Exerciţii de învăţare automată

Liviu Ciortuz, Alina Munteanu, Elena Bădărău
2019
ISBN: 978-973-0-30467-1

1. Fundamente: probabilităţi, variabile aleatoare, teoria informaţiei, funcţii-nucleu, şi metode de optimizare

Sumar

Evenimente aleatoare şi formula lui Bayes

– *funcţia de probabilitate* – câteva proprietăţi care derivă din definiţia ei: ex. 1, ex. 64;

– calcularea unor *probabilităţi* elementare: ex. 2.a, ex. 61;

– calcularea unor *probabilităţi condiţionate*: ex. 2.b, ex. 3, ex. 62.a;

– *regula de multiplicare*: ex. 64.b;

– *formula probabilităţii totale*: ex. 64.e;
formula probabilităţii totale – varianta condiţională: ex. 67.cd;

– *independenţa* evenimentelor aleatoare: ex. 4, ex. 5, ex. 62.bc;

– *independenţa condiţională* a evenimentelor aleatoare – legătura dintre forma „tare“ şi forma „slabă“ a definiţiei pentru această noţiune: ex. 63;

– *formula lui Bayes* – aplicare: ex. 6, ex. 7, ex. 65, ex. 66;
formula lui Bayes – varianta condiţională: ex. 67.b;

– recapitulare: probabilităţi elementare şi probabilităţi condiţionate – câteva proprietăţi (A/F): ex. 8, ex. 67.

Variabile aleatoare

– funcţie de *distribuţie cumulativă* (engl., cumulative distribution function, c.d.f.), un exemplu: ex. 27;

– proprietatea de *liniaritate* a *mediei*: ex. 9.a;

– *varianţa* şi *covarianţa*: proprietăţi de tip *caracterizare*: ex. 9.bc;

– covarianţa oricăror două variabile aleatoare independente este 0: ex. 10;
reciproca acestei afirmaţii nu este adevărată în general: ex. 11.a; totuşi ea are loc dacă variabilele sunt de tip binar (adică iau doar valorile 0 şi 1): ex. 11.b, ex. 71;

– o *condiţie suficientă* pentru ca $Var[X + Y] = Var[X] + Var[Y]$: independenţa variabilelor X şi Y: ex. 19.c

• pentru *variabile aleatoare discrete*:

– calcularea mediilor şi a varianţelor – exemplificare: ex. 12, ex. 68, ex. 70.a;

– definirea unei *variabile-indicator* cu ajutorul unui eveniment aleator; calcularea mediei acestei variabile: ex. 69;

– regula de *înlănţuire* (pt. var. aleat.) – aplicare: ex. 13;

– regula de *multiplicare* (pt. var. aleat.), varianta condiţională – demonstraţie: ex. 14;

– *independenţă*: ex. 70.b, ex. 72.a, ex. 73.a;
independenţă condiţională: ex. 15, ex. 72.b, ex. 73.b, ex. 74.b;
o condiţie suficientă pentru independenţa condiţională: ex. 75;

• pentru *variabile aleatoare continue*:

– dată fiind o funcţie care depinde de un parametru real, să se calculeze valoarea respectivului parametru astfel încât funcţia respectivă să fie o *funcţie de densitate de probabilitate* (engl., probability density function, p.d.f.): ex. 16.a, ex. 17, ex. 76;

– dat fiind un p.d.f., să se calculeze o anumită probabilitate: ex. 16.c, ex. 77;

– variabile aleatoare discrete vs. variabile aleatoare continue; p.m.f. vs. p.d.f.: ex. 78;

– variabile aleatoare discrete şi variabile aleatoare continue; independenţa şi calcul de medii: ex. 79;

• *vector de variabile aleatoare*:
o proprietate: matricea de covarianţă este simetrică şi pozitiv definită: ex. 18;
calculul matricei de covarianţă când asupra vectorului de variabile aleatoare operăm transformări liniare: ex. 80;

• recapitulare: ex. 19, ex. 81.

Distribuţii probabiliste uzuale

• distribuţii discrete

– distribuţia *Bernoulli*:
suma de variabile identic distribuite; media sumei: ex. 20;
mixtură de distribuţii Bernoulli: ex. 85;

– distribuţia *binomială*:
verificarea condiţiilor de definiţie pentru p.m.f.: ex. 21.a;
calculul mediei şi al varianţei: ex. 21.bc;
calcularea unor probabilităţi (simple şi respectiv condiţionale): ex. 82;

– distribuţia *categorială*:
calcularea unor probabilităţi şi a unor medii: ex. 83;
mixtură de distribuţii categoriale: calculul mediei şi al varianţei: ex. 83;

– distribuţia *geometrică*:
calcularea numărului „aşteptat“ / mediu de „observaţii“ necesare pentru ca un anumit eveniment să se producă: ex. 84;

– distribuţia *Poisson*:
verificarea condiţiilor de definiţie pentru p.m.f., calculul mediei şi al varianţei: ex. 22;

