h(C_2) = 1$.⁴³⁰
- $C_3 = \{0\} \cup C_1 = \{0, 4, 5\}$, distanța dintre clusterul “singleton” $\{0\}$ și clusterul C_1 fiind $\frac{4+5}{2} = 4.5$. Rezultă $h(C_3) = \frac{4+5+1}{3} = 3.(3)$.
- $C_4 = \{39\} \cup C_2 = \{39, 43, 44\}$. Distanța dintre clusterul $\{39\}$ și clusterul C_2 este $\frac{(43-39)+(44-39)}{2} = \frac{4+5}{2} = 4.5$. Înălțimea noului cluster este $h(C_4) = 3.(3)$.
- $C_5 = \{20, 25\}$, cu distanța dintre clusterele “singleton” constitutive 5 și înălțimea rezultantă tot 5.
- În acest moment există 3 clustere: $C_3 = \{0, 4, 5\}$, $C_4 = \{39, 43, 44\}$ și $C_5 = \{20, 25\}$. Sunt relevante două dintre cele trei distanțe mutuale dintre aceste clustere, și anume:

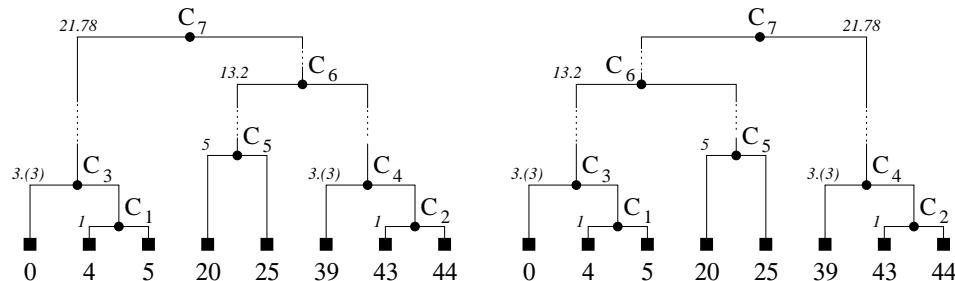
$$d(C_3, C_5) = \frac{20 + 25 + 16 + 21 + 15 + 20}{6} = \frac{117}{6} = 19.5$$

$$d(C_4, C_5) = \frac{19 + 14 + 23 + 18 + 24 + 19}{6} = \frac{117}{6} = 19.5$$

Se observă că aceste două distanțe sunt egale. Deoarece în enunț nu s-a specificat — pentru un astfel de caz — vreo regulă de prioritate de care să se țină cont atunci când se combină două clustere,⁴³¹ cele două top-clustere care se pot obține în cazul nostru sunt fie $\{0, 4, 5\}$ și $\{20, 25, 39, 43, 44\}$ — adică exact ca la punctul a —, fie $\{0, 4, 5, 20, 25\}$ și $\{39, 43, 44\}$. Acest fapt este ilustrat în cele două dendrograme de mai jos:

⁴³⁰Pentru calculul înălțimilor am procedat ca la problemele 1 și 2.

⁴³¹Vedeți regulile folosite la problemele 1 și 2.



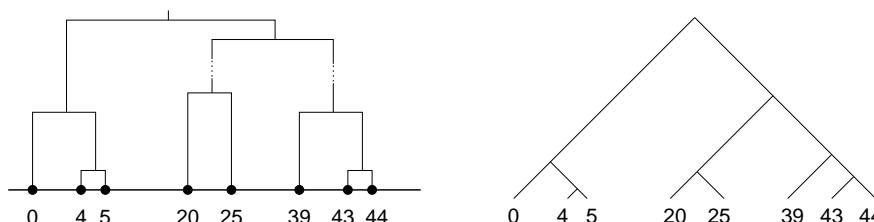
Concluzie: Chiar dacă, lucrând pe un același set de date, la clusterizare ierarhică folosind similaritate *single-linkage* și respectiv *complete-linkage* se obțin rezultate identice, este posibil ca atunci când se folosește *average-linkage* să obținem un alt rezultat.

4.

(Clusterizare ierarhică aglomerativă, folosind similaritate “single-linkage”: exemplificare pe date din \mathbb{R} ; corespondența cu numărul maxim / minim de niveluri din arborele clasic corespunzător dendrogramei)

CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW3, pr. 5.2

În acest exercițiu veți folosi metoda clusterizării ierarhice aglomerative cu similaritate de tip “single-linkage” la compararea a două clustere oarecare.⁴³² Vom defini *înălțimea* unei ierarhii ca fiind $l - 1$, unde l este numărul de niveluri din arborele (în sens clasic) corespunzător acestei ierarhii. De *exemplu*, pentru numerele 0, 4, 5, 20, 25, 39, 43, 44, ierarhia “single-linkage” (arătată în desenul de mai jos, partea stângă) are înnălțimea 4. Pentru convenientă (și pentru a elimina posibilele confuzii datorate înnălțimilor diferitelor noduri din dendrogramă), am desenat și arborele clasic care corespunde acestei ierarhii (vedeți desenul din partea dreaptă); evident, el are 5 niveluri.



a. Care este înnălțimea maximă a unei ierarhii care se poate construi cu N puncte? Dar înnălțimea minimă?

b. Dați un *exemplu* de 16 puncte din mulțimea numerelor întregi care la acest tip de clusterizare produce (i.) o ierarhie de înnălțime maximă, (ii.) o ierarhie de înnălțime minimă.

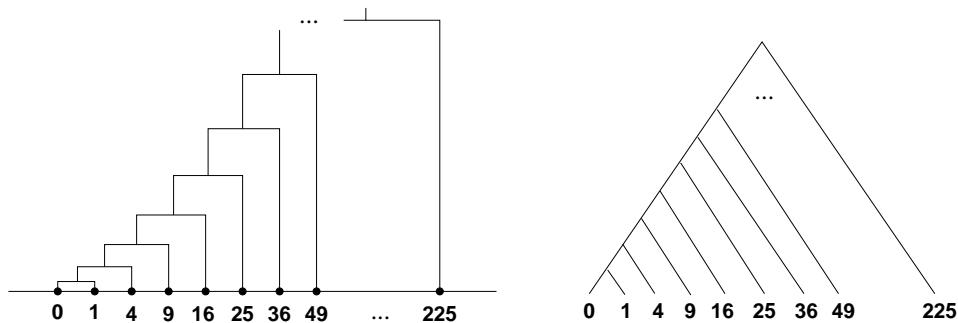
Răspuns:

⁴³²În acest caz, distanța dintre două clustere este definită ca minimul distanțelor $d(x, y)$, unde x aparține primului cluster, iar y aparține celui de-al doilea cluster.

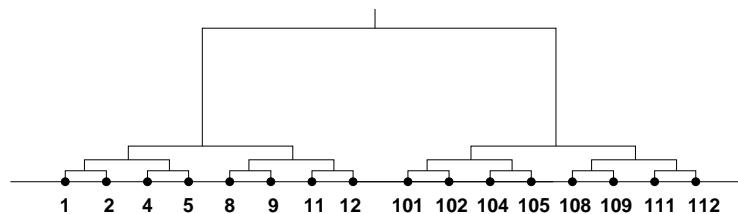
a. Pentru a avea o ierarhie de înălțime maximă, putem considera spre exemplu cazul în care distanțele dintre punctele care se clusterizează succesiv sunt în ordine strict crescătoare, obținând un arbore cu N niveluri. Deci înălțimea maximă a unei ierarhii este $N - 1$.

Pentru a avea înălțime minimă, trebuie ca arborele corespunzător să fie pe cât posibil un arbore binar echilibrat. Deci înălțimea minimă este partea întreagă superioară din $\log_2(N)$.

b. O mulțime de 16 puncte care conduce la o ierarhie de tip “single-linkage” de înălțime maximă este: 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225. Ierarhia este următoarea (am folosit ambele variante de reprezentare grafică):



Pentru o ierarhie de tip “single-linkage” de înălțime minimă putem considera următoarele 16 puncte: 1, 2, 4, 5, 8, 9, 11, 12, 101, 102, 104, 105, 108, 109, 111, 112. Ierarhia este următoarea:



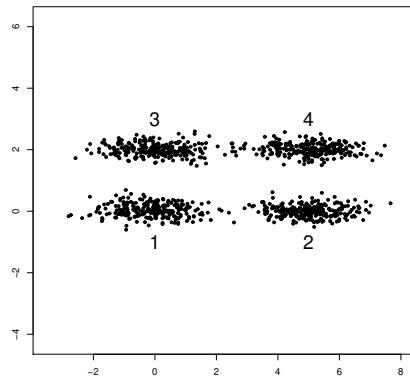
5.

(Clusterizare ierarhică: aplicare în manieră intuitivă pe date din \mathbb{R}^2)

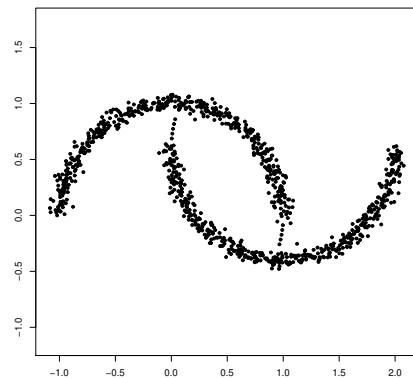
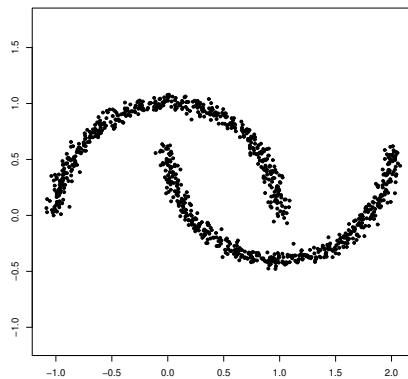
CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1.2.bc

a. Considerăm datele din figura de mai jos.

Ce rezultat vom obține dacă extragem cele două clustere de “top” din arborele obținut prin clusterizare ierarhică pe acest set de date folosind măsura de similaritate “single-linkage”? (Formulați răspunsul raportându-vă la etichetele 1–4 care identifică cele patru grupuri din date.) Procedați similar pentru “complete-linkage” și “average-linkage”.



- b. Poate vreuna dintre cele trei măsuri de similaritate să separe cu succes cele două „semiluni“ din figura de mai jos, partea stângă? Dar în ce privește figura din partea dreaptă? Justificați pe scurt.



Răspuns:

- a. „Single-linkage” va asigna aglomerările de puncte notate cu 1 și 2 la un cluster, iar pe cele notate cu 3 și 4 la celălalt cluster. „Complete-linkage” și „average-linkage” vor asigna aglomerările de puncte notate cu 1 și 3 la un cluster, iar pe cele notate cu 2 și 4 la alt cluster.
- b. „Single-linkage” va reuși să separe cu succes cele două semiluni din figura din partea stângă (din enunț), în vreme ce „complete-linkage” și „average-linkage” nu vor reuși. Niciuna dintre cele trei metode nu va avea succes pe datele din figura din partea dreaptă.

6. (Un algoritm de clusterizare ierarhică divizivă: aplicare; comparație cu variantele algoritmului de clusterizare aglomerativă; echivalență cu algoritmii de aflare a MST din teoria grafurilor)

*prelucrare făcută de Liviu Ciortuz, după
■ CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 9.3*

Clusterizarea ierarhică poate fi de două tipuri: *divizivă* (top-down) sau *aglomerativă* (bottom-up). Întrebarea la care am dori să răspundem prin această problemă este dacă un algoritm din categoria top-down poate să fie echivalent (relativ la rezultatul obținut ca atare, sau chiar la modul efectiv de elaborare a clusterizării) cu un algoritm din categoria bottom-up.

Să considerăm următorul *algoritm de clusterizare de tip top-down*:

- *Intrare*: o mulțime de instanțe $S = \{x_1, x_2, \dots, x_n\}$ și o măsură de distanță definită pe $S \times S$;
- *Ieșire*: o dendrogramă, adică un arbore de clusterizare binară în ale căruia noduri frunză sunt elementele din S , satisfăcând proprietățile dezirabile pentru clusterizare ierarhică (i.e., coezione cât mai mare în interiorul fiecărui cluster și distanțe maxime între clustere, pe fiecare nivel din dendrogramă);
- *Procedură*:

Pasul 1: Se calculează distanțele dintre oricare două puncte x_i, x_j din setul de date (S). Apoi se construiește graful neorientat, complet și ponderat, având ca mulțime de noduri chiar mulțimea S (așadar, fiecare nod din graf este reprezentat de o instanță $x_i \in S$), iar ponderea / costul fiecărei muchii din graf este exact distanța dintre punctele reprezentate de nodurile adiacente acestei muchii.

Pasul 2: Se calculează un arbore de acoperire de cost minim (engl., Minimum Spanning Tree) corespunzător grafului obținut la *Pasul 1*. Această operație revine la alegerea unei submulțimi de muchii din graful complet care *i.* constituie un arbore T ce folosește / conexează toate nodurile din graf, iar *ii.* suma costurilor / lungimilor muchiilor sale este minimă (în raport cu toți arborii care satisfac condiția precedentă). Acest arbore se poate obține folosind *algoritmul lui Kruskal*⁴³³ sau *algoritmul lui Prim*⁴³⁴.

Pasul 3: Se elimină din arborele de cost minim obținut la *Pasul 2* muchia cu costul maxim, obținându-se astfel doi arbori. Aceștia vor corespunde celor două clustere de pe nivelul cel mai de sus în dendrograma pe care o construim în manieră top-down.

Pasul 4: Se repetă recursiv *Pasul 3* atât timp cât este posibil, obținând astfel o clusterizare de tip top-down pe mulțimea de instanțe date.

⁴³³Publicat în 1956 în *Proceedings of American Mathematical Society*, pag. 48-50.

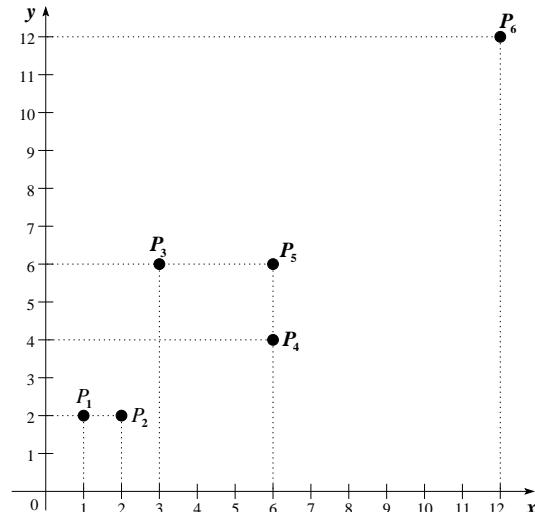
⁴³⁴Algoritm descoperit de Vojtech Jarnik în 1930, redescoperit de Robert Prim în 1957 și din nou de către Edsger Dijkstra în 1959. Din această cauză, el este numit uneori în literatura de specialitate *algoritmul DJP*. Atât acest algoritm cât și algoritmul lui Kruskal au complexitatea $O(n \log n)$.

- a. Aplicați acest algoritm pe setul de date din \mathbb{R}^2 din tabelul alăturat, folosind distanța euclidiană. Desenați rezultatul sub forma unei ierarhii „aplatizate“ sau ca dendrogramă (fără a face calculul exact al înălțimilor).

Punctul	x	y
P_1	1	2
P_2	2	2
P_3	3	6
P_4	6	4
P_5	6	6
P_6	12	12

- b. Este oare acest algoritm diviziv echivalent⁴³⁵ cu unul dintre algoritmii de clusterizare aglomerativă studiați? Justificați atât pe datele acestea cât și în cazul general. Atenție la situațiile în care se poate alege între mai mulți candidați.

Răspuns:



- a. Reprezentarea celor șase puncte în plan este dată în figura alăturată.

Vom aplica pașii algoritmului descris în enunț:

Pasul 1: La calcularea distanțelor dintre perechile de puncte din setul de instanțe date, se poate completa un tabel, așa cum apare mai jos. Deoarece aceste distanțe se pot calcula folosind teorema lui Pitagora, iar ulterior ele vor fi folosite doar pentru a face diverse comparații între ele, este suficient să le exprimăm cu ajutorul radicalului. Notăm faptul că acest tabel al distanțelor poate fi considerat *matricea de adiacență* a grafului complet pe care vom lucra în continuare.

	(1, 2) P_1	(2, 2) P_2	(3, 6) P_3	(6, 4) P_4	(6, 6) P_5	(12, 12) P_6
(1, 2) P_1	0	$\sqrt{1}$	$\sqrt{20}$	$\sqrt{29}$	$\sqrt{41}$	$\sqrt{221}$
(2, 2) P_2	$\sqrt{1}$	0	$\sqrt{17}$	$\sqrt{20}$	$\sqrt{32}$	$\sqrt{200}$
(3, 6) P_3	$\sqrt{20}$	$\sqrt{17}$	0	$\sqrt{13}$	$\sqrt{9}$	$\sqrt{117}$
(6, 4) P_4	$\sqrt{29}$	$\sqrt{20}$	$\sqrt{13}$	0	$\sqrt{4}$	$\sqrt{100}$
(6, 6) P_5	$\sqrt{41}$	$\sqrt{32}$	$\sqrt{9}$	$\sqrt{4}$	0	$\sqrt{72}$
(12, 12) P_6	$\sqrt{221}$	$\sqrt{200}$	$\sqrt{117}$	$\sqrt{100}$	$\sqrt{72}$	0

⁴³⁵Adică: produc exact același rezultat și eventual exact în aceeași ordine (vedeți pasul 2 și respectiv pașii 3-4).

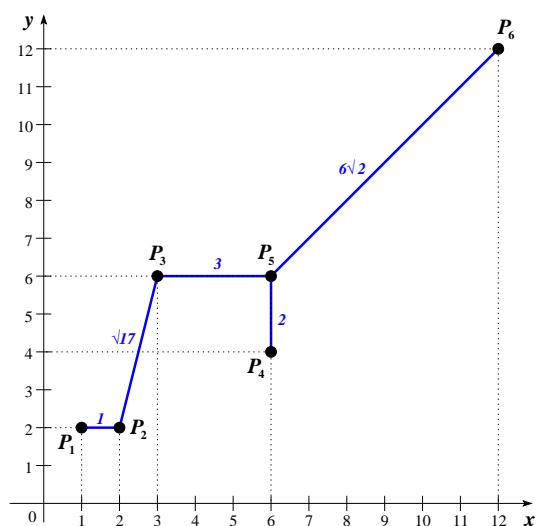
Pasul 2: Vom determina un arbore de cost minim folosind *algoritmul lui Kruskal*. Pentru un graf conex, acest algoritm poate fi descris pe scurt astfel:

Se pornește de la graful parțial de cost minim [care are toate nodurile grafului conex dat, dar] care nu are nicio muchie. Apoi, în mod iterativ se alege căte o muchie de cost minim din graful initial, care n-a fost încă folosită și care unește două componente conexe ale grafului parțial de la iteratărea anterioară. Algoritmul se încheie după $n - 1$ iterării, unde n este numărul de noduri din graf), adică atunci când graful parțial construit este conex. Se poate demonstra că acesta este chiar un arbore de cost minim.

Aplicat pe setul de date din enunț, algoritmul va alege în ordine următoarele muchii:

- (P_1, P_2) cu costul 1,
- (P_4, P_5) cu costul 2,
- (P_3, P_5) cu costul 3,
- (P_2, P_3) cu costul $\sqrt{17}$
- (P_5, P_6) cu costul $6\sqrt{2}$.

Arboarele de cost minim rezultat este reprezentat în figura alăturată.



Pasul 3: Eliminând muchia de cost maxim, adică muchia (P_5, P_6) , care are costul $\sqrt{72} = 6\sqrt{2}$, rezultă clusterele: $C_1 = \{P_1, P_2, P_3, P_4, P_5\}$ și $C_2 = \{P_6\}$.

La iteratărea următoare se elimină muchia (P_2, P_3) cu costul $\sqrt{17}$, rezultând clusterele: $C_3 = \{P_1, P_2\}$ și $C_4 = \{P_3, P_4, P_5\}$.

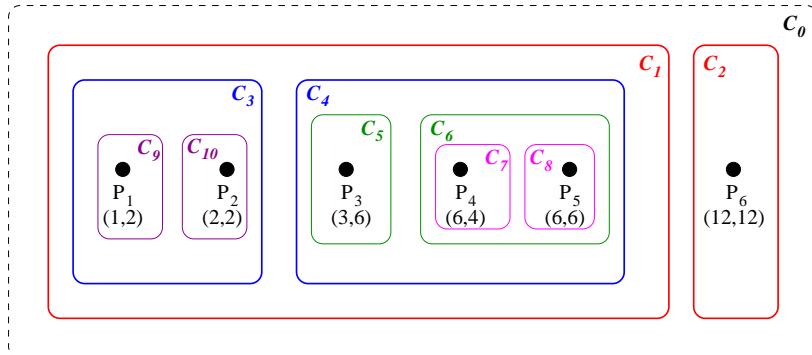
Apoi se elimină muchia (P_3, P_5) cu costul 3, rezultând clusterele: $C_5 = \{P_3\}$ și $C_6 = \{P_4, P_5\}$.

Au mai rămas două muchii, care se vor elimina pe rând, și anume: mai întâi muchia (P_4, P_5) , producând clusterele $C_7 = \{P_4\}$ și $C_8 = \{P_5\}$, și în final muchia (P_1, P_2) , rezultând clusterele $C_9 = \{P_1\}$ și $C_{10} = \{P_2\}$.

Observație: Eliminarea muchiilor din arbore se realizează exact în ordine inversă față de ordinea în care au fost alese inițial de către algoritmul lui Kruskal.

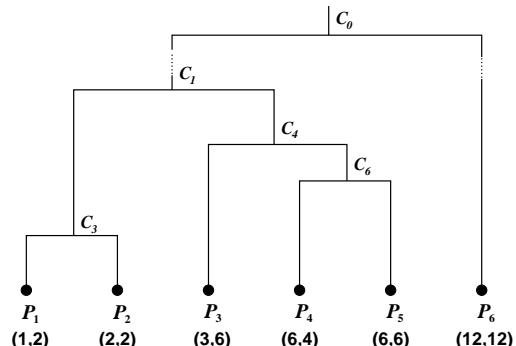
Putem reprezenta acum într-o formă aplatizată⁴³⁶ efectul aplicării algoritmului de clusterizare ierarhică top-down pe setul de instanțe date:

⁴³⁶În terminologia de limbă engleză există termenul “flat clustering”, care se folosește în special / general pentru clusterizare neierarhică.



b. Dacă se folosește algoritmul lui Kruskal pentru determinarea arborelui de cost minim, atunci algoritmul de clusterizare top-down din enunț este echivalent (în privința rezultatelor obținute) cu algoritmul de clusterizare aglomerativă care folosește similaritate de tip “single-linkage”. Chiar mai mult, alegerea muchiilor arborelui de cost minim corespunde (în ordine inversă, pas cu pas) calculului distanțelor de tip “single-linkage” dintre clustere.

Dendrograma obținută — fără a include calculul înălțimilor — de către algoritmul de clusterizare aglomerativă folosind similaritate de tip “single-linkage” pe datele din enunț este cea din figura alăturată.



Observația 1: Este posibil ca atunci când se folosește *algoritmul lui Prim* pentru determinarea arborelui de cost minim să se obțină rezultate diferite față de cazul când se folosește algoritmul lui Kruskal. Pentru un graf conex, algoritmul lui Prim poate fi descris pe scurt astfel:

Inițial, arborele este alcătuit dintr-un singur nod, ales în mod aleatoriu dintre nodurile grafului. La fiecare iterație se adaugă la acest arbore muchia de cost minim (și nodul corespunzător) care unește unul dintre nodurile nefolosite cu arborele deja existent. Algoritmul se încheie după $n - 1$ iterări, unde n este numărul de noduri din graf, adică atunci când graful parțial construit conectează toate nodurile grafului inițial. Acest graf parțial este chiar un arbore de cost minim.

Observația 2: Dacă graful considerat admite un singur arbore de cost minim — se poate verifica imediat că așa se întâmplă și în cazul de față —, atunci rezultatul aplicării algoritmilor lui Kruskal și respectiv al lui Prim este, evident, același. În caz contrar, arborii obținuți de fiecare dintre cei doi algoritmi depind de alegerile care pot fi făcute atunci când la un pas oarecare există mai multe muchii de cost minim. În plus, arborele obținut de către algoritmul lui Prim depinde de alegerea primului nod. Chiar dacă prin aplicarea celor doi algoritmi se obține același rezultat (adică același arbore de cost minim), în

general ordinea de alegere a muchiilor diferă. Algoritmul lui Prim extinde pas cu pas un arbore care în final devine arbore(le) de cost minim, în vreme ce algoritmul lui Kruskal construiește o *pădure de arbori* pe care-i unește până la final într-un arbore de cost minim. Așadar, în cazul în care la *Pasul 2* al algoritmului de clusterizare top-down prezentat în enunț se folosește algoritmul lui Prim, în general nu se poate stabili o echivalență directă între modul de lucru al algoritmului de clusterizare top-down pe de o parte (*Pasul 2*) și modul de lucru al algoritmului de clusterizare aglomerativă cu similaritate de tip “single-linkage” pe de altă parte,⁴³⁷ chiar dacă rezultatul final (adică, dendrograma) este exact același ca în cazul folosirii algoritmului lui Kruskal.

Indiferent dacă se folosește algoritmul lui Kruskal sau algoritmul lui Prim, la pașii 3-4 din algoritmul de clusterizare top-down se procedează exact în ordine inversă față de construirea dendrogramei prin clusterizare bottom-up folosind similaritate single-linkage (făcând alegerile corespunzătoare, în cazul situațiilor care comportă posibilități de alegere multiple).

7.1.2 Algoritmul *K-means*

7. (Algoritmul *K-means*:⁴³⁸ un exemplu simplu de aplicare pe date din \mathbb{R}^2)

prelucrare de Liviu Ciortuz, după Univ. of Utah, 2008 spring, HW3A, pr. 2

Considerăm în planul euclidian bidimensional un set de date format din cinci puncte: $A(-1, 0)$, $B(1, 0)$, $C(0, 1)$, $D(3, 0)$ și $E(3, 1)$.

a. Rulați manual două iterații ale algoritmului 2-means pe acest set de date,⁴³⁹ pornind de la următoarele poziții inițiale ale centroizilor: $(-1, 0)$ și $(3, 1)$. Veți folosi distanța euclidiană. La fiecare iterație indicați cum sunt asignate punctele la clustere și care sunt pozițiile actualizate ale centroizilor. A convers algoritmul după ce au fost efectuate cele două iterații?

b. Definim

$$J(\mu^{(t)}) = \sum_{x_i \in \{A, B, C, D, E\}} \|x_i - \mu^{(t)}(x_i)\|^2$$

unde $\mu^{(t)}$ desemnează ansamblul centroizilor la momentul / iterația t , iar $\mu^{(t)}(x_i)$ este centroidul clusterului la care este asignată instanța x_i la iterația t . Simbolul $\| \cdot \|$ reprezintă norma euclidiană. Pentru un vector oarecare $x \stackrel{\text{not.}}{=} (x_1, \dots, x_d) \in \mathbb{R}^d$, avem $\|x\|^2 \stackrel{\text{def.}}{=} x \cdot x \stackrel{\text{def.}}{=} x_1^2 + \dots + x_d^2 \in \mathbb{R}^+$. Arătați pur și simplu prin calcul numeric că pentru $t = 1$ avem

$$J(\mu^{(t)}) \leq J(\mu^{(t-1)}).$$

(Am desemnat cu $t = 0$ iterația inițială.)

⁴³⁷Și cu atât mai puțin pentru similaritate “complete-linkage” sau “average-linkage”.

⁴³⁸La problema aceasta, precum și la fiecare dintre problemele următoare, dacă nu se specifică altfel se va folosi forma algoritmului *K-means* din carteaua *Foundations of Statistical Natural Language Processing* (capitolul 14) de C. Manning and H. Schütze, editura MIT Press, 2002.

⁴³⁹Pentru pseudo-codul algoritmului *K-means*, puteți vedea enunțul problemei 12.

Răspuns:

a. Conform enunțului, centroizii inițiali sunt chiar două dintre punctele considerate: $\mu_1 \equiv A(-1,0)$ și $\mu_2 \equiv E(3,1)$. Atribuirea celorlalte puncte la clustere se poate face în manieră *analitică*, calculând distanțele dintre puncte și centroizi:

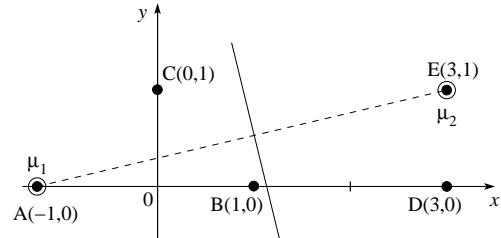
$$d(B, \mu_1) = 2 \text{ și } d(B, \mu_2) = \sqrt{5} \Rightarrow B \text{ se atribuie clusterului cu centroidul } \mu_1$$

$$d(C, \mu_1) = \sqrt{2} \text{ și } d(C, \mu_2) = 3 \Rightarrow C \text{ se atribuie clusterului cu centroidul } \mu_1$$

$$d(D, \mu_1) = 4 \text{ și } d(D, \mu_2) = 1 \Rightarrow D \text{ se atribuie clusterului cu centroidul } \mu_2$$

Alternativ, adică *geometric*, se reprezintă punctele date și centroizii inițiali în planul euclidian, după care se trasează mediatoarea segmentului $\mu_1\mu_2$. În figura de mai jos am reprezentat instanțele cu un cerculeț umplut (●), iar centroizii cu un cerculeț simplu. Cele două clustere sunt separate de către mediatoarea segmentului care unește cele două centroizi.

Se observă că punctele B și C sunt situate de aceeași parte a mediatoarei ca și centroidul μ_1 , deci rezultă că B și C vor apartine clusterului cu centroidul μ_1 . Similar, punctul D va apartine clusterului cu centroidul μ_2 .

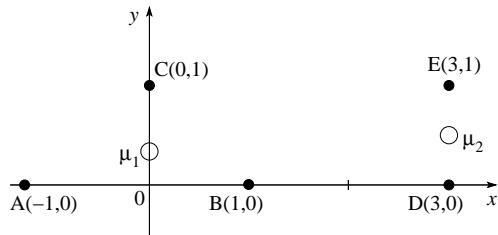


Având așadar componentă inițială a clusterelor $\{A(-1,0), B(1,0), C(0,1)\}$ și, respectiv, $\{D(3,0), E(3,1)\}$, la următoarea iterație a algoritmului se calculează noile poziții ale centroizilor:

$$\left. \begin{array}{l} x_{\mu_1} = \frac{x_A + x_B + x_C}{3} = 0 \\ y_{\mu_1} = \frac{y_A + y_B + y_C}{3} = \frac{1}{3} \end{array} \right\} \Rightarrow \mu_1(0, 1/3)$$

$$\left. \begin{array}{l} x_{\mu_2} = \frac{x_D + x_E}{2} = 3 \\ y_{\mu_2} = \frac{y_D + y_E}{2} = \frac{1}{2} \end{array} \right\} \Rightarrow \mu_2(3, 1/2)$$

Acum, reprezentarea grafică este cea din figura alăturată.



Apoi se re-atribuie punctele la centroizi, operație în urma căreia clusterele se pot modifica:

$$d(A, \mu_1) = \frac{\sqrt{10}}{3} \text{ și } d(A, \mu_2) = \frac{\sqrt{65}}{2} \Rightarrow A \text{ se atribuie la centroidul } \mu_1$$

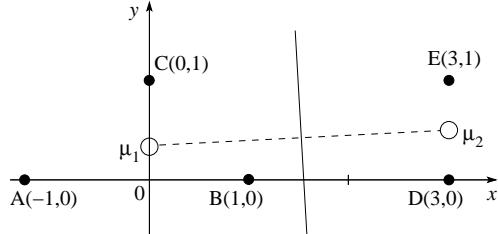
$$d(B, \mu_1) = \frac{\sqrt{10}}{3} \text{ și } d(B, \mu_2) = \frac{\sqrt{17}}{2} \Rightarrow B \text{ se atribuie la centroidul } \mu_1$$

$$d(C, \mu_1) = \frac{2}{3} \text{ și } d(C, \mu_2) = \frac{\sqrt{37}}{2} \Rightarrow C \text{ se atribuie la centroidul } \mu_1$$

$$d(D, \mu_1) = \frac{\sqrt{82}}{3} \text{ și } d(D, \mu_2) = \frac{1}{2} \Rightarrow D \text{ se asignează la centroidul } \mu_2$$

$$d(E, \mu_1) = \frac{\sqrt{85}}{3} \text{ și } d(E, \mu_2) = \frac{1}{2} \Rightarrow E \text{ se asignează la centroidul } \mu_2$$

Se observă deci analitic (dar și grafic, vedeti figura alăturată) că după această (a doua) iterare avem aceeași componentă a clusterelor (ca și la finalul precedentei iterării), și anume: $\{A(-1,0), B(1,0), C(0,1)\}$ și $\{D(3,0), E(3,1)\}$. Prin urmare, centroizii rămân $\mu_1(0, 1/3)$ și $\mu_2(3, 1/2)$.



Putem concluziona că după aceste două iterării algoritmul K -means a converz.

Observație importantă:

Atât în această problemă cât și în cele următoare, ne-am străduit să elaborăm soluția algoritmului K -means atât bazat pe calculul distanțelor (am numit-o deci soluția *analitică*), cât și bazat pe reprezentarea datelor în planul euclidian (am numit-o soluția *geometrică*). Aceasta din urmă are avantajul că oferă un suport vizual care ajută studentul să înțeleagă lucrurile într-o manieră mai intuitivă.

La implementare, cele două tipuri de soluții apelează la procedee de calcul care diferă parțial.⁴⁴⁰

În condiții de seminar sau de examen, cele două procedee pot fi combinate — iar acolo unde detaliile sunt evidente, calculele pot fi lăsate de o parte dacă nu se specifică altfel —, datorită restricțiilor de timp impuse.

b. Vom reprezenta punctele / instanțele de clusterizat sub formă de vectori-coloană, cu două componente (una pentru abscisa x și cealaltă pentru ordonata y).

$$\begin{aligned} J(\mu^{(0)}) &= \left\{ \left[\begin{pmatrix} -1 \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right]^2 \right\} \\ &\quad + \left\{ \left[\begin{pmatrix} 3 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix} \right]^2 \right\} \\ &= \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}^2 + \begin{pmatrix} 2 \\ 0 \end{pmatrix}^2 + \begin{pmatrix} 1 \\ 1 \end{pmatrix}^2 \right\} + \left\{ \begin{pmatrix} 0 \\ -1 \end{pmatrix}^2 + \begin{pmatrix} 0 \\ 0 \end{pmatrix}^2 \right\} \\ &= 0 + 4 + 2 + 1 + 0 = 7. \end{aligned}$$

$$\begin{aligned} J(\mu^{(1)}) &= \left\{ \left[\begin{pmatrix} -1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1/3 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1/3 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 1/3 \end{pmatrix} \right]^2 \right\} \\ &\quad + \left\{ \left[\begin{pmatrix} 3 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1/2 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 1/2 \end{pmatrix} \right]^2 \right\} \end{aligned}$$

⁴⁴⁰Este mai simplu de realizat implementarea bazată pe metoda analitică. Însă pentru vizualizarea rezultatelor intermediere sau finale — aşa cum se procedează la rezolvarea majorității problemelor din această secțiune a prezentului capitol — se utilizează și componente ale variantei geometrice.

$$\begin{aligned}
 &= \left\{ \left(\frac{-1}{-1/3} \right)^2 + \left(\frac{1}{-1/3} \right)^2 + \left(\frac{0}{2/3} \right)^2 \right\} + \left\{ \left(\frac{0}{-1/2} \right)^2 + \left(\frac{0}{1/2} \right)^2 \right\} \\
 &= \frac{10}{9} + \frac{10}{9} + \frac{4}{9} + \frac{1}{4} + \frac{1}{4} = \frac{24}{9} + \frac{1}{2} = \frac{8}{3} + \frac{1}{2} = \frac{19}{6}.
 \end{aligned}$$

Prin urmare, se verifică inegalitatea $J(\mu^{(1)}) \leq J(\mu^{(0)})$.

Observație:

Proprietatea de descreștere [nu neapărat strictă a] valorilor funcției J în cursul execuției algoritmului K -means (la orice iterație și pe orice set de date de clusterizat) va face obiectul problemei 12.

8.

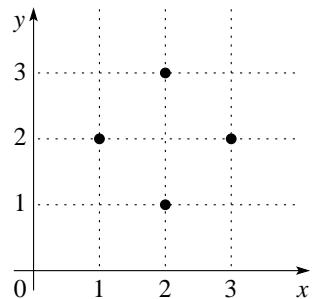
(Algoritmul K -means: exemplificare pentru alegerea centroizilor inițiali astfel încât clusterele obținute să satisfacă anumite restricții)

CMU, 2009 spring, Ziv Bar-Joseph, HW5, pr. 2.2

Considerăm 4 puncte dispuse în spațiu aşa cum se arată în figura alăturată.

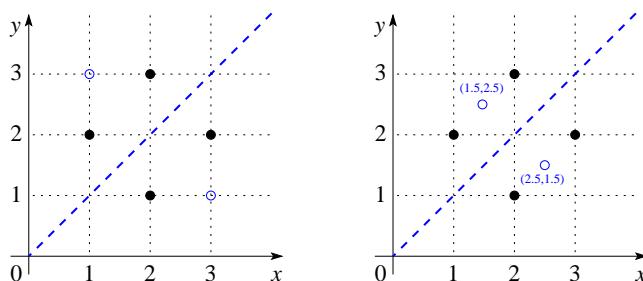
- Aplicând algoritmul K -means pentru $K = 2$, alegeti centroizii inițiali în aşa fel încât să obțineți o clusterizare în care fiecare cluster să conțină câte 2 elemente.
- Aceeași cerință, de data aceasta pentru a obține 2 clustere dintre care unul să conțină un element iar celălalt 3 elemente.

Observație. Se va folosi distanța euclidiană.

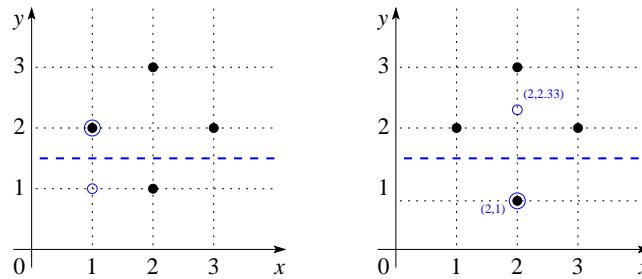


Răspuns:

- Pentru a se obține două clustere de câte două elemente, se pot alege centroizii inițiali $\mu_1 = (1, 3)$ și $\mu_2 = (3, 1)$. După prima iterație (care se dovedește a fi și ultima), se vor obține centroizii $\mu'_1 = (1.5, 2.5)$ și $\mu'_2 = (2.5, 1.5)$, ca în figura de mai jos. Instantele au fost reprezentate prin cerculete umplute (●), iar simbolurile ○ marchează pozițiile centroizilor. Liniile punctate constituie separatorii celor două clustere.



- Pentru a se obține două clustere cu 1 și respectiv 3 elemente, se pot alege centroizii inițiali $\mu_1 = (1, 1)$ și $\mu_2 = (1, 2)$. După o primă iterație, se vor obține centroizii $\mu'_1 = (2, 1)$ și $\mu'_2 = (2, 2)$, ca în figura de mai jos, după care algoritmul converge.



9.

(Algoritmul K -means: aplicare pe date din \mathbb{R}^2)
CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 1.8

Se dă un set de puncte din planul euclidian, conform tabelului alăturat.

Vă cerem să rulați manual algoritmul K -means pentru a identifica două clustere în acest set de date. Centroizii inițiali sunt chiar punctele P_1 și P_{10} . Se folosesc distanțe euclidiene.

Care este compoziția clusterelor după prima iterare? Dar la convergența algoritmului?

Punctul	x	y
P_1	1.90	0.97
P_2	1.76	0.84
P_3	2.32	1.63
P_4	2.31	2.09
P_5	1.14	2.11
P_6	5.02	3.02
P_7	5.74	3.84
P_8	2.25	3.47
P_9	4.71	3.60
P_{10}	3.17	4.96

Răspuns:

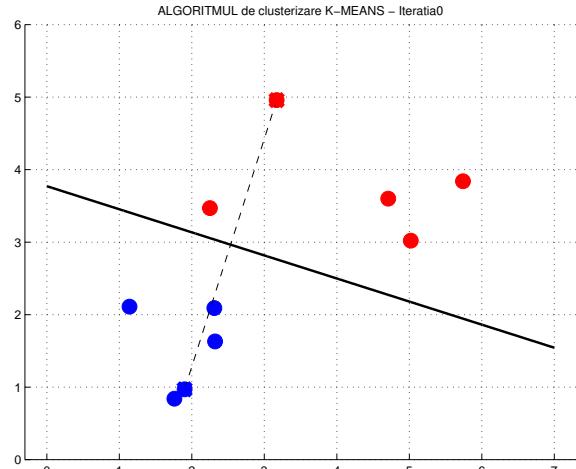
La prima iterare a algoritmului K -means, fiecare punct P_i va fi asignat unui dintre cele două clustere (C_1 și C_2), în funcție de distanțele dintre punctul respectiv și centroizii inițiali.

Construim un tabel cu aceste distanțe și asignările deduse din relația de ordine dintre ele. Vom nota cu $d(P_i, P_j)$ distanța de la punctul P_i la punctul P_j .

P_i	$d(P_i, P_1)$	$d(P_i, P_{10})$	$d(P_i, P_1) : d(P_i, P_{10})$	Clusterul
P_1	0	4.18	<	C_1
P_2	0.19	4.35	<	C_1
P_3	0.78	3.43	<	C_1
P_4	1.19	2.99	<	C_1
P_5	1.37	3.49	<	C_1
P_6	3.73	2.68	>	C_2
P_7	4.97	2.80	>	C_2
P_8	2.52	1.75	>	C_2
P_9	3.84	2.05	>	C_2
P_{10}	4.18	0	>	C_2

Așadar, la finalul primei iterări obținem clusterele $C_1 = \{P_1, P_2, P_3, P_4, P_5\}$ și $C_2 = \{P_6, P_7, P_8, P_9, P_{10}\}$.

Același rezultat se poate obține și procedând *geometric*. Pentru aceasta, în planul euclidian bidimensional trăsăm medianoarea segmentului P_1P_{10} și apoi analizăm poziția fiecărui punct dat în raport cu această medianoare. Rezultatul este arătat în figura alăturată.



Pozițiile centroizilor (marcate pe grafic prin caracterul \times) se recalculează astfel:

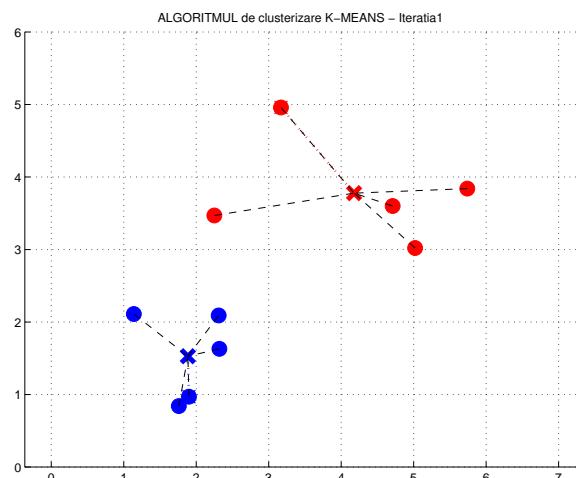
$$\begin{aligned}x_{\mu_1} &= \frac{x_{P_1} + x_{P_2} + x_{P_3} + x_{P_4} + x_{P_5}}{5} = 1.886 \\y_{\mu_1} &= \frac{y_{P_1} + y_{P_2} + y_{P_3} + y_{P_4} + y_{P_5}}{5} = 1.528\end{aligned}$$

Prin urmare, $\mu_1 = (1.886, 1.528)$. Analog, obținem $\mu_2 = (4.178, 3.778)$.

Dacă am decis să procedăm *analitic*, în continuare vom calcula distanțele de la fiecare punct P_i la centroizii μ_1 și respectiv μ_2 .

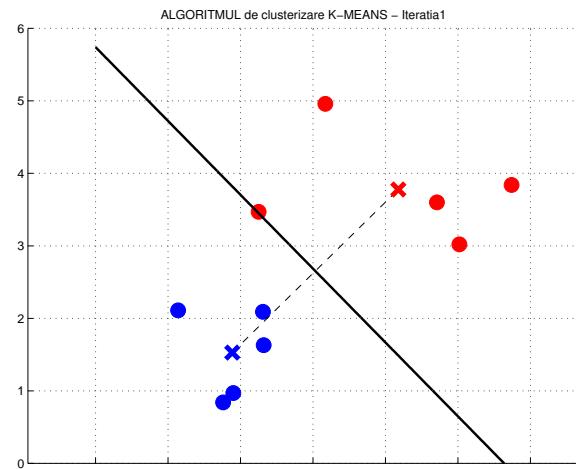
În figura alăturată am marcat noile poziții ale centroizilor (μ_1 și μ_2), precum și asignarea fiecărui dintre punctele P_i la cel mai apropiat centroid, în funcție de distanțe.

Așadar, punctele P_1, \dots, P_5 formează — și de data aceasta! — primul cluster (cel cu centroidul μ_1), iar restul punctelor, P_6, \dots, P_{10} , constituie cel de-al doilea cluster (cel cu centroidul μ_2).



Alternativ, adică în ipoteza că am optat pentru o *rezolvare geometrică*, am observat că punctul $P_8(2.25, 3.47)$ este „la limită” față de medianoarea segmentului care unește centroizii μ_1 și μ_2 , așa cum se observă din figura de mai jos.

Pentru a decide cărui cluster îi va apartine acest punct, fie calculăm $d^2(P_8, \mu_1)$ și $d^2(P_8, \mu_2)$ — și vom obține valorile 3.904 și respectiv 3.812 —, fie comparăm semnul expresiei $f(2.52, 1.75)$ cu cel al lui $f(x_{\mu_1}, y_{\mu_1})$ (sau $f(x_{\mu_2}, y_{\mu_2})$), unde f este funcția din membrul stâng al ecuației $f(x, y) = 0$ care definește mediatoarea segmentului determinat de punctele μ_1 și μ_2 .



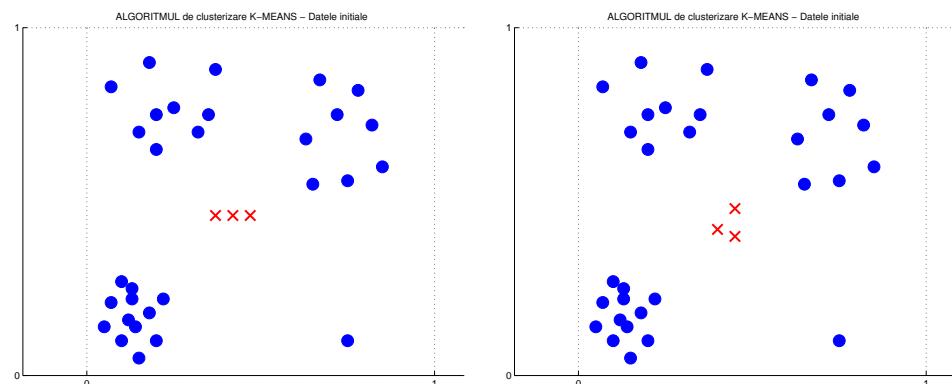
În ambele variante se ajunge la aceeași concluzie: punctul P_8 va apartine clusterului cu centroidul μ_2 . (Valoarea expresiei $f(2.52, 1.75)$ are același semn cu $f(x_{\mu_2}, y_{\mu_2})$.) Clusterele finale sunt deci: $\{P_1, P_2, P_3, P_4, P_5\}$ și $\{P_6, P_7, P_8, P_9, P_{10}\}$.

10.

(Algoritmul K -means: aplicare în \mathbb{R}^2 ; compararea rezultatelor obținute în cazul a două inițializări diferite ale centroizilor)

■ CMU, 2006 spring, Carlos Guestrin, HW5, pr. 1

Se consideră setul de date reprezentat în cele două figuri de mai jos. Sunt date două inițializări posibile ale centroizilor, câte una în fiecare din cele două figuri. Instanțele sunt reprezentate de cerculețe umplute (\bullet), iar simbolurile \times corespund pozițiilor inițiale ale centroizilor clusterelor.⁴⁴¹



⁴⁴¹Coordonatele exacte ale instanțelor, precum și cele ale centroizilor vă sunt puse la dispoziție în următoarele fișiere, depuse pe site-ul acestei cărți:
<http://profsci.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2006s.HW5.pr1.cl.dat>,
<http://profsci.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2006s.HW5.pr1.a.init.dat>,
<http://profsci.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2006s.HW5.pr1.b.init.dat>.

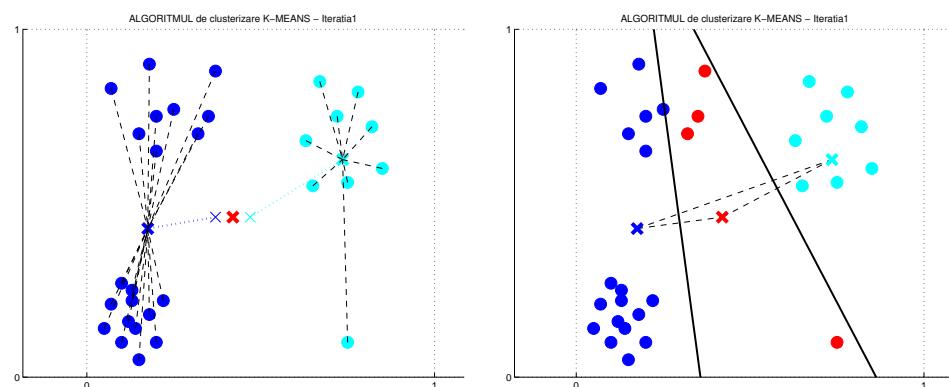
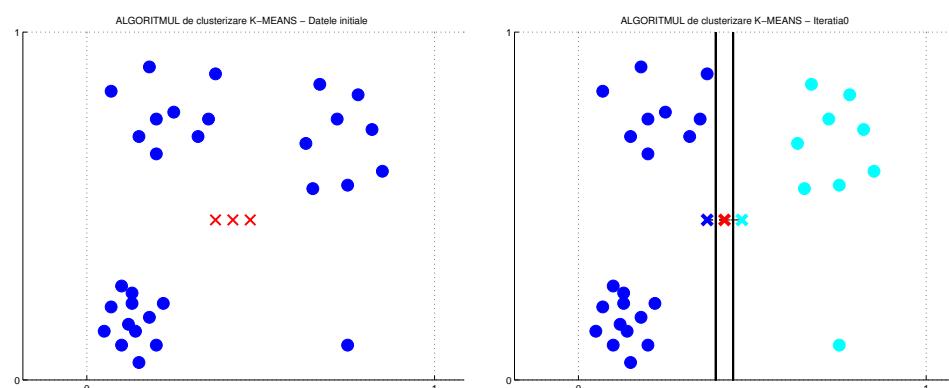
a-b. Execuți / aplicați algoritmul K -means separat pentru fiecare dintre cele două inițializări ale centroizilor, arătând la fiecare iterație cum evoluează centroizii și separatorii, până la convergență.

Observație: La execuția algoritmului se va considera că atunci când un centroid nu are puncte asignate lui, atunci el va rămâne pe loc la iterată respectivă.

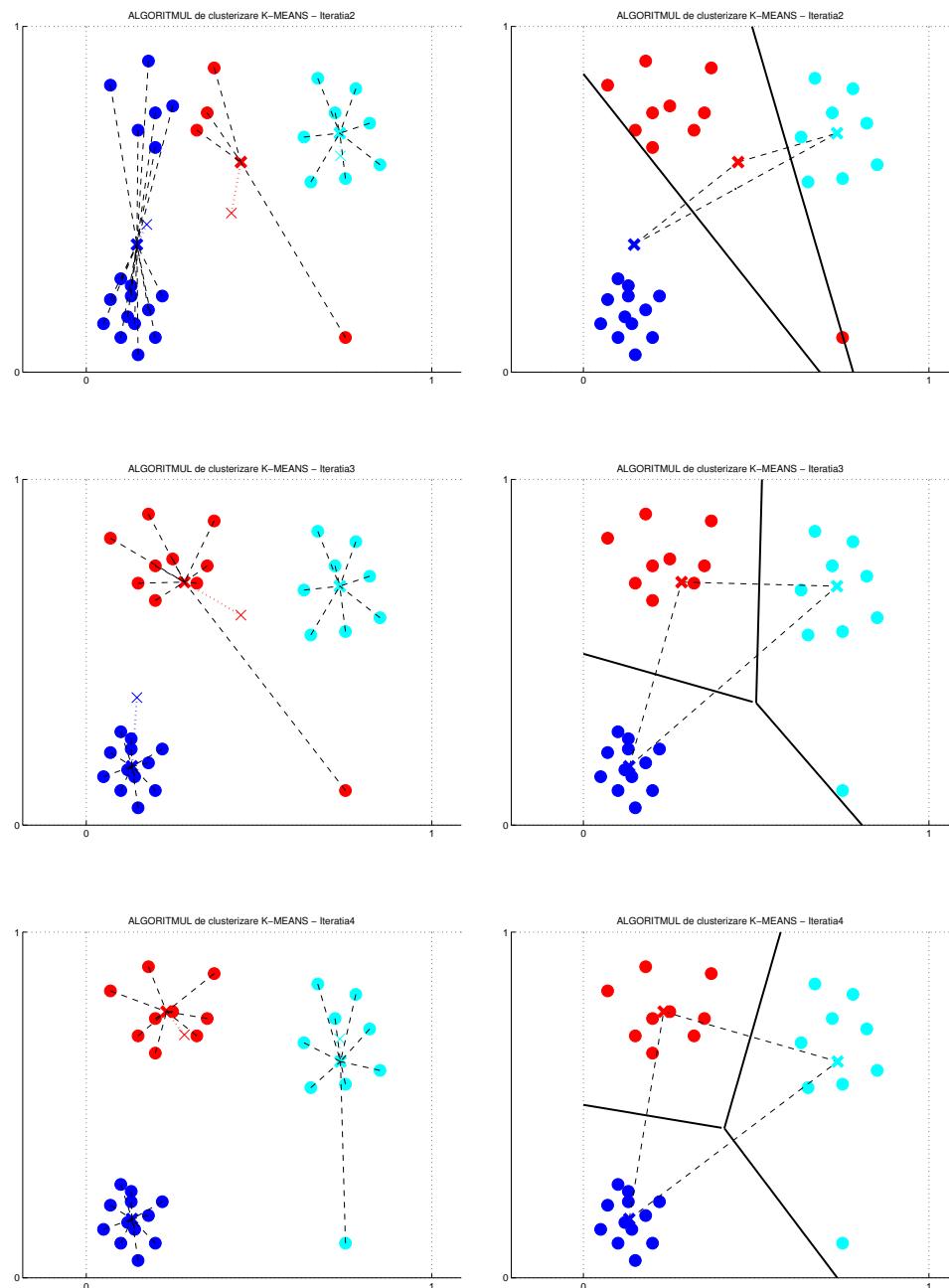
c. Având în vedere rezultatele obținute în cele două cazuri, ce puteți spune despre comportamentul algoritmului K -means?

Răspuns:

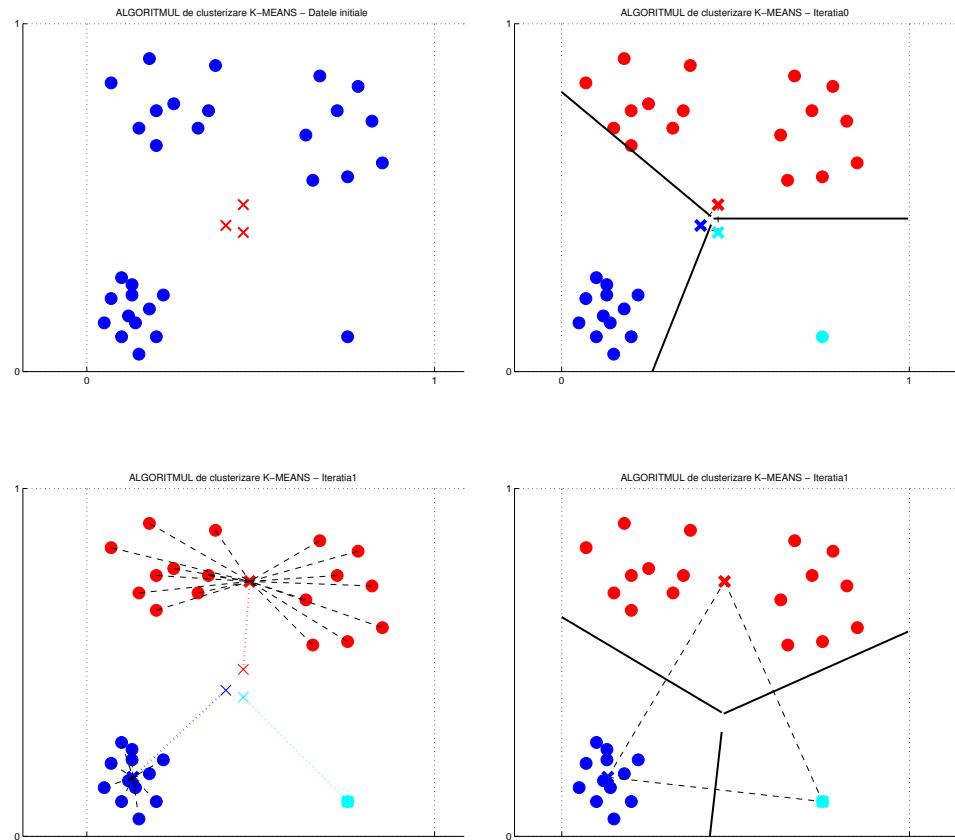
a. Pentru prima inițializare a centroizilor, algoritmul K -means parurge următoarele iterări:⁴⁴²



⁴⁴²La fiecare iterăție, în afară de cea inițială (care a fost numerotată cu 0), am desenat în graficul din partea stângă noile poziții ale centroizilor — vedeti simbolul \times mai îngroșat, spre deosebire de vechea lui poziție, care a fost redată neîngroșat pentru a sugera mișcarea —, iar în graficul din partea dreaptă am indicat noua componentă a clusterelor, în funcție de pozițiile actuale ale separatorilor (mediatoarele segmentelor care unesc centroizii).



b. Pentru a două inițializare a centroizilor, algoritmul *K-means* parcurge următoarele iterații:



c. Este evident din acest exercițiu că rezultatul algoritmului K -means depinde de poziționarea inițială a centroizilor. În cazul initializării de la punctul a au fost necesare 4 iterări până la se ajunge la convergență, pe când la punctul b algoritmul a convergat după doar o iterare.

Mai este încă ceva important *de remarcat*: faptul că la prima variantă de inițializare, punctul din dreapta jos, care este un *outlier* (rom., excepție, caz particular, aberație) este până la urmă asociat clusterului format de punctele din partea dreapta (sus), în vreme ce la cea de-a doua variantă de inițializare el constituie un cluster aparte / “singleton”, obligând împreună cu clusterul din stânga-sus și dreapta-sus să formeze un singur cluster. Detectia outlier-elor este un capitol important din învățarea automată. Ignorarea lor poate conduce la rezultate eronate sau chiar aberante ale modelărilor.

11.

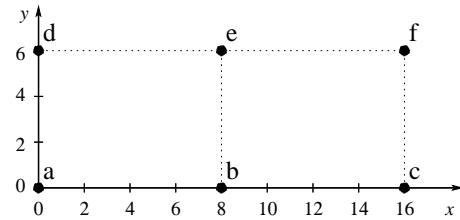
(Algoritmul K -means: aplicare în \mathbb{R}^2 ; realizarea corespondenței dintre K -configurația [obținută la o iterație] curentă a algoritmului și una sau mai multe K -configurații de start)

• ○ *prelucrare făcută de Liviu Ciortuz, după CMU, 2003 fall, T. Mitchell, A. Moore, final exam, pr. 9*

Considerăm mulțimea S formată de următoarele 6 puncte din plan: $a = (0, 0)$, $b = (8, 0)$, $c = (16, 0)$, $d = (0, 6)$, $e = (8, 6)$ și $f = (16, 6)$. Reprezentarea datelor în planul euclidian este dată în figura alăturată. Se rulează algoritmul K -means, unde $K = 3$, și se folosește distanța euclidiană.

Introducem următoarele două *noțiuni*:

- *K -partiție*: desemnează o partiționare a mulțimii S în K submulțimi nevide. Spre exemplu, $\{a, b, e\}, \{c, d\}, \{f\}$ este o 3-partiție;
- *K -configurație de start*: desemnează o submulțime de K puncte din mulțimea S , care vor juca rolul de centroizi inițiali. Iată un exemplu de 3-configurație de start, care generează una dintre 3-partițiile considerate mai jos (și anume, $\{a, b\}, \{d, e\}, \{c, f\}$): $\{a, d, c\}$.



În mod evident, orice K -partiție induce o mulțime de K centroizi. O K -partiție este numită *stabilă* dacă după execuția unei noi iterării a algoritmului K -means, partiția respectivă rămâne neschimbăță.

a. Câte 3-configurații de start există pentru mulțimea S dată mai sus?

b. Completați tabelul următor:

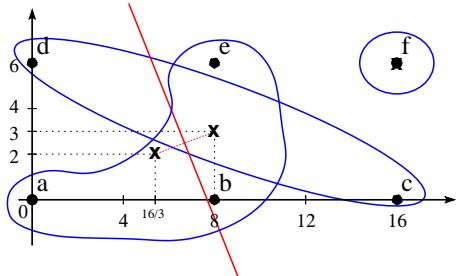
3-partiție	Stabilitate?	Un exemplu de 3-configurație de start care poate determina 3-partiția din prima coloană, după 0 sau mai multe iterări ale algoritmului K -means. (Dacă nu există o asemenea configurație, se va trece -.)	Numărul de 3-configurații de start care pot genera această 3-partiție
$\{a, b, e\}, \{c, d\}, \{f\}$	Nu	-	0
$\{a, b\}, \{d, e\}, \{c, f\}$	Da	$\{b, c, e\}$	4
$\{a, d\}, \{b, e\}, \{c, f\}$			
$\{a\}, \{d\}, \{b, c, e, f\}$			
$\{a, b\}, \{d\}, \{c, e, f\}$			
$\{a, b, d\}, \{c\}, \{e, f\}$			

Răspuns:

a. Pentru a forma o 3-configurație de start, se aleg oricare 3 puncte distincte din mulțimea S (care are 6 puncte), deci numărul acestor 3-configurații este $C_6^3 = \frac{6!}{3! \cdot 3!} = \frac{6 \cdot 5 \cdot 4}{1 \cdot 2 \cdot 3} = 20$.

b. Vom analiza pe rând fiecare dintre cele șase cazuri din tabel, inclusiv primele două cazuri pentru care avem deja anumite informații date în tabel, dar care merită a fi discutate. În figurile următoare, centroizii inițiali sunt reprezentați cu cerculețe, iar centroizii actuali cu x.

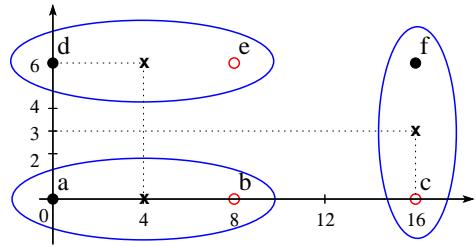
Se poate verifica ușor faptul că în *cazul 1*, 3-partiția dată nu este stabilă. Ea „evolu-ează“ în partiția $\{a, d\}$, $\{b, e\}$, $\{c, f\}$. Însă, ținând cont de faptul că pozițiile inițiale ale centroizilor sunt alese doar din mulțimea S , vom putea arăta mai jos că nu există nicio 3-configurație de start care să rezulte în 3-partiția dată.



Pentru algoritmul K -means, atunci când se folosește distanța euclidiană, are loc următoarea proprietate: la finalul oricărei iterării a algoritmului (deci și la terminarea algoritmului), orice două clustere sunt separabile liniar (și anume, cu ajutorul mediatoarei segmentului care unește centroizii acestor două clustere, dacă instanțele sunt din \mathbb{R}^2). Demonstrarea acestei proprietăți este imediată, ținând cont de modul în care sunt asignate instanțele la centroizi. (Vedeți pasul al doilea al corpului iterativ al algoritmului K -means, de exemplu în formularea din enunțul problemei 12.)

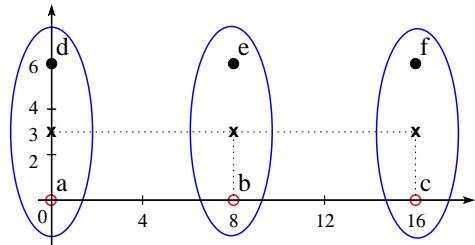
În condițiile de față (*cazul 1*), se observă imediat că mediatoarea segmentului care unește centroizii clusterelor $\{a, b, e\}$ și $\{c, d\}$ le separă pe fiecare în câte două submulțimi nevide, în loc să situeze un cluster de o parte a ei (adică, a mediatoarei) și celălalt cluster de cealaltă parte.

În *cazul 2*, se observă imediat că în afară de 3-configurația de start dată ($\{b, c, e\}$, deja stabilă), mai există 3 alte asemenea configurații de start care generează 3-partiția considerată ($\{a, b\}$, $\{d, e\}$, $\{c, f\}$): $\{b, e, f\}$, $\{a, d, c\}$ și $\{a, d, f\}$.



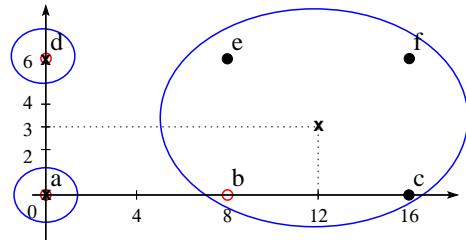
Observație: În acest caz,⁴⁴³ am presupus că este permisă libertatea de asociere a instanțelor care sunt egal distanțate față de doi centroizi. Aceeași presupozitie o vom folosi și în continuare, acolo unde va fi nevoie (*cazul 5* și *cazul 6*).

În *cazul 3*, se observă că 3-partiția dată ($\{a, d\}$, $\{b, e\}$, $\{c, f\}$) este stabilă și sunt ușor de enumerat cele 8 configurații de start care „termină“ în această 3-partiție a mulțimii S : $\{a, b, c\}$, $\{a, b, f\}$, $\{a, e, c\}$, $\{a, e, f\}$, $\{d, b, c\}$, $\{d, b, f\}$, $\{d, e, c\}$, $\{d, e, f\}$.



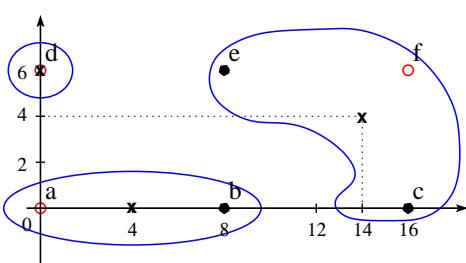
⁴⁴³Vedeți configurațiile de start $\{b, c, e\}$ și $\{b, e, f\}$ menționate mai sus.

În cazul 4, unde 3-partiția de start ($\{a\}, \{d\}, \{b, c, e, f\}$) este de asemenea stabilă, este imediat că instanțele a și d trebuie să facă parte în mod obligatoriu din configurația (eventual, configurațiile) de start. Într-adevăr, dacă măcar una dintre instanțele a și d ar lipsi din configurația de start, se observă că algoritmul K -means nu ar putea să ajungă în 3-partiția dată.

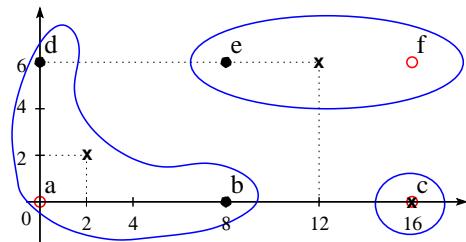


Pe de altă parte, c și f nu pot să apară în configurația de start (din exact același motiv). Rămân de analizat pozițiile b și e ; se verifică ușor că acestea pot completa (pe rând) 3-configurațiile de start, care sunt prin urmare $\{a, b, d\}$ și $\{a, d, e\}$.

În cazul 5, 3-partiția data ($\{a, b\}, \{d\}, \{c, e, f\}$) este stabilă. Se observă că pe de o parte instanța d trebuie să facă parte din orice configurație de start care ar putea să genereze 3-partiția dată, iar pe de altă parte instanțele b, c și e nu pot face parte din nicio astfel de configurație de start. Celelalte instanțe din S formează singura 3-configurație de start validă: $\{a, d, f\}$.



Cazul 6 este similar cazului 5 (datorită simetriei celor două 3-partiții date față de punctul $(8; 3)$), aşadar singura 3-configurație de start posibilă este $\{a, c, f\}$.



În final, centralizăm în tabel rezultatele obținute:

Caz	3-partiție	Stabilitate?	Un exemplu de 3-configurație de start care poate determina 3-partiția dată, după 0 sau mai multe iterații ale algoritmului K -means.	Numărul de 3-configurații de start care generază această 3-partiție
1	$\{a, b, e\}, \{c, d\}, \{f\}$	Nu	—	0
2	$\{a, b\}, \{d, e\}, \{c, f\}$	Da	$\{b, c, e\}$	4
3	$\{a, d\}, \{b, e\}, \{c, f\}$	Da	$\{a, b, c\}$	8
4	$\{a\}, \{d\}, \{b, c, e, f\}$	Da	$\{a, b, d\}$	2
5	$\{a, b\}, \{d\}, \{c, e, f\}$	Da	$\{a, d, f\}$	1
6	$\{a, b, d\}, \{c\}, \{e, f\}$	Da	$\{a, c, f\}$	1

Observație: Restul de 5 configurații de start — în total, în varianta completă (vedeți coloana a cincea) a tabelului de mai sus sunt 15 configurații de start, dintre care una, $\{a, d, f\}$, apare de două ori — conduce la alte trei 3-partiții decât cele menționate în coloana a doua a tabelului.

Observație importantă:

Este util de precizat acum — odată ce au fost „fixate“ noțiunile de K -partiție și K -configurație — că algoritmul K -means poate fi văzut ca un *algoritm de căutare*. K -means pornește de la o K -configurație (aleasă eventual în mod arbitrar) și o îmbunătățește în mod succesiv, prin intermediul K -partițiilor corespunzătoare, sau invers. *Spațiul de căutare* corespunzător algoritmului K -means este mulțimea tuturor K -partițiilor care se pot forma peste X , setul de instanțe de clusterizat.

12. $(K\text{-means}, \text{ca algoritm de optimizare a criteriului coeziunii intra-clustere} (\text{,,suma celor mai mici pătrate''}))$
 ■ prelucrare de Liviu Ciortuz, după CMU, 2009 spring, Ziv Bar-Joseph, HW5, pr. 2.1

Vă reamintim algoritmul K -means, datorat lui Lloyd,⁴⁴⁴ predat la curs:

Input: $x_1, \dots, x_n \in \mathbb{R}^d$ și $K \in \mathbb{N}^*$.

Output: o anumită K -partiție pentru $\{x_1, \dots, x_n\}$, adică o descompunere a acestei mulțimi într-o colecție de K mulțimi disjuncte (care nu sunt în mod neapărat nevide).

Procedură:

[Initializare / Iterația 0:] $t \leftarrow 0$;

se fixează în mod arbitrar μ_1^0, \dots, μ_K^0 , centroizii inițiali ai clusterelor, și se asignează fiecare instanță x_i la centroidul cel mai apropiat, formând astfel clusterele C_1^0, \dots, C_K^0 .

[Corpul iterativ:] Se execută iterația $+t$:

Pasul 1: se calculează noile poziții ale centroizilor:⁴⁴⁵

$$\mu_j^t = \frac{1}{|C_j^{t-1}|} \sum_{x_i \in C_j^{t-1}} x_i \text{ pentru } j = \overline{1, K};$$

Pasul 2:

se reasignează fiecare x_i la [clusterul cu] centroidul cel mai apropiat, adică se stabilește noua componentă a clusterelor la iterarea t : C_1^t, \dots, C_K^t ;

[Terminare:] până când o anumită condiție este îndeplinită

(de exemplu: până când pozițiile centroizilor — sau: componenta clusterelor — nu se mai modifică de la o iterare la alta).

- a. Demonstrați că, de la o iterare la alta, algoritmul K -means mărește *coeziunea de ansamblu* a clusterelor. Veți proceda astfel: considerând funcția

$$J(C^t, \mu^t) \stackrel{\text{def.}}{=} \sum_{i=1}^n \|x_i - \mu_{C^t(x_i)}^t\|^2 \stackrel{\text{def.}}{=} \sum_{i=1}^n (x_i - \mu_{C^t(x_i)}^t) \cdot (x_i - \mu_{C^t(x_i)}^t),$$

unde

$C^t = (C_1^t, C_2^t, \dots, C_K^t)$ este colecția de clustere (i.e., K -partiția) la momentul t ,

$\mu^t = (\mu_1^t, \mu_2^t, \dots, \mu_K^t)$ este colecția de centroizi ai clusterelor (K -configurația) la momentul t ,

$C^t(x_i)$ desemnează clusterul la care este asignat elementul x_i la iterarea t ,

operatorul \cdot desemnează produsul scalar al vectorilor din \mathbb{R}^d ,

⁴⁴⁴Lloyd, S. P. (1957). “Least square quantization in PCM”. Bell Telephone Laboratories Paper.

⁴⁴⁵Formula aceasta corespunde folosirii distanței euclidiene. Pentru alte măsuri de distanță, este posibil să fie necesare alte formule.

arătați că $J(C^t, \mu^t) \geq J(C^{t+1}, \mu^{t+1})$ pentru orice t .

Indicație: Inegalitatea de mai sus rezultă din două inegalități (care corespund pașilor 1 și 2 de la iterată t):

$$J(C^t, \mu^t) \stackrel{(1)}{\geq} J(C^t, \mu^{t+1}) \stackrel{(2)}{\geq} J(C^{t+1}, \mu^{t+1}).$$

La prima inegalitate (cea corespunzătoare pasului 1) se poate considera că parametrul C^t este fixat iar μ este variabil, în vreme ce la a doua inegalitate (cea corespunzătoare pasului 2) se consideră μ^t fixat și C variabil. Prima inegalitate se poate obține însumând o serie de inegalități, și anume câte una pentru fiecare cluster C_j^t : se demonstrează (de exemplu, cu ajutorul proprietăților derivatei) că $J(C_j^t, \mu) \geq J(C_j^t, \mu^{t+1})$ pentru $\forall \mu$, deci în particular și pentru μ^t . A doua inegalitate se demonstrează imediat.

b. Ce puteți spune despre oprirea algoritmului K -means? Termină oare acest algoritm într-un număr finit de pași, ori dimpotrivă — dat fiind faptul că există doar un număr finit de K -partiții ale mulțimii de instanțe $\{x_1, \dots, x_n\}$, și anume K^n — este posibil ca el să reviziteze de o infinitate de ori o K -configurație anterioară, $\mu = (\mu_1, \dots, \mu_K)$?

Răspuns:

a. Pentru conveniență, ne vom limita la cazul $d = 1$.⁴⁴⁶ Extinderea demonstrației la cazul $d > 1$ nu comportă dificultăți.⁴⁴⁷

Vom demonstra mai întâi inegalitatea (1): $J(C^t, \mu^t) \geq J(C^t, \mu^{t+1})$.

$$\text{După definiție, } J(C^t, \mu^t) = \sum_{i=1}^n (x_i - \mu_{C^t(x_i)})^2.$$

Conform notației din enunț, C_j^t reprezintă clusterul j de la iterată t , iar μ_j^t este centroidul corespunzător acestui cluster. Dacă notăm cu $x_{i_1}, x_{i_2}, \dots, x_{i_l}$ instanțele din componența clusterului C_j^t , unde $l \stackrel{\text{not.}}{=} |C_j^t|$, atunci (prinț-un ușor abuz de notație) vom putea scrie:

$$J(C_j^t, \mu_j^t) = \sum_{p=1}^l (x_{i_p} - \mu_j^t)^2,$$

iar $J(C^t, \mu^t)$ se va rescrie sub formă $J(C^t, \mu^t) = \sum_{j=1}^K J(C_j^t, \mu_j^t)$.

Dacă se consideră C_j^t fixat, iar μ_j^t variabil (vedeți pasul 1 al iterăției t), atunci putem minimiza funcția de gradul al doilea care „măsoară“ coeziunea din interiorul clusterului C_j^t :

$$f(\mu) \stackrel{\text{def.}}{=} J(C_j^t, \mu) = l\mu^2 - 2 \left(\sum_{p=1}^l x_{i_p} \right) \cdot \mu + \sum_{p=1}^l x_{i_p}^2$$

fie în mod direct (adică făcând apel la proprietățile funcției de gradul al doilea):

$$\arg \min_{\mu} J(C_j^t, \mu) = \frac{1}{l} \sum_{p=1}^l x_{i_p},$$

⁴⁴⁶Pentru o discuție interesantă asupra aplicării algoritmului K -means în cazul $d = 1$, vezi problema 43.

⁴⁴⁷Se dezvoltă în mod corespunzător expresia lui J sau se folosesc derivatele sale parțiale de ordinul întâi.

fie cu ajutorul derivatei vectoriale:⁴⁴⁸

$$\frac{\partial}{\partial \mu} f(\mu) = \sum_{p=1}^l (-2) (x_{i_p} - \mu) = -2 \left(\sum_{p=1}^l x_{i_p} - l\mu \right),$$

știind că punctul de minim al funcției $f(\mu) = J(C_j^t, \mu)$ se obține pentru rădăcina ecuației $\frac{\partial}{\partial \mu} f(\mu) = 0$, adică

$$\mu = \frac{1}{l} \sum_{p=1}^l x_{i_p} = \frac{1}{|C_j^t|} \sum_{p=1}^l x_{i_p} \stackrel{\text{def.}}{=} \mu_j^{t+1}.$$

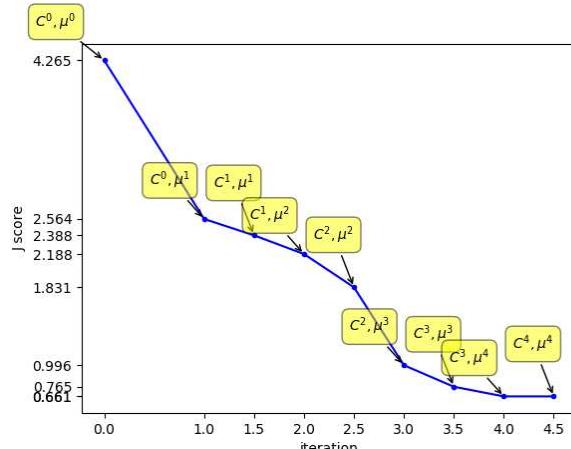
Aceasta înseamnă că $J(C_j^t, \mu) \geq J(C_j^t, \mu_j^{t+1})$, pentru $\forall \mu$. În particular, pentru $\mu = \mu_j^t$ vom avea: $J(C_j^t, \mu_j^t) \geq J(C_j^t, \mu_j^{t+1})$. Inegalitatea aceasta este valabilă pentru toate clusterele $j = 1, \dots, K$. Dacă sumăm toate aceste inegalități, rezultă: $J(C^t, \mu^t) \geq J(C^t, \mu^{t+1})$.

Mai rămâne de demonstrat inegalitatea (2): $J(C^t, \mu^{t+1}) \geq J(C^{t+1}, \mu^{t+1})$, corespunzătoare *pasului 2* din algoritm.

La acest pas, o instanță oarecare x_i , unde $i \in \{1, \dots, n\}$, este reasignată de la clusterul cu centroidul μ_j^{t+1} , la un alt centroid μ_q^{t+1} , dacă $\|x_i - \mu_j^{t+1}\|^2 \geq \|x_i - \mu_q^{t+1}\|^2$ pentru orice $j' = 1, \dots, K$. Această condiție este echivalentă cu următoarea: $(x_i - \mu_{j'}^{t+1})^2 \geq (x_i - \mu_q^{t+1})^2$ pentru orice j' . În contextul iterării t , acest lucru implică

$$(x_i - \mu_{C^t(x_i)}^{t+1})^2 \geq (x_i - \mu_{C^{t+1}(x_i)}^{t+1})^2.$$

Sumând membru cu membru inegalitățile de acest tip obținute pentru $i = \overline{1, n}$, rezultă: $J(C^t, \mu^{t+1}) \geq J(C^{t+1}, \mu^{t+1})$, ceea ce era de demonstrat.



Exemplu:

Pentru a ilustra grafic monotonia criteriului J , alăturat vă punem la dispoziție rezultatul obținut pe datele de la problema 10.a.⁴⁴⁹

b. Evident, numărul K -partițiilor care se pot forma cu cele n instanțe date este finit (K^n). A căuta minimul „criteriului“ J se poate face parcurgând în mod exhaustiv mul-

⁴⁴⁸Vedeți formula (5g) din documentul *Matrix Identities* de Sam Roweis.

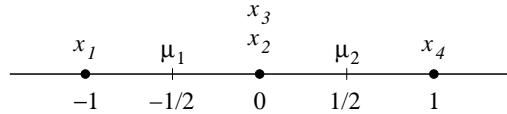
⁴⁴⁹Graficul a fost realizat de către studentul Gheorghe Balan de la Universitatea „Al. I. Cuza“ din Iași în semestrul I al anului universitar 2017-2018.

țimea acestor K -partiții,⁴⁵⁰ dar acest proces ar fi neficient (sau, practic imposibil de realizat) pentru valori mari ale lui n . Algoritmul K -means explorează — pornind de la o anumită inițializare a celor K centroizi —, doar un subset de K -partiții, asigurându-ne însă că are loc proprietatea $J(C^0, \mu^1) \geq J(C^1, \mu^2) \geq \dots \geq J(C^{t-1}, \mu^t) \geq J(C^t, \mu^{t+1})$, conform punctului precedent al acestei probleme. (În inegalitatea multiplă de mai sus, μ^{i+1} constituie vectorul de centroizi, i.e., mediile instanțelor din clusterele care compun K -partiția C^i .)

Dacă algoritmul revizitează o K -partiție, atunci rezultă că pentru un anumit t avem $J(C^{t-1}, \mu^t) = J(C^t, \mu^{t+1})$. Este posibil ca acest fapt să se întâmple, și anume atunci când:

- există instanțe multiple (i.e., $x_i = x_j$, deși $i \neq j$),
- criteriul de oprire al algoritmului K -means este de forma „până când componența clusterelor nu se mai modifică“,
- se presupune că, în cazul în care o instanță x_i este situată la egală distanță față de doi sau mai mulți centroizi, ea poate fi asignată în mod aleatoriu la oricare dintre ei.

Așa se întâmplă în *exemplul* din figura următoare



dacă se consideră că la o iterație t avem $x_2 = 0 \in C_1^t$ și $x_3 = 0 \in C_2^t$, iar la iterarea următoare alegem ca $x_3 = 0 \in C_1^{t+1}$ și $x_2 = 0 \in C_2^{t+1}$ și, din nou, invers la iterarea $t + 2$. Evident, dacă se impune restricția ca $x_2 = 0$ și $x_3 = 0$ să fie asignați la un același cluster,⁴⁵¹ atunci nu vom mai avea ciclare (iar $J(C, \mu)$ va avea în final valoarea minimă: $0 + \left(\frac{2}{3}\right)^2 + 2\left(\frac{1}{3}\right)^2 = \frac{2}{3}$.

Observația 1: Dacă se păstrează criteriul dat ca exemplu în enunțul problemei – adică se iterează până când centroizii „staționează“ – algoritmul se poate opri fără ca la ultima iterare $J(C, \mu)$ să fi atins minimul posibil. În cazul exemplului de mai sus, vom avea $\frac{1}{4} + 2 \cdot \frac{1}{4} + \frac{1}{4} = 1 > \frac{2}{3}$.

Observația 2: Dacă nu există instanțe multiple care să fie situate la distanțe egale față de doi sau mai mulți centroizi la o iterare oarecare a algoritmului K -means (precum sunt x_2 și x_3 în *exemplul* de mai sus), sau dacă se impune *restricția* ca în astfel de situații instanțele identice să fie asignate la un singur cluster, este evident că algoritmul K -means se oprește într-un număr finit de pași.

Observația 3: Atingerea minimului global al funcției $J(C, \mu)$ — unde C este o variabilă care parurge multimea tuturor K -partiților care se pot forma cu instanțele $\{x_1, \dots, x_n\}$ — nu este garantată pentru algoritmul K -means. Valoarea funcției J care se obține la

⁴⁵⁰Este suficient să explorăm doar spațiul format de valorile primului argument al criteriului J deoarece, conform inegalității (1), pentru orice poziționare a centroizilor μ , valoarea $J(C, \mu)$ este minorată de $J(C, \mu')$ unde μ' reprezintă pozițiile recalculate ale centroizilor [adică, centrele de greutate ale] clusterelor din K -partiția C , după cum se procedează la pasul 1 al algoritmului K -means.

⁴⁵¹O condiție mai generală poate fi următoarea: asignarea instanțelor x_i la centroizi să se facă astfel încât la pasul 1 inegalitatea (1) să fie satisfăcută în sens strict: $J(C^t, \mu^t) > J(C^t, \mu^{t+1})$.

oprirea algoritmului K -means este dependentă de plasarea inițială a centroizilor μ , precum și de modul concret în care sunt alcătuite clusterele în cazul în care o instanță oarecare se află la distanță egală de doi sau mai mulți centroizi, după cum am arătat în exemplul de mai sus.

Observația 4: Datorită rezultatului obținut la punctul a al acestui exercițiu, algoritmul de clusterizare K -means poate fi văzut [și reformulat] ca un *algoritm de optimizare*, folosind metoda *descreșterii pe coordonate*.⁴⁵² Ca *input*, vom considera (ca și în pseudo-codul anterior) punctele $x_1, \dots, x_n \in \mathbb{R}^d$, împreună cu numărul natural $K \geq 1$. *Obiectivul* este acela de a minimiza o funcție obiectiv care măsoară (indirect) coeziunea intra-clustere. Atunci când se folosește distanța euclidiană, această funcție este

$$J(L, \mu) = \sum_{i=1}^n \|x_i - \mu_{l_i}\|^2,$$

unde $\mu = (\mu_1, \dots, \mu_K)$ sunt centroizii clusterelor ($\mu_j \in \mathbb{R}^d$), iar lista $L = (l_1, \dots, l_n)$ memorează asignările instanțelor x_i la clustere ($l_i \in \{1, \dots, K\}$).⁴⁵³

Algoritmul K -means face *initializarea* centroizilor clusterelor μ cu anumite valori, după care procedează astfel:

Pasul 1: Păstrând μ fixat, găsește acea asignare L a instanțelor la clustere care minimizează funcția $J(L, \mu)$;⁴⁵⁴

Pasul 2: Păstrând asignarea L fixată, găsește acea valoare pentru μ pentru care se minimizează $J(L, \mu)$.⁴⁵⁵

Criteriul de oprire Dacă [aceasta nu este prima iterație și] niciuna dintre asignările din lista L nu s-a modificat în raport cu precedenta iterare, se trece la pasul următor (Terminare); altfel se repetă de la Pasul 1.

Terminare: Returnează L și μ .

13. (Algoritmul K -means: o proprietate de [*anti*]monotonie a valorilor minime pentru criteriul „sumei celor mai mici pătrate“ (J) în funcție de K)

CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1.1.d
CMU, 2010 fall, Aarti Singh, HW3, pr. 5.4

Cum evoluează valoarea minimă a funcției obiectiv J , aşa-numita sumă a celor mai mici pătrate (vedeți problema 12), la execuția algoritmului K -means pe un set de date arbitrar ales (dar fixat) atunci când valoarea lui K crește de la 1 la n ? Crește, scade, rămâne

⁴⁵²Ceea ce urmează în continuare este preluat — cu ușoare adaptări — din CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1.

⁴⁵³Găsirea minimului global al acestei funcții este un task *dificil* din punct de vedere computațional.

⁴⁵⁴Aceasta revine la a găsi pentru fiecare instanță x_i care este cel mai apropiat centroid. În cazul în care instanța se află la distanță minimă față de mai mulți centroizi, se pot folosi diverse euristică pentru a decide cum anume se face asignarea.

⁴⁵⁵Acesta este un pas simplu, care revine — în cazul în care se folosește distanța euclidiană — la a calcula pentru fiecare cluster media instanțelor (văzute ca vectori) care alcătuiesc clusterul respectiv.

constantă, variază arbitrar ori conform unei anumite legi / proprietăți? Justificați în mod riguros.

Răspuns:

Fie $X = \{x_1, \dots, x_n\}$ un set de instanțe de clusterizat. Întrucât mulțimea tuturor K -partițiilor ale lui X este [de fapt, poate fi văzută ca fiind] inclusă în mulțimea tuturor $(K+1)$ -partițiilor lui X ,⁴⁵⁶ rezultă că $\underline{J}_{K+1}(X) \leq \underline{J}_K(X)$, pentru orice $K = 1, \dots, n-1$. Am notat cu $\underline{J}_K(X)$ valoarea minimă a criteriului J pe mulțimea tuturor K -partițiilor C ale lui X (și similar $\underline{J}_{K+1}(X)$). Formal,

$$\begin{aligned}\underline{J}_K(X) &= \min\{J_K(X, C) \mid C \text{ este } K\text{-partiție a lui } X\} \\ J_K(X, C) &= \sum_{i=1}^n \|x_i - \mu_{C(x_i)}\|^2,\end{aligned}$$

unde $\mu_{C(x_i)}$ este centroidul clusterului $C(x_i)$, la care este asignat x_i în K -partiția (adică setul de clustere) C . Conform problemei 12.a, atunci când se folosește distanța euclidiană se poate considera

$$\mu_{C(x_i)} = \frac{1}{|C(x_i)|} \sum_{x_j \in C(x_i)} x_j.$$

14. (Clusterizarea ierarhică și algoritmul K -means: comparații)
CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, midterm, pr. 4.b

Enunțați câteva avantaje ale clusterizării ierarhice în raport cu clusterizarea bazată pe algoritmul K -means. Similar, enunțați câteva avantaje ale clusterizării bazate pe algoritmul K -means în raport cu clusterizarea ierarhică.

Răspuns:

Iată câteva avantaje ale clusterizării ierarhice:

- nu necesită fixarea dinainte a unui anumit număr (K) de clustere pe care dorim să le obținem;
- arborele ierarhic obținut poate fi tăiat la orice nivel, pentru a obține câte clustere dorim;
- pentru multe aplicații, ierarhia obținută este ușor de interpretat;
- se poate lucra cu date dispuse spațial într-o formă elongată (engl., long stringy data).

Și câteva avantaje ale clusterizării bazate pe algoritmul K -means:

- pe anumite seturi de date, rezultatul poate fi obținut mult mai rapid decât clusterizarea ierarhică;
- beneficiază de un cadru teoretic elegant;
- se pot incorpora ușor date noi și, de asemenea, este posibilă realcătuirea clusterelor în mod facil;
- poate fi kernel-izat (vedeți ex. 47).

⁴⁵⁶A se vedea cazul când, pentru o $(K+1)$ -partiție a lui X (să o notăm cu C_1, \dots, C_K, C_{K+1}), una dintre componente este mulțimea vidă.

7.1.3 Algoritmul EM pentru modele de mixturi gaussiene

15. (Algoritmul EM pentru GMM, cazul unidimensional,
 cu $\sigma_1 = \dots = \sigma_k$ și $\pi_1 = \dots = \pi_k$:
 demonstrația formulelor de „actualizare“)
 prelucrare de Liviu Ciortuz, după
 ■ Machine Learning, Tom Mitchell, 1997, pag. 193, 195–196.

Ne propunem să facem estimarea parametrilor unui model de mixtură de k distribuții gaussiene (engl., Gaussian Mixture Model, GMM), despre care se presupune că au [toate] aceeași varianță σ^2 , iar probabilitățile a priori de selecție sunt egale ($1/k$). Prin urmare, se vor estima doar mediile acestor k distribuții gaussiene. Pentru aceasta, vom considera următorul *algoritm EM*:

Inițializare: Date fiind instanțele $x_1, \dots, x_n \in \mathbb{R}$, se aleg în mod arbitrar valori pentru mediile $\mu \stackrel{\text{not.}}{=} (\mu_1, \dots, \mu_k)$. Vom nota aceste valori inițiale cu $\mu_1^{(0)}, \dots, \mu_k^{(0)}$.

Pentru $t = 0, 1, \dots$, atâtă timp cât nu este îndeplinit *criteriul de oprire*,⁴⁵⁷ execută:

Pasul E:

$$\begin{aligned} E[Z_{ij}] &\stackrel{\text{not.}}{=} E[Z_{ij}|X = x_i; \mu^{(t)}, \sigma^2] = \dots = P(Z_{ij} = 1|X = x_i; \mu^{(t)}, \sigma^2) \\ &\stackrel{F. Bayes}{=} \dots \\ &\stackrel{p.d.f.}{=} \dots \quad \text{pentru } i = 1, \dots, n \text{ și } j = 1, \dots, k. \end{aligned}$$

Pasul M:

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^n E[Z_{ij}]x_i}{\sum_{i=1}^n E[Z_{ij}]}, \quad \text{pentru } j = 1, \dots, k, \quad (160)$$

unde Z_{ij} , cu $i \in \{1, \dots, n\}$ și $j \in \{1, \dots, k\}$, sunt variabile-indicator *neobservabile* (sau „ascunse“ / „latente“), luând valoarea 1 dacă x_i a fost generat de gaussiana j și 0 în caz contrar.