• distribuţii continue

– **distribuţia *continuă uniformă*:**
exemplu de distribuţie continuă uniformă uni-variată; calcularea mediei şi a varianţei: ex. 87;
calculul unei p.d.f. corelate, pornind de la două variabile [urmând distribuţii continue uniforme uni-variate] independente; calculul unei anumite probabilităţi, folosind p.d.f. corelată: ex. 24;
exemplu de distribuţie continuă uniformă bivariată; calcularea unei p.d.f. condiţionale; verificarea independenţei celor două distribuţii marginale; calculul mediilor unor p.d.f.-uri condiţionale: ex. 86, ex. 79;

– **distribuţia *exponenţială*:**
verificarea condiţiilor de definiţie pentru p.d.f.,
calculul mediei şi al varianţei: ex. 25.a;

– **distribuţia *Gamma*:**
verificarea condiţiilor de definiţie pentru p.d.f., calculul mediei şi al varianţei: ex. 25.b;

– **distribuţia *gaussiană uni-variată*:**
verificarea condiţiilor de definiţie pentru p.d.f., calculul mediei şi al varianţei: ex. 26;
„standardizare“ (i.e., reducerea cazului nestandard la cazul standard): ex. 27;

– **distribuţia *gaussiană multi-variată*:**
matrice simetrice şi pozitiv definite[1] o proprietate de tip *factorizare* folosind *vectori proprii*: ex. 29;
p.d..f.-ul distribuţiei gaussiane multi-variate este într-adevăr p.d.f.:[2] ex. 30;
o *proprietate* importantă, în cazul în care matricea de covarianţă este diagonală: p.d.f.-ul corelat este produsul p.d.f.-urilor marginale (care sunt independente): ex. 28;
distribuţia *gaussiană bivariată*: exemplificare; calcularea explicită a p.d.f.-ului, dat fiind vectorul de medii şi matricea de covarianţă: ex. 88;
o *proprietate* pentru distribuţia *gaussiană bivariată*: distribuţia condiţională a unei componente în raport cu cealaltă componentă este tot de tip gaussian; calculul parametrilor acestei distribuţii condiţionale: ex. 31;

– ***mixturi* de distribuţii *gaussiane multi-variate*:**
exprimarea vectorului de medii şi a matricei de covarianţă în funcţie de mediile şi matricele de covarianţă ale distribuţiilor componente: ex. 89;

– ***mixturi* de distribuţii oarecare:**
calculul mediilor şi al varianţelor (în funcţie de mediile şi matricele de covarianţă ale distribuţiilor componente): ex. 90;

– **distribuţia Bernoulli şi distribuţia normală standard:**
***intervale de încredere*, *teorema limită centrală*; aplicaţie la calculul erorii reale a unui clasificator: ex. 32;**

• **chestiuni recapitulative (corespondenţa dintre nume de distribuţii şi expresiile unor p.d.f.-uri date): ex. 91.**

[1] Aşa sunt matricele de covarianţă ale variabilelor gaussiene multi-variate.
[2] Adică satisface condiţiile din definiţia noţiunii de p.d.f.

Elemente de teoria informaţiei

– re-descoperirea definiţiei entropiei, pornind de la un set de proprietăţi dezirabile: ex. 33;

– *definiţii* şi *proprietăţi imediate* pentru entropie, entropie corelată, entropie condiţională specifică, entropie condiţională medie, câştig de informaţie: ex. 34;
exemplificarea acestor noţiuni (varianta discretă): ex. 35, ex. 36, ex. 92;
un *exemplu* de calcul pentru entropia unei variabile continue (distribuţia exponenţială): ex. 37;
o *proprietate* a entropiei: ne-negativitatea: ex. 34.a, ex. 98.a;
o margine superioară pentru valorea entropiei unei variabile aleatoare discrete: ex. 93;
o *proprietate* a câştigului de informaţie: ne-negativitatea: ex. 38.c, ex. 96;

– o *aplicaţie* pentru câştigul de informaţie: *selecţia de trăsături*: ex. 39;

– *entropia corelată*: forma particulară a relaţiei de „înlănţuire“ în cazul variabilelor aleatoare independente: ex. 40, ex. 94, ex. 98.c;
Entropie corelată şi condiţionată: formula de „înlănţuire“ condiţională: ex. 95;

– *entropia relativă*: definiţie şi proprietăţi elementare; exprimarea câştigului de informaţie cu ajutorul entropiei relative: ex. 38;

– *cross-entropie*: definiţie, o proprietate (ne-negativitatea) şi un exemplu simplu de calculare a valorii cross-entropiei: ex. 41;
un exemplu de *aplicaţie* pentru cross-entropie: selecţia modelelor probabiliste: ex. 97;

– *inegalitatea lui Gibbs*: un caz particular; comparaţie între valorile entropiei şi ale cross-entropiei: ex. 42.

Funcţii-nucleu

– aflarea funcţiei de „mapare“ a trăsăturilor care corespunde unei funcţii-nucleu date: ex. 43, ex. 99, ex. 100.a;
comparaţii asupra numărului de operaţii efectuate la calcularea valorii unor funcţii-nucleu (în spaţiul iniţial vs. spaţiul nou de „trăsături“): ex. 100.b;
calculul distanţelor euclidiene în spaţii de „trăsături“ folosind doar funcţii-nucleu: ex. 47;

- *teorema lui Mercer* (1909): condiţii necesare şi suficiente pentru ca o funcţie să fie funcţie-nucleu: ex. 44.ab;

- rezultate de tip „constructiv“ pentru [obţinerea de noi] funcţii-nucleu: ex. 44.c, 45, ex. 46, ex. 101, 103, ex. 52.b;
„normalizarea“ funcţiilor-nucleu: ex. 102;

– o inegalitate [derivată din inegalitatea Cauchy-Buniakovski-Schwarz], care furnizează o margine superioară pentru $K(x, x')$, valoarea absolută a unei funcţii nucleu oarecare: ex. 104;

– un exemplu de funcţie-nucleu care serveşte la a măsura similaritatea dintre două imagini oarecare: ex. 48;

– o funcţie-nucleu particulară, care asigură separabilitate liniară [în spaţiul de trăsături] pentru orice set de instanţe de antrenament: ex. 105;

– **funcţia-nucleu gaussiană / *funcţia cu baza radială* (engl., Radial Basis Function, RBF):**
  • **demonstraţia faptului că RBF este într-adevăr funcţie-nucleu: ex. 49;**
  ◦ **funcţia de „mapare“ corespunzătoare funcţiei-nucleu RBF ia valori într-un spaţiu [de „trăsături“] de dimensiune infinită: ex. 50;**
  ◦ **proprietăţi simple ale nucleului RBF: ex. 51, ex. 106;**
  • **orice mulţime de instanţe distincte, având orice etichetare posibilă, este separabilă liniar în spaţiul de „trăsături“ dacă se foloseşte nucleul RBF cu parametrul ales în mod convenabil: ex. 26.a de la capitolul *Maşini cu vectori-suport*;**

– **recapitulare (A/F): ex. 52, ex. 108.**

Metode de optimizare în învăţarea automată

– **(P0) definiţii, caracterizări şi câteva proprietăţi pentru funcţii convexe: ex. 53;**

– ***metoda gradientului*, *metoda lui Newton*; exemplificare: ex. 54;**

  (P1) condiţii suficiente pentru convergenţa metodei gradientului: ex. 110;
  (P2) o proprietate interesantă a metodei lui Newton: în cazul oricărei funcţii de gradul al doilea (de una sau mai multe variabile), aplicarea acestei metode de optimizare implică / necesită execuţia unei singure iteraţii: ex. 111;
  (P3) *reparametrizarea liniară* a atributelor nu afectează [rezultatele obţinute cu] metoda lui Newton, însă afectează metoda gradientului: ex. 112;

– **metoda dualităţii Lagrange:**

  (P4) demonstrarea proprietăţii de *dualitate slabă*: ex. 55;
  (P5) demonstrarea unei părţi din *teorema KKT*: ex. 56;

  exemple de aplicare: ex. 57, ex. 58, ex. 113;
  un exemplu de problemă de optimizare convexă pentru care condiţiile KKT nu sunt satisfăcute: ex. 114;

– **două variante a algoritmului Perceptron,[3] pentru care relaţia de actualizare a ponderilor se obţine rezolvând [câte] o problemă de optimizare [convexă] cu restricţii;**

– **chestiuni recapitulative: ex. 109.**

[3] Vedeţi problema 16 de la capitolul *Reţele neuronale artificiale*.

2. Metode de estimare a parametrilor; metode de regresie

Sumar

Noţiuni preliminare

- elemente de calcul vectorial (în particular, produsul scalar) şi de calcul matriceal: ex. 29 de la cap. *Fundamente*;
  norma L_2 (euclidiană) şi norma L_1: ex. 14, ex. 47, ex. 49;
  calculul derivatelor parţiale [de ordinul întâi şi al doilea]: ex. 17;
  reguli de derivare cu argumente vectoriale: ex. 12;
- metode de optimizare (în speţă pentru aflarea maximului / minimului unei funcţii reale, derivabile): metoda analitică, metoda gradientului, metoda lui Newton; exemplificare: ex. 54 de la capitolul de Fundamente;

Estimarea parametrilor unor distribuţii probabiliste uzuale

- distribuţia Bernoulli: ex. 1 (+MAP, folosind distr. Beta), ex. 2, ex. 30 (un caz particular), ex. 29 (bias-ul şi varianţa estimatorului MLE);
- distribuţia categorială: ex. 31, ex. 32 (+MAP, folosind distr. Dirichlet);
- distribuţia geometrică: ex. 33 (+MAP, folosind distr. Beta);
- distribuţia Poisson: ex. 3 (+MAP, folosind distr. Gamma);
- distribuţia uniformă continuă: calcul de probabilităţi şi MLE:
  în $\mathbb{R}$: ex. 4, ex. 34, ex. 35; în $\mathbb{R}^2$: ex. 5, ex. 36;
- distribuţia gaussiană uni-variată:
  MLE pt. μ, considerând σ^2 cunoscut: ex. 2 (+MAP, folosind distribuţia gaussiană), ex. 38 (+analiza discriminativă);
  MLE pt. σ^2, atunci când nu se impun restricţii asupra lui μ: ex. 8 (+deplasare);
  MLE pt. σ^2, atunci când $\mu = 0$: ex. 39 (+nedeplasare);
- distribuţia gaussiană multi-variată: ex. 10, ex. 40 (+MAP, folosind distr. Gauss-Wishart);
- distribuţia exponenţială: ex. 6, ex. 37 (+MAP, folosind distr. Gamma);
- distribuţia Gamma: ex. 9 şi ex. 41 (ultimul, folosind metoda gradientului şi metoda lui Newton).