Elaborați formula de la pasul E și demonstrați formula (160) pentru pasul M.

Sugestie: Pentru pasul M,

- veți scrie mai întâi $p(y_i|\mu)$ verosimilitatea unei instanțe „complete“, $y_i \stackrel{\text{not.}}{=} (x_i, z_{i1}, \dots, z_{ik})$,⁴⁵⁸
- apoi veți scrie $\ln P(Y|\mu)$, funcția de log-verosimilitate a datelor complete $Y \stackrel{\text{not.}}{=} \{y_1, \dots, y_n\}$;
- veți calcula funcția „auxiliară“ $Q(\mu|\mu^{(t)}) \stackrel{\text{def.}}{=} E[\ln P(Y|\mu)]$, care este media funcției de log-verosimilitate a datelor complete în raport cu distribuția probabilistă a posteriori a variabilelor neobservabile Z_{ij} în raport cu datele observabile x_i și cu valorile parametrilor μ de la iterată t , adică $\mu^{(t)}$;
- în final, veți calcula valorile mediilor μ_j pentru care se obține valoarea optimă a funcției „auxiliare“ $Q(\mu|\mu^{(t)})$.

⁴⁵⁷ De exemplu, se execută un anumit număr de iterări, fixat în prealabil.

⁴⁵⁸ Pentru un exemplu simplu de definire și calculare a funcției de verosimilitate, vedeți ex. 39 de la capitolul *Fundamente*.

Răspuns:

Pasul E:

Așa cum a fost sugerat în enunț, mai întâi arătăm că media variabilei-indicator Z_{ij} (văzută ca variabilă aleatoare) este $P(Z_{ij} = 1 | x_i, \mu^{(t)})$.⁴⁵⁹ Apoi calculăm această probabilitate condiționată folosind formula lui Bayes combinată cu formula probabilității totale, iar la final ținem cont de expresia funcției de densitate pentru distribuția gaussiană.⁴⁶⁰

$$\begin{aligned} E[Z_{ij}] &\stackrel{\text{def.}}{=} 0 \cdot P(Z_{ij} = 0 | x_i, \mu^{(t)}) + 1 \cdot P(Z_{ij} = 1 | x_i, \mu^{(t)}) \\ &= P(Z_{ij} = 1 | x_i, \mu^{(t)}) \\ &\stackrel{T.\text{Bayes}}{=} \frac{P(x_i | Z_{ij} = 1; \mu^{(t)}) \cdot \overbrace{P(Z_{ij} = 1 | x_i, \mu^{(t)})}^{1/k}}{\sum_{l=1}^k (x_i | x_i, Z_{il} = 1; \mu^{(t)}) \cdot \underbrace{P(Z_{il} = 1 | x_i, \mu^{(t)})}_{1/k}} \\ &\stackrel{p.d.f.}{=} \frac{\mathcal{N}(X = x_i | \mu = \mu_j^{(t)})}{\sum_{l=1}^k \mathcal{N}(X = x_i | \mu = \mu_l^{(t)})} = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j^{(t)})^2}}{\sum_{l=1}^k e^{-\frac{1}{2\sigma^2}(x_i - \mu_l^{(t)})^2}}. \end{aligned}$$

Vă readucem aminte că probabilitățile a priori $P(Z_{il} = 1 | x_i, \mu^{(t)})$ au fost considerate de la început identice, indiferent de valoarea lui l .

Pasul M:

Pentru a calcula probabilitatea unei date complete $y_i \stackrel{\text{not.}}{=} (x_i, z_{i1}, \dots, z_{ik})$, vom folosi formula de multiplicare [a probabilităților] și, din nou, expresia funcției de densitate pentru distribuția gaussiană.

$$\begin{aligned} p(y_i | \mu) &\stackrel{\text{not.}}{=} p(x_i, z_{i1}, \dots, z_{ik} | \mu) = p(x_i | z_{i1}, \dots, z_{ik}; \mu) \underbrace{p(z_{i1}, \dots, z_{ik} | \mu)}_{1/k} \\ &= \frac{1}{k} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij}(x_i - \mu_j)^2} \end{aligned}$$

Funcția de log-verosimilitate a datelor complete $Y = \{y_1, \dots, y_n\}$ se calculează ușor, ținând cont de independența creării datelor x_i , precum și de rezultatul obținut anterior.

$$\begin{aligned} \ln P(Y | \mu) &\stackrel{i.i.d.}{=} \ln \prod_{i=1}^n p(y_i | \mu) = \sum_{i=1}^n \ln p(y_i | \mu) \\ &= \sum_{i=1}^n \left(-\ln k + \ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij}(x_i - \mu_j)^2 \right). \end{aligned}$$

Observați introducerea în ultima expresie a variabilelor-indicator Z_{ij} . Se ține cont de faptul că pentru fiecare $i = 1, \dots, n$, avem $\sum_{j=1}^k Z_{ij} = 1$, deci doar una dintre aceste

⁴⁵⁹ Pentru o formulare generală a acestei proprietăți, vedeti problema 87 de la capitolul de *Fundamente*.

⁴⁶⁰ Vedeti problema 29 de la capitolul de *Fundamente*.

variabile-indicator are valoarea 1 (și anume, cea care corespunde gaussienei care a generat instanța x_i), în vreme ce toate celelalte variabile-indicator au valoarea 0.

Media funcției de log-verosimilitate a datelor complete (așa-numita *funcție auxiliară*) se obține aplicând proprietatea de *liniaritate a mediei*.⁴⁶¹

$$\begin{aligned} Q(\mu^{(t)}|\mu) &= E[\ln P(Y|\mu)] \stackrel{\text{lin. med.}}{=} \\ &\sum_{i=1}^n (-\ln k + \ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[Z_{ij}](x_i - \mu_j)^2). \end{aligned} \quad (161)$$

Se poate observa că aceasta este o funcție de gradul al doilea în raport cu variabilele μ_j .⁴⁶² Se constată apoi că optimul ei se poate obține prim maximizare separată (sau, pe rând) în raport cu fiecare variabilă μ_j .

$$\begin{aligned} Q(\mu|\mu^{(t)}) &= n(-\ln k + \ln \frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2} \sum_{i=1}^n E[Z_{ij}] \sum_{j=1}^k E[Z_{ij}](x_i - \mu_j)^2 \\ &= n(-\ln k + \ln \frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2} \sum_{j=1}^k \sum_{i=1}^n E[Z_{ij}](x_i - \mu_j)^2 \\ &= const - \frac{1}{2\sigma^2} \sum_{j=1}^k \sum_{i=1}^n E[Z_{ij}](x_i^2 - 2\mu_j x_i + \mu_j^2) \\ &= const' - \frac{1}{2\sigma^2} \sum_{j=1}^k \left[(E[Z_{ij}])\mu_j^2 - 2 \left(\sum_{i=1}^n E[Z_{ij}]x_i \right) \mu_j \right]. \end{aligned}$$

Pentru fiecare $j = 1, \dots, k$, coeficientul dominant al lui μ_j este strict pozitiv (fiindcă $E[Z_{ij}] = P(Z_{ij} = 1 | x_i, \mu^{(t)}) > 0$, conform rezultatului de la pasul E). Așadar, soluția optimă este dată de relația

$$\mu_j^{(t+1)} \stackrel{\text{def.}}{=} \underset{\mu_j}{\operatorname{argmax}} Q(\mu|\mu^{(t)}) = -\frac{-2\frac{1}{2\sigma^2} \sum_{i=1}^n E[Z_{ij}] x_i}{2\frac{1}{2\sigma^2} \sum_{i=1}^n E[Z_{ij}]} = \frac{\sum_{i=1}^n E[Z_{ij}] x_i}{\sum_{i=1}^n E[Z_{ij}]}.$$

Observație: Rezultatul acesta (în enunț, formula (160)) reprezintă media ponderată a instanțelor x_i , $i = 1, \dots, n$. Așadar, această formulă reprezintă o generalizare a formulei pentru estimarea mediei (în sens MLE) pentru o distribuție gaussiană unidimensională (vedeți rezolvarea problemei 45.a de la capitolul *Fundamente*).

⁴⁶¹Vedeți problema 9.a de la capitolul de *Fundamente*.

⁴⁶²Așadar, pentru $k = 2$ graficul unei astfel de funcții este un *paraboloid*.

16. (Algoritmul EM/GMM, cazul $\sigma_1^2 = \sigma_2^2 = 1$, $\pi_1 = \pi_2 = 1/2$: executarea unei iterații)
prelucrare de Liviu Ciortuz, după CMU, 2012 spring, Ziv Bar-Joseph, final exam, pr. 3.1

Fie un model de mixtură gaussiană (engl., Gaussian mixture model, GMM) cu două componente având varianțe cunoscute și probabilități *a priori* egale pentru selecția celor două distribuții:

$$\frac{1}{2}\mathcal{N}(x|\mu_1, 1) + \frac{1}{2}\mathcal{N}(x|\mu_2, 1), \quad x \in \mathbb{R}.$$

În continuare se va considera că $n = 2$, $x_1 = 0.5$ și $x_2 = 2$, iar valorile inițiale pentru μ_1 și μ_2 sunt 1 și respectiv 2.

Execuți în mod manual o iterație a algoritmului EM din enunțul problemei 15 (preluat din carte Machine Learning a lui Tom Mitchell, pag. 193) pe aceste date, astfel:

- a. (Pasul E) Calculați mai întâi $P(Z_{ij} = 1|x_i, \mu)$, probabilitățile a posteriori de apartenență a datelor observate (x_1 și x_2) la cele două componente ale mixturii. Am folosit notația $\mu = (\mu_1, \mu_2)$, iar Z_{ij} este variabilă-indicator având valoarea 1 dacă instanța x_i a fost generată de gaussiana j și 0 în cazul contrar.

Indicație: În vederea efectuării calculelor, pentru conveniență puteți considera valorile distribuției normale / gaussiene standard $\mathcal{N}(x|\mu = 0, \sigma^2 = 1)$ în punctele 0, 0.5, 1, 1.5 și 2 ca fiind respectiv 0.4, 0.35, 0.24, 0.13 și 0.05.

- b. (Pasul M) Re-calculați valorile parametrilor μ_1 și μ_2 în funcție de probabilitățile calculate la punctul precedent. Care credeți că va fi tendința de mișcare a mediilor la următoarele iterații?

- c. Funcția de log-verosimilitate a datelor observabile este definită aşa cum este de așteptat, ca logaritmul unei funcții de probabilitate (marginală în raport cu distribuția comună $P(x, z|\mu)$):

$$\ell(\mu_1, \mu_2) \stackrel{\text{def.}}{=} \ln P(x_1, x_2|\mu_1, \mu_2) \stackrel{\text{indep.}}{=} \sum_{i=1}^2 \ln P(x_i|\mu_1, \mu_2) = \sum_{i=1}^2 \ln \left(\sum_{z_{ij}} P(x_i, z_{ij}|\mu_1, \mu_2) \right),$$

unde simbolul $\sum_{z_{ij}}$ desemnează parcurgerea tuturor asignărilor posibile pentru variabilele neobservabile Z_{ij} (adică, mai întâi $z_{11} = 1$ și $z_{21} = 1$, apoi $z_{11} = 1$ și $z_{21} = 0$, după care $z_{11} = 0$ și $z_{21} = 1$ și, în final, $z_{11} = 0$ și $z_{21} = 0$). Explicați expresia acestei funcții (păstrând μ_1 și μ_2 nespecificați). După aceea, calculați valoarea ei la începutul și respectiv la sfârșitul primei iterații a algoritmului EM pe datele x_1 și x_2 specificate în prima parte a acestui exercițiu.

Răspuns:

- a. Folosind teorema lui Bayes, vom exprima probabilitățile de apartenență ale instanțelor x_1 și respectiv x_2 la clusterul reprezentat de prima gaussiană:

$$\begin{aligned} P(Z_{i1} = 1|x_i, \mu) &\stackrel{T.B.}{=} \frac{P(x_i|Z_{i1} = 1, \mu_1)P(Z_{i1} = 1)}{P(x_i|Z_{i1} = 1, \mu_1)P(Z_{i1} = 1) + P(x_i|Z_{i2} = 1, \mu_2)P(Z_{i2} = 1)} \\ &= \frac{P(x_i|Z_{i1} = 1, \mu_1) \cdot \frac{1}{2}}{P(x_i|Z_{i1} = 1, \mu_1) \cdot \frac{1}{2} + P(x_i|Z_{i2} = 1, \mu_2) \cdot \frac{1}{2}} \\ &= \frac{P(x_i|Z_{i1} = 1, \mu_1)}{P(x_i|Z_{i1} = 1, \mu_1) + P(x_i|Z_{i2} = 1, \mu_2)} \text{ pentru } i \in \{1, 2\}. \end{aligned}$$

Tinând cont de faptul că valorile oricărei *distribuții gaussiene unidimensionale* pot fi puse în corespondență cu valorile *distribuției gaussiene standard*,⁴⁶³ utilizând valorile furnizate în *indicația* din enunț și folosind proprietatea de simetrie a valorilor distribuției gaussiene standard în raport cu originea, vom avea:

$$\begin{aligned} P(Z_{11} = 1|x_1, \mu) &= \frac{\mathcal{N}(0.5|1, 1)}{\mathcal{N}(0.5|1, 1) + \mathcal{N}(0.5|2, 1)} = \frac{\mathcal{N}(0.5|0, 1)}{\mathcal{N}(0.5|0, 1) + \mathcal{N}(1.5|0, 1)} \\ &= \frac{0.35}{0.35 + 0.13} = \frac{0.35}{0.48} = \frac{35}{48} \\ P(Z_{21} = 1|x_2, \mu) &= \frac{\mathcal{N}(2|1, 1)}{\mathcal{N}(2|1, 1) + \mathcal{N}(2|2, 1)} = \frac{\mathcal{N}(1|0, 1)}{\mathcal{N}(1|0, 1) + \mathcal{N}(0|0, 1)} \\ &= \frac{0.24}{0.24 + 0.4} = \frac{0.24}{0.64} = \frac{3}{8} \end{aligned}$$

În sfârșit, datorită faptului că $Z_{i1} + Z_{i2} = 1$ pentru $\forall i \in \{1, 2\}$ (cu $Z_{ij} \in \{0, 1\}$) și, de asemenea, datorită complementarității evenimentelor aleatoare, vom obține imediat și celelalte două probabilități cerute:

$$\begin{aligned} P(Z_{12} = 1|x_1, \mu) &= P(Z_{11} = 0|x_1, \mu) = 1 - P(Z_{11} = 1|x_1, \mu_1) = \frac{13}{48} \\ P(Z_{22} = 1|x_2, \mu) &= P(Z_{21} = 0|x_2, \mu) = 1 - P(Z_{21} = 1|x_2, \mu_1) = \frac{5}{8} \end{aligned}$$

b. Formulele (160) pentru actualizarea mediilor μ_j pot fi scrise folosind notațiile de aici (adică, explicitând direct mediile variabilelor-indicator Z_{ij}), astfel:⁴⁶⁴

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^2 P(Z_{ij} = 1|x_i, \mu^{(t)}) x_i}{\sum_{i=1}^2 P(Z_{ij} = 1|x_i, \mu^{(t)})}$$

Prin urmare,

$$\mu_1^{(1)} = \frac{\frac{35}{48} \cdot 0.5 + \frac{3}{8} \cdot 2}{\frac{35}{48} + \frac{3}{8}} = \frac{107}{106} \approx 1.009 \text{ și } \mu_2^{(1)} = \frac{\frac{13}{48} \cdot 0.5 + \frac{5}{8} \cdot 2}{\frac{13}{48} + \frac{5}{8}} = \frac{133}{86} \approx 1.547$$

Se observă că, în raport cu pozițiile inițiale, mediile celor două gaussiene au „migrat“ astfel: μ_1 foarte puțin spre dreapta, iar μ_2 considerabil mai la stânga. În noile poziții, mediile $\mu_1^{(1)}$ și $\mu_2^{(1)}$ sunt aproape simetrice în raport cu mijlocul segmentului determinat de punctele $x_1 = 0.5$ și $x_2 = 2$. Aceasta implică faptul că la pasul următor vom avea $P(Z_{11} = 1|x_1, \mu^{(1)}) \approx P(Z_{22} = 1|x_2, \mu^{(1)})$ și $P(Z_{12} = 1|x_1, \mu^{(1)}) \approx P(Z_{21} = 1|x_2, \mu^{(1)})$. Executând încă o iterație (nu dăm aici detaliile) obținem $\mu_1^{(2)} \approx 1.110$ și $\mu_2^{(2)} \approx 1.390$. Ne așteptăm ca pe parcursul următoarelor iterări mediile $\mu_1^{(t)}$ și $\mu_2^{(t)}$ să migreze către mijlocul intervalului $[0.5, 2]$.

⁴⁶³Vom folosi formula pentru „standardizare“ $\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma} \mathcal{N}\left(\frac{x-\mu}{\sigma}|0, 1\right)$, care este ușor de demonstrat, precum și proprietatea $\mathcal{N}(x|0, 1) = \mathcal{N}(-x|0, 1)$.

⁴⁶⁴Expresia *funcției auxiliare* $Q(\mu, \mu^{(t)})$ pentru care $\mu^{(t+1)} \stackrel{\text{not.}}{=} (\mu_1^{(t+1)}, \mu_2^{(t+1)})$ constituie absisa punctului de optim se obține — pentru acest caz particular — din formula 161 de la problema 15.

c. Vom face mai întâi explicitarea expresiei care a fost dată în enunț pentru funcția ℓ , specificând apoi termenii ei de bază într-un tabel:

$$\begin{aligned}
 \ell(\mu_1, \mu_2) &= \sum_{i=1}^2 \ln \left(\sum_{z_{ij}} P(x_i, Z_{ij} = z_{ij} | \mu_1, \mu_2) \right) \\
 &= \sum_{i=1}^2 \ln \left(\sum_{z_{ij}} P(x_i | z_{ij}, \mu_1, \mu_2) \cdot \underbrace{P(z_{ij} | \mu_1, \mu_2)}_{1/2} \right) \\
 &= \sum_{i=1}^2 \ln \left(\frac{1}{2} \sum_{z_{ij}} P(x_i | z_{ij}, \mu_1, \mu_2) \right) = \sum_{i=1}^2 \left[-\ln 2 + \ln \left(\sum_{j=1}^2 P(x_i | z_{ij} = 1, \mu_j) \right) \right]
 \end{aligned}$$

z_{ij}	$P(x_i z_{ij}, \mu_1, \mu_2)$
$z_{11} = 1, z_{12} = 0$	$\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_1 - \mu_1)^2)$
$z_{11} = 0, z_{12} = 1$	$\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_1 - \mu_2)^2)$
$z_{21} = 1, z_{22} = 0$	$\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_2 - \mu_1)^2)$
$z_{21} = 0, z_{22} = 1$	$\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_2 - \mu_2)^2)$

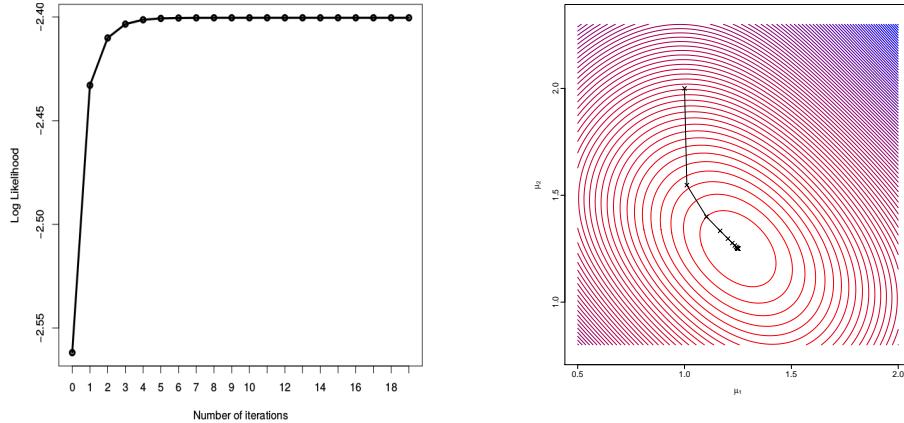
Prin urmare,

$$\begin{aligned}
 \ell(\mu_1, \mu_2) &= -2 \ln 2 + \ln \left(\frac{1}{\sqrt{2\pi}} \left(\exp \left(-\frac{1}{2}(x_1 - \mu_1)^2 \right) + \exp \left(-\frac{1}{2}(x_1 - \mu_2)^2 \right) \right) \right) \\
 &\quad + \ln \left(\frac{1}{\sqrt{2\pi}} \left(\exp \left(-\frac{1}{2}(x_2 - \mu_1)^2 \right) + \exp \left(-\frac{1}{2}(x_2 - \mu_2)^2 \right) \right) \right) \\
 &= -2 \ln 2 - \ln(2\pi) + \ln \left(\exp \left(-\frac{1}{2}(x_1 - \mu_1)^2 \right) + \exp \left(-\frac{1}{2}(x_1 - \mu_2)^2 \right) \right) \\
 &\quad + \ln \left(\exp \left(-\frac{1}{2}(x_2 - \mu_1)^2 \right) + \exp \left(-\frac{1}{2}(x_2 - \mu_2)^2 \right) \right)
 \end{aligned}$$

Valoarea acestei funcții ca urmare a inițializării mediilor μ_1 și μ_2 cu valorile 1 și respectiv 2 este -2.561833 , iar la finalul primei iterării a algoritmului EM ea devine -2.462877 . Se observă că valoarea funcției de log-verosimilitate crește, aşa cum era de așteptat (vedeți problema 2 de la capitolul *Algoritmul EM*).

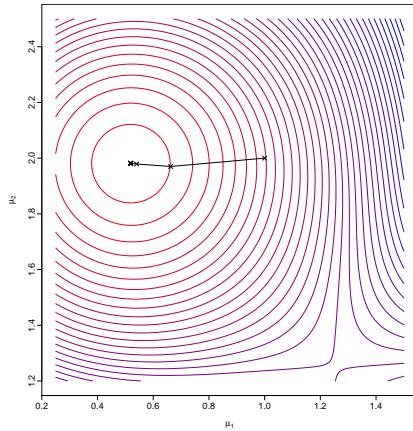
Pentru [confirmarea și] extinderea acestui rezultat, prezentăm mai jos rezultatele obținute cu ajutorul unei implementări a algoritmului EM:⁴⁶⁵

⁴⁶⁵Această implementare a fost realizată în 2018 de către studentul Sebastian Ciobanu.



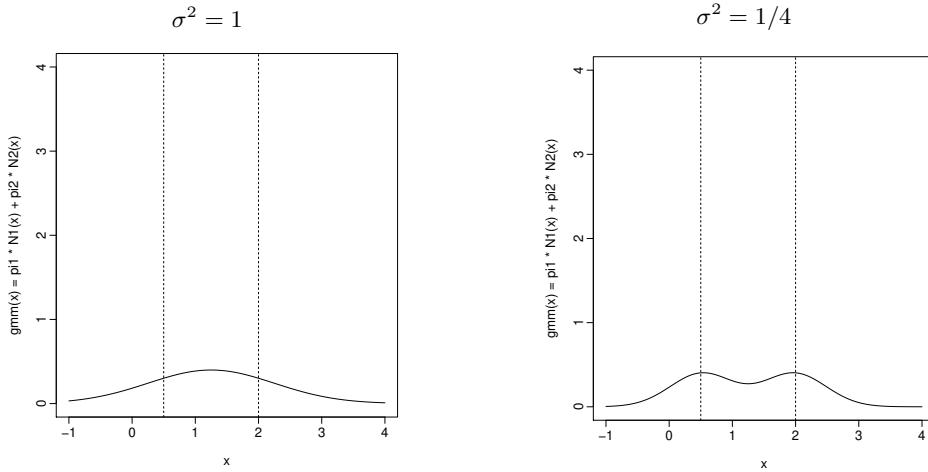
- în graficul din partea stângă sunt reprezentate valorile funcției de log-verosimilitate obținute de EM la inițializare și apoi la finalul fiecărei din primele 19 iterări,
- în graficul din partea dreaptă avem reprezentarea sub formă de curbe de izocontur a valorilor log-verosimilității în funcție de cei doi parametri, μ_1 și μ_2 . Pe acest al doilea grafic a fost adăugat un „drum“ care pune în evidență succesiunea de valori pentru perechile (μ_1, μ_2) de-a lungul iterărilor executate de algoritmul EM.

Observație (1): Se poate vedea în figura de mai sus, partea dreaptă, că mediile μ_1 și μ_2 vor tinde la convergență la aceeași valoare, 1.25, valoare care reprezintă jumătatea intervalului $[0.5, 2]$. Este interesant de constatat că dacă am fi lucrat cu valori mici ale varianței (de exemplu, $\sigma^2 = 1/4$, caz ilustrat în figura alăturată), atunci la convergență μ_1 ar fi tins la valoarea $x_1 = 0.5$, în vreme ce μ_2 ar fi tins la valoarea $x_2 = 2$. La problema 57 se va vedea că în general, atunci când $\sigma^2 \rightarrow 0$, algoritmul EM/GMM tinde să obțină (la convergență) aceleași rezultate ca și algoritmul K-means!



Observație (2): Modelele probabiliste obținute „învățate“ în urma aplicării algoritmului EM pentru cele două cazuri ($\sigma^2 = 1$ și $\sigma^2 = 1/4$) sunt prezentate în graficele următoare.⁴⁶⁶ În primul caz se poate observa că mixtura obținută coincide practic cu o distribuție gaussiană a cărei medie este situată la jumătatea intervalului determinat de cele două instanțe, x_1 și x_2 . În cel de-al doilea caz se poate constata că mediile celor două gaussiene care compun mixtura coincid cu (de fapt, sunt situate foarte aproape de) cele două instanțe, x_1 și x_2 .

⁴⁶⁶Aceste grafice, ca și cel precedent, au fost realizate de Sebastian Ciobanu.



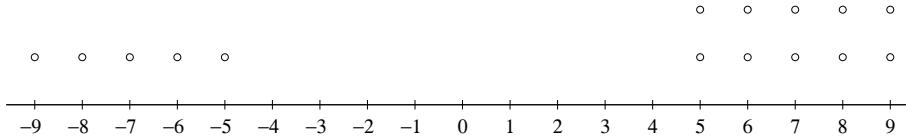
17. (Algoritmii K -means și EM pentru GMM, cazul unidimensional: aplicare)

prelucrare făcută de L. Ciortuz și A. Munteanu, după Edinburgh, 2009 fall, C. Williams, V. Lavrenko, HW4, pr. 3

Se consideră următorul set de instanțe – puncte în spațiul unidimensional:

$$-9, -8, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9$$

Acstea date au reprezentarea următoare:



Obiectivul este să se aplice pe de o parte algoritmul K -means și pe de altă parte algoritmul EM pentru un model de mixtură de distribuții gaussiene (GMM) ca să împărțim acest set de date în două clustere. Drept centroizi inițiali se consideră punctele -10 și -20 . Este evident că această alegere este una nefavorabilă / nestandard, însă dorim să comparăm evoluțiile celor doi algoritmi în acest caz.

- Să se elaboreze calculele numerice corespunzătoare aplicării algoritmului K -means pe acest set de date. Care sunt clusterele finale și centroizii corespunzători?
- Se consideră algoritmul EM/GMM descris în enunțul problemei 15. El a fost conceput pentru a face estimarea parametrilor unui model de mixtură de două distribuții gaussiene, pentru care se presupune că ambele distribuții au aceeași varianță σ^2 , iar probabilitățile a priori de selecție sunt egale ($1/2$). Prin urmare, se vor estima doar mediile celor două distribuții gaussiene (i.e., centroizii clusterelor corespunzătoare), aplicându-se formulele următoare:⁴⁶⁷

⁴⁶⁷Pentru deducerea acestor formule, vedeti problema 15 sau cartea *Machine Learning*, de Tom Mitchell, 1997, pag. 193, 195–196.

Pasul E:

$$E[Z_{ij}] = P(Z_{ij} = 1 | X = x_i; \mu, \sigma^2) \stackrel{F.Bayes}{=} \frac{\exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_j)^2\right)}{\sum_{p=1}^k \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu_p)^2\right)}$$

Pasul M:

$$\mu_j = \frac{\sum_{i=1}^n E[Z_{ij}]x_i}{\sum_{i=1}^n E[Z_{ij}]}$$

unde simbolul \exp denotă funcția exponentială (e^x), $n = 15$ este numărul de puncte, $k = 2$ este numărul de clustere, $\mu \stackrel{not.}{=} (\mu_1, \mu_2)$, iar Z_{ij} sunt variabilele-indicator „ascunse” / „latente”, cu $i = \overline{1, n}$ și $j = \overline{1, k}$.

Să se calculeze valorile numerice corespunzătoare execuției primei iterării a acestui algoritm, pornind ca și la punctul a cu valorile inițiale $\mu_1 = -20$, $\mu_2 = -10$ și considerând varianța (fixată) $\sigma^2 = 1$.

c. Se consideră o altă variantă a algoritmului EM (de asemenea pentru GMM, cazul unidimensional), în care însă se estimează toți parametrii (μ , σ , π). Ecuatiile de actualizare pentru parametrii primei distribuții gaussiene (desemnată prin a) sunt date mai jos, iar cele pentru cea de-a doua gaussiană (b) sunt obținute pur și simplu înlocuind a cu b :⁴⁶⁸

Pasul E:

$$a_i \stackrel{not.}{=} P(a | x_i) \stackrel{F.Bayes}{=} \frac{p(x_i | a) \cdot \pi_a}{p(x_i | a) \cdot \pi_a + p(x_i | b) \cdot \pi_b}, \text{ unde}$$

$$p(x_i | a) \stackrel{def.}{=} \frac{1}{\sqrt{2\pi}\sigma_a} \cdot \exp\left(-\frac{(x_i - \mu_a)^2}{2\sigma_a^2}\right)$$

Pasul M:

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \dots + a_n x_n}{a_1 + a_2 + \dots + a_n}$$

$$\sigma_a^2 = \frac{a_1(x_1 - \mu_a)^2 + a_2(x_2 - \mu_a)^2 + \dots + a_n(x_n - \mu_a)^2}{a_1 + a_2 + \dots + a_n}$$

$$\pi_a = (a_1 + a_2 + \dots + a_n)/n$$

Presupunem că valorile inițiale pentru parametri sunt: $\mu_a = -20$, $\mu_b = -10$, $\sigma_a^2 = \sigma_b^2 = 1$ și $\pi_a = \pi_b = 0.5$. Efectuați o singură iterare a acestui algoritm EM, pentru a determina noile valori ale parametrilor μ_a , μ_b , σ_a^2 , σ_b^2 , π_a și π_b . Continuând rularea acestui algoritm EM, valorile parametrilor vor rămâne neschimbate? Dacă da, de ce? Dacă nu, de ce?

Răspuns:

a. Algoritmul K -means va executa următoarele iterării:

Iterația 1:

$$\begin{aligned} \mu_1 &= -20 & C_1 &= \emptyset \\ \mu_2 &= -10 & C_2 &= \{-9, -8, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9\} \end{aligned}$$

Iterația 2:

$$\begin{aligned} \mu_1 &= -20 & C_1 &= \{-9\} \\ \mu_2 &= \frac{-9 + \dots + 9}{15} = \frac{7}{3} = 2.(3) & C_2 &= \{-8, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9\} \end{aligned}$$

⁴⁶⁸Pentru deducerea acestor formule, vedeți rezolvarea problemei 18.

Iterația 3:

$$\begin{aligned}\mu_1 &= -9 & C_1 &= \{-9, -8, -7, -6, -5\} \\ \mu_2 &= \frac{-8 + \dots + 9}{14} = \frac{44}{14} = \frac{22}{7} \approx 3.143 & C_2 &= \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}\end{aligned}$$

Iterația 4:

$$\begin{aligned}\mu_1 &= \frac{-9 - 8 - 7 - 6 - 5}{5} = -7 & C_1 &= \{-9, -8, -7, -6, -5\} \\ \mu_2 &= \frac{2(5 + 6 + 7 + 8 + 9)}{10} = 7 & C_2 &= \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}\end{aligned}$$

După această iterareție algoritmul K-means se oprește, întrucât centroizii nu se mai modifică. Așadar, clusterele finale sunt:

$$C_1 = \{-9, -8, -7, -6, -5\} \text{ și } C_2 = \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}, \text{ cu } \mu_1 = -7 \text{ și } \mu_2 = 7.$$

b. O iterareție a algoritmului EM/GMM în această variantă, cu valorile inițiale $\mu_1 = -20$, $\mu_2 = -10$ și $\sigma^2 = 1$ constă în calcularea mediilor variabilelor „ascunse“ Z_{ij} (pasul E) și apoi recalcularea mediilor, adică a parametrilor μ_1 și μ_2 (pasul M).

Așadar, la pasul E, pentru punctul $x_1 = -9$, vom avea:

$$\begin{aligned}E[Z_{11}] &= \frac{\exp\left(-\frac{1}{2}(-9+20)^2\right)}{\exp\left(-\frac{1}{2}(-9+10)^2\right) + \exp\left(-\frac{1}{2}(-9+20)^2\right)} \\ &= \frac{\exp\left(-\frac{121}{2}\right)}{\exp\left(-\frac{1}{2}\right) + \exp\left(-\frac{121}{2}\right)} = \frac{e^{-60}}{1 + e^{-60}} \\ E[Z_{12}] &= \frac{\exp\left(-\frac{1}{2}(-9+10)^2\right)}{\exp\left(-\frac{1}{2}(-9+10)^2\right) + \exp\left(-\frac{1}{2}(-9+20)^2\right)} \\ &= \frac{\exp\left(-\frac{1}{2}\right)}{\exp\left(-\frac{1}{2}\right) + \exp\left(-\frac{121}{2}\right)} = \frac{1}{1 + e^{-60}}\end{aligned}$$

Este evident că $P(Z_{11} = 1|x_1) = E[Z_{11}] \approx 0$, iar $P(Z_{12} = 1|x_1) = E[Z_{12}] \approx 1$. Așadar, dacă decidem să asociem fiecare instanță x_i la clusterul / centroidul desemnat de $\operatorname{argmax}_j E[Z_{ij}]$, punctul $x_1 = -9$ va fi asignat celui de-al doilea cluster.

Pentru punctul $x_2 = -8$:

$$\begin{aligned}E[Z_{21}] &= \frac{\exp\left(-\frac{1}{2} \cdot 144\right)}{\exp\left(-\frac{1}{2} \cdot 4\right) + \exp\left(-\frac{1}{2} \cdot 144\right)} = \frac{e^{-70}}{1 + e^{-70}} \\ E[Z_{22}] &= \frac{\exp\left(-\frac{1}{2} \cdot 4\right)}{\exp\left(-\frac{1}{2} \cdot 4\right) + \exp\left(-\frac{1}{2} \cdot 144\right)} = \frac{1}{1 + e^{-70}}\end{aligned}$$

Ca și mai sus, este evident că $E[Z_{21}] \approx 0$, iar $E[Z_{22}] \approx 1$. Deci punctul $x_2 = -8$ va fi asociat tot celui de-al doilea cluster.

Pentru punctul $x_3 = -7$:

$$E[Z_{31}] = \frac{\exp\left(-\frac{1}{2} \cdot 169\right)}{\exp\left(-\frac{1}{2} \cdot 9\right) + \exp\left(-\frac{1}{2} \cdot 169\right)} = \frac{e^{-80}}{1+e^{-80}}$$

$$E[Z_{32}] = \frac{\exp\left(-\frac{1}{2} \cdot 9\right)}{\exp\left(-\frac{1}{2} \cdot 9\right) + \exp\left(-\frac{1}{2} \cdot 169\right)} = \frac{1}{1+e^{-80}}$$

Este evident că $E[Z_{31}] \approx 0$, iar $E[Z_{32}] \approx 1$. Deci punctul $x_3 = -7$ va fi asociat de asemenea celui de-al doilea cluster.

Se observă că pentru celelalte puncte se va păstra acest comportament, deci $C_1 = \emptyset$ și $C_2 = \{-9, -8, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}$, iar $E[Z_{i1}] \approx 0, E[Z_{i2}] \approx 1$ pentru $i = \overline{1, 15}$.

Apoi, la pasul de *maximizare*, trebuie să calculăm noile valori ale mediilor:

$$\mu_1 = \frac{\frac{e^{-60}}{1+e^{-60}} \cdot (-9) + \frac{e^{-70}}{1+e^{-70}} \cdot (-8) + \frac{e^{-80}}{1+e^{-80}} \cdot (-7) + \dots}{\frac{e^{-60}}{1+e^{-60}} + \frac{e^{-70}}{1+e^{-70}} + \frac{e^{-80}}{1+e^{-80}} + \dots} \approx -9$$

$$\mu_2 \approx \frac{1 \cdot (-9) + \dots + 1 \cdot 9}{15} = \frac{35}{15} = \frac{7}{3} = 2.(3)$$

Observație (1): Folosind o implementare în Matlab a acestui algoritm EM/GMM, valorile numerice pe care le-am obținut pentru această iterație sunt:

x_i	-9	-8	-7	-6	-5
$E[Z_{i1}]$	$8.75 \cdot 10^{-27}$	$3.97 \cdot 10^{-31}$	$1.8 \cdot 10^{-35}$	$8.19 \cdot 10^{-40}$	$3.72 \cdot 10^{-44}$
$E[Z_{i2}]$	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1

x_i	5	6	7	8	9
$E[Z_{i1}]$	$1.38 \cdot 10^{-87}$	$6.28 \cdot 10^{-92}$	$2.85 \cdot 10^{-96}$	$1.29 \cdot 10^{-100}$	$5.87 \cdot 10^{-105}$
$E[Z_{i2}]$	≈ 1	≈ 1	≈ 1	≈ 1	≈ 1

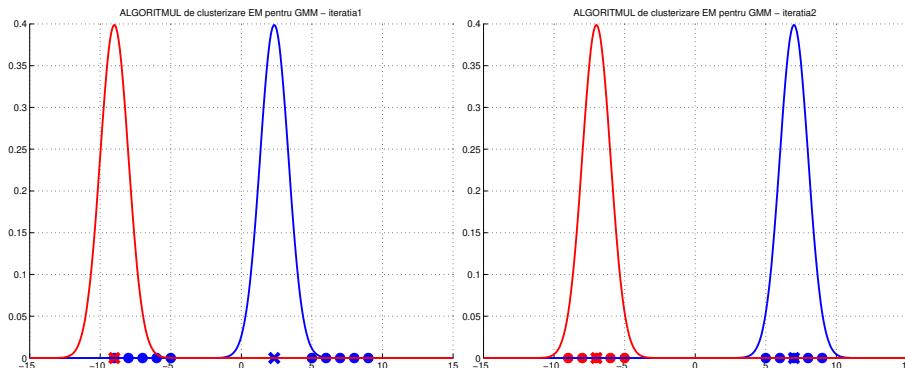
$$\mu_1 = -8.999955 \text{ și } \mu_2 = 2.333333$$

La următoarea iterație a acestui algoritm EM/GMM, componența clusterelor devine:

$$C_1 = \{-9, -8, -7, -6, -5\} \text{ și } C_2 = \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}, \text{ cu } \mu_1 \approx -7, \mu_2 \approx 7,$$

după cum se poate observa și din graficul de mai jos, partea dreaptă:⁴⁶⁹

⁴⁶⁹Graficul din partea stângă corespunde finalului primei iterării.



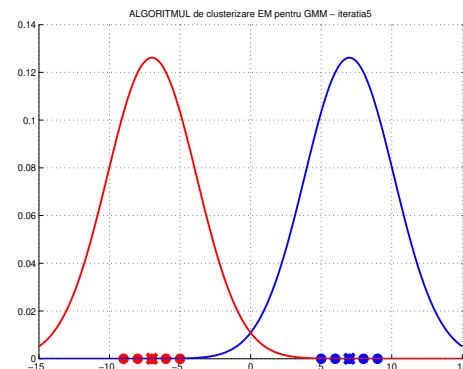
La următoarea iterație, algoritmul converge.⁴⁷⁰

Observație (2): Așadar, pentru cazul $\sigma^2 = 1$, soluția obținută de acest algoritm EM/GMM coincide cu cea a algoritmului K -means (deși numărul de iterării diferă). În cele ce urmează vom arăta că pentru alte valori ale lui σ^2 se pot obține rezultate semnificativ diferite pentru *valorile finale ale mediilor*:

Pentru $\sigma^2 = 20$, algoritmul EM/GMM converge după 10 iterării, obținându-se aceleasi clustere C_1 și C_2 , dar cu

$$\mu_1 = -6.657120 \text{ și } \mu_2 = 6.940543.$$

Se observă că aceste medii sunt foarte aproape de valorile 7 și -7 obținute de către algoritmul K -means, doar că au fost puțin „atrase” spre punctele din clusterul opus (și anume, mai mult μ_2 decât μ_1 , pentru că în clusterul din dreapta sunt de două ori mai multe puncte decât în clusterul din stânga).

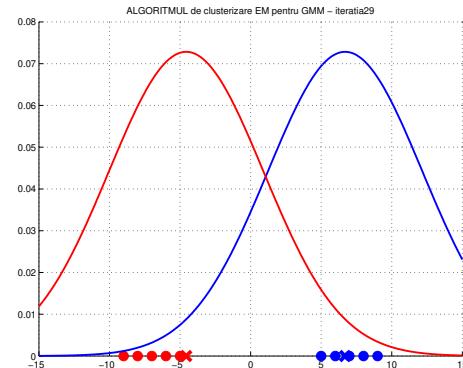


⁴⁷⁰ Criteriul de oprire pe care l-am folosit în implementarea noastră pentru algoritmul EM/GMM a fost următorul: la două iterării succesive, valorile mediilor nu se modifică cu mai mult de 10^{-5} .

Pentru $\sigma^2 = 30$, algoritmul EM/GMM converge după 29 interații, obținându-se aceeași clustere ca mai sus, dar cu

$$\mu_1 = -4.564930 \text{ și } \mu_2 = 6.698046.$$

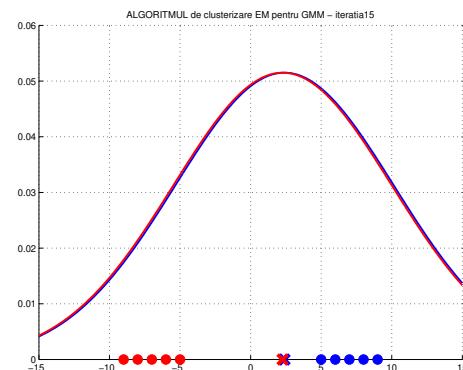
Media primei distribuții gausiene a migrat în mod considerabil spre cele 10 puncte din cel de-al doilea cluster, pe când media celei de-a doua distribuții gausiene este mult mai puțin „atras“ de cele 5 puncte din clusterul opus.



Pentru $\sigma^2 = 60$, algoritmul EM/GMM converge după 45 interații, obținându-se medii aproape identice:

$$\mu_1 = 2.333316 \text{ și } \mu_2 = 2.333351.$$

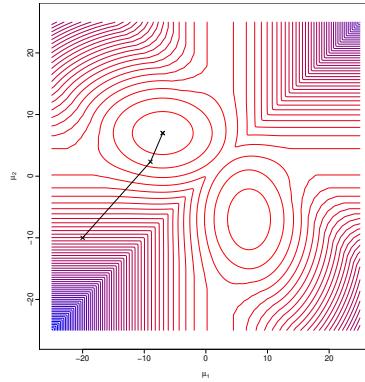
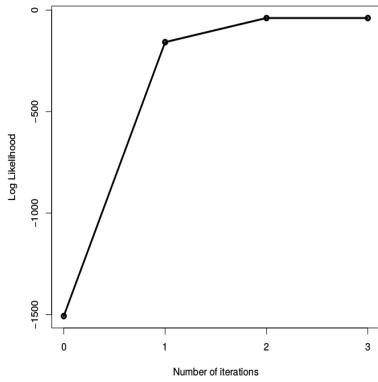
Cu toate acestea, probabilitățile de apartenență ale punctelor la cele două clustere sunt sensibil diferite. De exemplu, pentru punctul -9 aceste probabilități sunt $E[Z_{11}] = 0.5000022$ și $E[Z_{12}] = 0.4999978$. Acest fapt determină împărțirea punctelor în aceeași două clustere ca mai sus.



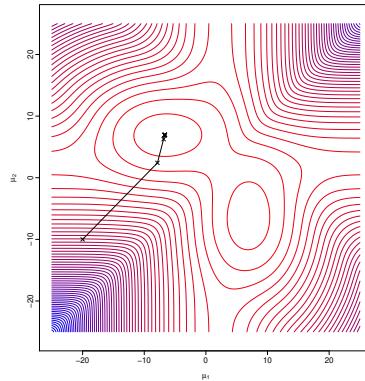
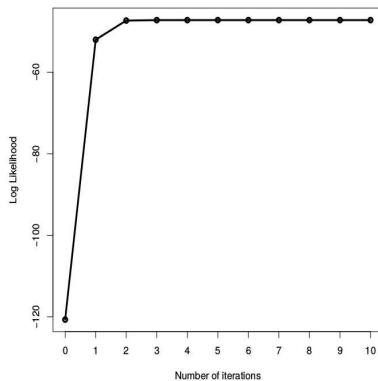
În imaginile următoare prezentăm,⁴⁷¹ pentru diferitele cazuri de mai sus (*i.* $\sigma^2 = 1$, *ii.* $\sigma^2 = 20$, *iii.* $\sigma^2 = 30$, *iv.* $\sigma^2 = 60$), valorile funcțiilor de log-verosimilitate [a datelor „observabile“] obținute de algoritmul EM la interații succesive (vedeți partea stângă) și respectiv graficele funcțiilor de log-verosimilitate reprezentate prin curbe de izocontur, pe care au fost puse în evidență valorile parametrilor μ_1 și μ_2 obținute de EM la interații succesive (partea dreaptă).⁴⁷²

⁴⁷¹Graficele au fost realizate de către studentul Sebastian Ciobanu în 2018.

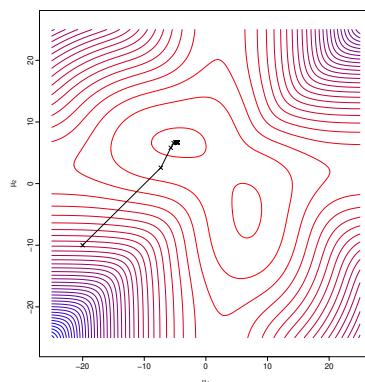
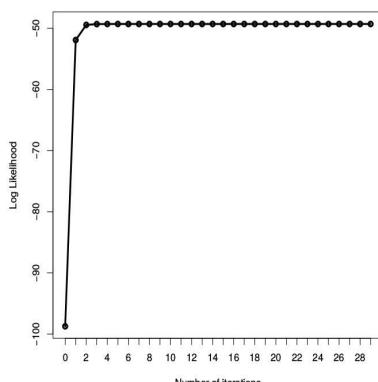
⁴⁷²Pentru un exemplu relativ simplu de calculare a funcției de log-verosimilitate a datelor observabile în cazul unei mixturi de două distribuții gausiene unidimensionale, vedeți ex. 16.c.



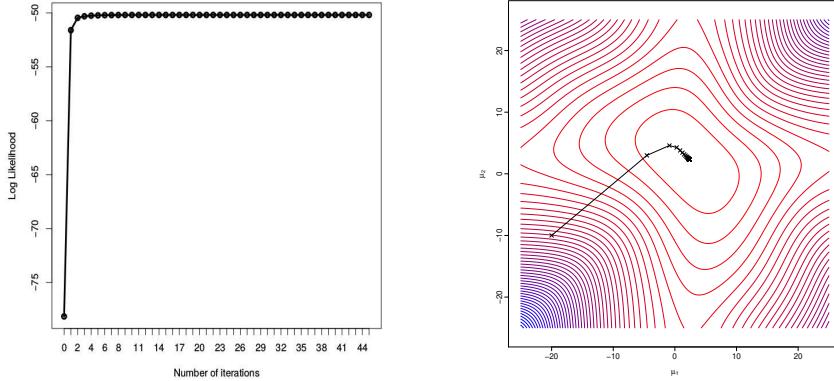
Cazul i. $\sigma^2 = 1.$



Cazul ii. $\sigma^2 = 20.$



Cazul iii. $\sigma^2 = 30.$



Cazul iv. $\sigma^2 = 60$.

Se observă, ca și la problema 16, că valoarile funcției de log-verosimilitate [a datelor „observabile“] obținute de algoritmul EM cresc, așa cum era de așteptat (vedeți problema 2 de la capitolul *Algoritmul EM*).

c. Pentru a realiza o iteratie a acestei variante algoritmului EM/GMM trebuie să calculăm „densitățile“ (adică, valorile funcțiilor densitate de probabilitate ale celor două gaussiene) $p(x_i | a)$ și $p(x_i | b)$ și probabilitățile a posteriori $a_i \stackrel{\text{not.}}{=} P(a | x_i)$ și $b_i \stackrel{\text{not.}}{=} P(b | x_i)$ corespunzătoare fiecărui punct x_i din setul de date considerat.

Spre exemplu, pentru punctul -9 , vom obține:

$$\begin{aligned} p(-9 | a) &= \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{(-9+20)^2}{2}\right) = \frac{1}{e^{60}\sqrt{2\pi e}} \approx 2 \cdot 10^{-27} \\ p(-9 | b) &= \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{(-9+10)^2}{2}\right) = \frac{1}{\sqrt{2\pi e}} \approx 0.24 \\ a_{-9} &\stackrel{T. Bayes}{=} \frac{2 \cdot 10^{-27} \cdot 0.5}{0.24 \cdot 0.5 + 2 \cdot 10^{-27} \cdot 0.5} = 8 \cdot 10^{-27} \\ b_{-9} &\stackrel{T. Bayes}{=} \frac{0.24 \cdot 0.5}{0.24 \cdot 0.5 + 2 \cdot 10^{-27} \cdot 0.5} \approx 1 \end{aligned}$$

În mod analog se fac calculele pentru toate punctele. Folosind o [altă] implementare în Matlab, am obținut:

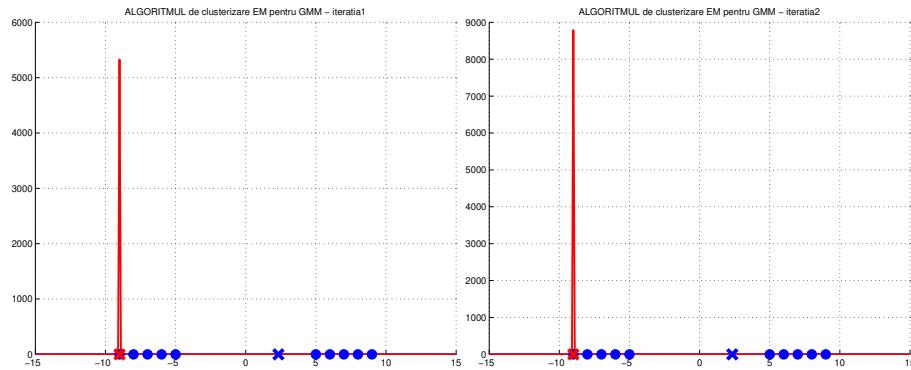
x_i	-9	-8	-7	-6	-5
$p(x_i a)$	$2 \cdot 10^{-27}$	$2 \cdot 10^{-32}$	$8 \cdot 10^{-38}$	10^{-43}	10^{-50}
$p(x_i b)$	0.24	0.05	0.004	0.0001	$1.4 \cdot 10^{-6}$

x_i	5	6	7	8	9
$p(x_i a)$	10^{-137}	10^{-148}	10^{-159}	10^{-171}	10^{-184}
$p(x_i b)$	10^{-50}	10^{-56}	10^{-64}	10^{-71}	10^{-79}

x_i	5	6	7	8	9
a_i	10^{-87}	10^{-92}	10^{-96}	10^{-100}	10^{-105}
b_i	≈ 1				

Apoi se calculează $\pi_a = \frac{1}{n} \sum_{i=1}^{15} a_i \simeq 5 \cdot 10^{-28}$ și $\pi_b = \frac{1}{n} \sum_{i=1}^{15} b_i \simeq 1$, deci toate punctele vor fi asignate clusterului b , iar noile medii și varianțe vor fi:⁴⁷³

$$\begin{aligned}\mu_a &\simeq -9, & \mu_b &= 2.3 \\ \sigma_a &= 0.000045, & \sigma_b &= 45.55\end{aligned}$$



La cea de-a două iterație, varianța componentei a se va apropiă foarte mult de 0, ceea ce va cauza un fapt suprinzător: cu toate că $\mu_a \simeq -9$, punctul $x_1 = -9$ va fi asociat celuilalt cluster, datorită varianței foarte mici a lui a , precum și datorită probabilității foarte mici de selecție a acestei gaussiene ($\pi_a \simeq 0$). Deci clusterele finale vor fi:

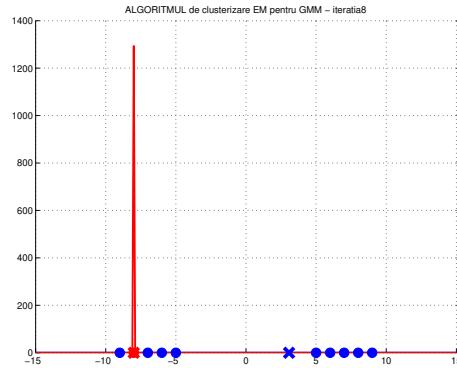
$$C_a = \emptyset \text{ și } C_b = \{-9, -8, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}$$

Observație (3): Soluția pe care tocmai am obținut-o cu această variantă a algoritmului EM/GMM nu este satisfăcătoare. Totuși, ca și la punctul b , vom arăta că rezultatul se poate schimba, în funcție de valorile inițiale considerate. Concret, dacă se variază initializarea varianțelor, se pot obține alte clustere, ca în exemplele următoare:

Pentru $\sigma_a^2 = \sigma_b^2 = 9$, algoritmul converge după 8 iterării, obținându-se:

$$\begin{aligned}\mu_a &\simeq -8, & \mu_b &= 3.071348 \\ \sigma_a &= 0.000308, & \sigma_b &= 40.638356 \\ \pi_a &= 0.066659, & \pi_b &= 0.933340\end{aligned}$$

Punctul $x_2 = -8$ va fi asociat gaussienei a , însă probabilitățile de apartenență la cele două clustere sunt: $a_2 = E[Z_{21}] = 0.999897b_2 = E[Z_{22}] = 0.000102$, însă restul punctelor vor fi asignate celuilalt cluster.



⁴⁷³La următoarele trei grafice nu am pus în evidență p.d.f.-urile corespunzătoare gaussienei b din cauza varianțelor prea mari. (În astfel de situații, curbele gaussiene nu se disting în mod semnificativ de axa Ox .)

Deci, la finalul execuției algoritmului vom avea:

$$C_a = \{-8\} \text{ și } C_b = \{-9, -7, -6, -5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}.$$

Este un rezultat interesant, care se datorează nerestricționării varianțelor celor două distribuții. Se pune astfel în evidență faptul că spre deosebire de cazul algoritmului K -means, *suprafețele de separare determinate de algoritmul EM nu sunt în mod neapărat liniare!*

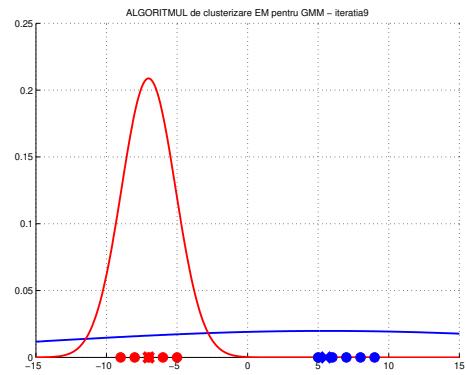
Observație (4): La ultimele două rezultate se observă că valoarea funcției densitate de probabilitate (p.d.f.) poate fi foarte mare dacă parametrul σ ia valori foarte mici. Într-o astfel de situație, valoarea funcției de log-verosimilitate a datelor complete poate deveni foarte mare.⁴⁷⁴ Astfel de situații (caracterizate de „puncte de singularitate“ pentru p.d.f.) nu sunt însă în general dezirabile pentru clusterizare.

Pentru $\sigma_a^2 = \sigma_b^2 = 20$, algoritmul EM/GMM converge după 9 iterări, obținându-se:

$$\begin{aligned}\mu_a &= -7.018194, & \mu_b &= 5.550311 \\ \sigma_a &= 1.910617, & \sigma_b &= 20.137057 \\ \pi_a &= 0.255955, & \pi_b &= 0.744044\end{aligned}$$

Clusterele obținute în acest caz sunt cele așteptate, și anume:

$$\begin{aligned}C_a &= \{-9, -8, -7, -6, -5\} \\ C_b &= \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}\end{aligned}$$



Pentru $\sigma_a^2 = 20$ și $\sigma_b^2 = 4$, acest algoritm EM/GMM converge după doar 5 iterări, obținându-se un rezultat similar cu cel al algoritmului de la punctul b (unde s-au folosit valori identice pentru σ^2):

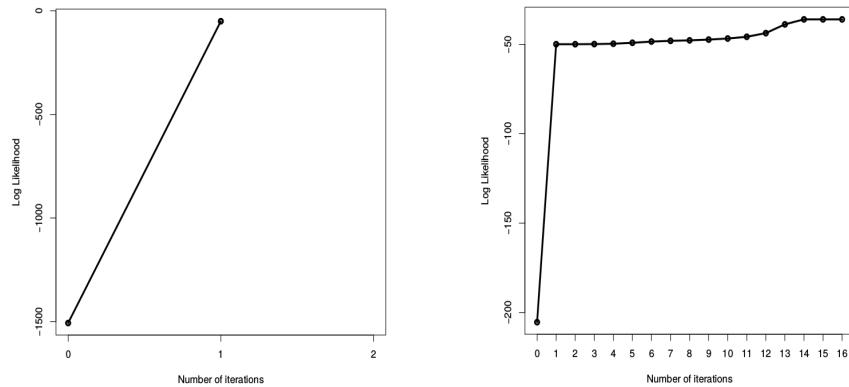
$$\begin{aligned}\mu_a &= 7, & \mu_b &= -7 \\ \sigma_a &= 2, & \sigma_b &= 2 \\ \pi_a &= 0.(6), & \pi_b &= 0.(3)\end{aligned}$$

Mai mult, clusterele și mediile / centripii sunt aceiași ca și în cazul algoritmului K -means, însă — interesant! — ele devin inversate: $C_a = \{5, 5, 6, 6, 7, 7, 8, 8, 9, 9\}$, $C_b = \{-9, -8, -7, -6, -5\}$

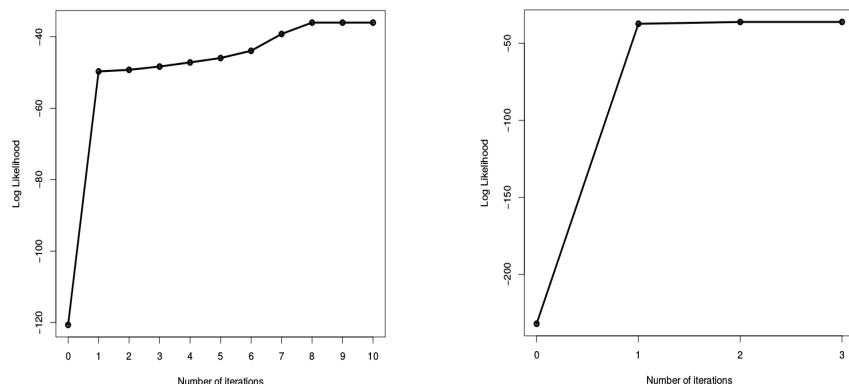
În imaginile următoare prezentăm,⁴⁷⁵ pentru diferitele cazuri de mai sus (i. $\sigma_a^2 = \sigma_b^2 = 1$, ii. $\sigma_a^2 = \sigma_b^2 = 9$, iii. $\sigma_a^2 = \sigma_b^2 = 20$, iv. $\sigma_a^2 = 20, \sigma_b^2 = 4$), valorile funcțiilor de log-verosimilitate obținute de algoritmul EM la iterări succesive.

⁴⁷⁴La distribuții discrete, funcția de log-verosimilitate a datelor complete (a cărei medie este maximizată de algoritmul EM) nu poate avea decât valori negative. (Justificarea este imediată, fiindcă funcția masă de probabilitate nu poate lua valori decât în intervalul $[0, 1]$.) La distribuții continue, funcția de log-verosimilitate a datelor complete poate lua și valori pozitive.

⁴⁷⁵Graficele au fost realizate de către studentul Sebastian Ciobanu în 2018.



Cazurile *i*. $\sigma_a^2 = \sigma_b^2 = 1$ (partea stângă) și, *ii*. $\sigma_a^2 = \sigma_b^2 = 9$ (partea dreaptă).



Cazurile *iii*. $\sigma_a^2 = \sigma_b^2 = 20$ (partea stângă) și, *iv*. $\sigma_a^2 = 20, \sigma_b^2 = 4$ (partea dreaptă).

Observație (5): T. Hastie, R. Tibshirani și J. Friedman recomandă în cartea *The Elements of Statistical Learning* (Springer, ed. a II-a, 2009) ca la pasul de inițializare în algoritmul EM/GMM să se folosească pentru varianțele empirice ale distribuțiilor gaussiene individuale valoarea varianței întregului set de date.

18.

(Algoritmul EM: estimarea tuturor parametrilor unei mixturi de două distribuții gaussiene unidimensionale)

*Liviu Ciortuz, 2012, după
■ MIT, ML course 6768, 2012 fall, Dahua Lin,
An Introduction to Expectation-Maximization*

Folosind algoritmul EM, rezolvați problema estimării parametrilor unei mixturi de două distribuții gaussiene unidimensionale în cazul cel mai general, adică lăsând liberi toți parametrii (μ – mediile, σ^2 – varianțele și π – probabilitățile a priori de selecție a celor două gaussiene).

Răspuns:

Vom urma etapele indicate de schema algoritmică EM, prezentată în secțiunea 6.12.2 din carte *Machine Learning* de Tom Mitchell (sau, echivalent problema 1, pag. 635 de la capitolul *Algoritmul EM* din această culegere).

Pasul E:

Vom considera instanțele $x_1, \dots, x_n \in \mathbb{R}$, vom nota parametrii $\mu = (\mu_1, \mu_2)$, $\sigma = (\sigma_1, \sigma_2)$ și $\pi = (\pi_1, \pi_2)$ și vom folosi variabilele aleatoare $Z_{i1}, Z_{i2} \in \{0, 1\}$ cu restricția $Z_{i1} + Z_{i2} = 1$ pentru orice $i \in \{1, \dots, n\}$. În aceste condiții, probabilitățile / mediile

$$p_{ij} \stackrel{\text{not.}}{=} P(Z_{ij} = 1 | X_i, \mu, \sigma, \pi) \stackrel{\text{calcul}}{=} E[Z_{ij} | X_i, \mu, \sigma, \pi], \text{ pentru } i = 1, \dots, n \text{ și } j = 1, 2,$$

se vor obține imediat folosind teorema lui Bayes:

$$\begin{aligned} p_{ij} &= \frac{P(X_i = x_i | Z_{ij} = 1, \mu, \sigma, \pi) \cdot P(Z_{ij} = 1 | \mu, \sigma, \pi)}{\sum_{j'} P(X_i = x_i | Z_{ij'} = 1, \mu, \sigma, \pi) \cdot P(Z_{ij'} = 1 | \mu, \sigma, \pi)} \\ &= \frac{\frac{1}{\sqrt{2\pi}\sigma_j} \cdot \exp\left(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}\right) \cdot \pi_j}{\frac{1}{\sqrt{2\pi}\sigma_1} \cdot \exp\left(-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}\right) \cdot \pi_1 + \frac{1}{\sqrt{2\pi}\sigma_2} \cdot \exp\left(-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}\right) \cdot \pi_2} \\ &= \frac{\frac{\pi_j}{\sigma_j} \cdot \exp\left(-\frac{(x_i - \mu_j)^2}{2(\sigma_j)^2}\right)}{\frac{\pi_1}{\sigma_1} \cdot \exp\left(-\frac{(x_i - \mu_1)^2}{2(\sigma_1)^2}\right) + \frac{\pi_2}{\sigma_2} \cdot \exp\left(-\frac{(x_i - \mu_2)^2}{2(\sigma_2)^2}\right)}. \end{aligned}$$

Verosimilitatea oricărei instanțe „complete” $y_i \stackrel{\text{not.}}{=} (x_i, z_{i1}, z_{i2})$ se va exprima sub forma:

$$\begin{aligned} P(X_i = x_i, Z_{i1} = z_{i1}, Z_{i2} = z_{i2} | \mu, \sigma, \pi) &= P(X_i = x_i | Z_{i1} = z_{i1}, Z_{i2} = z_{i2}, \mu_i, \sigma_i, \pi_i) \cdot P(Z_{i1} = z_{i1}, Z_{i2} = z_{i2} | \mu_i, \sigma_i, \pi_i) \\ &= \frac{1}{\sqrt{2\pi}\sigma_{j'}} \cdot \exp\left(-\frac{(x_i - \mu_{j'})^2}{2\sigma_{j'}^2}\right) \cdot \pi_{j'}, \text{ unde } z_{ij'} = 1 \text{ iar } z_{ij''} = 0 \text{ pentru } j'' \neq j'. \end{aligned}$$

Tinând cont de egalitatea $z_{i1} + z_{i2} = 1$ și de faptul că $z_{ij} \in \{0, 1\}$, putem continua dezvoltarea acestei expresii astfel:

$$\begin{aligned} P(X_i = x_i, Z_{i1} = z_{i1}, Z_{i2} = z_{i2} | \mu, \sigma, \pi) &= \frac{1}{\sqrt{2\pi}\sigma_1^{z_{i1}}\sigma_2^{z_{i2}}} \cdot \exp\left(-\frac{1}{2} \sum_{j \in \{1, 2\}} z_{ij} \frac{(x_i - \mu_j)^2}{\sigma_j^2}\right) \cdot \pi_1^{z_{i1}} \pi_2^{z_{i2}}. \end{aligned}$$

Observație (1): Acest mod de a scrie $P(X_i, Z_{i1}, Z_{i2} | \mu, \sigma, \pi)$, deși pare cumva artificial, este crucial pentru că include atât Z_{i1} cât și Z_{i2} . Ne pregătim astfel pentru aplicarea [ulterioră a] proprietății de liniaritate a mediei, la construirea funcției „auxiliare” Q .

Logaritărând ultima expresie obținută mai sus, vom obține *log-verosimilitatea* instanței „complete” $y_i = (x_i, z_{i1}, z_{i2})$:

$$\begin{aligned} \ln P(X_i = x_i, Z_{i1} = z_{i1}, Z_{i2} = z_{i2} | \mu, \sigma, \pi) &= -\frac{1}{2} \ln(2\pi) - \sum_{j=1}^2 z_{ij} \ln \sigma_j - \frac{1}{2} \sum_{j=1}^2 z_{ij} \frac{(x_i - \mu_j)^2}{\sigma_j^2} + \sum_{j=1}^2 z_{ij} \ln \pi_j. \end{aligned}$$

Notând $X = (X_1, \dots, X_n)$, $Z_1 = (Z_{11}, Z_{21}, \dots, Z_{n1})$ și $Z_2 = (Z_{12}, Z_{22}, \dots, Z_{n2})$, și ținând cont de independența statistică a datelor (x_i, z_{i1}, z_{i2}) cu $i = 1, \dots, n$, *log-verosimilitatea datelor complete* va fi data de expresia:

$$\begin{aligned} \ln P(X, Z_1, Z_2 | \mu, \sigma, \pi) &= \ln \prod_{i=1}^n P(X_i = x_i, Z_{i1}, Z_{i2} | \mu, \sigma, \pi) = \\ &\sum_{i=1}^n \ln P(X_i = x_i, Z_{i1}, Z_{i2} | \mu, \sigma, \pi) = \\ &- \frac{n}{2} \ln(2\pi) - \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \ln \sigma_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \frac{(x_i - \mu_j)^2}{\sigma_j^2} + \sum_{i=1}^n \sum_{j=1}^2 Z_{ij} \ln \pi_j. \end{aligned}$$

Mai departe, pentru a calcula *media log-verosimilității datelor complete*, vom ține cont de proprietatea de liniaritate a mediilor. Așadar, vom avea:

$$\begin{aligned} E[\ln P(X, Z_1, Z_2 | \mu, \sigma, \pi)] &= \\ &- \frac{n}{2} \ln(2\pi) - \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij}] \ln \sigma_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij}] \frac{(x_i - \mu_j)^2}{\sigma_j^2} + \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij}] \ln \pi_j. \end{aligned}$$

Acum vom pune în evidență faptul că media aceasta se calculează în funcție de probabilitatea condiționată $P_{Z|X, \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}}$, unde $\mu^{(t)}, \sigma^{(t)}, \pi^{(t)}$ notează valorile parametrilor la initializare ($t = 0$) sau, mai general, la iterația precedentă:

$$\begin{aligned} Q(\mu, \sigma, \pi | \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) &\stackrel{\text{not.}}{=} E_{Z|X, \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}} [\ln P(X, Z_1, Z_2 | \mu, \sigma, \pi)] = \\ &- \frac{n}{2} \ln(2\pi) - \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij} | X_i, \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}] \ln \sigma_j \\ &- \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij} | X_i, \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}] \frac{(x_i - \mu_j)^2}{\sigma_j^2} \\ &+ \sum_{i=1}^n \sum_{j=1}^2 E[Z_{ij} | X_i, \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}] \ln \pi_j. \end{aligned}$$

Extinzând în mod natural notația $p_{ij} = E[Z_{ij} | X_i, \mu, \sigma, \pi]$, care a fost introdusă mai sus la $p_{ij}^{(t)} = E[Z_{ij} | X_i, \mu^{(t-1)}, \sigma^{(t-1)}, \pi^{(t-1)}]$, vom rescrie în mod simplificat expresia așa-numitei *funcții auxiliare* Q :

$$\begin{aligned} Q(\mu, \sigma, \pi | \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) &= \\ &- \frac{n}{2} \ln(2\pi) - \sum_{i=1}^n \sum_{j=1}^2 p_{ij}^{(t)} \ln \sigma_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^2 p_{ij}^{(t)} \frac{(x_i - \mu_j)^2}{\sigma_j^2} + \sum_{i=1}^n \sum_{j=1}^2 p_{ij}^{(t)} \ln \pi_j. \quad (162) \end{aligned}$$

Pasul M:

Vom căuta optimul funcției $Q(\mu, \sigma, \pi | \mu^{(t)}, \sigma^{(t)}, \pi^{(t)})$ cu ajutorul derivatelor parțiale de ordinul întâi.

Pentru calculul derivatelor parțiale în raport cu π_j , vom ține cont de restricția $\pi_1 + \pi_2 = 1$:

$$\begin{aligned} Q(\mu, \sigma, \pi \mid \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) &= -\frac{n}{2} \ln 2\pi - \sum_{i=1}^n \sum_{j=1}^2 p_{ij}^{(t)} \ln \sigma_j \\ &\quad - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^2 p_{ij}^{(t)} \frac{(x_i - \mu_j)^2}{\sigma_j^2} + \sum_{i=1}^n (p_{i1}^{(t)} \ln \pi_1 + p_{i2}^{(t)} \ln (1 - \pi_1)). \end{aligned}$$

Așadar,

$$\begin{aligned} \frac{\partial}{\partial \pi_1} Q(\mu, \sigma, \pi \mid \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) = 0 &\Leftrightarrow \frac{1}{\pi_1} \sum_{i=1}^n p_{i1}^{(t)} = \frac{1}{1 - \pi_1} \sum_{i=1}^n p_{i2}^{(t)} \Leftrightarrow \\ \sum_{i=1}^n p_{i1}^{(t)} &= \pi_1 \left(\sum_{i=1}^n p_{i1}^{(t)} + \sum_{i=1}^n p_{i2}^{(t)} \right) \Leftrightarrow \sum_{i=1}^n p_{i1}^{(t)} = \pi_1 \underbrace{\sum_{i=1}^n (p_{i1}^{(t)} + p_{i2}^{(t)})}_1 \Leftrightarrow \sum_{i=1}^n p_{i1}^{(t)} = n\pi_1, \end{aligned}$$

de unde rezultă $\pi_1^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n p_{i1}^{(t)}$. Apoi, ținând cont de relațiile $p_{i1}^{(t)} + p_{i2}^{(t)} = 1$ pentru $i = 1, \dots, n$ și $\pi_1^{(t+1)} + \pi_2^{(t+1)} = 1$, vom obține imediat $\pi_2^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n p_{i2}^{(t)}$.

Mai departe, vom deriva funcția Q în raport cu μ_1 :

$$\frac{\partial}{\partial \mu_1} Q(\mu, \sigma, \pi \mid \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) = 0 \Leftrightarrow \frac{1}{\sigma_1^2} \sum_{i=1}^n p_{i1}^{(t)} (x_i - \mu_1) = 0 \Leftrightarrow \sum_{i=1}^n p_{i1}^{(t)} (x_i - \mu_1) = 0,$$

de unde avem $\mu_1^{(t+1)} \leftarrow \frac{\sum_{i=1}^n p_{i1}^{(t)} x_i}{\sum_{i=1}^n p_{i1}^{(t)}}$. Similar, vom obține $\mu_2^{(t+1)} \leftarrow \frac{\sum_{i=1}^n p_{i2}^{(t)} x_i}{\sum_{i=1}^n p_{i2}^{(t)}}$.

În final, vom deriva funcția Q în raport cu σ_1 :

$$\frac{\partial}{\partial \sigma_1} Q(\mu, \sigma, \pi \mid \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) = 0 \Leftrightarrow -\frac{1}{\sigma_1} \sum_{i=1}^n p_{i1}^{(t)} + \frac{1}{\sigma_1^3} \sum_{i=1}^n p_{i1}^{(t)} (x_i - \mu_1)^2 = 0,$$

de unde rezultă că $(\sigma_1^{(t+1)})^2 \leftarrow \frac{\sum_{i=1}^n p_{i1}^{(t)} (x_i - \mu_1^{(t+1)})^2}{\sum_{i=1}^n p_{i1}^{(t)}}$.

Similar, vom obține $(\sigma_2^{(t+1)})^2 \leftarrow \frac{\sum_{i=1}^n p_{i2}^{(t)} (x_i - \mu_2^{(t+1)})^2}{\sum_{i=1}^n p_{i2}^{(t)}}$.

Observație (2):

Cititorul atent va sesiza că în final, la calculul expresiei lui $\sigma_1^{(t+1)}$, s-a înlocuit μ_1 cu $\mu_1^{(t+1)}$. Explicația, în manieră *intuitivă*, este următoarea: per ansamblu, aici calculăm maximul funcției Q nu doar în raport cu σ_1 ci și cu ceilalți parametri, deci în particular și cu μ_1 , a cărui valoare optimă la iterată $t + 1$ este $\mu_1^{(t+1)}$. La calculul celorlalte valori optime ale parametrilor ($\pi^{(t+1)}$ și $\mu^{(t+1)}$) nu a fost necesară o astfel de coroborare a soluțiilor. Cazul lui $\sigma^{(t+1)}$ este însă ușor diferit față de cazul celorlalți parametri. În mod *riguros*, se poate demonstra că rădăcinile derivatelor parțiale de ordinul întâi ale funcției Q constituie într-adevăr valorile [parametrilor] pentru care această funcție își atinge *maximul*.⁴⁷⁶

⁴⁷⁶Ideeza demonstrației este similară cu cea de la problema 48.a de la capitolul de *Fundamente*. În anume, după fiecare etapă de calcul al unei derivate parțiale de la pasul M se pot scrie relații

Observație (3):

Este foarte util să comparăm *i.* forma relațiilor de actualizare pentru parametrii μ_j și σ_j^2 de la pasul M al algoritmului EM pentru modelarea unei mixturi de distribuții gaussiene unidimensionale cu *ii.* formulele obținute la estimarea în sens MLE a parametrilor μ și σ^2 ai unei distribuții gaussiene tot unidimensionale (vedeți problemele 45.a și 46.a de la capitolul de *Fundamente*). Se poate observa ușor că aceste două seturi de formule sunt foarte asemănătoare! Așa cum era așteptat, deosebirea dintre ele este dată de prezența probabilităților p_{ij} (calculate la pasul E) din algoritmul EM. Așadar, formulele *i.* sunt versiuni *poderate* ale formulelor *ii.* (deci reprezintă *medii ponderate*, respectiv *varianțe ponderate*).

Observație (4):

O comparație similară se poate face și între formula de actualizare a mediilor μ_j de la pasul M al algoritmului EM și formula pentru recalcularea centroizilor μ_j la pasul al doilea al algoritmului *K-means* (a se vedea *în special* varianta lui *K-means* de la problema 42, unde se folosesc variabilele-indicator γ_{ij}).

În final, putem face următoarea *generalizare*, pentru mixturi de $K > 2$ gaussiene unidimensionale:

$$\sum_{j=1}^K \pi_j N(x; \mu_j, \sigma_j^2).$$

Este de observat faptul că, în acest caz, în mixtură, distribuția Bernoulli este înlocuită cu o distribuție categorială, ale cărei valori $(1, \dots, K)$ sunt luate cu probabilitățile π_1, \dots, π_K . Pentru deducerea algoritmului EM în acest caz, singura schimbare de fond necesară în raport cu demonstrația de mai sus se referă la actualizarea parametrilor π_j :

Întrucât $\pi_1 + \dots + \pi_K = 1$, va trebui să rezolvăm următoarea *problemă de optimizare cu restricții*:

$$\max_{\pi, \mu, \sigma} Q(\pi, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)})$$

$$\text{cu restricția } \sum_{j=1}^K \pi_j = 1,$$

în care expresia lui Q provine din relația (162).⁴⁷⁷ Folosind *metoda multiplicatorilor Lagrange*, vom introduce variabila $\lambda \in \mathbb{R}$, iar problema de mai sus va deveni:

$$\max_{\pi, \mu, \sigma} \left(Q(\pi, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)}) + \lambda \left(1 - \sum_{j=1}^K \pi_j \right) \right).$$

de forma

$$\begin{aligned} Q(\pi_1, \pi_2, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)}) &\leq Q(\pi_1^{(t+1)}, \pi_2, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)}), \quad \forall \pi_1, \pi_2, \mu, \sigma \\ Q(\pi_1^{(t+1)}, \pi_2, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)}) &\leq Q(\pi_1^{(t+1)}, \pi_2^{(t+1)}, \mu, \sigma | \pi^{(t)}, \mu^{(t)}, \sigma^{(t)}), \quad \forall \pi_2, \mu, \sigma \end{aligned}$$

s.a.m.d.

⁴⁷⁷ *Observație:* Nu mai adăugăm la această problemă de optimizare și restricțiile $\pi_j \geq 0$ pentru $j = 1, \dots, K$, întrucât vom arăta la final că soluțiile obținute satisfac în mod implicit aceste restricții.

Derivând funcția obiectiv a acestei probleme în raport cu variabila π_j , unde $j \in \{1, \dots, K\}$, obținem:

$$\frac{\partial}{\partial \pi_j} Q(\mu, \sigma, \pi \mid \mu^{(t)}, \sigma^{(t)}, \pi^{(t)}) = 0 \Leftrightarrow \sum_{i=1}^n p_{ij}^{(t)} \frac{1}{\pi_j} = \lambda \Leftrightarrow \pi_j^{(t+1)} = \frac{1}{\lambda} \sum_{i=1}^n p_{ij}^{(t)}.$$

Întrucât $\sum_{j=1}^K \pi_j^{(t+1)} = 1$, urmează că

$$\lambda = \sum_{j=1}^K \sum_{i=1}^n p_{ij}^{(t)} = \sum_{i=1}^n \underbrace{\sum_{j=1}^K p_{ij}^{(t)}}_1 = n.$$

Așadar,

$$\pi_j^{(t+1)} \leftarrow \frac{1}{n} \sum_{i=1}^n p_{ij}^{(t)}.$$

Se observă că $\pi_j^{(t+1)} \geq 0$ pentru orice j , întrucât termenii $p_{ij}^{(t)}$ reprezintă niște probabilități (vedeți pasul E).

In concluzie, corpul iterativ al algoritmului EM pentru estimarea [tuturor] parametrilor unei mixturi de K distribuții gaussiene este:

Pasul E: Pentru $i = 1, \dots, n$ și $j = 1, \dots, K$, calculează

$$p_{ij}^{(t)} \stackrel{\text{not.}}{=} P(z_{ij} = 1 \mid x_i; \mu^{(t)}, (\sigma^2)^{(t)}, \pi^{(t)}) = \frac{\mathcal{N}(x_i \mid \mu_j^{(t)}, (\sigma_j^2)^{(t)}) \cdot \pi_j^{(t)}}{\sum_{l=1}^K \mathcal{N}(x_i \mid \mu_l^{(t)}, (\sigma_l^2)^{(t)}) \cdot \pi_l^{(t)}}$$

unde $\mathcal{N}(x_i \mid \mu_j, \sigma_j^2) \stackrel{\text{def.}}{=} \frac{1}{\sqrt{2\pi}\sigma_j} \cdot \exp\left(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}\right)$.

Pasul M: Pentru $j = 1, \dots, K$, calculează

$$\begin{aligned} \pi_j^{(t+1)} &\leftarrow \frac{1}{n} \sum_{i=1}^n p_{ij}^{(t)} \\ \mu_j^{(t+1)} &\leftarrow \frac{\sum_{i=1}^n p_{ij}^{(t)} x_i}{\sum_{i=1}^n p_{ij}^{(t)}} \\ (\sigma_j^{(t+1)})^2 &\leftarrow \frac{\sum_{i=1}^n p_{ij}^{(t)} (x_i - \mu_j^{(t+1)})^2}{\sum_{i=1}^n p_{ij}^{(t)}} \end{aligned}$$

Evident, aceste formule le generalizează pe cele care au fost deduse anterior, când s-a considerat $K = 2$.

Observație (5): Se va putea vedea la problema 20 că formulele care au fost obținute aici pot fi derivate și pe altă cale, și anume particularizând numărul de dimensiuni (d) la valoarea 1 în formulele care alcătuiesc corpul iterativ al algoritmului EM pentru mixturi de distribuții gaussiene multidimensionale (vedeți pag. 577).

19.

(Algoritmul EM:
chestiuni calitative / metodologice privind aplicarea
în cazul unei mixturi de două distribuții gaussiene unidimensionale
presupunând probabilitățile de selecție egale și fixate)
■ CMU, 2007 spring, Eric Xing, final exam, pr. 1.8

A fost odată, cu mult timp în urmă, un sat care era situat într-o regiune cu sute de lacuri. În acele lacuri trăiau două specii de pești, însă în fiecare lac, nu erau decât pești dintr-o singură specie. (Peștii din lacuri diferite puteau fi de specii diferite.) Peștii din aceste două specii arătau identic, miroseau identic atunci când erau gătiți și aveau exact același gust delicios, însă una dintre specii era otrăvitoare și oricine mâncă pești din specia respectivă murea. Singura diferență observabilă la aceste specii de pești consta în efectul asupra nivelului pH-ului (aciditatea) apei din lacul în care trăiau. Lacurile cu pești neotrăvitori aveau pH-ul distribuit conform unei distribuții gaussiene de medie (μ_{sigur}) și varianță (σ_{sigur}^2) necunoscute, iar pH-ul din lacurile cu pești otrăvitori era distribuit conform unei alte distribuții gaussiene de medie (μ_{mortal}) și varianță (σ_{mortal}^2) de asemenea necunoscute. (Peștii otrăvitori cauzau o aciditate care era în general mai mare / ridicată decât în cazul celorlalți pești.)

Așa cum era firesc, pentru a ieși din încurcătură — adică pentru a determina caracterul otrăvitor ori neotrăvitor al peștilor din fiecare lac (sau dintr-un lac oarecare, din care încă nu s-au consumat pești) în funcție de nivelul pH-ului apei —, sătenii au apelat la învățarea automată. Cu toate acestea, au avut loc dezbateri aprinse cu privire la modalitatea corectă de aplicare a algoritmului Expectation-Maximization la problema lor.

Pentru fiecare dintre modalitățile prezentate mai jos, indicați dacă ea constituie o aplicare corectă a algoritmului EM și dacă va conduce la o estimare rezonabilă pentru parametrii μ și σ^2 ai fiecărei clase.

- Se ghicesc valorile inițiale pentru parametrii μ și σ^2 corespunzători fiecărei specii. (1) Pentru fiecare lac, se determină — folosind teorema lui Bayes și distribuțiile gaussiene de parametri μ și σ^2 — care este cea mai probabilă specie de pești asociată lacului respectiv. (2) Se recalculează valorile pentru μ și σ^2 utilizând metoda estimării de verosimilitate maximă. Se repetă iterativ pașii (1) și (2) până se ajunge la convergență.
- Pentru fiecare lac, se ghicește probabilitatea (inițială) că lacul respectiv ar fi populat cu pești neotrăvitori. (1) Folosind aceste probabilități, se estimează în sensul verosimilității maxime valorile μ și σ corespunzătoare fiecărei clase. (2) Utilizând aceste estimări pentru μ și σ , se recalculează probabilitățile de ‘siguranță’ pentru fiecare lac. Se repetă iterativ pașii (1) și (2) până se ajunge la convergență.
- Se calculează media și varianța nivelului de pH pentru toată mulțimea lacurilor. Se folosesc aceste valori pentru a inițializa parametrii μ și σ^2 corespunzători fiecărei specii de pești. (1) Folosind aceste valori pentru μ și σ^2 , se calculează pentru fiecare lac probabilitatea să conțină pești otrăvitori. (2) Se găsesc valorile de verosimilitate maximă pentru μ și σ^2 . Se repetă iterativ pașii (1) și (2) până se ajunge la convergență.

Răspuns:

Din enunțul problemei, în mod implicit putem presupune că $P(sigur) = P(mortal) = 1/2$. Varianta algoritmui EM/GMM utilizată în acest caz diferă doar ușor de varianta (mai generală) formulată la problema 18: probabilitățile π_{sigur} și π_{mortal} se consideră totdeauna $1/2$ (deci nu se recalculează la pasul M).

a. Această modalitate de aplicare a algoritmului este cea clasică, deci corectă. Pasul (1) face în mod implicit calculul mediilor $E[z_{i,sigur}] = p(z_{i,sigur} = 1|x_i; \mu, \sigma^2)$ și $E[z_{i,mortal}] = p(z_{i,mortal} = 1|x_i; \mu, \sigma^2)$, unde notațiile folosite sunt cele clasice, iar pasul (2) reprezintă maximizarea funcției auxiliare, în spătă recalcularea parametrilor μ și σ^2 .

b. Această metodă de aplicare a algoritmului EM nu este la fel de evidentă ca metoda la punctul precedent, dar vom arăta că este totuși corectă.

O primă diferență față de metoda anterioară constă în faptul că în loc să se ghicească valorile inițiale pentru parametrii μ și σ^2 corespunzători fiecărei clase / specii, se ghicesc probabilitățile inițiale pentru ‘siguranță’ fiecărui lac, adică $p(z_{i,sigur} = 1|x_i; \mu, \sigma^2)$ (chiar dacă numărul acestor probabilități este în mod normal mult mai mare decât al parametrilor μ și σ^2). Știm că $z_{i,sigur} + z_{i,mortal} = 1$, aşadar ghicirea probabilității $p(z_{i,sigur} = 1|x_i; \mu, \sigma^2)$ antrenează în mod implicit setarea probabilității $p(z_{i,mortal} = 1|x_i; \mu, \sigma^2)$. Prin urmare, la acest pas de inițializare se încearcă să se „ghicească” valorile care ar trebui calculate la prima execuție a pasului E din varianta clasică a algoritmului EM, dacă s-ar dispune de valori inițiale pentru parametrii μ și σ^2 .

A doua diferență față de metoda clasică constă în inversarea pașilor din corpul iterativ al algoritmului EM: pasul (1) reprezintă maximizarea, iar pasul (2) estimarea.

Se observă că aceste două „diferențe” se compensează reciproc: valorile pentru parametrii μ și σ^2 care sunt calculate / estimate la prima execuție a pasului de maximizare (1) — chiar dacă probabilitățile „ghicite” inițial, pe care se bazează aceste calcule, sunt imperfecte sau chiar grav eronate — vor juca rolul valorilor inițiale din varianta (clasică) de la punctul a. După acest prim pas, prezenta variantă a algoritmului EM se comportă aidoma cu cea de la punctul a.

c. Această metodă nu este corectă / convenabilă deoarece utilizează valori inițiale egale pentru media (și respectiv pentru varianță) nivelului de pH pentru ambele clase de lacuri. Din acest motiv algoritmul rămâne „blocat” la valorile calculate pentru μ și σ^2 la prima iterație. Iată explicația:

La pasul (1) — estimare, se calculează:

$$P(\text{mortal} | x_i; \mu, \sigma^2) = \frac{P(x_i | \text{mortal}, \mu, \sigma^2) \cdot P(\text{mortal})}{P(x_i | \text{mortal}, \mu, \sigma^2) \cdot P(\text{mortal}) + P(x_i | \text{sigur}, \mu, \sigma^2) \cdot P(\text{sigur})}$$

Am văzut că din enunțul problemei putem presupune că $P(\text{sigur}) = P(\text{mortal}) = 1/2$. Întrucât s-a considerat $\mu_{\text{sigur}} = \mu_{\text{mortal}}$ și $\sigma_{\text{sigur}} = \sigma_{\text{mortal}}$, rezultă că $P(x | \text{sigur}, \mu, \sigma^2) = P(x | \text{mortal}, \mu, \sigma^2)$. Deci, conform formulei lui Bayes, vom obține $P(\text{sigur} | x; \mu, \sigma^2) = 1/2$. Prin urmare, și $P(\text{mortal} | x; \mu, \sigma^2) = 1/2$.

La pasul (2) — maximizare, valorile obținute pentru parametrii μ și σ^2 vor fi:⁴⁷⁸ $\mu_{\text{sigur}} = \mu_{\text{mortal}} = \frac{1}{n} \sum_{i=1}^n x_i$, iar $\sigma_{\text{sigur}}^2 = \sigma_{\text{mortal}}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_{\text{sigur}})^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_{\text{mortal}})^2$, unde x_i sunt nivelurile de pH ale lacurilor (și anume, cele pentru care s-a măsurat acest nivel).

Se observă ușor că valorile obținute pentru $P(\text{sigur} | x, \mu, \sigma^2)$, $P(\text{mortal} | x, \mu, \sigma^2)$, μ_{sigur} , μ_{mortal} , σ_{sigur}^2 și σ_{mortal}^2 vor rămâne neschimbate după execuția primei iterații, deci practic algoritmul „staționează” / „buclează”.⁴⁷⁹

⁴⁷⁸ A se vedea regulile de actualizare de la problema 18.

⁴⁷⁹ Este foarte puțin probabil ca aceste valori pentru π , μ și σ^2 să reprezinte coordonatele unui punct de maxim local pentru funcția de verosimilitate a datelor observabile.

20.

(Algoritmul EM pentru mixturi de distribuții gaussiene multidimensionale; cazul general)

*prelucrare de Liviu Ciortuz, după
■ Andrew Ng, Stanford University, ML course, 2009 fall,
lecture notes, parts VIII and IX,
and CMU, 2010 fall, Aarti Singh, HW4, pr. 1.1*

Presupunem, ca de obicei în învățarea automată, că avem un set de date $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$. Ne vom situa în paradigma învățării nesupervizate, deci vom considera că aceste puncte nu au nicio etichetă atașată.

Dorim să modelăm aceste date prin specificarea [funcției de densitate a] unei distribuții probabiliste comune

$$p(x_i, z_i) = p(x_i|z_i) p(z_i), \text{ pentru } i = 1, \dots, n,$$

cu $z_i \sim \text{Categorial}(\pi)$ și $\pi \stackrel{\text{not.}}{=} (\pi_1, \dots, \pi_K)$.

Așadar, $p(x_i) = \sum_{z_i} p(x_i, z_i) = \sum_{z_i} p(x_i|z_i) p(z_i)$, formulă cu care suntem obișnuiți. K este numărul de valori pe care le pot lua variabilele z_i . Pentru $j = 1, \dots, K$, probabilitatea a priori $p(z_i = j)$ va fi desemnată prin parametrul π_j . Vom presupune că $(x_i|z_i = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$. Așadar,

$$p(x_i|z_i = j; \mu_j, \Sigma_j) \stackrel{\text{def.}}{=} \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_j)^\top \Sigma_j^{-1} (x - \mu_j)\right),$$

unde $\mu \in \mathbb{R}^d$, iar Σ_j este matrice de dimensiune $d \times d$, simetrică și pozitiv definită (ultima condiție este suficientă pentru a asigura inversabilitatea).⁴⁸⁰ ⁴⁸¹ Vectorii din \mathbb{R}^d sunt considerați vectori-coloană în contextul acestei probleme.