- existenţa şi unicitatea MLE: ex. 11.

Regresia liniară

- prezentarea *generală* a metodei regresiei liniare:[4]

  – MLE şi corespondenţa cu estimarea în sens LSE (least squared errors): ex. 14.A;
  particularizare pentru cazul uni-variat: ex. 12.ab, ex. 42;
  exemplificare pentru cazul uni-variat (ex. 13, ex. 43) şi pentru cazul bivariat (ex. 45.a, ex. 53);

[4] În mod implicit, în această secţiune se va considera că termenul-zgomot este modelat cu distribuţia gaussiană (dacă nu se specifică altfel, în mod explicit).

– (P1) *scalarea atributelor* nu schimbă predicţiile obţinute (pentru instanţele de test) cu ajutorul *formulelor analitice*: ex. 15, 59.a;

– (P2) adăugarea de noi trăsături / atribute nu măreşte suma pătratelor erorilor: ex. 48;

– o *proprietate surprinzătoare* a regresiei liniare: adăugarea câtorva „observaţii“ 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. 17;

– MAP şi corespondenţa cu *regularizarea* de normă L_2 (regresia *ridge*): ex. 14.C; particularizare pentru cazul uni-variat: ex. 12.c;

– *regularizarea* de normă L_1 (regresia *Lasso*): ex. 47.a;

– (P3) efectul de diminuare a ponderilor (engl., weight decay) în cazul regularizării de normă L_2 (respectiv L_1) a regresiei liniare, în comparaţie cu cazul neregularizat: ex 47.b;

- bias-ul şi [co]varianţa estimatorului regresiei liniare; bias-ul regresiei *ridge*: ex. 16;
- regresia polinomială [LC: mai general: folosirea aşa-numitelor *funcţii de bază*]: ex. 14.B;
  exemplificare pentru cazul bivariat: CMU, 2015 spring, T. Mitchell, N. Balcan, HW4, pr. 1;

- cazul regresiei liniare cu termen de regularizare L_2 (regresia *ridge*):
  deducerea regulilor de actualizare pentru *medoda gradientului ascendent*:
  varianta "batch" / "steepest descent": ex. 18.a;
  şi varianta stohastică / secvenţială / "online": ex. 18.b; exemplu de aplicare: ex. 46;
- cazul regresiei liniare cu termen de regularizare L_1 (regresia Lasso):
  rezolvare cu *metoda descreşterii pe coordonate* (engl., "coordinate descent"): ex. 49;
  rezolvare cu *metoda sub-gradientului* (aplicare la selecţia de trăsături): CMU, 2009 fall, C. Guestrin, HW2, pr. 2;
- regresia liniară în cazul zgomotului modelat cu distribuţia *Laplace* (în locul zgomotului gaussian): ex. 19.B;
  exemplificare pentru cazul bivariat: ex. 45.c;
  rezolvare în cazul uni-variat [chiar particularizat] cu ajutorul derivatei, acolo unde aceasta există: ex. 50;

- regresia liniară şi *overfitting-ul*: ex. 22;
- regresie liniară folosită pentru *clasificare*: exemplificare: ex. 53;

- cazul *multi-valuat* al regresiei liniare, reducerea la cazul uninomial: ex. 52;
- regresia liniară cu regularizare L_2 (regresia *ridge*), *kernel-izarea* ecuaţiilor „normale“: ex. 20;
  (P4) folosind nucleu RBF, eroarea la antrenare devine 0 atunci când parametrul de regularizare λ tinde la 0: ex. 21;
- *regresia liniară ponderată*:: ex. 19.A;
  particularizare / exemplificare pentru cazul bivariat: ex. 45.b;

o *proprietate* a regresiei liniare local-ponderate [demonstrată în cazul uni-variat]: „netezirea“ liniară: ex. 51; cazul multi-valuat, cu regularizare L_2: Stanford, 2015 fall, Andrew Ng, midterm, pr. 2;

- *regresia liniară (kernelizată) local-ponderată*, ne-parametrică:
  particularizare / exemplificare pentru cazul uni-variat, cu nucleu gaussian: CMU, 2010 fall, Aarti Singh, midterm, pr. 4.

Regresia logistică

- prezentare generală,
  (•) calculul funcţiei de log-verosimilitate, estimarea parametrilor în sens MLE, folosind *metoda gradientului* (i.e., deducerea regulilor de actualizare a parametrilor): ex. 23, 59.b;

  particularizare pentru cazul datelor din $\mathbb{R}^2$: ex. 54 (inclusiv *regularizare* L_1 / estimarea parametrilor în sens MAP, folosind o distribuţie a priori Laplace);
- (P0) *graniţa de decizie* pentru regresia logistică: ex. 54.d;
- (P1) funcţia de log-verosimilitate în cazul regresiei logistice este concavă (deci are un maxim global), fiindcă matricea hessiană este pozitiv definită: ex. 24;