Prin urmare, modelul nostru postulează că

- fiecare instanță x_i a fost generată alegând în mod aleatoriu valorile variabilelor z_i din multimea $\{1, \dots, K\}$,
- iar apoi x_i a fost generat de una dintre cele K distribuții gaussiene, mai precis cea desemnată de z_i .

Acest model se numește *modelul mixturii de [distribuții] gaussiene*. Vom considera că z_i sunt variabile aleatoare *latente* (se mai spune că sunt variabile *ascunse*, sau *neobservabile*). Datorită acestui fapt, problema estimării în sensul MLE a parametrilor π , μ și Σ devine dificilă. (Am notat $\mu = (\mu_1, \dots, \mu_K)$ și $\Sigma = (\Sigma_1, \dots, \Sigma_K)$.)

Observație (1): Evident, dacă am cunoaște valorile variabilelor z_i , problema estimării parametrilor modelului nostru (adică π , μ și Σ) ar fi ușor de rezolvat.⁴⁸² Concret, am putea scrie *verosimilitatea* datelor noastre ca

$$l(\pi, \mu, \Sigma) = \sum_{i=1}^n [\ln p(x_i|z_i; \mu, \Sigma) + \ln p(z_i; \pi)].$$

⁴⁸⁰Vedeți problema 20 de la capitolul de *Fundamente*.

⁴⁸¹În penultima linie a egalității de mai sus, prin simbolul \exp am desemnat ca de obicei funcția exponentială cu baza e . (Așadar, $\exp(y) \stackrel{\text{not.}}{=} e^y$ pentru orice $y \in \mathbb{R}$.) Simbolul \top din ultima relație reprezintă operația de transpunere de matrice / vectori.

⁴⁸²Este o situație similară cu — și doar ușor mai elaborată decât — cea din problema 48 (punctul a) din capitolul *Fundamente*.

Făcând calculele, vom obține următoarele estimări MLE ale parametrilor π , μ și Σ :⁴⁸³

$$\pi_j = \frac{1}{n} \sum_{i=1}^n 1_{\{z_i=j\}}, \quad (163)$$

$$\mu_j = \frac{\sum_{i=1}^n 1_{\{z_i=j\}} x_i}{\sum_{i=1}^n 1_{\{z_i=j\}}}, \quad (164)$$

$$\Sigma_j = \frac{\sum_{i=1}^n 1_{\{z_i=j\}} (x_i - \mu_j)(x_i - \mu_j)^\top}{\sum_{i=1}^n 1_{\{z_i=j\}}}, \quad (165)$$

unde $1_{\{z_i=j\}}$ sunt „funcții indicator“, care desemnează gaussiana (j) care a generat o instanță oarecare (x_i) din setul de date pe care îl avem la dispoziție. Însă, atunci când valorile variaibilelor z_i sunt necunoscute, nu este posibil să obținem estimările în sens MLE ale parametrilor π , μ și Σ prin calcularea analitică a rădăcinilor derivatelor parțiale de ordinul întâi ale funcției de verosimilitate $l(\pi, \mu, \Sigma)$.

În astfel de cazuri, *algoritmul EM* constituie o metodă convenabilă pentru estimarea parametrilor în sensul verosimilității maxime. *Strategia* sa constă în a construi în mod iterativ o *limită inferioară* pe funcția de *log-verosimilitate* a datelor observabile x_1, \dots, x_n (pasul E), pentru ca apoi să calculeze *maximul* acelei limite inferioare (pasul M).

Folosind schema algoritmică EM,⁴⁸⁴ demonstrați că *regulile de actualizare* pentru parametrii π , μ și Σ în cazul unei mixturi de distribuții gaussiene multidimensionale sunt următoarele:⁴⁸⁵

Pasul E:

$$w_{ij} \stackrel{\text{not.}}{=} p(z_i = j | x_i; \pi', \mu', \Sigma') = \frac{p(x_i | z_i = j; \mu', \Sigma') p(z_i = j; \pi')}{\sum_{l=1}^K p(x_i | z_i = l; \mu', \Sigma') p(z_i = l; \pi')}$$

Pasul M:

$$\pi_j = \frac{1}{n} \sum_{i=1}^n w_{ij}, \quad (166)$$

$$\mu_j = \frac{\sum_{i=1}^n w_{ij} x_i}{\sum_{i=1}^n w_{ij}}, \quad (167)$$

$$\Sigma_j = \frac{\sum_{i=1}^n w_{ij} (x_i - \mu_j)(x_i - \mu_j)^\top}{\sum_{i=1}^n w_{ij}}. \quad (168)$$

În aceste formule, π' , μ' și Σ' reprezintă valorile parametrilor modelului nostru la iteratărea precedentă a algoritmului EM (respectiv, valorile atribuite inițial parametrilor, pentru prima iteratăre).⁴⁸⁶

Sugestie: Întrucât se lucrează cu instanțe din \mathbb{R}^d , vă recomandăm să folosiți derive de [parțiale] vectoriale. Pentru acest tip de derive, variabila în raport cu care se derivează

⁴⁸³Vedeți similaritatea dintre formulele (164) și (272), respectiv (165) și (276).

⁴⁸⁴Vedeți cartea *Machine Learning*, a lui Tom Mitchell, 1997, pag. 194-195 sau problema 1 (pag. 635) de la capitolul *Algoritmul EM* din această culegere.

⁴⁸⁵Este util de observat (și de analizat) corespondența dintre formulele (163) și (166), apoi (164) și (167) și, în sfârșit, (165) și (178).

⁴⁸⁶Pentru ultimele două reguli de actualizare de la pasul M trebuie să fie satisfăcute condițiile $\sum_{i=1}^n w_{ij} \neq 0$, pentru $j \in \{1, \dots, K\}$. Tinând cont de definiția dată mai sus pentru ponderile w_{ij} , cu $i \in \{1, \dots, n\}$ și $j \in \{1, \dots, K\}$, condiția $\sum_{i=1}^n w_{ij} \neq 0$ (pentru j fixat) revine la următoarea

este din \mathbb{R}^d (sau alt spațiu de genul acesta), nu din \mathbb{R} cum suntem obișnuiți. Unele din următoarele formule (preluate din documentul *Matrix Identities*, de Sam Roweis, 1999) vă pot fi de folos:

$$(1e) \quad (A^{-1})^\top = (A^\top)^{-1}$$

$$(2b) \quad |A^{-1}| = \frac{1}{|A|}$$

$$(4a) \quad \frac{\partial}{\partial X} |AXB| = |AXB|(X^{-1})^\top = |AXB|(X^\top)^{-1}$$

$$(4b) \quad \frac{\partial}{\partial X} \ln |X| = (X^{-1})^\top = (X^\top)^{-1}$$

$$(5a) \quad \frac{\partial}{\partial X} a^\top X = \frac{\partial}{\partial X} X^\top a = a$$

$$(5b) \quad \frac{\partial}{\partial X} X^\top AX = (A + A^\top)X$$

$$(5c) \quad \frac{\partial}{\partial X} a^\top Xb = ab^\top$$

$$(5e) \quad \frac{\partial}{\partial X} a^\top Xa = \frac{\partial}{\partial X} a^\top X^\top a = aa^\top$$

$$(5g) \quad \frac{\partial}{\partial X} (Xa + b)^\top C(Xa + b) = (C + C^\top)(Xa + b)a^\top$$

Răspuns:

Pasul E este ușor. Vom calcula

$$w_{ij} \stackrel{not.}{=} p(z_i = j | x_i; \pi', \mu', \Sigma').$$

Folosind regula lui Bayes (combinată cu formula probabilității totale), obținem:

$$p(z_i = j | x_i; \pi', \mu', \Sigma') = \frac{p(x_i | z_i = j; \mu', \Sigma') p(z_i = j; \pi')}{\sum_{l=1}^K p(x_i | z_i = l; \mu', \Sigma') p(z_i = l; \pi')}$$

Aici, probabilitatea $p(x_i | z_i = j; \mu', \Sigma')$ este obținută prin evaluarea densității gaussienei având media μ'_j și matricea de covarianță Σ'_j pentru argumentul x_i , în vreme ce probabilitatea $p(z_i = j; \pi')$ este dată de π'_j , §.a.m.d.

Apoi, la *pasul M*, trebuie să maximizăm, în raport cu parametrii π , μ și Σ , funcția

$$\sum_{i=1}^n \sum_j Q(z_i = j) \ln \frac{p(x_i, z_i; \pi, \mu, \Sigma)}{Q(z_i = j)}, \quad (169)$$

proprietate: $\exists i \in \{1, \dots, n\}$ astfel încât $p(z_i = j | x_i; \pi, \mu, \Sigma) \neq 0$, adică

$$\underbrace{p(x_i | z_i = j; \mu, \Sigma)}_{>0} \underbrace{p(z_i = j | \pi)}_{\pi_j} \neq 0.$$

Restricția aceasta este satisfăcută (în contextul mixturilor de distribuții gaussiene) pentru acei j pentru care $\pi_j > 0$.

Este natural să presupunem $\pi_j > 0$ pentru toți $j \in \{1, \dots, K\}$. Cazurile $\pi_j = 0$ nu sunt interesante (sunt „degenerate”), deci pot fi eliminate din start. Așadar, la pasul de inițializare a algoritmului EM, vom lua toți $\pi'_j > 0$. Vom arăta la finalul demonstrației că această premiză va implica faptul că restricția $\pi_j > 0$ este satisfăcută la fiecare iterație.

unde $Q(z_i = j) \stackrel{\text{not.}}{=} Q_{\pi', \mu', \Sigma'}(z_i = j) \stackrel{\text{not.}}{=} p(z_i = j | x_i; \pi', \mu', \Sigma')$. Conform schemei algoritmiei EM, funcția (169), ale cărei argumente sunt π , μ și Σ , constituie o margine inferioară pentru funcția de log-verosimilitate a datelor observabile, $\sum_{i=1}^n \ln p(x_i | \pi, \mu, \Sigma)$.⁴⁸⁷ Vom rescrie în mod convenabil expresia acestei funcții, ținând cont mai întâi de definiția probabilității condiționate, apoi de definiția distribuției gaussiene multidimensionale și, în sfârșit, de proprietățile logaritmului:

$$\begin{aligned} & \sum_{i=1}^n \sum_{j=1}^K Q(z_i = j) \ln \frac{p(x_i, z_i; \pi, \mu, \Sigma)}{Q(z_i = j)} = \\ & \sum_{i=1}^n \sum_{j=1}^K Q(z_i = j) \ln \frac{p(x_i | z_i = j; \mu, \Sigma) p(z_i = j; \pi)}{Q(z_i = j)} = \\ & \sum_{i=1}^n \sum_{j=1}^K w_{ij} \ln \frac{\frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \cdot \exp\left(-\frac{1}{2}(x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j)\right) \cdot \pi_j}{w_{ij}} = \\ & \sum_{i=1}^n \sum_{j=1}^K w_{ij} \left[-\ln((2\pi)^{d/2} |\Sigma_j|^{1/2} w_{ij}) - \frac{1}{2}(x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) + \ln \pi_j \right] \quad (170) \end{aligned}$$

Ca o consecință a calculelor de mai sus, obiectivul nostru a devenit maximizarea expresiei (170). Pentru aceasta, vom proceda în maniera clasică: vom căuta soluțiile derivatelor parțiale de ordinul întâi ale acestei expresii în raport cu parametrii μ , Σ și π .

Mai întâi vom calcula derivata parțială în raport unde μ_l , cu $l \in \{1, \dots, K\}$.⁴⁸⁸

$$\begin{aligned} & \frac{\partial}{\partial \mu_l} \sum_{i=1}^n \sum_{j=1}^K w_{ij} \left[-\ln((2\pi)^{d/2} |\Sigma_j|^{1/2} w_{ij}) - \frac{1}{2}(x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) + \ln \pi_j \right] \\ &= -\frac{\partial}{\partial \mu_l} \sum_{i=1}^n \sum_{j=1}^K w_{ij} \frac{1}{2}(x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) \\ &= -\frac{1}{2} \sum_{i=1}^n w_{il} \frac{\partial}{\partial \mu_l} (x_i - \mu_l)^\top \Sigma_l^{-1} (x_i - \mu_l) \quad (171) \end{aligned}$$

$$= \frac{1}{2} \sum_{i=1}^n w_{il} \frac{\partial}{\partial \mu_l} (2\mu_l^\top \Sigma_l^{-1} x_i - \mu_l^\top \Sigma_l^{-1} \mu_l) \quad (172)$$

$$= \sum_{i=1}^n w_{il} (\Sigma_l^{-1} x_i - \Sigma_l^{-1} \mu_l). \quad (173)$$

Detalii de calcul:

O primă posibilitate este ca, pornind de la expresia (171) să obținem direct expresia (173),

⁴⁸⁷ De fapt, orice distribuție de probabilitate q peste variabilele neobservabile z este o margine inferioară pentru funcția de verosimilitate a variabilelor observabile x . Însă dintre toate aceste distribuții, cea mai bună — adică, folosind notațiile de la problema 1 de la capitolul *Algoritmul EM*, cea pentru care se atinge $\max_{q(z)} F(q(z), \theta')$, cu $\theta' \stackrel{\text{not.}}{=} (\pi', \mu', \Sigma')$ — este funcția $Q_{\pi', \mu', \Sigma'}$ definită anterior.

⁴⁸⁸ Atenție: μ_l este vector (coloană) și, în consecință, derivata parțială în raport cu μ_l va fi de asemenea un vector (coloană). Similar, derivata parțială în raport cu Σ_l va fi o matrice.

folosind formulele (1e) și (5g) din documentul *Matrix Identities* de Sam Roweis. Formula (1e) se aplică matricei Σ_l^{-1} . Se ține cont că Σ_l este matrice simetrică și inversabilă.⁴⁸⁹

A doua posibilitate este ca, pornind de la expresia (171) să se treacă la expresia (172) — folosind pe de o parte distributivitatea operației de înmulțire a matricelor față de operația de adunare sau de scădere și, pe de altă parte, formula (1e) —, iar apoi de la expresia (172) să se treacă la expresia (173), ținând cont de formulele (5c) — sau, mai simplu, (5a) — și (5b) din documentul menționat.

O a treia posibilitate este să se lucreze în mod clasic (nu derivând în raport cu un vector), și anume scriind derivele parțiale ale expresiei (170) în raport cu fiecare dintre componentele vectorului μ_l (și scriind de asemenea pe componente vectorul x_i și respectiv matricea Σ_l^{-1}).⁴⁹⁰

Egalând expresia (173) cu vectorul-colonă 0 (format din K elemente, toate având valoarea 0 ∈ ℝ), vom obține regula de actualizare pentru μ_l :

$$\mu_l = \frac{\sum_{i=1}^n w_{il} x_i}{\sum_{i=1}^n w_{il}}. \quad (174)$$

Detalii de calcul:

$$\sum_{i=1}^n w_{il} (\Sigma_l^{-1} x_i - \Sigma_l^{-1} \mu_l) = 0 \Leftrightarrow \sum_{i=1}^n w_{il} \Sigma_l^{-1} x_i = \sum_{i=1}^n w_{il} \Sigma_l^{-1} \mu_l$$

Înmulțind ultima egalitate la stânga cu matricea Σ_l , obținem $\sum_{i=1}^n w_{il} x_i = \sum_{i=1}^n w_{il} \mu_l$. Este imediat că membrul drept al acestei ultime egalități se poate scrie ca $(\sum_{i=1}^n w_{il}) \mu_l$.

(Observați că $\sum_{i=1}^n w_{il}$ este un număr real.) Prin urmare, $\mu_l = \frac{\sum_{i=1}^n w_{il} x_i}{\sum_{i=1}^n w_{il}}$.

Să deducem acum relațiile de actualizare de la pasul pasul M pentru matricele Σ_j , unde $j = 1, \dots, K$. Reținând în expresia (170) doar termenii care depind de Σ_j , rezultă că trebuie să maximizăm (în raport cu Σ_j) expresia

$$\begin{aligned} & \sum_{i=1}^n \sum_{j=1}^K w_{ij} \left[\ln \frac{1}{|\Sigma_j|^{1/2}} - \frac{1}{2} (x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) \right] \\ &= \sum_{i=1}^n \sum_{j=1}^K w_{ij} \left[\frac{1}{2} \ln |\Sigma_j^{-1}| - \frac{1}{2} (x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) \right] \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^K w_{ij} \left[\ln |\Sigma_j^{-1}| - (x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) \right]. \end{aligned} \quad (175)$$

Detalii de calcul:

Pentru a obține prima egalitate de mai sus, se folosește formula (2b). Concret, $\frac{1}{|\Sigma_j|^{1/2}} = |\Sigma_j|^{-1/2} = (|\Sigma|^{-1})^{1/2} \stackrel{(2b)}{=} |\Sigma^{-1}|^{1/2}$.

⁴⁸⁹Vedeti problema 20 de la capitolul de *Fundamente*.

⁴⁹⁰În același mod se pot demonstra și formulele (5g), (1e), (5c) și (5b) menționate mai sus (precum și cele care vor fi utilizate mai jos).

În astfel de situații, pentru conveniență, se folosește următorul „artificiu de calcul“: în loc să se lucreze cu *matricea de covarianță* Σ_j (presupusă a fi inversabilă), se preferă să se lucreze cu *matricea de precizie* $\Lambda_j \stackrel{\text{not.}}{=} \Sigma_j^{-1}$.⁴⁹¹ Așadar, vom maximiza dublul expresiei (175) în raport cu Λ_j . Derivând (în raport cu Λ_j), vom obține:

$$\frac{\partial}{\partial \Lambda_j} \sum_{i=1}^n w_{ij} \left[\ln |\Lambda_j| - (x_i - \mu_j)^\top \Lambda_j (x_i - \mu_j) \right] \quad (176)$$

$$= \sum_{i=1}^n w_{ij} \frac{1}{|\Lambda_j|} \frac{\partial}{\partial \Lambda_j} |\Lambda_j| - \sum_{i=1}^n w_{ij} \frac{\partial}{\partial \Lambda_j} [(x_i - \mu_j)^\top \Lambda_j (x_i - \mu_j)] \quad (177)$$

$$= \sum_{i=1}^n w_{ij} \Lambda_j^{-1} - \sum_{i=1}^n w_{ij} (x_i - \mu_j) (x_i - \mu_j)^\top \quad (178)$$

$$= \left(\sum_{i=1}^n w_{ij} \right) \Lambda_j^{-1} - \sum_{i=1}^n w_{ij} (x_i - \mu_j) (x_i - \mu_j)^\top. \quad (179)$$

Detalii de calcul:

Expresia (178) se poate obține direct din expresia (176),⁴⁹² folosind pentru derivarea primului termen formula (4b) și pentru derivarea celui de-al doilea termen formula (5e). Alternativ, expresia (177) se poate obține din expresia (176) folosind pentru primul termen formula de derivare a funcțiilor compuse, iar expresia (178) se poate obține din expresia (177) folosind pentru primul termen formula (4a) (luând $A = B = I$, matricea identitate), iar pentru termenul al doilea formula (5e). În plus, la fiecare dintre aceste două variante de rezolvare, se ține cont și de formula (1e).

Egalând expresia (179) cu 0 (matrice pătratică de dimensiune $K \times K$, ale cărei elemente au, toate, valoarea $0 \in \mathbb{R}$), vom obține:

$$\Sigma_j = \Lambda_j^{-1} = \frac{\sum_{i=1}^n w_{ij} (x_i - \mu_j) (x_i - \mu_j)^\top}{\sum_{i=1}^n w_{ij}}. \quad (180)$$

În sfârșit, vom deriva regula de actualizare de la pasul M pentru parametrii π_j , unde $j = 1, \dots, K$. Reținând în expresia (170) doar acei termeni care depind de π_j , rezultă că va trebui să maximizăm

$$\sum_{i=1}^n \sum_{j=1}^K w_{ij} \ln \pi_j. \quad (181)$$

Să observăm însă — exact ca la rezolvarea problemei 18, cazul $K > 2$ — că soluțiile optime π_j trebuie să satisfacă o restricție suplimentară, și anume $\sum_{j=1}^K \pi_j = 1$, din moment ce $\pi_j \stackrel{\text{not.}}{=} p(z_i = j | \pi)$, pentru orice $j \in \{1, \dots, K\}$ și orice $i \in \{1, \dots, n\}$. Vom proceda la fel și aici, folosind *metoda multimplicatorilor lui Lagrange* pentru a identifica acele valori ale variabilelor π_j care maximizează expresia (181) satisfăcând în același timp restricția $\sum_{j=1}^K \pi_j = 1$. Așadar, vom căuta punctul de optim al polinomului lagrangean

$$\mathcal{L}(\pi) = \sum_{i=1}^n \sum_{j=1}^K w_{ij} \ln \pi_j + \lambda \left(\sum_{j=1}^K \pi_j - 1 \right),$$

⁴⁹¹Aşa s-a procedat și la problema 48 de la capitolul de *Fundamente*.

⁴⁹²Remarcați că $\Lambda_j^\top = \Lambda_j$. Într-adevăr, întrucât $\Sigma_j = \Sigma_j^\top$, urmează că $\Lambda_j = \Sigma_j^{-1} = (\Sigma_j^\top)^{-1} \stackrel{(1e)}{=} (\Sigma_j^{-1})^\top = \Lambda_j^\top$.

unde $\lambda \in \mathbb{R}$ este aşa-numita variabilă („multiplicator“) Lagrange.⁴⁹³ Derivatele parțiale ale polinomului $\mathcal{L}(\pi)$ în raport cu π_j , pentru $j = 1, \dots, K$ sunt:

$$\frac{\partial}{\partial \pi_j} \mathcal{L}(\pi) = \sum_{i=1}^n \frac{w_{ij}}{\pi_j} + \lambda = \frac{1}{\pi_j} \left(\sum_{i=1}^n w_{ij} \right) + \lambda.$$

Egalând cu 0, rezultă:

$$\pi_j = -\frac{\sum_{i=1}^n w_{ij}}{\lambda}.$$

Impunând restricția $\sum_j \pi_j = 1$, rezultă $-\lambda = \sum_{j=1}^K \sum_{i=1}^n w_{ij} = \sum_{i=1}^n \sum_{j=1}^K w_{ij} = \sum_{i=1}^n 1 = n$. (Aici am folosit faptul că $w_{ij} \stackrel{\text{not.}}{\equiv} Q_i(z_i = j)$ și, din moment ce Q reprezintă o funcție de probabilitate, rezultă că $\sum_j w_{ij} = 1$.) Așadar, relația de actualizare de la pasul M pentru parametrul π_j va fi:

$$\pi_j = \frac{1}{n} \sum_{i=1}^n w_{ij}. \quad (182)$$

Desigur, $\pi_j \geq 0$, întrucât $w_{ij} \geq 0$ pentru orice $j \in \{1, \dots, K\}$ și orice $i \in \{1, \dots, n\}$.⁴⁹⁴

Observație (2): Acum cititorul ar trebui să compare relațiile de actualizare de la pasul M⁴⁹⁵ cu formulele de estimare directă (MLE) atunci când variabilele z_i sunt cunoscute / observabile⁴⁹⁶: ele sunt identice, cu excepția faptului că în locul funcțiilor-indicator $1_{\{z_i=j\}}$ acum apar ponderile / mediile w_{ij} . Așadar, formulele de la pasul M al acestui algoritm EM (deci, în cazul învățării nesupervizate) sunt versiuni *poderate* ale formulelor MLE (cazul supervizat). De asemenea, forma acestui algoritm EM ne amintește și de algoritmul de clusterizare K -means: totul este similar, cu excepția faptului că în loc de asignare “hard” a instanțelor x_i la clustere $c(i)$, avem acum asignări “soft” (w_{ij}). Ca și K -means, algoritmul EM este sensibil la problema optimului local, deci rularea repetată, cu diferite valori inițiale pentru unii parametri (sau, pentru toți parametrii) poate fi folositoare. Este limpede că putem asocia algoritmului EM o *interpretare* foarte naturală, și anume aceea a încercării repetitive de a ghici necunoscutele z_i .

Observație importantă: La rezolvarea problemelor 21 și 22 (unde se compară rezultatele algoritmilor K -means și EM/GMM) am folosit algoritmul EM/GMM cazul multidimensional, care a fost descris la problema 20. La rezolvarea problemei 23 am folosit varianța de algoritm EM/GMM cazul multidimensional, descris la problema 56.

⁴⁹³Ca și la problema 18, vom observa că nu este nevoie să ne mai ocupăm și de restricțiile $\pi_j \geq 0$ — care ar trebui adăugate în mod normal la formularea problemei de optimizare —, pentru că, aşa cum vom vedea în curând, soluția pe care o vom găsi pur și simplu derivând polinomul $\mathcal{L}(\pi)$ va satisface (în mod implicit) aceste restricții.

⁴⁹⁴Conform notei de subsol 486, dacă valorile atribuite inițial parametrilor π_j satisfac restricția $\pi_j > 0$ pentru orice $j \in \{1, \dots, K\}$, rezultă că $w_{ij} > 0$ pentru orice $i \in \{1, \dots, n\}$, deci și noile valori obținute pentru π_j la pasul M (pentru $j = 1, \dots, K$) vor fi tot strict pozitive.

⁴⁹⁵Și anume (182), (174) și (180). Acestea coincid cu formulele (166), (167) și (168) din ultima parte a enunțului.

⁴⁹⁶Vedeți *Observația (1)* din enunț și / sau problema 48.a de la capitolul de *Fundamente*.

21.

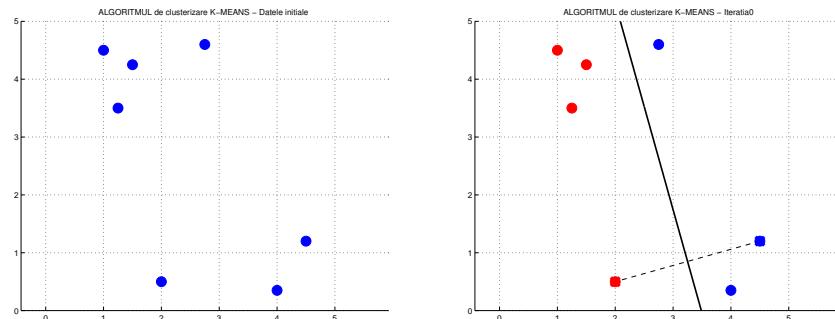
(Algoritmii K -means și EM/GMM: aplicare pe date din \mathbb{R}^2 ; compararea pozițiilor finale ale centroizilor / mediilor)

CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, midterm, pr. 4.a

Se consideră un set de date / instanțe din planul bidimensional, conform tabelului alăturat.

a. Folosind algoritmul K -means, se vor forma două clustere. Centroizii inițiali ai celor două clustere sunt instanțele 5 și respectiv 7. Componenta inițială a clusterelor este definită ca de obicei, cu ajutorul unui separator liniar. (Vedeți imaginea de mai jos, partea dreaptă; instanțele sunt reprezentate prin cerculete.)

Instanță	x	y
$P1$	1.50	4.25
$P2$	1.25	3.50
$P3$	1.00	4.50
$P4$	2.75	4.60
$P5$	4.50	1.20
$P6$	4.00	0.35
$P7$	2.00	0.50



Pentru fiecare iterație executată de algoritmul K -means pe aceste date, veți desena

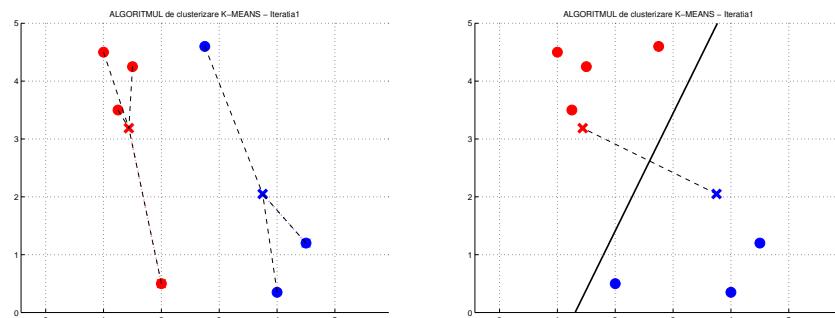
- centroizii clusterelor (reprezentați cu semnul \times);
- separatorii liniari pentru cele două clustere.

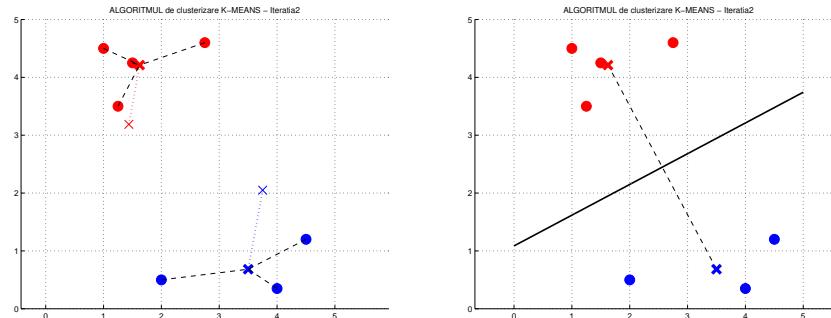
Folosiți oricărădiagramă sunt necesare, până se ajunge la convergență.

b. Cum va dифeri aplicarea pe aceste date a algoritmului EM pentru o mixtură de două distribuții gaussiene bidimensionale față de aplicarea algoritmului K -means?

Răspuns:

a. Aplicând algoritmul K -means, obținem următoarea evoluție a clusterelor:

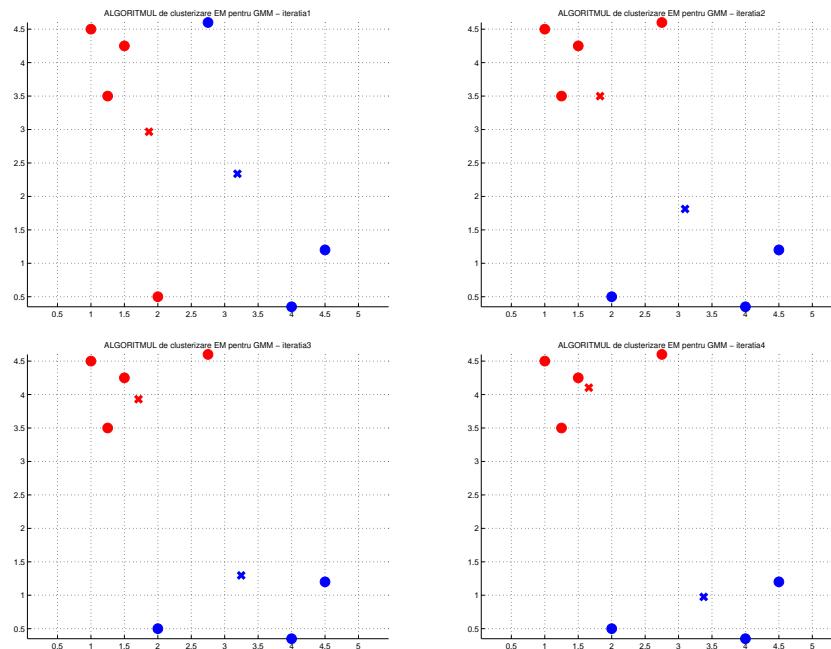




b. La aplicarea algoritmului EM, mediile celor două distribuții (corespondențul centrodizilor clusterelor în K -means) pot evolu mai încet de la o iterare la alta, deoarece sunt influențate de toate instanțele de antrenament (nu doar de cele din clusterul corespunzător, ca în cazul algoritmului K -means). Așadar, am putea avea:

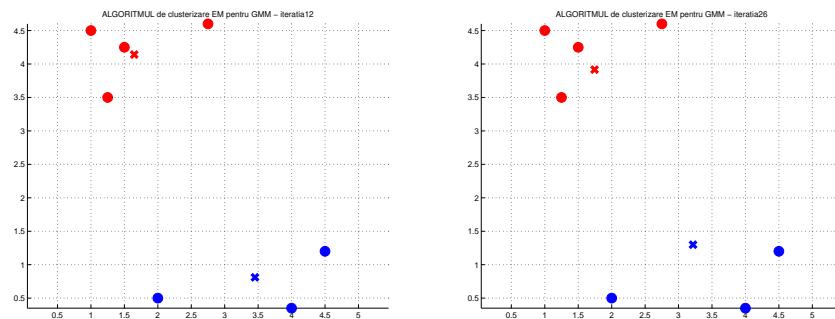
- mai multe interații până la convergență;
- o altă componentă a clusterelor la final;
- o altă poziționare finală a mediilor / centroizilor, chiar dacă se pleacă de la aceleasi poziții inițiale ca pentru algoritmul K -means.

Pentru exemplificare, dacă se aplică algoritmul EM pe aceste date, considerând varianța $\sigma^2 = 2$ (valoare fixată și identică pentru ambele distribuții gaussiene și pentru fiecare dintre cele două axe de coordonate), iar probabilitățile a priori de selecție ale celor gaussiene $1/2$, atunci primele 4 interații vor fi:



Se observă că „deplasarea“ mediilor diferă față de cea a centroizilor de la algoritmul K -means este, într-adevăr, mai „lentă“.

Pe datele considerate, algoritmul se oprește după 12 iterații, rezultatul final fiind cel din figura de mai jos, în partea stângă. Se observă că pozițiile finale ale mediilor sunt foarte apropiate de cele ale centroizilor de la terminarea algoritmului K -means.

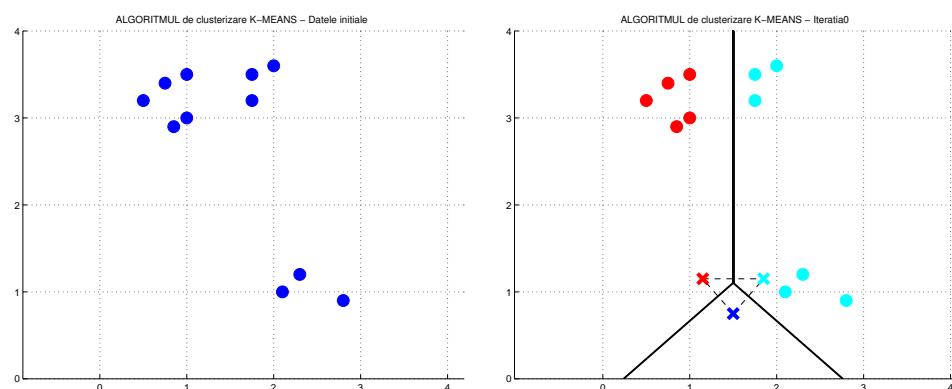


Dacă se consideră $\sigma^2 = 3$, atunci vor fi necesare mai multe iterații (și anume, 26) până la convergență, iar rezultatul este cel din figura de mai sus, în partea dreaptă. Se observă că, deși se obțin aceleași clustere, pozițiile finale ale mediilor sunt diferite față de cazul anterior ($\sigma^2 = 2$).

22. (Algoritmii K -means și EM/GMM: aplicare pe date din \mathbb{R}^2 ; compararea clusterelor finale)

CMU, 2003 fall, T. Mitchell, A. Moore, HW7, pr. 1.c

Executeți algoritmul K -means [eventual în mod manual] pe următorul set de date. Cerculețele reprezintă instanțele de clusterizat, iar cruciulitele (\times) sunt centroizii inițiali ai clusterelor. Separatorii definesc componentă inițială a clusterelor.



În eventualitatea că veți utiliza o implementare, coordinatele datelor și a centroizilor inițiali sunt date în tabelul din partea dreaptă.

- a. Pentru fiecare iterație a algoritmului, desenați centroizii și separatorii care definesc fiecare cluster. Folosiți oricără imagini sunt necesare până veți ajunge la convergență.

Observație: La execuția algoritmului, se consideră că în cazul în care un centroid nu are puncte asignate lui, atunci el rămâne pe loc în iterația respectivă.

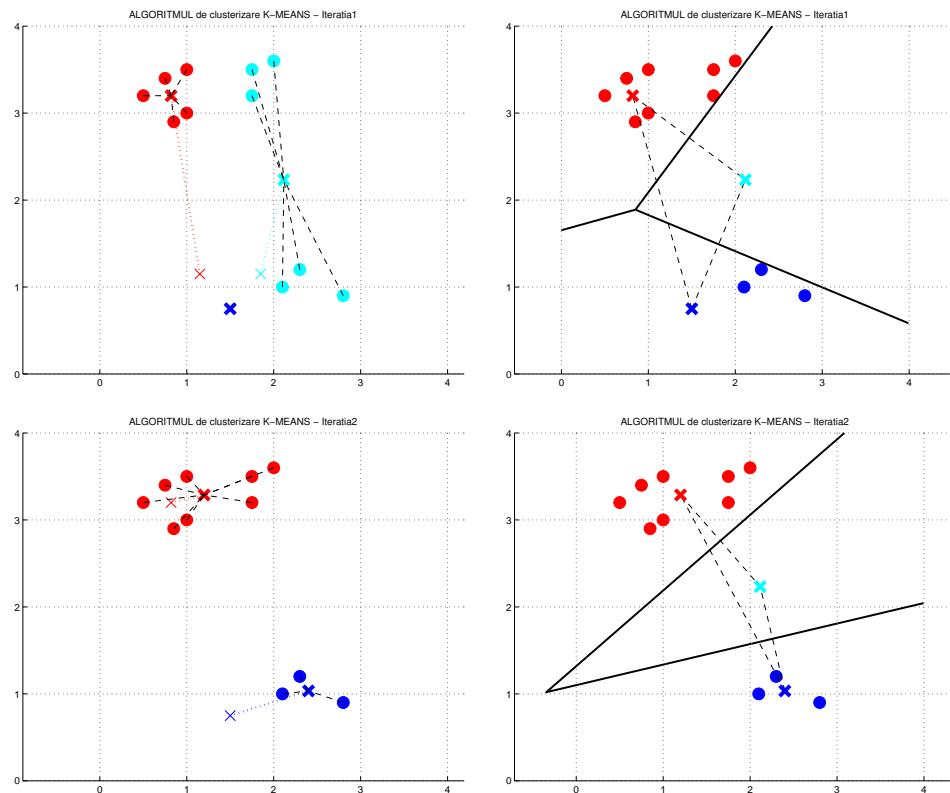
- b. Dacă am rula algoritmul EM pentru o mixtură de 3 distribuții gaussiane bidimensionale pe același set de date, menținând aceeași poziționare a centroizilor inițiali ca mai sus, am ajunge la același rezultat? Justificați răspunsul. Precizați la ce clustere se va ajunge în final.

Punctele	x	y
1	0.50	3.20
2	0.75	3.40
3	1.00	3.50
4	1.00	3.00
5	0.85	2.90
6	1.75	3.20
7	1.75	3.50
8	2.00	3.60
9	2.10	1.00
10	2.30	1.20
11	2.80	0.90

Centroizii	x	y
μ_1	1.50	0.75
μ_2	1.15	1.15
μ_3	1.85	1.15

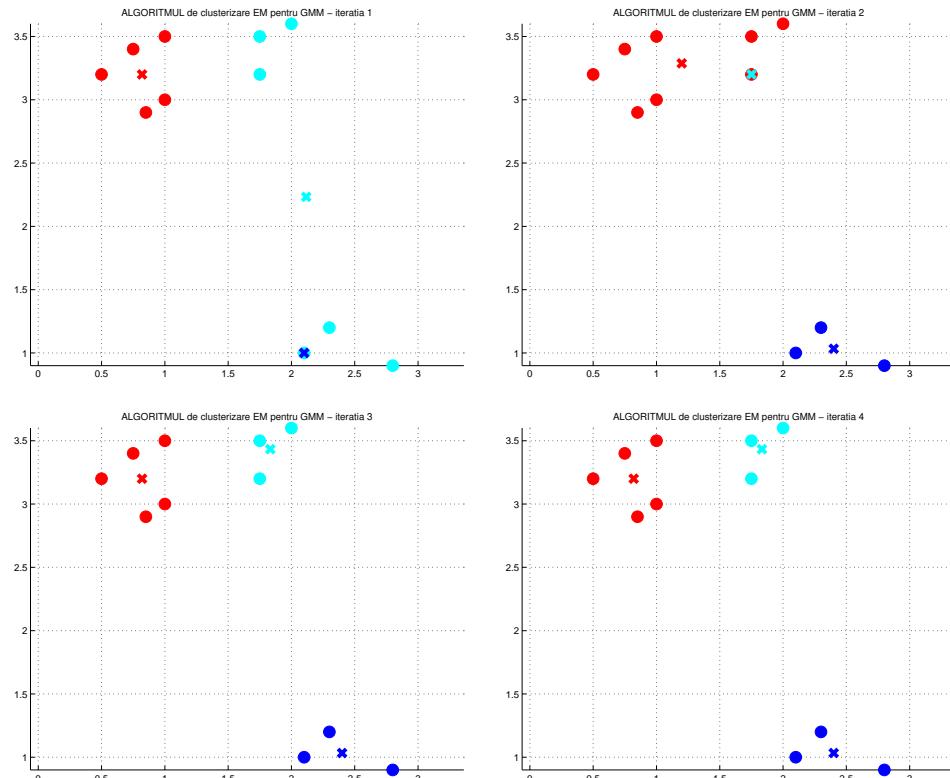
Răspuns:

- a. Aplicând algoritmul K -means obținem:



Se observă că algoritmul va împărți punctele în doar două clustere, al treilea cluster obținut fiind vid.

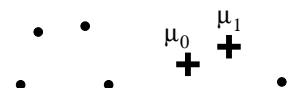
b. Figurile de mai jos reprezintă cele 4 iterații ale algoritmului EM pe aceleasi date inițiale, considerându-se varianță fixă $\sigma^2 = 0.01$ (valoare identică pentru ambele distribuții gaussiene și pentru fiecare din cele două axe de coordonate).



Observăm că se ajunge la o altă configurație finală (mai bună) decât cea obținută de algoritmul K -means: clusterele noi au o mai mare coeziune și nu mai avem un cluster vid.

23. (Algoritmul EM pentru GMM: aplicare pe date din \mathbb{R}^2 ; variante de „mișcare“ a mediilor; evoluția valorilor funcției de log-verosimilitate a datelor observabile)
CMU, 2010 spring, E. Xing, T. Mitchel, A. Singh, midterm, pr. 5.2

Considerăm aplicarea algoritmului EM/GMM pentru a clusteriza datele de mai jos în două clustere. Semnele + indică valorile inițiale ale mediilor (μ_0 și respectiv μ_1).



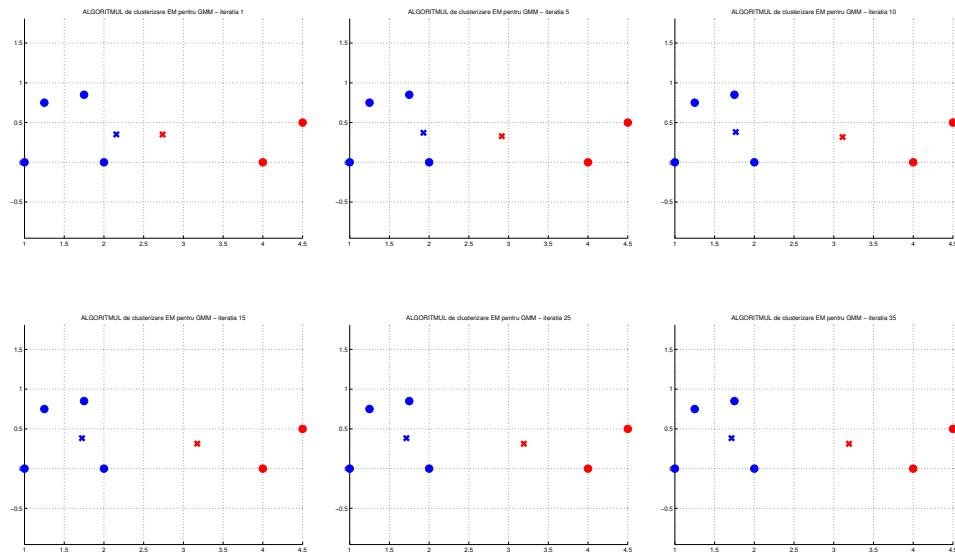
- a. În ce direcție se vor deplasa pozițiile mediilor μ_0 și μ_1 la execuția primelor iterării ale algoritmului EM?
- b. Considerând θ ansamblul parametrilor, precizați cum se comportă $\prod_j P(x_j | \theta)$, probabilitatea marginală a datelor „observabile“, după prima iterare a algoritmului EM: crește ori scade? Justificați pe scurt.

Răspuns:

- a. Folosind o implementare a variantei algoritmului EM/GMM prezentată la problema 58, vom arăta că mișcarea mediilor / centroizilor depinde de valorile varianțelor celor două distribuții gaussiene.⁴⁹⁷ Se disting următoarele cazuri:

Cazul 1: Pentru valori relativ mici ale lui σ^2 , se poate observa că la execuția algoritmului EM μ_0 se va muta din ce în ce mai mult către stânga, în vreme ce μ_1 se va muta inițial spre stânga, iar după aceea către dreapta.

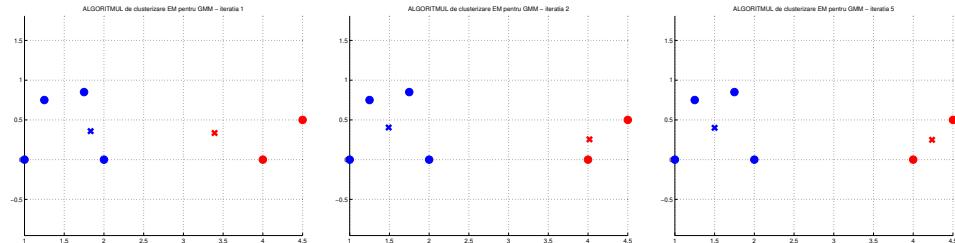
De exemplu, pentru $\sigma^2 = 1.5$ se obțin următoarele iterării mai relevante până la convergență (la care se ajunge în 35 de iterării):



În schimb, pentru $\sigma^2 = 0.5$ (mai mic decât valoarea precedentă), se obțin următoarele iterării mai relevante până la convergență (la care se ajunge în 5 iterării):

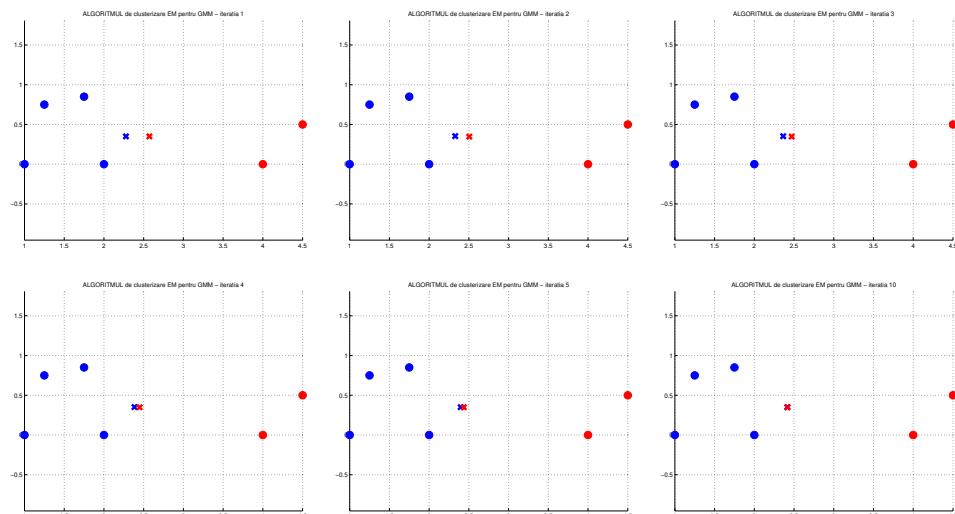
⁴⁹⁷ Coordonatele asociate acestor instanțe și respectiv ale mediilor vă sunt puse la dispoziție în următoarele fișiere, depuse pe site-ul acestei cărți:

<http://profsci.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2010s.EX+TM+AS.midterm.pr5.2.em.dat>,
<http://profsci.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2010s.EX+TM+AS.midterm.pr5.2.init.dat>.



Cazul 2: Pentru valori mai mari pentru σ^2 , se poate observa că μ_0 și μ_1 se vor deplasa unul spre celălalt, fiind posibil chiar să ajungă practic să coincidă.

De exemplu, pentru $\sigma^2 = 3$ se obțin următoarele iterații mai relevante până la convergență (la care se ajunge în 20 de iterații):



b. Probabilitatea $\prod_j P(x^j | \theta)$ reprezintă verosimilitatea datelor observabile (presupunând că aceste date sunt generate independent unele de altele). Conform rezultatului de convergență / corectitudine demonstrat la problema 2 de la capitolul *Algoritmul EM*, această probabilitate nu descrește — adică fie crește fie rămâne la fel — de la o iterație la alta a algoritmului EM. Cresterea se oprește doar la atingerea unui punct de optim local.

24. (Algoritmii *K*-means și EM pentru GMM în \mathbb{R}^2 ; o întrebare de ordin calitativ)

CMU, 2010 spring, E. Xing, T. Mitchel, A. Singh, midterm, pr. 5.1

Considerăm setul de date de antrenament din figura alăturată și doi algoritmi de clusterizare: *K*-means și EM pentru modelare de mixturi de gaussiene (EM/GMM).



Credeți că acești algoritmi vor produce în final aceeași centroizi pentru clustere dacă pornesc de la aceleași valori inițiale μ_1 și μ_2 ? Explicați pe scurt.

Răspuns:

Nu. Diferența esențială dintre cei doi algoritmi constă în faptul că algoritmul K -means folosește asignări de tip “hard”, adică fiecare punct aparține unui singur cluster, în timp ce algoritmul EM/GMM utilizează asignări de tip “soft”, ceea ce înseamnă că fiecare punct este asignat cu o anumită probabilitate fiecărui cluster.

Așadar, în cazul K -means, la fiecare nouă iterație poziția fiecărui centroid este determinată de media punctelor asignate clusterului corespunzător, pe când în cazul algoritmului EM/GMM poziția fiecărui centroid este o anumită medie ponderată obținută din coodinatele tuturor punctelor. Ponderile reprezintă probabilitatea de generare a punctului de către gaussiana asociată respectivului centroid. Prin urmare, centroizii finali obținuți de EM/GMM vor fi (cel puțin sensibil) deplasati față de centroizii determinați de K -means.

25.

(Adevărat sau Fals?)

a.

Liviu Ciortuz

Aplicând algoritmul de clusterizare EM pentru o mixtură de K distribuții gaussiene obținem același rezultat ca și în cazul folosirii algoritmului K -means.

b.

CMU, 2006 (10-701/15-781), final exam, pr. 1.e

În afară de algoritmul EM, pentru a învăța parametrii unei mixturi de distribuții gaussiene se mai poate folosi și metoda gradientului.

Răspuns:

a. Fals. Mai precis: posibil da, posibil nu; depinde de valorile inițiale atribuite varianțelor distribuțiilor gaussiene (chiar dacă valorile inițiale ale centroizilor / mediilor se corespund). Vedeți problema 22.

b. Adevărat. Algoritmul EM este însă preferat, fiindcă uneori derivatele parțiale ale funcției de verosimilitate a datelor observabile în raport cu parametrii de estimat sunt dificil sau chiar imposibil de calculat.

7.2 Probleme propuse

7.2.1 Clusterizare ierarhică

26.

(Clusterizare ierarhică aglomerativă, folosind similaritate de tip “single-linkage”, resp. “complete-linkage”)

prelucrare de Liviu Ciortuz, după CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW4, pr. 2.a

Tabelul următor reprezintă matricea de distanțe pentru [o mulțime de] șase obiecte.

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
<i>A</i>	0					
<i>B</i>	0.12	0				
<i>C</i>	0.51	0.25	0			
<i>D</i>	0.84	0.16	0.14	0		
<i>E</i>	0.28	0.77	0.70	0.45	0	
<i>F</i>	0.34	0.61	0.93	0.20	0.67	0

a. Aplicați algoritmul de clusterizare ierarhică aglomerativă pe aceste date, folosind mai întâi similaritate *single-linkage* și apoi similaritate *complete-linkage*. La fiecare pas al algoritmului, rescrieți în mod corespunzător matricea de distanțe. (De la o iterație la alta, se micșorează cu 1 numărul liniilor precum și al coloanelor folosite.) La final, desenați dendrogramele rezultate.

Indicație: Înălțimea corespunzătoare fiecărui cluster non-singleton (adică, a fiecărui nod intern) din dendrogramă va fi considerată ca fiind egală cu distanța (i.e., conform măsurii de similaritate) dintre cele două sub-clustere constitutive.

- b. Dacă ati lucrat corect, atunci cele două dendrograme obținute la punctul *a* nu coincid [nici măcar] ca structură. Modificați două valori din matricea de distanțe dată mai sus, în aşa fel încât de data aceasta cele două dendrograme care obțin să fie identice ca structură.
- c. Procedați similar pentru *average-linkage*. La actualizarea matricei de distanțe (sau, de „proximitate“) veți ține cont de formula:

$$\begin{aligned} \Delta(X \cup Y, Z) &\stackrel{\text{def.}}{=} \frac{1}{(|X| + |Y|)|Z|} \sum_{x \in X \cup Y} \sum_{z \in Z} d(x, z) \\ &\stackrel{\text{calcul}}{=} \frac{1}{|X| + |Y|} (|X| \Delta(X, Z) + |Y| \Delta(Y, Z)). \end{aligned}$$

unde X , Y și Z sunt clustere disjuncte două câte două, iar notația $|X|$ desemnează cardinalul lui X (adică, numărul de elemente din X).

- d. Demonstrați formula enunțată la punctul *c*.

27.

(Clusterizare ierarhică aglomerativă, folosind similaritate de tip “single-linkage”, “complete-linkage” și “average-linkage”; aplicare pe date din \mathbb{R})

CMU, 2009 spring, Ziv Bar-Joseph, final exam., pr. 9.2

Dorim să clusterizăm numerele de la 1 la 1024, folosind algoritmul de clusterizare ierarhică aglomerativă. Dacă la o iterație oarecare distanțele dintre două perechi de clustere au aceeași valoare, ne vom folosi de ordonarea numerică.⁴⁹⁸

Să se compare rezultatele obținute cu ajutorul celor trei variante ale funcției de similaritate discutate la curs — și anume, “single-linkage”, “complete-linkage” și “average-linkage” —, specificându-se pentru fiecare dintre aceste variante care este numărul de elemente asignate fiecăruiu dintre cele două clustere [care compun clusterul corespunzător nodului rădăcină] de la vîrful dendrogramei rezultate.

⁴⁹⁸Vedeți de exemplu cum s-a procedat la problema 1.

28. (Clusterizare ierarhică aglomerativă: aplicare pe date din \mathbb{R})

CMU, 2012 spring, Ziv Bar-Joseph, midterm, pr. 9.1

Ne propunem să clusterizăm în manieră ierarhică aglomerativă instanțele $2^0, 2^1, 2^2, \dots, 2^n$ (deci, în total, $n + 1$ puncte), folosind distanța euclidiană. Trasați dendrogramanele (adică arborii de clusterizare ierarhică) care se obțin pentru fiecare dintre cele trei tipuri de funcții de similaritate: single-, complete- și average-linkage. Vom considera că fiecare nod neterminal din dendrogramă — adică rădăcina fiecărui cluster non-singleton —, are înălțimea egală cu valoarea măsurii de similaritate (adică, distanța) dintre cele două clustere din care a fost obținut.

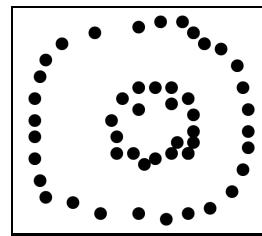
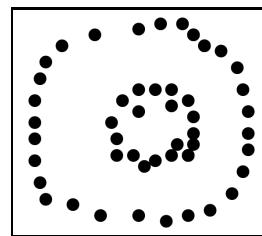
29. (Clusterizare ierarhică aglomerativă: aplicare în manieră intuitivă pe date din \mathbb{R}^2)

CMU, 2010 fall, Ziv Bar-Joseph, midterm, pr. 8.b

a. Folosind distanța euclidiană, aplicați algoritmul de clusterizare ierarhică aglomerativă cu similaritate de tip “single-linkage” pe datele din figura alăturată.

Indicați pe desen care sunt instanțele care formează cele două clustere de la vârful dendrogramei.

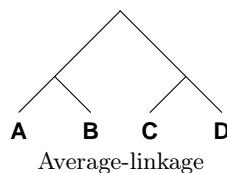
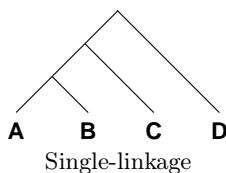
Observație: Nu este necesar să construiți efectiv dendrograma.



30. (Clusterizare ierarhică aglomerativă: raționamente calitative)

CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, midterm, pr. 9.AB

A. În figurile de mai jos sunt reprezentate rezultatele clusterizării a patru puncte (A, B, C și D) folosind o aceeași matrice de distanțe T și două tipuri diferite de măsuri de similaritate, după cum este indicat sub fiecare dintre cei doi arbori de clusterizare considerați.



În cele ce urmează, vom nota spre exemplu cu $T(A, B)$ distanța dintre A și B dată în matricea de intrare, și cu

$$T'(\{A, B\}, \{C, D\}) = \frac{T(A, C) + T(A, D) + T(B, C) + T(B, D)}{4}$$

distanța corespunzătoare măsurii de similaritate “average-linkage” (adică distanța medie) dintre clusterele $\{A, B\}$ și respectiv $\{C, D\}$. Ca de obicei, notăm cu $\min(X, Y)$ cea mai mică dintre valorile X și Y .

Facem *presupunerea* că toate distanțele din matricea de intrare sunt diferite și că în niciuna dintre inegalitățile de mai jos termenii care se compară nu sunt egali (așadar, răspunsul este fie $>$ fie $<$).

La fiecare din cele trei puncte de mai jos, precizați dacă inegalitatea indicată are loc, nu are loc, sau este imposibil de precizat. Justificați în mod riguros alegerea făcută.

a. $\min(T(A, C), T(B, C)) > T(C, D)$.

adevărat fals imposibil de precizat

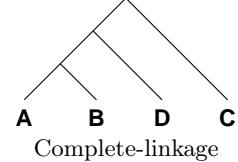
b. $T'(\{A, B\}, \{C\}) > T(C, D)$.

adevărat fals imposibil de precizat

c. $T(A, D) > T(A, C)$.

adevărat fals imposibil de precizat

B. Considerăm arborele de clusterizare din figura alăturată. Este oare posibil ca acest arbore să fi fost obținut pornind de la matricea de distanțe T de la punctul A de mai sus folosind similaritate de tip “complete-linkage” (așa cum se specifică în figură)?



31.

(Clusterizare ierarhică [aglomerativă]:
o altă funcție / măsură de similaritate între clustere:
metrică / distanță lui Ward)
prelucrare făcută de Liviu Ciortuz, după
■ CMU, 2010 fall, Aarti Singh, HW3, pr. 4.1

În acest exercițiu veți analiza o modalitate alternativă pentru a defini distanța dintre două clustere disjuncte, care a fost propusă de către Joe H. Ward în 1963.⁴⁹⁹ O vom numi *metrică lui Ward*.

Această metrică definește distanța dintre două clustere disjuncte X și Y (presupuse ca fiind finite și incluse în \mathbb{R}^d cu $d \in \mathbb{N}^*$) ca fiind dată de creșterea sumei pătratelor distanțelor dintre fiecare instantă x_i și *centroidul* la care ea este asignată, atunci când reunim cele două clustere ca să formăm un cluster nou. Din punct de vedere formal, vom scrie:

$$\Delta(X, Y) = \sum_{x_i \in X \cup Y} \|x_i - \mu_{X \cup Y}\|^2 - \sum_{x_i \in X} \|x_i - \mu_X\|^2 - \sum_{x_i \in Y} \|x_i - \mu_Y\|^2 \quad (183)$$

⁴⁹⁹Ward, J. H., Jr., *Hierarchical Grouping to Optimize an Objective Function*, Journal of the American Statistical Association, 58 (1963), 236–244.

unde, spre exemplu, μ_X este centroidul („centrul de greutate“) clusterului X , iar x_i este o instanță generică dintr-un cluster [oarecare, fixat]. Prin definiție, aici vom considera $\mu_X = \frac{1}{n_X} \sum_{x_i \in X} x_i$, unde n_X este numărul de elemente din X . Mărimea $\Delta(X, Y)$ din formula (183) poate fi interpretată ca reprezentând *costul pentru combinarea* celor două clustere X și Y într-un singur cluster.⁵⁰⁰

- Aduceți expresia din partea dreaptă a formulei (183) la o formă mai simplă. (Redactați toți pașii intermediari.) *Sugestie:* Formula obținută trebuie să fie doar în funcție de numărul de elemente din cele două clustere (n_X și respectiv n_Y) și de distanța $\|\mu_X - \mu_Y\|^2$ dintre centroizii acestor clustere, μ_X și respectiv μ_Y .
- Comentați metrica lui Ward. Ce credeți că urmărește ea să realizeze? *Sugestie:* Varianta simplificată a formulei de mai sus vă va ajuta să răspundeți la întrebarea care a fost pusă aici.
- Presupuneți că vi se dau două *perechi* de clustere, P_1 și P_2 , iar centroizii celor două clustere din P_1 sunt situați la o distanță mai mare decât distanța dintre centroizii clusterelor din perechea P_2 . Oare, folosind metrica lui Ward, algoritmul de clusterizare aglomerativă va alege *întotdeauna* să „combine“ mai întâi cele două clustere din P_2 (fiindcă sunt situate la o ‘distanță’ mai mică)? De ce (nu)? Justificați răspunsul dumneavoastră cu ajutorul unui exemplu simplu.
- La clusterizare, de obicei nu este ușor să decizi care este numărul potrivit de clustere în care trebuie grupate datele. Folosind metrica lui Ward în conjuncție cu clusterizarea aglomerativă, puteți concepe ouristică simplă care să vă ajute să alegeti [bine] numărul de clustere, K ?

32.

(Clusterizare ierarhică aglomerativă: aplicare pe date din \mathbb{R}^2 , folosind metrica lui Ward)

*Liviu Ciortuz, 2018, folosind datele de la
■ Edinburgh, 2009 fall, C. Williams, V. Lavrenko, HW4, pr. 3*

La problema 31 am prezentat o funcție de similaritate numită *metrica lui Ward*. Potrivit acestei metrici, distanța dintre două clustere disjuncte X și Y se definește astfel:

$$\Delta(X, Y) = \sum_{x_i \in X \cup Y} \|x_i - \mu_{X \cup Y}\|^2 - \sum_{x_i \in X} \|x_i - \mu_X\|^2 - \sum_{x_i \in Y} \|x_i - \mu_Y\|^2 \quad (184)$$

unde, spre exemplu, μ_X este centroidul [sau „centrul de greutate“ al] clusterului X , iar x_i este o instanță generică dintr-un cluster [oarecare, fixat]. Prin definiție, aici vom considera $\mu_X = \frac{1}{n_X} \sum_{x_i \in X} x_i$, unde n_X este numărul de elemente din X . (Similar sunt definiți centroizii μ_Y și $\mu_{X \cup Y}$.)

Se poate arăta (vedeți tot problema 31) că

$$\Delta(X, Y) = \frac{n_X n_Y}{n_X + n_Y} \|\mu_X - \mu_Y\|^2. \quad (185)$$

⁵⁰⁰Remarcați faptul că instanțele x_i fiind dintr-un spațiu \mathbb{R}^d , urmează că $\mu_X \in \mathbb{R}^d$ (ca și ceilalți centroizi), iar $\|x\|^2 = x \cdot x$ pentru orice $x \in \mathbb{R}^d$, operatorul · desemnând produsul scalar al vectorilor în \mathbb{R}^d .

Observații:

1. Pentru perechi de clustere (X, Y) și (X', Y') astfel încât $n_X = n_{X'}$ și $n_Y = n_{Y'}$,⁵⁰¹ formula (185) arată că la clusterizare ierarhică folosind merica lui Ward [la o iteratie oarecare] este „favorizată“ acea pereche pentru care centroizii (μ_X și μ_Y , respectiv $\mu_{X'}$ și $\mu_{Y'}$) sunt mai apropiati.
2. Invers, dacă $\|\mu_X - \mu_Y\| = \|\mu_{X'} - \mu_{Y'}\|$, atunci este favorizată perechea pentru care ponderea⁵⁰² (adică $\frac{n_X n_Y}{n_X + n_Y}$, respectiv $\frac{n_{X'} n_{Y'}}{n_{X'} + n_{Y'}}$) este mai mică.

Acste două *observații* vă vor ajuta să simplificați / reduceți foarte mult calculele pe care ar trebui să le faceți pentru a rezolva următoarea cerință!

Aplicați algoritmul de clusterizare ierarhică aglomerativă pe setul de date de la problema 2,

$$\begin{aligned} A &: (-4, -2), B : (-3, -2), C : (-2, -2), D : (-1, -2), E : (+1, -1) \\ F &: (+1, +1), G : (+2, +3), H : (+3, +2), I : (+3, +4), J : (+4, +3) \end{aligned}$$

utilizând [de această dată] metrica lui Ward. Ca rezultat al clusterizării, veți reprezenta *dendrograma* sub formă *aplatizată*, folosind elipse (și indici) pentru a indica clusterele formate.

Coincide rezultatul obținut aici cu vreunul din rezultatele de la problema 2?

Precizare: Dacă la o iterație a algoritmului de clusterizare distanțele (adică similaritățile) dintre două perechi de clustere au aceeași valoare, veți considera că prioritatea la alcătuirea noului cluster este dictată de ordinea alfabetică.

33. (Clusterizare ierarhică aglomerativă: calcularea eficientă a matricei de „proximitate“)
CMU, 2010 fall, Aarti Singh, HW3, pr. 4.2

La curs, atunci când am prezentat algoritmul de clusterizare ierarhică aglomerativă, am arătat că este necesar să actualizăm matricea de distanțe (numită și *matricea de proximitate*; engl., the closeness matrix) la fiecare iterație a algoritmului.⁵⁰³ Calcularea distanțelor dintre clusterul nou-obținut la iterația curentă (prin „contopirea“ clusterelor X și Y) și toate celelalte clustere se poate face, în mod *naiv*, folosind instanțele [individuale] din clustere.

a. Presupunem că dorim să obținem / realizăm o procedură mai eficientă de actualizare a matricei de proximitate, folosind *doar* elementele din forma curentă a matricei de proximitate, adică fără să folosim instanțele [individuale] din clustere. Pentru care dintre următoarele tipuri de funcții de similaritate putem să atingem acest obiectiv?

- i. single-linkage,
- ii. complete-linkage,
- iii. average-linkage,
- iv. matrica lui Ward.⁵⁰⁴

⁵⁰¹ Sau, mai general, atunci când $\frac{n_X n_Y}{n_X + n_Y} = \frac{n_{X'} n_{Y'}}{n_{X'} + n_{Y'}}$.

⁵⁰² Această pondere este jumătate din *media armonică* a cardinalilor n_X și n_Y .

⁵⁰³ Vedeti, ca exemplu, rezolvarea problemei 26.

⁵⁰⁴ Vedeti problema 31.

Prezentați raționamentul dumneavoastră, până la nivel de detaliu.

b. Dacă la vreunul dintre cazurile *i–iv* nu este posibil să atingem obiectivul pe care ni l-am propus la punctul *a* — repetăm, folosind *doar* conținutul curent al matricei de proximitate —, am putea totuși să reușim dacă (*în plus*) vom memora doar câteva informații adiționale pentru fiecare cluster? Pentru fiecare din cazurile identificate [la acest punct] veți specifica în mod clar ce informații adiționale trebuie memorate.

34. (Clusterizare ierarhică: particularizare pentru cazul când similaritatea se exprimă cu ajutorul funcției cosinus între instanțe „normalizate“)

Liviu Ciortuz, 2016, pornind de la Foundations of Statistical Natural Language Processing, C. Manning, H. Schütze, MIT Press, 1999, pag. 507-509

Stim că produsul scalar al doi vectori $x = (x_1, \dots, x_d)$ și $y = (y_1, \dots, y_d)$ din \mathbb{R}^d se definește astfel: $x \cdot y = \sum_{i=1}^d x_i y_i$. Dacă atât x cât și y sunt nenuli, atunci definim $\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$.⁵⁰⁵ Dacă x și y sunt vectori unitari, adică $\|x\| = \|y\| = 1$, atunci definiția de mai sus devine: $\cos(x, y) = x \cdot y$. Atunci când x și y nu sunt unitari, îi putem „normaliza“, înmulțindu-i cu scalarii $1/\|x\|$ și respectiv $1/\|y\|$.

În continuare vom presupune că folosim doar vectori unitari / normalizați.

Funcția cos definită mai sus este folosită uneori în clusterizarea ierarhică, ca măsură de similaritate între instanțe. De obicei, vectorii considerați într-un astfel de context sunt nenegativi (i.e., au toate componente pozitive sau 0), aşa că atunci valoarea maximă a lui cos va fi 1 (și anume, pentru cazul $x = y$), iar 0 va fi valoarea sa minimă (și anume, pentru cazul când x și y sunt vectori ortogonali).

Observație: Trebuie să stim că, deși uneori această funcție este numită „distanță cosinus“, ea nu este de fapt o *măsură de distanță*, fiindcă nu satisfac condițiile de nenegativitate, identitatea indiscernabililor și inegalitatea triunghiului.⁵⁰⁶

- a. Folosind funcția cos, definim *coeziunea* (internă) a unui cluster oarecare X ca fiind

$$S(X) = \frac{1}{C_{|X|}^2} \sum_{x \neq y \in X} \cos(x, y),$$

adică exact ca *medie a similarității* instanțelor din clusterul X . În formula aceasta, am notat cu $|X|$ cardinalul clusterului X , adică numărul de instanțe din X .

Arătați că $S(X)$ poate fi exprimată folosind doar $|X|$ și centrul de greutate (sau, *centrodul*) clusterului X . Acesta din urmă este notat cu μ_X și este definit ca $\frac{1}{|X|} \sum_{x \in X} x$.

- b. Date fiind două clustere oarecare disjuncte A și B , particularizați expresia funcției de coeziune (S) care a fost obținută la punctul precedent pentru clusterul $A \cup B$. Ulterior veți exprima $S(A \cup B)$ folosind doar $|A|$, $|B|$, μ_A și μ_B .

⁵⁰⁵ Se poate arăta că această definiție extinde în mod natural definiția funcției trigonometrice cosinus, aşa cum o stim de la geometria plană din școală. Cosinusul este [măsurat în] funcție de unghiul dintre cei doi vectori.

⁵⁰⁶ Pentru formalizarea acestor proprietăți, vedeti problema 2 de la capitolul *Învățare bazată pe memorare*.

c. Arătați că la clusterizare ierarhică cu similaritate de tip “average-linkage” bazată pe măsura \cos , date fiind două clustere oarecare disjuncte A și B , distanța dintre clusterul $A \cup B$ și un cluster oarecare X disjunct de A și de B se poate exprima folosind doar cardinalii $|A|$, $|B|$ și $|X|$ și centroizii μ_A , μ_B și μ_X .

Sugestie: Ca o consecință ce decurge din definiția similarității “average-linkage”, această distanță se poate calcula ca $S((A \cup B) \cup X) - S(A \cup B) - S(X)$.

În final, ca și la problema 33.cd, rezultă că la fiecare iterație a algoritmului de clusterizare aglomerativă, matricea de „proximitate” poate fi actualizată folosind doar conținutul acestei matrice la iterarea precedentă, cardinalii clusterelor (n_X) și centroizii lor (μ_X).

35.

(Clusterizare ierarhică bottom-up: implementare)

Liviu Ciortuz, 2015

a. Elaborați — de preferință în limbajul de programare C/C++ — implementarea algoritmului de clusterizare ierarhică bottom-up.

Veți considera că instanțele de clusterizat sunt puncte într-un spațiu euclidian \mathbb{R}^d (d fiind un număr natural, $d \geq 1$). Veți presupune de asemenea că instanțele sunt înregistrate într-un fișier text, câte o instanță $x_i = (x_{i_1}, x_{i_2}, \dots, x_{i_d})$ pe fiecare linie, și că între fiecare două attribute / coordonate succesive ale unei astfel de instanțe se află cel puțin un [caracter de tip] spațiu. Numele acestui fișier text, eventual numărul de dimensiuni al spațiului în care se lucrează (d), precum și tipul funcției de similaritate (*single-linkage*, *complete-linkage*, *average-linkage* sau *metrica lui Ward*⁵⁰⁷) vor fi indicate ca argumente în linia de comandă a programului.

Ca punct de plecare puteți considera pseudo-codul din *Foundations of Statistical Natural Processing*, C. Manning, H. Schütze, MIT Press, 2002, pag. 496, pe care însă îl veți extinde cu calculul *înălțimilor* asociate nodurilor din dendrograma rezultantă.

Distanțele dintre instanțele date vor fi calculate și apoi memorate într-o matrice.⁵⁰⁸ Pentru a face în mod eficient actualizarea acestei matrice la fiecare iterare a algoritmului de clusterizare ierarhică bottom-up, veți folosi rezultatele de la problema 33.

Înălțimea unui de nod oarecare din dendrogramă va fi dată de media tuturor distanțelor $d(x, y)$, unde x și y sunt puncte în clusterul respectiv (vedeți, ca exemplu, problema 1).⁵⁰⁹

Ca să „reprezentați” în mod convenabil ieșirea programului puteți folosi o notație bazată pe expresii cu paranteze imbricate.⁵¹⁰ Ulterior, veți putea extinde eventual programul,

⁵⁰⁷Vedeți problema 31.

⁵⁰⁸Inițial veți lucra cu o matrice pătratică. Ulterior, pentru motive de economie de memorie, veți putea înlocui această matrice cu una triunghiulară (intuitiv, aceasta din urmă reprezintă zona de desupra diagonalei principale a matricei pătratice, memorată sub o formă convenabilă).

⁵⁰⁹Coeziunea unui cluster se definește ca fiind inversul acestei medii.

⁵¹⁰De exemplu, pentru dendrograma obținută la problema 1 se poate folosi notația simplă (“flat hierarchy”):

$$((x_1, x_2), ((x_3, ((x_4, x_5), x_6)), ((x_7, x_8), (x_9, x_{10}))))$$

sau, o variantă extinsă cu informații despre ordinea de formare a clusterelor și înălțimile nodurilor (interne) corespunzătoare clusterelor în dendrogramă.

$$((x_1, x_2)_{C1}^{0,2}, ((x_3, ((x_4, x_5)_{C1}^{0,1}, x_6)_{C5}^{0,2})_{C7}^{0,3(6)}, ((x_7, x_8)_{C2}^{0,1}, (x_9, x_{10})_{C3}^{0,1})_{C6}^{0,2(3)})_{C8}^{1,1})_{C9}^{1,77(3)}.$$

cuplându-l cu o interfață grafică, atât pentru colectarea intrărilor (vedeți de exemplu problema 29) cât și pentru ieșiri (trasarea efectivă a dendrogramelor).

Vă sugerăm să testați programul pe datele din problemele 1, 2, 26 și 27.

b. Extindeți implementarea în aşa fel încât, dacă se solicită (printr-o anumită opțiune la linia de comandă), să se livreze toate soluțiile posibile. Testați această variantă a programului dumneavoastră folosind datele de la problema 3.

36.

(Clusterizare ierarhică top-down: implementare)

Liviu Ciortuz, 2016

Implementați — de preferință în C/C++ — algoritmul de clusterizare ierarhică top-down prezentat în problema rezolvată 6. Pentru conveniență, puteți porni de la o implementare oarecare disponibilă pe web pentru algoritmul lui Kruskal sau algoritmul lui Prim pentru aflarea arborelui de cost minim (MST) dintr-un graf. Testați implementarea realizată de dumneavoastră pe datele de la problema pe care am menționat-o anterior.

7.2.2 Algoritmul K -means

37.

(Algoritmul K -means: aplicare în \mathbb{R}^2)**■** *T.U. Dresden, 2006 summer, S. Hölldobler, A. Grossmann, HW3*

Folosiți algoritmul K -means și distanța euclidiană pentru a grupa următoarele 8 instanțe din \mathbb{R}^2 în 3 clustere:

$$A(2, 10), B(2, 5), C(8, 4), D(5, 8), E(7, 5), F(6, 4), G(1, 2), H(4, 9).$$

Se vor lua drept centroizi inițiali punctele A , D și G .

a. Rulați prima iterație a algoritmului K -means. Pe un grid de valori 10×10 veți marca instanțele date, pozițiile centroizilor la începutul primei iterări și compoziția fiecărui cluster la finalul acestei iterări. (Trasați mediatoarele segmentelor determinate de centroizi, ca *separatori* ai clusterelor.)

b. Câte iterări sunt necesare pentru ca algoritmul K -means să conveargă? Desenați pe câte un grid rezultatul rulării fiecărei iterări.

38.

(Algoritmul K -means: aplicare pe date din \mathbb{R}^2)*CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW3, pr. 5*

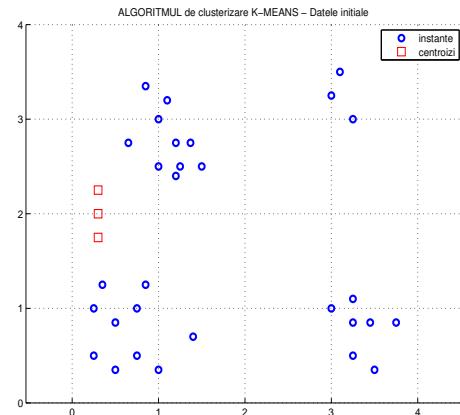
Aplicați algoritmul K -means pe setul de date din imaginea următoare.

Probleme propuse

CLUSTERIZARE

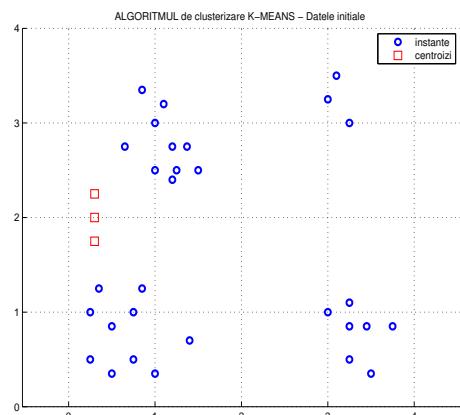
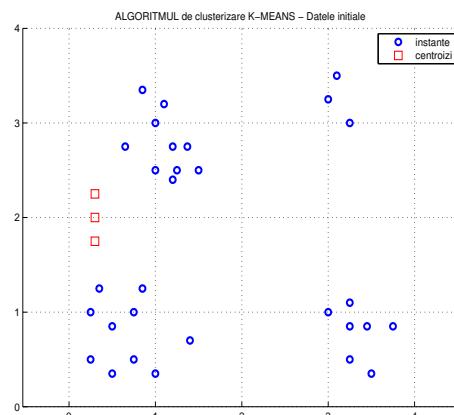
Cerculețele reprezintă instanțele de clusterizat iar pătrățele sunt centroizii inițiali ai clusterelor. Pentru fiecare iterație a algoritmului desenați centroizii și separatorii care definesc fiecare cluster. Folosiți ori căte imagini aveți nevoie până ajungeți la convergență.

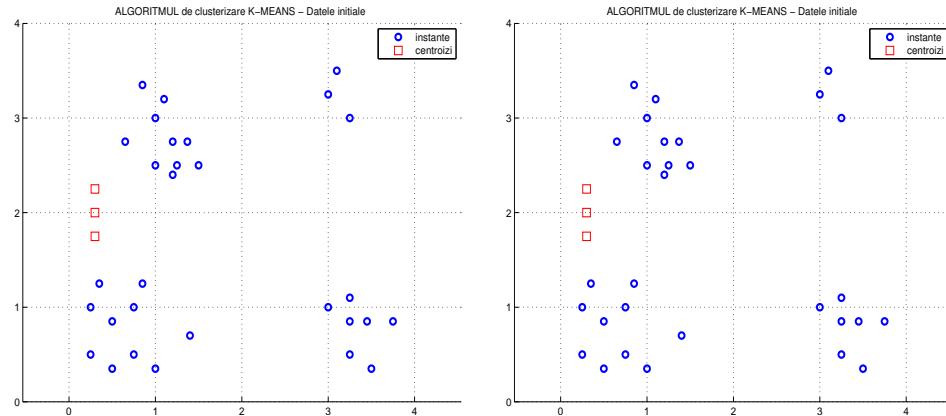
Coordonatele acestor instanțe, precum și cele ale centroizilor vă sunt puse la dispoziție în următoarele fișiere, depuse pe site-ul acestei cărți:



<http://profinfo.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2004f.TM+AM.HW3.pr5.cl.dat>,
<http://profinfo.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2004f.TM+AM.HW3.pr5.init.dat>.

Observație: La execuția algoritmului se consideră că în cazul în care un centroid nu are puncte asignate lui, atunci el rămâne pe loc în iterată respectivă.





39.

(Algoritmul K -means: aplicare în \mathbb{R} și \mathbb{R}^2 ; verificarea monotoniei „criteriului” coeziunii intra-clustere (J))

Liviu Ciortuz, 2020

În acest exercițiu veți folosi algoritmul K -means în varianta dată în enunțul exercițiului 12.⁵¹¹

Vă readucem aminte definiția aşa-numitului criteriu J , care este o măsură a coeziunii intra-clustere:

$$J(C^{(t)}, \mu^{(t)}) = \sum_{i=1}^n (x_i - \mu_{C^{(t)}(x_i)}^{(t)})^2,$$

unde $C^{(t)}$ este ansamblul clusterelor la momentul / iterată t , apoi $\mu^{(t)}$ desemnează ansamblul centroizilor la iterată t și, în sfârșit, $\mu_{C^{(t)}(x_i)}^{(t)}$ este centroidul clusterului la care este asignată instanța x_i la iterată t .

a. Fie următorul set de date din \mathbb{R} : -9, -8, -7, -6, -5, 5, 6, 6, 7, 7, 8, 8, 9, 9. Considerăm $\mu_1^{(0)} = -20$ și $\mu_2^{(0)} = -10$.⁵¹² Demonstrați în manieră *analitică* (NU numeric!) că pentru $t = 1$ avem

$$J(C^{(t)}, \mu^{(t)}) \leq J(C^{(t-1)}, \mu^{(t-1)}).$$

b. Cerințele de la acest punct sunt similare cu cele de la punctul precedent, însă de data aceasta veți lucra pe următorul set de date din \mathbb{R}^2 : $A(-1, 0)$, $B(1, 0)$, $C(0, 1)$, $D(3, 0)$, $E(3, 1)$; drept centroizi inițiali veți considera $\mu_1^{(0)} = A$ și $\mu_2^{(0)} = E$.⁵¹³

⁵¹¹Antenie! În eventualitatea că la o iterată oarecare a algoritmului K -means un cluster este vid, centroidul său rămâne pe loc la iterată respectivă.

⁵¹²Acestea sunt datele de la exercițiul 17.

⁵¹³Acestea sunt datele de la exercițiul 7.

40. (Algoritmul K -means: convergență)
CMU, 2009 spring, Ziv Bar-Joseph, final exam, pr. 1.7

Fie un set de date neetichetate și o K -partiție a acestui set de date, generată la sfârșitul unei iterații oarecare a algoritmului K -means. Este posibil ca algoritmul K -means să reviziteze această K -partiție?

Observație: Noțiunea de K -partiție a fost introdusă la problema 11.

41. (Algoritmul K -means: discuție asupra alegerii valorii lui K)
CMU, 2010 fall, Aarti Singh, HW3, pr. 3.4
CMU, 2015 spring, T. Mitchell, N. Balcan, HW7, pr. 1.1

Unul dintre dezavantajele algoritmului K -means este acela că trebuie specificată valoarea parametrului K . Ce părere aveți despre următoarea *strategie*:

Se poate alege K în mod automat, încercând toate valorile lui posibile ($K = 1, \dots, n$, unde n este numărul de instanțe de clusterizat), și reținând apoi acea valoare a lui K pentru care s-a obținut cea mai mică valoare a criteriului „sumei celor mai mici pătrate”, J_K .⁵¹⁴

Justificați de ce această strategie este bună (sau rea).

42. (Algoritmul K -means ca algoritm de optimizare:
maximizarea aproximativă a distanțelor dintre centroizii clusterelor)
■ *CMU, 2010 fall, Aarti Singh, HW3, pr. 5.2*

În această problemă vom lucra cu o versiune a algoritmului K -means ușor modificată față de cea dată în enunțul problemei 12.

Fie $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ o mulțime de instanțe, iar K numărul de clustere cu care vom lucra. De data aceasta vom specifica asignările instanțelor la clustere folosind o *matrice-indicator* $\gamma \in \{0, 1\}^{n \times K}$, cu $\gamma_{ij} = 1$ dacă și numai dacă \mathbf{x}_i aparține clusterului j . Vom impune ca fiecare instanță să aparțină la câte un singur cluster, deci $\sum_{j=1}^K \gamma_{ij} = 1$.

După cum s-a arătat la problema 12, algoritmul K -means „estimează“ matricea γ făcând minimizarea criteriului (sau, a „măsurii de distorsiune“) J , pe care, folosind matricea γ , îl rescriem sub forma

$$J(\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K) = \sum_{i=1}^n \sum_{j=1}^K \gamma_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2,$$

unde $\|\cdot\|$ desemnează norma vectorială L_2 . Concret, algoritmul K -means alternează „estimarea“ matricei γ cu re-calcularea centroizilor $\boldsymbol{\mu}_j$.

Acestea fiind spuse, putem da acum noua versiune a algoritmului K -means:⁵¹⁵

⁵¹⁴Pentru definiția riguroasă a acestui criteriu vedeti problema 12.

⁵¹⁵Această versiune a algoritmului K -means poate servi pentru a introduce [o variantă simplă pentru] algoritmul EM/GMM — vedeti pseudo-codul de la exercițiul 15 — ca versiune probabilistă a algoritmului K -means. De asemenea, această versiune a algoritmului K -means poate servi pentru a demonstra o proprietate interesantă, și anume, *convergența* [în anumite condiții a] rezultatelor acestor doi algoritmi; vedeti exercițiul 57.

- Se initializează în mod arbitrar centroizii $\mu_1, \mu_2, \dots, \mu_K$ și se ia $C = \{1, \dots, K\}$.
- Atâtă timp cât valoarea lui J descrește în mod strict,⁵¹⁶ repetă:
 - Calculează γ astfel:

$$\gamma_{ij} \leftarrow \begin{cases} 1, & \text{dacă } \|\mathbf{x}_i - \mu_j\|^2 \leq \|\mathbf{x}_i - \mu_{j'}\|^2, \forall j' \in C, \\ 0, & \text{în caz contrar.} \end{cases}$$

În caz de egalitate, alege în mod arbitrar cărui cluster (dintre cele eligibile) să-i aparțină \mathbf{x}_i .

- Recalculează μ_j folosind matricea γ actualizată:
Pentru fiecare $j \in C$, dacă $\sum_{i=1}^n \gamma_{ij} > 0$, asignează

$$\mu_j \leftarrow \frac{\sum_{i=1}^n \gamma_{ij} \mathbf{x}_i}{\sum_{i=1}^n \gamma_{ij}}.$$

Altfel, menține neschimbăt centroidul μ_j .

Vom nota cu $\bar{\mathbf{x}}$ media instanțelor date și vom considera următoarele trei cantități:⁵¹⁷

$$\begin{aligned} \text{Variația totală: } T(X) &= \frac{\sum_{i=1}^n \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}{n} \\ \text{Variația intra-clustere: } W_j(X) &= \frac{\sum_{i=1}^n \gamma_{ij} \|\mathbf{x}_i - \mu_j\|^2}{\sum_{i=1}^n \gamma_{ij}} \text{ pentru } j = 1, \dots, K \\ \text{Variația inter-clustere: } B(X) &= \sum_{j=1}^K \left(\frac{\sum_{i=1}^n \gamma_{ij}}{n} \right) \|\mu_j - \bar{\mathbf{x}}\|^2. \end{aligned}$$

- Care este relația dintre aceste trei cantități?

Observație: Veți ține cont că această relație poate să conțină un termen suplimentar care nu este menționat mai sus.

- Folosind relația stabilită la punctul a, arătați că algoritmul K -means poate fi interpretat ca minimizând o medie ponderată a variației intra-clustere în timp ce maximizează (aproximativ) variația inter-clustere.

⁵¹⁶Această condiție de oprire este ușor diferită de cea formulată de Lloyd: oprește execuția algoritmului atunci când matricea γ nu se mai schimbă.

⁵¹⁷Veți observa că $T(X)$, prima dintre aceste cantități este o „măsură“ a împărăstierii instanțelor \mathbf{x}_i față de $\bar{\mathbf{x}}$, centrul de greutate al întregii mulțimi; ea nu depinde de μ sau de γ . Mai exact, $T(X)$ este media pătratelor distanțelor de la instanțele din mulțimea X la $\bar{\mathbf{x}}$. Similar, $W_j(X)$ este media pătratelor distanțelor de la instanțele din clusterul j la centroidul μ_j , care este totuștună cu raportul dintre coeziunea clusterului j și cardinalul acestui cluster. În fine, $B(X)$ este suma ponderată a pătratelor distanțelor de la centroizii μ_j la $\bar{\mathbf{x}}$, centrul de greutate al mulțimii de instanțe X .

43. (Clusterizare partitioanală în \mathbb{R} :
 un algoritm de programare dinamică, cu complexitate $\mathcal{O}(Kn^2)$
 pentru calculul minimului criteriului „celor mai mici pătrate“ (J))
*prelucrare de Liviu Ciortuz, după
 CMU, 2015 spring, T. Mitchell, N. Balcan, HW7, pr. 1.5-8*

Stim că, date fiind instanțele $x_1, \dots, x_n \in \mathbb{R}^d$ cu $d \in \mathbb{N}^*$, algoritmul de clusterizare K -means urmărește să găsească centroizii a K clustere, și anume $\mu_j \in \mathbb{R}^d, j \in \{1, \dots, K\}$, astfel încât suma pătratelor distanțelor de la fiecare instanță la cel mai apropiat centroid să fie minimă (vedeți problema 12). Altfel spus, K -means încearcă să găsească acei μ_1, \dots, μ_K care minimizează valoarea așa-numitului criteriu al „celor mai mici pătrate“:

$$J \stackrel{\text{def.}}{=} \sum_{i=1}^n \min_{j \in \{1, \dots, K\}} \|x_i - \mu_j\|^2,$$

Pentru a realiza aceasta, algoritmul K -means procedează iterativ, alternând asignarea fiecărei instanțe x_i ($i = 1, \dots, n$) la cel mai apropiat centroid cu actualizarea centroizilor clusterelor (concret, μ_j va deveni media instanțelor asignate la clusterul j).

În general, găsirea minimului criteriului J pentru o valoare fixată a lui K este o problemă NP-dificilă (engl., NP-hard). Totuși, se poate arăta că ea poate fi rezolvată în timp polinomial (în n și K) dacă instanțele de clusterizat sunt într-un spațiu unidimensional ($d = 1$). La acest exercițiu ne vom concentra atenția asupra acestui caz.

- a. Să considerăm situația în care $K = 3$ și ne sunt date 4 instanțe, $x_1 = 1, x_2 = 2, x_3 = 5$ și $x_4 = 7$. Care este clusterizarea optimă pentru acest set de date? Cât este valoarea corespunzătoare pentru funcția obiectiv J ?
- b. Am putea fi tentați să credem că în cazul $d = 1$ algoritmul K -means converge în mod cert la valoarea minimă (globală) a criteriului J . Considerând din nou instanțele date la punctul a, arătați că există o asignare suboptimală a lor la clustere, pe care algoritmul K -means nu poate îmbunătăți. (Indicați asignarea, arătați de ce este suboptimală și explicați de ce anume ea nu va putea fi îmbunătățită.)
- c. Presupunem că sortăm instanțele noastre astfel încât $x_1 \leq x_2 \leq \dots \leq x_n$. Demonstrați că orice *asignare optimă* a acestor instanțe la clustere are proprietatea că fiecare cluster [LC: nevid] corespunde unui anumit „interval“ de instanțe. Adică, pentru fiecare cluster [LC: nevid] j există $i_1, i_2 \in \{1, \dots, n\}$, cu $i_1 \leq i_2$, astfel încât clusterul acesta constă din instanțele $\{x_{i_1}, x_{i_1+1}, \dots, x_{i_2}\}$.⁵¹⁸
- d. Concepți un algoritm de clusterizare de tip programare dinamică având complexitatea $\mathcal{O}(Kn^2)$, ca înlocuitor pentru algoritmul K -means în cazul unidimensional.

Indicație: Dat fiind rezultatul de la punctul c, ceea ce trebuie să „optimizăm“ / setăm sunt cele $K - 1$ granițe / margini (engl., boundaries) ale clusterelor, marginea cu numărul de ordine i fiind cea mai mare instanță din clusterul i .

⁵¹⁸Se poate observa imediat că de fapt această proprietate este satisfăcută mereu pe parcursul aplicării algoritmului K -means pe seturi de date din \mathbb{R} .

44. (*K*-means pentru comprimarea imaginilor)
CMU, 2017 fall, Nina Balcan, HW5, pr. 3, question 12

La curs am arătat că algoritmul *K*-means poate fi folosit pentru comprimarea imaginilor. Presupunem că sunt necesari 24 de biți pentru a memora valorile celor trei componente {R, G, B} în care se descompune culoarea fiecărui pixel. Câtăi biți sunt necesari pentru a memora o imagine care are N pixeli?

Vom presupune acum că folosim algoritmul *K*-means (cu K clustere) pentru a comprima imagini și că identificăm [culoarea pentru] fiecare pixel folosind id-ul clusterului său (un număr din multimea $\{0, \dots, K - 1\}$). Câtăi biți sunt necesari acum pentru a memora imaginea [care are N pixeli]?

Care este raportul de compresie?

45. (Algoritmul *K*-means: adaptare [și aplicare] pentru cazul în care, în definirea criteriului J se folosește distanța Manhattan / L_1 în locul pătratului distanței euclidiene; robustețea algoritmului *K*-means la prezența outlier-elor)
CMU, 2010 fall, Aarti Singh, HW3, pr. 5.3
CMU, 2014 spring, B. Poczos, A. Singh, HW3, pr. 1.4
CMU, 2014 spring, Seyoung Kim, HW3, pr. 1.1

a. În vreme ce la problema 12, la definirea criteriului J am folosit pătratul distanței euclidiene, aici vom folosi distanța Manhattan, desemnată prin $\|\cdot\|_1$, ceea ce înseamnă că vom defini [ca nouă „măsură de distorsiune”] funcția

$$J_1(\gamma, \mu_1, \mu_2, \dots, \mu_K) = \sum_{i=1}^n \sum_{k=1}^K \gamma_{ik} \|\mathbf{x}_i - \mu_k\|_1.$$

Vă reamintim faptul că *distanța Manhattan* (d_m) dintre un vector $x = (x_1, \dots, x_p)$ și un alt vector $y = (y_1, \dots, y_p)$, ambele din \mathbb{R}^p este definită astfel:

$$d_m = \sum_{i=1}^p |x_i - y_i|$$

Vom minimiza funcția J_1 folosind o [nouă] variantă a algoritmului *K*-means:⁵¹⁹

- Inițializează centroizii μ_1, \dots, μ_K și colecția de indici ai clusterelor, $C = \{1, \dots, K\}$.
- Atâtă timp cât valoarea lui J_1 încă scade, repetă următorii pași:
 1. Calculează [variabilele-indicator] γ astfel:

$$\gamma_{ik} \leftarrow \begin{cases} 1, & \|\mathbf{x}_i - \mu_k\|_1 \leq \|\mathbf{x}_i - \mu_{k'}\|_1, \forall k' \in C, \\ 0, & \text{în caz contrar.} \end{cases}$$

În cazul în care minimul se atinge pentru mai mulți astfel de k , alege unul dintre ei în mod arbitrar.

⁵¹⁹Această variantă este ușor modificată în raport cu varianta dată la problema 42. Diferența esențială este faptul că formula de actualizare a centroizilor μ_k de la *pasul 2* al corpului iterativ al algoritmului este lăsată nespecificată și va trebui completată.

2. Recalculează μ_k folosind variabilele γ care tocmai au fost actualizate:
Pentru orice $k \in C$, dacă $\sum_{i=1}^n \gamma_{ik} > 0$, redefinește

$$\mu_k \leftarrow ?$$

În caz contrar, elimină k din mulțimea C .

Completați în pseudo-codul de mai sus detaliul care lipsește — adică, regula de actualizare pentru μ_k —, în aşa fel încât algoritmul să producă o secvență descrescătoare de valori ale funcției [obiectiv] J_1 , atâtă timp cât nu a fost încă atins un [punct de] minim local.

Indicație: Puteți folosi următoarea proprietate:

Fie o mulțime formată din n numere reale, $\{r_1, r_2, \dots, r_n\}$.

Soluția problemei de minimizare

$$\min_x \sum_{i=1}^n |x - r_i|$$

este *valoarea mediană* pentru mulțimea $\{r_1, r_2, \dots, r_n\}$.

Dacă $r_1 < r_2 < \dots < r_n$, atunci

- pentru cazul în care n este impar, *valoarea mediană* a secevenței r_1, r_2, \dots, r_n este $r_{\lfloor n/2 \rfloor}$, unde simbolul $\lfloor z \rfloor$ desemnează partea întreagă inferioară a numărului z ;
 - pentru cazul în care n este par, *valoarea mediană* a secevenței r_1, r_2, \dots, r_n este orice număr din intervalul $[r_{n/2}, r_{(n/2)+1}]$. (În consecință, răspunsul la problema noastră nu este în mod neapărat unic!)
- b. Dacă setul de date de clusterizat conține *outlier-e*, ce versiune a algoritmului K -means ar fi indicat să o folosiți — aceasta (cu distanța Manhattan / L_1) ori cea originală (cu distanța euclidiană / L_2)? Justificați.

- c. Fie setul date alăturat (X), fiecare rând / linie reprezentând o instanță.

Dorim să aplicăm algoritmul K -means pe acest set de date, folosind $K = 3$ și distanța Manhattan în scrierea criteriului / funcției de „distorziune“, aşa cum s-a arătat la punctul a.

Clusterele sunt inițializate după cum urmează:

$C1 : \{(1, 1), (3, 3), (11, 11)\}$, $C2 : \{(6, 6), (3, 0), (9, 3)\}$, $C3 : \{(6, 12), (9, 9), (0, 3)\}$.

1	1
3	3
6	6
6	12
9	9
11	11
0	3
3	0
9	3

Execuțați o [singură] iterație a algoritmului K -means.

Care sunt centroizii clusterelor $C1$, $C2$ și $C3$?

Calculați distanțele dintre primul punct, $(1, 1)$, și centroizii fiecărui dintre cele trei clustere. Care anume (dintre acestea trei) este distanța cea mai mică?

Cărui cluster trebuie să-i asignăm instanța $(0, 3)$? Răspundeți pur și simplu indicând indicele clusterului respectiv, adică 1, 2 sau 3.

46.

(Algoritmul K -means++: comparație cu algoritmul K -means)*Liviu Ciortuz, 2018, după**CMU, 2017 fall, Nina Balcan, midterm, pr. 3.2**CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1**Comentariu:*

În exercițiul de față vom considera că la faza de inițializare algoritmul K -means alege centroizii în mod arbitrar dintre instanțele de clusterizat.

Dacă datele de clusterizat sunt [a priori] bine separate în K clustere, atunci este foarte posibil ca la finalul inițializării să existe [măcar] un cluster din care algoritmul nu a selectat niciun punct. În astfel de situații, algoritmul K -means nu va produce clusterele dorite de noi.

În schimb, varianta K -means++ a algoritmului K -means, propusă de David Arthur și Sergei Vassilvitskii în 2007, încearcă să selecteze pentru pozițiile inițiale ale celor K centroizi instanțe care sunt [pe cât se poate] mai distanțate unele de altele. În acest fel, pot fi selectate, cu o probabilitate [destul de] mare, instanțe din toate clusterele.

Formalizare:

K -means++ face inițializarea centroizilor în maniera următoare:

i. alege primul centroid, μ_1 , în manieră uniform aleatorie dintre instanțele de clusterizat, x_1, \dots, x_n . Cu alte cuvinte, alegem mai întâi un indice i în mod uniform aleatoriu din mulțimea $\{1, \dots, n\}$ și fixăm $\mu_1 = x_i$.

ii. pentru $j = 2, \dots, K$:

- Pentru fiecare instanță x_i , calculează distanța D_i până la cel mai apropiat centroid ales / fixat la o iterație anterioară:

$$D_i = \min_{j'=1, \dots, j-1} \|x_i - \mu_{j'}\|.$$

- Alege centroidul μ_j în mod aleatoriu dintre instanțele x_1, \dots, x_n , cu probabilitate proporțională cu D_1^2, \dots, D_n^2 . Altfel spus, alegem un indice i în mod aleatoriu din mulțimea $\{1, \dots, n\}$ cu probabilități egale cu $\frac{D_1^2}{\sum_{i'=1}^n D_{i'}^2}, \dots, \frac{D_n^2}{\sum_{i'=1}^n D_{i'}^2}$, și fixăm $\mu_j = x_i$.

iii. Returnează $\mu \stackrel{\text{not.}}{=} (\mu_1, \dots, \mu_K)$, setul de asignări ale pozițiilor inițiale ale centroizilor clusterelor pentru algoritmul lui Lloyd (K -means).

Vom ilustra acum diferența dintre K -means++ și K -means [la inițializare], folosind un set de date simplu, format din cinci puncte în planul euclidian bidimensional: $A(-1, 0)$, $B(1, 0)$, $C(0, 1)$, $D(3, 0)$ și $E(3, 1)$.⁵²⁰

a. Presupunem că aplicăm algoritmul K -means cu $K = 2$, făcând inițializarea centroizilor cu instanțe din mulțimea pe care tocmai am precizat-o. Presupunem că a fost deja ales centroidul $\mu_1 = A$. Dacă selecția următorului centroid (μ_2) se face în manieră uniform aleatorie — aşa cum procedează îndeobște algoritmul K -means — din mulțimea $\{B, C, D, E\}$, care este probabilitatea ca μ_2 să fie din submulțimea $\{B, C\}$? Justificați.

b. Aplicăm acum pe același set de date algoritmul K -means++, tot cu $K = 2$. Presupunem, ca și la punctul a, că a fost selectat centroidul $\mu_1 = A$. Care este acum probabilitatea ca μ_2 să fie selectat din submulțimea $\{B, C\}$? Rezultă oare într-adevăr o îmbunătățire semnificativă față de [inițializarea făcută de] algoritmul K -means? Justificați riguros.

⁵²⁰ Aceste date sunt preluate de la problema 7.

47. (Algoritmul K -means: varianta „kernelizată“)
*prelucrare de Liviu Ciortuz, după
 ■ CMU, 2015 spring, T. Mitchell, N. Balcan, HW7, pr. 1.2-4*

Atunci când aplicăm algoritmul K -means folosind distanța euclidiană, lucrăm în mod implicit cu *presupozitia* că orice două clustere sunt liniar-separabile. Însă, evident, datele nu satisfac întotdeauna această presupozitie. Un *exemplu* clasic este acela în care avem două clustere constituite din puncte situate în două zone / benzi circulare concentrice din planul euclidian (\mathbb{R}^2).

În general, pentru algoritmii de învățare automată care determină *separatori liniari* este posibil să folosim funcții-nucleu (engl., *kernel functions*, sau, pe scurt, *kernels*) ca să obținem versiuni neliniare. Algoritmul K -means nu face excepție.

Vă reamintim că există două aspecte principale ale problemelor rezolvate cu ajutorul algoritmilor „kernelizați“:

- Soluția unei astfel de probleme se exprimă ca o combinație liniară de „reprezentări“ / „imagini“ ale instanțelor într-un aşa-numit „spațiu de trăsături“,⁵²¹
- Algoritmul folosește doar produsele scalare dintre [vectorii care reprezintă] imagini de instanțe, nu imaginile propriu-zise ($\Phi(x_i)$).⁵²²

În cele ce urmează, vom arăta — în manieră ușor simplificată⁵²³ la punctele a , b și c — că aceste două aspecte pot fi satisfăcute în cazul algoritmului K -means.

- Fie γ_{ij} o *variabilă-indicator*, care este egală cu 1 dacă instanța x_i este [în prezent] asignată la clusterul j și 0 în caz contrar. Arătați că pentru orice j , centroidul clusterului j (notat cu μ_j , unde $j \in \{1, \dots, K\}$) care se actualizează în corpul iterativ al algoritmului K -means poate fi calculat folosind o formulă de genul $\mu_j = \sum_{i=1}^n \alpha_{ij} x_i$, unde coeficienții α_{ij} — pe care îi veți determina — pot fi calculați în funcție de [toate] variabilele-indicator γ .
- Arătați că pentru două puncte oarecare x și x' din \mathbb{R}^d , expresia $\|x - x'\|^2$ poate fi calculată folosind doar (combinări liniare de) produse scalare de elemente (în specă x și x') din \mathbb{R}^d .
- Tinând cont de rezultatele de la punctele a și b , arătați că expresiile $\|x_i - \mu_j\|^2$, care sunt utilizate în corpul iterativ al algoritmului K -means, se pot calcula folosind doar (combinări liniare de) produse scalare dintre instanțele x_1, \dots, x_n .
- Considerând dată o funcție de „mapare“ $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ și funcția-nucleu corespunzătoare $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, cu $K(x, y) = \Phi(x) \cdot \Phi(y)$, scrieți pseudo-codul algoritmului K -means kernel-izat.⁵²⁴ Concret, veți porni cu anumite puncte inițiale ca centroizi și veți folosi răspunsul de la punctul c ca să găsiți cel mai apropiat centroid pentru fiecare instanță,

⁵²¹Formal, date fiind instanțele $x_i \in \mathbb{R}^d$, cu $i = 1, \dots, n$ și $d \in \mathbb{N}^*$, se va lucra cu o funcție de „mapare“ $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$, cu $m \gg d$, operatorul relațional ' \gg ' având semnificația „mult mai mare decât“. Imaginea lui x_i prin funcția Φ este $\Phi(x_i)$. Spațiul de trăsături este \mathbb{R}^m .

⁵²²Această restricție se justifică prin faptul că se consideră doar funcții de mapare Φ pentru care dacă definim $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ prin $K(x, y) = \Phi(x) \cdot \Phi(y)$, atunci K este calculabilă în mod eficient. K se numește funcție-nucleu. *Observație*: nu există nicio legătură între această funcție K și numărul K care este apără în algoritmul de clusterizare K -means.

⁵²³Și anume, lucrând în \mathbb{R}^d în loc de în \mathbb{R}^m .

⁵²⁴Această variantă a algoritmului K -means caută separatori liniari între clusterele ce se vor forma cu $\Phi(x_1), \dots, \Phi(x_n) \in \mathbb{R}^m$, însă folosește efectiv doar instanțele $x_i \in \mathbb{R}^d$ (prin intermediul funcției-nucleu K) nu și imaginile lor prin funcția de mapare (Φ) în \mathbb{R}^m .

utilizând valorile variabilelor-indicator γ_{ij} . Apoi veți face uz repetat de răspunsul de la punctul a pentru a reasigna instanțele la centroizi și pentru a actualiza variabilele γ_{ij} .

48. (Comparație între clusterizarea ierarhică și algoritmul K -means)
CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1.2.a

Pornind de la o dendrogramă (arbore de clusterizare ierarhică) oarecare, putem defini o partiționare a datelor în K clustere „tăind“ ramurile arborelui în dreptul unor anumite nivele, sub rădăcina arborelui.

De exemplu, pentru $K = 2$ putem defini două clustere pornind de la cei doi subarbore ai rădăcinii dendrogramei. Pentru $K = 4$, putem folosi subarboreii nodurilor descendente din nodul-rădăcină, și așa mai departe. (Observați că dacă valoarea lui K nu este putere a lui 2, atunci ar trebui să definim un anumit criteriu pentru a stabili care este prioritatea în alegerea subarborelor.)

Folosind această procedură pentru a forma o partiție (a setului de date de intrare) din rezultatul unui clustering de tip ierarhizat, care dintre cele trei măsuri de similaritate prezentate la curs — “single-linkage”, “complete-linkage” și “average-linkage” — va conduce cel mai probabil la formarea unor clustere foarte asemănătoare cu cele obținute de către algoritmul K -means? (Presupunem că valoarea lui K este o putere a lui 2).

Vă readucem aminte că măsurile de similaritate “single-linkage”, “complete-linkage” și “average-linkage” dintre două multimi X și Y dintr-un spațiu dotat cu o măsură de distanță sunt definite în raport cu minimul, maximul și respectiv media distanțelor dintre perechile de puncte (x, y) cu $x \in X$ și $y \in Y$.

49. (Algoritmul K -means: un caz special; comparație cu algoritmi de clasificare)
CMU, 2003 fall, T. Mitchell, A. Moore, HW7, pr. 1.b

Pentru această problemă folosim algoritmul K -means, unde K este egal cu numărul de instanțe / puncte de clusterizat, iar fiecare cluster este format dintr-un singur punct. Considerând că o anumită măsură de distanță, vom putea clasifica o instanță nouă (de test) asociind-o cu clusterul al cărui centroid se află cel mai aproape de instanța respectivă. Atenție: centroizii clusterelor nu se modifică în faza de test.

Cu ce metodă de clasificare automată este echivalent modelul descris aici?

50. (Algoritmul K -means: implementare)
Liviu Ciortuz, 2016

Implementați — de preferință în C/C++ — algoritmul K -means. Ca și la problema 35, veți considera că instanțele de clusterizat sunt puncte într-un spațiu euclidian \mathbb{R}^d și că ele sunt înregistrate într-un fișier în format CSV (Comma Separated Value). Numele acestui fișier și eventual d vor fi indicate ca argumente în linia de comandă a programului.

Testați implementarea pe datele de la problema 37. Ulterior, puteți extinde implementarea cu funcții de reprezentare grafică, ca la rezolvarea problemelor 7, 8, 9, 10 și 38. O altă extensie utilă este calculul valorii criteriului J la fiecare iterație, așa cum s-a arătat la problemele 12 și 42.

Încă o direcție posibilă pentru extinderea acestei implementări se referă la varianta kernelizată a algoritmului K -means, al cărei obiectiv l-a constituit problema 47. Implementați și apoi testați această variantă pe setul de date corespunzător conceptului XOR, folosind funcția-nucleu polinomială de ordinul al doilea

$$K(x, x') = (x \cdot x' + 1)^2 \quad \forall x, x' \in \mathbb{R}^2.$$

7.2.3 Algoritmul EM pentru modele de mixturi gaussiene

51. (EM/GMM, cazul unidimensional:
aplicarea manuală a unei iterări,
pentru o mixtură de tipul $\mu_1 = \mu_2$ (liber),
 $\pi_1 = \pi_2, \sigma_1 = 1, \sigma_2 = 2$)
*U. Toronto, Radford Neal,
"Statistical Methods for Machine Learning and Data Mining" course,
2014 spring, Practice problems, set 2, pr. 4*

Considerăm o mixtură de două gaussiene unidimensionale, pentru care funcția de densitate de probabilitate (p.d.f.) are pentru o „observație“ oarecare x expresia

$$\frac{1}{2}\mathcal{N}(x|\mu, 1) + \frac{1}{2}\mathcal{N}(x|\mu, 2^2).$$

În această formulă, $\mathcal{N}(x|\mu, \sigma^2)$ desemnează, pentru x , valoarea densității distribuției normale unidimensionale de medie μ și varianță σ^2 . Remarcați faptul că în acest model de mixtură probabilitățile de mixare / selecție sunt egale, apoi că mediile celor două componente sunt egale și, de asemenea, că deviațiile standard ale celor două componente au valorile fixate 1 și respectiv 2. Modelul acesta de mixtură are un singur parametru, μ .

Presupunem că dorim să estimăm valoarea parametrului μ prin metoda maximizării verosimilității, folosind algoritmul EM. Răspundeți la următoarele întrebări privitoare la modul în care operează pașii E și M ai acestui algoritm, atunci când considerăm cele trei instanțe / „observații“ de mai jos:

4.0, 4.6, 2.0.

Vă punem la dispoziție un tabel cu anumite valori ale funcției de densitate pentru distribuția normală standard, de care veți avea probabil nevoie în rezolvarea acestui exercițiu:

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\mathcal{N}(x 0, 1)$	0.40	0.40	0.39	0.38	0.37	0.35	0.33	0.31	0.29	0.27	0.24
x	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	
$\mathcal{N}(x 0, 1)$	0.22	0.19	0.17	0.15	0.13	0.11	0.09	0.08	0.07	0.05	

- a. Găsiți valoarea probabilităților condiționate calculate la pasul E, presupunând că la execuția precedentului pas M estimarea obținută pentru parametrul modelului a fost $\mu = 4$ (iar $\sigma_1 = 1$ și $\sigma_2 = 2$, totdeauna). Remarcați faptul că, întrucât probabilitățile condiționate pentru cele două componente ale mixturii trebuie să se sumeze la valoarea 1, este suficient să se calculeze $p_{i1} \stackrel{\text{not.}}{=} P(\text{componenta}_1|x_i)$ pentru $i = 1, 2, 3$.

Atenție! Nu este suficient să faceți schimbarea de variabilă necesară pentru „standardizare“, și anume $x \leftarrow \frac{x - \mu}{\sigma}$. Trebuie să folosiți formula de legătură între o distribuție gaussiană unidimensională oarecare și cea standard (o puteți demonstra ușor):

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma} \mathcal{N}\left(\frac{x - \mu}{\sigma} | 0, 1\right).$$

b. Folosind probabilitățile pe care le-ați calculat la punctul a , găsiți estimarea pentru parametrul μ care va fi obținută la următoarea execuție a pasului M. Vă reamintim că pasul M maximizează media (engl., expected value) log-probabilității comune pentru instanțele x_1, x_2 și x_3 și variabilele-indicator [pentru componentele mixturii], care sunt latente. Media aceasta se calculează în raport cu distribuția de probabilitate condiționată a variabilelor-indicator [pentru componentele mixturii], care a fost determinată / calculată la precedentul pas E.

Sugestie: Nu este necesar ca în prealabil să faceți elaborarea formulelor algoritmului EM pentru acest caz specific de mixtură de distribuții gaussiane unidimensionale. Este suficient să lucrați cu funcția „auxiliară“ Q corespunzătoare iterației respective.

52.

(EM/GMM, cazul unidimensional, cu parametrii μ_1 și π_1 liberi, $\mu_2 = 3/2\mu_1$, $\pi_2 = 1 - \pi_1$, $\sigma_1 = \sigma_2 = 1$; elaborare + implementare + rulare)

*U. Toronto, Radford Neal,
"Statistical Computation" course, 2003 spring, HW3, pr. 2*

Să presupunem că un ornitolog⁵²⁵ face un studiu referitor la când anume păsările femele se întorc la cuiburile lor după ce fac „expediții“ / zboruri de strângere de hrană pentru puișorii lor. Pentru toate păsările femele studiate, ornitologul cunoaște momentele de timp când puii lor ies din ouă, iar toate măsurările de timp făcute de către ornitolog sunt relative la aceste momente.

Se crede că, după ce au scos puii din ouă, păsările se întorc din „expedițiile“ / zborurile de strângere a hranei la intervale [aproximativ] regulate de timp, θ , 2θ , 3θ , etc., însă cu o anumită variație aleatoare, despre care vom presupune că urmează o distribuție normală / gaussiană de medie 0 și varianță 1. Se presupune că păsările acționează în mod independent unele de altele.

Pentru a evita să deranjeze păsările la momente critice, ornitologul nu a „observat“ / notat timpul de întoarcere din prima „expediție“, pentru nicio pasăre. În schimb, pentru fiecare pasăre, ornitologul a „observat“ / notat timpul de întoarcere fie de la a doua fie de la a treia „expediție“. (Păsările sunt foarte dificil de sărit, aşa că anumite întoarceri nu au fost observate.) Momentele de timp când au avut loc aceste întoarceri constituie datele noastre, X_1, \dots, X_n . Din nefericire, ornitologul nu cunoaște, în legătură cu fiecare instanță X_i , dacă ea reprezintă timpul de întoarcere corespunzător celei de-a doua „expediții“ ori celei de-a treia. Probabilitatea de a observa o a doua întoarcere versus o a treia nu este cunoscută.

Datele pot fi modelate cu ajutorul unei mixturi, ale cărei componente sunt distribuțiile $\mathcal{N}(2\theta, 1)$ și $\mathcal{N}(3\theta, 1)$, cu probabilitățile de mixare / selecție p și respectiv $1 - p$. Cu

⁵²⁵Ornitologia este o ramură a zoologiei, care studiază păsările.

alte cuvinte, expresia funcției de densitate de probabilitate pentru o „observație“ x este următoarea:

$$f(x) = p \cdot \frac{1}{\sqrt{2\pi}} \cdot \exp(-(x - 2\theta)^2/2) + (1 - p) \cdot \frac{1}{\sqrt{2\pi}} \cdot \exp(-(x - 3\theta)^2/2).$$

Considerând datele X_1, \dots, X_n , va trebui să găsiți estimările de verosimilitate maximă pentru θ și p , folosind un algoritm EM pe care îl veți deduce dumneavoastră. Variabilele neobservabile ar trebui să fie variabilele-indicator relativ la care întoarcere — a doua ori a treia — se referă fiecare „observație“ / instantă X_i .

- a. Deducreți formulele de actualizare care se folosesc la pașii E și M ai algoritmului EM corespunzător acestei probleme.
- b. Elaborați o implementare a algoritmului EM pe care l-ați dezvoltat și testați-l pe datele pe care le furnizăm pe site-ul web asociat acestei cărți,⁵²⁶ precum și pe alte date pe care le considerați potrivite. Programul dumneavoastră ar trebui să execute un număr de iterații pe care-l veți specifica (nu trebuie să detectați în mod automat când anume se realizează convergență). De asemenea, ar trebui ca programul să aibă o opțiune de depanare (“debug”) la selectarea căreia să se imprime, la fiecare iterație, valorile parametrilor și ale log-verosimilității. (Aduceți-vă amintire că log-verosimilitatea datelor observabile nu trebuie să descrească de la o iteratie la alta.) Programul trebuie să primească la intrare datele „observabile“, precum și valorile inițiale ale parametrilor.

Vă recomandăm să evaluați cât de bine s-a comportat algoritmul dumneavoastră. În particular, precizați dacă [au apărut indicii că] există maxime locale multiple pe suprafața funcției de verosimilitate.

53. (EM/GMM, cazul unidimensional: estimarea parametrilor μ și π pentru o mixtură de distribuții gaussiene unidimensionale, presupunând $\sigma_1^2 = \sigma_2^2$)

prelucrare de Liviu Ciortuz, după CMU, 2010 fall, Aarti Singh, HW4, pr. 2.1-2

În această problemă, vom rezolva problema mixturii a două distribuții gaussiene pentru o variantă mai generală decât cea care este prezentată în cartea *Machine Learning* a lui Tom Mitchell, pag. 193 (și preluată de noi în problema 15), unde se consideră că $\pi_1 = \pi_2 = 1/2$ și $\sigma_1^2 = \sigma_2^2$. Aici parametrii π_1 și π_2 sunt liberi, deci vom estima atât μ cât și π , pe când acolo se estimau doar mediile μ . Vom vedea și de data aceasta — ca și în problema 15 / cartea lui T. Mitchell — că necunoașterea valorilor varianțelor $\sigma_1^2 = \sigma_2^2$ nu va impiedica calculelor.

Vă reamintim faptul că un model probabilist de tip mixtură este de fapt o funcție densitate de probabilitate care a fost obținută prin extragerea fiecărei instanțe X conform uneia din două distribuții posibile, $P(X|Z = 0)$ sau $P(X|Z = 1)$. Z este o variabilă aleatoare neobservabilă / „ascunsă“ care este definită peste două clase. Pur și simplu, Z indică pentru fiecare instanță X care anume dintre cele două distribuții considerate a generat-o.

Vom considera că $P(Z)$ este o distribuție de tip Bernoulli și că fiecare funcție (densitate) de

⁵²⁶<https://profs.info.uaic.ro/~ciortuz/ML.ex-book/implementation-exercises/UToronto.2003s.HW3.pr2.EM-for-a-particular-uni-varGMM.data+R-code+sol/data2>.

probabilitate condiționată $P(X|Z)$ este o distribuție gaussiană 1-dimensională cu varianță σ^2 . Așadar, funcția (densitate) de probabilitate marginală este:

$$P(X = x) = \sum_{z \in \{1,2\}} P(X = x|Z = z) \cdot P(Z = z),$$

iar ca funcție de parametrii μ și π ea se rescrie astfel:

$$P(X = x | \mu, \pi) = \sum_{z \in \{1,2\}} \pi_z \cdot \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x - \mu_z)^2}{2\sigma^2}}.$$

Parametrii acestui model sunt $\mu = (\mu_1, \mu_2)$ și $\pi = (\pi_1, \pi_2)$, unde μ_z este media gaussienei care corespunde clasei z , iar $\pi_z = P(Z = z)$ este probabilitatea de a extrage / genera o instanță din clasa z . (Rețineți faptul că $\pi_1 + \pi_2 = 1$.) Dorim să folosim algoritmul EM pentru a estima acești parametri din setul de date independent generate $\{x_i\}_{i=1}^m$, unde $x_i \in \mathbb{R}$.

- a. Vom nota cu p_{iz} probabilitatea ca instanța i să fie extrasă din clasa z (adică, $p_{iz} = P(Z = z|X = x_i, \mu, \pi)$). În cadrul iterării t a algoritmului EM, la pasul E se calculează probabilitățile p_{iz} pentru toate perechile de valori posibile ale indicilor i, z , folosind valorile parametrilor care au fost calculate la iterarea precedentă, și anume $\mu^{(t-1)} = (\mu_1^{(t-1)}, \mu_2^{(t-1)})$ și $\pi^{(t-1)} = (\pi_1^{(t-1)}, \pi_2^{(t-1)})$. Scrieți / deduceți expresia de calcul pentru probabilitățile p_{iz} în funcție de acești parametri.
- b. La pasul M de la iterarea t a algoritmului EM, se tratează cantitățile p_{iz} ca fiind niște count-uri fractionale pentru valorile neobservabile z și se actualizează / estimează valorile parametrilor μ și π ca și când punctul (x_i, z) ar fi fost generat / observat de p_{iz} ori. Scrieți / deduceți regula de actualizare a parametrilor $\mu^{(t)}$ și $\pi^{(t)}$ în funcție de probabilitățile p_{iz} .

54.

(EM/GMM, cazul unidimensional, varianta generală: executarea manuală a pasului M)

N. de Freitas, preluat de Kevin Murphy, în ex. 11.7 din "Machine Learning, A Probabilistic Perspective", MIT Press, 2012

În acest exercițiu urmărim să clusterizăm setul de date $x = \{1, 10, 20\}$ din \mathbb{R} , folosind algoritmul EM pentru o mixtură de două distribuții gaussiene (EM/GMM). Presupunem că rezultatul execuției pasului E este cel dat în următoarea matrice:

$$\begin{bmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{bmatrix},$$

în care elementul generic $p_{i,j}$ reprezintă probabilitatea ca „observația“ x_i să fi fost generată de către gaussiana j .

Vă se cere să executați pasul M al algoritmului EM. Veți scrie mai întâi formulele pentru actualizarea parametrilor mixturii formate din cele două gaussiene (formule pe care ar trebui să le știți!), fără să trebuiască să faceți demonstrația lor; trebuie doar să aplicați aceste formule pe setul de date indicat mai sus. Elaborați toate detaliile necesare.

- a. Scrieți expresia așa-numitei *funcții auxiliare* (Q) care se optimizează la acest pas. Vă readucem aminte că funcția auxiliară este media funcției de log-verosimilitate a *datelor complete* (adică, cele *observable* (x_i) și cele *neobservable* (z_{ij})).
- b. Care vor fi noile valori ale probabilităților de selecție π_1 și π_2 după execuția pasului M? Dar ale mediilor μ_1 și μ_2 ? Dar ale varianțelor σ_1^2 și σ_2^2 ?
55. (Algoritmul EM/GMM, cazul unidimensional, varianta generală:
aplicare pe un set de date din \mathbb{R})
MIT, 2016 spring, T. Jaakkola, R. Barzilay, HW5, pr. 5
- Fie mixtura de două distribuții gaussiene $P(x|\theta) = p_1\mathcal{N}(x|\mu_1, \sigma_1^2) + p_2\mathcal{N}(x|\mu_2, \sigma_2^2)$, care are parametrii $\theta = (p_1, p_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$, adică mediile, varianțele și ponderile (probabilitățile de selecție) pentru fiecare dintre cele două gaussiene. Valorile inițiale $\theta^{(0)}$ asignate parametrilor mixturii sunt: $p_1 = p_2 = 0.5$, $\mu_1 = 6$, $\mu_2 = 7$, $\sigma_1 = 1$ și $\sigma_2 = 2$.
- Avem următoarele instanțe ale lui x : $x_1 = 0$, $x_2 = 1$, $x_3 = 5$, $x_4 = 6$ și $x_5 = 7$.
- a. Scrieți expresia funcției de log-verosimilitate, $\ell(x|\theta)$, pe care urmărим să o maximizăm. De ce [trebuie să] folosim algoritmul EM? Furnizează oare *întotdeauna* acest algoritm soluția optimă?
- b. Referindu-ne la prima execuție a pasului E, care sunt instanțele x_i care vor fi cel mai probabil (dar nu în totalitate) asignate celei de-a doua gaussiene? Altfel spus, care sunt punctele x_i pentru care $P(z_i = 2|x_i, \theta^{(0)}) > P(z_i = 1|x_i, \theta^{(0)})$?
- c. Referindu-ne acum la prima execuție a pasului M, în ce direcție se vor deplasa mediile celor două gaussiene?
- d. Tot în legătură cu prima execuție a pasului M, varianțele σ_1^2 și σ_2^2 vor crește sau vor descrește?
- e. Folosind o implementare a algoritmului EM/GMM specifică pentru cazul de față, faceți o rulare pe datele acestei probleme. Cât sunt valorile mediilor celor două gaussiene atunci când algoritmul ajunge la convergență? Care dintre varianțele celor două gaussiene obținute la final este mai mare?

56. (Algoritmul EM/GMM, cazul multidimensional, cu $\Sigma_j = \sigma^2 I$ pt. $j = 1, \dots, K$:
deducerea formulelor de actualizare)
*prelucrare făcută de Liviu Ciortuz, după
■ U. Utah, 2009 fall, ML (CS5350/6350), Piyush Rai*

A. Fie mixtura de distribuții gaussiene

$$gmm(x) = \sum_{j=1}^K \pi_j \mathcal{N}(x|\mu_j, \sigma^2 I),$$

unde $x \in \mathbb{R}^d$, probabilitățile a priori de selecție $\pi_j \in \mathbb{R}$ satisfac (ca de obicei) restricțiile $\pi_j \geq 0$ pentru $j = 1, \dots, K$ și $\sum_{j=1}^K \pi_j = 1$, mediile gaussienelor $j = 1, \dots, K$ sunt vectorii $\mu_j \in \mathbb{R}^d$, iar matricele de covarianta ale acestor gaussiene sunt identice, ba chiar au forma particulară $\sigma^2 I$, cu $\sigma \in \mathbb{R}$ și $\sigma > 0$, matricea I fiind matricea identitate.

Se consideră instanțele $x_1, \dots, x_n \in \mathbb{R}^d$ generate cu distribuția probabilistă de mai sus (*gmm*). Vom asocia fiecărei instanțe x_i un vector-indicator (mai precis, un vector de variabile aleatoare $z_i \in \{0, 1\}^K$, cu $z_{ij} = 1$ dacă și numai dacă x_i a fost generat de gaussiana $\mathcal{N}(x| \mu_j, \sigma^2 I)$.

Deducreți regulile de actualizare din cadrul [pasului E și al pasului M al] algoritmului EM. După cum știm, acest algoritm face estimarea parametrilor $\pi \stackrel{\text{not.}}{=} (\pi_1, \dots, \pi_K)$, $\mu \stackrel{\text{not.}}{=} (\mu_1, \dots, \mu_K)$ și σ , asigurând la convergență atingerea unui maxim local pentru media funcției de log-verosimilitate a datelor complete, adică a celor „observabile“ x_1, \dots, x_n și a celor „neobservabile“ z_1, \dots, z_n .

Sugestie: Pentru rezolvarea problemei, vă recomandăm să parcurgeți următoarele etape:

- a. Se știe că expresia funcției de densitate a distribuției gaussiene multidimensionale (d -dimensionale) de medie $\mu \in \mathbb{R}^d$ și matrice de covarianță $\Sigma \in \mathbb{R}^{d \times d}$ este

$$\frac{1}{(\sqrt{2\pi})^d \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right),$$

unde x și μ sunt considerați vectori-coloană din \mathbb{R}^d , iar operatorul \top desemnează operația de transpunere a vectorilor / matricelor.

Aduceți expresia de mai sus la forma cea mai simplă pentru cazul $\Sigma = \sigma^2 I$.

Vă recomandăm să folosiți faptul că $\|x - \mu\|^2 = (x - \mu)^\top (x - \mu) = (x - \mu) \cdot (x - \mu)$, unde operatorul \cdot desemnează produsul scalar al vectorilor.⁵²⁷

- b. Pentru *pasul E* al algoritmului EM, veți demonstra mai întâi că media $E[z_{ij}] \stackrel{\text{not.}}{=} E[z_{ij}|x_i; \pi, \mu, \sigma]$, unde $x_i = (x_{i,1}, \dots, x_{i,d}) \in \mathbb{R}^d$, $\mu \stackrel{\text{not.}}{=} (\mu_1, \dots, \mu_K) \in (\mathbb{R}^d)^K$ și $\sigma \in \mathbb{R}^+$, are valoarea $P(z_{ij} = 1|x_i; \pi, \mu, \sigma)$, iar apoi veți elabora formula de calcul a acestei probabilități, folosind teorema lui Bayes.

- c. Arătați că expresia funcției de log-verosimilitate a datelor „complete“ în raport cu parametrii π, μ și σ este

$$\ln p(x, z|\pi, \mu, \sigma) = \sum_{i=1}^n \sum_{j=1}^K z_{ij} (\ln \pi_j + \ln \mathcal{N}(x_i|\mu_j, \sigma^2 I)),$$

unde $x \stackrel{\text{not.}}{=} (x_1, \dots, x_n)$ și $z \stackrel{\text{not.}}{=} (z_1, \dots, z_n)$.

Deducreți apoi expresia „funcției auxiliare“ $Q(\pi, \mu, \sigma|\pi^{(t)}, \mu^{(t)}, \sigma^{(t)}) \stackrel{\text{def.}}{=} E[\ln p(x, z|\pi, \mu, \sigma)]$, cu precizarea că media aceasta este calculată în raport cu distribuția / distribuțiile $P(z_{ij}|x_i, \pi^{(t)}, \mu^{(t)}, \sigma^{(t)})$.

Pasul M: Problema de optimizare EM în acest context este

$$(\pi^{(t+1)}, \mu^{(t+1)}, \sigma^{(t+1)}) = \underset{\pi, \mu, \sigma}{\operatorname{argmax}} Q(\pi, \mu, \sigma|\pi^{(t)}, \mu^{(t)}, \sigma^{(t)}), \quad (186)$$

cu restricțiile $\pi_j^{(t+1)} \geq 0$ pentru $j = 1, \dots, K$ și $\sum_{j=1}^K \pi_j^{(t+1)} = 1$.

⁵²⁷Prima din ultimele două egalități implică un ușor abuz (sau, mai degrabă, o convenție) de notație: o matrice reală de dimensiune 1×1 este identificată cu un număr real, care este chiar singurul ei element. Același tip de abuz / convenție a intervenit și în scrierea expresiei $\exp(\dots)$ de mai sus.

Această problemă se rezolvă optimizând funcția ei obiectiv în mod separat în raport cu variabilele π, μ și σ .

- d. Aplicați metoda multiplicatorilor lui Lagrange pentru a rezolva problema de optimizare cu restricții (186) în raport (doar) cu variabilele π .
- e. Optimizați funcția Q (i.e., rezolvați problema de optimizare (186)) în raport cu variabilele μ .
- f. Optimizați funcția Q (i.e., rezolvați problema de optimizare (186)) în raport cu variabila σ .
- g. Sumarați rezultatele obținute la punctele de mai sus (b și d-f), redactând în pseudocod algoritmul EM pentru rezolvarea mixturii de gaussiene din enunț.

B. Analizați ce se întâmplă atunci când modelul de mixturuă de mai sus este generalizat la gaussiene cu matrice de covariantă diagonale oarecare (adică acestea nu mai sunt de forma $\sigma^2 I$). Concret, modelul devine:

$$z_i \sim \text{Categorical}(p_1, \dots, p_K)$$

$$x_i | z_i = j \sim \mathcal{N} \left(\begin{bmatrix} \mu_{j,1} \\ \vdots \\ \mu_{j,d} \end{bmatrix}, \begin{bmatrix} (\sigma_{j,1})^2 & 0 & \dots & 0 \\ 0 & (\sigma_{j,2})^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & (\sigma_{j,d})^2 \end{bmatrix} \right)$$

57. (EM/GMM, cazul multidimensional; o proprietate importantă:
atunci când $\Sigma_j = \sigma^2 I$, $\forall j$ și $\sigma^2 \rightarrow 0$,
algoritmul EM/GMM tinde să se comporte
asemenea algoritmului K-means)
■ CMU, 2008 fall, Eric Xing, HW4, pr. 2.2

Vă readucem aminte că, date fiind instanțele $\mathbf{x}_1, \dots, \mathbf{x}_n$, algoritmul K-means le va grupa în K clustere minimizând criteriul de „distorsiune“ (sau: coeziune intra-clustere) $J_K = \sum_{i=1}^n \sum_{j=1}^K \gamma_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$, unde $\boldsymbol{\mu}_j$ este centroidul clusterului j , iar $\gamma_{ij} = 1$ dacă \mathbf{x}_i aparține clusterului j și $\gamma_{ij} = 0$ în caz contrar. În acest exercițiu vom folosi pentru algoritmul K-means următoarea procedură:⁵²⁸

- Inițializează cu valori arbitrarе ($\boldsymbol{\mu}_j$) centroizii clusterelor ($j = 1, \dots, K$);
 - Iterează până se ajunge la „convergență“:
 - pentru fiecare punct \mathbf{x}_i ($i = 1, \dots, n$), revizuește asignarea sa la clustere: $\gamma_{ij} = 1$ dacă $j = \arg \min_{j'} \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|^2$ și $\gamma_{ij} = 0$ în caz contrar;
 - actualizează pozițiile centroizilor clusterelor:
- $$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^n \gamma_{ij} \mathbf{x}_i}{\sum_{i=1}^n \gamma_{ij}}$$
- pentru $j = 1, \dots, K$.

Considerăm acum o variantă de GMM în care toate componentele mixturii au matricea de covariantă $\sigma^2 I$, unde $\sigma^2 > 0$, iar I este matricea identitate. Presupunem că ni se dă un set de instanțe $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ și că aplicăm algoritmul EM pentru a estima ponderile (π_j)

⁵²⁸Această procedură este ușor simplificată în raport cu formularea de la problema 42.

și mediile ($\boldsymbol{\mu}_j$) acestei mixturi și pentru a obține pentru fiecare instanță \mathbf{x}_i probabilitățile de apartenență la fiecare cluster j .⁵²⁹

Vom presupune că următoarele proprietăți sunt adevărate / satisfăcute pe parcursul întregii execuții a algoritmului EM:

- Există $\varepsilon > 0$ astfel încât $\pi_j \geq \varepsilon, \forall j \in \{1, \dots, K\}$ — adică ponderile mixturii sunt menținute la (o anumită) distanță față de 0 — pe parcursul tuturor iterațiilor.
- Pe parcursul tuturor iterațiilor,

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \neq \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|^2 \quad \forall i \in \{1, \dots, n\}, j \neq j'.$$

Arătați că atunci când $\sigma^2 \rightarrow 0$, expresiile calculate la pasul E al algoritmului EM tind la valorile calculate de regula de actualizare (engl., update rule) pentru variabilele-indicator γ_{ij} din cadrul algoritmului K -means (adică $p(z_{ij} = 1 | x_i) \rightarrow \gamma_{ij}$), deci asignarea “soft” devine “hard”.

58.

(EM/GMM, cazul multidimensional:
deducerea regulii de actualizare pentru Σ ,
atunci când $\Sigma_j = \Sigma$ pt. $j = 1, \dots, K$)
CMU, 2010 fall, Aarti Singh, HW4, pr. 1.1-2

Un model de mixturi de distribuții gaussiene (engl., Gaussian mixture model, GMM) este o familie de distribuții ale căror densități de probabilitate (engl., probability density functions, p.d.f.) au forma următoare:

$$gmm(\mathbf{x}) = \sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \Sigma_j),$$

unde prin $\mathcal{N}(\cdot | \boldsymbol{\mu}, \Sigma)$ am notat funcția de *densitate* a distribuției gaussiene de medie $\boldsymbol{\mu}$ și matrice de covariantă Σ ,⁵³⁰ iar π_1, \dots, π_K sunt ponderile mixturii, care satisfac restricțiile $\sum_{j=1}^K \pi_j = 1$ și $\pi_j \geq 0$ pentru $j \in \{1, \dots, K\}$.

Pseudo-codul algoritmului EM (Expectation-Maximization) pentru învățarea [valorilor parametrilor] unui GMM pornind de la un set de instanțe $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ poate fi schițat astfel:⁵³¹

- Inițializează $\boldsymbol{\mu}_j$, Σ_j și π_j , pentru $j \in \{1, \dots, K\}$.
- Repetă următorii doi pași până la „convergență“:

Pasul E:

$$E[z_{ij}] = P\left(\mathbf{x}_i \in \text{cluster } j \mid \mathbf{x}_i, \{(\pi_{j'}, \boldsymbol{\mu}_{j'}, \Sigma_{j'})\}_{j'=1}^K\right),$$

⁵²⁹Așadar, σ^2 este considerat fixat, iar π_j și $\boldsymbol{\mu}_j$ sunt liberi. Pentru derivarea completă a algoritmului EM/GMM corespunzător situației în care $\Sigma_j = \sigma^2 I, \forall j$, dar σ^2 , $\boldsymbol{\mu}_j$ și π_j sunt toti parametri variabili / liberi, a se vedea problema 56.

⁵³⁰ $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d} \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$, unde d este dimensiunea spațiului (\mathbb{R}^d) în care se lucrează.

⁵³¹Această procedură este ușor simplificată în raport cu formularea algoritmului EM/GMM folosită (în mod implicit) la problema 20.

Pasul M:⁵³²

$$\{(\pi_j, \boldsymbol{\mu}_j, \Sigma_j)\}_{j=1}^K \leftarrow \arg \max \sum_{i=1}^n \sum_{j=1}^K E[z_{ij}] (\ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j, \Sigma_j) + \ln \pi_j)$$

Considerăm o variantă simplificată de GMM, în care toate componentele mixturii au o aceeași matrice de covarianță, adică $\Sigma_j = \Sigma$ pentru $j = 1, \dots, K$. Deducreți regulile de actualizare pentru matricea Σ_j , de la pasul M. (Răspunsul dumneavoastră poate folosi valoarea lui $\boldsymbol{\mu}_j$ deja actualizată la prezentul pas M.)

Indicație: Vă recomandăm să faceți apel la următoarele formule de derivare a matricelor în raport cu vectori / matrice de variabile:⁵³³

$$|A|^{-1} = |A^{-1}| \quad (1e)$$

$$\frac{\partial \ln |\mathbf{X}|}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^\top = (\mathbf{X}^\top)^{-1} \quad (4b)$$

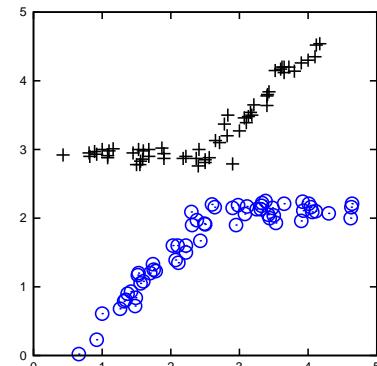
$$\frac{\partial a^\top \mathbf{X} b}{\partial \mathbf{X}} = ab^\top \quad (5c)$$

unde operatorul \top desemnează transpunerea matricelor / vectorilor.⁵³⁴

59.

(Algoritmul EM/GMM, cazul bivariat; chestiuni de ordin calitativ, discutate pe date din \mathbb{R}^2)
CMU, 2006 fall, E. Xing, T. Mitchell, final exam, pr. 4

Profesorul de la cursul de învățare automată a rugat pe trei dintre studenții săi (Alin, Bogdan și Cezar) să folosească GMM (Gaussian Mixture Models) pe setul de date ilustrat în figura alăturată. Exemplele etichetate cu $+$ sunt pozitive, iar cele etichetate cu \circ sunt negative.⁵³⁵



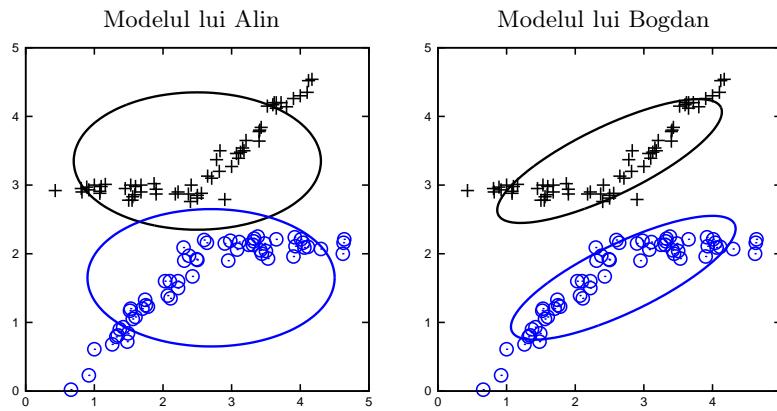
- a. Alin și Bogdan au decis să folosească o distribuție gaussiană pentru exemplele pozitive și una pentru exemplele negative, obținând modelele de mai jos. Dintre acestea, ce model preferi pentru acest set de date? Ce anume determină diferența dintre ele?

⁵³²Pentru justificarea formulei date aici la Pasul M, a se vedea problema 20 de la acest capitol: termenul $-\sum_i \sum_j E[z_{ij}] \ln w_{ij} = -\sum_i \sum_j w_{ij} \ln w_{ij}$ din formula (170) de la pag. 579 a fost lăsat deoparte aici, întrucât el [reprezintă o entropie; vedeti problema 1.c de la capitolul *Algoritmul EM* din prezența culegere, și] nu depinde de π_j , $\boldsymbol{\mu}_j$ și Σ_j .

⁵³³A se vedea *Matrix identities*, Sam Roweis, New York University, June 1999.

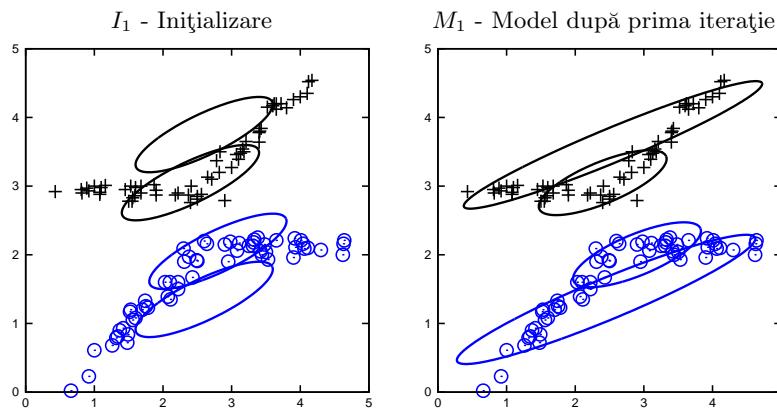
⁵³⁴Ca și la alte probleme de acest gen, vom considera că vectorii \mathbf{x} și $\boldsymbol{\mu}_j$ din \mathbb{R}^d sunt vectori coloană.

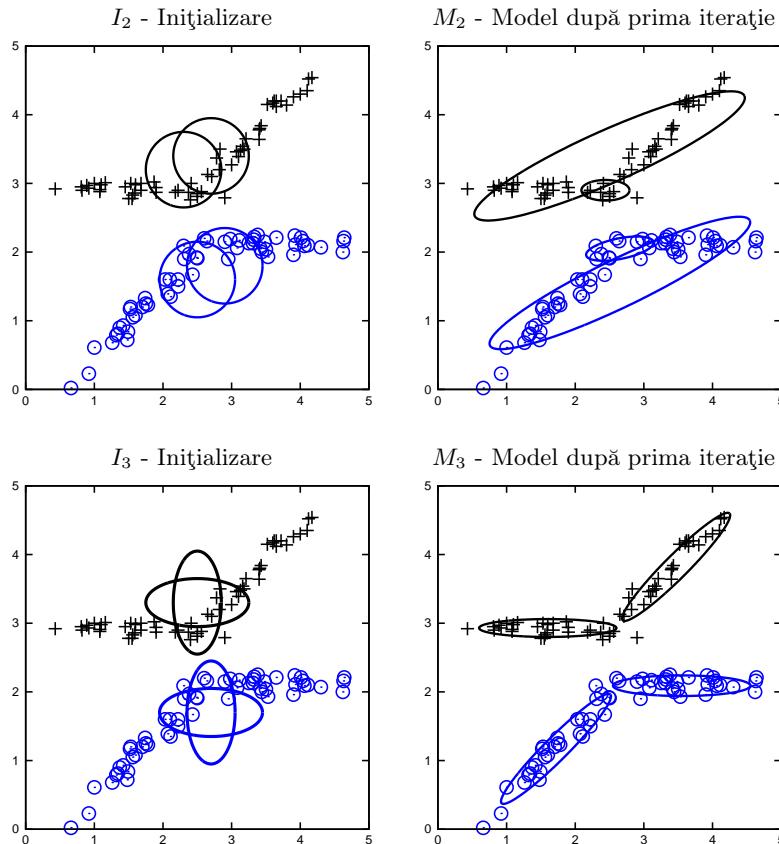
⁵³⁵Coordonatele acestor instanțe vă sunt puse la dispoziție într-un fișier depus pe site-ul acestei cărți: <http://profs.info.uaic.ro/~ciortuz/ML.ex-book/res/CMU.2006f.EX+TM.final.pr4.em.m>.



b. Cezar a decis să utilizeze [LC: o mixtură de] două distribuții gaussiane pentru exemplele pozitive și [LC: o altă mixtură de] două distribuții gaussiane pentru cele negative. El a utilizat algoritmul EM pentru estimarea parametrilor, și de asemenea, a încercat diferite inițializări.

În coloana din stânga sunt reprezentate 3 inițializări diferite, iar în coloana din dreapta sunt reprezentate cele 3 modele obținute după prima iterație. Precizați corespondența dintre modelele din cele două coloane.





60.

(Comparații privind (in)adecvarea algoritmilor de clusterizare K -means și EM/GMM pe diverse tipuri de seturi de date din \mathbb{R}^2)
CMU, 2010 fall, Ziv Bar-Joseph, HW4, pr. 2.2-4

- a. Schițați în planul euclidian un set de date pentru care o mixtură de distribuții gaussiene sferice (i.e., pentru care matricea de covarianță este matricea identitate înmulțită cu un număr pozitiv) poate să modeleze bine datele, însă algoritmul K -means nu poate.
- b. Schițați (tot în planul euclidian) un set de date pentru care o mixtură de distribuții gaussiene diagonale (i.e., pentru care matricea de covarianță poate avea valori diferite de zero doar pe prima diagonală) poate să modeleze bine datele, dar algoritmul K -means și o mixtură de distribuții gaussiene sferice nu pot.
- c. Schițați (tot în planul euclidian) un set de date pentru care o mixtură de distribuții gaussiene având matricile de covarianță nerestricționate pot să modeleze bine datele, dar algoritmul K -means și o mixtură de distribuții gaussiene diagonale nu pot.

Cerință suplimentară: În fiecare dintre cazurile a , b și c , veți multiplica desenele de câte ori este nevoie în aşa fel încât să puteți pune în evidență *granițele de separare / decizie* induse de către fiecare dintre algoritmii menționați în enunț.

61.

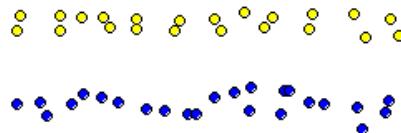
(Comparații privind (in)adecvarea algoritmilor de clusterizare ierarhică folosind diverse măsuri de similaritate, K -means și EM/GMM, pe diverse tipuri de seturi de date din \mathbb{R}^2)
CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, HW4, pr. 2.b

Pentru fiecare dintre figurile următoare, precizați care anume dintre metodele de clusterizare listate mai jos va / vor produce rezultatele indicate, considerând $K = 2$. Alegeți de fiecare dată cea mai probabilă metodă (sau, cele mai probabile metode) și explicați pe scurt de ce ea va (respectiv, ele vor) lucra mai bine decât celelalte metode pe datele respective.

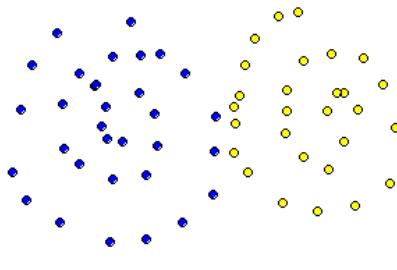
Iată lista de metode de clusterizare pe care le veți considera:

- clusterizare ierarhică cu similaritate *single-linkage*
- clusterizare ierarhică cu similaritate *complete-linkage*
- clusterizare ierarhică cu similaritate *average-linkage*
- K -means
- EM/GMM (fără a face vreo presupunere particulară în legătură cu matricea de covarianță).

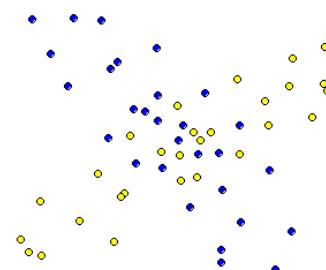
a.



b.



c.



62.

(K -means vs. EM/GMM, cazul multidimensional (π, μ, Σ))
CMU, 2014 fall, W. Cohen, Z. Bar-Joseph, midterm, pr. 8

Redăm mai jos pseudo-codul algoritmului EM — de fapt, doar partea sa iterativă, conținând cei doi pași, E și M — pentru clusterizare prin modelare de mixturi de gaussiene (GMM), cazul multidimensional.⁵³⁶ Îți cerem să faci schimbările minimale(!) necesare pentru a-l transforma într-un pseudo-cod corespunzător buclei principale a algoritmului

⁵³⁶Pentru versiunea completă a acestui algoritm, vedeti problema 20.

de clusterizare K -means. Rescrie pașii care trebuie modificați, folosind spațiul lăsat disponibil sub fiecare pas. Dacă un pas nu necesită schimbări, scrie „*Nicio modificare*“ în spațiul disponibil sub pasul respectiv. Dacă un anumit pas nu este necesar, scrie „*Elimină acest pas*“ în spațiul disponibil sub pasul respectiv.

Pasul E:

Calculează probabilitatea $w_{ij}^{(t)}$ de asignare a instanței x_i la clusterul [corespunzător gaussienei] j , ținând cont de valorile actuale ale parametrilor $\pi^{(t)}$, $\mu^{(t)}$, $\Sigma^{(t)}$, unde t identifică iterația curentă a algoritmului EM.

$$w_{ij}^{(t)} = p(z_i = j | x_i, \pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}) \text{ pentru } i \in \{1, \dots, n\} \text{ și } j \in \{1, \dots, K\}$$

Pasul M:

A. Re-calculează probabilitățile a priori pentru fiecare cluster / componentă a mixturii:

$$\pi_j^{(t+1)} = \sum_{i=1}^n \frac{w_{ij}^{(t)}}{n} \text{ pentru } j \in \{1, \dots, K\}$$

B. Re-calculează [centroizii clusterelor, reprezentăți de] mediile distribuțiilor gaussiene care „modelează“ clusterele:

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^n w_{ij}^{(t)} x_i}{\sum_{i=1}^n w_{ij}^{(t)}} \text{ pentru } j \in \{1, \dots, K\}$$

C. Re-calculează varianțele distribuțiilor gaussiene care „modelează“ clusterele:

$$\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^n \left\{ w_{ij}^{(t)} (x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^\top \right\}}{\sum_{i=1}^n w_{ij}^{(t)}} \text{ pentru } j \in \{1, \dots, K\}$$

63. (EM/GMM, cazul multidimensional, cazul când $\Sigma_k = \sigma_k^2 I$ și probabilitățile de selecție π_k sunt cunoscute / fixate: legătura dintre *metoda gradientului ascendent* pentru maximizarea funcției de log-verosimilitate a datelor observabile și o versiune particulară a algoritmului EM)
CMU, 2010 fall, Aarti Singh, HW4, pr. 1.3

În acest exercițiu vom explora conexiunile dintre algoritmul EM și gradientul ascendent [ca metode de identificare a maximului funcției de verosimilitate a datelor].

Considerăm un model de mixtură de gaussiene (GMM) în care $\Sigma_k = \sigma_k^2 I$ pentru $k = 1, \dots, K$, deci „clopotele“ corespunzătoare acestor K distribuții sunt cu „deschidere“ / secțiune circulară, însă mărimile acestor „deschideri“ sunt în general diferite. Pe lângă aceasta, vom presupune că π_k , ponderile (engl., weights) distribuțiilor din mixtura, sunt cunoscute. Expresia funcției de log-verosimilitate este următoarea:

$$l(\{\boldsymbol{\mu}_k, \sigma_k^2\}_{k=1}^K) = \ln \prod_{i=1}^n \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \sigma_k^2 I) \right) = \sum_{i=1}^n \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \sigma_k^2 I) \right).$$

Formulăm mai jos un algoritm de identificare a maximului funcției de log-verosimilitate, bazat pe metoda gradientului ascendent:

- Inițializează parametrii $\boldsymbol{\mu}_k$ și σ_k^2 , pentru $k \in \{1, \dots, K\}$ cu valori alese în mod aleatoriu ($\boldsymbol{\mu}_k \in \mathbb{R}$ și $\sigma_k^2 \in \mathbb{R}^+$). Setează t , contorul de iterare, la valoarea 1.
- Repetă pașii următori până când se ajunge la convergență:
 - Pentru $k = 1, \dots, K$,

$$\boldsymbol{\mu}_k^{(t+1)} \leftarrow \boldsymbol{\mu}_k^{(t)} + \eta_k^{(t)} \frac{\partial}{\partial \boldsymbol{\mu}_k} l(\{\boldsymbol{\mu}_k^{(t)}, (\sigma_k^2)^{(t)}\}_{k=1}^K),$$

unde $\eta_k^{(t)} > 0$ desemnează o rată de învățare;

- Pentru $k = 1, \dots, K$,

$$(\sigma_k^2)^{(t+1)} \leftarrow (\sigma_k^2)^{(t)} + s_k^{(t)} \frac{\partial}{\partial \sigma_k^2} l(\{\boldsymbol{\mu}_k^{(t+1)}, (\sigma_k^2)^{(t)}\}_{k=1}^K),$$

unde $s_k^{(t)} > 0$ desemnează de asemenea o rată de învățare;

- Incrementează contorul de iterare: $t \leftarrow t + 1$.

Demonstrați că dacă alegem în mod convenabil mărimile ratelor de învățare (engl., step sizes) $\eta_k^{(t)}$ și $s_k^{(t)}$, acest algoritm bazat pe metoda gradientului ascendent este echivalent cu următorul algoritm de tip EM (care este însă ușor modificat în raport cu algoritmii de tip EM pe care i-am întâlnit până acum):

- Inițializează în mod aleatoriu parametrii $\boldsymbol{\mu}_k \in \mathbb{R}$ și $\sigma_k^2 \in \mathbb{R}^+$, pentru $k \in \{1, \dots, K\}$. Setează t la valoarea 1.
- Repetă pașii următori până când se ajunge la convergență:
 - Pas E:

$$\tilde{z}_{ik}^{(t+0.5)} \leftarrow \text{Prob}\left(\mathbf{x}_i \in \text{cluster } k \mid \{(\boldsymbol{\mu}_j^{(t)}, (\sigma_j^2)^{(t)})\}_{j=1}^K, \mathbf{x}_i\right);$$

Pas M:

$$\{\boldsymbol{\mu}_k^{(t+1)}\}_{k=1}^K \leftarrow \arg \max_{\{\boldsymbol{\mu}_k\}_{k=1}^K} \sum_{i=1}^n \sum_{k=1}^K \tilde{z}_{ik}^{(t+0.5)} \left(\ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, (\sigma_k^2)^{(t)} I) + \ln \pi_k \right);$$

Pas E:

$$\tilde{z}_{ik}^{(t+1)} \leftarrow \text{Prob}\left(\mathbf{x}_i \in \text{cluster } k \mid \{(\boldsymbol{\mu}_j^{(t+1)}, (\sigma_j^2)^{(t)})\}_{j=1}^K, \mathbf{x}_i\right);$$

Pas M:

$$\{(\sigma_k^2)^{(t+1)}\}_{k=1}^K \leftarrow \arg \max_{\{\sigma_k^2\}_{k=1}^K} \sum_{i=1}^n \sum_{k=1}^K \tilde{z}_{ik}^{(t+1)} \left(\ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k^{(t+1)}, \sigma_k^2 I) + \ln \pi_k \right);$$

Incrementează contorul de iterare: $t \leftarrow t + 1$.

Ați observat, desigur, că principala modificare (în raport cu ceilalți algoritmi de tip EM) este faptul că a fost inserat un al doilea pas E între pasul M (i.e., aplicarea regulii de actualizare) pentru mediile μ_k și pasul M pentru σ_k^2 .

Sugestie: Mărimile alese pentru ratele de învățare $\eta_k^{(t)}$ și $s_k^{(t)}$ trebuie să fie puse în corespondență cu cei doi pași E.

64. (O legătură între clasificatorul Bayes Naiv gaussian și algoritmul EM/GMM
 [pentru rezolvarea unei mixturi de distribuții gaussiene multidimensionale satisfăcând presupoziția de independență condițională de tip Bayes Naiv];
 o variantă semisupervizată a algoritmului EM/GMM)
CMU, 2010 spring, T. Mitchell, E. Xing, A. Singh, midterm, pr. 5.3

În acest exercițiu vom analiza relația dintre un tip particular de mixturi de distribuții gaussiene (engl., Gaussian Mixture Models, GMMs) multidimensionale și clasificatorul Bayes Naiv gaussian (engl., Gaussian Naive Bayes, GNB). Vom arăta mai întâi că ambii algoritmi folosesc în esență același model probabilist.

Pentru simplitate, în cele ce urmează vom presupune că *mixturile* cu care vom lucra sunt constituite din doar două [componente] gaussiene.⁵³⁷ Vom nota cu μ_0, μ_1, σ_0^2 și σ_1^2 mediile și varianțele celor două gaussiene și vom considera că proporțiile de mixare corespunzătoare celor două componente ale mixturii sunt π_0 și respectiv $1 - \pi_0$. De asemenea, vom folosi simbolul θ pentru a ne referi la întregul set de parametri $(\mu_0, \mu_1, \sigma_0^2, \sigma_1^2, \pi_0)$ care definesc modelul nostru de mixtură:

$$p(x) = \pi_0 \mathcal{N}(\mu_0, \sigma_0^2 I) + (1 - \pi_0) \mathcal{N}(\mu_1, \sigma_1^2 I).$$

Pe de altă parte, *clasificatorul GNB*, știm, face presupunerea că probabilitatea condiționată $p(Y|X)$ satisface relația

$$p(Y|X) = \frac{p(Y) \prod_{k=1}^d p(X^k|Y)}{p(X)},$$

iar fiecare atribut X^k , la condiționare cu [o valoare oarecare a lui] Y , este guvernăt de o distribuție gaussiană. În contextul exercițiului nostru, este natural să considerăm că variabila aleatoare Y este de tip Bernoulli, cu $P(Y = 0) = \pi_0$ și (din nou, pentru simplitate) că toate atributele au aceeași varianță, deci

$$P(X^k|Y = j) \sim \mathcal{N}(\mu_{jk}, \sigma^2).$$

Remarcați faptul că ambii algoritmi specificați mai sus (GNB și EM/GMM) folosesc acest model probabilist „generativ“. Cu alte cuvinte, ambele modele presupun că generăm instanțe alegând mai întâi o valoare pentru Y (în funcție de parametrul π_0), iar apoi extragem / generăm un X conform distribuției gaussiene condiționate / determinate de valoarea lui Y . Singura diferență este că antrenăm clasificatorul GNB folosind instanțe

⁵³⁷Veți putea constata singuri că rezultatele care vor fi obținute aici se pot extinde ușor la un număr oarecare (supra-unitar) de gaussiene.

etichetate pentru care valoarea lui Y este cunoscută, pe când în cazul [execuției] algoritmului EM/GMM presupunem că valorile lui Y sunt necunoscute.

- a. [Se știe că] la antrenarea clasificatorului GNB se alege setul de parametri θ care maximizează verosimilitatea datelor:

$$\operatorname{argmax}_{\theta} \prod_{i=1}^n p(x_i, y_i | \theta),$$

unde indicele i a fost folosit pentru a desemna exemplul de antrenament [LC: cu numărul de ordine] i . Vă cerem să scrieți expresia funcției obiectiv pe care urmărește să o maximizeze celălalt algoritm, EM/GMM, atunci când învăță parametrii aceluiasi model, însă fără a cunoaște valorile [pentru] y_i .

- b. Scrieți regulile de actualizare pentru pașii E și M din algoritmul EM standard pentru rezolvarea mixturii de gaussiene.

- c. Clasificatorul GNB este antrenat folosind exemple / instanțe etichetate, în vreme ce algoritmul EM/GMM folosește instanțe neetichetate. Presupunem că avem un set de date de antrenament în care avem instanțe de ambele tipuri. Cunoaștem valorile etichetei y pentru instanțele x_1, x_2, \dots, x_m , în vreme ce instanțele x_{m+1}, \dots, x_{m+n} nu au valori cunoscute pentru y . Propuneți o modalitate de învățare a parametrilor acestui model generativ „semisupervizat“. (Vă sugerăm să adaptați algoritmul EM/GMM pentru acest tip de date și să formulați pașii E și M corespunzători.)

- d. Scrieți funcția obiectiv pe care o maximizează algoritmul EM/GMM modificat (pe care tocmai l-ați obținut). În expresia pe care o veți scrie, va trebui să distingeți între exemplele de antrenament pentru care y este cunoscut și cele pentru care y este necunoscut.

65.

(Algoritmul EM/GMM: implementare)

Liviu Ciortuz, 2016

Se consideră numărul natural $d \geq 1$, precum și o mulțime formată din n instanțe ($X = \{x_1, \dots, x_n\}$) din spațiul \mathbb{R}^d . Obiectivul acestei probleme este să implementați în manieră orientată pe obiecte (pentru aceasta, vă sugerăm să folosiți limbajul C++) algoritmul EM pentru a clusteriza instanțele din mulțimea X în K clustere, folosind o mixtură de distribuții gaussiene de forma $\sum_{k=1}^K \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$.

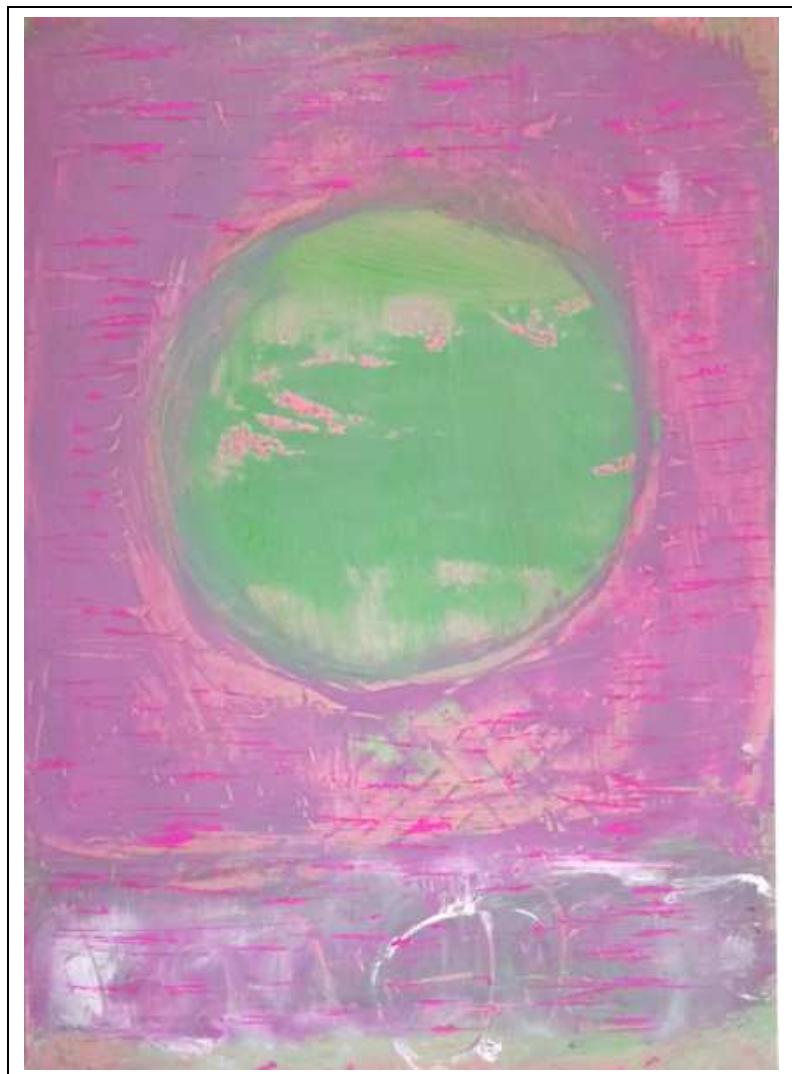
Cerințe:

1. Veți implementa câte o funcție / „metodă“ pentru fiecare componentă a algoritmului EM: pasul de initializare, pasul E, pasul M și condiția de oprire. Adițional, veți implementa și o „metodă“ pentru calculul funcției de verosimilitate a datelor „observabile“ X în raport cu parametrii modelului probabilist asociat. În ce privește condiția de oprire, veți crea posibilitatea ca utilizatorul să poată să aleagă (printr-o opțiune la linia de comandă) între două variante: fie să se efectueze un număr (prestabilit) de iterații, fie să se testeze diferența dintre verosimilitatea datelor X la iterația t și verosimilitatea datelor X la iterația $t - 1$ în raport cu un prag (prestabilit) ε .
2. Veți trata separat — folosind supra-scrierea „metodelor“ precizate la punctul anterior — cazul distribuțiilor unidimensionale ($d = 1$) față de cazul distribuțiilor multidimensionale ($d > 1$). Pentru primul caz veți implementa ambele variante ale algoritmului EM/GMM

care apar în enunțul problemei 17.bc (și veți face testarea pe datele de la problema aceea, precum și pe cele de la problema 16). Pentru a doilea caz veți implementa algoritmul care a fost dedus la problema 20 (pentru testare, folosiți datele de la problemele 21, 22, 23, 24 și 59).

3. Veți implementa — tot prin supra-scrierea „metodelor” — încă două variante ale algoritmului EM/GMM cazul multidimensional, conform problemelor 56 și 58. (Pentru testare, puteți folosi datele de la problemele indicate la punctul precedent.)

Această pagină a fost lăsată liberă în mod intenționat.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

8 Algoritmul EM

Sumar

Noțiuni preliminare

- distribuții probabiliste: vedeti secțiunea *Distribuții probabiliste uzuale* de la capitolul de *Fundamente*;
- estimarea parametrilor distribuțiilor probabiliste în sensul verosimilității maxime (MLE) și respectiv în sensul probabilității maxime a posteriori (MAP): vedeti secțiunea *Estimarea parametrilor unor distribuții probabiliste* de la capitolul de *Fundamente*;
- mixturi de distribuții probabiliste: vedeti ex. 26 și ex. 103 de la capitolul de *Fundamente*, ex. 3.AB de la prezentul capitol;⁵³⁸
- metoda maximizării alternante pe coordonate (engl., *coordinate ascent*) pentru rezolvarea unor probleme de optimizare: ex. 1;⁵³⁹
- metoda multiplicatorilor lui Lagrange pentru rezolvarea unor probleme de optimizare cu restricții: vedeti secțiunea *Metode de optimizare în învățarea automată* de la capitolul de *Fundamente*, precum și ex. 7, ex. 8, ex. 9, ex. 10, ex. 17 și ex. 20 de la prezentul capitol.

Schema algoritmică EM

- pseudo-cod: *Machine Learning*, Tom Mitchell, 1997, pag. 194-195;
- fundamentare teoretică:
 - ex. 1: pentru funcția de log-verosimilitate a datelor complete, există o margine inferioară, $F(q, \theta)$; algoritmul EM face maximizarea acestei margini inferioare aplicând metoda [iterativă] creșterii alternative pe coordonate (engl., coordinate ascent);
 - ex. 2: monotonia valorilor funcției de log-verosimilitate a datelor complete, care sunt calculate la iterării succesive ale lui EM.⁵⁴⁰

EM pentru modelarea de mixturi de distribuții probabiliste

- mixturi de distribuții *Bernoulli*: ex. 3 și ex. 4;
- mixturi de distribuții *binomiale*: ex. 5;
- mixturi de distribuții *categoriale*: ex. 8 și ex. 17;
 - algoritmul *Bayes Naiv nesupervizat*, i.e. algoritmul EM pentru rezolvare de mixturi de distribuții *categoriale multidimensionale*, cu presupunerea de independentă condițională a atributelor de intrare în raport cu atributul de ieșire (eticheta): ex. 9 pentru varianta de *asignare “soft”* a instanțelor la clustere și, respectiv, ex. 22 pentru varianta *asignării “hard”*;

⁵³⁸Pentru mixturi de distribuții gaussiene vedeti secțiunea *Algoritmul EM pentru modele de mixturi gaussiene* de la capitolul de *Clusterizare*.

⁵³⁹Vedeți, de asemenea, utilizarea aceleiași metode de optimizare (eventual pentru minimizare în loc de maximizare) în cazul altor algoritmi de învățare automată: pentru algoritmul AdaBoost, ex. 22, ex. 28 și ex. 29 de la capitolul de *Arbore de decizie*; pentru algoritmul *K-means*, ex. 12 de la capitolul de *Clusterizare*; în sfârșit, pentru algoritmul SMO, ex. 22 de la capitolul de *Mașini cu vectori-suport*.

⁵⁴⁰Acestea, plus alte câteva *proprietăți generale* ale schemei algoritmice EM au fost deja sintetizate sub forma proprietăților (P0)-(P3) din secțiunea *Algoritmul EM pentru clusterizare* de la *Sumarul* capitolului de *Clusterizare*.

- o *aplicație*: EM pentru *modelul mixturii domeniilor semantice* (engl., topic model) pentru clusterizare de *documente*: ex. 10 [și ex. 19];
mixturi de distribuții *Poisson*: ex. 20;
mixturi de distribuții *Gamma*: ex. 21;
- EM pentru *estimarea probabilității de selecție* a unei componente din cadrul unei mixturi (i.e., combinație liniară) de două distribuții probabiliste oarecare: ex. 14.

EM pentru estimarea parametrilor unor distribuții probabiliste

- EM pentru estimarea unui parametru [cu ajutorul căruia se definește funcția masă de probabilitate] pentru o distribuție *categorială*: ex. 6;
similar, pentru o distribuție *multinomială*: ex. 18;
- EM pentru estimarea tuturor parametrilor unei distribuții *categoriale*: ex. 7.

EM pentru estimarea parametrilor unei distribuții, atunci când o parte din date lipesc

- cazul distribuției *Poisson*: ex. 11.

EM pentru estimarea parametrilor unei sume de două distribuții⁵⁴¹

- cazul distribuțiilor *exponențiale*: ex. 12;
- cazul distribuțiilor *gaussiane*: ex. 13.

Alte instanțe / variații ale schemei algoritmice EM

- “hard” EM (algoritmul EM cu asignare “hard” a instanțelor la clustere): ex. 22;
- EM pentru *estimarea* nu în sens MLE, cum a fost cazul până aici, ci *în sens MAP*: ex. 15.

Alte probleme

- chestiuni metodologice (relativ la inițializarea parametrilor): ex. 23;
- probleme recapitulative (A/F): ex. 16 și ex. 24.

⁵⁴¹Exlicăție: ne referim aici la aplicarea algoritmului EM pentru estimarea parametrilor a două distribuții probabiliste atunci când se dau instanțe care sunt generate prin însumarea unor perechi de valori generate de cele două distribuții;

8.1 Probleme rezolvate

1.

(Algoritmul EM, fundamentare teoretică:
pasul E [și pasul M])

prelucrare de Liviu Ciortuz, după

■ CMU, 2008 fall, Eric Xing, HW4, pr. 1.1-3

Algoritmul EM (Expectation-Maximization) este unul dintre cele mai importante proce-
dee din învățarea automată. El permite crearea unor modele probabiliste care pe de o
parte depind de un set de parametri θ iar pe de altă parte includ pe lângă variabilele obiș-
nuite („observabile“ sau „vizibile“) x și variabile necunoscute („neobservabile“, „ascunse“
sau „latente“) z .⁵⁴² În general, în astfel de situații / modele, nu se poate face în mod
direct o estimare a parametrilor modelului (θ), în aşa fel încât să se garanteze obținerea
maximului verosimilității datelor observabile x .⁵⁴³ În schimb, algoritmul EM procedează
în manieră iterativă, constituind astfel o modalitate foarte convenabilă de estimare a
parametrilor θ .

Definim log-verosimilitatea datelor *complete* (observabile, x , și neobservabile, z) ca fiind
 $\log P(x, z | \theta)$, iar log-verosimilitatea datelor observabile ca fiind $\log P(x | \theta)$.

Observație importantă: Pe tot parcursul acestui exercițiu, ba chiar pentru întreg capitolul,
se va considera în mod implicit funcția log ca având baza supraunitară, fixată.⁵⁴⁴

a. Log-verosimilitatea datelor *observabile* (x) se poate exprima în funcție de datele neob-
servabile (z), astfel:⁵⁴⁵

$$\ell(\theta) \stackrel{not.}{=} \log P(x | \theta) = \log \left(\sum_z P(x, z | \theta) \right)$$

În continuare vom nota cu q o funcție / distribuție de probabilitate definită peste varia-
bile ascunse / neobservabile z .

⁵⁴²Un astfel de model sunt *mixturile de distribuții probabiliste*. O introducere în acest subiect
a fost făcută — pentru mixturi de distribuții Bernoulli — la *Comentariul important* de la pro-
blema 39. Rezolvarea mixturilor de distribuții gaussiene folosind algoritmul EM a constituit
subiectul unei întregi secțiuni din capitolul de *Clusterizare*.

⁵⁴³Atunci când funcția de log-verosimilitate este derivabilă și toate variabilele sunt cunoscute
/ observabile, se poate calcula maximul acestei funcții fie în mod direct, calculând rădăcinile
derivativele parțiale, fie aplicând o metodă de aproximare, aşa cum este metoda gradientului. În
cazul în care o parte din variabile sunt necunoscute / neobservabile, afarea soluțiilor derivativele
parțiale ale funcției de log-verosimilitate necesită adeseori aplicarea unor metode de calcul numeric
sofisticate. Algoritmul EM, care este foarte ușor de implementat, ne oferă o cale alternativă, foarte
utilă în astfel de cazuri.

⁵⁴⁴În secțiunea *Estimarea parametrilor unor distribuții probabiliste* de la capitolul de *Fundamente*, precum și în secțiunea *Algoritmul EM pentru modele de mixturi gaussiene* de la capitolul de *Clusterizare*, am folosit ca bază a logaritmului numărul e . Întrucât și acolo și aici ne interesează
în general doar(!) să obținem maximul unor funcții de log-verosimilitate, alegerea bazei nu con-
tează (atât timp cât este un număr supraunitar). Totuși, va trebui să observați că la punctul b și
 c al acestei probleme, precum și la problema 2, intervenă calcule atât verosimilități cât și notiuni
legate de teoria informației (entropii, cross-entropii și entropii relative). La secțiunea *Elemente*
de teoria informației de la capitolul de *Fundamente*, am folosit ca bază a logaritmului numărul
2. Așadar, în astfel de situații va trebui să avem grijă cum „armonizăm“ între ele enunțurile și
relațiile.

⁵⁴⁵Cititorul trebuie să rețină că x , vectorul de date observabile, este fixat (dat), în vreme ce z ,
vectorul de date neobservabile, este liber (variabil).

Folosiți *inegalitatea lui Jensen*⁵⁴⁶ pentru a demonstra că are loc următoarea inegalitate:

$$\log P(x | \theta) \geq \sum_z q(z) \log \left(\frac{P(x, z | \theta)}{q(z)} \right)$$

pentru orice set de date observabile x (fixat), pentru orice valoare a parametrului θ și pentru orice distribuție probabilistă q definită peste variabilele neobservabile z .

Observație (1): Semnificația acestei inegalități este următoarea:

Funcția

$$F(q, \theta) \stackrel{\text{def.}}{=} \sum_z q(z) \log \left(\frac{P(x, z | \theta)}{q(z)} \right) \quad (187)$$

constituie o margine inferioară pentru funcția de log-verosimilitate a datelor observabile / incomplete, $\ell(\theta) \stackrel{\text{not.}}{=} \log P(x | \theta)$.⁵⁴⁷ Remarcați faptul că F este o funcție de două variabile, iar prima variabilă nu este de tip numeric (cum este θ), ci este de tip funcțional.⁵⁴⁸ Mai mult, se observă că expresia funcției F este de fapt (similară cu) o medie, $E_{q(z)} \left[\log \frac{P(x, z | \theta)}{q(z)} \right]$, atunci când x, q și parametrul θ se consideră fixați, iar z este lăsat să varieze.

b. Conform definiției divergenței Kullback-Leibler (numită și entropia relativă), care a fost dată la problema 55 de la capitolul de *Fundamente*, putem scrie:⁵⁴⁹

$$KL(q(z) || P(z | x, \theta)) = - \sum_z q(z) \log \left(\frac{P(z | x, \theta)}{q(z)} \right) \quad (188)$$

Arătați că — dacă folosim numărul 2 ca bază a logaritmului — avem:

$$\log P(x | \theta) = F(q(z), \theta) + KL(q(z) || P(z | x, \theta)).$$

Observație (2): Semnificația egalității care trebuie demonstrată la acest punct este foarte interesantă: diferența dintre funcția obiectiv $\ell(\theta) \stackrel{\text{not.}}{=} \log P(x | \theta)$ și marginea sa inferioară $F(q(z), \theta)$ — a se vedea punctul a — este $KL(q(z) || P(z | x, \theta))$. Aceasta este divergența Kullback-Leibler dintre distribuția (arbitră) considerată $q(z)$ și distribuția condițională a variabilei ascunse z în raport cu variabila observabilă x . Tocmai pe această chestiune se va „construi” punctul final, și cel mai important, al problemei noastre. Însă înainte de aceasta, este foarte util să formulăm încă o observație.

Observație (3): Ideile de bază ale algoritmului EM sunt două:

1. În loc să calculeze maximul funcției de log-verosimilitate $\log P(x | \theta)$ în raport cu

⁵⁴⁶Vedeți problema 71 de la capitolul de *Fundamente*, și în special punctul c al acestei probleme.

⁵⁴⁷În inegalitatea $\ell(\theta) \geq F(q, \theta)$, deși poate părea surprinzător, funcția ℓ are un singur argument, în vreme ce funcția F are două argumente. Diversele aspecte (foarte interesante!) ale relației dintre cele două funcții vor fi „aduse la lumină” în cele ce urmează.

⁵⁴⁸În continuare, pentru a aduce mereu aminte cititorului că distribuția q se referă la datele neobservabile z , vom folosi notația $q(z)$ în loc de q . În consecință, în cele ce urmează, în funcție de context, $q(z)$ va desemna fie la distribuția q , fie la valoarea acestei distribuții pentru o valoare oarecare [a variabilei neobservabile] z . (Suntem conștienți de faptul că această lejeră ambiguitate poate induce în eroare cititorul neexperimentat.)

⁵⁴⁹Veți observa că în relația (187) apare $P(x, z | \theta)$, iar în relația (188) apare $P(z | x, \theta)$.

θ , algoritmul EM va maximiza marginea sa inferioară, $F(q(z), \theta)$, în raport cu ambele argumente, $q(z)$ și θ .

2. Pentru a căuta maximul (de fapt, un maxim local al) marginii inferioare $F(q(z), \theta)$, algoritmul EM aplică metoda *creșterii pe coordonate* (engl., coordinate ascent): după ce inițial se fixează $\theta^{(0)}$ eventual aleatoriu, se maximizează *iterativ* funcția $F(q(z), \theta)$, în mod *alternativ*: mai întâi în raport cu distribuția $q(z)$ și apoi în raport cu parametrul θ .

$$\begin{aligned} \text{Pasul E: } q^{(t)}(z) &= \underset{q(z)}{\operatorname{argmax}} F(q(z), \theta^{(t)}) \\ \text{Pasul M: } \theta^{(t+1)} &= \underset{\theta}{\operatorname{argmax}} F(q^{(t)}(z), \theta) \end{aligned}$$

c. Fie $\theta^{(t)}$ valoarea obținută pentru parametrul / parametrii θ la iteratăia t a algoritmului EM. Considerând această valoare fixată, arătați că maximul lui F în raport cu argumentul / distribuția $q(z)$ este atins pentru distribuția $P(z | x, \theta^{(t)})$, iar — dacă baza logaritmului este 2 — valoarea maximului se poate exprima astfel:

$$\max_{q(z)} F(q(z), \theta^{(t)}) = E_{P(z|x, \theta^{(t)})} [\log P(x, z | \theta^{(t)})] + H(P(z | x, \theta^{(t)}))$$

Așadar, atunci când se consideră primul argument al funcției F fixat, valoarea maximă a acestei funcții (F) în raport cu primul său argument, $q(z)$, este suma a doi termeni: media log-verosimilității datelor complete (observabile și neobservabile) în raport cu distribuția posterioară a variabilelor neobservabile z (primul termen) și entropia acestei distribuții posterioare (al doilea termen), care nu depinde de $q(z)$.

Răspuns:

a. În contextul teoriei probabilităților, inegalitatea lui Jensen este exprimată astfel: dacă X este o variabilă aleatoare iar φ este o funcție convexă, atunci $\varphi(E[X]) \leq E[\varphi(X)]$. Dacă φ este funcție concavă, inegalitatea lui Jensen devine $\varphi(E[X]) \geq E[\varphi(X)]$.⁵⁵⁰

În cazul nostru, folosim funcția log cu bază supraunitară, care este o funcție concavă, deci aplicând inegalitatea lui Jensen obținem: $\log(E[X]) \geq E[\log(X)]$.

Log-verosimilitatea datelor observabile este:

$$\begin{aligned} \log P(x | \theta) &= \log \left(\sum_z P(x, z | \theta) \right) = \log \left(\sum_z q(z) \frac{P(x, z | \theta)}{q(z)} \right) \\ &\stackrel{\text{def.}}{=} \log \left(E_{q(z)} \left[\frac{P(x, z | \theta)}{q(z)} \right] \right) \end{aligned}$$

De aici, conform inegalității lui Jensen — concret, înlocuind X din inegalitatea menționată cu $\frac{P(x, z | \theta)}{q(z)}$ —, rezultă:

$$\log P(x | \theta) \geq E_{q(z)} \left[\log \frac{P(x, z | \theta)}{q(z)} \right] \stackrel{\text{def.}}{=} \sum_z q(z) \log \frac{P(x, z | \theta)}{q(z)},$$

adică tocmai ceea ce trebuia să demonstrăm.

⁵⁵⁰Vedeți problema 71.c de la capitolul de *Fundamente*.

b. Pentru a demonstra egalitatea cerută, vom pleca de la definiția dată în enunț pentru funcția $F(q(z), \theta)$. Apoi, în expresia din definiție vom înlocui $P(x, z | \theta)$ cu $P(z | x, \theta) \cdot P(x | \theta)$ — conform formulei de „multiplicare“ a probabilităților — și vom obține:

$$\begin{aligned} F(q(z), \theta) &\stackrel{\text{def.}}{=} \sum_z q(z) \log \left(\frac{P(x, z | \theta)}{q(z)} \right) = \sum_z q(z) \log \left(\frac{P(z | x, \theta) \cdot P(x | \theta)}{q(z)} \right) \\ &= \sum_z q(z) \left[\log \frac{P(z | x, \theta)}{q(z)} + \log P(x | \theta) \right] \\ &= \sum_z q(z) \log \left(\frac{P(z | x, \theta)}{q(z)} \right) + \sum_z q(z) \log P(x | \theta) \\ &= -KL(q(z) || P(z | x, \theta)) + \underbrace{\log P(x | \theta) \cdot \sum_z q(z)}_{=1} \end{aligned}$$

Rezultă că $\log P(x | \theta) = F(q(z), \theta) + KL(q(z) || P(z | x, \theta))$.

Observație (4): Conform proprietății $KL(p || q) \geq 0$ pentru $\forall p, q$, care a fost demonstrată și de noi la exercițiul 55 de la capitolul de *Fundamente*, rezultă că $KL(q(z) || P(z | x, \theta)) \geq 0$. Așadar, din egalitatea care tocmai a fost demonstrată la punctul b obținem (din nou!, după rezultatul de la punctul a) că $F(q(z), \theta)$ este o margine inferioară pentru log-verosimilitatea datelor observabile, $\ell(\theta) \stackrel{\text{not.}}{=} \log P(x | \theta)$.

c. Trebuie să maximizăm $F(q(z), \theta^{(t)})$ — marginea inferioară a log-verosimilității datelor observabile x — în raport cu distribuția $q(z)$.

Pe de o parte, rezultatul de la punctul a ne spune că $F(q(z), \theta) \leq \log P(x | \theta)$, pentru orice valoare a lui θ ; în particular, pentru $\theta^{(t)}$ avem

$$\log P(x | \theta^{(t)}) \geq F(q(z), \theta^{(t)})$$

Pe de altă parte, dacă în egalitatea demonstrată la punctul b se înlocuiește θ cu $\theta^{(t)}$, rezultă:

$$\log P(x | \theta^{(t)}) = F(q(z), \theta^{(t)}) + KL(q(z) || P(z | x, \theta^{(t)}))$$

În fine, dacă alegem $q(z) = P(z | x, \theta^{(t)})$, atunci termenul $KL(q(z) || P(z | x, \theta^{(t)}))$ din dreapta egalității de mai sus devine zero (a se vedea punctul a de la același exercițiu 55 de la capitolul de *Fundamente*). Așadar, valoarea $\max_{q(z)} F(q(z), \theta^{(t)})$ se obține pentru distribuția $q(z) = P(z | x, \theta^{(t)})$.