  Observaţie: Demonstraţia furnizează tot ce este necesar pentru obţinerea [ulterioară a] relaţiei de actualizare a parametrilor la aplicarea *metodei lui Newton* în cazul regresiei logistice;
- (P2) analiza efectului duplicării atributelor: ex. 55;
- (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 distribuţie a priori distribuţia gaussiană multi-variată sferică —, în comparaţie cu cazul estimării parametrilor în sensul MLE: ex. 25;
- *Variante* / *extensii* ale regresiei logistice:

  regresia logistică local-ponderată, cu regularizare L_2:
  (•) calcularea vectorului gradient şi a matricei hessiene (necesare pentru aplicarea metodei lui Newton în acest caz): ex. 56;

  regresia logistică kernel-izată:
  (•) adaptarea metodei gradientului: ex. 26;

  regresia logistică n-ară (aşa-numita regresie *softmax*), cu regularizare L_2:
  (•) calculul funcţiei de log-verosimilitate, deducerea regulilor de actualizare a ponderilor, folosind metoda gradientului: ex. 27;
- (P4) o [interesantă] proprietate comună pentru regresia liniară şi regresia logistică: ex. 57;
- întrebări (cu răspuns A/F) cu privire la aplicarea *metodei lui Newton* comparativ cu *metoda gradientului* (în contextul rezolvării problemelor de regresie liniară şi / sau regresie logistică): ex. 59.c;
- comparaţii între regresia logistică şi alţi clasificatori (Bayes Naiv, ID3): ex. 54.c, ex. 58.ab.

3. Arbori de decizie

Sumar

Noţiuni preliminare

- **partiţie a unei mulţimi: ex. 64 de la cap. *Fundamente*;**
- **proprietăţi elementare ale funcţiei logaritm; formule uzuale pentru calcule cu logaritmi;**
- **entropie, definiţie: T. Mitchell, *Machine Learning*, 1997 (desemnată în continuare simplu prin *cartea ML*), pag. 57; ex. 2.a, ex. 37.a;**
- **entropie condiţională specifică: ex. 15.a;**
- **entropie condiţională medie: ex. 2.cd;**
- **câştig de informaţie (definiţie: cartea ML, pag. 58): ex. 2.cd, ex. 5.a, ex. 31, ex. 37.b;**
- ***arbori de decizie*, văzuţi *ca structură de date*: ex. 1, ex. 29**
  şi, respectiv, ca program în logica propoziţiilor: ex. 2.e, ex. 36.bc;
  • **(P0) *expresivitatea arborilor de decizie* cu privire la *funcţii boolene*: ex. 30;**
- ***spaţiu de versiuni* pentru un concept (de învăţat): ex. 1, ex. 29, ex. 35.**

Algoritmul ID3

- **pseudo-cod: cartea ML, pag. 56;**
- ***bias-ul inductiv*: *ibidem*, pag. 63-64;**
- **exemple simple de aplicare: ex. 2, ex. 3, ex. 5, ex. 34, ex. 35, ex. 37, ex. 38;**
- **ID3 ca algoritm *per se*:**
  - **este un *algoritm de căutare*;**
    ***spaţiul de căutare* — mulţimea tuturor arborilor de decizie care se pot construi cu atributele de intrare în nodurile de test şi cu valorile atributului de ieşire în nodurile de decizie — este de dimensiune exponenţială în raport cu numărul de atribute: ex. 1, ex. 3, ex. 29, ex. 35;**
    ID3 are ca *obiectiv* căutarea unui arbore / *model* care *i.* să explice cât mai bine datele (în particular, atunci când datele sunt *consistente*, modelul trebuie să fie *consistent* cu acestea), *ii.* să fie cât mai *compact*, din motive de *eficienţă* la generalizare şi *iii.* în final să aibă o [cât mai] bună putere de *generalizare*;[5]
    - **ID3 ar putea fi văzut şi ca algoritm de *optimizare*;**[6]
  - ***greedy* ⇒ nu garantează obţinerea soluţiei optime d.p.v. al numărului de niveluri / noduri:**
    ex. 4, ex. 22.a, ex. 36 (vs. ex. 35.b, ex. 3.b), ex. 44;

[5] LC: Alternativ, putem spune că algoritmul ID3 produce o *structură* de tip *ierarhie* (arbore) între diferite *partiţionări* ale setului de instanţe de antrenament, această ierarhie fiind generată pe baza *corespondenţei* dintre atributul de *ieşire* şi atributele de *intrare*, care sunt adăugate la model câte unul pe rând.

[6] LC: Am putea să-l interpretăm pe ID3 ca fiind un algoritm care caută între diferitele *distribuţii probabiliste* discrete care pot fi definite pe setul de date de antrenament una care să satisfacă cerinţa de *ierarhizare* (vedeţi mai jos), şi pentru care entropia să fie *minimală* (vedeţi proprietatea de structuralitate de la ex. 33 de la cap. *Fundamente*. Cerinţa ca arborele ID3 să fie *minimal* (ca număr de niveluri / noduri) este însă mai importantă, mai practică şi mai uşor de înţeles.

  - de tip *divide-et-impera* (⇒ "*Iterative Dichotomizer*"), recursiv;
  - ◦ *1-step look-ahead*;
  - complexitate de timp (vedeţi Weka book, 2011, pag. 199):
    la antrenare, în anumite condiţii: $\mathcal{O}(d\,m \log m)$; la testare $\mathcal{O}(d)$,
    unde d este numărul de atribute, iar m este numărul de exemple;

– ID3 ca algoritm de învăţare 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 învăţare de tip *"eager"*;
  - *analiza erorilor*:
    la antrenare: ex. 7.a, ex. 10.a, ex. 40;
    [acurateţe la antrenare: ex. 6;]
    la validare: CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 1;
    la n-fold cross-validare
    la cross-validare leave-one-out (CVLOO): ex. 10.b, ex. 43.bc;
  - *robusteţe la „zgomote" şi overfitting*: ex. 10, ex. 22.bc, ex. 43, ex. 66.b;
  - *zone de decizie* şi *graniţe de separare / decizie* pentru arbori de decizie cu variabile continue: ex. 10, ex. 42, ex. 43, ex. 44.

Extensii / variante ale algoritmului ID3

– *atribute cu valori continue*: ex. 10-12, ex. 15.c, ex. 41-45; cap. *Învăţare bazată pe memorare*, ex. 11.b;
alte variante de partiţionare a intervalelor de valori pentru atributele continue: ex. 47;

– atribute discrete cu multe valori: ex. 14;

– atribute cu valori nespecificate / absente pentru unele instanţe;

– atribute cu diferite costuri asociate: ex. 15;

– reducerea caracterului "eager" al învăţării: ex. 17;

– reducerea caracterului "greedy" al învăţării:
IG cu "2-step look-ahead": ex. 18, ex. 19;
variante de tip "look-ahead" specifice atributelor continue: ex. 48;
3-way splitting (sau, mai general, n-way splitting) pentru atributele continue: ex. 13, ex. 46;

– folosirea altor *măsuri de „impuritate"* în locul câştigului de informaţie:
Gini Impurity, Misclassification Impurity: ex. 16;

– reducerea *overfitting*-ului:
reduced-error pruning (folosind un set de date de validare):
cartea ML, pag. 69-71; A. Cornuéjols, L. Miclet, 2nd ed., pag. 418-421;
rule post-pruning: cartea ML, pag. 71-72; ex. 50
top-down vs. bottom-up pruning: ex. 20, ex. 49;
pruning folosind testul statistic χ^2: ex. 21, ex. 51.

Proprietăţi ale arborilor ID3

- (P1) arborele produs de algoritmul ID3 este *consistent* (adică, în concordanţă) cu datele de antrenament, dacă acestea sunt *consistente* (adică, necontradictorii). Altfel spus, *eroarea la antrenare* produsă de algoritmul ID3 pe orice set de *date consistente* este 0: ex. 2-4, ex. 35-36;
- (P2) arborele produs de algoritmul ID3 *nu* este în mod neapărat *unic*: ex. 3, ex. 35;
- (P3) arborele ID3 nu este neapărat *optimal* (ca nr. de noduri / niveluri): ex. 4, ex. 22, ex. 36;
- (P4) influenţa *atributelor identice* şi, respectiv, a *instanţelor multiple* asupra arborelui ID3: ex. 8;
  - (P5) *o margine superioară* pentru *eroarea la antrenare* a algoritmului ID3, în funcţie de numărul de valori ale variabilei de ieşire): ex. 7.b;
  - (P6) o aproximare simplă a numărului de *instanţe greşit clasificate* din totalul de M instanţe care au fost asignate la un nod frunză al unui arbore ID3, cu ajutorul entropiei (H) nodului respectiv: ex. 39;
- (P7) *graniţele de separare* / *decizie* pentru arborii ID3 cu atribute de intrare continue sunt întotdeauna paralele cu axele de coordonate: ex. 10, ex. 12, ex. 42, ex. 43, ex. 44 şi cap. *Învăţare bazată pe memorare*, ex. 11.b.

  Observaţie: Următoarele trei proprietăţi se referă la arbori de decizie în general, nu doar la arbori ID3.
- (P8) *adâncimea maximă* a unui arbore de deczie, când atributele de intrare sunt categoriale: numărul de atribute: ex. 52.c;
  - (P9) o *margine superioară* pentru *adâncimea* unui arbore de deczie când atributele 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 funcţie de numărul de exemple şi de numărul de atribute de intrare, atunci când acestea (atributele de intrare) sunt binare: ex. 9;

Învăţare automată de tip ansamblist folosind arbori de decizie: Algoritmul AdaBoost

- Noţiuni preliminare:
  distribuţie de probabilitate discretă, factor de normalizare pentru o distribuţie de probabilitate, ipoteze „slabe“ (engl., weak hypotsesis), 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 (vedeţi ex. 64), funcţii de cost / pierdere (engl., loss function) (vedeţi ex. 63 şi ex. 23);

– pseudo-codul algoritmului AdaBoost + convergenţa erorii la antrenare: ex. 23;

– exemple de aplicare: ex. 24, 54, 55, 56, 57;

- **AdaBoost ca algoritm *per se*:**
  algoritm iterativ, algoritm de căutare (spaţiul de căutare este mulţimea combinaţiilor liniare care se pot construi peste clasa de ipoteze „slabe“ considerate), algoritm de opimizare secvenţială (minimizează o margine superioară pentru eroarea la antrenare), algoritm greedy.