Acum vom calcula această valoare maximă:

$$\begin{aligned} \log P(x | \theta^{(t)}) &= \max_{q(z)} F(q(z), \theta^{(t)}) = F(P(x | \theta^{(t)}), \theta^{(t)}) \\ &\stackrel{\text{def. } F}{=} \sum_z P(z | x, \theta^{(t)}) \log \left(\frac{P(x, z | \theta^{(t)})}{P(z | x, \theta^{(t)})} \right) \\ &= E_{P(z|x, \theta^{(t)})} \left[\log \frac{P(x, z | \theta^{(t)})}{P(z | x, \theta^{(t)})} \right] \\ &= E_{P(z|x, \theta^{(t)})} \left[\log P(x, z | \theta^{(t)}) - \log P(z | x, \theta^{(t)}) \right] \\ &= E_{P(z|x, \theta^{(t)})} \left[\log P(x, z | \theta^{(t)}) \right] - E_{P(z|x, \theta^{(t)})} \left[\log P(z | x, \theta^{(t)}) \right] \\ &= E_{P(z|x, \theta^{(t)})} \left[\log P(x, z | \theta^{(t)}) \right] + H[P(z | x, \theta^{(t)})] \end{aligned}$$

$$= Q(\theta^{(t)} | \theta^{(t)}) + H[P(z | x, \theta^{(t)})],$$

unde $Q(\theta | \theta^{(t)}) \stackrel{\text{not.}}{=} E_{P(z|x,\theta^{(t)})}[\log P(x, z | \theta)]$. Așadar, am obținut rezultatul care a fost cerut în enunț.

Observație (5): [Fundamentarea pasului M]

Notând $G_t(\theta) \stackrel{\text{def.}}{=} F(P(z | x, \theta^{(t)}), \theta)$, din calculul de mai sus rezultă că $\log P(x | \theta^{(t)}) = G_t(\theta^{(t)}) = Q(\theta^{(t)} | \theta^{(t)}) + H(P(z | x, \theta^{(t)}))$. Se poate demonstra ușor — procedând similar cu calculul de mai sus — egalitatea

$$G_t(\theta) = Q(\theta | \theta^{(t)}) + H[P(z | x, \theta^{(t)})]$$

Observând că termenul $H[P(z | x, \theta^{(t)})]$ din această ultimă egalitate nu depinde de θ , rezultă imediat că

$$\operatorname{argmax}_{\theta} G_t(\theta) = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})$$

În consecință,

$$\theta^{(t+1)} \stackrel{\text{def.}}{=} \operatorname{argmax}_{\theta} \underbrace{F(P(z | x, \theta^{(t)}), \theta)}_{G_t(\theta)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})$$

Ultima egalitate de mai sus este responsabilă pentru următoarea reformulare (cea uzuală!) a corpului iterativ al algoritmului EM:

$$\begin{aligned} \text{Pasul E':} & \quad \text{calculează } Q(\theta | \theta^{(t)}) = E_{P(z|x,\theta^{(t)})}[\log P(x, z | \theta)] \\ \text{Pasul M':} & \quad \text{calculează } \theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)}) \end{aligned}$$

Așadar, pasul E (“expectation”) al algoritmului EM va consta în calcularea funcției auxiliare $Q(\theta | \theta^{(t)})$. Pentru calculul expresiei acestei funcții (care este o medie, vedeti definiția ei de mai sus) se folosesc mediile variabilelor neobservabile, $E[z_i | x, \theta^{(t)}]$. Adeseori, în practică, la pasul E al algoritmului EM se face doar(!) calcularea mediilor acestor variabile neobservabile.⁵⁵¹ În continuare, la pasul M al aceluiași algoritm se vor calcula parametrii $\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})$, de obicei ca rădăcini ale derivatelor parțiale ale funcției $Q(\theta | \theta^{(t)})$, atunci când aceste derivate există.

2.

(Algoritmul EM: monotonie / convergență)

■ *prelucrare de Liviu Ciortuz, 2012, după
en.wikipedia.org/wiki/Expectation-maximization*

La problema precedentă (1) am văzut că în loc să urmărească în mod direct maximizarea funcției de log-verosimilitate a datelor observabile, adică $\ell(\theta) \stackrel{\text{def.}}{=} \log P(x | \theta)$, unde

⁵⁵¹În cazul în care $E[z_i | x, \theta^{(t)}]$ reprezintă totodată un parametru al distribuției probabiliste care guvernează datele observabile — de exemplu, factorul de combinare a unor distribuții în cadrul unei mixturi, care este adeseori parametrul unei distribuții de tip Bernoulli —, calculul lui $E[z_i | x, \theta^{(t)}]$ este în sine o procedură de *estimare* a acestui parametru. Alți parametri (de exemplu μ și σ^2 pentru distribuții gaussiene) pot fi determinați în funcție de mediile acestor variabile neobservabile. Așa se explică de ce termenul “Expectation” este folosit de către unii autori în locul termenului “Expectation” (după părerea noastră, cel mai adekvat) în denumirea algoritmului EM.

baza logaritmului (nespecificată) este considerată supraunitară, algoritmul EM procesează în mod iterativ, optimizând la pasul M al fiecărei iterații (t) o funcție „auxiliară“ $Q(\theta | \theta^{(t)}) \stackrel{\text{def.}}{=} E_{P(z|x, \theta^{(t)})}[\log P(x, z | \theta)]$, reprezentând media log-verosimilității datelor complete (observabile și neobservabile) în raport cu distribuția a posteriori $P(z | x, \theta^{(t)})$. Vom considera iterațiile $t = 0, 1, \dots$ și $\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})$, cu $\theta^{(0)}$ ales în mod arbitrar.

În acest exercițiu veți demonstra că pentru orice t fixat (arbitrар) și pentru orice θ astfel încât $Q(\theta | \theta^{(t)}) \geq Q(\theta^{(t)} | \theta^{(t)})$ are loc inegalitatea:

$$\log P(x | \theta) - \log P(x | \theta^{(t)}) \geq Q(\theta | \theta^{(t)}) - Q(\theta^{(t)} | \theta^{(t)}) \quad (189)$$

Observație (1): Această inegalitate are o semnificație pe cât de simplă pe atât de importantă: orice îmbunătățire a valorii funcției $Q(\theta | \theta^{(t)})$ conduce la o îmbunătățire cel puțin la fel de mare a valorii funcției obiectiv, $\ell(\theta) \stackrel{\text{def.}}{=} \log P(x | \theta)$.

Observație (2): Dacă în inegalitatea

(189) se înlocuiește θ cu

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)}),$$

va rezulta

$$\log P(x | \theta^{(t+1)}) \geq \log P(x | \theta^{(t)}).$$

Altfel spus, la fiecare iterație a algoritmului EM, odată cu trecerea de la $\theta^{(t)}$ la $\theta^{(t+1)}$, valoarea funcției de log-verosimilitate a datelor observabile, $\ell(\theta) \stackrel{\text{def.}}{=} \log P(x | \theta)$ crește sau, în cel mai rău caz, rămâne pe loc.

În final, vom avea $\ell(\theta^{(0)}) \leq \ell(\theta^{(t)}) \leq \ell(\theta^{(t+1)}) \leq \dots$. Sirul acesta (monoton) este mărginit superior de 0 (vedeți definiția lui ℓ), deci converge la o anumită valoare ℓ^* . În anumite cazuri / condiții, această valoare este un maxim (în general, local) al funcției de log-verosimilitate.⁵⁵²

Observație (3): Conform același inegalități (189), la pasul M de la iterația t a algoritmului EM, este suficient ca în loc să se ia $\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})$, să se aleagă $\theta^{(t+1)}$ astfel încât $Q(\theta^{(t+1)} | \theta^{(t)}) > Q(\theta^{(t)} | \theta^{(t)})$. Aceasta constituie *versiunea „generalizată“* a algoritmului EM.

Răspuns:

$P(x, z | \theta) = P(z | x, \theta) \cdot P(x | \theta)$, conform regulei de „multiplicare“ a probabilităților. Prin logaritmarea acestei egalități, rezultă:

$$\log P(x | \theta) = \log P(x, z | \theta) - \log P(z | x, \theta)$$

⁵⁵²Vedeți capitolul *The EM Algorithm*, de Shu Kay Ng et al., în *Handbook of Computational Statistics*, Springer, 2004, pag. 139.

Ca să obținem expresia funcției „auxiliare“ Q pe care o optimizează algoritmul EM la pasul M al iterării t , vom înmulți ambii membri ai egalității precedente cu $P(z | x, \theta^{(t)})$ și apoi vom suma după toate valorile posibile ale lui z :

$$\begin{aligned} \sum_z P(z | x, \theta^{(t)}) \cdot \log P(x | \theta) = \\ \sum_z P(z | x, \theta^{(t)}) \cdot \log P(x, z | \theta) - \sum_z P(z | x, \theta^{(t)}) \cdot \log P(z | x, \theta) \end{aligned} \quad (190)$$

Membrul stâng al acestei egalități poate fi rescris astfel:

$$\sum_z P(z | x, \theta^{(t)}) \cdot \log P(x | \theta) = \log P(x | \theta) \cdot \underbrace{\sum_z P(z | x, \theta^{(t)})}_{1} = \log P(x | \theta)$$

În ceea ce privește membrul drept al aceleiași egalități (190), întrucât termenul al doilea, și anume $-\sum_z P(z | x, \theta^{(t)}) \cdot \log P(z | x, \theta)$ reprezintă o cross-entropie (vedeți problema 58 de la capitolul de *Fundamente*), o vom nota cu $CH(\theta | \theta^{(t)})$. Așadar, egalitatea (190) devine:

$$\log P(x | \theta) = Q(\theta | \theta^{(t)}) + CH(\theta | \theta^{(t)}) \quad (191)$$

Această egalitate este valabilă pentru toate valorile posibile ale parametrului θ . În particular pentru $\theta = \theta^{(t)}$, vom avea:

$$\log P(x | \theta^{(t)}) = Q(\theta^{(t)} | \theta^{(t)}) + CH(\theta^{(t)} | \theta^{(t)}) \quad (192)$$

Scăzând membru cu membru ultimele două egalități, obținem:

$$\log P(x | \theta) - \log P(x | \theta^{(t)}) = Q(\theta | \theta^{(t)}) - Q(\theta^{(t)} | \theta^{(t)}) + CH(\theta | \theta^{(t)}) - CH(\theta^{(t)} | \theta^{(t)})$$

Conform inegalității lui Gibbs — a se vedea problema 59 de la capitolul de *Fundamente*⁵⁵³ —, avem $CH(\theta | \theta^{(t)}) \geq CH(\theta^{(t)} | \theta^{(t)})$, deci în final rezultă:

$$\log P(x | \theta) - \log P(x | \theta^{(t)}) \geq Q(\theta | \theta^{(t)}) - Q(\theta^{(t)} | \theta^{(t)})$$

⁵⁵³ Inegalitatea lui Gibbs se poate demonstra mai direct decât s-a procedat la problema 59 de la capitolul de *Fundamente* (unde s-a pornit de la o proprietate a divergenței KL) după cum urmează.

Se pleacă de la inegalitatea lui Jensen în versiune probabilistă: dacă φ este o funcție convexă, iar X este variabilă aleatoare de distribuție p , atunci $E[\varphi(X)] \geq \varphi(E[X])$. (Vedeți problema 71.c de la capitolul de *Fundamente*.)

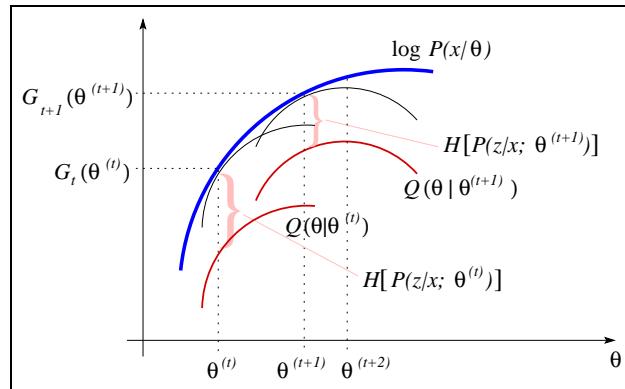
Particularizând această inegalitate pentru $\varphi = -\log_a$, cu $a > 1$, și înlocuind X cu $\frac{q(x)}{p(x)}$ unde q este o distribuție aleatoare arbitrară, vom avea:

$$\begin{aligned} E \left[-\log_a \frac{q(x)}{p(x)} \right] &\geq -\log_a E \left[\frac{q(x)}{p(x)} \right] \Leftrightarrow \\ -\sum_x p(x) \log_a \frac{q(x)}{p(x)} &\geq -\log_a \left(\underbrace{\sum_x p(x) \cdot \frac{q(x)}{p(x)}}_1 \right) \Leftrightarrow \\ -\sum_x p(x) \log_a \frac{q(x)}{p(x)} &\geq 0 \Leftrightarrow -\sum_x p(x) \log_a q(x) \geq -\sum_x p(x) \log_a p(x). \end{aligned}$$

În notațiile / termenii din rezolvarea problemei noastre, această inegalitate se scrie $CH(p||q) \geq CH(p||p) = H(p)$. Folosind notiunea de divergență Kullback-Leibler, inegalitatea aceasta se rescrie sub forma $KL(p || q) \geq 0$.

Aceasta este chiar inegalitatea pe care trebuie să-o demonstrăm.

Observație (4):
Cu ajutorul figurii alăturate, putem să ilustrăm și totodată să sumarizăm în mod grafic rezultatele pe care le-am demonstrat la aceste ultime două probleme (1 și 2).



Observație (5): [Proprietăți ale algoritmului EM⁵⁵⁴]

Algoritmul EM este util datorită următoarelor *avantaje*: simplitate conceptuală, posibilitatea estimării datelor neobservabile, ușurința implementării, precum și faptul că la fiecare iterație se îmbunătățesc valorile parametrului / parametrilor θ . Rata de convergență la primele câteva iterări ale algoritmului EM este în general foarte bună, însă ea poate deveni foarte mică pe măsură ce ne apropiem de punctul de optim local. În general, algoritmul EM dă rezultate bune atunci când proporția datelor neobservabile este mică, iar dimensiunea datelor (adică, numărul de atribute ale instanțelor) nu este prea mare. EM poate necesita executarea multor iterări, iar în cazul dimensiunilor mari pasul E poate deveni foarte lent.

3. (Mixturi de distribuții Bernoulli, cu și fără variabile neobservabile: un *exemplu* de rezolvare, folosind un set simplu de date; deducerea regulilor de actualizare pentru două variante ale algoritmului EM; aplicare)

■ *Liviu Ciortuz, 2020, folosind datele de la CMU, 2005 fall, T. Mitchell, A. Moore, midterm, pr. 1.3*

Comentariu:

[Estimarea parametrului unei *mixturi de distribuții Bernoulli*, în sens MLE]

În secțiunile A și B, care urmează mai jos, vom face o „revizitare“ a problemei 39 de la capitolul de *Fundamente* — ea oferă un exemplu simplu de estimare a unui parametru probabilist în sensul verosimilității maxime —, analizând acum datele *în mod explicit* din perspectiva unei *mixturi de distribuții Bernoulli*,⁵⁵⁵ și anume

$$X \sim \pi \text{Bernoulli}(\theta) + (1 - \pi) \text{Bernoulli}(2\theta),$$

⁵⁵⁴Cf. documentului *The EM Algorithm*, Ajit Singh, CMU, November 20, 2005. Vedeți de asemenea secțiunea 5.2.5 din capitolul *The EM Algorithm*, de Shu Kay Ng et al., din volumul *Handbook of Computational Statistics*, Springer, 2004.

⁵⁵⁵Cititorul a avut ocazia să facă un prim contact cu noțiunea de *mixtură de distribuții* [categoriale] la problema 26 de la capitolul de *Fundamente*.

sau, altfel spus,⁵⁵⁶

$$(X|Z=1) \sim Bernoulli(\theta) \text{ și } (X|Z=0) \sim Bernoulli(2\theta), \text{ unde } Z \sim Bernoulli(\pi),$$

cu $\pi \in (0, 1)$ fixat. Este util să precizăm (și să comentăm) următoarele două situații posibile, în funcție de cum este variabila Z , *observabilă* sau *neobservabilă*.

A. [MLE, fără date neobservabile]

Considerând variabila Z *observabilă*, putem scrie nu doar una ci două funcții de log-verosimilitate și apoi putem (de fapt, în consecință, trebuie) să arătăm care este relația dintre aceste două funcții.

	Moneda	Rezultat
i	Z_i	X_i
1	1	1 (stemă)
2	0	0 (ban)
3	0	0 (ban)
4	0	0 (ban)
5	0	1 (stemă)

Funcția de log-verosimilitate a datelor comune („complete“) este următoarea:

$$\begin{aligned} \ell_{compl}(\theta) &\stackrel{not.}{=} \ln P(X, Z|\theta) \stackrel{i.i.d.}{=} \ln \prod_{i=1}^5 P(X_i, Z_i|\theta) = \\ &= \ln \prod_{i=1}^5 P(X_i|Z_i, \theta) \cdot P(Z_i|\theta) = \sum_{i=1}^5 [\ln P(X_i|Z_i, \theta) + \ln P(Z_i|\theta)] = \\ &= \ln P(X_1 = 1|Z_1 = 1, \theta) + \ln P(Z_1 = 1|\theta) + \ln P(X_2 = 0|Z_2 = 0, \theta) + \ln P(Z_2 = 0|\theta) + \\ &+ \ln P(X_3 = 0|Z_3 = 0, \theta) + \ln P(Z_3 = 0|\theta) + \ln P(X_4 = 0|Z_4 = 0, \theta) + \ln P(Z_4 = 0|\theta) + \\ &+ \ln P(X_5 = 1|Z_5 = 0, \theta) + \ln P(Z_5 = 0|\theta) = \\ &= \ln \theta + \ln \pi + 3(\ln(1 - 2\theta) + \ln(1 - \pi)) + \ln(2\theta) + \ln(1 - \pi) = \\ &= \ln \theta + 3 \ln(1 - 2\theta) + \ln(2\theta) + \ln \pi + 4 \ln(1 - \pi). \end{aligned}$$

Funcția de log-verosimilitate a datelor condiționate este:

$$\ell_{cond}(\theta) \stackrel{not.}{=} \ln P(X|Z, \theta) \stackrel{i.i.d.}{=} \ln \prod_{i=1}^5 P(X_i|Z_i, \theta) = \ln \theta + 3 \ln(1 - 2\theta) + \ln(2\theta),$$

identică cu funcția de log-verosimilitate care a fost calculată la punctul a .

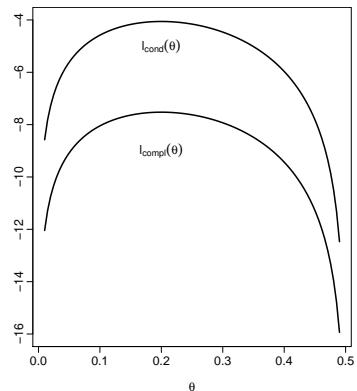
Presupunând că π este fixat (și nu depinde de θ), rezultă imediat că

$$\operatorname{argmax}_{\theta \in (0, 1/2)} \ell_{compl}(\theta) = \operatorname{argmax}_{\theta \in (0, 1/2)} \ell_{cond}(\theta).$$

pentru că $\ell_{compl}(\theta) = \ell_{cond}(\theta) + const.$

Prin urmare, în condițiile presupunerii de mai sus, $\hat{\theta}_{MLE}$, care este estimarea de verosimilitate maximă a parametrului θ poate fi calculat folosind oricare dintre cele două funcții de log-verosimilitate indicate mai sus.

Graficul alăturat a fost realizat (de către Sebastian Ciobanu) pentru $\pi = 1/2$.



⁵⁵⁶Vezi observă că, pentru a obține o formă mai convenabilă la calculele care urmează, moneda a două a fost asociată / desemnată cu $Z = 0$.

B. [MLE, în manieră analitică, în prezența unor date neobservabile]

	Moneda	Rezultat
i	Z_i	X_i
1	?	1 (stema)
2	?	0 (ban)
3	?	0 (ban)
4	?	0 (ban)
5	?	1 (stema)

În cazul când variabila Z este *neobservabilă* (sau *latentă*, sau încă, *ascunsă*), vom opera — mai mult, veți constata la secțiunea C că și algoritmul EM operează — cu log-verosimilitatea datelor „complete“ și cu log-verosimilitatea datelor „incomplete“ (sau, „observabile“):

Pentru prima dintre acestea, deși definiția rămâne aceeași (ca mai sus), exprimarea concretă diferă, întrucât valorile variabilelor Z_i nu mai sunt observabile.

$$\begin{aligned}
 \ell_{compl}(\theta) &\stackrel{not.}{=} \ln P(X = (x_1 \dots, x_5), Z = (z_1, \dots, z_5) | \theta) \stackrel{i.i.d.}{=} \ln \prod_{i=1}^5 P(X_i = x_i, Z_i = z_i | \theta) \\
 &= \ln \prod_{i=1}^5 P(X_i = x_i, Z_i = z_i | \theta) = \sum_{i=1}^5 \ln [P(X_i = x_i | Z_i = z_i, \theta) \cdot P(Z_i = z_i | \theta)] \\
 &= \ln((\theta\pi)^{z_1}(2\theta(1-\pi))^{1-z_1}) + \ln(((1-\theta)\pi)^{z_2}((1-2\theta)(1-\pi))^{1-z_2}) + \\
 &\quad \ln(((1-\theta)\pi)^{z_3}((1-2\theta)(1-\pi))^{1-z_3}) + \ln(((1-\theta)\pi)^{z_4}((1-2\theta)(1-\pi))^{1-z_4}) + \\
 &\quad \ln((\theta\pi)^{z_5}(2\theta(1-\pi))^{1-z_5}) \\
 &= z_1 \ln(\theta\pi) + (1-z_1) \ln(2\theta(1-\pi)) + z_2 \ln((1-\theta)\pi) + (1-z_2) \ln((1-2\theta)(1-\pi)) + \\
 &\quad z_3 \ln((1-\theta)\pi) + (1-z_3) \ln((1-2\theta)(1-\pi)) + \\
 &\quad z_4 \ln((1-\theta)\pi) + (1-z_4) \ln((1-2\theta)(1-\pi)) + \\
 &\quad z_5 \ln(\theta\pi) + (1-z_5) \ln(2\theta(1-\pi)). \tag{193}
 \end{aligned}$$

Pe lângă faptul că această scriere a funcției de log-verosimilitate a datelor „complete“ este *generală* în raport cu diferite valori posibile ale variabilelor Z_i , observăm că ea mai are un avantaj: ea ne permite să scriem într-un mod foarte convenabil media funcției de log-verosimilitate a datelor „complete“, aplicând proprietatea de liniaritate a mediilor, adică pur și simplu înlocuind fiecare z_i din ultima expresie de mai sus cu $E[Z_i]$.⁵⁵⁷

În ce privește log-verosimilitatea datelor „incomplete“ (sau, „observabile“), atunci când variabilele Z_i sunt *neobservabile*, putem scrie:

$$\ell_{obs}(\theta) \stackrel{def.}{=} \ln P(X | \theta) = \ln \sum_Z P(X, Z | \theta),$$

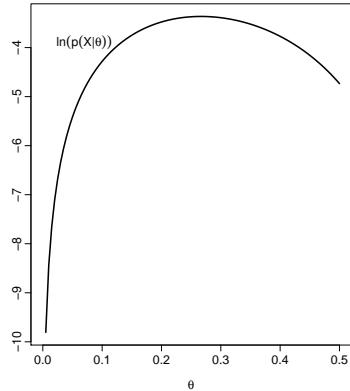
înțelegând prin simbolul \sum_Z că se însumează probabilitățile $P(X, Z | \theta)$ obținute pentru toate asignările posibile ale variabilelor Z . Concret, pentru exemplul de mai sus (când variabilele Z_i sunt neobservabile), log-verosimilitatea datelor observabile se calculează astfel:

$$\begin{aligned}
 \ell_{obs}(\theta) &\stackrel{def.}{=} \ln P(X | \theta) \stackrel{i.i.d.}{=} \ln \prod_{i=1}^5 P(X_i | \theta) = \sum_{i=1}^5 \ln P(X_i | \theta) = \sum_{i=1}^5 \ln \left(\sum_{Z_i \in \{0,1\}} P(X_i, Z_i | \theta) \right) \\
 &= \sum_{i=1}^5 \ln \left(\sum_{Z_i \in \{0,1\}} P(X_i | Z_i, \theta) \cdot P(Z_i | \theta) \right) \\
 &= 2 \ln(\pi\theta + (1-\pi)2\theta) + 3 \ln(\pi(1-\theta) + (1-\pi)(1-2\theta)).
 \end{aligned}$$

⁵⁵⁷Vă vom preciza / indica la momentul corespunzător care anume este distribuția de probabilitate în raport cu care vom calcula această medie.

În figura alăturată este prezentat graficul funcției $\ell'_{obs}(\theta)$, realizat de către Sebastian Ciobanu. Valoarea parametrului θ pentru care se atinge minimul acestei funcții se obține rezolvând ecuația $\ell'_{obs}(\theta) = 0$, care depinde de valoarea care a fost atribuită lui π . De exemplu, pentru $\pi = 1/2$, vom avea:

$$\begin{aligned}\ell'_{obs}(\theta) = 0 &\Leftrightarrow \frac{2}{\theta} = \frac{9}{2 - 3\theta} \Leftrightarrow 4 - 6\theta = 9\theta \Leftrightarrow \\ \theta &= \frac{9}{15} = 0.2(6).\end{aligned}$$



Se observă că această valoare aparține intervalului $(0, 1/2)$. Similar, pentru $\pi = 1/5$, care este chiar probabilitatea (estimată în sensul verosimilității maxime) de alegere a monedei 1 în experimentul aleator din prima parte a acestei probleme, se obține soluția $\theta = 2/9$. Remarcăm faptul că ea *diferă!* ușor de $\hat{\theta}_{MLE} = 1/5$, estimarea de verosimilitate maximă care a fost obținută atunci când s-a considerat că variabilele Z_i sunt observabile.

C. [MLE, în prezența unor date neobservabile, folosind algoritmul EM]

Elaborați regulile de actualizare pentru pașii E și M ai algoritmului EM și apoi aplicați-l la rezolvarea unei de mixturi de distribuții Bernoulli,⁵⁵⁸ conform datelor din secțiunea B.

Vă cerem să elaborați *două variante* ale acestui algoritm EM:

- La acest punct veți considera că probabilitatea de selecție π este un număr fixat (în intervalul $(0, 1/2)$; aşadar, în acest caz singurul parametru (liber) al algoritmului EM este θ . Vă cerem să executați primele două iterații ale acestui algoritm, pentru cazul când $\pi = 1/2$, atribuind lui θ valoarea inițială $\theta^{(0)} = 0.45$. Veți reprezenta grafic (folosind în acest scop R, Matlab sau limbajul dumneavoastră preferat) atât funcția de log-verosimilitate a datelor observabile, $\ln P(x_1, \dots, x_5|\theta)$, cât și funcțiile auxiliare $Q(\theta|\theta^{(0)})$ și $Q(\theta|\theta^{(1)})$. Marcați pe grafic punctele de maxim ale funcțiilor auxiliare.
- De adăsta aceasta veți considera că probabilitatea de selecție π este — și ea — parametru liber (de asemenea în intervalul $(0, 1/2)$; aşadar, în acest caz, parametrii algoritmului EM sunt θ și π .

Răspuns:

Conform enunțului, vom lucra cu mixtura de distribuții Bernoulli

$$X \sim \pi Bernoulli(\theta) + (1 - \pi) Bernoulli(2\theta), \text{ cu } \theta \in (0, 1/2).$$

Altfel spus, considerând $Z \sim Bernoulli(\pi)$ cu $\pi \in (0, 1)$, vom lucra cu $X|(Z = 1) \sim Bernoulli(\theta)$ și $X|(Z = 0) \sim Bernoulli(2\theta)$

- La acest punct *probabilitatea de selecție π* este considerată *fixată*.

La *pasul E* calculăm mai întâi media variabilelor neobservabile z_i în raport cu datele observabile x_i și cu valoarea curentă a parametrului θ , adică $\theta^{(t)}$.

⁵⁵⁸Pentru o prezentare generală a acestui algoritm vedeti problema 4.

$$E[z_i|x_i, \theta^{(t)}] = P(z_i = 1|x_i, \theta^{(t)})$$

$$\stackrel{F. Bayes}{=} \frac{\pi(\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i}}{\pi(\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i} + (1 - \pi)(2\theta^{(t)})^{x_i} (1 - 2\theta^{(t)})^{1-x_i}} \stackrel{not.}{=} \mu_i^{(t)} \quad (194)$$

Apoi calculăm log-verosimilitatea datelor complete $\ell(\theta)$.

$$\begin{aligned} \ell_{compl}(\theta) &\stackrel{not.}{=} \ln P(X, Z|\theta) \\ &= Z_1 \ln(\theta\pi) + (1 - E[Z_1]) \ln(2\theta(1 - \pi)) + \\ &\quad Z_2 \ln((1 - \theta)\pi) + (1 - E[Z_2]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad Z_3 \ln((1 - \theta)\pi) + (1 - E[Z_3]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad Z_4 \ln((1 - \theta)\pi) + (1 - E[Z_4]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad Z_5 \ln(\theta\pi) + (1 - E[Z_5]) \ln(2\theta(1 - \pi)) \end{aligned}$$

În sfârșit, vom calcula funcția auxiliară $Q(\theta|\theta^{(t)})$, folosind relația (193) și proprietatea de liniaritate a mediei.

$$\begin{aligned} Q(\theta|\theta^{(t)}) &\stackrel{not.}{=} E[\ell_{compl}(\theta)] \\ &= E[Z_1] \ln(\theta\pi) + (1 - E[Z_1]) \ln(2\theta(1 - \pi)) + \\ &\quad E[Z_2] \ln((1 - \theta)\pi) + (1 - E[Z_2]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_3] \ln((1 - \theta)\pi) + (1 - E[Z_3]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_4] \ln((1 - \theta)\pi) + (1 - E[Z_4]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_5] \ln(\theta\pi) + (1 - E[Z_5]) \ln(2\theta(1 - \pi)) \quad (195) \\ &\stackrel{\pi}{=} const. + E[Z_1] \ln \theta + (1 - E[Z_1]) \ln \theta + \\ &\quad E[Z_2] \ln(1 - \theta) + (1 - E[Z_2]) \ln(1 - 2\theta) + \\ &\quad E[Z_3] \ln(1 - \theta) + (1 - E[Z_3]) \ln(1 - 2\theta) + \\ &\quad E[Z_4] \ln(1 - \theta) + (1 - E[Z_4]) \ln(1 - 2\theta) + \\ &\quad E[Z_5] \ln \theta + (1 - E[Z_5]) \ln \theta \\ &= const. + 2 \ln \theta + \\ &\quad (E[Z_2] + E[Z_3] + E[Z_4]) \ln(1 - \theta) + (3 - (E[Z_2] + E[Z_3] + E[Z_4])) \ln(1 - 2\theta). \end{aligned}$$

La pasul M , trebuie să calculăm noua valoare a parametrului θ :

$$\theta^{(t+1)} \stackrel{def.}{=} \underset{\theta \in (0, 1/2)}{\operatorname{argmax}} Q(\theta|\theta^{(t)}).$$

Pentru aceasta, vom calcula derivata funcției $Q(\theta|\theta^{(t)})$ în raport cu θ și o vom egala cu 0:

$$\begin{aligned} \frac{\partial}{\partial \theta} Q(\theta|\theta^{(t)}) = 0 &\Leftrightarrow \frac{2}{\theta} = \frac{E[Z_2] + E[Z_3] + E[Z_4]}{1 - \theta} + \frac{2[3 - (E[Z_2] + E[Z_3] + E[Z_4])]}{1 - 2\theta} \\ &\Leftrightarrow 2(1 - \theta)(1 - 2\theta) = (E[Z_2] + E[Z_3] + E[Z_4])\theta(1 - 2\theta) + \\ &\quad 2(3 - (E[Z_2] + E[Z_3] + E[Z_4]))\theta(1 - \theta) \\ &\Leftrightarrow 2(1 - \theta)(1 - 2\theta) = (\mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)})\theta(1 - 2\theta) + 2(3 - (\mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)}))\theta(1 - \theta) \\ &\Leftrightarrow a\theta^2 + b\theta + c = 0, \text{ unde } a = 10, b = -12 + \mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)}, c = 2 \in \mathbb{R}. \quad (196) \end{aligned}$$

Remarcați faptul că

$$\frac{\partial^2}{\partial \theta^2} Q(\theta|\theta^{(t)}) = -\frac{2}{\theta^2} - \underbrace{\frac{\mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)}}{(1-\theta)^2}}_{\geq 0} - \underbrace{\frac{4(3 - (\mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)}))}{(1-2\theta)^2}}_{\geq 0} > 0.$$

Așadar, $Q(\theta|\theta^{(t)})$ este o funcție strict concavă și, prin urmare, ea admite un punct unic de maxim pe intervalul $(0, 1/2)$.

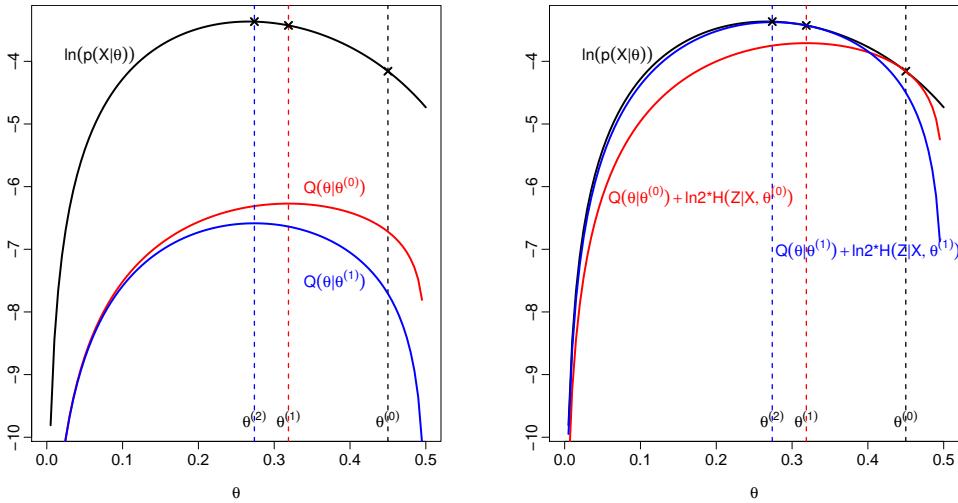
În cazul în care $\Delta \stackrel{\text{def.}}{=} b^2 - 4ac \geq 0$, îi vom atribui lui $\theta^{(t+1)}$ oricare dintre valorile $\theta_{1,2} \stackrel{\text{not.}}{=} \frac{-b \pm \sqrt{\Delta}}{2a}$, cu condiția ca respectiva valoare să aparțină intervalului $(0, 1/2)$. Altfel, adică atunci când $\Delta \geq 0$ și $\theta_{1,2} \notin (0, 1/2)$ sau când $\Delta < 0$, $\theta^{(t+1)}$ va fi luat $\operatorname{argmax}_{\theta \in \{0, 1/2\}} Q(\theta|\theta^{(t)})$.⁵⁵⁹

Sintetizând, putem scrie:

$$\theta^{(t+1)} = \begin{cases} \theta_1 \text{ sau } \theta_2, & \text{dacă } \theta \geq 0 \text{ și } \theta_1 \text{ sau } \theta_2 \in (0, 1/2); \\ \operatorname{argmax}_{\theta \in (0, 1/2)} Q(\theta|\theta^{(t)}), & \text{în cazul contrar.} \end{cases} \quad (197)$$

Regulile de actualizare pentru această versiune a algoritmului EM sunt deci relațiile (194) și (197).

Graficul cerut în enunț — pentru primele două iterări ale acestui algoritm, pentru cazul când $\pi = 1/2$ și atribuind lui θ valoarea inițială $\theta^{(0)} = 0.45$ — este redat mai jos, în partea stângă. În graficul din partea dreaptă am adăugat la [fiecare dintre] funcțiile auxiliare entropia distribuției a posteriori a variabilelor neobservabile, conform proprietății (192) din problema 2.⁵⁶⁰



⁵⁵⁹ Altfel sus, dacă $-\frac{b}{2a} < 0$, atunci $\theta^{(t+1)} = 1/2$, iar în caz contrar (adică, atunci când $-\frac{b}{2a} > 1/2$) se ia $\theta^{(t+1)} = 0$.

⁵⁶⁰ Ambele grafice — ca și implementarea algoritmului EM — au fost realizate de către Sebastian Ciobanu.

Menționăm că am obținut $\theta^{(1)} = 0.3187987$ și $\theta^{(2)} = 0.2738492$, iar valorile log-verosimilității datelor observabile ℓ pentru aceste valori ale lui θ sunt -3.426862 și respectiv -3.366261 (pentru $\theta^{(0)}$, ea a fost -4.157875).

b. Așa cum s-a specificat în enunț, aici vom elabora — în vederea aplicării pe datele noastre — varianta de algoritm EM pentru rezolvarea unei mixturi de două distribuții Bernoulli pentru cazul când parametrul π , care reprezintă *probabilitatea de selecție* (a primei componente dintre cele două ale mixturii), este lăsat liber în intervalul $(0, 1)$. Similar cu modul în care am procedat la punctul a , vom elabora calculele necesare pentru obținerea *formulelor de actualizare* care vor fi folosite la cei doi pași ai algoritmului EM, în noile condiții.

Pasul E:

Calculăm media variabilelor neobservabile z_i în raport cu x_i și $\theta^{(t)}$.

$$E[z_i|x_i, \theta^{(t)}, \pi^{(t)}] = P(z_i = 1|x_i, \theta^{(t)}, \pi^{(t)}) \quad (198)$$

$$\stackrel{F. Bayes}{=} \frac{\pi^{(t)} (\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i}}{\pi^{(t)} (\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i} + (1 - \pi^{(t)}) (2\theta^{(t)})^{x_i} (1 - 2\theta^{(t)})^{1-x_i}} \stackrel{not.}{=} \mu_i^{(t)} \quad (199)$$

și apoi funcția auxiliară $Q(\theta, \pi|\theta^{(t)}, \pi^{(t)})$, folosind relația (195) și proprietatea de liniaritate a mediei:

$$\begin{aligned} Q(\theta, \pi|\theta^{(t)}, \pi^{(t)}) &\stackrel{not.}{=} E[\ell_{compl}(\theta, \pi)] \\ &= E[Z_1] \ln(\theta\pi) + (1 - E[Z_1]) \ln(2\theta(1 - \pi)) + \\ &\quad E[Z_2] \ln((1 - \theta)\pi) + (1 - E[Z_2]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_3] \ln((1 - \theta)\pi) + (1 - E[Z_3]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_4] \ln((1 - \theta)\pi) + (1 - E[Z_4]) \ln((1 - 2\theta)(1 - \pi)) + \\ &\quad E[Z_5] \ln(\theta\pi) + (1 - E[Z_5]) \ln(2\theta(1 - \pi)) \end{aligned}$$

Pasul M:

$\theta^{(t+1)}$ va fi calculat folosind aceeași regulă de actualizare ca la punctul a , iar în ce privește $\pi^{(t+1)}$, reamintim că $\pi^{(t+1)} \stackrel{def.}{=} \text{argmax}_{\pi \in (0,1)} Q(\theta, \pi|\theta^{(t)}, \pi^{(t)})$, deci îl vom putea calcula egalând cu 0 derivata parțială a lui $Q(\theta, \pi|\theta^{(t)}, \pi^{(t)})$ în raport cu π .

$$\begin{aligned} \frac{\partial}{\partial \pi} Q(\theta, \pi|\theta^{(t)}, \pi^{(t)}) &= 0 \Leftrightarrow \\ \frac{\partial}{\partial \pi} [const. + (E[Z_1] + E[Z_2] + E[Z_3] + E[Z_4] + E[Z_5]) \ln \pi + \\ &\quad (5 - (E[Z_1] + E[Z_2] + E[Z_3] + E[Z_4] + E[Z_5])) \ln(1 - \pi)] &= 0 \Leftrightarrow \\ \frac{E[Z_1] + E[Z_2] + E[Z_3] + E[Z_4] + E[Z_5]}{\pi} &- \frac{5 - (E[Z_1] + E[Z_2] + E[Z_3] + E[Z_4] + E[Z_5])}{1 - \pi} \\ &\Leftrightarrow \pi = \frac{1}{5} (E[Z_1] + E[Z_2] + E[Z_3] + E[Z_4] + E[Z_5]) \end{aligned}$$

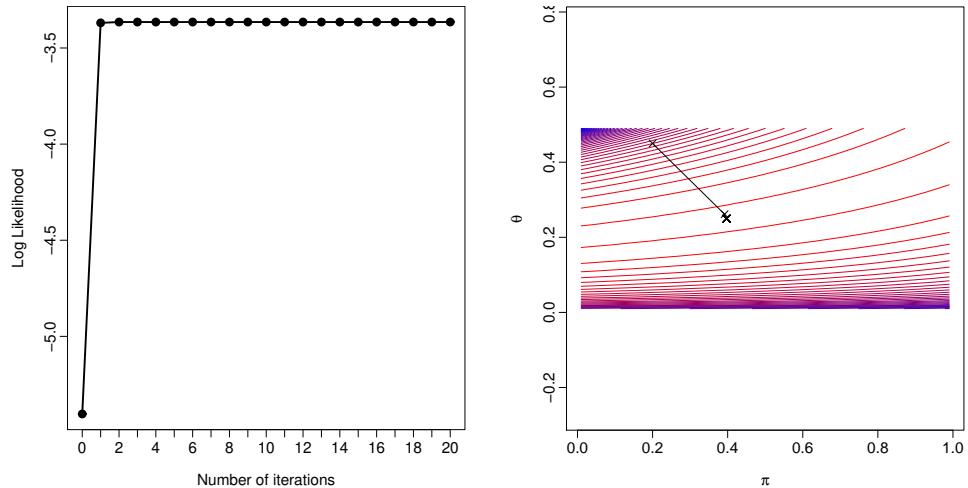
Așadar,

$$\pi^{(t+1)} = \frac{1}{5} (\mu_1^{(t)} + \mu_2^{(t)} + \mu_3^{(t)} + \mu_4^{(t)} + \mu_5^{(t)}) \in [0, 1]. \quad (200)$$

Remarcați faptul că $Q(\theta, \pi | \theta^{(t)}, \pi^{(t)})$ este funcție concavă în raport cu parametrul π .⁵⁶¹

Regulile de actualizare pentru această nouă versiune a algoritmului EM sunt relațiile (199), (195) și (200).

Graficele de mai jos reprezintă evoluția valorilor funcției de log-verosimilitate a datelor observabile în timpul primelor 20 de iterații ale acestei versiuni de algoritm EM, folosind aceleași datele și aceleași inițializări ca la punctul precedent și, în plus, $\pi^{(0)} = 0.2$. În graficul din partea stângă se observă o creștere substanțială a valorilor acestei funcții chiar de la prima iterație, după care, în câteva iterații, se atinge un platou. Graficul din partea dreapta folosește curbe de izocontur pentru a arăta evoluția parametrilor θ și π în cursul execuției algoritmului EM.⁵⁶²



Observație importantă — pentru cazul când π este fixat

Datorită proprietății de *liniaritate a mediei*, urmează că media sumei unor variabile aleatoare este suma mediilor lor. Folosind această proprietate simplă vom arăta acum că putem să reformulăm algoritmul nostru EM pentru estimarea parametrilor unei mixturi de distribuții Bernoulli sub o *altă formă*, foarte convenabilă.

La rezolvarea dată la punctul *a* pentru *pasul E* (așadar, când probabilitatea de selecție, π , era considerată fixată), am obținut

$$\mu_i^{(t)} = P(z_i = 1 | x_i, \theta^{(t)}) \stackrel{F. Bayes}{=} \frac{\pi (\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i}}{\pi (\theta^{(t)})^{x_i} (1 - \theta^{(t)})^{1-x_i} + (1 - \pi) (2\theta^{(t)})^{x_i} (1 - 2\theta^{(t)})^{1-x_i}}.$$

Putem scrie

pentru *cazul* $x_i = 1$ (adică, pentru $x_i \equiv$ *stema*):

$$q_1^{(t)} \stackrel{not.}{=} P(z_i = 1 | x_i = 1, \theta^{(t)}) = \frac{\pi \theta^{(t)}}{\pi \theta^{(t)} + (1 - \pi) 2\theta^{(t)}}, \quad (201)$$

⁵⁶¹De fapt, mai mult, este ușor de arătat că matricea hessiană a lui $Q(\theta, \pi | \theta^{(t)}, \pi^{(t)})$ este negativ definită, deci Q admite un singur punct de maximum.

⁵⁶²Ambele grafice — ca și implementarea algoritmului EM — au fost realizate de către Sebastian Ciobanu.

iar pentru *cazul* $x_i = 0$ (adică, pentru $x_i \equiv \text{ban}$):

$$q_0^{(t)} \stackrel{\text{not.}}{=} P(z_i = 1 | x_i = 0, \theta^{(t)}) = \frac{\pi (1 - \theta^{(t)})}{\pi (1 - \theta^{(t)}) + (1 - \pi) (1 - 2\theta^{(t)})}. \quad (202)$$

Fie z_1 numărul (neobservabil) de apariții ale feței *stemă* pentru moneda 1, din totalul de $n_S \stackrel{\text{not.}}{=} 2$ apariții ale stemei în datele noastre.⁵⁶³

Similar, fie z_0 numărul (neobservabil) de apariții ale feței *ban* pentru moneda 1, din totalul de $n_B \stackrel{\text{not.}}{=} 3$ apariții ale banului în datele noastre.

Funcția de verosimilitate corespunzătoare datelor din tabelul prezentat la secțiunea B este

$$L(\theta) = \theta^{z_1} (1 - \theta)^{z_0} \cdot (2\theta)^{n_S - z_1} (1 - 2\theta)^{n_B - z_0},$$

iar funcția de log-verosimilitate este

$$\ell(\theta) = z_1 \ln \theta + z_0 \ln(1 - \theta) + (n_S - z_1) \ln(2\theta) + (n_B - z_0) \ln(1 - 2\theta).$$

Aceasta ne conduce imediat la *noua expresie a funcției auxiliare*:

$$\begin{aligned} Q(\theta | \theta^{(t)}) &\stackrel{\text{def.}}{=} E[\ell(\theta)] \\ &\stackrel{\text{lin. med.}}{=} E[z_1] \ln \theta + E[z_0] \ln(1 - \theta) + (n_S - E[z_1]) \ln(2\theta) + (n_B - E[z_0]) \ln(1 - 2\theta). \end{aligned}$$

La *pasul E*, mediile

- $E[z_1]$, adică numărul așteptat (engl., expected number) de apariții ale feței *stemă* pentru moneda 1 din totalul de $n_S \stackrel{\text{not.}}{=} 2$ apariții ale stemei în datele noastre, și
- $E[z_0]$, adică numărul așteptat (engl., expected number) de apariții ale feței *ban* pentru moneda 1 din totalul de $n_B \stackrel{\text{not.}}{=} 3$ apariții ale banului în datele noastre, se calculează — la iterația t — folosind fie proprietatea de liniaritate a mediilor fie formula mediei distribuției binomiale:

$$E[z_1] = n_S \cdot q_1^{(t)} \text{ și } E[z_0] = n_B \cdot q_0^{(t)}$$

În această nouă formulare a algoritmului EM pentru problema noastră, pasul E este constituit de formulele (201) și (202).

Pasul M se elaborează după cum urmează.

Stim că

$$\theta^{(t+1)} = \underset{\theta \in (0, 1/2)}{\operatorname{argmax}} Q(\theta | \theta^{(t)}). \quad (203)$$

Dacă maximul funcției $Q(\theta | \theta^{(t)})$ se situează în *interiorul* intervalului $(0, 1/2)$, atunci cu necesitate vom avea

⁵⁶³Vă rugăm să *remarcați* faptul că variabila z_1 din acest context nu are nimic de-a face cu variabila z_1 care a fost folosită în rezolvările de la punctele *a* și *b*. Sperăm că această „suprascriere“ a semnificației variabilei z_1 să nu incomodeze cititorul. Noi am preferat / optat să păstrăm terminologia din problema 1, și anume: cu x se notează variabilele observabile și cu z variabilele neobservabile.

$$\begin{aligned}
& \frac{\partial}{\partial \theta} \left(E[z_1] \ln \theta + E[z_0] \ln(1 - \theta) + (n_S - E[z_1]) \ln(2\theta) + (n_B - E[z_0]) \ln(1 - 2\theta) \right) = 0 \\
& \Leftrightarrow \frac{E[z_1]}{\theta} - \frac{E[z_0]}{1 - \theta} + \frac{2(n_S - E[z_1])}{2\theta} - \frac{2(n_B - E[z_0])}{1 - 2\theta} = 0 \\
& \Leftrightarrow \frac{n_S \cdot q_1^{(t)}}{\theta} - \frac{n_B \cdot q_0^{(t)}}{1 - \theta} + \frac{2(n_S - n_S \cdot q_1^{(t)})}{2\theta} - \frac{2(n_B - n_B \cdot q_0^{(t)})}{1 - 2\theta} = 0 \\
& \Leftrightarrow \frac{n_S \cdot q_1^{(t)}}{\theta} - \frac{n_B \cdot q_0^{(t)}}{1 - \theta} + \frac{2n_S(1 - q_1^{(t)})}{2\theta} - \frac{2n_B(1 - q_0^{(t)})}{1 - 2\theta} = 0 \\
& \Leftrightarrow n_S(1 - \theta)(1 - 2\theta) - n_B q_0^{(t)} \theta(1 - 2\theta) - 2n_B(1 - q_0^{(t)}) \theta(1 - \theta) = 0 \\
& \Leftrightarrow n_S + \theta(-3n_S - n_B q_0^{(t)} - 2n_B + 2n_B q_0^{(t)}) + \theta^2(2n_S + 2n_B q_0^{(t)} + 2n_B - 2n_B q_0^{(t)}) = 0 \\
& \Leftrightarrow 2(n_S + n_B)\theta^2 - (3n_S + 2n_B - n_B q_0^{(t)})\theta + n_S = 0. \tag{204}
\end{aligned}$$

Prin urmare, în acest caz — adică, atunci când măcar una dintre soluțiile ecuației (204) se situează în intervalul $(0, 1/2)$ —, soluția problemei de optimizare (203) este acel (sau unul dintre acei) $\theta_i \in (0, 1/2)$, cu $i \in \{1, 2\}$.

În cazul contrar, adică atunci când $\Delta < 0$ sau când $\Delta \geq 0$ dar $\theta_1, \theta_2 \notin (0, 1/2)$, soluția acestei probleme de optimizare este într-unul din capetele intervalului $(0, 1/2)$. (Care anume dintre ele este soluția, este simplu de determinat.)

Remarcați următoarele fapte interesante:

1. Din formula (204) se vede faptul că nu este necesar să mai calculăm $q_1^{(t)}$ la pasul E. Chiar și dacă ar fi trebuit să calculăm $q_1^{(t)}$, pasul E tot ar fi fost mai eficient în noua versiune decât în versiunea „clasică“ a algoritmului EM pentru rezolvare de mixturi de distribuții Bernoulli, fiindcă acolo calculam n medii $\mu_i^{(t)}$. Acest fapt este foarte semnificativ pentru cazuri în care avem foarte multe instanțe x_i !
2. Se constată ușor că ecuația (204) este de fapt aceeași cu ecuația (196) pe care am obținut-o la pasul M atunci când probabilitatea de selecție π era considerată fixată.

4.

(Algoritmul EM: estimarea parametrilor unei mixturi de două distribuții Bernoulli (cazul general))

**■ CMU, 2008 fall, Eric Xing, HW4, pr. 1.4-7
CMU, “The EM Algorithm,” Ajit Singh, November 20, 2005**

Să presupunem că avem două monede imperfecte / măsluite. La aruncarea primei monede se obține față ‘stemă’ cu probabilitatea p , în vreme ce la aruncarea celei de-a doua monede se obține ‘stemă’ cu probabilitatea q . Mai presupunem că se efectuează n aruncări, iar la fiecare aruncare se alege prima monedă cu probabilitatea π , iar moneda a două cu probabilitatea $1 - \pi$. Rezultatul fiecărei aruncări i este $x_i \in \{0, 1\}$, notația aceasta din urmă codificând mulțimea ordonată $\{T, H\} = \{‘tail’, ‘head’\} = \{‘ban’, ‘stemă’\}$. Jocul pe care îl propunem este următorul:

Noi îți furnizăm doar rezultatul celor n aruncări, adică $x = \{x_1, x_2, \dots, x_n\}$, fără a-ți spune ce monedă am folosit pentru fiecare aruncare. Sarcina ta este următoarea: folosind algoritmul EM și dispunând de datele „observabile“ x , va trebui să estimezi valorile [de

verosimilitate maximă] pentru parametrii probabiliști p, q și π . (Vom desemna ansamblul acestor parametri prin θ .)

Pentru a calcula aceste estimări, se va considera $z = \{z_1, z_2, \dots, z_n\}$, cu $z_i \in \{0, 1\}$ variabilă „ascunsă“ indicând moneda utilizată la aruncarea i . Dacă, de exemplu, avem $z_2 = 1$, aceasta înseamnă că la aruncarea a două a fost folosită prima monedă.

- Arată că $E[z_i | x_i, \theta] = P(z_i = 1 | x_i, \theta)$.
- Folosește regula lui Bayes pentru a calcula $P(z_i = 1 | x_i, \theta)$ în funcție de x_i, z_i, p, q și π .
- Calculează log-verosimilitatea datelor „complete“, $\log P(x, z | \theta)$, ca funcție de x_i, z_i (pentru $i = 1, \dots, n$), p, q și π .
- Pasul E:* Arată că media log-verosimilității datelor complete $Q(\theta | \theta^{(t)}) \stackrel{n \rightarrow \infty}{\equiv} E_{P(z|x, \theta^{(t)})}[\log P(x, z | \theta)]$ este dată de expresia:

$$\begin{aligned} Q(\theta | \theta^{(t)}) &= \sum_{i=1}^n E[z_i | x_i, \theta^{(t)}] \cdot (\log \pi + x_i \log p + (1 - x_i) \log(1 - p)) + \\ &\quad + (1 - E[z_i | x_i, \theta^{(t)}]) \cdot (\log(1 - \pi) + x_i \log q + (1 - x_i) \log(1 - q)), \end{aligned}$$

unde $\theta^{(t)} \stackrel{\text{def.}}{=} \{p^{(t)}, q^{(t)}, \pi^{(t)}\}$ desemnează valorile celor trei parametri la iterația t a algoritmului EM.

- Pasul M:* Elaborează formulele de calcul prin care, la finalul iterației t a algoritmului EM, se obțin noile valori pentru parametri, $p^{(t+1)}, q^{(t+1)}$ și $\pi^{(t+1)}$.
- Scrie pseudo-codul algoritmului EM pentru rezolvarea acestui model de mixtură și apoi execută manual o iterație pe inputul $x = \{1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1\}$, cu valorile inițiale $1/3, 2/3$ și $1/2$ pentru parametrii p, q și respectiv π . Ce se poate observa [în privința convergenței]?

Răspuns:

- Este ușor de calculat $E[z_i | x_i, \theta]$, utilizând definiția mediei unei variabile aleatoare. (Stim că variabila z_i ia valori în multimea $\{0, 1\}$.)

$$\begin{aligned} E[z_i | x_i, \theta] &= \sum_{z \in \{0, 1\}} z_i P(z_i | x_i, \theta) = 0 \cdot P(z_i = 0 | x_i, \theta) + 1 \cdot P(z_i = 1 | x_i, \theta) \\ &\Rightarrow E[z_i | x_i, \theta] = P(z_i = 1 | x_i, \theta) \end{aligned}$$

- Folosind regula lui Bayes și probabilitățile din enunț, obținem:

$$\begin{aligned} P(z_i = 1 | x_i, \theta) &= \frac{P(x_i | z_i = 1, \theta)P(z_i = 1 | \theta)}{P(x_i | z_i = 1, \theta)P(z_i = 1 | \theta) + P(x_i | z_i = 0, \theta)P(z_i = 0 | \theta)} \\ &= \frac{p^{x_i} \cdot (1 - p)^{1-x_i} \cdot \pi}{p^{x_i} \cdot (1 - p)^{1-x_i} \cdot \pi + q^{x_i} \cdot (1 - q)^{1-x_i} \cdot (1 - \pi)} \end{aligned}$$

La ultima egalitate am ținut cont că $x_i \in \{0, 1\}$. Observați că produsul

$$p^{x_i} (1 - p)^{1-x_i} \tag{205}$$

exprimă simultan ambele cazuri posibile pentru $P(x_i|z_i = 1)$, adică $P(x_i = 1|z_i = 1) = p$ și $P(x_i = 0|z_i = 1) = 1 - p$. Similar pentru $q^{x_i}(1 - q)^{1-x_i}$.

c. Pentru a exprima log-verosimilitatea datelor complete, ținem cont de independența celor n aruncări:⁵⁶⁴

$$\begin{aligned} \log P(x, z | \theta) &\stackrel{i.i.d.}{=} \log \prod_{i=1}^n P(x_i, z_i | \theta) = \log \prod_{i=1}^n P(x_i | z_i, \theta) \cdot P(z_i | \theta) \\ &= \log \prod_{i=1}^n (p^{x_i}(1-p)^{1-x_i}\pi)^{z_i} (q^{x_i}(1-q)^{1-x_i}(1-\pi))^{1-z_i} \\ &= \sum_{i=1}^n \log \left((p^{x_i}(1-p)^{1-x_i}\pi)^{z_i} (q^{x_i}(1-q)^{1-x_i}(1-\pi))^{1-z_i} \right) \\ &= \sum_{i=1}^n [z_i \log (p^{x_i}(1-p)^{1-x_i}\pi) + (1-z_i) \log (q^{x_i}(1-q)^{1-x_i}(1-\pi))] \end{aligned}$$

La a treia egalitate de mai sus am folosit un „artificiu“ similar cu cel de la punctul c, ținând cont că $z_i \in \{0, 1\}$, iar valoarea $z_i = 1$ desemnează prima monedă în vreme ce valoarea $z_i = 0$ desemnează cea de-a doua monedă.

d. Pentru a calcula media log-verosimilității datelor complete în raport cu distribuția a posteriori $P(z | x, \theta^{(t)})$, vom ține cont de rezultatul de la punctul c și de proprietatea de liniaritate a mediilor.

$$\begin{aligned} Q(\theta | \theta^{(t)}) &\stackrel{not.}{=} E_{P(z|x, \theta^{(t)})} [\log P(x, z | \theta)] \\ &= E_{P(z|x, \theta^{(t)})} \left[\sum_{i=1}^n [z_i \log (p^{x_i}(1-p)^{1-x_i}\pi) + (1-z_i) \log (q^{x_i}(1-q)^{1-x_i}(1-\pi))] \right] \\ &= \sum_{i=1}^n \left[E[z_i | x_i, \theta^{(t)}] \cdot \log p^{x_i}(1-p)^{1-x_i}\pi + (1 - E[z_i | x_i, \theta^{(t)}]) \cdot q^{x_i}(1-q)^{1-x_i}(1-\pi) \right] \\ &= \sum_{i=1}^n \left[E[z_i | x_i, \theta^{(t)}] \cdot (\log \pi + x_i \log p + (1-x_i) \log(1-p)) + (1 - E[z_i | x_i, \theta^{(t)}]) \cdot (\log(1-\pi) + x_i \log q + (1-x_i) \log(1-q)) \right] \end{aligned}$$

e. Pentru a calcula relațiile de actualizare pentru $p^{(t+1)}$, $q^{(t+1)}$ și $\pi^{(t+1)}$, se maximizează funcția $Q(\theta | \theta^{(t)})$, care reprezintă media log-verosimilității datelor complete, în raport cu parametrii $\theta = (p, q, \pi)$. Mai exact, se folosesc derivele parțiale de ordinul întâi în funcție de p , q și respectiv π . Pentru a spori claritatea calculelor de mai jos, vom nota cu $\mu_i^{(t)}$ media $E[z_i | x_i, \theta^{(t)}]$, conform calculelor de la punctele a și b:

$$\mu_i^{(t)} = E[z_i | x_i, \theta^{(t)}] \stackrel{a,b}{=} \frac{(p^{(t)})^{x_i} \cdot (1-p^{(t)})^{1-x_i} \cdot \pi^{(t)}}{(p^{(t)})^{x_i} \cdot (1-p^{(t)})^{1-x_i} \cdot \pi^{(t)} + (q^{(t)})^{x_i} \cdot (1-q^{(t)})^{1-x_i} \cdot (1-\pi^{(t)})}$$

⁵⁶⁴Din punct de vedere metodologic, pentru a calcula $P(x, z | \theta)$ ne putem inspira de la punctul precedent: observăm că la numitorul fracției care ne dă valoarea lui $P(z_i | x_i, \theta)$ avem $P(x_i, z_i = 1 | \theta)$ și $P(x_i, z_i = 0 | \theta)$. Pornind de la aceste două expresii, ne vom propune să le exprimăm în mod unitar, adică sub forma unei singure expresii.

Cu această notație, funcția $Q(\theta | \theta^{(t)})$, a cărei expresie a fost calculată la punctul d devine:

$$\begin{aligned} Q(\theta | \theta^{(t)}) &= \sum_{i=1}^n \left[\mu_i^{(t)} (\log \pi + x_i \log p + (1 - x_i) \log(1 - p)) + \right. \\ &\quad \left. + (1 - \mu_i^{(t)}) (\log(1 - \pi) + x_i \log q + (1 - x_i) \log(1 - q)) \right] \end{aligned}$$

Ecuația de *actualizare* (engl., update) pentru parametrul p se obține astfel:

$$\begin{aligned} \frac{\partial Q(\theta | \theta^{(t)})}{\partial p} = 0 &\Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} \left(\frac{x_i}{p} - \frac{1 - x_i}{1 - p} \right) = 0 \\ \Leftrightarrow \frac{1}{p} \sum_{i=1}^n \mu_i^{(t)} x_i &= \frac{1}{1-p} \sum_{i=1}^n \mu_i^{(t)} (1 - x_i) \Leftrightarrow (1-p) \sum_{i=1}^n \mu_i^{(t)} x_i = p \sum_{i=1}^n \mu_i^{(t)} (1 - x_i) \\ \Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} x_i &= p \left(\sum_{i=1}^n \mu_i^{(t)} (1 - x_i) + \sum_{i=1}^n \mu_i^{(t)} x_i \right) \Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} x_i = p \sum_{i=1}^n \mu_i^{(t)} \\ \Rightarrow p^{(t+1)} &= \frac{\sum_{i=1}^n \mu_i^{(t)} x_i}{\sum_{i=1}^n \mu_i^{(t)}} \in [0, 1] \end{aligned} \tag{206}$$

Ecuația de actualizare pentru parametrul q este:

$$\frac{\partial Q(\theta | \theta^{(t)})}{\partial q} = 0 \Leftrightarrow \sum_{i=1}^n (1 - \mu_i^{(t)}) \left(\frac{x_i}{q} - \frac{1 - x_i}{1 - q} \right) = 0$$

Se observă similaritatea cu derivata parțială în raport cu parametrul p , care a fost calculată mai sus. Așadar, rezultă:

$$q^{(t+1)} = \frac{\sum_{i=1}^n (1 - \mu_i^{(t)}) x_i}{\sum_{i=1}^n (1 - \mu_i^{(t)})} \in [0, 1] \tag{207}$$

Ecuația de actualizare pentru parametrul π se obține astfel:

$$\begin{aligned} \frac{\partial Q(\theta | \theta^{(t)})}{\partial \pi} = 0 &\Leftrightarrow \sum_{i=1}^n \left(\frac{\mu_i^{(t)}}{\pi} - \frac{1 - \mu_i^{(t)}}{1 - \pi} \right) = 0 \\ \Leftrightarrow \frac{1}{\pi} \sum_{i=1}^n \mu_i^{(t)} &= \frac{1}{1-\pi} \sum_{i=1}^n (1 - \mu_i^{(t)}) \Leftrightarrow (1 - \pi) \sum_{i=1}^n \mu_i^{(t)} = \pi \sum_{i=1}^n (1 - \mu_i^{(t)}) \\ \Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} &= \pi \left(\sum_{i=1}^n (1 - \mu_i^{(t)}) + \sum_{i=1}^n \mu_i^{(t)} \right) \Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} = \pi \sum_{i=1}^n 1 \Leftrightarrow \sum_{i=1}^n \mu_i^{(t)} = n\pi \\ \Rightarrow \pi^{(t+1)} &= \frac{1}{n} \sum_{i=1}^n \mu_i^{(t)} \in [0, 1] \end{aligned} \tag{208}$$

Se poate constata ușor că toate cele trei derive parțiale de ordin secund ($\frac{\partial^2}{\partial p^2}$, $\frac{\partial^2}{\partial q^2}$ și $\frac{\partial^2}{\partial \pi^2}$) au doar valori negative pe domeniile lor de definiție. De asemenea, matricea hessiană (adică, matricea derivatelor parțiale de ordin secund) pentru funcția Q este diagonală și

— datorită proprietății precedente — negativ definită. În consecință, cele trei soluții aflate mai sus ($p^{(t+1)}$, $q^{(t+1)}$ și $\pi^{(t+1)}$) corespund punctului de maxim al funcției Q .

Observație: Se constată relativ ușor că formulele (206), (207) și (208) sunt variante probabiliste pentru formulele care ne-ar da estimările de verosimilitate maximă (MLE) pentru parametrii p , q și π în cazul în care valorile variabilelor z_i ar fi cunoscute.

f. Folosind rezultatele de la punctele b și e , putem scrie imediat pseudo-codul algoritmului EM pentru rezolvare de mixturi de distribuții Bernoulli.

Inițializare:

$t = 0$; atribuie valori arbitrale $(\pi^{(0)}, p^{(0)}, q^{(0)}$ în intervalul $(0, 1)$)
pentru parametrii π , p și respectiv q ;

Corpul iterativ:

pentru $t = 1, \dots, T$ (cu T fixat în avans)

(sau: cât timp log-verosimilitatea datelor observabile nu mai crește semnificativ),

(sau: până când $\|\pi^{(t)} - \pi^{(t+1)}\|^2 < \varepsilon$, $\|p^{(t)} - p^{(t+1)}\|^2 < \varepsilon$, $\|q^{(t)} - q^{(t+1)}\|^2 < \varepsilon$),
cu ε fixat în avans)

execuță

Pasul E:

$$\mu_i^{(t)} = \frac{(p^{(t)})^{x_i} \cdot (1-p^{(t)})^{1-x_i} \cdot \pi^{(t)}}{(p^{(t)})^{x_i} \cdot (1-p^{(t)})^{1-x_i} \cdot \pi^{(t)} + (q^{(t)})^{x_i} \cdot (1-q^{(t)})^{1-x_i} \cdot (1-\pi^{(t)})};$$

Pasul M:

$$p^{(t+1)} = \frac{\sum_{i=1}^n \mu_i^{(t)} x_i}{\sum_{i=1}^n \mu_i^{(t)}}, \quad q^{(t+1)} = \frac{\sum_{i=1}^n (1-\mu_i^{(t)}) x_i}{\sum_{i=1}^n (1-\mu_i^{(t)})}, \quad \pi^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \mu_i^{(t)};$$

Returnează $\pi^{(t+1)}, p^{(t+1)}, q^{(t+1)}$;

La execuția primei iterării a algoritmului EM pe datele $x = \{1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1\}$, făcând calculele se constată ușor că la pasul E obținem

$$\mu_1^{(1)} = \mu_2^{(1)} = \mu_4^{(1)} = \mu_7^{(1)} = \frac{1}{3} \text{ și } \mu_3^{(1)} = \mu_5^{(1)} = \mu_6^{(1)} = \mu_8^{(1)} = \frac{2}{3}$$

iar la pasul M

$$p^{(1)} = \frac{1}{3} = p^{(0)}, \quad q^{(1)} = \frac{2}{3} = q^{(0)} \text{ și } \pi^{(1)} = \frac{1}{2} = \pi^{(0)}.$$

Așadar, inițializând parametrii p , q și π cu valorile $p^{(0)}$, $q^{(0)}$ și respectiv $\pi^{(0)}$, algoritmul EM va bucla!

5. (Un exemplu de rezolvare a unei mixturi de două distribuții binomiale

A. când toate variabilele sunt observabile:

MLE, în manieră directă;

B. când unele variabile sunt neobservabile:

MLE, folosind algoritmul EM)

prelucrare de Liviu Ciortuz, 2015, după

■ “What is the expectation maximization algorithm?”,

Chuong B. Do, Serafim Batzoglou,

Nature Biotechnology, vol. 26, no. 8, 2008, pag. 897-899

Fie următorul experiment probabilist:

Dispunem de două monede, A și B . Efectuăm 5 serii de operațiuni de tipul următor:

- alegem în mod aleatoriu una dintre monedele A și B , cu probabilitate egală ($1/2$);
- aruncăm de 10 ori moneda care tocmai a fost aleasă (Z) și notăm rezultatul, sumarizat ca număr (X) de fețe ‘head’ (rom., ‘stemă’) obținute în urma aruncării monedei respective.⁵⁶⁵

Observație (1): Facem *presupunerea* că am reținut succesiunea [tuturor] rezultatelor obținute la aruncările celor două monede. Precizarea *de facto* a acestei succesiuni nu este esențială. Dacă nu s-ar face această *presupunere*, ar trebui să lucrăm cu *distribuția binomială*.

A. La acest punct vom considera că s-a obținut următorul rezultat pentru experimentul nostru:

i	Z_i	X_i
1	B	$5H$ (5T)
2	A	$9H$ (1T)
3	A	$8H$ (2T)
4	B	$4H$ (6T)
5	A	$7H$ (3T)

Semnificația variabilelor aleatoare Z_i și X_i pentru $i = 1, \dots, 5$ din tabelul de mai sus este imediată.

i. Calculați $\hat{\theta}_A$ și $\hat{\theta}_B$, probabilitățile de apariție a feței ‘head’ pentru cele două monede, folosind definiția clasică pentru probabilitatea evenimentelor aleatoare, și anume raportul dintre numărul de cazuri favorabile și numărul de cazuri posibile, relativ la întregul experiment.

ii. Calculați $L_1(\theta_A, \theta_B) \stackrel{not.}{=} P(X, Z | \theta_A, \theta_B)$, funcția de verosimilitate a datelor $X \stackrel{not.}{<} X_1, \dots, X_5 >$ și $Z \stackrel{not.}{=}< Z_1, \dots, Z_5 >$ în raport cu parametrii θ_A și respectiv θ_B ai distribuțiilor (Bernoulli) care modelează aruncarea celor două monede.

iii. Calculați $\hat{\theta}_A \stackrel{not.}{=} \text{argmax}_{\theta_A} \log L_1(\theta_A, \theta_B)$ și $\hat{\theta}_B \stackrel{not.}{=} \text{argmax}_{\theta_B} \log L_1(\theta_A, \theta_B)$ cu ajutorul derivatelor parțiale de ordinul întâi. Baza logaritmului, lăsată aici nespecificată, se va considera supraunitară (de exemplu 2, e sau 10).

Observație (2): Lucrând corect, veți obține același rezultat ca la punctul i.

B. La acest punct se va relua experimentul de la punctul A, însă de data aceasta vom considera că valorile variabilelor Z_i nu sunt cunoscute.

i	Z_i	X_i
1	?	$5H$ (5T)
2	?	$9H$ (1T)
3	?	$8H$ (2T)
4	?	$4H$ (6T)
5	?	$7H$ (3T)

iv. Pentru conveniență, în locul variabilelor „neobservabile“ Z_i pentru $i = 1, \dots, 5$ vom considera variabilele-indicator (de asemenea neobservabile) $Z_{i,A}, Z_{i,B} \in \{0, 1\}$, cu $Z_{i,A} +$

⁵⁶⁵Observați că dacă în loc de 10 aruncări am considera o singură aruncare, atunci am fi practic în condițiile problemei 4, unde am prezentat algoritmul EM pentru mixturi de două distribuții Bernoulli.

$Z_{i,B} = 1$, adică $Z_{i,A} = 1$ dacă și numai dacă $Z_{i,B} = 0$, și $Z_{i,A} = 0$ dacă și numai dacă $Z_{i,B} = 1$.⁵⁶⁶

Folosind teorema lui Bayes, calculați mediile variabilelor neobservabile $Z_{i,A}$ și $Z_{i,B}$ condiționate de variabilele observabile X_i . Veți considera că parametrii acestor distribuții (Bernoulli) care modeleză aruncarea monedelor A și B au valorile $\theta_A^{(0)} = 0.6$ și respectiv $\theta_B^{(0)} = 0.5$.

Așadar, se cer: $E[Z_{i,A} | X_i, \theta_A^{(0)} = 0.6, \theta_B^{(0)} = 0.5] = E[Z_{i,B} | X_i, \theta_A^{(0)} = 0.6, \theta_B^{(0)} = 0.5]$ pentru $i = 1, \dots, 5$. Ca și mai sus, probabilitățile a priori $P(Z_{i,A} = 1)$ și $P(Z_{i,B} = 1)$ se vor considera egale cu 1/2.

v. Calculați media funcției de log-verosimilitate a datelor „complete“, X și Z :

$$L_2(\theta_A, \theta_B) \stackrel{\text{def.}}{=} E_{P(Z|X, \theta^{(0)})}[\log P(X, Z | \theta)],$$

unde $\theta = (\theta_A, \theta_B)$ și $\theta^{(0)} = (\theta_A^{(0)}, \theta_B^{(0)})$. Semnificația notației de mai sus este următoarea: funcția $L_2(\theta_A, \theta_B)$ este o medie a variabilei aleatoare reprezentată de log-verosimilitatea datelor complete (observabile și, respectiv, neobservabile), iar această medie se calculează în raport cu distribuția probabilistă condițională a datelor neobservabile, $P(Z | X, \theta^{(0)})$.

Observație (3): La elaborarea calculului, veți folosi mai întâi proprietatea de liniaritate a mediilor variabilelor aleatoare, și apoi rezultatele de la punctul iv.

vi. Calculați $\theta_A^{(1)} \stackrel{\text{not.}}{=} \operatorname{argmax}_{\theta_A} L_2(\theta_A, \theta_B)$ și $\theta_B^{(1)} \stackrel{\text{not.}}{=} \operatorname{argmax}_{\theta_B} L_2(\theta_A, \theta_B)$.

C. Formalizați pașii E și M ai algoritmului EM pentru estimarea parametrilor θ_A și θ_B în condițiile de la punctul B.

Răspuns:

A. Acesta este un experiment probabilist în care toate variabilele sunt observabile.

i. Analizând datele din tabelul din enunț, rezultă imediat $\hat{\theta}_A = \frac{24}{24+6} = 0.8$ și $\hat{\theta}_B = \frac{9}{9+11} = 0.45$. Este de remarcat faptul că termenii 6 și respectiv 11 de la numitorii acestor fracții reprezintă numărul de fețe ‘tail’ (rom., ‘ban’) care au fost obținute la aruncarea monedei A și respectiv B : $6T = 1T + 2T + 3T$, $11T = 5T + 6T$.

Observație (4): Dacă în locul variabilelor binare $Z_i \in \{A, B\}$ pentru $i = 1, \dots, 5$ introducem în mod natural variabilele-indicator $Z_{i,A} \in \{0, 1\}$ și $Z_{i,B} \in \{0, 1\}$ tot pentru $i = 1, \dots, 5$, definite prin $Z_{i,A} = 1$ iff $Z_i = A$ și $Z_{i,B} = 1$ iff $Z_i = B$, atunci procesările necesare pentru calculul probabilităților / parametrilor $\hat{\theta}_A$ și $\hat{\theta}_B$ pot fi prezentate în mod sintetizat ca în tabelul de mai jos.⁵⁶⁷

⁵⁶⁶Evident, întrucât variabilele Z_i sunt aleatoare, rezultă că și variabilele $Z_{i,A}$ și $Z_{i,B}$ sunt aleatoare.

⁵⁶⁷Prezentăm acest „artificiu“ ca pregătire pentru rezolvarea (ulterioră a) punctului B al prezentei probleme.

i	$Z_{i,A}$	$Z_{i,B}$	X_i	$X_i \cdot Z_{i,A}$	$X_i \cdot Z_{i,B}$
1	0	1	$5H$	$0H (0T)$	$5H (5T)$
2	1	0	$9H$	$9H (1T)$	$0H (0T)$
3	1	0	$8H$	$8H (2T)$	$0H (0T)$
4	0	1	$4H$	$0H (0T)$	$4H (6T)$
5	1	0	$7H$	$7H (3T)$	$0H (0T)$
\Rightarrow				$\sum_{i=1}^5 X_i \cdot Z_{i,A} = 24H$	$\sum_{i=1}^5 X_i \cdot Z_{i,B} = 9H$
				$\sum_{i=1}^5 (10 - X_i) \cdot Z_{i,A} = 6T$	$\sum_{i=1}^5 (10 - X_i) \cdot Z_{i,B} = 11T$
				$\hat{\theta}_A = \frac{24}{24 + 6} = 0.8$ $\hat{\theta}_B = \frac{9}{9 + 11} = 0.45$	

ii. Calculul verosimilității datelor X și Z :

$$\begin{aligned}
L_1(\theta_A, \theta_B) &\stackrel{\text{def.}}{=} P(X, Z_A, Z_B \mid \theta_A, \theta_B) = \prod_{i=1}^5 P(X_i, Z_{i,A}, Z_{i,B} \mid \theta_A, \theta_B) \\
&\stackrel{i.i.d.}{=} \prod_{i=1}^5 P(X_i \mid Z_{i,A}, Z_{i,B}, \theta_A, \theta_B) \cdot P(Z_{i,A}, Z_{i,B} \mid \theta_A, \theta_B) \\
&= P(X_1 \mid Z_{B,1} = 1, \theta_B) \cdot 1/2 \cdot P(X_2 \mid Z_{A,2} = 1, \theta_A) \cdot 1/2 \cdot \\
&\quad P(X_3 \mid Z_{A,3} = 1, \theta_A) \cdot 1/2 \cdot P(X_4 \mid Z_{B,4} = 1, \theta_B) \cdot 1/2 \cdot \\
&\quad P(X_5 \mid Z_{A,5} = 1, \theta_A) \cdot 1/2 \\
&= \theta_B^5 (1 - \theta_B)^5 \cdot \theta_A^9 (1 - \theta_A) \cdot \theta_A^8 (1 - \theta_A)^2 \cdot \theta_B^4 (1 - \theta_B)^6 \cdot \theta_A^7 (1 - \theta_A)^3 \cdot \frac{1}{2^5} \\
&= \frac{1}{2^5} \theta_A^{24} (1 - \theta_A)^6 \theta_B^9 (1 - \theta_B)^{11}
\end{aligned}$$

iii. Funcția de log-verosimilitate a datelor complete se exprimă astfel:

$$\log L_1(\theta_A, \theta_B) = -5 \log 2 + 24 \log \theta_A + 6 \log(1 - \theta_A) + 9 \log \theta_B + 11 \log(1 - \theta_B)$$

Prin urmare, maximul acestei funcții în raport cu parametrul θ_A se calculează astfel:

$$\begin{aligned}
\frac{\partial \log L_1(\theta_A, \theta_B)}{\partial \theta_A} = 0 &\Leftrightarrow \frac{\partial}{\partial \theta_A} [24 \log \theta_A + 6 \log(1 - \theta_A)] = 0 \\
\Leftrightarrow \frac{24}{\theta_A} - \frac{6}{1 - \theta_A} = 0 &\Leftrightarrow \frac{4}{\theta_A} = \frac{1}{1 - \theta_A} \Leftrightarrow 4 - 4\theta_A = \theta_A \Leftrightarrow \hat{\theta}_A = 0.8
\end{aligned}$$

Similar, se face calculul și pentru $\frac{\partial \log L_1(\theta_A, \theta_B)}{\partial \theta_B}$ și se obține $\hat{\theta}_B = 0.45$.⁵⁶⁸ Cele două valori obținute, $\hat{\theta}_A$ și $\hat{\theta}_B$ reprezintă estimarea de verosimilitate maximă a probabilităților de apariție a feței ‘head’ / stemă pentru moneda A și respectiv moneda B .

⁵⁶⁸Se verifică ușor faptul că într-adevăr rădăcinile derivatelor parțiale de ordinul întâi pentru funcția de log-verosimilitate reprezintă puncte de maxim. Pentru aceasta, se studiază semnele acestor derivate.

Observații:

5. Am arătat pe acest caz particular că metoda de calculare a probabilităților ($\hat{\theta}_A$ și $\hat{\theta}_B$) direct din datele observate (așa cum o știm din liceu) corespunde de fapt metodei de estimare în sensul verosimilității maxime (MLE).

6. La punctul B vom arăta cum anume se poate face estimarea acelorași parametri θ_A și θ_B în cazul în care o parte din date, și anume variabilele Z_i (pentru $i = 1, \dots, 5$) sunt neobservabile.

B. Algoritmul EM ne permite să facem în mod iterativ estimarea parametrilor θ_A și θ_B în funcție de valorile variabilelor observabile, X_i , și de valorile inițiale atribuite parametrilor (în cazul nostru, $\theta_A^{(0)} = 0.6$ și $\theta_B^{(0)} = 0.5$).

Notă:

Vom sintetiza calculele de la prima iterație a algoritmului EM — care vor fi detaliate la punctele iv , v și vi de mai jos — sub forma următoare, care seamănă (dar și diferă!) într-o anumită măsură de tabelele de la punctul A:

i	$Z_{i,A}$	$Z_{i,B}$	X_i	$E[Z_{i,A}]$	$E[Z_{i,B}]$	$X_i \cdot E[Z_{i,A}]$	$X_i \cdot E[Z_{i,B}]$
1	—	—	5H	0.45H	0.55H	2.2H (2.2T)	2.8H (2.8T)
2	—	—	9H	\xrightarrow{E} 0.80H	0.20H	\xrightarrow{M} 7.2H (0.8T)	1.8H (0.2T)
3	—	—	8H	0.73H	0.27H	5.9H (1.5T)	2.1H (0.5T)
4	—	—	4H	0.35H	0.65H	1.4H (2.1T)	2.6H (3.9T)
5	—	—	7H	0.65H	0.35H	4.5H (1.9T)	2.5H (1.1T)

$$\Rightarrow \begin{cases} \sum_{i=1}^5 X_i \cdot E[Z_{i,A}] = 21.3H \\ \sum_{i=1}^5 (10 - X_i) \cdot E[Z_{i,A}] = 8.7T \\ \sum_{i=1}^5 X_i \cdot E[Z_{i,B}] = 11.7H \\ \sum_{i=1}^5 (10 - X_i) \cdot E[Z_{i,B}] = 8.3T \end{cases} \Rightarrow \begin{cases} \hat{\theta}_A^{(1)} = \frac{21.3}{21.3 + 8.7} \approx 0.71 \\ \hat{\theta}_B^{(1)} = \frac{11.7}{11.7 + 8.3} \approx 0.58 \end{cases}$$

Vom arăta că

- într-adevăr, este posibilă calcularea mediilor variabilelor neobservabile $Z_{i,A}$ și $Z_{i,B}$, condiționate de variabilele observabile X_i și în funcție de valorile asignate inițial pentru parametrii θ_A și θ_B ;
- față de tabloul de sinteză de la punctul precedent, când toate variabilele erau observabile și se calculau produsele $X_i \cdot Z_{i,A}$ și $X_i \cdot Z_{i,B}$, aici se înlocuiesc variabilele $Z_{i,A}$ și $Z_{i,B}$ cu mediile $E[Z_{i,A}]$ și $E[Z_{i,B}]$ în produsele respective. De fapt, în loc să se calculeze $\sum_i X_i \cdot Z_{i,A}$ se calculează media $E[\sum_i X_i \cdot Z_{i,A}]$, și similar pentru B .

Observație importantă (7): Pentru simplitate, în cele de mai sus (inclusiv în tabelele precedente), prin $E[Z_{i,A}]$ am notat $E[Z_{i,A} | X_i, \theta^{(0)}]$, iar prin $E[Z_{i,B}]$ am notat $E[Z_{i,B} | X_i, \theta^{(0)}]$, unde $\theta^{(0)} \stackrel{\text{not.}}{=} (\theta_A^{(0)}, \theta_B^{(0)})$.

iv. Întrucât variabilele $Z_{i,A}$ au valori booleene (0 sau 1), rezultă că

$$E[Z_{i,A} | X_i, \theta^{(0)}] = 0 \cdot P(Z_{i,A} = 0 | X_i, \theta^{(0)}) + 1 \cdot P(Z_{i,A} = 1 | X_i, \theta^{(0)}) = P(Z_{i,A} = 1 | X_i, \theta^{(0)})$$

Probabilitățile $P(Z_{i,A} = 1 | X_i, \theta^{(0)}) = P(Z_{i,A} = 1 | X_i, \theta^{(0)})$, pentru $i = 1, \dots, 5$ se pot calcula folosind teorema lui Bayes:

$$\begin{aligned} &P(Z_{i,A} = 1 | X_i, \theta^{(0)}) \\ &= \frac{P(X_i | Z_{i,A} = 1, \theta^{(0)}) \cdot P(Z_{i,A} = 1 | \theta^{(0)})}{P(X_i | Z_{i,A} = 1, \theta^{(0)}) \cdot P(Z_{i,A} = 1 | \theta^{(0)}) + P(X_i | Z_{i,A} = 0, \theta^{(0)}) \cdot P(Z_{i,A} = 0 | \theta^{(0)})} \end{aligned}$$

$$= \frac{P(X_i | Z_{i,A} = 1, \theta_A^{(0)})}{P(X_i | Z_{i,A} = 1, \theta_A^{(0)}) + P(X_i | Z_{i,B} = 1, \theta_B^{(0)})}$$

S-a ținut cont că $P(Z_{i,A} = 1 | \theta^{(0)}) = P(Z_{i,B} = 1 | \theta^{(0)}) = 1/2$ (a se vedea enunțul).

De exemplu, pentru $i = 1$ vom avea:

$$\begin{aligned} E[Z_{A,1} | X_1, \theta^{(0)}] &= \frac{0.6^5(1-0.6)^5}{0.6^5(1-0.6)^5 + 0.5^5(1-0.5)^5} = \frac{0.24^5}{0.24^5 + 0.25^5} = \frac{1}{1 + \left(\frac{25}{24}\right)^5} \\ &\approx 0.45 \end{aligned}$$

Similar cu $E[Z_{A,1} | X_1, \theta^{(0)}]$ se calculează și celelalte medii $E[Z_{i,A} | X_i, \theta^{(0)}]$ pentru $i = 2, \dots, 5$ și $E[Z_{i,B} | X_i, \theta^{(0)}]$ pentru $i = 1, \dots, 5$.⁵⁶⁹ Am înregistrat aceste valori / medii în tabelul din mijloc din cadrul seriei formate din cele trei tabele din *Nota* de mai sus.

v. Media funcției de log-verosimilitate a datelor complete, $L_2(\theta_A, \theta_B)$, se calculează astfel:

$$\begin{aligned} L_2(\theta_A, \theta_B) &\stackrel{\text{def.}}{=} E_{P(Z|X, \theta^{(0)})}[\log P(X, Z | \theta)] \\ &\stackrel{\text{indep. cdt.}}{=} E_{P(Z|X, \theta^{(0)})}[\log \prod_{i=1}^5 P(X_i, Z_{i,A}, Z_{i,B} | \theta_A, \theta_B)] \\ &\stackrel{\text{reg. de mult.}}{=} E_{P(Z|X, \theta^{(0)})}[\log \prod_{i=1}^5 P(X_i | Z_{i,A}, Z_{i,B}; \theta_A, \theta_B) \cdot P(Z_{i,A}, Z_{i,B} | \theta_A, \theta_B)] \end{aligned}$$

În continuare, omițând din nou distribuția probabilistă în raport cu care se calculează media aceasta întrucât ea poate fi subînțeleasă, vom scrie:

$$\begin{aligned} L_2(\theta_A, \theta_B) &= \\ &= E[\log \prod_{i=1}^5 (\theta_A^{Z_{i,A}})^{X_i} \cdot [(1 - \theta_A)^{Z_{i,A}}]^{10-X_i} \cdot (\theta_B^{Z_{i,B}})^{X_i} \cdot [(1 - \theta_B)^{Z_{i,B}}]^{10-X_i} \cdot \frac{1}{2}] \\ &= E[\sum_{i=1}^5 [X_i \cdot Z_{i,A} \cdot \log \theta_A + (10 - X_i) \cdot Z_{i,A} \cdot \log(1 - \theta_A) + \\ &\quad X_i \cdot Z_{i,B} \cdot \log \theta_B + (10 - X_i) \cdot Z_{i,B} \cdot \log(1 - \theta_B) - \log 2]] \\ &\stackrel{\text{lin. med.}}{=} \sum_{i=1}^5 [X_i \cdot E[Z_{i,A}] \cdot \log \theta_A + (10 - X_i) \cdot E[Z_{i,A}] \cdot \log(1 - \theta_A) + \\ &\quad X_i \cdot E[Z_{i,B}] \cdot \log \theta_B + (10 - X_i) \cdot E[Z_{i,B}] \cdot \log(1 - \theta_B) - \log 2] \\ &= \sum_{i=1}^5 \log[\theta_A^{X_i \cdot E[Z_{i,A}]} \cdot (1 - \theta_A)^{(10 - X_i) \cdot E[Z_{i,A}]} \cdot \\ &\quad \theta_B^{X_i \cdot E[Z_{i,B}]} \cdot (1 - \theta_B)^{(10 - X_i) \cdot E[Z_{i,B}]} \cdot \frac{1}{2}] \\ &= \log(\theta_A^{2.2} \cdot (1 - \theta_A)^{2.2} \cdot \theta_B^{2.8} \cdot (1 - \theta_B)^{2.8} \cdot \dots \cdot \theta_A^{4.5} \cdot (1 - \theta_A)^{1.9} \cdot \theta_B^{2.5} \cdot (1 - \theta_B)^{1.1} \cdot \frac{1}{2^5}). \end{aligned}$$

⁵⁶⁹Se poate ține cont că, de îndată ce s-a calculat $E[Z_{i,A} | X_i, \theta^{(0)}]$, se poate obține imediat și $E[Z_{i,B} | X_i, \theta^{(0)}] = 1 - E[Z_{i,A} | X_i, \theta^{(0)}]$, fiindcă $Z_{i,A} + Z_{i,B} = 1$.

La ultima egalitate de mai sus, cantitățile fracționare provin din calculele simple $X_1 \cdot E[Z_{1,A} | X, \theta] \approx 2.2$, $X_1 \cdot E[Z_{1,B} | X, \theta] \approx 2.8$, ..., $X_5 \cdot E[Z_{1,A} | X, \theta] \approx 4.5$, $X_5 \cdot E[Z_{1,B} | X, \theta] \approx 2.5$ (a se vedea tabelele din cadrul *Notei* de mai sus).

Observație (8): Comparând funcția de [log]-verosimilitate de la partea A (vedeți punctul *ii.*) cu cea de aici, se constată că

- $X_i \cdot (Z_{i,A} + Z_{i,B})$ de acolo — mai precis: $X_i \cdot Z_{i,A}$ sau $X_i \cdot Z_{i,B}$, în funcție de valoarea lui Z_i — devine aici $X_i \cdot E[Z_{i,A} + Z_{i,B}] = X_i \cdot (E[Z_{i,A}] + E[Z_{i,B}])$,
- $\theta_A^{X_i \cdot Z_{i,A}} \cdot \theta_B^{X_i \cdot Z_{i,B}}$ — adică $\theta_A^{X_i}$ sau $\theta_B^{X_i}$, în funcție de valoarea lui Z_i — se înlocuiește cu $\theta_A^{X_i \cdot E[Z_{i,A}]} \cdot \theta_B^{X_i \cdot E[Z_{i,B}]}$.
- $(1 - \theta_A)^{(10 - X_i) \cdot Z_{i,A}} \cdot (1 - \theta_B)^{(10 - X_i) \cdot Z_{i,B}}$ — adică $(1 - \theta_A)^{10 - X_i}$ sau $(1 - \theta_B)^{10 - X_i}$, în funcție de valoarea lui Z_i — se înlocuiește cu $(1 - \theta_A)^{(10 - X_i) \cdot E[Z_{i,A}]} \cdot (1 - \theta_B)^{(10 - X_i) \cdot E[Z_{i,B}]}$.