– ***învăţabilitate empirică γ-slabă*:**
  definiţie: ex. 23.f
  exemplificarea unor cazuri când nu există *garanţie* pentru învăţabilitate γ-slabă: ex. 59, 60;

– **AdaBoost ca algoritm de *optimizare secvenţială* în raport cu funcţia de cost / „pierdere“ negativ-exponenţială: ex. 25;**

– **marginea de votare: ex. 26;**

– ***selectarea trăsăturilor* folosind AdaBoost; aplicare la clasificarea de documente: ex. 61;**

– **o variantă generalizată a algoritmului AdaBoost: ex. 63;**

– **recapitulare (întrebări cu răspuns *adevărat* / *fals*): ex. 28 şi 65;**

- **Proprietăţi 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) $err_{D_{t+1}}(h_t) = \frac{1}{2}$ **(ex. 23.a);**
  ca o *consecinţă*, rezultă că ipoteza h_t nu poate fi reselectată şi la iteraţia $t+1$; ea poate fi reselectată la o iteraţie ulterioară;

  (P2) Din relaţia de definiţie pentru distribuţia D_{t+1} rezultă
  $Z_t = e^{-\alpha_t} \cdot (1-\varepsilon_t) + e^{\alpha_t} \cdot \varepsilon_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$ **(ex. 23.a);**
  $\varepsilon_t \in (0, 1/2) \Rightarrow Z_t \in (0,1)$**.**

  (P3) $D_{t+1}(i) = \frac{1}{m\prod_{t'=1}^{t} Z_{t'}} e^{-y_i f_t(x_i)}$**, unde** $f_t(x_i) \stackrel{def.}{=} -\sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x_i)$ **(ex. 23.b). Produsul $y_i f_t(x_i)$ se numeşte *margine algebrică*;**

  (P4) $err_S(H_t) \leq \prod_{t'=1}^{t} Z_t$**, adică eroarea la antrenare comisă de ipoteza combinată produsă de AdaBoost este majorată de produsul factorilor de normalizare (ex. 23.c);**

  (P5) AdaBoost nu optimizează în mod direct $err_S(H_t)$, ci marginea sa superioară, $\prod_{t'=1}^{t} Z_t$; optimizarea se face în mod secvenţial (greedy): la iteraţia t se minimizează valoarea lui Z_t ca funcţie de α_t, ceea ce conduce la $\alpha_t = \ln\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}$
  (ex. 23.d);

  (P5′) $\varepsilon_i > \varepsilon_j \Rightarrow \alpha_i < \alpha_i$ **(consecinţă imediată din (P5)): ex. 60.c;**

  (P6) O consecinţă din relaţia (118) **şi (P5):** $D_{t+1}(i) = \begin{cases} \frac{1}{2\varepsilon_t} D_t(i), & i \in M \\ \frac{1}{2(1-\varepsilon_t)} D_t(i), & i \in C \end{cases}$

  (P7) $err_S(H_t)$ **nu neapărat descreşte de la o iteraţie la alta; în schimb, descresc marginile sale superioare:** $\prod_{t'=1}^{t} Z_{t'}$ **şi** $\exp(-\sum_{t'=1}^{t} \gamma_{t'}^2)$ **(ex. 23.e);**

  (P8) O condiţie suficientă pentru învăţabilitate γ-slabă, bazată pe marginea de votare: marginea de votare a oricărei instanţe de antrenament să fie de cel

puţin 2γ, la orice iteraţie a algoritmului AdaBoost (ex. 27);

(P9) Orice mulţime formată din m instanţe din $\mathbb{R}$ care sunt etichetate în mod consistent poate fi corect clasificată de către o combinaţie liniară formată din cel mult $2m$ compaşi de decizie (ex. 64.a);

(P10) Orice mulţime de instanţe distincte [şi etichetate] din $\mathbb{R}$ este γ-slab învăţabilă cu ajutorul compaşilor de decizie (ex. 62).

- **Alte metode de învăţare ansamblistă bazate pe arbori de decizie: Bagging, Random Forests;**
- **Alte metode de învăţare automată bazate pe arbori: arbori de regresie (CART).**

4. Clasificare bayesiană

Sumar

Noţiuni preliminare

- **probabilităţi şi probabilităţi condiţionate;**
- **formula lui Bayes: ex. 5.b;**
  cap. *Fundamente*, ex. 6, ex. 7, ex. 65, ex. 66;
- **independenţa [condiţională a] evenimentelor aleatoare:**
  cap. *Fundamente*, ex. 4, ex. 62, ex. 63;
- **independenţa [condiţională a] variabilelor aleatoare: ex. 9, ex. 10, ex. 12, ex. 28-35; vedeţi şi cap. *Fundamente*, ex. 15, ex. 24, ex. 70.b, ex. 79, ex. 86;**
- **distribuţii probabiliste corelate, marginale şi condiţionale: ex. 8, ex. 10, ex. 12, ex. 28; vedeţi şi cap. *Fundamente*, ex. 13, ex. 14;**
- **distribuţia gaussiană: de la cap. *Fundamente*, ex. 26, ex. 27 (pentru cazul uni-variat), ex. 88 (pentru cazul bivariat), ex. 18, ex. 28, ex. 29, ex. 30 (pentru cazul multi-variat);**
- **estimarea parametrilor pentru distribuţii de tip Bernoulli, categorial şi gaussian (ultimul doar pentru cazul clasificării bayesiene de tip gaussian);**[7]
- **ipoteze MAP vs. ipoteze ML:**
  formulare [ca soluţii la] probleme de optimizare:[8] **ex. 22;**
  exemplificare: ex. 1, ex. 2, ex. 3, ex. 21, ex. 34;
  exemplificare în cazul arborilor de decizie: ex. 4;
- **regresia logistică, chestiuni introductive:**[9] **de la cap. *Estimarea parametrilor; metode de regresie*, ex. 23.**