Aceste înlocuiri / corespondențe, deși dau acum o *perspectivă intuitivă* asupra acestei instanțe particulare a schemei algoritmice EM — iar aceasta se va regăsi și în cazul altor instanțe ale aceleiași scheme algoritmice — se datorează în mod riguros calculelor de mai sus.

vi. Valorile parametrilor θ_A și θ_B pentru care se atinge maximul mediei funcției de log-verosimilitate a datelor complete se obțin cu ajutorul derivatelor parțiale de ordinul întâi:⁵⁷⁰

$$\begin{aligned} \frac{\partial L_2(\theta_A, \theta_B)}{\partial \theta_A} &= 0 \\ \Rightarrow \frac{\partial}{\partial \theta_A} (2.2 \log \theta_A + 2.2 \log(1 - \theta_A) + \dots + 4.5 \log \theta_A + 1.9 \log(1 - \theta_A)) &= 0 \\ \Rightarrow \frac{2.2}{\theta_A} - \frac{2.2}{1 - \theta_A} + \dots + \frac{4.5}{\theta_A} - \frac{1.9}{1 - \theta_A} &= 0 \Rightarrow \dots \Rightarrow \theta_A^{(1)} \approx 0.71. \end{aligned}$$

Similar, vom obține $\theta_B^{(1)} \approx 0.58$.

C. Formulele care se folosesc în cadrul algoritmului EM pentru rezolvarea problemei date (adică estimarea parametrilor θ_A și θ_B când variabilele Z_i sunt neobservabile) se deduc astfel:

Pasul E:

$$\begin{aligned} E[Z_{i,A} | X, \theta] &= P(Z_{i,A} = 1 | X, \theta) = P(Z_{i,A} = 1 | X_i, \theta) \\ &= \frac{P(X_i | Z_{i,A} = 1; \theta) \cdot \overbrace{P(Z_{i,A} = 1 | \theta)}^{1/2}}{P(X_i | Z_{i,A} = 1; \theta) \cdot P(Z_{i,A} = 1 | \theta) + P(X_i | Z_{i,B} = 1; \theta) \cdot \overbrace{P(Z_{i,B} = 1 | \theta)}^{1/2}} \\ &= \frac{P(X_i | Z_{i,A} = 1; \theta)}{P(X_i | Z_{i,A} = 1; \theta) + P(X_i | Z_{i,B} = 1; \theta)} \\ &= \frac{\theta_A^{X_i} (1 - \theta_A)^{10 - X_i}}{\theta_A^{X_i} (1 - \theta_A)^{10 - X_i} + \theta_B^{X_i} (1 - \theta_B)^{10 - X_i}} \end{aligned}$$

Procedând în mod similar, vom obține:

$$E[Z_{i,B} | X, \theta] = \frac{\theta_B^{X_i} (1 - \theta_B)^{10 - X_i}}{\theta_A^{X_i} (1 - \theta_A)^{10 - X_i} + \theta_B^{X_i} (1 - \theta_B)^{10 - X_i}}$$

⁵⁷⁰Este imediat că derivatele de ordinul al doilea au valori negative pe tot domeniul de definiție.

Notând cu

- x_i valoarea variabilei X_i ,
- $\theta_A^{(t)}$ și respectiv $\theta_B^{(t)}$ estimările parametrilor θ_A și θ_B la iteratăia t a algoritmului EM,
- $p_{i,A}^{(t+1)}$ și respectiv $p_{i,B}^{(t+1)}$, mediile $E[Z_{i,A} | X_i, \theta_A^{(t)}]$ și $E[Z_{i,B} | X_i, \theta_B^{(t)}]$,

vom avea:

$$\begin{aligned} p_{i,A}^{(t+1)} &= \frac{(\theta_A^{(t)})^{x_i} (1 - \theta_A^{(t)})^{10-x_i}}{(\theta_A^{(t)})^{x_i} (1 - \theta_A^{(t)})^{10-x_i} + (\theta_B^{(t)})^{x_i} (1 - \theta_B^{(t)})^{10-x_i}} \\ p_{i,B}^{(t+1)} &= \frac{(\theta_B^{(t)})^{x_i} (1 - \theta_B^{(t)})^{10-x_i}}{(\theta_A^{(t)})^{x_i} (1 - \theta_A^{(t)})^{10-x_i} + (\theta_B^{(t)})^{x_i} (1 - \theta_B^{(t)})^{10-x_i}} \end{aligned}$$

Pasul M: Ca și mai înainte, în formulele de mai jos vom nota $E[Z_{i,A} | X_i, \theta^{(t)}]$ cu $E[Z_{i,A}]$ și $E[Z_{i,B} | X_i, \theta^{(t)}]$ cu $E[Z_{i,B}]$. Cu aceste notății, procedând similar cu calculul de la partea B, punctul v , vom avea:

$$L_2(\theta_A, \theta_B) = \log \prod_{i=1}^5 \theta_A^{x_i E[Z_{i,A}]} (1 - \theta_A)^{(10-x_i)E[Z_{i,A}]} \theta_B^{x_i E[Z_{i,B}]} (1 - \theta_B)^{(10-x_i)E[Z_{i,B}]}$$

Prin urmare,

$$\begin{aligned} \frac{\partial}{\partial \theta_A} L_2(\theta_A, \theta_B) = 0 &\Rightarrow \frac{1}{\theta_A} \sum_{i=1}^5 x_i E[Z_{i,A}] = \frac{1}{1 - \theta_A} \sum_{i=1}^5 (10 - x_i) E[Z_{i,A}] \\ \Rightarrow (1 - \theta_A) \sum_{i=1}^5 x_i E[Z_{i,A}] &= \theta_A \sum_{i=1}^5 (10 - x_i) E[Z_{i,A}] \Rightarrow \sum_{i=1}^5 x_i E[Z_{i,A}] = 10 \theta_A \sum_{i=1}^5 E[Z_{i,A}] \\ \Rightarrow \theta_A &= \frac{\sum_{i=1}^5 x_i E[Z_{i,A}]}{10 \sum_{i=1}^5 E[Z_{i,A}]} \quad \text{și, similar, } \theta_B = \frac{\sum_{i=1}^5 x_i E[Z_{i,B}]}{10 \sum_{i=1}^5 E[Z_{i,B}]} \end{aligned}$$

Așadar, la pasul M al algoritmului EM vom avea:

$$\theta_A^{(t+1)} = \frac{\sum_{i=1}^5 x_i p_{i,A}^{(t+1)}}{10 \sum_{i=1}^5 p_{i,A}^{(t+1)}} \quad \theta_B^{(t+1)} = \frac{\sum_{i=1}^5 x_i p_{i,B}^{(t+1)}}{10 \sum_{i=1}^5 p_{i,B}^{(t+1)}}$$

Observație (9): Implementând algoritmul EM cu relațiile obținute pentru pasul E și pasul M, după execuția a 10 iterări se vor obține valorile $\theta_A^{(10)} \approx 0.80$ și $\theta_B^{(10)} \approx 0.52$. Este interesant de observat că estimarea obținută pentru parametrul θ_A este acum la același nivel cu cea obținută prin metoda verosimilității maxime (MLE) în cazul observării tuturor variabilelor (0.80, vedeți rezolvarea de la partea A, punctul i), iar estimarea obținută pentru parametrul θ_B a coborât de la valoarea 0.58 care a fost obținută la prima iteratăie a algoritmului EM la o valoare (0.52) care este considerabil mai apropiată de estimarea prin metoda MLE (0.45).

6.

(Algoritmul EM:
 „învățarea“ unei distribuții probabiliste categoriale,
 determinate de un singur parametru (de estimat),
 în condițiile existenței unei variabile neobservabile)

prelucrare de Liviu Ciortuz, după
CMU, 2008 fall, Eric Xing, final exam, pr. 6

Considerăm un curs de învățare automată la care probabilitatea ca un student să fie notat cu calificativul A este $P(A) = 1/2$, cu B este $P(B) = \theta$, cu C este $P(C) = 2\theta$, iar cu D este $P(D) = 1/2 - 3\theta$, unde $\theta \in (0, 1/6)$ este un parametru. Ni se mai spune că un număr de c studenți au luat calificativul C , iar d studenți au luat calificativul D . Nu știm cu exactitate căți studenți au luat calificativul A sau căți studenți au luat calificativul B , dar știm că în total h studenți au luat unul dintre aceste două calificative.

Ne propunem să utilizăm algoritmul Expectation-Maximization pentru a obține o estimare de verosimilitate maximă a parametrului θ .

Observații:

1. Remarcăm faptul că în această problemă nu „rezolvăm“ o mixtură de distribuții, ci învățăm o distribuție (categorială), a cărei definiție depinde de un parametru (θ).

2. Am specificat distribuția de învățat în tabelul alăturat (mai precis, în primele două coloane din acest tabel). Evident, dacă am cunoaște valoarea lui a , am putea calcula imediat estimarea de verosimilitate maximă (MLE) a lui θ .

	prob.	count-uri
A	$1/2$	$a = ?$
B	θ	$b = h - a$
C	2θ	c
D	$1/2 - 3\theta$	d

De exemplu, folosind ultimele trei linii din tabel, am putea scrie:

$$\frac{3\theta}{1} = \frac{b + c}{b + c + d} \Leftrightarrow 6\theta = \frac{b + c}{b + c + d} \Leftrightarrow \theta = \frac{1}{6} \cdot \frac{b + c}{b + c + d} \Leftrightarrow \theta = \frac{1}{6} \cdot \frac{h - a + c}{h - a + c + d}$$

3. Conform formalizării generale a algoritmului EM, datele (observabile și neobservabile) ar trebui să fie (în contextul acestei probleme) x_1, \dots, x_{h+c+d} , fiecare x_i aparținând mulțimii $\{A, B, C, D\}$. Datele observabile ar fi acele instanțe x_i care aparțin mulțimii $\{C, D\}$, iar datele neobservabile restul ($x_i \in \{A, B\}$). Cum ordinea instanțelor x_i nu contează (întrucât operăm cu distribuția categorială, nu cea multinomială), se constată că este suficient să considerăm ca date observabile h , c și d . Variabila neobservabilă va fi considerată a . (Corepunzător, vom avea $b = h - a$.) Se observă că toate cele patru linii din tabel vor fi necesare la scrierea funcției „auxiliare“ pentru algoritmul EM, care reprezintă media log-verosimilității datelor complete.

4. În mod normal, algoritmul EM face la pasul E calculul unei medii (care, în cazul mixturilor, este o probabilitate de forma $P(z_i = j|x_i, \theta^{(t)})$). În problema noastră, cunoscând $\theta^{(t)}$, valoarea parametrului θ la iterată t , vom putea calcula valoarea medie / „așteptată“ a lui a .

a. Pasul de calculare a mediilor (E): Care dintre formulele următoare reprezintă valorile „așteptate“ (engl., expected values) pentru variabilele a și b (văzute ca variabile aleatoare), exprimate în funcție de parametrul θ ?

$$\begin{array}{lll}
 (i) \quad \hat{a} = \frac{1}{\frac{1}{2} + h} \theta & \hat{b} = \frac{\theta}{\frac{1}{2} + h} \theta & (ii) \quad \hat{a} = \frac{1}{\frac{1}{2} + \theta} h & \hat{b} = \frac{\theta}{\frac{1}{2} + \theta} h \\
 (iii) \quad \hat{a} = \frac{\theta}{\frac{1}{2} + \theta} h & \hat{b} = \frac{1}{\frac{1}{2} + \theta} h & (iv) \quad \hat{a} = \frac{1}{1 + \theta^2} h & \hat{b} = \frac{\theta}{1 + \theta^2} h
 \end{array}$$

b. Pasul de maximizare (M): Având mediile pentru variabilele a și b (notate cu \hat{a} și respectiv \hat{b} ca mai sus), care dintre formulele următoare reprezintă estimarea de verosimilitate maximă a parametrului θ ?

$$\begin{array}{ll}
 (i) \quad \hat{\theta} = \frac{h - \hat{a} + c}{6(h - \hat{a} + c + d)} & (ii) \quad \hat{\theta} = \frac{h - \hat{a} + d}{6(h - 2\hat{a} - d)} \\
 (iii) \quad \hat{\theta} = \frac{h - \hat{a}}{6(h - 2\hat{a} + c)} & (iv) \quad \hat{\theta} = \frac{2(h - \hat{a})}{3(h - \hat{a} + c + d)}
 \end{array}$$

Răspuns:

a. Valoarea medie pentru a se calculează astfel:

Notând $P(A \cup B)$ probabilitatea ca un student oarecare să ia fie calificativul A fie calificativul B, rezultă:⁵⁷¹

$$\hat{a} = P(A \mid A \cup B) \cdot h \stackrel{\text{def.}}{=} \frac{P(A)}{P(A) + P(B)} \cdot h = \frac{\frac{1}{2}}{\frac{1}{2} + \theta} \cdot h$$

La cea de-a doua egalitate am ținut cont și de faptul că A și B , văzute ca multimi, sunt disjuncte.

Procedând în mod similar, vom obține $\hat{b} = E[b \mid h, \theta] = \frac{\theta}{\frac{1}{2} + \theta} \cdot h$. Deci răspunsul corect este (ii).

b. La pasul M, algoritmul EM maximizează media log-verosimilității datelor complete în funcție de θ . Ținând cont de independența datelor, funcția de verosimilitate a datelor complete se exprimă astfel:

$$P(a, b, c, d \mid \theta) = \left(\frac{1}{2}\right)^a (\theta)^b (2\theta)^c \left(\frac{1}{2} - 3\theta\right)^d.$$

Așadar,

$$\log P(a, b, c, d \mid \theta) = a \log \frac{1}{2} + b \log \theta + c \log(2\theta) + d \log \frac{1 - 6\theta}{2}$$

Înlocuind în această expresie variabilele necunoscute / neobservabile a și b cu mediile lor, obținem

$$E[\log P(a, b, c, d \mid \theta)] = \hat{a} \log \frac{1}{2} + \hat{b} \log \theta + c \log(2\theta) + d \log \frac{1 - 6\theta}{2}$$

⁵⁷¹Valoarea „așteptată“ pentru a (numărul de studenți care au obținut calificativul A) este dată de media *distribuției binomiale* de parametri h și $\frac{P(A)}{P(A) + P(B)}$. Se știe că această medie este exact produsul celor doi parametri. Vedeți [și] problema 23 de la capitolul de *Fundamente*.

Maximizarea acestei medii (în raport cu parametrul θ) se realizează cu ajutorul derivatei de ordinul întâi:⁵⁷²

$$\begin{aligned} \frac{\partial E[\log p(a, b, c, d | \theta)]}{\partial \theta} = 0 &\Leftrightarrow \frac{\hat{b}}{\theta} + \frac{2c}{2\theta} - \frac{3d}{1-6\theta} = 0 \Leftrightarrow \frac{\hat{b}}{\theta} + \frac{c}{\theta} - \frac{6d}{1-6\theta} = 0 \\ &\Leftrightarrow \frac{\hat{b}+c}{\theta} = \frac{6d}{1-6\theta} \Leftrightarrow 6d\theta = (\hat{b}+c)(1-6\theta) \Leftrightarrow 6\theta(d+\hat{b}+c) = \hat{b}+c \\ &\Rightarrow \hat{\theta} = \frac{\hat{b}+c}{6(\hat{b}+c+d)} = \frac{h-\hat{a}+c}{6(h-\hat{a}+c+d)} \end{aligned}$$

Prin urmare, răspunsul corect este (i).

Observații:

5. Regula de actualizare care tocmai a fost obținută la punctul b este un corespondent natural al formulei de estimare a parametrului θ corespunzătoare cazului când nu există date ascunse. (Vedeti *Observația 2* de mai sus.)
6. Folosind notația uzuală pentru algoritmul EM, putem spune că mediile $\hat{a} \stackrel{\text{not.}}{=} a^{(t+1)}$ și $\hat{b} \stackrel{\text{not.}}{=} b^{(t+1)}$ au fost calculate în funcție de „ipoteza“ curentă $\theta^{(t)}$, iar apoi expresia $E[\log P(a, b, c, d | \theta)]$ a fost calculată în funcție de θ , \hat{a} și \hat{b} (deci în funcție de θ și $\theta^{(t)}$).

7. (Algoritmul EM pentru „învățarea“ unei distribuții categoriale (și implicit a unei distribuții multinomiale); o aplicație în domeniul bioinformaticii)

Liviu Ciortuz, 2017, pornind de la
■ Ex. 20.10 din “Probability for Statistics and Machine Learning”
Anirban DasGupta, Springer, 2011

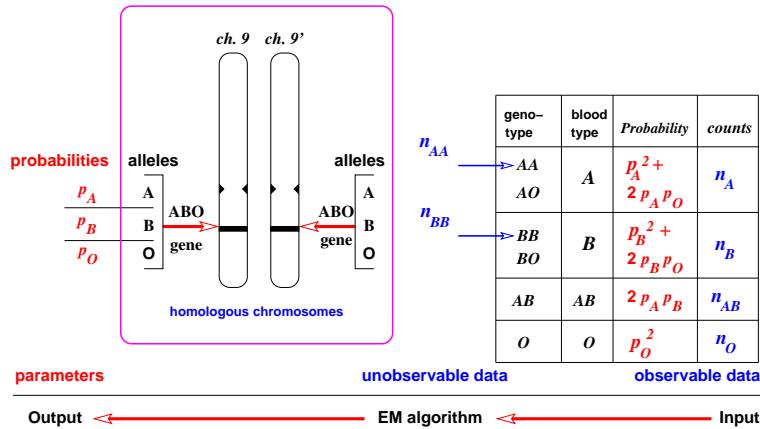
Grupele sangvine ale oamenilor sunt determinate de variantele („alelele“) unei gene situate pe cromozomul 9 (mai exact, la poziția 9q34.2), numită gena *ABO*. Se știe că fiecare dintre noi dispunem de câte o pereche de astfel de cromozomi, deci de câte două copii ale genei *ABO*, și anume una moștenită de la tată și cealaltă moștenită de la mamă.

Se notează cu *A*, *B* și *O* cele trei tipuri de alele ale genei *ABO*. Alelele *A* și *B* sunt *dominante* în raport cu alela *O*. (Altfel spus, alela *O* este *recesivă* în raport cu fiecare dintre alele *A* și *B*.) Alelele *A* și *B* sunt *codominante*. Prin urmare, grupele sangvine pot fi specificate conform tabelului alăturat.

grupe sangvine	alele moștenite
<i>A</i>	<i>AA</i> <i>AO</i>
<i>B</i>	<i>BB</i> <i>BO</i>
<i>AB</i>	<i>AB</i>
<i>O</i>	<i>OO</i>

Presupunem că, într-o populație oarecare, fiecare dintre alelele *A*, *B* și *O* are la bărbați aceeași frecvență ca și la femei. În cele ce urmează, aceste frecvențe (notate respectiv cu p_A , p_B și p_O) vor fi considerate a priori necunoscute, dar urmează să le determinăm.

⁵⁷²Este imediat că derivata a două este negativă pe tot domeniul ei de definitie, deci funcția de log-verosimilitate este strict concavă și, în consecință, admite un singur punct de maxim.



a. Considerăm că într-un eșantion de populație format din n persoane sunt n_A persoane cu grupa sanguină A , n_B persoane cu grupa sanguină B , n_{AB} persoane cu grupa sanguină AB și n_O persoane cu grupa sanguină O . (Evident, $n = n_A + n_B + n_{AB} + n_O$.) Pornind de la aceste date „observabile“, să se deriveze algoritmul EM pentru determinarea probabilităților p_A , p_B și p_O . (Evident, este suficient să se determine două dintre ele, fiindcă suma lor este 1.)

b. Implementați algoritmul EM pe care l-ați conceput la punctul a și rulați-l pentru inputul $n_A = 186$, $n_B = 38$, $n_{AB} = 13$, $n_O = 284$ ($n = 521$). Ca valori inițiale pentru parametri, veți lucra mai întâi cu $p_A = p_B = p_O = \frac{1}{3}$, iar apoi cu $p_A = p_O = 0.01$ și $p_B = 0.98$. Pentru oprire, veți cere ca $p_A^{(t)}$, $p_B^{(t)}$ și $p_O^{(t)}$ să nu difere (fiecare în parte) cu mai mult de 10^{-4} față de valorile calculate la iteratărea precedentă. Comparați rezultatele obținute pentru fiecare din cele două inițializări.

Indicații:

1. Presupunând că procesul de asociere a alelor moștenite de către un individ de la părinții lui respectă proprietățile specifice evenimentelor aleatoare independente, rezultă că probabilitățile de realizare a combinațiilor (în termeni genetici: „fenotipurile“) AA , AO , BB , BO , AB și OO într-o populație oarecare sunt p_A^2 , $2p_A p_O$, p_B^2 , $2p_B p_O$, $2p_A p_B$, și respectiv p_O^2 .
2. Considerând $n_A = n_{AA} + n_{AO}$ și $n_B = n_{BB} + n_{BO}$, unde semnificațiile numerelor n_{AA} , n_{AO} , n_{BB} și n_{BO} sunt similare cu semnificațiile numerelor n_A , n_B , n_{AB} , și n_O care au fost precizate mai sus, este natural ca în formularea algoritmului EM datele n_{AA} și n_{BB} să fie considerate „neobservabile“. Ca parametri ai modelului, se vor considera probabilitățile p_A , p_B și p_O .
3. La pasul E al algoritmului EM veți calcula \hat{n}_{AA} și \hat{n}_{BB} , care reprezintă numărul „așteptat“ de apariții ale combinației de alele AA și respectiv numărul „așteptat“ de apariții ale combinației de alele BB în populația dată. Veți scrie apoi funcția de log-verosimilitate a datelor complete („observabile“ și „neobservabile“), exprimată cu ajutorul distribuției $Multinomial(n; p_A^2, 2p_A p_O, p_B^2, 2p_B p_O, 2p_A p_B, p_O^2)$.
4. La pasul M al algoritmului EM, pornind de la media funcției de log-verosimilitate care a fost calculată la pasul E, veți stabili regulile de actualizare pentru probabilitățile p_A , p_B și p_O . Atenție: suma acestor probabilități fiind 1, problema de optimizare pe care va trebui

să o rezolvați la pasul M este una cu restricții. În acest sens, *metoda multiplicatorilor lui Lagrange* vă poate fi de folos.

Răspuns:

Vom folosi următoarele notații:

- setul de parametri: $p = \{p_A, p_B, p_O\}$;
de fapt, va fi suficient să estimăm doar p_A și p_B , fiindcă $p_A + p_B + p_O = 1$
- variabilele observabile: $n_{obs} = \{n_A, n_B, n_{AB}, n_O\}$
- variabilele neobservabile: $n_{unobs} = \{n_{AA}, n_{AO}, n_{BB}, n_{BO}\}$;
de fapt, vom reduce setul de variabile n_{unobs} la $\{n_{AA}, n_{BB}\}$
- datele complete: $n_{compl} = n_{obs} \cup n_{unobs}$
- $n = n_A + n_B + n_{AB} + n_O$, $n_A = n_{AA} + n_{AO}$, $n_B = n_{BB} + n_{BO}$.

Pornind de la definiția funcției masă de probabilitate (p.m.f.) pentru *distribuția multinomială*, rezultă că expresia funcției de verosimilitate a datelor complete din problema noastră este următoarea:⁵⁷³

$$\begin{aligned} L(p) &\stackrel{not.}{=} P(n_{compl}|p) = \\ &= \frac{n!}{n_{AA}! n_{AO}! n_{BB}! n_{BO}! n_{AB}! n_O!} \cdot \\ &\quad (p_A^2)^{n_{AA}} \cdot (2 p_A p_O)^{n_{AO}} \cdot (p_B^2)^{n_{BB}} \cdot (2 p_B p_O)^{n_{BO}} \cdot (2 p_A p_B)^{n_{AB}} \cdot (p_O^2)^{n_O} \end{aligned}$$

Logaritmând această expresie, obținem funcția de log-verosimilitate:

$$\begin{aligned} \ell(p) &\stackrel{def.}{=} \ln L(p) \\ &= \ln c + n_{AA} \ln(p_A^2) + n_{AO} \ln(2 p_A p_O) + n_{BB} \ln(p_B^2) + n_{BO} \ln(2 p_B p_O) + \\ &\quad n_{AB} \ln(2 p_A p_B) + n_O \ln(p_O^2) \\ &= \ln c' + 2 n_{AA} \ln p_A + n_{AO} (\ln p_A + \ln p_O) + \\ &\quad 2 n_{BB} \ln p_B + n_{BO} (\ln p_B + \ln p_O) + n_{AB} (\ln p_A + \ln p_B) + 2 n_O \ln p_O \\ &= \ln c' + 2 n_{AA} \ln p_A + (n_A - n_{AA}) (\ln p_A + \ln p_O) + \\ &\quad 2 n_{BB} \ln p_B + (n_B - n_{BB}) (\ln p_B + \ln p_O) + n_{AB} (\ln p_A + \ln p_B) + 2 n_O \ln p_O, \end{aligned}$$

unde c și c' sunt constante care nu depind de parametrul p .

Funcția auxiliară se obține din expresia funcției de log-verosimilitate, prin aplicarea operatorului E , care se referă la media variabilelor neobservabile:

$$\begin{aligned} Q(p|p^{(t)}) &\stackrel{def.}{=} E[\ell(p)|n_{obs}; p^{(t)}] \\ &= \ln c' + 2 \hat{n}_{AA} \ln p_A + (n_A - \hat{n}_{AA}) (\ln p_A + \ln p_O) + \\ &\quad 2 \hat{n}_{BB} \ln p_B + (n_B - \hat{n}_{BB}) (\ln p_B + \ln p_O) + n_{AB} (\ln p_A + \ln p_B) + 2 n_O \ln p_O, \end{aligned}$$

⁵⁷³Observați că factorul de normalizare $\frac{n!}{n_{AA}! n_{AO}! n_{BB}! n_{BO}! n_{AB}! n_O!}$ generalizează C_n^k (combinări de n luate câte k), ceea ce este natural, întrucât distribuția multinomială este o generalizare a distribuției binomiale. (Vedeți problema 23 de la capitolul de *Fundamente*.)

unde

$$\begin{aligned}\hat{n}_{AA} &\stackrel{not.}{=} E[n_{AA}|n_{obs}; p^{(t)}] = E[n_{AA}|n_A, n_B, n_{AB}, n_O; p_A^{(t)}, p_B^{(t)}, p_O^{(t)}] \\ \hat{n}_{BB} &\stackrel{not.}{=} E[n_{BB}|n_{obs}; p^{(t)}] = E[n_{BB}|n_A, n_B, n_{AB}, n_O; p_A^{(t)}, p_B^{(t)}, p_O^{(t)}].\end{aligned}$$

Pasul E:

După cum se observă în expresia funcției auxiliare $Q(p|p^{(t)})$, acum trebuie să calculăm mediile \hat{n}_{AA} și \hat{n}_{BB} , adică numărul „așteptat” (engl., expected number) de persoane care au *fenotipul* (adică, perechea de alele) *AA*, și respectiv *BB*. Înțând cont că n_{AA} (sau, echivalent, fenotipul *AA*), văzut ca variabilă aleatoare, urmează *distribuția binomială* de parametri n_A și $\frac{(p_A^{(t)})^2}{(p_A^{(t)})^2 + 2 p_A^{(t)} p_O^{(t)}}$, rezultă că media sa este:⁵⁷⁴

$$\begin{aligned}\hat{n}_{AA} &\stackrel{not.}{=} E[n_{AA}|n_A, n_B, n_{AB}, n_O; p_A^{(t)}, p_B^{(t)}, p_O^{(t)}] \\ &= \frac{(p_A^{(t)})^2}{(p_A^{(t)})^2 + 2 p_A^{(t)} p_O^{(t)}} \cdot n_A\end{aligned}\tag{209}$$

În mod similar,

$$\begin{aligned}\hat{n}_{BB} &\stackrel{not.}{=} E[n_{BB}|n_A, n_B, n_{AB}, n_O; p_A^{(t)}, p_B^{(t)}, p_O^{(t)}] \\ &= \frac{(p_B^{(t)})^2}{(p_B^{(t)})^2 + 2 p_B^{(t)} p_O^{(t)}} \cdot n_B\end{aligned}\tag{210}$$

Pasul M:

Întrucât probabilitățile p_A , p_B și p_O trebuie să satisfacă restricția $p_A + p_B + p_O = 1$, vom aplica metoda multiplicatorilor lui Lagrange și vom introduce variabila / „multiplicatorul” $\lambda \in \mathbb{R}$ cu scopul de rezolvă *problema de optimizare* următoare:

$$\begin{aligned}p^{(t+1)} &\stackrel{not.}{=} (p_A^{(t+1)}, p_B^{(t+1)}, p_O^{(t+1)}) \\ &= \underset{p_A, p_B, p_O}{\operatorname{argmax}} [Q(p_A, p_B, p_O | p_A^{(t)}, p_B^{(t)}, p_O^{(t)}) + \lambda(1 - (p_A + p_B + p_O))] \\ &= \underset{p_A, p_B, p_O}{\operatorname{argmax}} [\ln c' + 2 \hat{n}_{AA} \ln p_A + (n_A - \hat{n}_{AA})(\ln p_A + \ln p_O) + 2 \hat{n}_{BB} \ln p_B + (n_B - \hat{n}_{BB})(\ln p_B + \ln p_O) + n_{AB}(\ln p_A + \ln p_B) + 2 n_O \ln p_O + \lambda(1 - (p_A + p_B + p_O))].\end{aligned}$$

Calculând derivatele parțiale ale *funcției obiectiv* din această problemă de optimizare în raport cu p_A , p_B și p_O și egalându-le apoi cu 0, vom avea:

$$\begin{aligned}\frac{1}{p_A}(2 \hat{n}_{AA} + n_A - \hat{n}_{AA} + n_{AB}) - \lambda &= 0 \Rightarrow \hat{p}_A = \frac{1}{\lambda}(\hat{n}_{AA} + n_A + n_{AB}) \\ \frac{1}{p_B}(2 \hat{n}_{BB} + n_B - \hat{n}_{BB} + n_{AB}) - \lambda &= 0 \Rightarrow \hat{p}_B = \frac{1}{\lambda}(\hat{n}_{BB} + n_B + n_{AB}) \\ \frac{1}{p_O}(n_A - \hat{n}_{AA} + n_B - \hat{n}_{BB} + 2 n_O) - \lambda &= 0 \Rightarrow \hat{p}_O = \frac{1}{\lambda}(n_A - \hat{n}_{AA} + n_B - \hat{n}_{BB} + 2 n_O).\end{aligned}$$

⁵⁷⁴Vedeți problema 23.b de la capitolul de *Fundamente*.

În continuare, aplicând restricția $\hat{p}_A + \hat{p}_B + \hat{p}_O = 1$, rezultă:

$$\frac{1}{\lambda}(\hat{n}_{AA} + n_A + n_{AB} + \cancel{\hat{n}_{BB}} + n_B + n_{AB} + n_A - \cancel{\hat{n}_{AA}} + n_B - \cancel{\hat{n}_{BB}} + 2n_O) = 1 \Leftrightarrow \\ \frac{2}{\lambda}(n_A + n_B + n_{AB} + n_O) = 1.$$

Întrucât $n = n_A + n_B + n_{AB} + n_O$, obținem imediat $\frac{1}{\lambda}2n = 1$ și, în consecință, $\lambda = 2n$. Înlocuindu-l pe λ cu această valoare ($2n$) în expresiile pe care le-am obținut anterior pentru \hat{p}_A , \hat{p}_B și \hat{p}_O , vom obține următoarele *reguli de actualizare* pentru pasul M al algoritmului EM:

$$\hat{p}_A^{(t+1)} = \frac{1}{2n}(\hat{n}_{AA} + n_A + n_{AB}) \quad (211)$$

$$\hat{p}_B^{(t+1)} = \frac{1}{2n}(\hat{n}_{BB} + n_B + n_{AB}) \quad (212)$$

$$\hat{p}_O^{(t+1)} = \frac{1}{2n}(n_A - \hat{n}_{AA} + n_B - \hat{n}_{BB} + 2n_O) = \frac{1}{2n}(n_{AO} + n_{BO} + 2n_O). \quad (213)$$

Se observă ușor că toate aceste valori sunt nenegative.

Sumarizând, relațiile de actualizare pentru algoritmul EM sunt (209) și (210) pentru pasul E, și respectiv (211), (212) și (213) pentru pasul M.

b. Rulând o implementare a algoritmului EM pe datele specificate în enunț și făcând inițializările indicate acolo (și menționate în următoarele tabele), am obținut rezultate:

Valori inițiale:

$$p_A = p_B = p_O = 1/3.$$

Iterații:

t	p_A	p_B	p_C
1	0.2505	0.0611	0.6884
2	0.2185	0.0505	0.7311
3	0.2142	0.0502	0.7357
4	0.2137	0.0501	0.7362
5	0.2136	0.0501	0.7363
6	0.2136	0.0501	0.7363

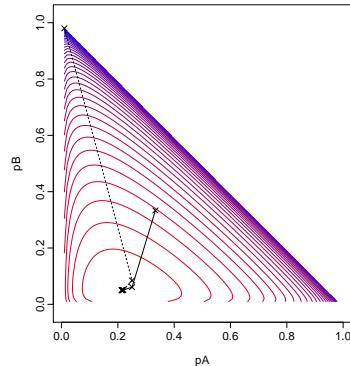
Valori inițiale:

$$p_A = p_O = 0.01, p_B = 0.98.$$

Iterații:

t	p_A	p_B	p_C
1	0.2505	0.0847	0.6648
2	0.2193	0.0511	0.7296
3	0.2143	0.0502	0.7355
4	0.2137	0.0501	0.7362
5	0.2136	0.0501	0.7363
6	0.2136	0.0501	0.7363

Se constată că rezultatele finale care au fost obținute de către algoritmul EM pornind de la cele două inițializări diferite sunt identice. Acest fapt era de așteptat, întrucât funcția de log-verosimilitate a datelor observabile este concavă și are un singur punct de maxim, aşa cum se observă și în graficul alăturat, realizat de către Sebastian Ciobanu.



8. (Algoritmul EM pentru estimarea parametrilor unei mixturi de două distribuții probabiliste categoriale; elaborare / ilustrare într-un un caz particular⁵⁷⁵)
prelucrare de A. Munteanu și L. Ciortuz, după CMU, 2009 spring, Ziv Bar-Joseph, HW5, pr. 1

Un student ia în fiecare dimineață autobuzul ca să vină la universitate. Dacă studentul ia autobuzul 71C, probabilitatea de a găsi un scaun liber lângă geam este μ_{11} , probabilitatea de a găsi un scaun liber lângă interval este μ_{12} , iar probabilitatea ca să călătorească în picioare este μ_{13} , unde $\mu_{11} + \mu_{12} + \mu_{13} = 1$. În schimb, dacă studentul ia autobuzul 500, probabilitățile corespunzătoare sunt μ_{21} , μ_{22} , μ_{23} , cu $\mu_{21} + \mu_{22} + \mu_{23} = 1$. Probabilitatea de a lua autobuzul 71C este β_1 , iar probabilitatea de a lua autobuzul 500 este β_2 , cu $\beta_1 + \beta_2 = 1$.

În timpul a n deplasări pe care le-a făcut la universitate, studentul și-a notat de fiecare dată poziția p_i pe care a ocupat-o în autobuz — pe un scaun de lângă geam (1), pe un scaun de lângă interval (2) și respectiv în picioare (3) — văzută ca valoare a variabilei aleatoare Pos_i , dar a omis să-și noteze și tipul autobuzului $B_i \in \{1, 2\}$, corespunzând numerelor 71C respectiv 500.

În acest exercițiu vi se va cere să folosiți algoritmul EM pentru a estima parametrii $\theta = <\mu_{11}, \mu_{12}, \mu_{13}, \mu_{21}, \mu_{22}, \mu_{23}, \beta_1, \beta_2>$ ai acestui model.

- Expremați $\log P(Pos, B | \theta)$, adică log-verosimilitatea datelor complete — observabile (Pos_i) și, respectiv, neobservabile (B_i) — în funcție de θ .
- Aplicând formula lui Bayes, calculați probabilitățile condiționate ale variabilelor neobservabile (B_i) în raport cu variabilele observabile (Pos_i) și cu $\theta^{(t)}$, care reprezintă valorile parametrilor θ la iterată t :

$$P(B_i | Pos_i, \theta^{(t)})$$

- Calculați expresia funcției „auxiliare“

$$Q(\theta | \theta^{(t)}) \stackrel{\text{def.}}{=} E_{P(B|Pos, \theta^{(t)})} [\log P(Pos, B | \theta)]$$

care reprezintă media log-verosimilității datelor complete ($\log P(Pos, B | \theta)$) în raport cu funcția de probabilitate condițională $P(B | Pos, \theta^{(t)})$.

- Calculați $\theta^{(t+1)} = (\mu_{11}^{(t+1)}, \mu_{12}^{(t+1)}, \mu_{13}^{(t+1)}, \mu_{21}^{(t+1)}, \mu_{22}^{(t+1)}, \mu_{23}^{(t+1)}, \beta_1^{(t+1)}, \beta_2^{(t+1)})$. Acestea sunt valorile parametrilor pentru care se atinge maximul expresiei $Q(\theta | \theta^{(t)})$:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)}).$$

Indicație: Aceste valori pot fi calculate cu ajutorul metodei multiplicatorilor Lagrange. Metoda aceasta va fi aplicată problemei de optimizare constând din

- funcția obiectiv $Q(\theta | \theta^{(t)})$
- restricțiile $\mu_{11} + \mu_{12} + \mu_{13} = 1$, $\mu_{21} + \mu_{22} + \mu_{23} = 1$ și $\beta_1 + \beta_2 = 1$, cu $\mu_{ij} \geq 0$ și $\beta_k \geq 0$.

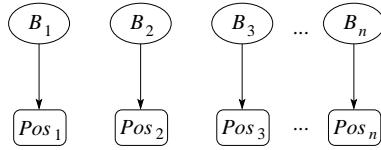
- Presupunem că, spre deosebire de situația din enunțul de mai sus, studentul a înregistrat numărul autobuzului B_i , dar nu a înregistrat poziția P_i pe care a avut-o în autobuz.

⁵⁷⁵Pentru varianta generală a algoritmului EM pentru rezolvare de mixturi de K distribuții categoriale, vedeți problema 17.

Este oare posibil ca și în această variantă a problemei să folosim algoritmul EM pentru a calcula parametrii modelului? Justificați.

Răspuns:

a. Cele n deplasări făcute de student cu autobuzul sunt [considerate] independente între ele. Prin urmare, nu există nicio condiționare între diversele variabile B_i . Mai departe, de valoarea acestor variabile ascunse / neobservabile depinde *poziția* liberă observată în autobuz Pos_i . Așadar, putem asocia acestei probleme *modelul grafic* următor:



Probabilitatea comună a datelor complete (observabile și neobservabile) în funcție de θ este:

$$P(Pos, B | \theta) \stackrel{indep.}{=} \prod_{i=1}^n P(Pos_i, B_i | \theta)$$

Aplicând regula de multiplicare, obținem:

$$P(Pos, B | \theta) = \prod_{i=1}^n P(Pos_i | B_i, \theta) \cdot P(B_i | \theta)$$

Pentru $i \in \{1, \dots, n\}$, $j \in \{1, 2\}$ și $k \in \{1, 2, 3\}$ considerăm următoarele *variabile indicator*:

$$b_{ij} = \begin{cases} 1 & \text{dacă } B_i = j \\ 0 & \text{altfel} \end{cases} \quad p_{ik} = \begin{cases} 1 & \text{dacă } Pos_i = k \\ 0 & \text{altfel} \end{cases}$$

Cu alte cuvinte b_{ij} este 1 dacă în ziua i studentul a mers cu autobuzul j , și 0 în caz contrar, cu convenția că valoarea $j = 1$ corespunde autobuzului 71C, iar $j = 2$ autobuzului 500. Similar, p_{ik} este 1 dacă în ziua i studentul a mers într-o poziție de tip k , unde $k = 1$ înseamnă pe scaun lângă geam, $k = 2$ – pe scaun lângă interval, iar $k = 3$ – în picioare. Cu aceste notății, putem exprima (în manieră compactă) verosimilitatea datelor complete astfel:

$$\begin{aligned} P(Pos, B | \theta) &= \prod_{i=1}^n P(B_i | \theta) \cdot P(Pos_i | B_i, \theta) = \prod_{i=1}^n \left(\prod_{j=1}^2 \beta_j^{b_{ij}} \prod_{k=1}^3 (u_{jk})^{b_{ij} p_{ik}} \right) \\ &= \prod_{i=1}^n \prod_{j=1}^2 \left(\beta_j \prod_{k=1}^3 u_{jk}^{p_{ik}} \right)^{b_{ij}} \end{aligned}$$

În consecință, log-verosimilitatea datelor complete este:

$$\log P(Pos, B | \theta) = \sum_{i=1}^n \sum_{j=1}^2 b_{ij} \left(\log \beta_j + \sum_{k=1}^3 p_{ik} \log u_{jk} \right)$$

b. Probabilitatea condiționată cerută este:

$$\begin{aligned} P(B_i = j \mid Pos_i = k, \theta^{(t)}) &\stackrel{T. Bayes}{=} \frac{P(Pos_i = k \mid B_i = j, \theta^{(t)})P(B_i = j \mid \theta^{(t)})}{\sum_{j'=1}^2 P(Pos_i = k \mid B_i = j', \theta^{(t)})P(B_i = j' \mid \theta^{(t)})} \\ &= \frac{\beta_j^{(t)} \prod_{k=1}^3 (\mu_{jk}^{(t)})^{p_{ik}}}{\sum_{j'=1}^2 \beta_{j'}^{(t)} \prod_{k=1}^3 (\mu_{j'k}^{(t)})^{p_{ik}}} \end{aligned}$$

Prin urmare, mediile variabilelor neobservabile b_{ij} la iterația t vor avea și ele aceeași expresie:

$$E[b_{ij}^{(t)}] \stackrel{not.}{=} E[b_{ij} \mid Pos, \theta^{(t)}] = \frac{\beta_j^{(t)} \prod_{k=1}^3 (\mu_{jk}^{(t)})^{p_{ik}}}{\sum_{j'=1}^2 \beta_{j'}^{(t)} \prod_{k=1}^3 (\mu_{j'k}^{(t)})^{p_{ik}}}$$

c. Calculăm media log-verosimilității datelor complete combinând rezultatele de la punctele a și b și ținând cont de proprietatea de liniaritate a mediilor:

$$Q(\theta \mid \theta^{(t)}) = \sum_{i=1}^n \sum_{j=1}^2 E[b_{ij}^{(t)}] \left(\log \beta_j + \sum_{k=1}^3 p_{ik} \log \mu_{jk} \right)$$

d. Pentru a calcula $\theta^{(t+1)}$, trebuie maximizată funcția $Q(\theta \mid \theta^{(t)})$ în raport cu θ . Vom calcula separat valorile optime ale parametrilor, și anume mai întâi pentru β_j , cu $j \in \{1, 2\}$ și apoi pentru μ_{jk} , cu $j \in \{1, 2\}$ și $k \in \{1, 2, 3\}$.

Stim că variabilele β_j se supun restricției $\beta_1 + \beta_2 = 1$. Reținând din expresia funcției $Q(\theta \mid \theta^{(t)})$ doar termenii care conțin variabilele β_j , rezultă că vom avea de optimizat funcția

$$\sum_{i=1}^n (E[b_{i1}^{(t)}] \log \beta_1 + E[b_{i2}^{(t)}] \log \beta_2) = \sum_{i=1}^n (E[b_{i1}^{(t)}] \log \beta_1 + E[b_{i2}^{(t)}] \log(1 - \beta_1))$$

Derivând ultima expresie în raport cu β_1 și egalând apoi cu 0, obținem:⁵⁷⁶

$$\begin{aligned} \sum_{i=1}^n \left(\frac{E[b_{i1}^{(t)}]}{\beta_1} - \frac{E[b_{i2}^{(t)}]}{1 - \beta_1} \right) &= 0 \Leftrightarrow (1 - \beta_1) \sum_{i=1}^n E[b_{i1}^{(t)}] = \beta_1 \sum_{i=1}^n E[b_{i2}^{(t)}] \Leftrightarrow \\ \sum_{i=1}^n E[b_{i1}^{(t)}] &= \beta_1 \sum_{i=1}^n (E[b_{i1}^{(t)}] + E[b_{i2}^{(t)}]) \Leftrightarrow \sum_{i=1}^n E[b_{i1}^{(t)}] = \beta_1 \underbrace{\sum_{i=1}^n E[b_{i1}^{(t)} + b_{i2}^{(t)}]}_1 \Leftrightarrow \\ \sum_{i=1}^n E[b_{i1}^{(t)}] &= n\beta_1 \Leftrightarrow \beta_1 = \frac{1}{n} \sum_{i=1}^n E[b_{i1}^{(t)}] \end{aligned}$$

Este imediat că

$$\beta_2 = \frac{1}{n} \sum_{i=1}^n E[b_{i2}^{(t)}],$$

iar atât β_1 cât și β_2 se situează în intervalul $[0, 1]$, întrucât mediile $E[b_{ij}^{(t)}]$ sunt de fapt tot niște probabilități.

⁵⁷⁶Se demonstrează ușor că luând baza logaritmului supraunitară (așa cum am procedat de-a lungul întregului capitol), rezultă că soluțiile derivatelor parțiale de ordinul întâi — care sunt calculate aici și mai jos — reprezintă într-adevăr punctul de maxim al funcției $Q(\theta \mid \theta^{(t)})$.

Pentru a calcula valorile optime ale parametrilor μ_{jk} , cu indicii $j \in \{1, 2\}$ și $k \in \{1, 2, 3\}$ vom apela la metoda multiplicatorilor Lagrange întrucât știm că acești parametri se supun constrângerii $\mu_{j1} + \mu_{j2} + \mu_{j3} = 1$. Așadar, vom avea problema de optimizare cu obiectivul $\max Q(\theta \mid \theta^{(t)})$ și restricțiile $\sum_{k=1}^3 \mu_{jk} - 1 = 0$ și $\mu_{ij} \geq 0$. Funcția Lagrange asociată acestei probleme este:

$$\Lambda(\mu, \lambda) = \sum_{i=1}^n \sum_{j=1}^2 \sum_{k=1}^3 E[b_{ij}^{(t)}] p_{ik} \log \mu_{jk} + \sum_{j=1}^2 \lambda_j \left(\sum_{k=1}^3 \mu_{jk} - 1 \right)$$

Egalând cu 0 derivata parțială a lui $\Lambda(\mu, \lambda)$ în raport cu μ_{jk} , obținem:

$$\frac{\partial \Lambda(\mu, \lambda)}{\partial \mu_{jk}} = 0 \Leftrightarrow \frac{1}{\mu_{jk}} \cdot \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik} + \lambda_j = 0 \Leftrightarrow \mu_{jk} = -\frac{1}{\lambda_j} \cdot \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik}, \text{ cu } \begin{array}{l} j \in \{1, 2\} \\ k \in \{1, 2, 3\} \end{array}$$

Știm că

$$\frac{\partial \Lambda(\mu, \lambda)}{\partial \lambda_j} = 0 \Leftrightarrow \sum_{k=1}^3 \mu_{jk} - 1 = 0, \text{ pentru } j \in \{1, 2\}$$

Dacă în această relație vom înlocui μ_{jk} cu valorile obținute anterior, vom determina valoarea lui λ_j pentru $j \in \{1, 2\}$:

$$\begin{aligned} \sum_{k=1}^3 \left(-\frac{1}{\lambda_j} \cdot \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik} \right) - 1 &= 0 \Leftrightarrow \frac{1}{\lambda_j} \cdot \sum_{k=1}^3 \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik} = -1 \\ &\Leftrightarrow \lambda_j = -\sum_{k=1}^3 \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik}, \text{ cu } j \in \{1, 2\}. \end{aligned}$$

Prin urmare, valorile căutate pentru μ_{jk} sunt:

$$\begin{aligned} \mu_{jk}^{(t+1)} &= -\frac{1}{\lambda_j} \cdot \sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik} = \frac{\sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik}}{\sum_{l=1}^3 \sum_{i=1}^n E[b_{ij}^{(t)}] p_{il}} \\ &= \frac{\sum_{i=1}^n E[b_{ij}^{(t)}] p_{ik}}{\sum_{i=1}^n \sum_{l=1}^3 E[b_{ij}^{(t)}] p_{il}}, \text{ cu } j \in \{1, 2\} \text{ și } k \in \{1, 2, 3\} \end{aligned}$$

Este imediat că aceste valori se situează în intervalul $[0, 1]$.

e. Așa cum se observă și din modelul grafic reprezentat la punctul a, dacă se cunoaște tipul autobuzului B_i , nu se pot estima parametrii [reprezentând distribuția] variabilei care reprezintă poziția Pos_i ocupată de student în autobuz. Altfel spus, variabilele observabile B_i nu oferă nicio informație relevantă pentru a putea calcula variabilele neobservabile Pos_i .

9. (O legătură între clasificatorul Bayes Naiv și algoritmul EM:
rezolvarea unei mixturi de distribuții categoriale multidimensionale,
folosind presupozitia de independentă condițională
de tip Bayes Naiv⁵⁷⁷)
*prelucrare de Liviu Ciortuz, după
CMU, 2014 spring, A. Singh, B. Poczos, HW3, pr. 2.1*

Fie setul de date etichetate $\mathcal{D} = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$. Atunci când se folosește presupozitia de independentă condițională specifică clasificatorului Bayes Naiv, putem scrie verosimilitatea [comună a] datelor \mathcal{D} astfel:

$$P(\mathcal{D}) \stackrel{i.i.d.}{=} \prod_{i=1}^n P(X_i, Y_i) = \prod_{i=1}^n (P(X_i|Y_i) \cdot P(Y_i)) = \prod_{i=1}^n \left(P(Y_i) \cdot \prod_{j=1}^d P(X_i^j|Y_i) \right)$$

Notăția X_i^j , unde $j \in \{1, \dots, d\}$, desemnează atributul de intrare j al instanței etichetate (X_i, Y_i) .

Observație: În cele ce urmează vom nota variabilele aleatoare cu majuscule, iar valorile lor cu minuscule.⁵⁷⁸

Presupunem că $y_i \in \{0, 1\} \forall i$ și $x_i^j \in \{1, 2, \dots, V\} \forall i, j$. Așadar, considerăm (doar pentru simplitate) că V , numărul de valori, este același pentru toate atributele X^j , cu $j \in \{1, \dots, d\}$, și [în plus] multimele de valori ale acestor atribută coincid. Vom nota $\pi_0 = P(Y = 0)$, $\pi_1 = P(Y = 1)$ și $\pi = (\pi_0, \pi_1)$. Similar, pentru fiecare $j \in \{1, \dots, d\}$, vom nota parametrii distribuțiilor $P(X^j|Y)$ astfel: $\beta_0^j = (\beta_{10}^j, \beta_{20}^j, \dots, \beta_{V0}^j)$ și $\beta_1^j = (\beta_{11}^j, \beta_{21}^j, \dots, \beta_{V1}^j)$, deci $\beta_{k0}^j \stackrel{not.}{=} P(X^j = k|Y = 0)$ și $\beta_{k1}^j \stackrel{not.}{=} P(X^j = k|Y = 1)$. În sfârșit, $\beta \stackrel{not.}{=} (\beta_0^1, \beta_1^1, \dots, \beta_0^d, \beta_1^d)$ și $\theta \stackrel{not.}{=} (\pi, \beta)$.

Evident, în cazul în care dispunem de date etichetate — adică variabila Y_i este „observabilă“, pentru toate valorile lui i —, vom putea obține ușor estimări (în sensul MLE) pentru parametrii π și β , folosind procedeul clasic de enumerare (engl., counting) și normalizare. În acest exercițiu vom presupune însă că nu dispunem de date etichetate. *Scopul* nostru este de a arăta că și în situația aceasta putem estima parametrii θ (adică π și β^j , cu $j = 1, \dots, d$), și anume folosind algoritmul de estimare-maximizare (EM).

a. Se știe — vedeti problema 1 — că funcția de log-verosimilitate a datelor observabile ($\log P(x_1, \dots, x_n|\theta)$) este minorată de o funcție de două argumente $F(q, \theta)$, unde q este o distribuție probabilistică oarecare definită peste variabilele-indicator z_i .⁵⁷⁹

⁵⁷⁷ Pentru cazul când în loc de 2 distribuții categoriale multidimensionale se consideră K distribuții Bernoulli multidimensionale, vedeti problema CMU, 2015 fall, A. Smola, B. Poczos, HW2, pr. 1, 2.

În cazul $d = 1$ (vedeti notațiile din problemă), se obține varianta de algoritm EM pentru o mixtură de două distribuții categoriale, care a fost deja ilustrată la problema 8. Pentru mixturi de K distribuții categoriale, vedeti problema 4.

Pentru un exercițiu de tip *implementare* care face clusterizare de caractere scrise de mână, aplicând un algoritm EM de tipul celui prezentat în problema de față, însă lucrând cu variabile Bernoulli în locul variabilelor categoriale, vedeti CMU, 2015 fall, A. Smola, B. Poczos, HW2, pr. 1, 2.

⁵⁷⁸ De exemplu, Y_i este o variabilă aleatoare, iar y_i desemnează o instanțiere a lui Y_i la una din valorile ei posibile.

⁵⁷⁹ *Observație:* Veți ține cont că rolul celor variabile z_i (de la problema 1) va fi jucat aici de către y_i .

Vă cerem⁵⁸⁰

- să scrieți expresia funcției de (log-)verosimilitate pe care urmărește să o maximizeze algoritmul EM în contextul prezentului exercițiu;⁵⁸¹
- să introduceti apoi probabilitățile $q(y_i)$ în mod forțat — însă într-o manieră convenabilă — în expresia scrisă anterior pentru funcția de log-verosimilitate;
- și, în sârșit, folosind inegalitatea lui Jensen, să obțineți expresia marginii inferioare $F(q, \theta)$ pentru funcția de log-verosimilitate care a fost menționată mai sus.

Observație importantă: Spre deosebire de cum am procedat la toate exercițiile de până acum legate de algoritmul EM, aici nu vom [mai] lucra cu funcția „auxiliară“ Q ,⁵⁸² ci direct cu funcția F .

- b. Pasul E: Calculați probabilitatea fiecarei asignări posibile [la o anumită clasă] pentru fiecare instanță, date fiind valorile actuale ale parametrilor π și β . Concret, veți determina valorile optime pentru $q(y_i)$ (i.e., $q(Y_i = 0)$ și $q(Y_i = 1)$), folosind derivele partiale ale funcției $F(q(y_i), \theta)$ care a fost dedusă la punctul precedent.
- c. Pasul M: Deducreți regulile pentru actualizarea parametrilor π și β în funcție de probabilitățile de asignare [la o anumită clasă] pentru fiecare instanță, $q(y_i)$, care au fost calculate la pasul E.
- d. Schițați o strategie convenabilă pentru inițializarea parametrilor (π și β) și indicați motivele care au stat la baza alegерii / conceperii respectivei strategii.

Răspuns:

- a. Algoritmul EM urmărește să maximizeze funcția de log-verosimilitate a datelor observabile, $\log P(x_1, \dots, x_n | \theta)$. Fără a mai menționa parametrul θ în calculul care urmează, putem scrie:

$$\begin{aligned}
\log P(x_1, \dots, x_n) &\stackrel{i.i.d.}{=} \log \prod_{i=1}^n P(x_i) = \log \prod_{i=1}^n \sum_{y_i \in \{0,1\}} P(x_i, y_i) \\
&= \sum_{i=1}^n \log \sum_{y_i \in \{0,1\}} P(x_i, y_i) = \sum_{i=1}^n \log \sum_{y_i \in \{0,1\}} P(x_i | y_i) P(y_i) \\
&= \sum_{i=1}^n \log \sum_{y_i \in \{0,1\}} P(y_i) \prod_{j=1}^d P(x_i^j | y_i) \\
&= \sum_{i=1}^n \log \sum_{y_i \in \{0,1\}} q(y_i) \frac{P(y_i) \prod_{j=1}^d P(x_i^j | y_i)}{q(y_i)} \\
&\stackrel{\text{def.}}{=} \sum_{i=1}^n \log E_q \left[\frac{P(y_i) \prod_{j=1}^d P(x_i^j | y_i)}{q(y_i)} \right] \\
&\geq \sum_{i=1}^n E_q \left[\log \frac{P(y_i) \prod_{j=1}^d P(x_i^j | y_i)}{q(y_i)} \right] \quad (\text{Jensen})
\end{aligned}$$

⁵⁸⁰ Sugestie: Procedați în mod similar cu rezolvarea dată la problema 1 punctul a.

⁵⁸¹ Indicație: Verosimilitatea unei instanțe neetichetate, $P(x_i | \theta)$ se poate obține însumând probabilitățile corespunzătoare tuturor asignărilor posibile y_i ale variabilei latente / neobservabile Y_i .

⁵⁸² Vedeti finalul rezolvării problemei 1 de la acest capitol.

$$\begin{aligned}
&\stackrel{\text{def.}}{=} \sum_{i=1}^n \sum_{y_i \in \{0,1\}} q(y_i) \log \frac{P(y_i) \prod_{j=1}^d P(x_i^j | y_i)}{q(y_i)} \\
&= \sum_{i=1}^n \sum_{y_i \in \{0,1\}} q(y_i) \left[\log P(y_i) + \left(\sum_{j=1}^d \log P(x_i^j | y_i) \right) - \log q(y_i) \right].
\end{aligned}$$

Așadar, reintroducând parametrul θ , marginea inferioară a funcției de log-verosimilitate a datelor observabile x_1, \dots, x_n este:

$$F(q, \theta) = \sum_{i=1}^n \sum_{y_i \in \{0,1\}} q(y_i) \left[\log P(y_i | \theta) + \left(\sum_{j=1}^d \log P(x_i^j | y_i, \theta) \right) - \log q(y_i) \right]. \quad (214)$$

b. La pasul E, pentru $i = 1, \dots, n$, vom actualiza $q(y_i)$ în funcție de valorile curente pentru parametrii π și β , mai exact în funcție de $P_\pi(y_i) \stackrel{\text{not.}}{=} P(y_i | \pi)$ și $P_\beta(x_i^j | y_i) \stackrel{\text{not.}}{=} P(x_i^j | y_i, \beta)$.⁵⁸³ Calculând derivata parțială a funcției F în raport cu $q(y_i)$, obținem:

$$\log P_\pi(y_i) + \sum_{j=1}^d \log P_\beta(x_i^j | y_i) - \log q(y_i) - 1,$$

iar apoi egalându-o cu 0, deducem ușor următoarele reguli de actualizare:

$$\begin{aligned}
y_i = 0 : \quad q(Y_i = 0) &\propto P_\pi(Y_i = 0) \prod_{j=1}^d P_\beta(x_i^j | Y_i = 0) \\
y_i = 1 : \quad q(Y_i = 1) &\propto P_\pi(Y_i = 1) \prod_{j=1}^d P_\beta(x_i^j | Y_i = 1),
\end{aligned}$$

unde simbolul \propto înseamnă *proporțional cu*. Se poate observa ușor că factorul de proporționalitate în precedentele două relații este același. Așadar, ținând cont de restricția $q(y_i = 0) + q(y_i = 1) = 1$, putem scrie:

$$\begin{aligned}
q(Y_i = 0) &= \frac{P_\pi(Y_i = 0) \prod_{j=1}^d P_\beta(x_i^j | Y_i = 0)}{\sum_{\ell \in \{0,1\}} P_\pi(Y_i = \ell) \prod_{j=1}^d P_\beta(x_i^j | Y_i = \ell)} \\
q(Y_i = 1) &= \frac{P_\pi(Y_i = 1) \prod_{j=1}^d P_\beta(x_i^j | Y_i = 1)}{\sum_{\ell \in \{0,1\}} P_\pi(Y_i = \ell) \prod_{j=1}^d P_\beta(x_i^j | Y_i = \ell)}
\end{aligned}$$

c. La pasul M, cunoaștem valorile $q(y_i)$ și vom maximiza funcția [obiectiv] $F(q, \theta)$ în raport cu parametrul θ (i.e., π și β). După calcule relativ simple, cu care de acum suntem

⁵⁸³ Pentru a determina valorile optime pentru $q(y_i = 0)$ și $q(y_i = 1)$, am putea proceda ca și la exercițiile precedente:

i. fie ținând cont de relația $q(y_i = 0) + q(y_i = 1) = 1$, $\forall i \in \{1, \dots, n\}$ și înlocuind, de exemplu, $q(y_i = 1)$ în funcție de $q(y_i = 0)$ în relația (214), după care calculăm rădăcina derivatei $\frac{\partial F(q, \theta)}{\partial q(y_i = 0)}$ și verificăm dacă această rădăcină aparține intervalului $[0, 1]$;

ii. fie folosim metoda multiplicatorilor Lagrange pentru a rezolva (pentru fiecare $i \in \{1, \dots, n\}$) problema de optimizare care are funcția obiectiv (214) și restricția $q(y_i = 0) + q(y_i = 1) = 1$.

Însă rezolvarea pentru care am optat aici este de un alt tip, care într-o anumită măsură seamănă cu metoda *i*, dar inversează cei doi pași.

obișnuiți, vom obține următoarele reguli de actualizare pentru parametrii π :⁵⁸⁴

$$\begin{cases} \pi_0 \stackrel{\text{not.}}{=} P(Y = 0) \propto \sum_{i=1}^n q(Y_i = 0) \\ \pi_1 \stackrel{\text{not.}}{=} P(Y = 1) \propto \sum_{i=1}^n q(Y_i = 1) \end{cases} \Rightarrow \begin{cases} \pi_0 = \frac{1}{n} \sum_{i=1}^n q(Y_i = 0) \\ \pi_1 = \frac{1}{n} \sum_{i=1}^n q(Y_i = 1) \end{cases}$$

și apoi pentru parametrii β :⁵⁸⁵

$$\begin{aligned} & \begin{cases} \beta_{v0}^j \stackrel{\text{not.}}{=} P(X^j = v | Y = 0) \propto \sum_{i=1}^n q(Y_i = 0) \cdot 1_{\{x_i^j = v\}} \\ \beta_{v1}^j \stackrel{\text{not.}}{=} P(X^j = v | Y = 1) \propto \sum_{i=1}^n q(Y_i = 1) \cdot 1_{\{x_i^j = v\}}, \end{cases} \\ & \Rightarrow \begin{cases} \beta_{v0}^j = \frac{\sum_{i=1}^n q(Y_i = 0) \cdot 1_{\{x_i^j = v\}}}{\sum_{\ell \in \{0,1\}} \sum_{i=1}^n q(Y_i = \ell) \cdot 1_{\{x_i^j = v\}}} \\ \beta_{v1}^j = \frac{\sum_{i=1}^n q(Y_i = 1) \cdot 1_{\{x_i^j = v\}}}{\sum_{\ell \in \{0,1\}} \sum_{i=1}^n q(Y_i = \ell) \cdot 1_{\{x_i^j = v\}}} \end{cases} \end{aligned}$$

Notația $1_{\{\dots\}}$ folosită mai sus desemnează o *funcție-indicator*. Așa cum știm, ea ia valoarea 1 atunci când condiția specificată [între acolade] este satisfăcută și valoarea 0 în caz contrar.

Este interesant *de observat* că relațiile obținute la acest pas al algoritmului EM sunt foarte similară cu relațiile corespunzătoare folosite [pentru estimarea parametrilor] de către clasificatorul Bayes Naiv (supervizat).

d. O strategie convenabilă pentru inițializarea parametrilor mixturii ar trebui să țină cont de cunoștințele / informațiile pe care le avem în legătură cu o soluție plauzibilă. Dacă avem motive să credem că o anumită clasă este mai probabilă decât cealaltă clasă, vom putea „injecta“ această informație în valoarea folosită la inițializarea lui π . Pentru parametrii β , dacă intuim că o anumită valoare (k) a atributului X^j se coreleză mai bine cu (adică este mai „indicativă“ în raport cu) o anumită clasă ℓ , atunci vom da o valoare (probabilitate) mai mare parametrului $\beta_{k\ell}^j$, care corespunde respectivei valori a atributului X^j . Dacă nu avem niciun fel de informații / cunoștințe a priori, putem să inițializăm acești parametri în mod aleator.

⁵⁸⁴Pentru actualizarea parametrilor π_0 și π_1 , putem folosi oricare dintre metodele *i* și *ii* menționate la nota de subsol precedentă.

În cazul *i*, reținând din expresia (214) doar termenii care-l conțin pe $P(Y = y_i | \theta)$, vom avea:

$$\frac{\partial}{\partial P_\pi(Y = 0)} \left(\sum_{i=1}^n (q(y_i = 0) \log P_\pi(Y = 0) + q(y_i = 1) \log(1 - P_\pi(Y = 0))) \right) = 0 \Leftrightarrow \text{s.a.m.d.}$$

În cazul *ii*, lagrangeanul de optimizat va fi

$$\begin{aligned} \mathcal{L}(\lambda) &= F(q, \theta) + \lambda(1 - P_\pi(Y = 0) - P_\pi(Y = 1)), \text{ deci} \\ \frac{\partial \mathcal{L}(\lambda)}{\partial P_\pi(Y = 0)} &= 0 \Leftrightarrow \frac{1}{P_\pi(Y = 0)} \sum_{i=1}^n q(Y_i = 0) = \lambda \Leftrightarrow P_\pi(Y = 0) = \frac{1}{\lambda} \sum_{i=1}^n q(Y_i = 0) \end{aligned}$$

și similar pentru $P_\pi(Y = 1)$.

⁵⁸⁵Folosim metoda multiplicatorilor lui Lagrange (*ii*), din cauza restricțiilor $\sum_{v=1}^V \beta_{v0}^j = 1$ și $\sum_{v=1}^V \beta_{v1}^j = 1$. Detaliile de calcul sunt similare cu cele din ultima parte de la nota de subsol precedentă.

10. (Algoritmul EM pentru mixturi de distribuții categoriale;
 [aplicare la] identificarea domeniilor semantice
 asociate cuvintelor dintr-un document-text)
 ■ CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 3

În acest exercițiu vi se va cere să deduceți relațiile corespunzătoare pașilor E și M din corpul iterativ al algoritmului EM pentru „modelarea“ variabilelor „neobservabile“ care desemnează *domeniile semantice* (engl., topics) implicate în generarea unui document oarecare de tip text.

Vom considera că fiecare *cuvânt* din documentul dat este reprezentat de o variabilă aleatoare w care poate lua valorile $1, \dots, V$ relativ la un *vocabulary* dat. De fapt, în cele ce urmează vom desemna fiecare cuvânt w printr-un *vector-indicator* format din V componente astfel încât $w(i) = 1$ dacă w ia valoarea cuvântului de pe poziția i din vocabular și 0 în caz contrar. Așadar, $\sum_{i=1}^V w(i) = 1$.

Dacă fiind un *document* constituit din cuvintele w_j , $j = 1, \dots, N$, unde N este lungimea documentului, vom presupune că aceste cuvinte sunt generate de către o mixtură de K distribuții categoriale:

$$\begin{aligned} P(w_j) &= \sum_{s=1}^K \pi_s P(w_j | \beta_s) \\ P(w_j | \beta_s) &= \prod_{i=1}^V P(w_j(i) = 1 | Z_j = s) = \prod_{i=1}^V \beta_s(i)^{w_j(i)}, \end{aligned}$$

unde

$\pi_s \stackrel{\text{not.}}{=} P(Z = s)$ este probabilitatea (a priori) ca variabila latentă Z , care desemnează domeniul semantic asociat unui cuvânt oarecare, să ia valoarea s ;

$\beta_s \stackrel{\text{not.}}{=} (\beta_s(1), \dots, \beta_s(i), \dots, \beta_s(V))$, cu $\beta_s(i) \geq 0$ pentru $i = 1, \dots, V$ și $\sum_{i=1}^V \beta_s(i) = 1$, pentru fiecare $s = 1, \dots, K$. Vectorul β_s determină distribuția categorială asociată cu domeniul semantic s ;

$\beta_s(i) \stackrel{\text{not.}}{=} P(w_j(i) = 1 | Z_j = s)$ desemnează probabilitatea ca $w_j \stackrel{\text{not.}}{=} (w_j(1), \dots, w_j(V))$ să fi fost generat de către domeniul semantic s .

Observație: Remarcăți faptul că acest model nu ține cont de ordinea / asocierea cuvintelor din documentul considerat. Deși faptul aceasta constituie o încălcare evidentă a proprietăților definitoare ale unui text, modelul propus aici are totuși o utilitate practică dovedită.⁵⁸⁶

a. Referitor la pasul E al algoritmului EM, pentru fiecare cuvânt w_j , calculați $q_j(s) \stackrel{\text{not.}}{=} P(Z_j = s | w_j; \theta')$, probabilitatea (a posteriori) ca, dat un cuvânt w_j , acesta să corespundă unui anumit domeniu semantic s din ansamblul celor K domenii semantice considerate. Prin θ desemnăm ansamblul parametrilor din modelul de mixtură considerat: π_s și β_s cu $s = 1, \dots, K$, iar θ' notează valoarea acestor parametri la pasul de inițializare [al algoritmului EM], respectiv valoarea obținută la iteratărea precedentă.

b. Pentru pasul M, determinați valoarea parametrului θ care maximizează marginea inferioară (vedeți problema 1) pentru log-verosimilitatea datelor observabile, i.e., a cuvintelor din documentul dat:

⁵⁸⁶Vedeți capitolul *Word Sense Disambiguation* din carteau *Foundations of Statistical Natural Language Processing*, Christopher Manning, Hinrich Schütze, MIT Press, 2002, pag 252-256.

$$\ell(w|\theta) \stackrel{not.}{=} \log \prod_{j=1}^N P(w_j|\theta).$$

Indicație: Procedând în mod clasic, adică sumând în raport cu variabila latentă s care reprezintă tematica / domeniul semantic, putem scrie funcția de log-verosimilitate $\ell(w|\theta)$ astfel:

$$\ell(w|\theta) = \sum_{j=1}^N \log \sum_s P(w_j, s|\theta) = \sum_{j=1}^N \log \sum_s q_j(s) \frac{P(w_j, s|\theta)}{q_j(s)},$$

unde cantitățile $q_j(s)$ cu $j \in \{1, \dots, V\}$ și $s \in \{1, \dots, K\}$ au fost calculate la pasul E al iterării curente. Mai departe, folosind inegalitatea lui Jensen (vedeți pr. 71 de la capitolul *Fundamente*), obținem:

$$\begin{aligned} \ell(w|\theta) &\geq \sum_{j=1}^N \sum_s q_j(s) \log \frac{P(w_j, s|\theta)}{q_j(s)} = \\ &= \sum_{j=1}^N \sum_s q_j(s) \log P(w_j, s|\theta) - \sum_{j=1}^N \sum_s q_j(s) \log q_j(s) = \\ &= \sum_{j=1}^N \sum_s q_j(s) \log P(w_j, s|\theta) + \sum_{j=1}^N H(q_j). \end{aligned}$$

Am notat cu $H(q_j)$ entropia $\sum_s q_j(s) \log q_j(s) = \sum_s P(s|w_j; \theta') \log P(s|w_j; \theta')$.⁵⁸⁷ Evident, aceasta nu depinde de θ . În concluzie, la pasul M veți calcula acea valoare a lui θ pentru care se atinge maximul *funcției auxiliare*

$$Q(\theta|\theta') \stackrel{not.}{=} \sum_{j=1}^N \sum_s q_j(s) \log P(w_j, s|\theta).$$

Răspuns:

- a. La pasul E, calculăm probabilitățile a posteriori $q_j(s) \stackrel{not.}{=} P(Z_j = s|w_j; \theta')$. Pentru conveniență, în cele ce urmează vom renunța la a folosi accentul $'$ pentru componentele din setul de parametri $\theta' = (\pi', \beta'_1, \dots, \beta'_K)$ indicat în enunț — ar fi trebuit, de fapt, să scriem $\theta^{(t)} = (\pi^{(t)}, \beta_1^{(t)}, \dots, \beta_K^{(t)})$ — deoarece notațiile ar deveni prea complicate. Cititorul atent va ști din context la ce anume se referă notarea $\theta = (\pi, \beta_1, \dots, \beta_K)$.

$$\begin{aligned} q_j(Z_j = s) &\stackrel{not.}{=} P(Z_j = s|w_j; \theta) \stackrel{\text{Bayes T.}}{=} \frac{P(w_j|Z_j = s; \theta) P(Z_j = s|\theta)}{P(w_j|\theta)} \\ &= \frac{\pi_s P(w_j|\beta_s)}{\sum_{s'=1}^K \pi_{s'} P(w_j|\beta_{s'})} = \frac{\pi_s \prod_{l=1}^V \beta_s(l)^{w_j(l)}}{\sum_{s'=1}^K \pi_{s'} \prod_{l=1}^V \beta_{s'}(l)^{w_j(l)}}. \end{aligned}$$

- b. Conform *Indicației* din enunț, calculăm mai întâi funcția auxiliară asociată iterării curente (t):

$$Q(\theta|\theta') \stackrel{not.}{=} \sum_{j=1}^N \sum_{s=1}^K q_j(Z_j = s) \log P(w_j, Z_j = s|\theta)$$

⁵⁸⁷Pentru simplitate, am presupus că atât funcția de log-verosimilitate cât și entropia se calculează folosind pentru logaritm o aceeași bază, supraunitară.

$$\begin{aligned}
&= \operatorname{argmax}_{\theta} \sum_{j=1}^N \sum_{s=1}^K q_j(Z_j = s) \log P(w_j | Z_j = s; \theta) P(Z_j = s | \theta) \\
&= \operatorname{argmax}_{\theta} \sum_{j=1}^N \sum_{s=1}^K q_j(Z_j = s) \log \left(\pi_s \prod_{l=1}^V \beta_s(l)^{w_j(l)} \right) \\
&= \operatorname{argmax}_{\theta} \sum_{j=1}^N \sum_{s=1}^K [q_j(Z_j = s) \log \pi_s + q_j(Z_j = s) \sum_{l=1}^V \log \beta_s(l)^{w_j(l)}] \\
&= \operatorname{argmax}_{\theta} \sum_{j=1}^N \sum_{s=1}^K [q_j(Z_j = s) \log \pi_s + q_j(Z_j = s) \sum_{l=1}^V w_j(l) \log \beta_s(l)]. \quad (215)
\end{aligned}$$

Se știe că $\theta^{(t+1)} \stackrel{\text{def.}}{=} \operatorname{argmax}_{\theta} Q(\theta | \theta')$. Problema aceasta de optimizare se poate rezolva maximizând separat $Q(\theta | \theta')$ în raport cu parametrii $\beta_s(l)$ și respectiv în raport cu parametrii π_s (unde $s \in \{1, \dots, K\}$).⁵⁸⁸

În ce privește optimizarea lui $Q(\theta | \theta')$ în raport cu $\beta_s(l)$:

După ce eliminăm din expresia (215) termenii care sunt constanți în raport cu β_s , obținem:

$$\sum_{j=1}^N q_j(Z_j = s) \sum_{l=1}^V w_j(l) \log \beta_s(l).$$

Apoi, deoarece $\beta_s \stackrel{\text{not.}}{=} (\beta_s(1), \dots, \beta_s(i), \dots, \beta_s(V))$, cu restricțiile $\beta_s(i) \geq 0$ pentru $i = 1, \dots, V$ și $\sum_{i=1}^V \beta_s(i) = 1$, vom folosi metoda multiplicatorilor lui Lagrange pentru a ne asigura că cea de-a doua restricție este satisfăcută.⁵⁸⁹ Așadar, introducând variabila / multiplicatorul Lagrange $\lambda \in \mathbb{R}$, pornind de la expresia anterior obținută vom scrie funcția „lagrangeană“

$$\mathcal{L}(\beta_s(l)) = \sum_{j=1}^N q_j(Z_j = s) \sum_{l=1}^V w_j(l) \log \beta_s(l) + \lambda \left(\sum_{l=1}^V \beta_s(l) - 1 \right),$$

pe care o vom optimiza – prin derivare – în raport cu $\beta_s(l)$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_s(l)} \mathcal{L}(\beta_s(l)) &= 0 \Leftrightarrow \sum_{j=1}^N q_j(Z_j = s) \frac{w_j(l)}{\beta_s(l)} + \lambda = 0 \\
\Leftrightarrow \frac{1}{\beta_s(l)} \sum_{j=1}^N q_j(Z_j = s) w_j(l) + \lambda &= 0 \Leftrightarrow \frac{1}{\beta_s(l)} = \frac{-\lambda}{\sum_{j=1}^N q_j(Z_j = s) w_j(l)} \\
\Leftrightarrow \beta_s(l) &= \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{-\lambda}. \quad (216)
\end{aligned}$$

Întrucât $\sum_{l=1}^V \beta_s(l) = 1$, din relația (216) va rezulta

$$\sum_{l=1}^V \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{-\lambda} = 1 \Leftrightarrow -\lambda = \sum_{l=1}^V \sum_{j=1}^N q_j(Z_j = s) w_j(l)$$

⁵⁸⁸Cititorul pedant va putea observa după calculele de mai jos că matricea hessiană a funcției auxiliare $Q(\theta | \theta')$ este diagonală și negativ definită, deci $Q(\theta | \theta')$ este funcție concavă, așadar ea admite o singură valoare maximă, care este globală.

⁵⁸⁹Vezi vedea că soluția obținută ulterior va satisface și prima restricție.

Substituind această valoare a lui $-\lambda$ în relația (216), obținem ceea ce va constitui *noua valoare* a parametrului $\beta_s(l)$, pentru iterată $t + 1$:

$$\begin{aligned}\beta_s(l) &= \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{\sum_{l=1}^V \sum_{j=1}^N q_j(Z_j = s) w_j(l)} = \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{\sum_{j=1}^N \sum_{l=1}^V q_j(Z_j = s) w_j(l)} \\ &= \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{\sum_{j=1}^N q_j(Z_j = s) \underbrace{\sum_{l=1}^V w_j(l)}_1} = \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{\sum_{j=1}^N q_j(Z_j = s)}.\end{aligned}$$

În mod intuitiv, expresia pe care tocmai am obținut-o (și care este, evident, nenegativă) poate fi interpretată ca fiind „ponderea“ care îi revine cuvântului de pe poziția l din vocabular — fiindcă $w(l) = 1$ — în ansamblul [clusterului corespunzător] domeniului semantic s .

În mod similar, pentru a optimiza funcția auxiliară $Q(\theta|\theta')$ în raport cu probabilitatea de selecție π_s , vom începe prin a elimina din expresia (215) termenii care sunt constanți în raport cu π_s . Expresia care rezultă este:

$$\sum_{j=1}^N q_j(Z_j = s) \log \pi_s.$$

După aceea, folosind din nou metoda multiplicatorilor lui Lagrange pentru a impune satisfacerea restricției $\sum_{s=1}^K \pi_s = 1$, vom scrie funcția „lagrangeană“

$$\mathcal{L}(\pi_s) = \sum_{j=1}^N q_j(Z_j = s) \log \pi_s + \lambda \left(\sum_{s=1}^K \pi_s - 1 \right).$$

Valoarea optimă pentru π_s va fi obținută cu ajutorul derivatei parțiale:

$$\begin{aligned}\frac{\partial}{\partial \pi_s} \mathcal{L}(\pi_s) = 0 &\Leftrightarrow \sum_{j=1}^N \frac{q_j(Z_j = s)}{\pi_s} + \lambda = 0 \Leftrightarrow \frac{1}{\pi_s} \sum_{j=1}^N q_j(Z_j = s) = -\lambda \\ &\Leftrightarrow \pi_s = \frac{\sum_{j=1}^N q_j(Z_j = s)}{-\lambda}.\end{aligned}\tag{217}$$

Întrucât $\sum_{s=1}^K \pi_s = 1$, vom avea:

$$\begin{aligned}\sum_{s=1}^K \frac{\sum_{j=1}^N q_j(Z_j = s)}{-\lambda} = 1 &\Leftrightarrow \frac{1}{-\lambda} \sum_{s=1}^K \sum_{j=1}^N q_j(Z_j = s) = 1 \\ &\Leftrightarrow -\lambda = \sum_{s=1}^K \sum_{j=1}^N q_j(Z_j = s).\end{aligned}$$

Substituind această valoare a lui $-\lambda$ în expresia (217), obținem *noua valoare* a parametrului π_s :

$$\pi_s = \frac{\sum_{j=1}^N q_j(Z_j = s)}{\sum_{s=1}^K \sum_{j=1}^N q_j(Z_j = s)} = \frac{\sum_{j=1}^N q_j(Z_j = s)}{\sum_{j=1}^N \underbrace{\sum_{s=1}^K q_j(Z_j = s)}_1} = \frac{\sum_{j=1}^N q_j(Z_j = s)}{N}.$$

În mod intuitiv, această ultimă expresie (care este, evident, nenegativă) poate fi interpretată ca fiind „ponderea“ care revine clusterului s din ansamblul celor N cuvinte din vocabular.

Sumarizând, *regulile de actualizare* de la iterația t a algoritmului EM sunt

Pasul E:

$$q_j(Z_j = s) = \frac{\pi_s \prod_{l=1}^V \beta_s(l)^{w_j(l)}}{\sum_{s'=1}^K \pi_{s'} \prod_{l=1}^V \beta_{s'}(l)^{w_j(l)}} \text{ pentru } j = 1, \dots, N \text{ și } s = 1, \dots, K,$$

unde π_s , $\pi_{s'}$, β_s și $\beta_{s'}$ provin de la pasul M al iterației $t - 1$ atunci când $t > 1$ (și respectiv de la inițializare, în cazul iterației $t = 1$);

Pasul M:

$$\beta_s(l) = \frac{\sum_{j=1}^N q_j(Z_j = s) w_j(l)}{\sum_{j=1}^N q_j(Z_j = s)} \text{ pentru } s = 1, \dots, K \text{ și } l = 1, \dots, V$$

$$\pi_s = \frac{\sum_{j=1}^N q_j(Z_j = s)}{N} \text{ pentru } s = 1, \dots, K.$$

11.

(Algoritmul EM pentru învățarea parametrului distribuției Poisson, când se consideră că o parte din date lipsesc)

formulare de Liviu Ciortuz, după Ex. 20.8 din "Probability for Statistics and Machine Learning", Anirban DasGupta, Springer, 2011

Presupunem că vrem să modelăm statistic numărul de accidente ușoare care s-au produs în n locații într-un anumit interval de timp, să zicem o săptămână. În acest scop, vom folosi o distribuție Poisson de parametru λ .⁵⁹⁰ La sfârșitul perioadei de timp respective, ni se transmite de la m din cele n locații câte o „înregistrare“ (notată cu x_i), reprezentând numărul de accidente ușoare produse în locația i .

În mod implicit, ar trebui să considerăm că în cele $n - m$ locații de la care n-am primit înregistrări nu s-a produs niciun accident. Însă, în urma „inspectării“ datelor suntem determinați să luăm în considerare *presupunerea* că în unele din aceste $n - m$ locații s-a produs câte un [singur] accident ușor, care a fost „trecut sub tăcere“ la raportare.

Așadar, formalizând, vom considera datele „neobservabile“ $z_1 = 0, \dots, z_{n_0} = 0, z_{n_0+1} = 1, \dots, z_{n_0+n_1} = 1$, cu $n_0 + n_1 = n - m$, alături de datele observabile $x_1, \dots, x_m \in \mathbb{N}$. Toate aceste date sunt produse de variabile aleatoare urmând distribuția Poisson de [același] parametru λ .

Elaborați algoritmul EM (în speță pasul E și pasul M) pentru estimarea parametrului λ . *Sugestie:* În loc să se lucreze cu z_j , cu $j = 1, \dots, m$, va fi suficient să considerați ca date neobservabile n_0 și n_1 . Mai mult, considerând numerele n și m cunoscute, va fi suficient să lucrați doar cu n_1 ca dată neobservabilă (bineînțeles, pe lângă datele observabile x_i).

Răspuns:

Tinând cont de faptul că datele $z_1, \dots, z_{n_0}, z_{n_0+1}, \dots, z_{n_0+n_1}, x_1, \dots, x_m$ urmează distribuția Poisson de parametru λ , a cărei funcție de densitate este $P(x|\lambda) = \frac{1}{e^\lambda} \cdot \frac{\lambda^x}{x!}$, putem scrie expresia care ne dă verosimilitatea datelor „complete“:

⁵⁹⁰Am precizat deja la pr. 41 de la capitolul de *Fundamente* faptul că distribuția Poisson este utilă la modelarea fenomenelor rare.

$$\begin{aligned}
L(\lambda) &\stackrel{\text{def.}}{=} P(z_1, \dots, z_{n_0}, z_{n_0+1}, \dots, z_{n_0+n_1}, x_1, \dots, x_m | \lambda) \\
&\stackrel{i.i.d.}{=} \prod_{j=1}^{n_0+n_1} P(z_j | \lambda) \cdot \prod_{i=1}^m P(x_i | \lambda) \\
&= \prod_{j=1}^{n_0} \frac{1}{e^\lambda} \frac{\lambda^0}{0!} \cdot \prod_{j=n_0+1}^{n_0+n_1} \frac{1}{e^\lambda} \frac{\lambda}{1!} \cdot \prod_{i=1}^m \frac{1}{e^\lambda} \frac{\lambda^{x_i}}{x_i!} \\
&= \frac{1}{(e^\lambda)^{n_0+n_1+m}} \cdot \lambda^{n_1} \cdot \prod_{i=1}^m \frac{\lambda^{x_i}}{x_i!} = \frac{1}{(e^\lambda)^n} \cdot \lambda^{n_1} \cdot \prod_{i=1}^m \frac{\lambda^{x_i}}{x_i!} \\
&= e^{-n\lambda} \cdot \lambda^{n_1} \cdot \lambda^{\sum_{i=1}^m x_i} \cdot \frac{1}{\prod_{i=1}^m x_i!} = e^{-n\lambda} \cdot \lambda^{n_1 + \sum_{i=1}^m x_i} \cdot \frac{1}{\prod_{i=1}^m x_i!}
\end{aligned}$$

Log-verosimilitatea datelor complete este:

$$\ell(\lambda) \stackrel{\text{def.}}{=} \ln L(\lambda) = -n\lambda + \left(n_1 + \sum_{i=1}^m x_i \right) \ln \lambda - \sum_{i=1}^m \ln x_i!$$

Funcția „auxiliară“ va fi scrisă cu ajutorul distribuției a posteriori a datelor „neobservabile“ (n_1) în raport cu datele observabile (x_i , cu $i = 1, \dots, m$) și cu cu $\lambda^{(t)}$, care desemnează valoarea parametrului λ la iterată t . Așadar,

$$Q(\lambda | \lambda^{(t)}) \stackrel{\text{def.}}{=} E_{n_1 | x_i, \lambda^{(t)}} [\ell(\lambda)] = -n\lambda + \left(E[n_1 | x_i, \lambda^{(t)}] + \sum_{i=1}^m x_i \right) \ln \lambda - \sum_{i=1}^m \ln x_i!.$$

Pasul E:

$E_{n_1 | x_i, \lambda^{(t)}} [\ell(\lambda)]$ este numărul „așteptat“ de instanțe neobservabile z_j care au valoarea 1, din totalul de $n - m$ instanțe neobservabile. În cazul distribuției Poisson, definiția funcției de densitate implică $P(x = 1 | \lambda) = \frac{1}{e^\lambda} \cdot \frac{\lambda^1}{1!} = \frac{1}{e^\lambda} \cdot \lambda$ și $P(x = 0 | \lambda) = \frac{1}{e^\lambda} \cdot \frac{\lambda^0}{0!} = \frac{1}{e^\lambda}$. Rezultă că $E[n_1 | x_i, \lambda^{(t)}]$ este chiar media distribuției binomiale $Bin(n - m; \frac{\lambda^{(t)}}{1 + \lambda^{(t)}})$, deci

$$E[n_1 | x_i, \lambda^{(t)}] = (n - m) \frac{\lambda^{(t)}}{1 + \lambda^{(t)}}.$$

Pasul M:

Tinând cont de expresia care tocmai a fost obținută la pasul E, vom scrie funcția auxiliară sub forma următoare:

$$Q(\lambda | \lambda^{(t)}) = -n\lambda + \left[(n - m) \frac{\lambda^{(t)}}{1 + \lambda^{(t)}} + \sum_{i=1}^m x_i \right] \ln \lambda - \sum_{i=1}^m \ln x_i!.$$

Derivatele întâi și a doua ale acestei funcții în raport cu λ sunt:

$$\begin{aligned}
\frac{\partial}{\partial \lambda} Q(\lambda | \lambda^{(t)}) &= -n + \left[(n - m) \frac{\lambda^{(t)}}{1 + \lambda^{(t)}} + \sum_{i=1}^m x_i \right] \frac{1}{\lambda} \\
\frac{\partial^2}{\partial \lambda^2} Q(\lambda | \lambda^{(t)}) &= - \left[(n - m) \frac{\lambda^{(t)}}{1 + \lambda^{(t)}} + \sum_{i=1}^m x_i \right] \frac{1}{\lambda^2}
\end{aligned}$$

Conform enunțului, $n > m$ și $\sum_{i=1}^m x_i \geq 0$. La pasul de inițializare al algoritmului EM, parametrul λ își asignează o valoare $(\lambda^{(0)})$ pozitivă, ceea ce implică (vom vedea înălțată) $\lambda^{(t)} > 0$ la orice iterare $t > 0$. Prin urmare, derivata a două a funcției auxiliare este totdeauna negativă, ceea ce înseamnă că funcția auxiliară Q admite un maxim, și anume exact valoarea pentru care se anulează derivata întâi:

$$\lambda^{(t+1)} = \frac{1}{n} \left[(n-m) \frac{\lambda^{(t)}}{1 + \lambda^{(t)}} + \sum_{i=1}^m x_i \right]$$

Se poate vedea acum (prin inducție completă) că această valoare este strict pozitivă.

12. (Algoritmul EM: estimarea parametrilor pentru o sumă de două distribuții exponențiale)

■ CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 2.2

Considerăm Z_1 și Z_2 două variabile aleatoare independente, distribuite exponențial, de parametri λ_1 și respectiv λ_2 , cu $\lambda_1 \neq \lambda_2$.⁵⁹¹ Vă reamintim că funcția densitate de probabilitate a unei variabile aleatoare exponențiale Z de parametru $\lambda > 0$ este definită astfel:⁵⁹²

$$f_\lambda(z) = \begin{cases} \lambda \cdot e^{-\lambda z} & \text{pentru } z \geq 0 \\ 0 & \text{pentru } z < 0. \end{cases}$$

Se consideră variabila aleatoare $X = Z_1 + Z_2$. Se dau instanțele x_1, x_2, \dots, x_n independente și identic distribuite conform distribuției probabiliste a lui X .

- a. Să se calculeze funcția densitate de probabilitate a lui X în funcție de parametrii λ_1 și λ_2 .

Sugestie: Calculați mai întâi funcția de distribuție cumulativă a lui X . Pentru aceasta, puteți folosi formula $F(x) = \int_0^x \int_0^{x-z_1} f_{\lambda_1}(z_1) \cdot f_{\lambda_2}(z_2) dz_2 dz_1$.⁵⁹³

- b. Elaborați în detaliu cei doi pași ai algoritmului EM pentru estimarea parametrilor λ_1 și λ_2 . Precizați la final care sunt ecuațiile de actualizare pentru acești doi parametri, pornind de la setul de date $\{x_1, x_2, \dots, x_n\}$.

Răspuns:

- a. Funcția de distribuție cumulativă a lui X se calculează astfel:

$$\begin{aligned} F(x) &= P(Z_1 + Z_2 < x) = \int_0^x \int_0^{x-z_1} f_{\lambda_1}(z_1) \cdot f_{\lambda_2}(z_2) dz_2 dz_1 \\ &= \int_0^x \int_0^{x-z_1} \lambda_1 e^{-\lambda_1 z_1} \cdot \lambda_2 e^{-\lambda_2 z_2} dz_2 dz_1 \end{aligned}$$

⁵⁹¹Notație: $Z_1 \sim \exp\left(\frac{1}{\lambda_1}\right)$ și $Z_2 \sim \exp\left(\frac{1}{\lambda_2}\right)$.

⁵⁹²Pentru graficul acestei funcții, a se vedea problema 28.a de la capitolul de *Fundamente*.

⁵⁹³Deducerea acestei formule este destul de simplă:

$$\begin{aligned} F(x) &\stackrel{\text{def.}}{=} P(Z_1 + Z_2 < x) \\ &= \int_0^x \left(\int_0^{x-z_2} f_{\lambda_1}(z_1) dz_1 \right) \cdot f_{\lambda_2}(z_2) dz_2 \\ &= \int_0^x \left(\int_0^{x-z_1} f_{\lambda_2}(z_2) dz_2 \right) \cdot f_{\lambda_1}(z_1) dz_1 = \int_0^x \int_0^{x-z_1} f_{\lambda_1}(z_1) \cdot f_{\lambda_2}(z_2) dz_2 dz_1. \end{aligned}$$

$$\begin{aligned}
&= \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} \left(\int_0^{x-z_1} (-\lambda_2) e^{-\lambda_2 z_2} dz_2 \right) dz_1 \\
&= \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} \left(e^{-\lambda_2 z_2} \Big|_0^{x-z_1} \right) dz_1 \\
&= \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} \left(e^{-\lambda_2(x-z_1)} - e^{-\lambda_2 \cdot 0} \right) dz_1 \\
&= \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} e^{-\lambda_2(x-z_1)} dz_1 - \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} dz_1 \\
&= \frac{-\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 x} \int_0^x (\lambda_2 - \lambda_1) e^{(\lambda_2 - \lambda_1) z_1} dz_1 - \int_0^x (-\lambda_1) e^{-\lambda_1 z_1} dz_1 \\
&= -\frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 x} \left(e^{(\lambda_2 - \lambda_1) x} \Big|_0^x \right) - e^{-\lambda_1 x} \Big|_0^x \\
&= -\frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 x} \left(e^{(\lambda_2 - \lambda_1) x} - 1 \right) - (e^{-\lambda_1 x} - 1) \\
&= -\frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_1 x} + \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 x} - e^{-\lambda_1 x} + 1 \\
&= 1 - \frac{1}{\lambda_2 - \lambda_1} \left(\lambda_1 e^{-\lambda_1 x} - \lambda_1 e^{-\lambda_2 x} + \lambda_2 e^{-\lambda_1 x} - \lambda_1 e^{-\lambda_1 x} \right) \\
&= 1 - \frac{\lambda_2 e^{-\lambda_1 x} - \lambda_1 e^{-\lambda_2 x}}{\lambda_2 - \lambda_1}
\end{aligned}$$

Cunoscând funcția de distribuție cumulativă a lui X , vom calcula funcția de densitate de probabilitate a lui X :

$$p(x) = \frac{\partial F(x)}{\partial x} = -\frac{1}{\lambda_2 - \lambda_1} \left(-\lambda_1 \lambda_2 e^{-\lambda_1 x} + \lambda_1 \lambda_2 e^{-\lambda_2 x} \right) = \frac{\lambda_1 \lambda_2}{\lambda_2 - \lambda_1} \left(e^{-\lambda_1 x} - e^{-\lambda_2 x} \right)$$

b. Vom prezenta două variante / metode de elaborare a algoritmului EM pentru estimarea parametrilor λ_1 și λ_2 :

- Prima metodă constă în aplicarea clasicei schemei algoritmice EM:

Pasul E (Expectation): Se calculează funcția „auxiliară“ (media log-verosimilității datelor complete) pentru iteratăția t :

$$Q(\lambda | \lambda^{(t)}) = E_{P(Z|X, \lambda^{(t)})} [\log P(X, Z | \lambda)]$$

Pasul M (Maximization): Se maximizează media log-verosimilității datelor complete, calculată la pasul E, în raport cu λ :

$$\lambda^{(t+1)} = \underset{\lambda}{\operatorname{argmax}} Q(\lambda | \lambda^{(t)})$$

Notățile de mai sus au următoarele semnificații:

$$\begin{aligned}
\lambda &= (\lambda_1, \lambda_2) \\
\lambda^{(t)} &= \text{valoarea parametrului } \lambda \text{ la iteratăția } t \\
X &= \text{variabila observabilă, cu instanțele } x_1, x_2, \dots, x_n \\
Z &= (Z_1, Z_2) \text{ variabilele ascunse / neobservabile} \\
&\quad \text{având valorile } z_{1j}, z_{2j}, \dots, z_{nj} \text{ cu } j \in \{1, 2\}, \\
&\quad \text{așa încât } x_i = z_{i1} + z_{i2}.
\end{aligned}$$

În continuare vom elabora calculele corespunzătoare celor doi pași (E și M).

(Pasul E) Expresia funcției „auxiliare“ este:

$$\begin{aligned}
 Q(\lambda | \lambda^{(t)}) &= E_{P(Z|X, \lambda^{(t)})} \left[\log \prod_{i=1}^n p(x_i, z_{i1}, z_{i2} | \lambda) \right] \\
 &= E_{P(Z|X, \lambda^{(t)})} \left[\log \prod_{i=1}^n f_{\lambda_1}(z_{i1}) \cdot f_{\lambda_2}(z_{i2}) \right] \\
 &= E_{P(Z|X, \lambda^{(t)})} \left[\sum_{i=1}^n \log (\lambda_1 e^{-\lambda_1 z_{i1}} \cdot \lambda_2 e^{-\lambda_2 z_{i2}}) \right] \\
 &= E_{P(Z|X, \lambda^{(t)})} \left[\sum_{i=1}^n (\log \lambda_1 - \lambda_1 z_{i1} + \log \lambda_2 - \lambda_2 z_{i2}) \right] \\
 &= E_{P(Z|X, \lambda^{(t)})} \left[n \log \lambda_1 + n \log \lambda_2 - \sum_{i=1}^n (\lambda_1 z_{i1} + \lambda_2 z_{i2}) \right] \\
 &= n \log \lambda_1 + n \log \lambda_2 - \lambda_1 \sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] - \lambda_2 \sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}]
 \end{aligned}$$

Ultima egalitate de mai sus are loc datorită proprietății de liniaritate a mediilor.

Vom trece acum la calcularea mediilor variabilelor neobservabile. Media variabilei z_{i1} în raport cu probabilitatea condiționată $p(z_{i1} | x_i, \lambda^{(t)})$ este:

$$E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] \stackrel{\text{def.}}{=} \int_0^{x_i} z_{i1} \cdot p(z_{i1} | x_i, \lambda^{(t)}) dz_{i1}$$

Așadar, pentru a calcula această medie trebuie să obținem mai întâi expresia probabilității condiționate

$$p(z_{i1} | x_i, \lambda^{(t)}) \stackrel{\text{def.}}{=} \frac{p(z_{i1}, x_i | \lambda^{(t)})}{p(x_i | \lambda^{(t)})}$$

Probabilitatea de la numitor a fost calculată la punctul a , iar pentru cea de la numărător vom folosi relația $x_i = z_{i1} + z_{i2}$. Prin urmare,

$$p(z_{i1} | x_i, \lambda^{(t)}) = \frac{p(z_{i1}, z_{i2} | \lambda^{(t)})}{p(x_i | \lambda^{(t)})} = \frac{p(z_{i1} | \lambda_1^{(t)}) \cdot p(z_{i2} | \lambda_2^{(t)})}{p(x_i | \lambda^{(t)})},$$

ultima egalitate având loc datorită faptului că variabilele aleatoare Z_1 și Z_2 sunt independente. În consecință,

$$\begin{aligned}
 p(z_{i1} | x_i, \lambda^{(t)}) &= \frac{f_{\lambda_1^{(t)}}(z_{i1}) \cdot f_{\lambda_2^{(t)}}(x_i - z_{i1})}{\frac{\lambda_1^{(t)} \lambda_2^{(t)}}{\lambda_2^{(t)} - \lambda_1^{(t)}} (e^{-\lambda_1^{(t)} x_i} - e^{-\lambda_2^{(t)} x_i})} \\
 &= (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot \frac{\lambda_1^{(t)} e^{-\lambda_1^{(t)} z_{i1}} \cdot \lambda_2^{(t)} e^{-\lambda_2^{(t)} (x_i - z_{i1})}}{\lambda_1^{(t)} \lambda_2^{(t)} (e^{-\lambda_1^{(t)} x_i} - e^{-\lambda_2^{(t)} x_i})} \\
 &= (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot \frac{e^{(\lambda_2^{(t)} - \lambda_1^{(t)}) z_{i1}}}{(e^{-\lambda_1^{(t)} x_i} - e^{-\lambda_2^{(t)} x_i}) \cdot e^{\lambda_2^{(t)} x_i}}
 \end{aligned}$$

$$= (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot \frac{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}}}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1}$$

Așadar, media variabilei z_{i1} în raport cu probabilitatea condiționată $p(z_{i1} | x_i, \lambda^{(t)})$ este:

$$\begin{aligned} E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] &\stackrel{\text{def.}}{=} \int_0^{x_i} z_{i1} \cdot p(z_{i1} | x_i, \lambda^{(t)}) dz_{i1} \\ &= \int_0^{x_i} z_{i1} \cdot (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot \frac{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}}}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} dz_{i1} \\ &= \frac{1}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} \int_0^{x_i} z_{i1} \cdot (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} dz_{i1} \end{aligned}$$

Vom rezolva ultima integrală de mai sus utilizând formula de integrare prin părți:

$$\int f \cdot g' = f \cdot g - \int f' \cdot g$$

Așadar,

$$\begin{aligned} &\int_0^{x_i} z_{i1} \cdot (\lambda_2^{(t)} - \lambda_1^{(t)}) \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} dz_{i1} \\ &= \int_0^{x_i} z_{i1} \cdot \frac{\partial}{\partial z_{i1}} \left(e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} \right) dz_{i1} \\ &= \left(z_{i1} \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} \right) \Big|_0^{x_i} - \int_0^{x_i} e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} dz_{i1} \\ &= \left(x_i \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 0 \right) - \frac{1}{\lambda_2^{(t)} - \lambda_1^{(t)}} \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})z_{i1}} \Big|_0^{x_i} \\ &= x_i \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - \frac{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1}{\lambda_2^{(t)} - \lambda_1^{(t)}} \end{aligned}$$

Prin urmare,

$$\begin{aligned} E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] &= \frac{1}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} \cdot \left(x_i \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - \frac{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1}{\lambda_2^{(t)} - \lambda_1^{(t)}} \right) \\ &= \frac{x_i \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i}}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} - \frac{1}{\lambda_2^{(t)} - \lambda_1^{(t)}} \end{aligned}$$

Pentru a calcula media lui z_{i2} în raport cu probabilitatea condiționată $p(z_{i2} | x_i, \lambda^{(t)})$, vom utiliza relația $x_i = z_{i1} + z_{i2}$ și media lui z_{i1} calculată anterior:

$$\begin{aligned} E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}] &= E_{p(z_{i2}|x_i, \lambda^{(t)})}[x_i - z_{i1}] = x_i - E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i1}] \\ &= x_i - E_{p(z|x_i, \lambda^{(t)})}[z_{i1}] = x_i - E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] \\ &= x_i - \frac{x_i \cdot e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i}}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} + \frac{1}{\lambda_2^{(t)} - \lambda_1^{(t)}} \\ &= \frac{1}{\lambda_2^{(t)} - \lambda_1^{(t)}} - \frac{x_i}{e^{(\lambda_2^{(t)} - \lambda_1^{(t)})x_i} - 1} \end{aligned}$$

(Pasul M) Etapa de maximizare: Folosind expresia obținută pentru funcția auxiliară Q , vom calcula

$$\lambda^{(t+1)} = \underset{\lambda_1 > 0, \lambda_2 > 0}{\operatorname{argmax}} \left(n \log \lambda_1 + n \log \lambda_2 - \lambda_1 \sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] - \right. \\ \left. - \lambda_2 \sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}] \right)$$

Valorile parametrilor λ_1 și λ_2 corespunzătoare maximului acestei funcții se obțin folosind metoda derivelor partiale:

$$\frac{\partial}{\partial \lambda_1} Q(\lambda | \lambda^{(t)}) = 0 \Leftrightarrow \frac{n}{\lambda_1} = \sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] \Rightarrow \lambda_1^{(t+1)} = \frac{n}{\sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}]} > 0$$

$$\frac{\partial}{\partial \lambda_2} Q(\lambda | \lambda^{(t)}) = 0 \Leftrightarrow \frac{n}{\lambda_2} = \sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}] \Rightarrow \lambda_2^{(t+1)} = \frac{n}{\sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}]} > 0$$

Se verifică imediat că aceste soluții intr-adevăr maximizează $Q(\lambda | \lambda^{(t)})$.

- O a doua metodă de estimare a parametrilor λ_1 și λ_2 constă în aplicarea schemei algoritmice EM pentru funcția / variabila $X = Z_1 + Z_2$, urmând ideile de bază ale acestei scheme, dar făcând la pasul M [în locul calculelor] analogia uzuale cu soluția metodei de estimare directă (MLE) pentru parametrul unei distribuții exponențiale. Ca și mai înainte, variabilele Z_1 și Z_2 pot fi văzute ca variabile ascunse.

Pasul E (Expectation):

Se calculează mediile variabilelor ascunse z_{i1} și z_{i2} :

$$E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] \text{ și } E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}]$$

Calculele efective sunt cele realizate mai sus.

Pasul M (Maximization):

Se calculează / actualizează parametrul $\lambda \stackrel{\text{not.}}{=} (\lambda_1, \lambda_2)$ în funcție de valorile medii obținute la pasul precedent:⁵⁹⁴

Deoarece $Z_1 \sim \exp\left(\frac{1}{\lambda_1}\right)$, iar estimarea în sensul verosimilității maxime (MLE) a parametrului distribuției exponențiale este inversul mediei setului de exemple (vedeți problema 44 de la capitolul de *Fundamente*), aplicând operatorul E (media) și ținând cont de proprietatea de liniaritate a mediilor, rezultă:

$$\frac{1}{\lambda_1} = \frac{1}{n} \sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}] \Rightarrow \lambda_1^{(t+1)} = \frac{n}{\sum_{i=1}^n E_{p(z_{i1}|x_i, \lambda^{(t)})}[z_{i1}]}$$

Similar, pentru $Z_2 \sim \exp\left(\frac{1}{\lambda_2}\right)$ obținem:

$$\frac{1}{\lambda_2} = \frac{1}{n} \sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}] \Rightarrow \lambda_2^{(t+1)} = \frac{n}{\sum_{i=1}^n E_{p(z_{i2}|x_i, \lambda^{(t)})}[z_{i2}]}$$

⁵⁹⁴Este imediat că estimarea parametrului λ_1 poate fi făcută în mod independent de estimarea parametrului λ_2 (până la distribuția probabilistă în raport cu care s-au calculat mediile de la pasul E).

13. (Algoritmul EM pentru învățarea parametrilor a două distribuții gausiene, pornind de la instanțe generate de *suma* a două variabile care urmează aceste distribuții)
 ■ *Stanford, 2016 fall, A. Ng, J. Duchi, HW4, pr. 2*

La o conferință de învățare automată au fost trimise spre recenzare și, eventual, publicare P lucrări (engl., papers). Comitetul de recenzare a lucrărilor este format din R recenzori. Fiecare dintre acești R recenzori va citi toate cele P lucrări și va da fiecărei lucrări un scor, indicând astfel cât de bună crede el că este lucrarea respectivă. Vom nota cu x_{pr} scorul pe care recenzorul r îl atribuie lucrării p . Dacă scorul este mare, înseamnă că recenzorului i-a plăcut lucrarea respectivă; acest scor reprezintă o recomandare din partea recenzorului ca lucrarea respectivă să fie acceptată pentru prezentare la conferință. Dacă scorul este mic, înseamnă că recenzorului nu i-a plăcut lucrarea respectivă.

Vom presupune că fiecare lucrare p are o anumită valoare „intrinsecă“, pe care o vom nota cu μ_p ; atunci când valoarea aceasta este mare înseamnă că lucrarea respectivă este bună. Fiecare recenzor încearcă să estimeze, în urma citirii / analizării lucrării p , cât este μ_p . În mod concret, scorul x_{pr} , care este raportat de către recenzorul r , reprezintă încercarea acestuia de a ghici cât este μ_p .

Există tot soiul de factori aleatori care influențează procesul de recenzare a lucrărilor. Din acest motiv, în această problemă vom folosi / propune un model care incorporează mai multe surse de „zgomot“ / perturbații (engl., noise).

Unii recenzori sunt înclinați să credă că toate lucrările sunt bune; ei tind să acorde tuturor lucrărilor scoruri mari. Alții recenzori sunt, din contră, foarte severi și tind să acorde scoruri mici tuturor lucrărilor. În mod similar, este foarte posibil ca scorurile pe care doi recenzori diferiți le acordă lucrărilor pe care le recenzează să manifeste varianțe / dispersii foarte diferite, ceea ce înseamnă că unii recenzori sunt mai credibili decât alții. Vom nota cu ν_r , *bias-ul* recenzorului r ; asta înseamnă că scorurile puse de acest recenzor tind în general să fie cu cantitatea ν_r mai mari decât ar trebui să fie.

Așadar, din punct de vedere formal vom presupune că scorurile puse de recenzori sunt generate de către un proces aleatoriu, care este definit astfel:

$$\begin{aligned} y_{pr} &\sim \mathcal{N}(\mu_p, \sigma_p^2), \\ z_{pr} &\sim \mathcal{N}(\nu_r, \tau_r^2), \\ x_{pr}|y_{pr}, z_{pr} &\sim \mathcal{N}(y_{pr} + z_{pr}, \sigma^2). \end{aligned}$$

Variabilele y_{pr} și z_{pr} sunt independente; variabilele comune (x, y, z) care corespund unor perechi diferite de tip lucrare-recenzor sunt de asemenea independente. Pe lângă aceasta, trebuie precizat că noi observăm doar valorile variabilelor x_{pr} ; așadar, toate variabilele y_{pr} și z_{pr} sunt latente / neobservabile.

Ceea ce dorim este să estimăm valorile parametrilor μ_p , σ_p^2 , ν_r și τ_r^2 pentru $p = 1, \dots, P$ și $r = 1, \dots, R$. Din rațiuni de simplitate, vom trata σ^2 (varianța condițională a variabilei x_{pr} în raport cu y_{pr} și z_{pr}) ca și cum ar fi o constantă cunoscută, fixată. Dacă vom obține estimări bune pentru valorile „intrinseci“ (μ_p) ale lucrărilor, atunci aceste estimări vor putea fi folosite pentru fundamentarea deciziilor care trebuie luate în privința acceptării / respingerii lucrărilor pentru [prezentare la] conferință.

Vom estima valorile parametrilor μ_p , σ_p^2 , ν_r și τ_r^2 maximizând verosimilitatea datelor observabile $\{x_{pr} ; p = 1, \dots, P, r = 1, \dots, R\}$. În această problemă, aşa cum am precizat

deja, variabilele latente sunt y_{pr} și z_{pr} , iar maximizarea verosimilității nu se poate face în mod direct (engl., in closed form). Prin urmare, vom folosi algoritmul EM.

Sarcina dumneavoastră va fi să derivați regulile de actualizare specifice celor doi pași (E și M) ai algoritmului EM. Aceste reguli vor putea să conțină doar operatori de adunare, scădere, înmulțire, împărțire, log, exp și extragere de rădăcină pătrată ($\sqrt{\cdot}$) din constante reale, precum și adunare, scădere, înmulțire, inversare de matrice de numere reale și calculul de determinanți.

a. La acest punct veți obține relațiile de actualizare pentru pasul E:

i. Distribuția comună $p(y_{pr}, z_{pr}, x_{pr})$ este de tip gaussian multidimensional. Găsiți vectorul de medii asociat acestei distribuții, precum și matricea de covarianță corespunzătoare, în funcție de parametrii μ_p , σ_p^2 , ν_r , τ_r^2 și σ^2 .

Sugestie: x_{pr} poate fi scris ca $x_{pr} = y_{pr} + z_{pr} + \varepsilon_{pr}$, unde $\varepsilon_{pr} \sim \mathcal{N}(0, \sigma^2)$ este un „zgomot“ reprezentat de o variabilă de tip gaussian, independentă în raport cu y_{pr} și z_{pr} .

ii. Deducreți expresia distribuției condiționale $q_{pr}(y_{pr}, z_{pr}) \stackrel{\text{def.}}{=} p(y_{pr}, z_{pr} | x_{pr})$ de la pasul E, folosind regulile pentru condiționare în raport cu submulțimi de variabile aleatoare gaussiene comune.⁵⁹⁵

b. Deducreți regulile de actualizare de la pasul M pentru parametrii μ_p , ν_r , σ_p^2 și τ_r^2 .

Sugestie: S-ar putea să vă fie de folos să exprimați marginea inferioară a log-verosimilității datelor observabile ca o medie (engl., expectation) pentru valorile (y_{pr}, z_{pr}) generate de o distribuție aleatoare având funcția de densitate $q_{pr}(y_{pr}, z_{pr})$.

Comentariu: În articolul *Learning from the Wisdom of Crowds by Minimax Entropy* (NIPS, 2012), autorii Dengyong Zhou, John C. Platt,⁵⁹⁶ Sumit Basu și Yi Mao au descris implementarea unei metode care este destul de asemănătoare cu cea pe care am prezentat-o în acest exercițiu, cu scopul de a estima valorile „intrinseci“ ale lucrărilor, μ_p . În acel articol, problema este ceva mai complicată, fiindcă nu toți recenzorii evaluează fiecare lucrare, însă ideile de bază sunt în esență aceleași cu cele din acest exercițiu. Întrucât modelul acesta încearcă să estimeze și să corecteze biasurile recenzorilor (ν_r), estimările obținute pentru μ_p sunt mult mai folosite pentru fundamentarea deciziilor de acceptare / respingere a lucrărilor decât scorurile „brute“ acordate de rezatori.

Răspuns:

Vom desemna cu θ întregul set de parametri pe care îi estimăm.

Pașii algoritmului EM pentru problema noastră se scriu — la un nivel înalt — astfel:

Pasul E: Pentru fiecare pereche p, r vom calcula

$$q_{pr}^{(t)}(y_{pr}, z_{pr}) \stackrel{\text{not.}}{=} p(y_{pr}, z_{pr} | x_{pr}; \theta^{(t)});$$

Pasul M: $\theta^{(t+1)} = \operatorname{argmax}_{\theta} \sum_{p=1}^P \sum_{r=1}^R E_{q_{pr}^{(t)}(y_{pr}, z_{pr})} \ln p(x_{pr}, y_{pr}, z_{pr}; \theta).$

⁵⁹⁵Vedeți relația (256) din *Observația* finală de la rezolvarea problemei 35 de la capitolul de *Fundamente*.

⁵⁹⁶John C. Platt este autorul algoritmului SMO (Sequential Minimal Optimization), care este „inima“ implementărilor pentru mașinile cu vectori-suport (SVM). Vedeți problemele 23 și 51 de la capitolul *Mașini cu vectori-suport*.

Chestiunea care se pune acum este cum se calculează în mod concret aceste reguli de actualizare.

a. Pentru pasul E, facem următoarea *observație importantă*:

Dacă la calcularea probabilităților $p(y_{pr}, z_{pr}|x_{pr}; \theta^{(t)})$ am folosi regula lui Bayes combinată cu formula probabilității totale, atunci la numitor ar apărea integralele unor funcții de densitate gaussiene, care n-ar fi deloc ușor de calculat. În schimb, putem observa că funcția densitate de probabilitate

$$\begin{aligned} p(y_{pr}, z_{pr}, x_{pr}) &= p(y_{pr}, z_{pr}) \cdot p(x_{pr}|y_{pr}, z_{pr}) \\ &= p(y_{pr}) \cdot p(z_{pr}) \cdot p(x_{pr}|y_{pr}, z_{pr}) \end{aligned}$$

este un produs de alte trei funcții de densitate gaussiene, deci este și ea o densitate *gaussiană multidimensională*.⁵⁹⁷

Ca o *consecință*, distribuția [comună și] condiționată $y_{pr}, z_{pr}|x_{pr}$ va fi, la rândul ei, tot o distribuție gaussiană, pe care o vom putea calcula folosind *regulile pentru condiționarea gaussianelor*.

Pentru a obține expresia funcției de densitate pentru distribuția comună (y_{pr}, z_{pr}, x_{pr}) , vom ține cont de faptul că o distribuție gaussiană multidimensională este parametrizată [complet] de către vectorul medie și matricea de covarianță.

Pentru a calcula *vectorul medie* al distribuției comune (y_{pr}, z_{pr}, x_{pr}) , vom rescrie x_{pr} în forma următoare:

$$x_{pr} = y_{pr} + z_{pr} + \varepsilon_{pr},$$

unde variabila $\varepsilon_{pr} \sim \mathcal{N}(0, \sigma^2)$ reprezintă un „zgomot“ (engl., noise) care urmează o distribuție gaussiană și este independentă de variabilele y_{pr} și z_{pr} .⁵⁹⁸ Așadar,

$$\begin{aligned} E[y_{pr}] &= \mu_p \\ E[z_{pr}] &= \nu_p \\ E[x_{pr}] &= E[y_{pr} + z_{pr} + \varepsilon_{pr}] = E[y_{pr}] + E[z_{pr}] + E[\varepsilon_{pr}] \\ &= \mu_p + \nu_r + 0 = \mu_p + \nu_r. \end{aligned}$$

Pentru a calcula *matricea de covarianță* a distribuției comune (y_{pr}, z_{pr}, x_{pr}) , observăm că $Var(y_{pr}) = \sigma_p^2$, $Var(z_{pr}) = \tau_r^2$, iar $Cov(y_{pr}, z_{pr}) = Cov(z_{pr}, y_{pr}) = 0$ întrucât y_{pr} și

⁵⁹⁷Vedeți problema 31 de la capitolul de *Fundamente*, coroborată cu explicația de la nota de subsol 598.

Se poate demonstra că în general (deci nu doar pentru distribuții gaussiene independente), produsul a două distribuții gaussiene multidimensionale este tot o distribuție gaussiană multidimensională. Vedeți Appendix A.2 în cartea *Gaussian processes in machine learning*, Karl Rasmussen, Christopher Williams, MIT Press, 2006.

⁵⁹⁸Pentru a vedea de ce aceasta rezultă din definiția dată în enunțul problemei, observați că probabilitatea ca $\varepsilon_{pr} = x_{pr} - y_{pr} - z_{pr}$ să ia o anumită valoare ε este

$$\begin{aligned} p(\varepsilon_{pr} = \varepsilon|y_{pr}, z_{pr}) &= p(x_{pr} - y_{pr} - z_{pr} = \varepsilon|y_{pr}, z_{pr}) \\ &= p(x_{pr} = \varepsilon + y_{pr} + z_{pr}|y_{pr}, z_{pr}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}\varepsilon^2\right), \end{aligned}$$

care nu depinde nici de y_{pr} și nici de z_{pr} . Astfel, ε_{pr} poate fi privit ca un „zgomot“ independent, gaussian, cu media 0 și varianță σ^2 .

z_{pr} sunt independente.⁵⁹⁹ De asemenea, fiindcă y_{pr} , z_{pr} și ε_{pr} sunt independente, vom avea:⁶⁰⁰

$$\begin{aligned} \text{Var}(x_{pr}) &= \text{Var}(y_{pr} + z_{pr} + \varepsilon_{pr}) = \text{Var}(y_{pr}) + \text{Var}(z_{pr}) + \text{Var}(\varepsilon_{pr}) \\ &= \sigma_p^2 + \tau_r^2 + \sigma^2. \end{aligned}$$

În fine,

$$\begin{aligned} \text{Cov}(y_{pr}, x_{pr}) &= \text{Cov}(x_{pr}, y_{pr}) = \text{Cov}(y_{pr} + z_{pr} + \varepsilon_{pr}, y_{pr}) \\ &= \text{Cov}(y_{pr}, y_{pr}) + \text{Cov}(z_{pr}, y_{pr}) + \text{Cov}(\varepsilon_{pr}, y_{pr}) \\ &= \sigma_p^2 + 0 + 0 = \sigma_p^2, \end{aligned}$$

unde antipenultima egalitate se verifică ușor,⁶⁰¹ iar penultima egalitate rezultă din independența variabilelor y_{pr} , z_{pr} și ε_{pr} . În mod similar obținem $\text{Cov}(z_{pr}, x_{pr}) = \text{Cov}(x_{pr}, x_{pr}) = \tau_r^2$. Așadar, putem scrie:

$$y_{pr}, z_{pr}, x_{pr} \sim \mathcal{N} \left(\begin{bmatrix} \mu_p \\ \nu_r \\ \mu_p + \nu_r \end{bmatrix}, \begin{bmatrix} \sigma_p^2 & 0 & \sigma_p^2 \\ 0 & \tau_r^2 & \tau_r^2 \\ \sigma_p^2 & \tau_r^2 & \sigma_p^2 + \tau_r^2 + \sigma^2 \end{bmatrix} \right)$$

Acum vom folosi un rezultat clasic de la *conditionarea subseturilor de variabile* din distribuțiile gaussiene multidimensionale — și anume, relația (256) de la problema 256 de la capitolul de *Foundamente* — pentru a obține expresia căutată pentru $p(y_{pr}, z_{pr}|x_{pr}; \theta^{(t)})$:

$$q_{pr}^{(t)}(y_{pr}, z_{pr}) \stackrel{\text{not.}}{=} p(y_{pr}, z_{pr}|x_{pr}; \theta) = \mathcal{N} \left(\begin{bmatrix} \mu_{pr,Y} \\ \mu_{pr,Z} \end{bmatrix}, \begin{bmatrix} \Sigma_{pr,YY} & \Sigma_{pr,ZY} \\ \Sigma_{pr,YZ} & \Sigma_{pr,ZZ} \end{bmatrix} \right), \quad (218)$$

unde⁶⁰²

$$\mu_{pr} \stackrel{\text{not.}}{=} \begin{bmatrix} \mu_{pr,Y} \\ \mu_{pr,Z} \end{bmatrix} = \begin{bmatrix} \mu_p + \frac{\sigma_p^2}{\sigma^2 + \sigma_p^2 + \tau_r^2} (x_{pr} - \mu_p - \nu_r) \\ \nu_r + \frac{\tau_p^2}{\sigma^2 + \sigma_p^2 + \tau_r^2} (x_{pr} - \mu_p - \nu_r) \end{bmatrix} \quad (219)$$

$$\Sigma_{pr} \stackrel{\text{not.}}{=} \begin{bmatrix} \Sigma_{pr,YY} & \Sigma_{pr,ZY} \\ \Sigma_{pr,YZ} & \Sigma_{pr,ZZ} \end{bmatrix} = \frac{1}{\sigma^2 + \sigma_p^2 + \tau_r^2} \begin{bmatrix} \sigma_p^2(\tau_r^2 + \sigma^2) & -\sigma_p^2\tau_r^2 \\ -\sigma_p^2\tau_r^2 & \tau_r^2(\sigma_p^2 + \sigma^2) \end{bmatrix}. \quad (220)$$

Am omis detaliile de calcul, dar ele nu sunt dificil de elaborat.

b. Pentru pasul M, este bine să ne readucem aminte că distribuția a posteriori $q^{(t)}$ este calculată în funcție de $\theta^{(t)}$, în vreme ce parametrii pe care vrem să-i calculăm pentru pasul următor sunt $\theta^{(t+1)}$. Aceasta înseamnă că parametrii din distribuția $q^{(t)}$ sunt constanti în

⁵⁹⁹Vedeți problema 10 de la capitolul de *Fundamente*.

⁶⁰⁰Vedeți problema 21 de la capitolul de *Fundamente*, corelată cu problema 10 de la același capitol.

⁶⁰¹Se poate demonstra proprietatea $\text{Cov}(X, Y + Z) = \text{Cov}(Y + Z, X) = \text{Cov}(Y, X) + \text{Cov}(Z, X)$ pentru orice variabile aleatoare X , Y și Z . În locul lui $Y + Z$ se poate apoi considera orice sumă finită de variabile aleatoare.

⁶⁰²Pentru a crește lizibilitatea, în expresiile următoare am renunțat la a mai scrie la exponent numărul iterăției (t) pentru parametrii μ , σ^2 , τ^2 și Σ .

raport cu acei parametri în funcție de care se face maximizarea de la acest pas. Desemnând — într-o formă ușor simplificată — cu $E_q[\cdot]$ media calculată în raport cu distribuția de probabilitate $q_{pr}^{(t)}(y_{pr}, z_{pr})$ pentru fiecare p și r , vom face maximizarea media log-verosimilității datelor observabile:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta|\theta^{(t)}),$$

unde

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \sum_{p=1}^P \sum_{r=1}^R E_q \ln p(x_{pr}, y_{pr}, z_{pr}|\theta) \\ &\stackrel{i.i.d.}{=} \sum_{p=1}^P \sum_{r=1}^R E_q \ln \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x_{pr}-y_{pr}-z_{pr})^2} \cdot \frac{1}{\sqrt{2\pi}\sigma_p} e^{-\frac{1}{2\sigma_p^2}(y_{pr}-\mu_p)^2} \right. \\ &\quad \left. \frac{1}{\sqrt{2\pi}\tau_r} e^{-\frac{1}{2\tau_r^2}(z_{pr}-\nu_r)^2} \right] \\ &= \sum_{p=1}^P \sum_{r=1}^R E_q \left[\ln \frac{1}{(\sqrt{2\pi})^{3/2}\sigma\sigma_p\tau_r} - \frac{1}{2\sigma^2}(x_{pr}-y_{pr}-z_{pr})^2 \right. \\ &\quad \left. - \frac{1}{2\sigma_p^2}(y_{pr}-\mu_p)^2 - \frac{1}{2\tau_r^2}(z_{pr}-\nu_r)^2 \right] \\ &\stackrel{\theta=(\sigma_p, \tau_r)}{=} \sum_{p=1}^P \sum_{r=1}^R E_q \left[\ln \frac{1}{\sigma_p\tau_r} - \frac{1}{2\sigma_p^2}(y_{pr}-\mu_p)^2 - \frac{1}{2\tau_r^2}(z_{pr}-\nu_r)^2 \right] \\ &= \sum_{p=1}^P \sum_{r=1}^R E_q \left[\ln \frac{1}{\sigma_p\tau_r} - \frac{1}{2\sigma_p^2}((y_{pr})^2 - 2y_{pr}\mu_p + \mu_p^2) \right. \\ &\quad \left. - \frac{1}{2\tau_r^2}((z_{pr})^2 - 2z_{pr}\nu_r + \nu_r^2) \right] \\ &\stackrel{lin.}{=} \stackrel{med.}{\sum_{p=1}^P \sum_{r=1}^R \left[\ln \frac{1}{\sigma_p\tau_r} - \frac{1}{2\sigma_p^2}(E_q[(y_{pr})^2] - 2E_q[y_{pr}]\mu_p + \mu_p^2) \right.} \\ &\quad \left. - \frac{1}{2\tau_r^2}(E_q[(z_{pr})^2] - 2E_q[z_{pr}]\nu_r + \nu_r^2) \right]} \\ &\stackrel{lin.}{=} \stackrel{med.}{\sum_{p=1}^P \sum_{r=1}^R \left[\ln \frac{1}{\sigma_p\tau_r} - \frac{1}{2\sigma_p^2}(\Sigma_{pr,YY} + \mu_{pr,Y}^2 - 2\mu_{pr,Y} \cdot \mu_p + \mu_p^2) \right.} \\ &\quad \left. - \frac{1}{2\tau_r^2}(\Sigma_{pr,ZZ} + \mu_{pr,Z}^2 - 2\mu_{pr,Z} \cdot \nu_r + \nu_r^2) \right]. \end{aligned}$$

Ultima egalitate este o consecință a faptului că $E_q[y_{pr}] = \mu_{pr,Y}$ și $E_q[(y_{pr})^2] = (E_q[(y_{pr})^2] - E_q[y_{pr}]^2) + E_q[y_{pr}]^2 = \Sigma_{pr,YY} + \mu_{pr,Y}^2$. Relații similare se scriu și pentru $E_q[z_{pr}]$ și $E_q[(z_{pr})^2]$.

Calculând derivatele parțiale ale funcției „auxiliare” $Q(\theta|\theta^{(t)})$ în raport cu parametrii μ_p , ν_r , σ_p și respectiv τ_r și egalându-le apoi cu 0, vom obține:

$$-\frac{1}{2\sigma_p^2} \sum_{r=1}^R (2\mu_p - 2\mu_{pr,Y}) = 0 \Rightarrow \mu_p^{(t+1)} = \frac{1}{R} \sum_{r=1}^R \mu_{pr,Y} \quad (221)$$

$$-\frac{1}{2\tau_r^2} \sum_{p=1}^P (2\nu_r - 2\mu_{pr,Z}) = 0 \Rightarrow \nu_p^{(t+1)} = \frac{1}{P} \sum_{p=1}^P \mu_{pr,Z} \quad (222)$$

$$\sum_{r=1}^R \left[-\frac{1}{\sigma_p} + \frac{1}{\sigma_p^3} (\Sigma_{pr,YY} + \mu_{pr,Y}^2 - 2\mu_{pr,Y} \mu_p + \mu_p^2) \right] = 0 \Rightarrow \\ (\sigma_p^2)^{(t+1)} = \frac{1}{R} \sum_{r=1}^R (\Sigma_{pr,YY} + \mu_{pr,Y}^2 - 2\mu_{pr,Y} \mu_p + \mu_p^2) \quad (223)$$

$$\sum_{p=1}^P \left[-\frac{1}{\tau_r} + \frac{1}{\tau_r^3} (\Sigma_{pr,ZZ} + \mu_{pr,Z}^2 - 2\mu_{pr,Z} \nu_r + \nu_r^2) \right] = 0 \Rightarrow \\ (\tau_r^2)^{(t+1)} = \frac{1}{P} \sum_{p=1}^P (\Sigma_{pr,ZZ} + \mu_{pr,Z}^2 - 2\mu_{pr,Z} \nu_r + \nu_r^2) \quad (224)$$

Folosind acum rezultatele pe care le-am obținut, putem reformula pașii E și M ai algoritmului EM astfel:

Pasul E: Pentru fiecare p și r , calculează $q_{pr}^{(t)}(y_{pr}, z_{pr})$, conform relației (218), folosind μ_{pr} și Σ_{pr} din relațiile (219) și respectiv (220);

Pasul M: Pentru fiecare p și r , calculează $\mu_p^{(t+1)}$, $\nu_r^{(t+1)}$, $(\sigma_p^2)^{(t+1)}$ și $(\tau_r^2)^{(t+1)}$ folosind relațiile (221), (222), (223) și respectiv (224).

14. (Algoritmul EM: estimarea probabilității de selecție a unei componente din cadrul unei mixturi – combinație liniară de două distribuții probabiliste oarecare)
 ■ CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW2, pr. 2.1
 CMU, 2006 spring, Carlos Guestrin, final exam, pr. 8

Se consideră variabila aleatoare X având funcția densitate de probabilitate sub forma unei mixturi de două componente:

$$p_\alpha(x) = \alpha \cdot p_1(x) + (1 - \alpha) \cdot p_2(x)$$

Se cunosc componentele $p_1(x)$ și $p_2(x)$ care reprezintă modele / funcții de densitate de probabilitate (considerate nespecificate aici), dar nu se cunoaște valoarea parametrului $\alpha \in [0, 1]$.

Cunoscând exemplele $\{x_1, x_2, \dots, x_n\}$ care au fost generate în mod independent și sunt identic distribuite conform distribuției variabilei X , formulați algoritmul EM pentru estimarea parametrului α . Descrieți în mod explicit cei doi pași — pasul E (pentru calculul mediilor) și pasul M (de maximizare) — ai algoritmului.

Observație: Se poate face în mod natural extensia la mixturi cu un număr arbitrar (dar fixat) de componente probabiliste.

Răspuns:

Pentru a aplica algoritmul EM considerăm variabilele ascunse $\{z_1, z_2, \dots, z_n\}$, cu $z_i \in \{1, 2\}$, pentru $i = \overline{1, n}$. Pentru fiecare valoare (fixată) a lui i , dacă $z_i = 1$, atunci exemplul x_i a fost generat de componenta $p_1(x)$ a mixturii, iar dacă $z_i = 2$, atunci exemplul x_i a fost generat de componenta $p_2(x)$.

Pasul E al algoritmului EM constă în calcularea mediei log-verosimilității datelor complete:

$$Q(\alpha | \alpha^{(t)}) \stackrel{\text{def.}}{=} E_{P(Z|X,\alpha^{(t)})}[\log P(X, Z | \alpha)]$$

Deoarece fiecare variabilă z_i depinde doar de x_i , iar datele x_i sunt independente între ele, putem scrie:

$$\begin{aligned} Q(\alpha | \alpha^{(t)}) &\stackrel{\text{def.}}{=} E_{P(Z|X,\alpha^{(t)})} \left[\sum_{i=1}^n \log p(x_i, z_i | \alpha) \right] \stackrel{i.i.d.}{=} \sum_{i=1}^n E_{p(z_i|x_i,\alpha^{(t)})}[\log p(x_i, z_i | \alpha)] \\ &\stackrel{\text{calcul}}{=} \sum_{i=1}^n \left[\sum_{z_i \in \{1,2\}} p(z_i | x_i, \alpha^{(t)}) \cdot \log p(x_i, z_i | \alpha) \right] \\ &= \sum_{i=1}^n \left[p(z_i = 1 | x_i, \alpha^{(t)}) \cdot \log p(x_i, z_i = 1 | \alpha) + \right. \\ &\quad \left. + p(z_i = 2 | x_i, \alpha^{(t)}) \cdot \log p(x_i, z_i = 2 | \alpha) \right] \\ &= \sum_{i=1}^n \left[p(z_i = 1 | x_i, \alpha^{(t)}) \cdot \log(\alpha p_1(x_i)) + \right. \\ &\quad \left. + p(z_i = 2 | x_i, \alpha^{(t)}) \cdot \log((1 - \alpha)p_2(x_i)) \right] \\ &= \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) \cdot \log(\alpha) + \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) \cdot \log p_1(x_i) + \\ &\quad \sum_{i=1}^n p(z_i = 2 | x_i, \alpha^{(t)}) \cdot \log(1 - \alpha) + \sum_{i=1}^n p(z_i = 2 | x_i, \alpha^{(t)}) \cdot \log p_2(x_i) \end{aligned}$$

Probabilitățile condiționate implicate în această formulă pot fi calculate cu ajutorul teoremei lui Bayes:

$$\begin{aligned} p(z_i = 1 | x_i, \alpha^{(t)}) &\stackrel{T.B.}{=} \frac{p(x_i | z_i = 1, \alpha^{(t)}) \cdot p(z_i = 1 | \alpha^{(t)})}{p(x_i, \alpha^{(t)})} \\ &= \frac{p_1(x_i) \cdot \alpha^{(t)}}{p_1(x_i) \cdot \alpha^{(t)} + p_2(x_i) \cdot (1 - \alpha^{(t)})} \\ p(z_i = 2 | x_i, \alpha^{(t)}) &\stackrel{T.B.}{=} \frac{p(x_i | z_i = 2, \alpha^{(t)}) \cdot p(z_i = 2 | \alpha^{(t)})}{p(x_i, \alpha^{(t)})} \\ &= \frac{p_2(x_i) \cdot (1 - \alpha^{(t)})}{p_1(x_i) \cdot \alpha^{(t)} + p_2(x_i) \cdot (1 - \alpha^{(t)})} \end{aligned}$$

Pasul M al algoritmului EM constă în determinarea valorii parametrului α pentru care se obține maximul expresiei $Q(\alpha | \alpha^{(t)})$, care a fost deja calculată mai sus. Pentru a determina $\alpha^{(t+1)} \stackrel{\text{def.}}{=} \operatorname{argmax}_{\alpha} Q(\alpha | \alpha^{(t)})$ vom folosi derivata de ordinul întâi în raport cu α :

$$\begin{aligned} \frac{\partial Q(\alpha | \alpha^{(t)})}{\partial \alpha} &= 0 \Leftrightarrow \frac{1}{\alpha} \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) - \frac{1}{1 - \alpha} \sum_{i=1}^n p(z_i = 2 | x_i, \alpha^{(t)}) = 0 \\ &\Leftrightarrow (1 - \alpha) \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) - \alpha \sum_{i=1}^n p(z_i = 2 | x_i, \alpha^{(t)}) = 0 \end{aligned}$$

$$\begin{aligned}
&\Leftrightarrow \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) - \alpha \left(\sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) + \sum_{i=1}^n p(z_i = 2 | x_i, \alpha^{(t)}) \right) = 0 \\
&\Leftrightarrow \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) - \alpha \sum_{i=1}^n \underbrace{\left(p(z_i = 1 | x_i, \alpha^{(t)}) + p(z_i = 2 | x_i, \alpha^{(t)}) \right)}_{=1} = 0 \\
&\Leftrightarrow \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}) - n \cdot \alpha = 0 \Rightarrow \alpha = \frac{1}{n} \sum_{i=1}^n p(z_i = 1 | x_i, \alpha^{(t)}).
\end{aligned}$$

Se observă că această soluție corespunde unui punct de maxim (de fapt, unicul punct de maxim) al funcției Q . Prin urmare, ținând cont de expresia probabilității $p(z_i = 1 | x_i, \alpha^{(t)})$ care a fost calculată mai sus (vedeți pasul E), formula de actualizare pentru parametrul α este:

$$\alpha^{(t+1)} = \frac{1}{n} \sum_{i=1}^n \frac{\alpha^{(t)} \cdot p_1(x_i)}{\alpha^{(t)} \cdot p_1(x_i) + (1 - \alpha^{(t)}) \cdot p_2(x_i)}$$

15. (Algoritmul EM pentru estimarea parametrilor în sens MAP: fundamentare teoretică)

Stanford, 2016 fall. A. Ng, J. Duchi, HW4, pr. 1

Algoritmul EM, aşa cum l-am folosit până la acest exercițiu, a fost conceput pentru a rezolva [unele] probleme de *estimare de parametri în sensul verosimilității maxime* (engl., maximum likelihood estimation, MLE). În astfel de probleme se urmărește să se maximizeze o expresie de forma următoare:⁶⁰³

$$\prod_{i=1}^n p(x_i | \theta) = \prod_{i=1}^n \left(\sum_{z_i} p(x_i, z_i | \theta) \right), \quad (225)$$

unde

- θ este setul de parametri pentru distribuția probabilistă p ,
- x_i (cu $i = 1, \dots, n$) sunt variabile aleatoare observabile,
- z_i (cu $i = 1, \dots, n$) sunt variabile aleatoare neobservabile / latente.

În acest exercițiu presupunem că lucrăm într-un *cadrul bayesian*, adică dorim să găsim *estimarea în sensul probabilității maxime a posteriori* (engl., maximum a posteriori probability, MAP) pentru parametru θ . Aceasta revine la a maximiza o expresie de forma

⁶⁰³Justificarea este aceasta: log-probabilitatea a posteriori pentru datele observabile este

$$\begin{aligned}
\log p(\theta, x_1, \dots, x_n) &\stackrel{F. Bayes}{=} \log \frac{p(x_1, \dots, x_n | \theta) \cdot p(\theta)}{p(x_1, \dots, x_n)} \\
&= \log(p(x_1, \dots, x_n | \theta) \cdot p(\theta)) - \log p(x_1, \dots, x_n) \\
&\stackrel{i.i.d.}{=} \log \left(\left(\prod_{i=1}^n p(x_i | \theta) \right) \cdot p(\theta) \right) - \log p(x_1, \dots, x_n).
\end{aligned}$$

Întrucât cel de-al doilea termen nu depinde de θ , rezultă

$$\operatorname{argmax}_{\theta} \log p(\theta, x_1, \dots, x_n) = \operatorname{argmax}_{\theta} \log \left(\left(\prod_{i=1}^n p(x_i | \theta) \right) \cdot p(\theta) \right).$$

$$\left(\prod_{i=1}^n p(x_i|\theta) \right) \cdot p(\theta) = \left(\prod_{i=1}^n \left(\sum_{z_i} p(x_i, z_i|\theta) \right) \right) \cdot p(\theta),$$

unde $p(\theta)$ desemnează o distribuție de probabilitate definită a priori peste valorile parameterilor θ .

Generalizați algoritmul EM astfel încât să realizeze estimări în sens MAP. Veți presupune că ambele distribuții — $p(x, z|\theta)$ și $p(\theta)$ — sunt concave în raport cu parametrul θ . Aceasta va implica faptul că pasul M al algoritmului EM este *realizabil* (engl., tractable) atunci când se cere doar să se maximizeze o combinație liniară de aceste cantități ($p(x, z|\theta)$ și $p(\theta)$).⁶⁰⁴

Asigurați-vă că pasul M din formularea pe care o veți adopta pentru noul algoritm EM este *realizabil*. De asemenea, demonstrați că valorile expresiei $(\prod_{i=1}^n p(x_i|\theta)) \cdot p(\theta)$, văzută ca funcție de θ , nu descresc — adică, fie cresc, fie rămân pe loc — de la o iterare la alta a noului algorithm.

Răspuns:

Vom deduce regulile de actualizare de la cei doi pași ai algoritmului EM într-un mod similar celui în care am procedat la problema 1 pentru estimare în sensul verosimilității maxime (MLE). La fiecare iterare (t) , creșterea este monotonă, din aceleași motive ca și la problema 1: la pasul E calculăm [o] cea mai bună *margine inferioară* (engl., lower bound) a log-probabilității a posteriori pentru datele observabile, în raport cu $\theta^{(t)}$, care reprezintă valorile curente ale parametrilor θ , iar la pasul M alegem pentru θ o nouă valoare, $\theta^{(t+1)}$, care este optimă în raport cu această margine inferioară.

Pornind de la expresia (225) din enunț, vom identifica marginea inferioară despre care tocmai am vorbit:

$$\begin{aligned} \log \left(\left(\prod_{i=1}^n p(x_i|\theta) \right) \cdot p(\theta) \right) &= \log p(\theta) + \sum_{i=1}^n \log p(x_i|\theta) \\ &= \log p(\theta) + \sum_{i=1}^n \log \sum_{z_i} p(x_i, z_i|\theta) = \log p(\theta) + \sum_{i=1}^n \log \sum_{z_i} q(z_i) \cdot \frac{p(x_i, z_i|\theta)}{q(z_i)} \\ &\geq \log p(\theta) + \sum_{i=1}^n \sum_{z_i} q(z_i) \cdot \log \frac{p(x_i, z_i|\theta)}{q(z_i)}. \end{aligned} \quad (226)$$

La ultimul pas de mai sus am aplicat inegalitatea lui Jensen.⁶⁰⁵ Ca și la problema 1, inegalitatea are loc cu egalitate atunci când la *pasul E* (de la aiterația t) luăm

$$q(z_i) = p(z_i|x_i; \theta^{(t)}).$$

Pentru *pasul M*, trebuie să maximizăm marginea inferioară (vedeți relația (226)), adică vom calcula

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \left(\log p(\theta) + \sum_{i=1}^n \sum_{z_i} q(z_i) \log \frac{p(x_i, z_i|\theta)}{q(z_i)} \right).$$

⁶⁰⁴Grosso-modo, aceasta revine la a presupune că estimarea în sens MAP este *realizabilă* atunci când variabilele x, z sunt în totalitate observabile, exact cum procedam în cadrul „frecvenționist“ (adică, non-bayesian).

⁶⁰⁵Vedeți problema 71.c de la capitolul de *Fundamente*.

Pasul M este *realizabil*, întrucât ceea ce se cere aici este doar să maximizăm o combinație liniară de termenii $\log p(\theta)$ și $\log p(x, z|\theta)$, care, conform ipotezei din enunț, sunt concavi. În sfârșit, nu este deloc dificil să demonstrăm faptul că produsul $(\prod_{i=1}^n p(x_i|\theta)) \cdot p(\theta)$, văzut ca o funcție de argumentul θ , crește monoton de la o iterare la alta a algoritmului EM. Practic, tot ceea ce trebuie făcut este să extindem ușor demonstrația de la problema 2. Într-adevăr, notând

$$Q'(\theta | \theta^{(t)}) = \log p(\theta) + \underbrace{\sum_{i=1}^n \sum_{z_i} q(z_i) \log \frac{p(x_i, z_i | \theta)}{q(z_i)}}_{Q(\theta | \theta^{(t)})},$$

relațiile (191) și (192) devin

$$\begin{aligned} \log P(x | \theta) + \log P(\theta) &= Q(\theta | \theta^{(t)}) + CH(\theta | \theta^{(t)}) + \log P(\theta) \\ \log P(x | \theta^{(t)}) + \log P(\theta^{(t)}) &= Q(\theta^{(t)} | \theta^{(t)}) + CH(\theta^{(t)} | \theta^{(t)}) + \log P(\theta^{(t)}). \end{aligned}$$

Scăzându-le membru cu membru, obținem:

$$\begin{aligned} &\log P(x | \theta) + \log P(\theta) - (\log P(x | \theta^{(t)}) + \log P(\theta^{(t)})) \\ &= Q(\theta | \theta^{(t)}) + \log P(\theta) - (Q(\theta^{(t)} | \theta^{(t)}) + \log P(\theta^{(t)})) + CH(\theta | \theta^{(t)}) - CH(\theta^{(t)} | \theta^{(t)}) \\ &= Q'(\theta | \theta^{(t)}) - Q'(\theta^{(t)} | \theta^{(t)}) + \underbrace{CH(\theta | \theta^{(t)}) - CH(\theta^{(t)} | \theta^{(t)})}_{\geq 0}. \end{aligned}$$

Concluzia este că în mod garantat valoarea funcției de probabilitate a posteriori a datelor observabile nu descrește, ci fie crește fie rămâne pe loc, la fiecare iterare a acestui algoritm de tip EM pentru estimare de parametri în sens MAP.

16. (Adevărat ori Fals?)

- a. *CMU, 2002 fall, Andrew Moore, final exam, pr. 1.e*
 Spre deosebire de metoda gradientului, care se poate bloca în poziția unui optim local, algoritmul EM (Expectation-Maximization) identifică întotdeauna optimul global.
- b. *CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm, pr. 1.8.b*

Algoritmul EM nu micșorează valoarea funcției obiectiv de la o iterare la alta.

Răspuns:

- a. Fals. Ambele metode se pot bloca într-un optim local.
- b. Adevărat. La problema 1 s-a arătat că algoritmul EM maximizează o margine inferioară ($F(q(z), \theta)$) pentru funcția de log-verosimilitate a datelor observabile ($\log P(x|\theta) = \log \sum_z P(x, z|\theta)$). Metoda folosită pentru maximizare este una iterativă ("coordinate ascent"). La problema 2 s-a demonstrat că valoarea funcției de log-verosimilitate a datelor observabile nu se micșorează de la o iterare la alta a algoritmului EM.

8.2 Probleme propuse

17. (Algoritmul EM: estimarea parametrilor unei mixturi de distribuții categoriale: cazul general⁶⁰⁶)
 prelucrare de Liviu Ciortuz, după
 ■ CMU, 2015 spring, T. Mitchell, N. Balcan, HW6, pr. 1

La acest exercițiu vom lucra cu modele de mixturi de distribuții *categoriale*.⁶⁰⁷

Fie un set de date $x = \{x_1, \dots, x_n\}$, unde fiecare $x_i \in \{v_1, \dots, v_M\}$ este generat [în mod independent de ceilalți x_j , cu $j \neq i$] de către una din K distribuții categoriale posibile, notată cu X_i . Vom considera că distribuția care l-a generat pe x_i a fost desemnată de către o altă variabilă aleatoare categorială, luând valori în multimea $\{1, \dots, K\}$ și având vectorul de parametri $\pi \in [0, 1]^K$, cu $\sum_k \pi_k = 1$. Vom nota cu $\theta_k \in \mathbb{R}^M$ parametrul distribuției categoriale asociate cu componenta k din mixtură, deci $\theta_k \in (0, 1)^M$ și $\sum_{j=1}^M \theta_{kj} = 1$ pentru $k = 1, \dots, K$.

Procesul generativ pentru un *model de mixtură categorială* poate fi sumarizat astfel:

$$\begin{aligned} Z_i &\sim \text{Categorical}(\pi) \\ X_i &\sim \text{Categorical}(\theta_{Z_i}) \end{aligned}$$

Pentru acest model, în care observăm variabilele X dar nu și variabilele Z , obiectivul este să învățăm parametru $\Theta = \{\pi, \theta_1, \dots, \theta_K\}$. Veți folosi algoritmul EM pentru a realiza acest obiectiv.

Observație: Atunci când lucrăm cu distribuții categoriale, este utilă folosirea funcțiilor indicator [în locul variabilelor-indicator]. Prin definiție, funcția-indicator $1_{\{x=j\}}$ are valoarea 1 dacă $x = j$ și 0 în caz contrar.⁶⁰⁸

- Calculați *distribuția comună* a datelor observabile (Z) și neobservabile (Z): $P(X, Z; \Theta)$.
- Calculați *probabilitățile a posteriori* corespunzătoare *variabilelor latente*, $P(Z_i = k | X_i; \Theta)$.
- Calculați *media log-verosimilități* datelor complete,

$$Q(\Theta | \Theta') \stackrel{\text{def.}}{=} E_{Z|X; \Theta'} [\log P(X, Z; \Theta)].$$

⁶⁰⁶Problema aceasta reprezintă o generalizare în raport cu problema 4, unde am prezentat algoritmul EM pentru o mixtură de două distribuții Bernoulli. Generalizarea se referă la faptul că în locul distribuțiilor Bernoulli aici se consideră distribuții categoriale, iar pe de altă parte, numărul de distribuții cu care se lucrează poate fi oricare ($K \geq 1$).

Problema 8 este un caz particular (sau, o ilustrare) pentru problema de față. Pe de altă parte, atunci când mixtura are $K = 2$ componente, problema 9 reprezintă o generalizare a problemei de față.

⁶⁰⁷La problema 115 de la capitolul de *Fundamente* am arătat cum se face estimarea în sens MLE pentru parametrii unei distribuții categoriale. Aici vom estima parametrii unei mixturi de K distribuții categoriale.

⁶⁰⁸De exemplu, dacă vom considera o variabilă aleatoare Y care ia valori în multimea $\{1, \dots, N\}$ și urmează o distribuție categorială (notăție: $Y \sim \text{Categorical}(\phi)$, unde $\phi \in (0, 1)^N$), vom putea să exprimăm probabilitatea ca Y să ia o anunită valoare în felul următor:

$$P(Y) = \prod_{i=1}^N \phi_i^{1_{\{Y=i\}}}.$$

d. Deducreți regulile de actualizare pentru parametrii Θ . Altfel spus, care este valoarea lui Θ care maximizează funcția $Q(\Theta|\Theta')$ calculată la punctul c ?

Indicație: Aveți grijă că soluția pe care o veți obține trebuie să satisfacă restricția ca parametrii distribuțiilor categoriale trebuie să se sumeze la valoarea 1. Metoda multiplicatorilor Lagrange este o cale foarte convenabilă pentru rezolvarea unor astfel de probleme de optimizare cu restricții.

18.

(Algoritmului EM pentru „învățarea“ unei distribuții multinomiale care este definită cu ajutorul unui [singur] parametru; o aplicație în domeniul bioinformaticii)

Liviu Ciortuz, 2017, după

■ Georgia Institute of Technology,
Bayesian Statistics course (ISyE 8843A),
Brani Vidakovic, 2004, Handout 12, sec. 1.2.1

Introducere: La problema 7 am arătat cum se pot estima parametrii unei distribuții categoriale (și, în mod implicit, ai unei distribuții multinomiale) în cazul unei probleme de bioinformatică, în care este implicată o genă (numită ABO) care are trei alele. Distribuția categorială din acea problemă avea trei parametri (de fapt doi, fiindcă suma celor trei parametri trebuie să fie 1). În problema de față, vom folosi un singur parametru (ψ) pentru a estima probabilitățile unei alte distribuții multinomiale, care este specifică unei alte probleme de bioinformatică, unde de această dată sunt două perechi de gene, fiecare genă având câte două alele, iar probabilitățile de „recombinare“ pentru aceste alele sunt diferite în cazul bărbaților, față de cazul femeilor.⁶⁰⁹

Fie o variabilă aleatoare X care ia valori în mulțimea $\{v_1, v_2, v_3, v_4\}$ și urmează distribuția $Multinomial\left(n; \frac{2+\psi}{4}, \frac{1-\psi}{4}, \frac{1-\psi}{4}, \frac{\psi}{4}\right)$, unde ψ este un parametru cu valori în intervalul $(0, 1)$. Cele patru probabilități listate în definiția acestei distribuții multinomiale sunt în corespondență directă cu valorile v_1, v_2, v_3 și v_4 .

a. Presupunem că n_1, n_2, n_3 și n_4 sunt numărul de „realizări“ ale valorilor v_1, v_2, v_3 și v_4 în totalul celor n „observații“. (Pentru fixarea ideilor, vom considera $n_1 = 125$, $n_2 = 18$, $n_3 = 20$ și $n_4 = 34$.) Calculați estimarea de verosimilitate maximă (MLE) a parametrului ψ .

b. Fie acum variabila aleatoare $X' \sim Multinomial\left(n; \frac{1}{2}, \frac{\psi}{4}, \frac{1-\psi}{4}, \frac{1-\psi}{4}, \frac{\psi}{4}\right)$. Valurile luate de variabila X' , corespunzător acestor cinci probabilități, sunt (în ordine) v_1, v_1, v_2, v_3 și v_4 .⁶¹⁰ Vom considera n_{11}, n_{12}, n_2, n_3 și respectiv n_4 numărul de „realizări“ ale acestor valori, însă de data aceasta n_{11} și n_{12} vor fi neobservabile. În schimb, vom

⁶⁰⁹Pentru explicații de tip biologic / genetic pentru această problemă, vedeți secțiunea 1.2.1 din documentul indicat mai sus: *Handout 12*, de la cursul de Statistică bayesiană (ISyE 8843A) care a fost tăinut de către profesorul Brani Vidakovic la Georgia Institute of Technology, în anul 2004.

⁶¹⁰Observați că, în raport cu distribuția multinomială de la punctul a, am „descompus“ probabilitatea $\frac{2+\psi}{4}$ în două probabilități, $\frac{1}{2}$ și $\frac{\psi}{4}$, ca și cum ele ar corespunde unor evenimente disjuncte.

furniza suma $n_1 = n_{11} + n_{12}$ ca dată „observabilă“, alături de celelalte date observabile, n_2 , n_3 și n_4 .

Să se estimeze parametrul ψ folosind algoritmul EM. Cum este această estimare față de valoarea obținută la punctul a?

Indicație: Veți elabora formulele corespunzătoare pasului E și pasului M. Apoi veți face o implementare și veți rula algoritmul EM (pornind, de exemplu, cu valoarea inițială 0.5 pentru ψ) până când valorile acestui parametru până la cea de-a șasea zecimală nu se mai modifică.

c. Pentru a pune în evidență convergența algoritmului EM pe datele de mai sus, calculați — și apoi reprezentați grafic pe intervalul $(0, 1)$ — funcția de log-verosimilitate a datelor observabile

$$\ell_c(\psi) \stackrel{\text{def.}}{=} \ln \sum_{n_{11}+n_{12}=n_1} g_c(\underbrace{n_{11}, n_{12}}_{\text{neobs.}}, \underbrace{n_2, n_3, n_4}_{\text{obs.}}, \psi)$$

unde

$$\begin{aligned} g_c(n_{11}, n_{12}, n_2, n_3, n_4, \psi) &= \frac{n!}{n_{11}! n_{12}! n_2! n_3! n_4!} \left(\frac{1}{2}\right)^{n_{11}} \left(\frac{\psi}{4}\right)^{n_{12}} \left(\frac{1-\psi}{4}\right)^{n_2} \left(\frac{1-\psi}{4}\right)^{n_3} \left(\frac{\psi}{4}\right)^{n_4} \\ &= \frac{n!}{n_{11}! n_{12}! n_2! n_3! n_4!} \left(\frac{1}{2}\right)^{n_{11}} \left(\frac{\psi}{4}\right)^{n_{12}+n_4} \left(\frac{1-\psi}{4}\right)^{n_2+n_3} \end{aligned}$$

cu $n = n_{11} + n_{12} + n_2 + n_3 + n_4$.

d. La fiecare iterație t a algoritmului EM,

- la pasul E, calculați estimările \hat{n}_{11} și \hat{n}_{12} , precum și $H^{(t)}$, entropia distribuției probabiliste $\text{Multinomial}\left(n; \frac{1}{2}, \frac{\psi^{(t)}}{4}, \frac{1-\psi^{(t)}}{4}, \frac{1-\psi^{(t)}}{4}, \frac{\psi^{(t)}}{4}\right)$;
- la pasul M, calculați $\psi^{(t+1)}$ și trasați graficul funcției auxiliare $Q(\psi|\psi^{(t)}) = \ln \frac{n!}{\hat{n}_{11}! \hat{n}_{12}! n_2! n_3! n_4!} + \hat{n}_{11} \ln \frac{1}{2} + (\hat{n}_{12} + n_4) \ln \frac{\psi}{4} + (\hat{n}_2 + n_3) \ln \frac{1-\psi}{4}$;
- verificați, conform proprietății pe care am demonstrat-o la problema 1.c, că $\ell_c(\psi^{(t)}) = Q(\psi^{(t)}|\psi^{(t)}) + H^{(t)}$.

19.

(Modele de mixturi pentru analiza semantăcă

a documentelor de tip text:

modelul K-bag-of-words și modelul domeniilor semantice)

CMU, 2011 fall, Eric Xing, HW5, pr. 2

Introducere:

În acest exercițiu veți compara două modele de analiză semantică pentru documente de tip text: modelul *K*-bag-of-words (în secțiunea A de mai jos) și modelul mixturii domeniilor semantice (în secțiunea B).⁶¹¹ Ambele modele sunt bazate pe *variable latente / neobservabile*: valorile anumitor variabile sunt necunoscute, iar noi suntem interesați să găsim aceste valori. În modelul *K*-bag-of-words, vrem să aflăm pentru fiecare document

⁶¹¹O versiune simplificată a modelului domeniilor semantice a fost prezentată la problema 10.

(i) dintr-o colecție de documente date care este domeniul semantic „ascuns“ (t_i) asociat documentului respectiv (în manieră “soft”), dintr-o mulțime de K domenii. În modelul domeniilor semantice, suntem interesati mai ales de distribuțiile „ascunse“ ($\theta_i \stackrel{not.}{=} (\theta_i^1, \dots, \theta_i^K)$, cu $\theta_i^k \in [0, 1]$ și $\sum_{k=1}^K \theta_i^k = 1$) asociate documentului i și de domeniul semantic (sau: sensul) asociat (și aici în manieră “soft”) fiecărui cuvânt (z_{ij}).

A. Modelul K -bag-of-words — un sens pentru fiecare document

În cadrul primei părți a acestui exercițiu veți face cunoștință cu o mixtură (oarecum simplificată) de distribuții categoriale care este folosită pentru a modela [generarea de] cuvinte în documente de tip text. Acest model este cunoscut sub numele de *modelul celor K urne de cuvinte* (engl., K -bag-of-words).

Punctul de plecare îl constituie K distribuții *categoriale* care sunt asociate „urnelor“ de cuvinte. Fiecare dintre aceste K distribuții categoriale este definită peste cuvintele dintr-un (același) vocabular. Numărul de cuvinte din vocabular este V . *Parameterii* acestor distribuții sunt desemnați respectiv prin β_1, \dots, β_K . Fiecare β_k (cu $k = 1, \dots, K$) este un vector V -dimensional, ale căruia componente sunt numere nenegative care, însumate dău valoarea 1.

Vom considera N documente, numerotate cu $1, \dots, N$, fiecare document i având M_i cuvinte. Remarcați faptul că documentele pot conține un număr diferit de cuvinte. Pentru fiecare document i , cuvântul cu numărul de ordine j va fi desemnat prin $w_{ij} \in \{1, \dots, V\}$. Așadar, vom folosi numere întregi (reprezentând *indicele* poziției corespunzătoare din vocabular) pentru a desemna cuvintele din documente.

Cuvintele (de fapt, *aparițiile* cuvintelor) din cele N documente sunt modelate după cum urmează:

Se consideră [încă] o distribuție de probabilitate de tip discret, [la care ne vom referi cu termenul *a priori* și] care este reprezentată prin vectorul π . Pentru fiecare document i se alege în mod aleatoriu, conform distribuției menționate, un *indicator* [de domeniu semantic / tematică] $t_i \in \{1, \dots, K\}$. Indicatorul [tematic] t_i ne spune care dintre cele K distribuții categoriale generează cuvintele din documentul i . Apoi, extragem (adică, generăm) fiecare cuvânt w_{ij} al documentului i în conformitate cu parametrii β_{t_i} ai distribuției categoriale t_i , extragerile făcându-se în mod independent unele de altele.⁶¹²

Acest design corespunde următorului *proces generativ*:

$$\begin{aligned} t_i &\sim \text{Categorial}(\pi) \text{ pentru } i \in \{1, \dots, N\} \\ w_{ij} &\sim \text{Categorial}(\beta_{t_i}) \text{ pentru } i \in \{1, \dots, N\} \text{ și } j \in \{1, \dots, M_i\}. \end{aligned}$$

Remarcați faptul că în acest model, în raport cu modelul clasic de mixtură de K distribuții categoriale, diferența este faptul că se consideră [seturi de] instanțe / date „observate“ care au în comun același indicator de mixtură t_i (și anume, cuvintele w_{ij} , cu $j = 1, \dots, M_i$).

Indicație: Vă cerem ca, în rezolvarea acestui exercițiu, să folosiți indici superiori (engl., superscripts) pentru a desemna componentele vectorilor. De exemplu, prin notația π^k veți desemna elementul de pe poziția k din vectorul π . De asemenea, variabila-indicator t_i care a fost introdusă mai sus va fi rescrisă sub forma vectorului-indicator $t_i \stackrel{not.}{=} (t_i^1, t_i^2, \dots, t_i^K)$. Deci prin t_i^k veți indica elementul de pe poziția k din vectorul-indicator t_i . Precizăm că, prin convenție, în acest vector doar una dintre componente este 1,⁶¹³ toate celelalte fiind 0.

⁶¹²Așadar, nu se ține cont de *ordinea* de apariție a cuvintelor în document.

⁶¹³Și anume, poziția corespunzătoare *variabilei*-indicator t_i , care a fost introdusă anterior.

- a. Vectorul de probabilități de selecție π se notează cu $\pi = (\pi^1, \dots, \pi^K)$. Am precizat deja că probabilitatea selectării unui domeniu semantic $t_i \in \{1, \dots, K\}$ este π^{t_i} . Scriem formal acest fapt astfel: $P(t_i|\pi) = \pi^{t_i}$.

Vă cerem să exprimați într-o manieră unitară probabilitățile $P(t_i|\pi)$ pentru $i = 1, \dots, N$, folosind un artificiu de genul celui pe care l-am introdus la rezolvarea mixturilor de distribuții Bernoulli (vedeți relația (205) de la problema 4), extins acum la distribuția categorială.

- b. Exprimăți probabilitatea $P(w_{ij}|t, \beta, \pi)$ într-o formă cât mai simplă, ținând cont de relațiile de (in)dependență. Altfel spus, va trebui să scrieți o egalitate de forma $P(w_{ij}|t, \beta, \pi) = P(w_{ij}|\dots)$, unde simbolul '...' desemnează o anumită submulțime din $\{t, \beta, \pi\}$. Simbolii t și β scriși fără indici inferiori (engl., subscripts) reprezintă ansamblul tuturor vectorilor t_1, \dots, t_N , respectiv β_1, \dots, β_K .

Sugestie: Încercați să reprezentați grafic modelul de mixtură de tip *K-bag-of-words*. Probabilitatea pe care o veți indica în răspunsul final va trebui să fie condiționată în raport cu toți β_1, \dots, β_K .

- c. Explicați probabilitatea „simplificată“ pe care ați indicat-o în răspunsul de la punctul b, folosind definiția distribuției categoriale.

B. Modelul mixturii domeniilor semantice — *un sens pentru fiecare cuvânt*

În cea de-a doua parte a exercițiului nostru vom explora „mixtura de mixturi“ care formează baza așa-numitului *model al [mixturii] domeniilor semantice* (engl., topic model) pentru clusterizare de documente de tip text.⁶¹⁴ Putem gândi acest model ca fiind obținut prin operarea a două modificări asupra modelului (mai simplu) *K-bag-of-words* care a fost prezentat în secțiunea A:

În primul rând, în loc ca documentului i să-i fie asignat un singur *domeniu semantic*⁶¹⁵ t_i , acum îi vom permite acestuia să fie caracterizat / modelat de o *mixtură de domenii semantice*. Vom asocia acestei mixturi un vector $\theta_i \stackrel{\text{not.}}{=} (\theta_i^1, \dots, \theta_i^K)$, cu $\theta_i^k \in [0, 1]$ și $\sum_{k=1}^K \theta_i^k = 1$. Mai precis, θ_i trebuie să fie o distribuție de probabilitate definită peste K domenii semantice reprezentate respectiv de tuplurile de parametri β_1, \dots, β_K . Ca și în secțiunea A, pentru fiecare $k \in \{1, \dots, K\}$, β_k este un vector, $(\beta_k^1, \dots, \beta_k^V)$, cu $\beta_k^v \in [0, 1]$ și $\sum_{v=1}^V \beta_k^v = 1$, unde V este mărimea vocabularului. Aceasta din urmă este comun pentru toate documentele ($i = 1, \dots, N$). Distribuția probabilistă desemnată prin $\theta_i = (\theta_i^1, \dots, \theta_i^K)$ este una de tip *Dirichlet* și va fi prezentată mai jos.

În al doilea rând, introducem pentru fiecare cuvânt w_{ij} câte un *indicator* $z_{ij} \in \{1, \dots, K\}$ care determină domeniul semantic asociat cuvântului w_{ij} . Remarcați cum diferă acest design de modelul *K-bag-of-words*: acum permitem fiecarui cuvânt să aibă propriul său domeniu semantic, în loc să-i impunem să „adopte“ domeniul semantic al documentului (t_i).⁶¹⁶ Desigur, vom genera valorile variabilei z_{ij} conform distribuției $\theta_i = (\theta_i^1, \dots, \theta_i^K)$ asociate documentului i .

⁶¹⁴Pentru mai multe informații, vă recomandăm articolul *Latent Dirichlet Allocation*, Blei et al, 2003.

⁶¹⁵Sau: tematică.

⁶¹⁶Cele două abordări / principii sunt cunoscute în domeniul lingvisticii computaționale sub forma *un sens la fiecare apariție* (engl., “one sense per occurrence”), respectiv *un [singur] sens pe întregul document* (engl., “one sense per document”).

Aceste două schimbări dau naștere următorului *proces generativ*:

$$\begin{aligned}\theta_i &= (\theta_i^1, \dots, \theta_i^K) \sim \text{Dirichlet}(\alpha) \text{ pentru } i \in \{1, \dots, N\} \\ z_{ij} &\in \{1, \dots, K\} \sim \text{Categorial}(\theta_i) \text{ pentru } i \in \{1, \dots, N\} \text{ și } j \in \{1, \dots, M_i\}, \\ w_{ij} &\in \{1, \dots, V\} \sim \text{Categorial}(\beta_{z_{ij}}) \text{ pentru } i \in \{1, \dots, N\} \text{ și } j \in \{1, \dots, M_i\},\end{aligned}$$

unde M_i este lungimea documentului i , iar $\alpha > 0$ este un parametru scalar pentru distribuția (simetrică) Dirichlet. Distribuția Dirichlet este definită sub forma următoare:

$$P(\theta_i|\alpha) = \frac{\Gamma(K\alpha)}{[\Gamma(\alpha)]^K} \prod_{k=1}^K (\theta_i^k)^{\alpha-1},$$

symbolul Γ desemnând funcția Gamma.⁶¹⁷

Observați că acest model este într-adevăr o „mixtură de mixturi” de distribuții categoriale: fiecare document i i se asociază o mixtură θ_i peste domeniile β_1, \dots, β_K , și există N astfel de mixturi, $\theta_1, \dots, \theta_N$, care constituie împreună mixtura de mixturi de distribuții categoriale.

- d. Exprimăți probabilitatea $P(z_{ij}|\theta, \alpha, \beta)$ într-o formă cât mai simplă, ținând cont de relațiile de (in)dependență. Simbolii θ și β scriși fără indici (engl., subscripts) reprezintă ansamblurile de parametri $\theta_1, \dots, \theta_N$ și respectiv β_1, \dots, β_K .
- e. Explicitați probabilitatea „simplificată” pe care ati indicat-o ca răspuns la punctul precedent, folosind definiția distribuției categoriale.
- f. Exprimăți probabilitatea $P(w_{ij}|z, \theta, \alpha, \beta)$ într-o formă cât mai simplă, ținând cont de relațiile de (in)dependență.
- g. Explicitați probabilitatea „simplificată” pe care ati indicat-o ca răspuns la punctul precedent, folosind definiția distribuției categoriale.

C. Comparăție

Punctele / întrebările următoare vă vor ajuta să faceti o *comparație* între modelul K -bag-of-words și modelul mixturii domeniilor semantice pe care le-am prezentat la secțiunile A și respectiv B.

- h. Atât indicatorul t_i din modelul K -bag-of-words, cât și distribuția θ_i din modelul mixturii domeniilor semantice spun ceva anume despre conținutul tematic (sau: domeniul semantic) al documentului i . Formulați într-o singură frază diferența principală dintre t_i și θ_i .
- i. Discutați implicațiile răspunsului de la punctul h. În ce fel este mai utilă modelarea de tipul mixturii domeniilor semantice decât *modelarea* de tip K -bag-of-words?
- j. Atât în modelul K -bag-of-words cât și în modelul mixturii domeniilor semantice, parametrii β_k reprezintă vocabularul pentru fiecare domeniu semantic k . Nu discutăm aici despre învățarea valorilor parametrilor β , dar se poate arăta că anumite strategii de învățare clasice (de exemplu, *Algoritmul EM*⁶¹⁸ și *Gibbs sampling*) vor „produce” uneori domenii semantice care folosesc în comun (engl., share) anumite cuvinte — altfel spus, putem avea $\beta_k^v > 0$ și $\beta_l^v > 0$ pentru un cuvânt v și două domenii semantice distincte k și l . De ce este oare util așa ceva (ne referim la *word sharing*)?

⁶¹⁷Vedeți problema 41 de la capitolul de *Fundamente*, precum și http://en.wikipedia.org/wiki/Gamma_function.

⁶¹⁸Vedeți problema 10 de la acest capitol.

20. (Algoritmul EM: estimarea parametrilor unei mixturi de [vectori de] distribuții Poisson)
Univ. of Utah, 2008 fall, Hal Daumé III, HW9, pr. 1

Am făcut cunoștință cu distribuția Poisson la problemele 24 și 41 de la capitolul de *Fundamente*. Vă reamintim că această distribuție este definită peste numerele naturale pozitive, și anume: dat fiind parametrul λ , funcția masă de probabilitate (p.m.f.) a distribuției Poisson este dată de expresia

$$p(x|\lambda) = \frac{1}{e^\lambda} \cdot \frac{\lambda^x}{x!}, \text{ pentru orice } x \in \mathbb{N}.$$

În problema 41 menționată mai sus, am văzut că, dată fiind o secvență de numere naturale pozitive x_1, \dots, x_n , estimarea de verosimilitate maximă (MLE) pentru parametrul λ este $\frac{1}{n} \sum_{i=1}^n x_i$, adică exact media [aritmetică a] numerelor date.

În acest exercițiu vom considera o generalizare a acestei distribuții: modelul mixturii de [vectori de] distribuții Poisson. Nu știm dacă ati aflat [sau nu] până acum, însă acest model este folosit la monitorizarea serverelor de internet: numărul de cereri de acces care sunt adresate unui server de internet într-o unitate de timp (de exemplu, un minut) urmează de obicei o distribuție Poisson.

Considerăm că avem n servere de internet și că le monitorizăm pe fiecare pe o durată de M minute. Așadar, vom obține $n \times M$ numere pozitive (count-uri); vom nota cu $x_{i,m}$ numărul de cereri adresate serverului i în minutul m . Scopul nostru este să clusterizăm serverele de internet în funcție de frecvența cererilor adresate lor [în timp].⁶¹⁹

Presupunem că dorim să formăm K clustere. Definiți un *model* de tip *mixtură de [vectori de] distribuții Poisson* pentru această problemă. Veți nota cu z_i variabila latentă care desemnează cărui cluster de servere de internet îi aparține serverul i (mai precis, indicând un număr de la 1 la K). De asemenea, veți nota cu λ_k parametrul distribuției Poisson care este urmată de fiecare componentă a vectorilor de variabile Poisson din clusterul k și cu π_k probabilitatea de selecție a respectivului cluster.

Elaborați algoritmul EM pentru rezolvarea acestui de tip *mixtură de [vectori de] distribuții Poisson*. Stabiliti pentru pasul E relațiile de calcul pentru mediile variabilelor latente z_i , iar pentru pasul M relațiile de actualizare pentru parametrii λ și probabilitățile de selecție π pentru distribuțiile din acest model.

Indicație: Verosimilitatea datelor complete va arăta astfel:

$$\begin{aligned} L(\lambda, \pi) &\stackrel{\text{def.}}{=} P(\bar{x}, z|\lambda, \pi) \stackrel{i.i.d.}{=} \prod_{i=1}^n P(x_i, z_i|\lambda, \pi) = \prod_{i=1}^n P(x_i|z_i, \lambda, \pi) \cdot P(z_i|\lambda, \pi) \\ &= \prod_{i=1}^n \prod_{k=1}^K \left[\pi_k \prod_{m=1}^M p(x_{i,m}|\lambda_k) \right]^{1_{\{z_i=k\}}}, \end{aligned}$$

unde

⁶¹⁹În cazul (limită!) în care $M = 1$, modelul nostru va deveni modelul unei mixturi de K distribuții de tip Poisson (nu *vectori* de distribuții de tip Poisson). Pentru cazul (și mai restrictiv!) $K = 2$, puteți vedea U. Chicago, Statistics course, 2004 spring, Michael Eichler, HW3, pr. 3.

$\bar{x} \stackrel{not.}{=} (x_1, \dots, x_n)$, cu $x_i \stackrel{not.}{=} (x_{i,1}, \dots, x_{i,M})$ pentru $i = 1, \dots, n$,
 $z \stackrel{not.}{=} (z_1, \dots, z_K)$, $\pi \stackrel{not.}{=} (\pi_1, \dots, \pi_K)$, $\lambda \stackrel{not.}{=} (\lambda_1, \dots, \lambda_K)$, iar
 $1_{\{z_i=k\}}$ este funcția-indicator; ea ia valoarea 1 dacă $z_i = k$ și 0 în caz contrar.

21. (Algoritmul EM: pasul E pentru estimarea parametrilor unei mixturi de distribuții Gamma)
prelucrare de Liviu Ciortuz, după CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final exam, pr. 2

Considerăm instanțele $X_i \in \mathbb{R}^+$, ($i = 1, \dots, n$) produse de către următoarea mixtură de distribuții:

$$\begin{aligned} Z_i &\sim \text{Categorial}(\pi_1, \pi_2, \dots, \pi_K) \\ X_i &\sim \text{Gamma}(2, \beta_{Z_i}) \end{aligned}$$

Funcția de densitate a distribuției de probabilitate $\text{Gamma}(2, \beta)$ este definită astfel: $P(X = x) = \beta^2 x e^{-\beta x}$.

- a. Vom considera numărul de distribuții din mixtura $K = 3$ și parametrii (secunzi) ai acestor distribuții, $\beta_1 = 1, \beta_2 = 2, \beta_3 = 4$. Calculați $P(Z_i = 1 | X_i = 1)$.
- b. Elaborați pasul E al algoritmului EM pentru estimarea parametrilor acestei mixturi, scriind câte o formulă / relație matematică pentru fiecare expresie / cantitate care trebuie calculată la acest pas.
- c. Credeti că acest model de mixtura de distribuții Gamma poate opera cu clustere non-disjuncte (engl., overlapping), similar cu modelul mixturii de distribuții gaussiene?

22. (Algoritmul “hard” EM; exemplificare pentru rezolvarea unei mixturi de distribuții categoriale multidimensionale, folosind presupoziția de independentă condițională de tip Bayes Naiv, cu asignare “hard” a instanțelor la clustere)
prelucrare de Liviu Ciortuz, după CMU, 2014 spring, A. Singh, B. Poczos, HW3, pr. 2.2

Fie un set de instanțe neetichetate $\mathcal{D} = \{x_1, \dots, x_n\}$. La problema 1.a (fundamentarea teoretică a algoritmului EM), am folosit la scrierea funcției de log-verosimilitate $(\log P(\mathcal{D}|\theta) = \sum_i \log P(x_i|\theta))$ faptul că $P(x_i|\theta)$ poate fi rescris / exprimat însușând probabilitățile comune corespunzătoare lui x_i (pe de o parte) și fiecărei asignări posibile a variabilelor latente / neobservabile (pe de altă parte).

În acest exercițiu ne interesează să vedem ce se întâmplă atunci când în loc să facem sumarea despre care am vorbit mai sus, vom seta variabilele neobservabile la valorile lor cele mai probabile în raport cu estimările / valorile actuale ale parametrilor. Această strategie este numită adeseori *algoritmul “hard” EM*. În anumite cazuri, el funcționează bine.⁶²⁰

⁶²⁰Spre exemplu, putem obține o variantă a algoritmului K-means aplicând algoritmul “hard” EM unei mixturi de distribuții gaussiene.

- a. Arătați că și în situația în care în loc să procedăm la sumarea probabilităților pentru toate asignările posibile ale variabilelor latente — $P(X_i = x_i | \pi, \beta) = \sum_z P(X_i = x_i, Z = z | \pi, \beta)$ — facem maximizare, are loc o optimizare a unei margini inferioare pentru funcția de log-verosimilitate a datelor observabile.
- b. Elaborați pasul E pentru varianta “hard” a algoritmului EM.
- c. Elaborați pasul M pentru varianta “hard” a algoritmului EM într-un cadru particular, și anume atunci când *obiectivul* nostru este să rezolvăm problema de *învățare nesupervizată de tip Bayes Naïv*, însă nu folosind asignare “soft” a instantelor la clustere cum am procedat la exercițiul 9, ci folosind asignare “hard”.

23.

(Algoritmul EM: chestiuni metodologice)

CMU, 2014 spring, A. Singh, B. Poczos, HW3, pr. 2.3

Algoritmul EM converge în general la un optim local. Legat de această chestiune, formulați în mod succint câteva strategii care ar putea fi folosite pentru a se obține totuși estimări acceptabil de bune ale parametrilor, atunci când se utilizează algoritmul EM în practică.

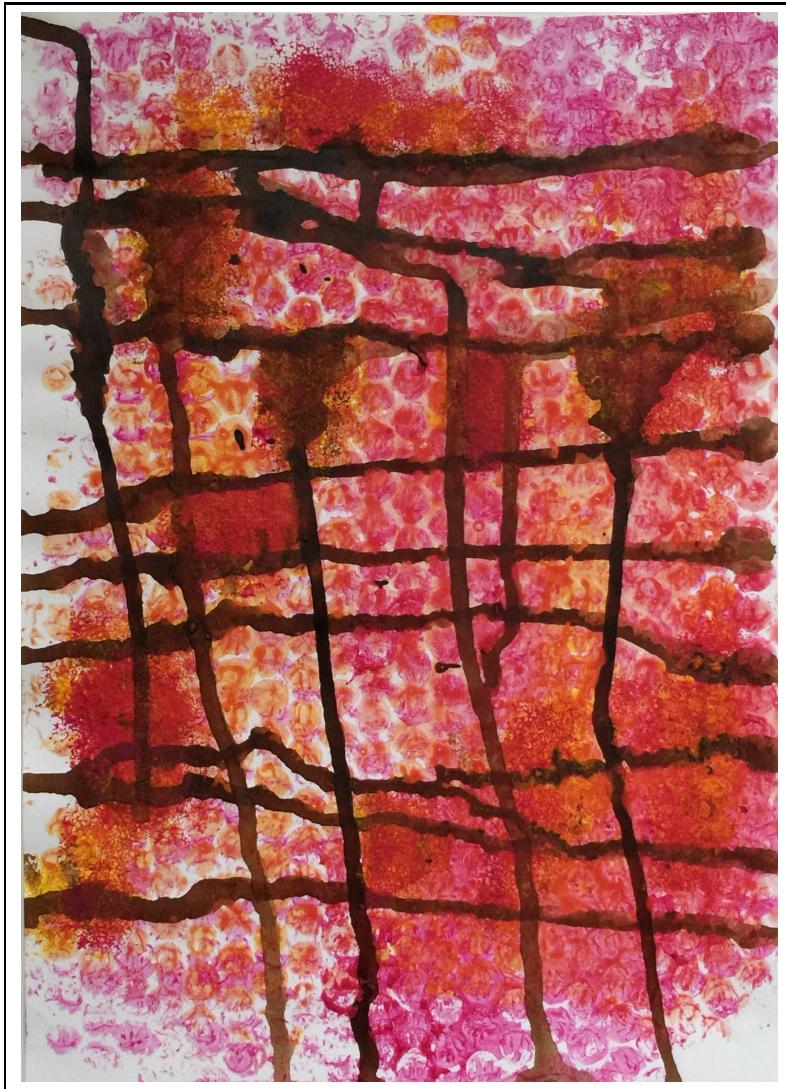
24.

(Adevărat ori Fals?)

CMU, 2010 spring, E. Xing, T. Mitchell, A. Singh, midterm exam., pr. 1.8.ab

- a. Algoritmul EM optimizează o margine inferioară (engl., lower bound) a funcției sale obiectiv, $\log \prod_i P(x_i | \theta)$, unde x_1, \dots, x_n sunt datele observate, iar θ sunt parametrii modelului asociat acestor date.
- b. Funcția obiectiv optimizată de algoritmul EM poate fi optimizată și cu metoda gradientului, care va găsi optimul global, în vreme ce EM găsește soluția sa mai rapid dar poate returna doar un optim local. Adevărat sau fals? Justificați.

Această pagină a fost lăsată liberă în mod intenționat.



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.

9 Modele Markov ascunse

Sumar

- Noțiuni preliminare: programare dinamică, schema algoritmică EM.
- Verificarea înțelegerii unor noțiuni de bază în referitoare la modelele Markov ascunse (engl., Hidden Markov Models, HMM): ex. 1-4, ex. 15-17.
- Model Markov vizibil: exemplificare, legătura cu HMM: ex. 5.
- HMM ca model probabilist total: ex. 14.
- Algoritmul Forward:⁶²¹
exempe de aplicare: ex. 6, ex. 7, ex. 9.a;
demonstrarea formulei de la pasul inductiv: ex. 18.a;
calcularea probabilității de emitere a unei secvențe, folosind probabilitățile Forward: ex. 18.b.
- Algoritmul Backward:⁶²²
demonstrarea formulei de la pasul inductiv: ex. 8;
exemplu de aplicare: ex. 9.a.
- Algoritmul Viterbi:⁶²³
exemplu de aplicare: ex. 9.b;
determinarea căii celei mai probabile de generare a unei secvențe: ex. 9.c;
determinarea probabilității ca un anumit simbol [din secvența de semnale] să fi fost generat într-o anumită stare: ex. 9.d;
o variație pe tema algoritmului Viterbi: ex. 20.
- Algoritmul Forward-Backward / Baum-Welch / EM pentru HMM:⁶²⁴
exemplu de aplicare: ex. 11.b;
demonstrarea formulei necesare pentru calcularea mediilor variabilelor neobservabile care corespund tranzițiilor: ex 10;
demonstrarea faptului că algoritmul Forward-Backward lasă neschimbate probabilitățile de tranzitie sau de emisie care sunt nule [la initializare]: ex 12.
- Demonstrația unei formule alternative — în raport cu formulele bazate pe probabilitățile Forward și respectiv probabilitățile Backward — pentru calcularea probabilității de emitere a unei secvențe: ex. 19.
- HMM cu emisii gaussiene: ex. 13, ex. 22.

⁶²¹Pentru calculul probabilităților Forward, notate cu $\alpha_i(t) = P(O_1, \dots, O_t, X_{t+1} = S_i)$.

⁶²²Pentru calculul probabilităților Backward, notate cu $\beta_i(t) = P(O_t O_{t+1} \dots O_T | X_t = S_i)$.

⁶²³Pentru calculul cantităților $\delta_i(t) = \max_{X_1 \dots X_{t-1}} P(X_1 \dots X_{t-1}, O_1 \dots O_{t-1}, X_t = s_i)$.

⁶²⁴Pentru „învățarea“ parametrilor (i.e., a probabilităților de tranzitie și respectiv de emisie) ai / ale unui HMM.

9.1 Probleme rezolvate

1. (Verificarea înțelegерii unor noțiuni de bază)
CMU, 2001 fall, Andrew Moore, final, pr. 7

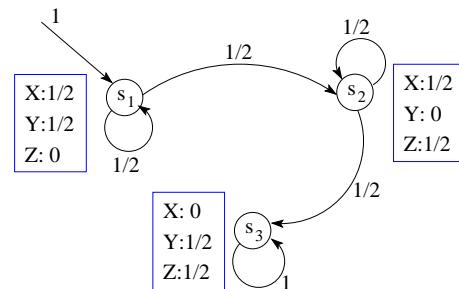
Fie un model Markov ascuns (engl., Hidden Markov Model, HMM) cu trei stări s_1, s_2, s_3 și trei simboli de emisie observabili X, Y, Z . Probabilitățile de start, de tranziție și respectiv de emisie sunt definite de următoarele matrice:

$$\begin{aligned} \pi &= \begin{bmatrix} \pi_1 = 1 \\ \pi_2 = 0 \\ \pi_3 = 0 \end{bmatrix} \\ A &= \begin{bmatrix} a_{11} = 1/2 & a_{12} = 1/2 & a_{13} = 0 \\ a_{21} = 0 & a_{22} = 1/2 & a_{23} = 1/2 \\ a_{31} = 0 & a_{32} = 0 & a_{33} = 1 \end{bmatrix} \\ B &= \begin{bmatrix} b_1(X) = 1/2 & b_1(Y) = 1/2 & b_1(Z) = 0 \\ b_2(X) = 1/2 & b_2(Y) = 0 & b_2(Z) = 1/2 \\ b_3(X) = 0 & b_3(Y) = 1/2 & b_3(Z) = 1/2 \end{bmatrix} \end{aligned}$$

- a. Desenați automatul nedeterminist corespunzător acestui model Markov.
b. Presupunând că acest model a emis secvența de semnale $XZXYYZYZZ$, identificați cea mai probabilă secvență de stări corespunzătoare, precum și probabilitatea acestei emisii. Se consideră că emisiile au loc la intrarea în stările respective.⁶²⁵ (Nu este necesară aplicarea algoritmului Viterbi.)

Răspuns:

- a. Automatul este cel din figura alăturată:
b. Deoarece în acest caz particular, în fiecare stare a automatului este o probabilitate de emisie nulă, putem identifica relativ ușor din structura sa cea mai probabilă secvență de stări pentru a fi emisă secvența de semnale $XZXYYZYZZ$, și anume: $s_1s_2s_2s_3s_3s_3s_3s_3s_3$.



Probabilitatea acestei emisii se poate calcula astfel:

$$\begin{aligned} P(X) &= \pi_1 \cdot b_1(X) = \frac{1}{2} \\ P(XZ) &= P(X) \cdot a_{12}b_2(Z) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2^3} \\ P(XZX) &= P(XZ) \cdot a_{22}b_2(X) = \frac{1}{2^3} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2^5} \\ P(XZXY) &= P(XZX) \cdot a_{23}b_3(Y) = \frac{1}{2^7} \end{aligned}$$

⁶²⁵Atenție! La fel vom proceda și în toate problemele care urmează, dacă nu se specifică altfel, în mod explicit.

$$\begin{aligned}
 P(XZXYY) &= P(XZXY) \cdot a_{33}b_3(Y) = \frac{1}{2^8} \\
 P(XZXYYZ) &= P(XZXYY) \cdot a_{33}b_3(Z) = \frac{1}{2^9} \\
 P(XZXYYZY) &= P(XZXYYZ) \cdot a_{33}b_3(Y) = \frac{1}{2^{10}} \\
 P(XZXYYZY) &= P(XZXYYZY) \cdot a_{33}b_3(Z) = \frac{1}{2^{11}} \\
 P(XZXYYZYZZ) &= P(XZXYYZY) \cdot a_{33}b_3(Z) = \frac{1}{2^{12}}
 \end{aligned}$$

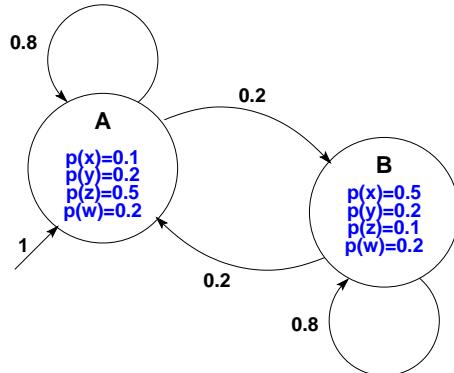
2.

(Verificarea înțelegерii unor noțiuni de bază)
CMU, 2002 fall, Andrew Moore, final, pr. 3

Considerăm modelul Markov ascuns ilustrat în figura alăturată. Notație: q_t și O_t sunt starea și respectiv semnalul observat la momentul t .

Calculați:

- $P(q_2 = A)$
- $P(O_2 = x)$
- $P(q_2 = A | O_2 = x)$
- $P(O_{100} = w)$
- Care este cea mai probabilă secvență de stări dacă $O_1 = O_2 = O_3 = O_4 = O_5 = x$. (Nu este necesară aplicarea algoritmului Viterbi.)



Răspuns:

- a. Probabilitatea ca la momentul 2 automatul să fie în starea A este:

$$P(q_2 = A) = 1 \cdot 0.8 = 0.8$$

- b. Probabilitatea ca la momentul 2 automatul să emită simbolul x este:

$$\begin{aligned}
 P(O_2 = x) &= P(O_2 = x, q_2 = A) + P(O_2 = x, q_2 = B) \\
 &= 0.8 \cdot 0.1 + 0.2 \cdot 0.5 = 0.08 + 0.1 = 0.18
 \end{aligned}$$

- c. Probabilitatea ca automatul să fie în starea A la momentul 2, știind că al doilea simbol emis este x :

$$P(q_2 = A | O_2 = x) = \frac{P(q_2 = A, O_2 = x)}{P(O_2 = x)} = \frac{0.1 \cdot 0.8}{0.18} = \frac{0.08}{0.18} = \frac{4}{9}$$

- d. Probabilitatea cerută este:

$$\begin{aligned}
 P(O_{100} = w) &= P(q_{100} = A, O_{100} = w) + P(q_{100} = B, O_{100} = w) \\
 &= 0.2 \cdot P(q_{100} = A) + 0.2 \cdot P(q_{100} = B) \\
 &= 0.2 \cdot [P(q_{100} = A) + P(q_{100} = B)] = 0.2 \cdot 1 = 0.2
 \end{aligned}$$

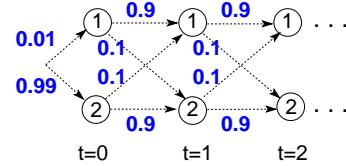
- e. Cea mai probabilă secvență de stări este $ABBBB$, încrucișat probabilitatea de a emite x este mai mare în starea B decât în starea A . Se intră inițial în starea A deoarece $\pi_A = 1$, dar apoi orice rămânere în starea A conduce la penalizare.

3.

(Verificarea înțelegерii unor noțiuni de bază)

CMU, 2003 fall, T. Mitchell, A. Moore, final, pr. 10

Considerăm un model Markov ascuns care modelează aruncarea unei monede imperfecte. Figura alăturată ilustrează tranzițiile împreună cu probabilitățile asociate lor în acest model.



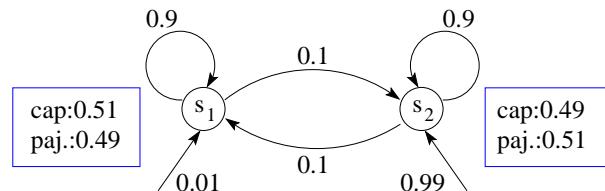
Probabilitatea ca în urma aruncării monedei să se obțină *cap* respectiv *pajură* este:

$$\begin{aligned} P(x = \text{cap} | s = 1) &= 0.51, \quad P(x = \text{pajură} | s = 1) = 0.49 \\ P(x = \text{cap} | s = 2) &= 0.49, \quad P(x = \text{pajură} | s = 2) = 0.51 \end{aligned}$$

- a. Presupunând că la trei aruncări succesive ale monedei s-a obținut de fiecare dată *cap*, care este cea mai probabilă secvență de stări? (Nu este necesar să se folosească algoritmul Viterbi.)
- b. Care este cea mai probabilă secvență de stări dacă la un număr foarte mare (de exemplu: 10^6) de aruncări consecutive ale monedei obținem un același rezultat: *cap*.

Răspuns:

Automatul este:



- a. Cea mai probabilă secvență de stări este 2, 2, 2. Probabilitățile de a obține simbolul *cap* sunt aproape identice pentru cele două stări, este foarte probabil ca sistemul să înceapă cu starea 2 și să rămână aici. Se observă că se pierde mult din probabilitate dacă se trece în starea 1. (Emiterea acestei secvențe de simboluri în starea 2 se face cu probabilitatea $0.99 \cdot 0.49 \cdot 0.9 \cdot 0.49 \cdot 0.9 \cdot 0.49$.)

- b. Când crește foarte mult numărul de aruncări și de obținere a rezultatului *cap*, crește și presiunea sistemului de a merge în starea 1, aceasta având un mic avantaj. La un moment dat se va face trecerea în starea 1, iar de aici nu există niciun avantaj de a merge înapoi în starea 2. Costul tranziției de la starea 2 la starea 1 este același indiferent de moment, însă realizarea tranziției mai devreme e mai favorabilă, întrucât crește probabilitatea obținerii unei serii lungi de *cap*. Așadar, cel mai bine este să se înceapă cu *s*₂ și să se continue imediat cu *s*₁. Se obține astfel secvența de stări: 2, 1, 1, ..., 1 și probabilitatea $0.99 \cdot 0.49 \cdot 0.1 \cdot 0.51 \cdot 0.9 \cdot 0.51 \cdot 0.9 \dots 0.51 \cdot 0.9$.

4.

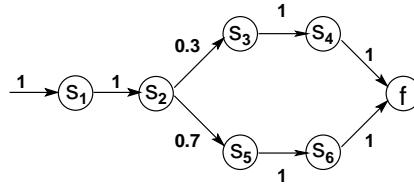
(Verificarea înțelegерii unor noțiuni de bază)

CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, final, pr. 5

Fie modelul Markov ascuns definit de matricea de tranziții și emisii din tabelul următor:

	0	s_1	s_2	s_3	s_4	s_5	s_6	f	A	C	G	T
0	0	1	0	0	0	0	0	0				
s_1	0	0	1	0	0	0	0	0	0.5	0.3	0	0.2
s_2	0	0	0	0.3	0	0.7	0	0	0.1	0.1	0.2	0.6
s_3	0	0	0	0	1	0	0	0	0.2	0	0.1	0.7
s_4	0	0	0	0	0	0	0	1	0.1	0.3	0.4	0.2
s_5	0	0	0	0	0	0	1	0	0.1	0.3	0.3	0.3
s_6	0	0	0	0	0	0	0	1	0.2	0.3	0	0.5

Automatul este cel din figura următoare:



La fiecare din punctele de mai jos, puneți unul din semnele $<$, $>$, $=$ între cei doi termeni specificați.

- a. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_1 = s_1, X_2 = s_2)$
 $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_1 = s_1, X_2 = s_2).$
- b. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4)$
 $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_3, X_4 = s_4).$
- c. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4)$
 $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_5, X_4 = s_6).$
- d. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$
 $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4).$
- e. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$
 $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_3, X_4 = s_4).$
- f. $P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$
 $P(O_1 = A, O_2 = T, O_3 = T, O_4 = G).$

Răspuns:

- a. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_1 = s_1, X_2 = s_2),$
 $p_2 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_1 = s_1, X_2 = s_2).$

Se observă că $p_1 = p_2 \cdot P(X_1 = s_1, X_2 = s_2) = p_2 \cdot 1 \cdot 1$ ($\pi_1 = 1$ și $a_{s_1 s_2} = 1$). Deci rezultă $p_1 = p_2$.

- b. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4),$
 $p_2 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_3, X_4 = s_4).$

Putem scrie că $p_1 = p_2 \cdot P(X_3 = s_3, X_4 = s_4) = p_2 \cdot 0.3 \cdot 1$. Așadar, $p_1 < p_2$.

- c. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4),$
 $p_2 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_5, X_4 = s_6).$

În acest caz se vor calcula cele două probabilități:

$$\begin{aligned}
 p_1 &= P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_3, X_4 = s_4) \cdot P(X_3 = s_3, X_4 = s_4) \\
 &= P(O_1 = A, O_2 = C) \cdot P(O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4) \\
 &= 0.5 \cdot 0.1 \cdot 0.7 \cdot 0.1 \cdot 0.3 = 0.05 \cdot 0.021 \\
 p_2 &= P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_5, X_4 = s_6) \cdot P(X_3 = s_5, X_4 = s_6) \\
 &= P(O_1 = A, O_2 = C) \cdot P(O_3 = T, O_4 = A, X_3 = s_5, X_4 = s_6) \\
 &= 0.5 \cdot 0.1 \cdot 0.3 \cdot 0.2 \cdot 0.7 = 0.05 \cdot 0.042
 \end{aligned}$$

Prin urmare, $p_1 < p_2$.

d. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$,
 $p_2 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4)$.

Se observă că $p_1 = p_2 + P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_5, X_4 = s_6)$, iar cel de-al doilea termen al doilea al acestei sume este strict pozitiv (conform calculelor de mai sus). În consecință, $p_1 > p_2$.

e. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$,
 $p_2 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_3, X_4 = s_4)$.

$$\begin{aligned}
 p_1 &= P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_3, X_4 = s_4) + \\
 &\quad + P(O_1 = A, O_2 = C, O_3 = T, O_4 = A, X_3 = s_5, X_4 = s_6) \\
 &= p_2 \cdot P(X_3 = s_3, X_4 = s_4) + \\
 &\quad + P(O_1 = A, O_2 = C, O_3 = T, O_4 = A \mid X_3 = s_5, X_4 = s_6) \cdot P(X_3 = s_5, X_4 = s_6)
 \end{aligned}$$

Rezultă că $p_1 < p_2$.

f. $p_1 = P(O_1 = A, O_2 = C, O_3 = T, O_4 = A)$,
 $p_2 = P(O_1 = A, O_2 = T, O_3 = T, O_4 = G)$.

Se calculează imediat

$$\begin{aligned}
 p_1 &= 0.5 \cdot 0.1 \cdot (0.3 \cdot 0.7 \cdot 0.1 + 0.7 \cdot 0.3 \cdot 0.2) = 0.5 \cdot 0.1 \cdot 0.7 \cdot (0.03 + 0.06) = 0.5 \cdot 0.1 \cdot 0.7 \cdot 0.09 \\
 \text{și } p_2 &= 0.5 \cdot 0.6 \cdot (0.3 \cdot 0.7 \cdot 0.4 + 0.7 \cdot 0.3 \cdot 0) = 0.5 \cdot 0.6 \cdot 0.7 \cdot 0.3 \cdot 0.4. \\
 p_1 < p_2 &\Leftrightarrow 0.1 \cdot 0.09 < 0.6 \cdot 0.12 \text{ (A). În concluzie, } p_1 < p_2.
 \end{aligned}$$

5.

(Model Markov vizibil: exemplificare; legătura cu un model Markov ascuns)

Helène Touzet, Université de Lille 1, France

Se consideră secvența de nucleotide a unui organism imaginär *Bizarrus Examinus*, care are următoarele caracteristici:

- Nucleotida actuală este cu probabilitate de 25% din bazele A, C, G sau T, dacă precedentele două nucleotide sunt identice.
- Altfel (adică: în cazul în care precedentele două nucleotide sunt diferite), probabilitatea ca nucleotida actuală să fie C sau G este de două ori mai mare decat A sau T. Mai mult, între C și G (respectiv A și T), *purinele* (A și G) vor apărea mai frecvent decât *pirimidinele* (C sau T), și anume în 60% din cazuri.

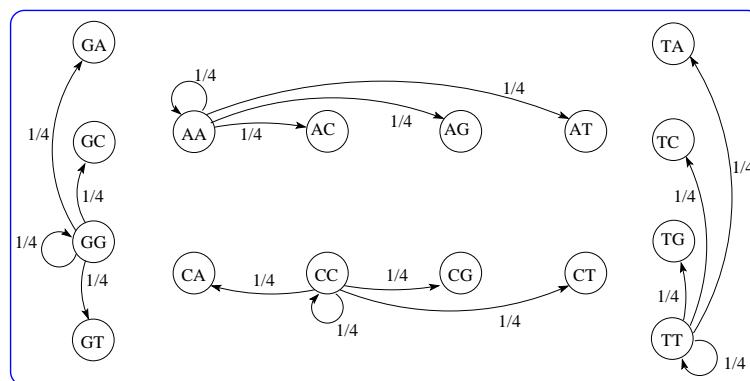
Arătați că secvența *Bizarrus Examinus* poate fi modelată folosind un model Markov vizibil; veți indica stările și probabilitățile de tranziție.

Răspuns:

Stările modelului Markov cerut sunt toate combinațiile posibile de câte 2 nucleotide, care înseamnă ultimele două nucleotide produse, fiind deci posibilă tranziția dintr-o stare în alta în cazul în care ultima nucleotidă a primei stări este egală cu prima nucleotidă din starea a două.

Observație: Fiecare stare a modelului Markov are 4 posibile tranziții. Cum există $4 \cdot 4 = 16$ stări, vor fi 72 de tranziții.

Să începem cu stările cu nucleotide identice. Din enunț obținem că probabilitățile de tranziție din aceste stări sunt $\frac{1}{4}$. Putem reprezenta aceste tranziții astfel:



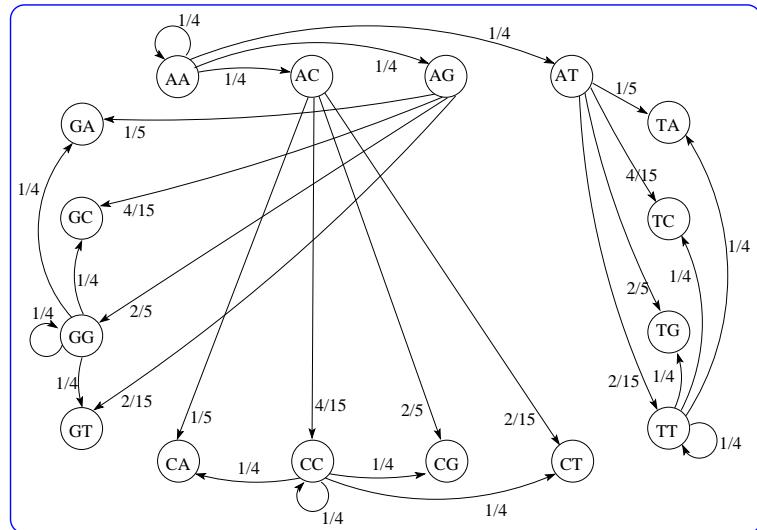
În continuare ne vom ocupa de tranzițiile din stările cu nucleotide diferite. Să notăm cu pq prima stare, unde $p \neq q$, și probabilitățile vor fi:

$$\begin{aligned}
 P(X_{i-1} = pq, X_i = qA) &= \frac{1}{3} \cdot \frac{6}{10} = \frac{1}{5} \\
 P(X_{i-1} = pq, X_i = qC) &= \frac{2}{3} \cdot \frac{4}{10} = \frac{4}{15} \\
 P(X_{i-1} = pq, X_i = qG) &= \frac{2}{3} \cdot \frac{6}{10} = \frac{2}{5} \\
 P(X_{i-1} = pq, X_i = qT) &= \frac{1}{3} \cdot \frac{4}{10} = \frac{2}{15}
 \end{aligned}$$

Altfel scris, probabilitățile de tranziție dintr-o stare cu nucleotide diferite sunt în funcție de nucleotida actuală:

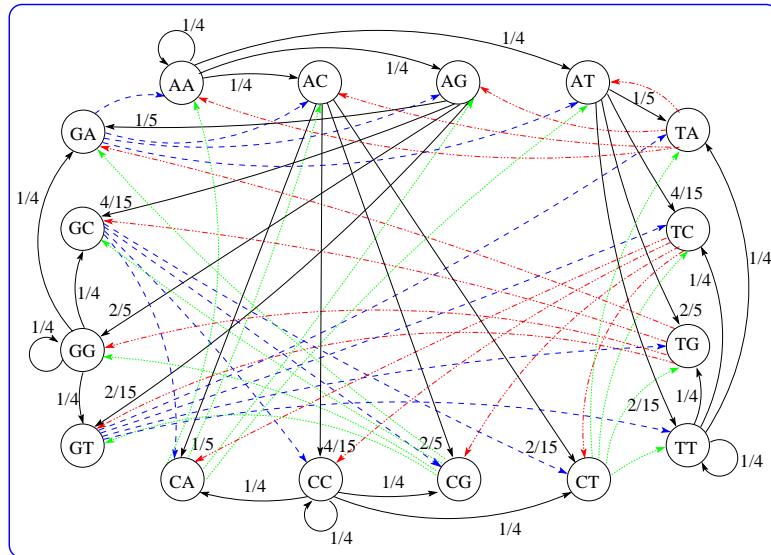
$$A : \frac{1}{5}, \quad C : \frac{4}{15}, \quad G : \frac{2}{5}, \quad T : \frac{2}{15}$$

Dacă trasăm tranzitiiile din stările AC , AG și AT obținem:



În mod similar vor fi trasate și tranzitiiile din celelalte stări (GA , GC , GT , CA , CG , CT , TA , TC și TG).

Rezultatul este un model Markov de forma:

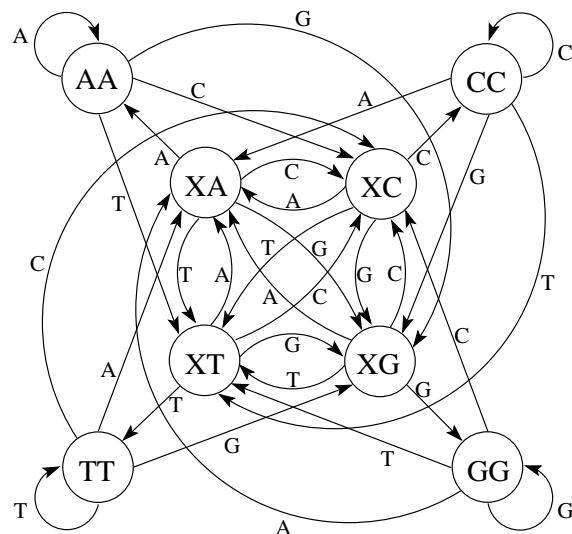


Simplificând, putem asocia acestui model Markov vizibil un model Markov ascuns cu

- 8 stări: AA , X_A (unde X semnifică literă diferită de A), CC , X_C (cu X diferit de C), GG , X_G (cu X diferit de G), TT , X_T (cu X diferit de T);
- emisii de câte un caracter, conform desenului de mai jos, având probabilitatea 1;

- tranziții având probabilități (calculate mai sus):

AA, AA	$1/4$	GG, XA	$1/4$	XA, AA	$1/5$	XG, XA	$1/5$
AA, XC	$1/4$	GG, XC	$1/4$	XA, XC	$4/15$	XG, XC	$4/15$
AA, XG	$1/4$	GG, GG	$1/4$	XA, XG	$2/5$	XG, GG	$2/5$
AA, XT	$1/4$	GG, XT	$1/4$	XA, XT	$2/15$	XG, XT	$2/15$
CC, XA	$1/4$	TT, XA	$1/4$	XC, XA	$1/5$	XT, XA	$1/5$
CC, CC	$1/4$	TT, XC	$1/4$	XC, CC	$4/15$	XT, XC	$4/15$
CC, XG	$1/4$	TT, XG	$1/4$	XC, XG	$2/5$	XT, XG	$2/5$
CC, XT	$1/4$	TT, TT	$1/4$	XC, XT	$2/15$	XT, TT	$2/15$



6.

(Algoritmul Forward: aplicare)

CMU, 2006 fall, E. Xing, T. Mitchell, final exam, pr. 6

Fie un model Markov ascuns cu stările $X_t \in \{S_1, S_2, S_3\}$, observațiile $O_t \in \{A, B, C\}$ și parametrii

$\pi_1 = 1$	$a_{11} = 1/2$	$a_{12} = 1/4$	$a_{13} = 1/4$	$b_1(A) = 1/2$	$b_1(B) = 1/2$	$b_1(C) = 0$
$\pi_2 = 0$	$a_{21} = 0$	$a_{22} = 1/2$	$a_{23} = 1/2$	$b_2(A) = 1/2$	$b_2(B) = 0$	$b_2(C) = 1/2$
$\pi_3 = 0$	$a_{31} = 0$	$a_{32} = 0$	$a_{33} = 1$	$b_3(A) = 0$	$b_3(B) = 1/2$	$b_3(C) = 1/2$

a. Calculați $P(X_5 = S_3)$.

Pentru punctele b , c și d , vom presupune că observăm secvența $AABCABC$. Se consideră că emisia simbolului O_t are loc la ieșirea din starea X_t .

b. Calculați $P(X_5 = S_3 | O = AABCABC)$.

c. Completați următorul tabel, presupunând observația $O = AABCABC$. Notație:

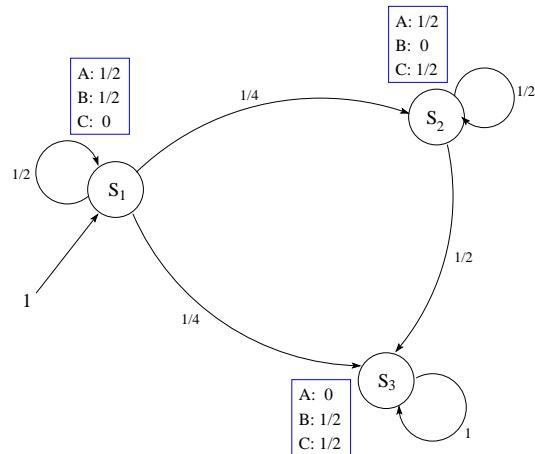
$$\alpha_i(t) = P(O_1, \dots, O_t, X_{t+1} = S_i).$$

t	1	2	3	4	5	6	7	8
$\alpha_1(t)$								
$\alpha_2(t)$								
$\alpha_3(t)$								

d. Știind că a fost emisă secvența $O = AABCABC$, scrieți secvența de stări X care are probabilitate maximă a posteriori, $P(X|O)$.

Cât este această probabilitate a posteriori?

Răspuns:



Modelul Markov este cel din figura alăturată:

a. A calcula $P(X_5 = S_3)$ enumerând toate posibilitățile pentru X_1, X_2, X_3 și X_4 conduce la prea multe variante. Probabilitatea evenimentului contrar, $P(X_5 \neq S_3) = P(X_5 = S_1 \text{ sau } X_5 = S_2)$ este însă mult mai ușor de calculat. Așadar, vom folosi formula $P(X_5 = S_3) = 1 - P(X_5 = S_1) - P(X_5 = S_2)$.

Pentru ca la momentul 5 automatul să fie în starea 1, se observă că există o singură posibilitate, și anume ca automatul să intre în starea 1 și să rămână acolo, deci:

$$P(X_5 = S_1) = 1 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{16}$$

Pentru $P(X_5 = S_2)$ avem patru posibilități, corespunzătoare celor patru momente în care se poate face trecerea din starea 1 în starea 2, deci:

$$P(X_5 = S_2) = 4 \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$$

$$P(X_5 = S_3) = 1 - \frac{1}{16} - \frac{1}{8} = \frac{13}{16}$$

b. Întrucât la momentul 5 s-a emis simbolul A , iar probabilitatea de emitere a lui A în starea 3 este 0, rezultă $P(X_5 = S_3 | O = AABCABC) = 0$.

c. Tabelul completat arată astfel:

t	1	2	3	4	5	6	7	8
O_t	A	A	B	C	A	B	C	
$\alpha_1(t)$	1	$\frac{1}{2^2}$	$\frac{1}{2^4}$	$\frac{1}{2^6}$	0	0	0	0
$\alpha_2(t)$	0	$\frac{1}{2^3}$	$\frac{1}{2^4}$	$\frac{1}{2^7}$	$\frac{1}{2^9}$	$\frac{1}{2^{11}}$	0	0
$\alpha_3(t)$	0	$\frac{1}{2^3}$	$\frac{1}{2^4}$	$\frac{5}{2^7}$	$\frac{11}{2^9}$	$\frac{1}{2^{11}}$	$\frac{1}{2^{12}}$	$\frac{1}{2^{13}}$

d. Secvența de stări este $X = S_1S_1S_1S_2S_2S_3S_3$ și are probabilitatea a posteriori $P(X | O) = 1$ datorită probabilităților specifice de emisie ale acestui model Markov.

7.

(Algoritmul Forward: aplicare)

CMU, 2008 spring, Eric Xing, HW4, pr. 3.1

Fie modelul Markov ascuns pentru cazinoul necinstit.⁶²⁶ Probabilitatea de a începe cu un zar corect / cinstit este 0.99. $P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6$ pentru un zar cinstit, iar $P(1) = P(2) = P(3) = P(4) = P(5) = 1/10$ și $P(6) = 1/2$ pentru un zar măsluit. Probabilitatea de rămânere în starea „cinstit“ este 0.95, iar cea de rămînere în starea „măsluit“ este 0.90.

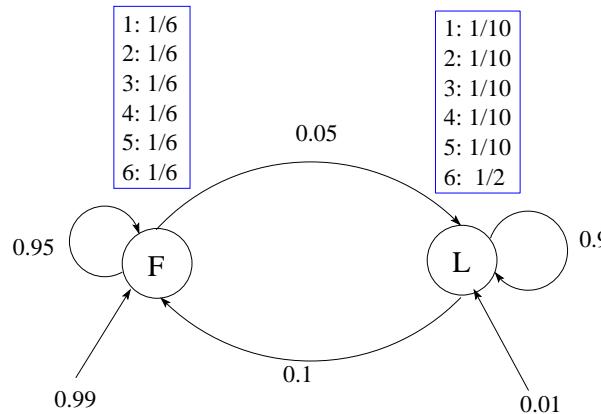
a. Calculați probabilitatea ca în patru aruncări de zar, primele trei să provină de la un zar cinstit, cu rezultatele 3, 1 și 5, iar cea de-a patra aruncare de zar să provină de la un zar măsluit, cu rezultatul 6, aşadar $P(O_1 = 3, O_2 = 1, O_3 = 5, O_4 = 6, X_1 = F, X_2 = F, X_3 = F, X_4 = L)$.

b. Calculați $P(O_1 = 3, O_2 = 1, O_3 = 5)$ folosind algoritmul Forward.

Răspuns:

Modelul Markov este următorul (se folosesc notațiile: F – Fair dice = zar cinstit, L – Loaded dice = zar măsluit):

⁶²⁶A se vedea *Biological sequence analysis*, R. Durbin et al, 1998, pag. 54-55, 56-57.



a. Probabilitatea cerută este:

$$\begin{aligned}
 & P(O_1 = 3, O_2 = 1, O_3 = 5, O_4 = 6 | X_1 = F, X_2 = F, X_3 = F, X_4 = L) = \\
 & = P(X_1 = F) \cdot P(O_1 = 3 | X_1 = F, X_2 = F) \cdot P(X_2 = F | X_1 = F) \cdot \\
 & \quad P(O_2 = 1 | X_2 = F, X_3 = F) \cdot P(X_3 = F | X_2 = F) \cdot P(O_3 = 5 | X_3 = F, X_4 = L) \cdot \\
 & \quad P(X_4 = L | X_3 = F) \cdot P(O_4 = 6 | X_4 = L) \\
 & = \pi_F \cdot b_{FF3} \cdot a_{FF} \cdot b_{FF5} \cdot a_{FF} \cdot b_{FL1} \cdot a_{FL} \cdot b_{LL6} \\
 & = 0.99 \cdot \frac{1}{6} \cdot 0.95 \cdot \frac{1}{6} \cdot 0.95 \cdot \frac{1}{6} \cdot 0.05 \cdot \frac{1}{2} = 0.000103411
 \end{aligned}$$

b. Probabilitățile Forward se calculează folosind un tabel corespunzător:

Output	1	2	3	4	5	6
t	1		2		3	
$\alpha_F(t)$	0.99		0.15685		0.024926083	
$\alpha_L(t)$	0.01		0.00915		0.002130583	
$P(O_1 \dots O_{t-1})$	1		0.166		0.027056666	
					0.004367406	

Deci $P(O_1 = 3, O_2 = 1, O_3 = 5) = 0.004367406$.

8.

(Algoritmul Backward; demonstrație – pasul inductiv)

Liviu Ciortuz

a. Redăm mai jos demonstrația formulei de recurență de la pasul inductiv al algoritmului Backward pentru HMM. Pentru fiecare egalitate din cadrul demonstrației, precizați motivul/motivele pentru care are loc egalitatea respectivă. Acolo unde este cazul, detaliați.

$$\begin{aligned}
 \beta_i(t) &= P(O_t O_{t+1} \dots O_T | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_t O_{t+1} \dots O_T | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_{t+1} \dots O_T | O_t, X_t = i, X_{t+1} = j, \mu) P(O_t | X_t = i, X_{t+1} = j, \mu) a_{ij}
 \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^N P(O_{t+1} \dots O_T | X_{t+1} = j, \mu) b_{ijO_t} a_{ij} \\
&= \sum_{j=1}^N \beta_j(t+1) b_{ijO_t} a_{ij}
\end{aligned} \tag{227}$$

b. Vă reamintim că la pasul de inițializare al algoritmului Backward se fixează $\beta_i(T+1) = 1$ pentru $i = 1, \dots, N$, unde N este numărul de stări din modelul Markov ascuns μ . Cum se poate calcula $P(O_1 O_2 \dots O_T | \mu)$, probabilitatea de emisie a unei secvențe de semnale $O_1 O_2 \dots O_T$ de către modelul μ , folosind probabilitățile Backward?

Răspuns:

a. Demonstrăm relația (1): Din definiție, $\beta_i(t)$ este probabilitatea de a emite secvența de simboluri de la O_t la O_T cu condiția ca la momentul de timp t să fie în starea s_i .

Demonstrăm că (2)=(1): Pentru probabilitățile din această egalitate se aplică definiția probabilității condiționate, obținându-se:

$$(2) = \frac{\sum_{j=1}^N P(O_t \dots O_T, X_t = i, X_{t+1} = j, \mu)}{P(X_t = i, X_{t+1} = j, \mu)} \cdot \frac{P(X_{t+1} = j, X_t = i, \mu)}{P(X_t = i, \mu)}$$

Simplificând această relație și apoi renunțând la sumă (folosind formula probabilității totale), se obține probabilitatea din relația (1):

$$(2) = \frac{P(O_t O_{t+1} \dots O_T, X_t = i | \mu)}{P(X_t = i | \mu)} = P(O_t O_{t+1} \dots O_T | X_t = i, \mu) = (1)$$

Demonstrăm egalitatea (3)=(2): Se aplică definiția probabilității de tranziție din starea i în starea j , $a_{ij} = P(X_{t+1} = j | X_t = i, \mu)$, apoi definiția probabilităților condiționate și se simplifică:

$$\begin{aligned}
(3) &= \sum_{j=1}^N P(O_{t+1} \dots O_T | O_t, X_t = i, X_{t+1} = j, \mu) P(O_t | X_t = i, X_{t+1} = j, \mu) a_{ij} \\
&= \sum_{j=1}^N \frac{P(O_t \dots O_T, X_t = i, X_{t+1} = j, \mu)}{P(O_t, X_t = i, X_{t+1} = j, \mu)} \cdot \frac{P(O_t, X_t = i, X_{t+1} = j, \mu)}{P(X_t = i, X_{t+1} = j, \mu)} \cdot \\
&\quad P(X_{t+1} = j | X_t = i, \mu) \\
&= \sum_{j=1}^N \frac{P(O_t \dots O_T, X_t = i, X_{t+1} = j, \mu)}{P(X_t = i, X_{t+1} = j, \mu)} \cdot P(X_{t+1} = j | X_t = i, \mu) \\
&= \sum_{j=1}^N P(O_t \dots O_T | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) = (2)
\end{aligned}$$

Demonstrăm egalitatea (4)=(3): b_{ijO_t} reprezintă probabilitatea de emitere a simbolului O_t la trecerea de la starea i la starea j , și deci are formula $b_{ijO_t} = P(O_t | X_t = i, X_{t+1} = j, \mu)$. Probabilitatea $P(O_{t+1} \dots O_T | O_t, X_t = i, X_{t+1} = j, \mu)$ aflată într-o sumă după j , nu depinde nici de O_t , nici de $X_t = i$, deci este egală cu $P(O_{t+1} \dots O_T | X_{t+1} = j, \mu)$. Egalitatea este astfel evidentă.

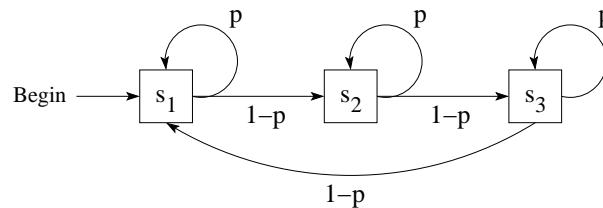
Egalitatea (4)=(5) este o chestiune de notație, $\beta_j(t+1)$ fiind probabilitatea de a emite secvența $O_{t+1} \dots O_T$ cu condiția ca la momentul $t+1$ să se afle în starea j .

b. Probabilitatea de emisie a unei secvențe de semnale $O = O_1 O_2 \dots O_T$ folosind probabilitățile Backward este:

$$P(O | \mu) = \sum_{i=1}^N \pi_i \beta_i(1).$$

9. (Algoritmii Forward, Backward, Viterbi: aplicare)
Marita Olsson, Univ. of Stockholm

Fie modelul Markov ascuns (HMM) de mai jos.



Fiecare stare/nod reprezintă o urnă care conține un număr de bile, dintre care unele sunt roșii (R) iar celelalte sunt albastre (B). O secvență observabilă x este generată de acest HMM în modul următor:

- se alege la întâmplare o bilă din prima urnă, după care se notează culoarea ei (R sau B), apoi bila este repusă în urnă;
- procedura de mai sus se repetă de mai multe ori, cu singura diferență că bila se alege cu probabilitatea p din aceeași urnă ca la pasul precedent și cu probabilitatea $1 - p$ din urna următoare.

În prima urnă sunt 3 bile roșii și 2 bile albastre, a doua urnă conține o bilă roșie și 4 bile albastre, iar în a treia urnă sunt 4 bile roșii și o bilă albastră. Se consideră $p = 1/3$.

- a. Calculați $P(x = RBB)$.

În continuare, se consideră secvența $y = RRB BB$. Pentru a determina π^* , calea cea mai probabilă pentru generarea acestei secvențe, s-a rulat algoritmul Viterbi și s-au obținut rezultatele de mai jos:

O		R		R		B		B		B	
t	1		2		3		4		5		6
$\delta_1(t)$	1		$1/5$		$1/25$		$2/375$		$32/5625$		$128/84375$
$\delta_2(t)$	0		$2/5$		$2/25$		$8/375$		$32/5625$		$128/84375$
$\delta_3(t)$	0		0		$4/75$		$16/375$		$64/5625$		$256/84375$
$\psi_1(t)$	1		1		1		1		3		3
$\psi_2(t)$	–		1		1		2		2		1 sau 2
$\psi_3(t)$	–		–		2		2		2		2

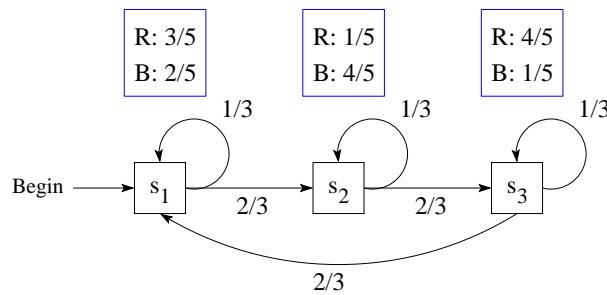
- b. Arătați cum a fost calculat $\delta_2(3)$.

Observație: Am notat cu $\delta_t(i)$ probabilitatea calculată de către algoritm Viterbi: cea mai mare dintre probabilitățile de generare a secvenței de semnale $O_1 \dots O_t$, cu ajungere (la finalul generării secvenței) în starea i .

- c. Determinați π^* .
- d. Presupunând că se dă secvența $O = O_1 O_2 \dots O_{100}$, precizați cum anume se poate calcula probabilitatea ca O_{77} să fie generat în starea s_3 .

Răspuns:

Modelul Markov ascuns (HMM) corespunzător datelor problemei este:



- a. Probabilitatea cerută $P(x = RBB)$ se poate calcula în mai multe moduri:

Metoda 1 (Forță brută): Pentru producerea secvenței de semnale RBB sunt posibile 4 drumuri: s_1, s_1, s_1 ; s_1, s_1, s_2 ; s_1, s_2, s_2 ; sau s_1, s_2, s_3 . Se poate calcula probabilitatea pentru fiecare dintre acestea:

$$\begin{aligned}
 p_1 &= P(x = RBB, X = 111) = \frac{3}{5} \cdot \frac{1}{3} \cdot \frac{2}{5} \cdot \frac{1}{3} \cdot \frac{2}{5} = \frac{2^2}{5^3 \cdot 3} \\
 p_2 &= P(x = RBB, X = 112) = \frac{3}{5} \cdot \frac{1}{3} \cdot \frac{2}{5} \cdot \frac{2}{3} \cdot \frac{4}{5} = \frac{2^4}{5^3 \cdot 3} \\
 p_3 &= P(x = RBB, X = 122) = \frac{3}{5} \cdot \frac{2}{3} \cdot \frac{4}{5} \cdot \frac{1}{3} \cdot \frac{4}{5} = \frac{2^5}{5^3 \cdot 3} \\
 p_4 &= P(x = RBB, X = 123) = \frac{3}{5} \cdot \frac{2}{3} \cdot \frac{4}{5} \cdot \frac{2}{3} \cdot \frac{1}{5} = \frac{2^4}{5^3 \cdot 3}
 \end{aligned}$$

Probabilitatea secvenței de semnale RBB este suma acestor probabilități:

$$P(x = RBB) = \sum_{i=1}^4 p_i = \frac{4 + 16 + 32 + 16}{375} = \frac{68}{375}$$

Metoda 2 (Algoritmul Forward): Se calculează probabilitățile Forward $\alpha_i(t)$:⁶²⁷

x	R	B	B	
t	1	2	3	4
$\alpha_1(t)$	1	1/5	2/75	36/1125
$\alpha_2(t)$	0	2/5	4/25	56/1125
$\alpha_3(t)$	0	0	16/75	112/1125
$P(x)$	1	3/5	2/5	68/375

$$\alpha_1(2) = 1 \cdot 1/3 \cdot 3/5 = 1/5$$

$$\alpha_2(2) = 1 \cdot 2/3 \cdot 3/5 = 2/5$$

$$\alpha_3(2) = 0$$

$$\alpha_1(3) = 1/5 \cdot 1/3 \cdot 2/5 = 2/75$$

$$\alpha_2(3) = 1/5 \cdot 2/3 \cdot 2/5 + 2/5 \cdot 1/3 \cdot 4/5 = 4/75 + 8/75 = 12/75 = 4/25$$

$$\alpha_3(3) = 2/5 \cdot 2/3 \cdot 4/5 = 16/75$$

$$\alpha_1(4) = 2/75 \cdot 1/3 \cdot 2/5 + 16/75 \cdot 2/3 \cdot 1/5 = \frac{36}{3^2 \cdot 5^3}$$

$$\alpha_2(4) = 2/75 \cdot 2/3 \cdot 2/5 + 4/25 \cdot 1/3 \cdot 4/5 = \frac{56}{3^2 \cdot 5^3}$$

$$\alpha_3(4) = 4/25 \cdot 2/3 \cdot 4/5 + 16/75 \cdot 1/5 \cdot 1/3 = \frac{112}{3^2 \cdot 5^3}$$

Metoda 3 (Algoritmul Backward): Se calculează probabilitățile Backward $\beta_i(t)$:⁶²⁸

x	R	B	B	
t	1	2	3	4
$\beta_1(t)$	68/375	4/15	2/5	1
$\beta_2(t)$	34/1125	8/25	4/5	1
$\beta_3(t)$	4/25	1/15	1/5	1
$P(x)$	68/375			

$$\beta_1(3) = 1 \cdot 1/3 \cdot 2/5 \cdot 1 + 1 \cdot 2/3 \cdot 2/5 = 2/5$$

$$\beta_2(3) = 1 \cdot 1/3 \cdot 4/5 + 1 \cdot 2/3 \cdot 4/5 = 4/5$$

$$\beta_3(3) = 1 \cdot 1/3 \cdot 1/5 + 1 \cdot 2/3 \cdot 1/5 = 1/5$$

$$\beta_1(2) = 2/5 \cdot 1/3 \cdot 2/5 + 4/5 \cdot 2/3 \cdot 2/5 = 4/15$$

$$\beta_2(2) = 4/5 \cdot 1/3 \cdot 4/5 + 1/5 \cdot 2/3 \cdot 4/5 = 8/25$$

$$\beta_3(2) = 2/5 \cdot 2/3 \cdot 1/5 + 1/5 \cdot 1/3 \cdot 1/5 = 1/15$$

⁶²⁷ $\alpha_i(t) \stackrel{\text{def.}}{=} P(O_1 \dots O_t, X_{t+1} = s_i)$, cu $\alpha_i(1) = \pi_i$, pentru $i = 1, 2, 3$, iar pentru $t \leq T$, $\alpha_i(t+1) = \sum_{j=1}^N \alpha_j(t) \cdot a_{ji} \cdot b_{jiO_t}$, unde N este numărul de stări, a_{ji} este probabilitatea de tranziție din starea s_i în starea s_j , iar b_{jiO_t} este probabilitatea de emisie a simbolului O_t la trecerea din starea s_i în starea s_j .

⁶²⁸ $\beta_i(t) \stackrel{\text{def.}}{=} P(O_t O_{t+1} \dots O_T | X_t = s_i)$, cu $\beta_i(T+1) = 1$, pentru $i = 1, 2, 3$, iar pentru $t \leq T$, conform problemei 8, $\beta_i(t) = \sum_{j=1}^N \beta_j(t+1) \cdot a_{ji} \cdot b_{jiO_t}$.

$$\begin{aligned}\beta_1(1) &= 4/15 \cdot 1/3 \cdot 3/5 + 8/25 \cdot 2/3 \cdot 3/5 = \frac{4}{3 \cdot 5^2} + \frac{16}{5^3} = \frac{68}{3 \cdot 5^3} \\ \beta_2(1) &= 8/25 \cdot 1/3 \cdot 1/5 + 1/15 \cdot 2/3 \cdot 1/5 = \frac{8}{3 \cdot 5^3} + \frac{2}{3^2 \cdot 5^2} = \frac{34}{3^2 \cdot 5^3} \\ \beta_3(1) &= 4/15 \cdot 2/3 \cdot 4/5 + 1/15 \cdot 1/3 \cdot 4/5 = \frac{32}{3^2 \cdot 5^2} + \frac{4}{3^2 \cdot 5^2} = \frac{4}{5^2}\end{aligned}$$

b. $\delta_2(3)$ se calculează după formula:

$$\begin{aligned}\delta_2(3) &= \max(\delta_1(2) \cdot a_{12} \cdot b_1(R); \delta_2(2) \cdot a_{22} \cdot b_2(R); \delta_3(2) \cdot a_{32} \cdot b_3(R)) \\ \Rightarrow \delta_2(3) &= \max\left(\frac{1}{5} \cdot \frac{2}{3} \cdot \frac{3}{5}; \frac{2}{5} \cdot \frac{1}{3} \cdot \frac{1}{5}; 0\right) \Rightarrow \delta_2(3) = \max\left(\frac{2}{25}; \frac{2}{75}\right) \Rightarrow \delta_2(3) = \frac{2}{25}\end{aligned}$$

c. Pentru a determina π^* folosind algoritmul Viterbi se procedează astfel: Se alege maximul de pe ultima coloană (din primele 3 rânduri - adică dintre $\delta_i(T)$). Acesta este obținut pe linia 3, deci $\delta_3(6)$. Așadar, ultima stare a drumului va fi s_3 . În continuare se trece în tabel la $\psi_3(6)$, unde este valoarea 2. Această valoare indică penultima stare ca fiind s_2 și de asemenea linia pe care se trece în continuare pentru a reconstituи calea cea mai probabilă de generare a secvenței respective. Se trece pe linia 2; de aici se continuă tot pe linia 2 de două ori, apoi se trece la linia 1. Se obține în final drumul: $s_1 s_1 s_2 s_2 s_2 s_3$.

O	R	R	B	B	B	
t	1	2	3	4	5	6
δ_1	1	1/5	1/25	2/375	32/5625	128/84375
δ_2	0	2/5	2/25	8/375	32/5625	128/84375
δ_3	0	0	4/75	16/375	64/5625	256/84375
Ψ_1	1	1	1	1	3	3
Ψ_2	-	1	1	2	2	1 sau 2
Ψ_3	-	-	2	2	2	2

d. Stîm că $P(O, X_t = s_i) = \alpha_i(t) \cdot \beta_i(t)$, unde $\alpha_i(t)$ și $\beta_i(t)$ sunt probabilităile Forward și respectiv Backward. Așadar:

$$P(X_{77} = s_3 | O) = \frac{P(X_{77} = s_3, O_1 \dots O_{100})}{P(O | \mu)} = \frac{\alpha_3(77) \cdot \beta_3(77)}{\sum_{i=1}^3 \alpha_i(101)}$$

10.

(Demonstrația formulei necesare pentru calculul mediilor variabilelor neobservabile corespunzătoare tranzitiei în algoritmul Forward-Backward)

Liviu Ciortuz

Fie un model Markov ascuns având probabilităile de tranzitie $a_{ij} = P(X_{t+1} = j | X_t = i, \mu)$ și probabilităile de emisie $b_{ijO_t} = P(O_t | X_{t+1} = j, X_t = i, \mu)$.

Demonstrați formula:

$$P(X_t = i, X_{t+1} = j | O, \mu) = \frac{\alpha_i(t) a_{ij} b_{ijO_t} \beta_j(t+1)}{P(O | \mu)}$$

unde $\alpha_i(t) \stackrel{not.}{=} P(O_1 O_2 \cdots O_{t-1}, X_t = i \mid \mu)$ și $\beta_i(t) \stackrel{not.}{=} P(O_t O_{t+1} \cdots O_T \mid X_t = i, \mu)$.

Răspuns:

Se pornește de la membrul stâng al relației și se aplică formula probabilității condiționate de mai multe ori (fie în sens direct fie în sens invers):

$$\begin{aligned}
P(X_t = i, X_{t+1} = j \mid O, \mu) &= \frac{P(X_t = i, X_{t+1} = j, O \mid \mu)}{P(O \mid \mu)} \\
&= \frac{1}{P(O, \mu)} \cdot P(O_1 O_2 \cdots O_{t-1} O_t O_{t+1} \cdots O_T, X_t = i, X_{t+1} = j) \\
&= \frac{1}{P(O, \mu)} P(O_{t+1} \cdots O_T \mid X_t = i, X_{t+1} = j, O_1 \dots O_t) P(X_t = i, X_{t+1} = j, O_1 \dots O_t) \\
&= \frac{1}{P(O, \mu)} P(O_{t+1} \cdots O_T \mid X_{t+1} = j) P(X_t = i, X_{t+1} = j, O_1 \dots O_t) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot P(X_t = i, X_{t+1} = j, O_1 \dots O_t) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot P(O_1 \dots O_{t-1} O_t, X_t = i, X_{t+1} = j) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) P(O_1 \dots O_{t-1} \mid O_t, X_t = i, X_{t+1} = j) P(O_t, X_t = i, X_{t+1} = j) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot P(O_1 \dots O_{t-1} \mid X_t = i) \cdot P(O_t, X_t = i, X_{t+1} = j) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot \alpha_i(t) \cdot \frac{P(O_t, X_t = i, X_{t+1} = j)}{P(X_t = i)} \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot \alpha_i(t) \cdot \frac{P(O_t \mid X_t = i, X_{t+1} = j) \cdot P(X_t = i, X_{t+1} = j)}{P(X_t = i)} \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot \alpha_i(t) \cdot b_{ij O_t} \cdot \frac{P(X_t = i, X_{t+1} = j)}{P(X_t = i)} \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot \alpha_i(t) \cdot b_{ij O_t} \cdot P(X_{t+1} = j \mid X_t = i) \\
&= \frac{1}{P(O, \mu)} \cdot \beta_j(t+1) \cdot \alpha_i(t) \cdot b_{ij O_t} \cdot a_{ij} = \frac{\alpha_i(t) a_{ij} b_{ij O_t} \beta_j(t+1)}{P(O \mid \mu)}
\end{aligned}$$

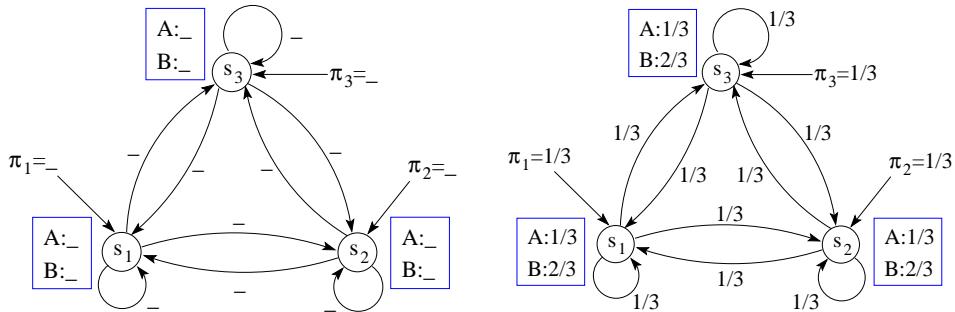
11.

(HMM: noțiuni de bază; algoritmul Forward-Backward)

CMU, 2003 fall, T. Mitchell, A. Moore, final, pr. 10.cd

a. Se consideră modelul Markov ascuns de mai jos, în partea stângă, cu trei stări s_1, s_2 și s_3 . Probabilitățile de intrare în fiecare dintre stări sunt π_1, π_2 , respectiv π_3 .

Stabilii un set de probabilități astfel încât modelul Markov rezultat să maximizeze probabilitatea obținerii secvenței ABA .



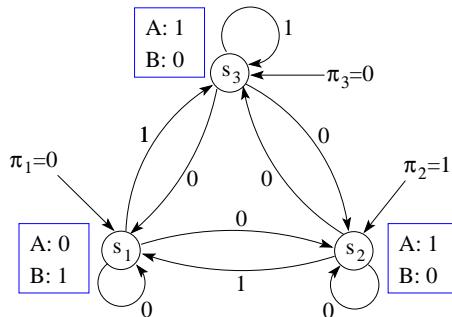
b. Se va folosi algoritmul EM (cunoscut pentru HMM și sub numele Forward-Backward sau Baum-Welch⁶²⁹) pentru estimarea parametrilor modelului Markov ascuns. Înaintea primei iterării s-au realizat inițializările prezentate în figura de mai sus din partea dreaptă. Pentru aceste valori initiale, va converge cu succes algoritmul EM la un model care maximizează probabilitatea secvenței *ABA*?

Răspuns:

- a. Pentru ca modelul Markov ascuns obținut să fie valid, trebuie ca parametrii acestuia, adică probabilitățile, să respecte următoarele ecuații:

$$\begin{aligned}\pi_1 + \pi_2 + \pi_3 &= 1 \\ a_{i1} + a_{i2} + a_{i3} &= 1, \forall i \in \{1, 2, 3\} \\ b_i(A) + b_i(B) &= 1, \forall i \in \{1, 2, 3\}\end{aligned}$$

Sunt mai multe modele Markov ascunse pentru care probabilitatea obținerii secvenței *ABA* să fie 1, adică maximă. Unul dintre acestea este următorul:



⁶²⁹Fizicianul Ruslan Stratonovich este cel dintâi care are prezentat algoritmul Forward-Backward, în anul 1960, în articolul *Conditional Markov Processes* din revista *Theory of Probability and Its Applications*, 5(2):156–178, în limba rusă. Procesele Markov poartă numele matematicianului rus Andrei Markov (1856-1922). Andrei Markov și fratele său Vladimir Markov (1871-1896) sunt cei care au demonstrat inegalitatea probabilistă cunoscută sub numele de inegalitatea [frăților] Markov; vedeți pr. 100.a de la capitolul de *Fundamente*. Vladimir Markov a murit de tuberculoză la vîrstă de doar 25 de ani.

Leonard Baum (1931 – 2017) și Lloyd Welch (1927-) au descris algoritmul Forward-Backward (și formalismul teoretic al modelelor Markov ascunse) în limba engleză, în cadrul unor cercetări întreprinse la sfârșitul anilor '60 și începutul anilor '70 din secolul trecut, la *Institute for Defence Analyses* din S.U.A. (Cf. Lawrence Rabiner, *First-Hand: The Hidden Markov Model*, https://ethw.org/First-Hand:The_Hidden_Markov_Model.)

Într-adevăr, în acest model, secvența ABA poate fi obținută pe următoarea cale: $q_1 = s_2$, $q_2 = s_1$, $q_3 = s_3$.

b. Algoritmul EM (Forward-Backward) constă în repetarea iterativă a următorilor doi pași:

Estimare: Folosind probabilități Forward și Backward, $\alpha_i(t)$ și respectiv $\beta_i(t)$, se calculează $p_t(i, j) = \frac{\alpha_i(t) a_{ij} b_{ij}(O_t) \beta_j(t+1)}{\sum_{m=1}^N \alpha_m(t) \beta_m(t)}$ pentru $t \in \{1, 2, 3\}$ și $i, j \in \{1, 2, 3\}$.⁶³⁰

Maximizare: Se recalculează probabilitățile inițiale ale modelului

$$\hat{\pi}_i = \sum_{j=1}^N p_1(i, j) \quad \hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{l=1}^N \sum_{t=1}^T p_t(i, l)} \quad \hat{b}_{ijk} = \frac{\sum_{t: O_t=k, 1 \leq t \leq T} p_t(i, j)}{\sum_{t=1}^T p_t(i, j)}$$

Vom aplica prima iterație a algoritmului EM. Pentru aceasta trebuie calculate probabilitățile Forward și Backward. Datorită simetriei probabilităților celor trei stări s_1 , s_2 și s_3 în modelul considerat, vom avea $\alpha_1(t) = \alpha_2(t) = \alpha_3(t)$ și $\beta_1(t) = \beta_2(t) = \beta_3(t)$, $\forall t$ fixat. Aceste probabilități sunt:

Output	A	B	A
t	1	2	3
$\alpha_i(t)$, $i \in \{1, 2, 3\}$	$1/3$	$1/3^2$	$2/3^3$
$\beta_i(t)$, $i \in \{1, 2, 3\}$	$2/3^3$	$2/3^2$	$1/3$
			4

Se observă că probabilitatea de a obține secvența ABA este foarte mică:

$$P(ABA) = \sum_i \alpha_i(4) = 3 \cdot \frac{2}{3^4} = \frac{2}{3^3} \approx 0.074$$

Pasul de estimare:

$$\begin{aligned} p_1(i, j) &= \frac{\alpha_i(1) a_{ij} b_{ij}(A) \beta_j(2)}{\sum_{m=1}^3 \alpha_m(1) \beta_m(1)} = \frac{\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3^2}}{3 \left(\frac{1}{3} \cdot \frac{2}{3^3} \right)} = \frac{1}{3^2} \\ p_2(i, j) &= \frac{\alpha_i(2) a_{ij} b_{ij}(B) \beta_j(3)}{\sum_{m=1}^3 \alpha_m(2) \beta_m(2)} = \frac{\frac{1}{3^2} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}}{3 \left(\frac{1}{3^2} \cdot \frac{2}{3^2} \right)} = \frac{1}{3^2} \\ p_3(i, j) &= \frac{\alpha_i(3) a_{ij} b_{ij}(A) \beta_j(4)}{\sum_{m=1}^3 \alpha_m(3) \beta_m(3)} = \frac{\frac{2}{3^3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 1}{3 \left(\frac{2}{3^3} \cdot \frac{1}{3} \right)} = \frac{1}{3^2} \end{aligned}$$

⁶³⁰Numitorul $\sum_{m=1}^N \alpha_m(t) \beta_m(t)$ este probabilitatea generării secvenței de semnale $O_1 \dots O_T$. (Formula aceasta este una dintre posibilitățile de calculare a acestor probabilități.) Numărătorul $\alpha_i(t) a_{ij} b_{ij}(O_t) \beta_j(t+1)$ este probabilitatea ca la momentul t să se treacă din starea i în starea j , știind că a fost emisă secvența $O_1 \dots O_T$. Raportul acestor două probabilități, adică $p_t(i, j)$, poate fi interpretat ca fiind media unei variabile (aleatoare) indicator neobservabile, $X_t(i, j)$, care ia valoarea 1 dacă la momentul t se face trecerea din starea i în starea j (și se emite semnalul O_t), și 0 în caz contrar.

Pasul de maximizare:

$$\begin{aligned}\hat{\pi}_i &= \sum_{j=1}^3 p_1(i, j) = 3 \cdot \frac{1}{3^2} = \frac{1}{3} \text{ pentru } i \in \{1, 2, 3\} \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^3 p_t(i, j)}{\sum_{l=1}^3 \sum_{t=1}^3 p_t(i, l)} = \frac{3 \cdot \frac{1}{3^2}}{3 \cdot 3 \cdot \frac{1}{3^2}} = \frac{1}{3} \text{ pentru } i, j \in \{1, 2, 3\} \\ \hat{b}_{ij}(A) &= \frac{\sum_{t \in \{1, 3\}} p_t(i, j)}{\sum_{t=1}^3 p_t(i, j)} = \frac{2 \cdot \frac{1}{3^2}}{3 \cdot \frac{1}{3^2}} = \frac{2}{3} \text{ pentru } i, j \in \{1, 2, 3\} \\ \hat{b}_{ij}(B) &= \frac{\sum_{t \in \{2\}} p_t(i, j)}{\sum_{t=1}^3 p_t(i, j)} = \frac{1 \cdot \frac{1}{3^2}}{3 \cdot \frac{1}{3^2}} = \frac{1}{3} \text{ pentru } i, j \in \{1, 2, 3\}\end{aligned}$$

Se observă că nu se modifică probabilitățile de intrare π_i și nici probabilitățile de tranziție a_{ij} . De asemenea, deși se modifică probabilitățile de emisie, acestea își păstrează simetria: $b(A)$, și respectiv, $b(B)$ sunt egale pentru toate cele 3 stări. (De fapt, s-au inversat valorile precedente pentru $b(A)$ și $b(B)$.)

Probabilitățile Forward și Backward recalculate sunt:

<i>Output</i>		<i>A</i>		<i>B</i>		<i>A</i>	
<i>t</i>	1		2		3		4
$\alpha_i(t)$, $i \in \{1, 2, 3\}$	1/3		2/3 ²		2/3 ³		4/3 ⁴
$\beta_i(t)$, $i \in \{1, 2, 3\}$	4/3 ³		2/3 ²		2/3		1

iar probabilitatea de a obține secvența *ABA* s-a dublat:

$$P(ABA) = \sum_i \alpha_i(4) = 3 \cdot \frac{4}{3^4} = \frac{4}{3^3} \approx 0.148$$

Se poate constata ușor (prin calcule) că orice iterație ulterioară a algoritmului EM nu va aduce nicio modificare, toate probabilitățile rămânând neschimbate.

În concluzie, pentru valorile inițiale date în enunț, algoritmul EM nu converge la un model care maximizează probabilitatea secvenței *ABA*. (Vă reamintim că la punctul *a* am demonstrat că această probabilitate maximă este 1.)

12.

(O proprietate a algoritmului Forward-Backward)

*Christopher Manning, Hinrich Schütze,
Foundations of statistical natural language processing, 1999*

Demonstrați că dacă într-un HMM o emisie/tranziție are probabilitate 0, atunci algoritmul Forward-Backward o lasă neschimbată.

Răspuns:

Algoritmul Forward-Backward are ca date de intrare probabilităile modelului Markov: π_i , a_{ij} și b_{ijk} . Folosind probabilități Forward și Backward, $\alpha_i(t)$ și respectiv $\beta_i(t)$, algoritmul recalculează probabilităile inițiale, obținând $\hat{\pi}_i$, \hat{a}_{ij} și \hat{b}_{ijk} .

Arătăm că dacă $a_{ij} = 0$ atunci $\hat{a}_{ij} = 0$, iar dacă $b_{ijk} = 0$ atunci $\hat{b}_{ijk} = 0$.

Algoritmul Forward-Backward constă din mai multe iterări, fiecare iterare constând în doi pași:

- *Estimare*: Se calculează $p_t(i, j) = \frac{\alpha_i(t)a_{ij}b_{ijk}\beta_j(t+1)}{\sum_{m=1}^N \alpha_m(t)\beta_m(t)}$.

Dacă oricare dintre a_{ij} sau b_{ijk} (unde $O_t = k$) este 0, rezultă că $p_t(i, j) = 0$, indiferent de valoarea lui t .

- *Maximizare*:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{l=1}^N \sum_{t=1}^T p_t(i, l)}$$

Dacă $a_{ij} = 0$, atunci $p_t(i, j) = 0$, deci $\hat{a}_{ij} = 0$.

$$\hat{b}_{ijk} = \frac{\sum_{t: O_t=k, 1 \leq t \leq T} p_t(i, j)}{\sum_{t=1}^T p_t(i, j)}$$

Dacă $b_{ijk} = 0$, atunci $p_t(i, j) = 0$ pentru orice t pentru care $O_t = k$, deci $\hat{b}_{ijk} = 0$.

Am demonstrat astfel că algoritmul Forward-Backward lasă aceste probabilități neschimbăte, dacă ele sunt inițial nule.

13.

(Un model Markov ascuns cu emisii gaussiene)

CMU, 2008 fall, Eric Xing, final, pr. 8

Fie un model Markov ascuns cu emisii continue, având K stări. Notăm cu x_i emisia de la momentul i , iar cu z_i starea ascunsă corespunzătoare. Emisiile stării $k \in \{1, \dots, K\}$ au o distribuție gaussiană de medie μ_k și deviație standard σ_k .⁶³¹ Prin urmare, probabilitățile de emisie sunt $P(x_i | z_i = k, \theta) = \mathcal{N}(x_i | \mu_k, \sigma_k)$. θ reprezintă mulțimea parametrilor modelului, adică probabilitățile inițiale π , matricea probabilităților de tranziție a , mediile μ_1, \dots, μ_K și deviațiile standard $\sigma_1, \dots, \sigma_K$.

- Calculați log-verosimilitatea secvenței de emisii (x_1, \dots, x_n) dacă succesiunea de stări în care se produc aceste emisii este (z_1, \dots, z_n) .
- Calculați formulele de update pentru probabilitățile Forward și Backward pentru acest HMM și explicați pe scurt diferența față de cele studiate în cazul discret.
- Presupunem că ni se dă secvența de observații $X = (x_1, \dots, x_n)$ și secvența de stări corespunzătoare, $Z = (z_1, \dots, z_n)$. Vrem să găsim valorile parametrilor θ pentru acest model Markov ascuns, folosind un algoritm de tipul EM / Forward-Backward.

Sunt oare relațiile de actualizare pentru probabilitățile a_{ij} și π_i diferite de cele ale modelului Markov ascuns cu emisii discrete? Explicați de ce da, sau de ce nu.

Care vor fi relațiile de actualizare pentru parametrii μ_k și σ_k ai distribuțiilor gaussiene care modelează emisiile? Sugestie: Nu este necesar să derivați aceste relații. Date fiind

⁶³¹Se va considera că aceste emisii depind doar de starea k , nu și de stările în care se poate trece pornind din starea k .

stările, emisiile sunt independente unele de altele, iar fiecare emisie urmează distribuția gaussiană asociată stării în care se produce emisia respectivă.

Pentru punctele d și e de mai jos, vom presupune că ni se dă doar secvența de observații $X = (x_1, \dots, x_n)$. Vrem să găsim valorile parametrilor θ pentru acest model Markov ascuns.

d. Algoritmul de învățare nesupervizată EM / Forward-Backward optimizează la fiecare iterație media log-verosimilității datelor complete. Oare de ce este aceasta o funcție obiectiv rezonabilă?

Scrieți expresia acestei funcții (notată cu $E[\ell_c(x, y; \theta)]$) pentru modelul Markov ascuns cu emisii gaussiene. Nu este necesar să demonstrați cum anume se obține această expresie.

e. Presupunem că vrem să găsim estimările de verosimilitate maximă ($\hat{\mu}_k$ și $\hat{\sigma}_k$) pentru parametrii μ_k și σ_k . Vor avea oare formulele [pentru obținerea] acestor estimări exact aceeași formă ca și în cazul mixturilor de distribuții gaussiene? Justificați pe scurt. *Sugestie:* Identificați acei termeni din expresia funcției $E[\ell_c(x, y; \theta)]$ care sunt relevanți pentru optimizare (adică, termenii care conțin parametrii μ_k și / sau σ_k).

Răspuns:

a. Log-verosimilitatea condițională cerută este:

$$\begin{aligned} \ln P(x_1, \dots, x_n | z_1, \dots, z_n) &= \ln \prod_{i=1}^n P(x_i | z_i) = \ln \prod_{i=1}^n \mathcal{N}(x_i | \mu_{z_i}, \sigma_{z_i}) \\ &= \sum_{i=1}^n \ln \mathcal{N}(x_i | \mu_{z_i}, \sigma_{z_i}) = \sum_{i=1}^n \left[-\ln(\sqrt{2\pi}\sigma_{z_i}) - \frac{(x_i - \mu_{z_i})^2}{2\sigma_{z_i}^2} \right] \end{aligned}$$

b. Probabilitățile Forward se calculează astfel:

$$\begin{aligned} \alpha_k(t+1) &\stackrel{\text{def.}}{=} P(x_1, \dots, x_t, z_{t+1} = k) \\ &= \sum_i P(x_1, \dots, x_t, z_{t+1} = k, z_t = i) \\ &= \sum_i P(x_1, \dots, x_{t-1}, z_t = i) \cdot P(x_t, z_{t+1} = k | x_1, x_2, \dots, x_{t-1}, z_t = i) \\ &= \sum_i \alpha_i(t) \cdot P(x_t, z_{t+1} = k | z_t = i) \\ &= \sum_i \alpha_i(t) \cdot P(x_t | z_{t+1} = k, z_t = i) \cdot P(z_{t+1} = k | z_t = i) \\ &= \sum_i \alpha_i(t) \cdot \mathcal{N}(x_t | \mu_k, \sigma_k) \cdot a_{ik} = \mathcal{N}(x_t | \mu_k, \sigma_k) \sum_i \alpha_i(t) \cdot a_{ik} \end{aligned}$$

Similar, probabilitățile Backward se calculează astfel:

$$\begin{aligned} \beta_k(t) &\stackrel{\text{def.}}{=} P(x_t, \dots, x_T | z_t = k) \\ &= \sum_i P(x_t, \dots, x_T, z_{t+1} = i | z_t = k) \\ &= \sum_i P(x_t, \dots, x_T | z_{t+1} = i, z_t = k) \cdot \underbrace{P(z_{t+1} = i | z_t = k)}_{=a_{ki}} \\ &= \sum_i P(x_{t+1}, \dots, x_T | x_t, z_{t+1} = i, z_t = k) \cdot P(x_t | z_{t+1} = i, z_t = k) \cdot a_{ki} \end{aligned}$$

$$\begin{aligned}
&= \sum_i P(x_{t+1}, \dots, x_T | z_{t+1} = i) \cdot \mathcal{N}(x_t | \mu_i, \sigma_i) \cdot a_{ki} \\
&= \sum_i \beta_i(t+1) \cdot \mathcal{N}(x_t | \mu_i, \sigma_i) \cdot a_{ki}
\end{aligned}$$

Așadar, formulele de recurență pentru un model Markov ascuns cu emisii continue au o formă similară cu cele din cazul discret,⁶³² singura diferență constând în probabilitățile de emisie, care în cazul continuu sunt modelate de distribuții probabiliste gaussiene.

c. Relațiile de actualizare pentru probabilitățile a_{ij} și π_i sunt aceleași cu cele din cazul emisiilor discrete, fiindcă ele implică doar numărarea tranzițiilor dintre stări și sunt independente de forma aleasă pentru probabilitățile de emisie.

Relațiile de actualizare pentru parametrii μ_k și σ_k ai distribuțiilor gaussiene care modelează emisiile sunt următoarele:

$$\begin{aligned}
\mu_k &= \frac{\sum_i 1_{\{z_i=k\}} x_i}{\sum_i 1_{\{z_i=k\}}} \\
\sigma_k &= \frac{\sum_i 1_{\{z_i=k\}} (x_i - \mu_k)^2}{\sum_i 1_{\{z_i=k\}}}
\end{aligned}$$

pentru $k = 1, \dots, K$.⁶³³ (Vă reamintim că notația $1_{\{\cdot\}}$ desemnează funcția indicator.)

d. Media log-verosimilității datelor complete (X și Z) este o margine inferioară pentru funcția de log-verosimilitate a datelor observabile.⁶³⁴ Algoritmul EM / Forward-Backward converge la un optim local al funcției de log-verosimilitate a datelor observabile, așadar este rezonabil să folosim ca funcție obiectiv [la fiecare iterație a acestui algoritm] media log-verosimilității datelor complete.

$$E[\ell_c(\theta; x, y)] = \sum_n E[y_{n,1}^i] \pi_i + \sum_n \sum_{t=2}^T E[y_{n,t-1}^i y_{n,t}^j] \ln a_{ij} + \sum_n \sum_{t=1}^T E[y_{n,t}^i] \ln \mathcal{N}(x_n | \mu_i, \sigma_i),$$

unde n indexează stările modelului Markov ascuns,⁶³⁵ iar $y_{n,t}^i$ este o variabilă aleatoare care ia valoarea 1 dacă la momentul de timp t modelul se află în starea cu indicele i și respectiv valoarea 0 în cazul contrar.⁶³⁶

e. Da, formulele pentru estimările de verosimilitate maximă pentru parametrii μ_k și σ_k au aceeași formă în ambele cazuri. În expresia funcției $E[\ell_c(\theta; x, y)]$ termenul relevant este (doar) ultimul termen, care seamănă foarte mult cu cel din expresia mediei log-verosimilității datelor complete de la mixtura de distribuții gaussiene.⁶³⁷

⁶³²Pentru formulele de calcul ale probabilităților Forward și respectiv Backward în cazul emisiilor discrete, vedeti problemele 18.a și 8.a, relațiile (228) și (227).

⁶³³Vedeti problemele 45.a și respectiv 46.a de la capitolul de *Fundamente*.

⁶³⁴Vedeti problema 1 de la capitolul *Algoritmul EM*.

⁶³⁵Nu are nicio legătură cu indicele n din notația secvenței $X = (x_1, \dots, x_n)$.

⁶³⁶Pentru fiecare valoare fixată pentru $t \in \{1, \dots, T\}$, numai una dintre variabilele $y_{n,t}^i$ ia valoare diferită de 0.

⁶³⁷Vedeti problema 18 de la capitolul *Clusterizare*.

9.2 Probleme propuse

14.

(Modelul Markov – model probabilistic total)

*prelucrare de Liviu Ciortuz, după
Problems and solutions in biological sequence analysis,
M. Borodovsky, S. Ekișeva, 2006, pr. 3.1-3.3*

Fie μ un model Markov de ordinul întâi.

a. Suma probabilităților tuturor secvențelor de stări de lungime L se poate scrie ca

$$\sum_x P(x) = \sum_{x_1} \sum_{x_2} \dots \sum_{x_L} P(x_1) \Pi_{i=2}^L a_{x_{i-1} x_i}$$

Arătați că această sumă este 1.

b. Presupunem că acest model are o stare finală ‘silent’ notată End , și că tranzitările din orice stare în starea End au aceeași probabilitate, τ . Arătați că suma probabilităților tuturor secvențelor de lungime L (și care se termină prin tranzitie în starea End) este $\tau(1 - \tau)^{L-1}$.

c. Arătați că suma probabilităților tuturor secvențelor posibile (de orice lungime) este 1. Aceasta dovedește că un model/lanț Markov este într-adevăr o distribuție de probabilitate peste spațiul tuturor secvențelor definite peste alfabetul respectiv.

15.

(Verificarea înțelegerii unor noțiuni de bază)

CMU, 2003 fall, T. Mitchell, A. Moore, HW7, pr. 2.a

Fie un model Markov ascuns cu două simboluri observate: X și Y . Se consideră următoarea probabilitate (cu notațiile tradiționale pentru HMM):

$$P(q_{t+2} = s_i \wedge q_{t+3} = s_j \wedge O_{t+3} = X \mid q_t = s_k \wedge q_{t+1} = s_m \wedge O_t = X)$$

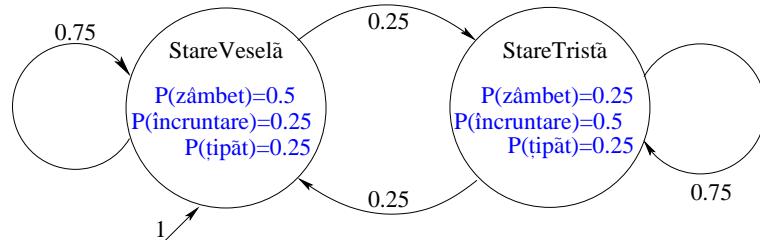
Calculați valoarea acestei probabilități, în funcție de elementele corespunzătoare din matricea de tranzitie A și din matricea de emisie B .

16.

(Verificarea înțelegerii unor noțiuni de bază)

CMU, 2003 fall, T. Mitchell, A. Moore, HW7, pr. 2.b

Vulpea Foxy duce o viață simplă. În unele zile e veselă iar în alte zile e tristă. Dar își ascunde starea emoțională, și poți să observi doar dacă zâmbește, se încruntă sau țipă. Începem în prima zi cu starea veselă și continuăm cu o tranzitie în fiecare zi. Vom reprezenta evoluția stărilor emoționale a lui Foxy cu ajutorul unui model Markov ascuns, și anume:



Dacă notăm prin q_t starea din ziua t , iar prin O_t observația din ziua t , realizați următoarele cerințe:

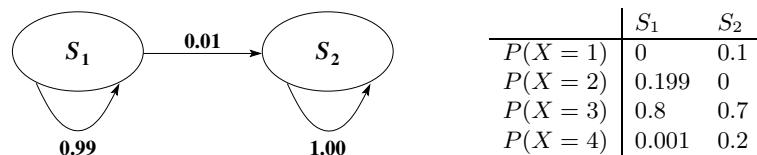
- Calculați $P(q_2 = \text{Tristă})$.
- Calculați $P(O_2 = \text{zâmbet})$.
- Calculați $P(q_2 = \text{Tristă} | O_2 = \text{zâmbet})$.
- Cât este $P(O_2 = \text{tipăt})$.
- Notăm $\phi_t = P(q_t = \text{Veselă})$. Este evident că $\phi_1 = 1$. Se poate da o definiție inductivă a lui ϕ_{t+1} în funcție de ϕ_t printr-o expresie de forma: $\phi_{t+1} = X + Y \cdot \phi_t$. Calculați valorile numerice ale lui X și Y .
- Se consideră $O_1 = \text{incruntare}$, $O_2 = \text{incruntare}$, $O_3 = \text{incruntare}$, $O_4 = \text{incruntare}$ și $O_5 = \text{incruntare}$. Care este cea mai probabilă secvență de stări? (Nu este necesar să aplicați algoritmul Viterbi.)

17.

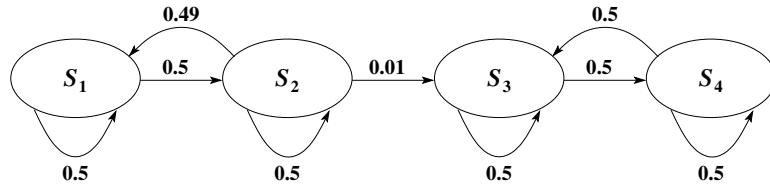
(Verificarea înțelegerii unor noțiuni de bază)

CMU, 2010 fall, Aarti Singh, HW4, pr. 5

Figura de mai jos reprezintă un model Markov ascuns cu două stări. Probabilitățile de tranziție sunt înscrise pe diagramă. Probabilitățile de emisie corespunzătoare fiecărei stări sunt definite pe alfabetul de simboli $\{1, 2, 3, 4\}$ și apar în tabelul de lângă diagramă. Se consideră că probabilitățile de intrare sunt aceleași pentru ambele stări.



- Dați un exemplu de secvență de ieșire de lungime 2 care nu poate fi generată de modelul Markov dat. Justificați răspunsul.
- Am generat o secvență de 10701^{2010} de simboli cu acest HMM și am observat că ultimul simbol din secvență a fost 3. Care este starea cea mai probabilă în care a fost generat acest ultim simbol?
- Considerăm secvență de semnale $\{3, 3\}$. Care este cea mai probabilă secvență de stări corespunzătoare acestei secvențe de semnale? Faceți calculele.
- Acum vom considera secvență de semnale $\{3, 3, 4\}$. Care sunt primele două stări ale celei mai probabile secvențe de stări corespunzătoare secvenței de semnale considerate? Faceți calculele.
- Putem încerca să extindem întrucâtva capacitatea de modelare a acestui HMM prin „duplicarea” fiecăreia din cele două stări. Urmând această idee, am creat diagrama de mai jos. Putem stabili probabilitățile de intrare și cele de emisie în aşa fel încât acest model cu 4 stări, având probabilitățile de tranziție indicate în diagramă, să fie echivalent cu modelul original cu 2 stări (adică, pentru orice secvență de semnale O , $P(O | \text{HMM}_{2\text{states}}) = P(O | \text{HMM}_{4\text{states}})$)? Dacă da, cum anume? Dacă nu, justificați.



18.

(Algoritmul Forward; demonstrație – pasul inductiv)

*Liviu Ciortuz, 2020*Fie un model Markov ascuns μ .

- a. La pasul de inițializare al algoritmului Forward se fixează $\alpha_i(1) = \pi_i$ pentru $i = 1, \dots, N$, unde N este numărul de stări din modelul Markov ascuns μ , iar π_i este probabilitatea de intrare în starea i . Formula de recurență de la pasul inductiv al algoritmului Forward pentru HMM este următoarea:

$$\alpha_j(t+1) = \sum_{i=1}^N \alpha_i(t) a_{ij} b_{ij} O_t, \text{ pentru } j = 1, \dots, N \text{ și } t = 1, \dots, T. \quad (228)$$

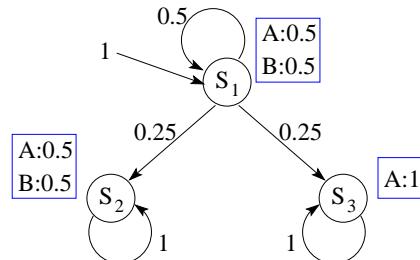
Faceți demonstrația acestei formule de recurență. Pentru fiecare egalitate din cadrul demonstrației, precizați motivul/motivele pentru care are loc egalitatea respectivă.

- b. Cum se poate calcula $P(O_1 O_2 \dots O_T | \mu)$, probabilitatea de emisie a unei secvențe de semnale $O_1 O_2 \dots O_T$ de către modelul μ , folosind probabilitățile Forward (α)?

19.

(HMM, calculul probabilității secvențelor emise:
o formulă alternativă; demonstrație și aplicare)
CMU, 2009 spring, Ziv Bar-Joseph, final, pr. 5

Pentru calcularea probabilității unei secvențe de observații, $P(O_1, \dots, O_T)$, suntem obișnuiți cu folosirea probabilităților Forward sau a probabilităților Backward (vedeți problemele 18.b și respectiv 8.b). În această problemă veți folosi alte probabilități pentru a determina o formulă de calcul pentru $P(O_1, \dots, O_T)$. Apoi veți aplica noua formulă pentru secvențe de observații corespunzătoare următorului model Markov ascuns:



- a. Fie următoarele notății:

$$\nu_i^t = P(O_1, \dots, O_T | q_t = s_i) \text{ și } p_t(i) = P(q_t = s_i).$$

- Determinați o formulă pentru $P(O_1, \dots, O_T)$ care să folosească doar ν_i^t și $p_t(i)$.
- b. Folosind formula găsită la punctul precedent, calculați $P(O_1 = B, \dots, O_{200} = B)$ pentru modelul dat, adică probabilitatea observării a 200 de emisii B consecutive. Pentru aceasta trebuie să găsiți un moment t convenabil, să calculați ν_i^t și $p_t(i)$ corespunzători și apoi să folosiți aceste valori pentru a calcula probabilitatea cerută. (*Indicație:* pentru calcularea $p_t(i)$ se observă că tranzițiile din și spre stările S_2 și S_3 sunt simetrice pentru orice t , deci $p_t(S_2) = p_t(S_3)$.)
- c. Folosind formula găsită la punctul a, calculați $P(O_1 = A, \dots, O_{200} = A)$ pentru modelul dat, adică probabilitatea observării a 200 de emisii A consecutive. Trebuie să găsiți din nou un moment t convenabil, dar de data aceasta este permisă calcularea în funcție de ν_i^t , pentru care nu trebuie obținută valoarea numerică. Toate celelalte probabilități implicate în formulă trebuie însă calculate. (*Indicație:* momentul t ales la acest punct poate fi diferit de cel ales la punctul b.)

20. (HMM: o variație pe tema algoritmului Viterbi)
*York University, UK, 2001 spring,
 Univ. of York (UK), J. Cussens, D. Kudenko, open exam., pr. 1.1*

Trebuie să călătorescă între două puncte, A și B. Aceste puncte sunt despărțite de n râuri. Pe fiecare din cele n râuri sunt construite m poduri. Poți trece peste un râu doar dacă folosești unul din cele m poduri de pe râul respectiv. Considerăm că se cunosc distanțele dintre orice două poduri care sunt situate pe râuri vecine.

- a. Definește un algoritm eficient care găsește cel mai scurt drum dintre A și B și returnează (și) lungimea acestui drum. *Observație:* Nu trebuie să implementezi acest algoritm, ci doar să descrii în pseudo-cod cum lucrează.
 b. Comparaș algoritmul rezultat la punctul a cu algoritmul Viterbi.

21. (Algoritmul Forward-Backward: implementare)
Liviu Ciortuz

Pentru un model Markov ascuns se consideră date (ca variabile globale):

- N (numărul de stări) și o secvență O de lungime T peste un alfabet de K simboli;
- vectorul de probabilități de intrare $\pi[N]$, matricea de probabilități de tranziție $a[N,N]$ și matricea probabilități de emisie $b[N,N,K]$;
- matricele $\alpha[N,T+1]$ și $\beta[N,T+1]$ care se presupun a fi calculate de algoritmii Forward și respectiv Backward;
- matricea $p[T+1][N][N]$ pentru claculul mediilor variabilelor neobservabile (repräsentând, fiecare, tranziția de la starea i la starea j la momentul t).

Definiți în limbajul Python / C două funcții, **expectation()** și **maximization()**, care servesc la implementarea algoritmului EM/Forward-Backward pentru HMM. Funcția **expectation** va calcula matricea $p[T+1,N,N]$, iar funcția **maximization** va înlocui π , a și b cu noile valori calculate de algoritmul EM.

Vă reamintim algoritmul EM/Forward-Backward:

• Estimare:

Dată o secvență (observație) O , se definește probabilitatea traversării unui anumit arc la momentul de timp t :

$$\begin{aligned} p_t(i, j) &\stackrel{\text{def.}}{=} P(X_t = i, X_{t+1} = j | O, \mu) \\ &= \frac{P(X_t = i, X_{t+1} = j, O | \mu)}{P(O | \mu)} = \frac{\alpha_i(t) a_{ij} b_{ij O_t} \beta_j(t+1)}{\sum_{m=1}^N \alpha_m(t) \beta_m(t)} \\ &= \frac{\alpha_i(t) a_{ij} b_{ij O_t} \beta_j(t+1)}{\sum_{m=1}^N \sum_{n=1}^N \alpha_m(t) a_{mn} b_{mn O_t} \beta_n(t+1)} \end{aligned}$$

Sumând după t , obținem:

$\sum_{t=1}^T p_t(i, j)$ = media (valoarea așteptată) a numărului de tranziții de la s_i la s_j în timpul emisiei lui O ;

$\sum_{j=1}^N \sum_{t=1}^T p_t(i, j)$ = media (valoarea așteptată) a numărului de tranziții din s_i în timpul emisiei lui O .

• Maximizare:

Folosind $\mu = (A, B, \Pi)$, se calculează $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$:

$$\begin{aligned} \hat{\pi}_i &= \frac{\sum_{j=1}^N p_1(i, j)}{\sum_{l=1}^N \sum_{j=1}^N p_1(l, j)} = \sum_{j=1}^N p_1(i, j) = \gamma_i(1) \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{l=1}^N \sum_{t=1}^T p_t(i, l)} \\ \hat{b}_{ijk} &= \frac{\sum_{t: O_t=k, 1 \leq t \leq T} p_t(i, j)}{\sum_{t=1}^T p_t(i, j)}. \end{aligned}$$

22.

(Un HMM cu emisii gaussiene: varianta normalizată pentru probabilitățile Forward, Backward și pentru algoritmul EM/Forward-Backward; implementare și aplicare)

CMU, 2009 spring, Ziv Bar-Joseph, HW4, pr. 3

Robert merge să înoteze zilnic. El folosește fie culoarul 1, fie culoarul 2 din piscină. Dacă piscina este aglomerată, atunci el alege să înoteze pe culoarul 2, în caz contrar alege culoarul 1. Își notează întotdeauna timpul dus-întors de înnot. Descoperă astfel că performanța sa pe culoarul 2 corespunde unei distribuții normale în jurul a 60 de secunde cu deviația standard 1, iar pe culoarul 1 unei distribuții normale în jurul a 63 de secunde cu deviația standard 1. Robert observă de asemenea un tipar al utilizării piscinei: dacă într-o zi este aglomerat, atunci probabilitatea ca în ziua următoare să fie de asemenea aglomerat este de 0.3, iar dacă nu este aglomerat, atunci probabilitatea ca în ziua următoare să fie aglomerat este de 0.6.

Deși își notează zilnic timpul efectuat, Robert nu își notează pe ce culoar înnotă. Astfel în însemnările sale se găsesc următoarele date:

Ziua (n)	Timpul (x) în secunde
1	60.0
2	62.0
3	59.0
4	63.0
5	62.0
6	64.0
7	61.0

a. Desenați un *model grafic* care să descrie cât mai bine măsurătorile făcute de Robert (notate sub forma x_n) precum și variabilele ascunse (notate cu z_n).

b. Calculați distribuția condiționată a variabilelor z_n în raport cu x_1, \dots, x_n . Vă cerem să elaborați calculul/demonstrația în întregime.

Indicație: Ați putea formula $\alpha(z_n) = P(z_n | x_1, \dots, x_n)$ în funcție de $\alpha(z_{n-1})$, într-o formă parametrizată.

c. Derivați ecuația de recursie înapoi (engl., backward equation) pentru probabilitatea a posteriori $P(z_n | x_1, \dots, x_N)$, notată cu $\beta(z_n)$, unde n ia valori în $\{1, \dots, N\}$. Faceți calculul/demonstrația în întregime.

Indicație: Ați putea formula $\beta(z_n)$ în funcție de $\beta(z_{n+1})$ și $\alpha(z_n)$, într-o formă parametrizată.

d. Cât este media condiționată pentru tranzitie z_n, z_{n+1} dată fiind secvența de emisii x_1, \dots, x_N , adică $E(z_n z_{n+1}^T | x_1, \dots, x_N)$? Faceți calculul / demonstrația în întregime.

Indicație: Tineți cont că aveți variabile ascunse discrete (de tip categorial) pentru stări și variabile continue (de tip gaussian) pentru observații. Așadar, $E(z_n z_{n+1}^T | x_1, \dots, x_N)$ va fi o matrice 2×2 în acest caz.

Pentru punctele următoare ($e - h$), este necesar ca în prealabil să faceți implementarea formulelor pe care le-ați determinat la punctele $b - d$.

e. Folosind datele din tabelul de mai sus, stabiliți dacă piscina este aglomerată sau nu în ziua a patra.

f. Robert nu este sigur în legătură cu estimările sale pentru culoarele 1 și 2. Ați putea să faceți o re-estimare? În mod concret, puteți presupune că performanțele lui pe culoarele 1 și 2 corespund distribuțiilor normale $\mathcal{N}(\mu_1, \sigma_1^2)$ și respectiv $\mathcal{N}(\mu_2, \sigma_2^2)$. Puteți estima parametrii $\mu_1, \sigma_1, \mu_2, \sigma_2$?

Sugestie: Veți putea apela la un algoritm de tip EM care folosește probabilitățile $\alpha(z_n)$ și $\beta(z_n)$ determinante anterioare. Veți considera că probabilitățile de tranzitie, precum și valorile inițiale ale parametrilor $\mu_1, \sigma_1, \mu_2, \sigma_2$ nu se re-estimează aici.

g. Pe de altă parte, Robert nu este sigur în legătură cu pattern-ul zilelor aglomerate de la piscină, adică nu știe dacă probabilitățile de tranzitie au fost corect estimate ca fiind 0.3 și 0.6. Ați putea să faceți o re-estimare a acestor probabilități folosind observațiile de mai sus?

Sugestie: Veți putea apela la un [alt] algoritm de tip EM care folosește probabilitățile $\alpha(z_n)$ și $\beta(z_n)$ determinante anterioare. Veți considera că valorile inițiale ale probabilităților

de tranziție nu se re-estimează aici. Veți folosi ca valori pentru parametrii celor două distribuții gaussiene mediile $\mu_1 = 60, \mu_2 = 63$ și varianțele $\sigma_1^2 = \sigma_2^2 = 1$.

h. Să presupunem că Robert nu este sigur nici în legătură cu parametrii $\mu_1, \sigma_1, \mu_2, \sigma_2$ și nici cu parametrii zilelor aglomerate de la piscină. Ați putea să faceți o re-estimare a tuturor acestor parametri folosind observațiile de mai sus?

Vă solicităm să implementați algoritmul dumneavoastră de învățare automată în limbajul de programare preferat. Indicați rezultatele obținute în urma estimării parametrilor folosind datele din tabel.

„Mai bun este sfârșitul unui lucru decât începutul lui...“

Cartea Eclesiastului 7:8.a

Alcătuirea acestei culegeri a fost pentru noi, autorii ei, asemenea unui drum, pe care l-am parcurs sub îndrumarea unor profesori și asistenți de departe. La capătul acestui drum, ne dăm seama că lungimea drumului și greutățile prin care am trecut n-au fost determinante, ci cel mai mult au contat „peisajele“ care se desfășoară de o parte și de alta a drumului.

Dragă „cititorule“, sperăm că după ce vei fi parcurs, la rândul tău, alături de noi o parte din acest drum (care pe noi ne-a încântat!), și-ai devenit și tu călăuze, dar despărțindu-ne aici, îți vei continua propriul tău drum, fie aplicând cele arătate aici, fie parcurgând alte capitole. Sau, pur și simplu, folosind abilitățile pe care le-ai dobândit *antrenându-te* cu astfel de exerciții, vei putea să le *generalizezi* în propriile tale domenii de interes. Îți dorim sincer *Mult succes!*

Autorii



© M. Romanică

Această pagină a fost lăsată liberă în mod intenționat.