Algoritmi de clasificare bayesiană

- **Algoritmul Bayes Naiv şi algoritmul Bayes Corelat:**[10]
  formulare ca probleme de optimizare / estimare în sens MAP: cartea ML, pag. 167;
  pseudo-cod: vedeţi slide-uri;
  exemple de aplicare: ex. 5, ex. 7, ex. 8, ex. 9, ex. 23, ex. 24, ex. 25;
- **aplicarea / adaptarea algoritmului Bayes Naiv pentru clasificare de texte:**[11]
  ex. 6, ex. 26;
  folosirea regulii "add-one" [a lui Laplace] pentru „netezirea" parametrilor: ex. 6, ex. 27;

[7] De la cap. *Estimarea parametrilor; metode de regresie*, pentru estimarea parametrului unei distribuţii Bernoulli vedeţi ex. 1 şi ex. 29.a, pentru estimarea parametrilor unei distribuţii categoriale vedeţi ex. 31, iar pentru estimarea parametrilor unei distribuţii gaussiene vedeţi ex. 7, ex. 38, ex. 8, ex. 39 (pentru cazul uni-variat) şi ex. 10 (pentru cazul multi-variat).

[8] Vedeţi cartea ML, pag. 156-157.

[9] Vedeţi draftul capitolului suplimentar pentru cartea ML a lui T. Mitchell, *Generative and discriminative classifiers: Naive Bayes and logistic regression* (în special secţiunea 3).

[10] La secţiunea aceasta, precum şi la următoarea secţiune, considerăm (implicit) că toate variabilele de intrare sunt de tip Bernoulli sau, mai general, de tip categorial. După aceea vom considera şi variabile de intrare de tip continuu, în genere de tip gaussian. Variabila de ieşire se consideră întotdeauna de tip Bernoulli / categorial.

[11] Atenţie: Noi am folosit aici versiunea de bază a algoritmului Bayes Naiv; varianta "bag of words" (vedeţi cartea Machine Learning a lui Tom Mitchell, pag. 183) diferă uşor de aceasta.

– **calculul ratei medii a erorilor pentru algoritmii Bayes Naiv şi Bayes Corelat: ex. 10, ex. 11, ex. 28, ex. 29, ex. 30, ex. 31, ex. 35;**

– **evidenţierea grafică a neconcordanţei predicţiilor făcute de clasificatorii Bayes Naiv şi Bayes Corelat: ex. 12.**

Proprietăţi ale algoritmilor Bayes Naiv şi Bayes Corelat

– **(P0) dacă proprietatea de independenţă condiţională a atributelor de intrare în raport cu variabila de ieşire se verifică, atunci rezultatele produse de către cei doi algoritmi (Bayes Naiv şi Bayes Corelat) în faza de testare coincid;**

– **(P1) numărul de parametri necesari de estimat din date: liniar pentru Bayes Naiv ($2d+1$) şi exponenţial pentru Bayes Corelat ($2^{d+1}-1$): ex. 7.e, ex. 25.ab, ex. 30;**

– **(P2) complexitatea algoritmului Bayes Naiv:**

  complexitatea de spaţiu: $\mathcal{O}(dn)$
  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 instanţa de test];**

– **(P3) algoritmul Bayes Corelat poate produce eroare [la clasificare] din cauza faptului că ia decizia în sensul unui vot majoritar. Algoritmul Bayes Naiv are şi el această „sursă“ de eroare; în plus el poate produce eroare şi din cauza faptului că lucrează cu presupoziţia de independenţă condiţională (care nu este satisfăcută în mod neapărat);**

– **(P4) acurateţea [la clasificare a] algoritmului Bayes Naiv scade atunci când unul sau mai multe atribute de intrare sunt duplicate: ex. 10.d, ex. 28.def;**

– **(P5) în cazul „învăţării“ unei funcţii booleene (oarecare), rata medie a erorii produse la antrenare de către algoritmul Bayes Corelat (spre deosebire de Bayes Naiv!) este 0: ex. 30.d;**

– **(P6) complexitatea de eşantionare: de ordin logaritmic pentru Bayes Naiv şi de ordin exponenţial pentru Bayes Corelat: ex. 13;**

– **(P7) corespondenţa dintre regula de decizie a algoritmului Bayes Naiv (când toate variabilele de intrare sunt de tip Bernoulli) şi regula de decizie a *regresiei logistice* şi, în consecinţă, liniaritatea graniţelor de decizie: ex. 14.**

– ***comparaţii* între algoritmul Bayes Naiv şi alţi algoritmi de clasificare automată: ex. 33, ex. 35.**

Algoritmii Bayes Naiv şi Bayes Corelat cu variabile de intrare *de tip gaussian*

- **Aplicare: G[N]B: ex. 15, ex. 36, ex. 40;**[12] **GJB: ex. 37; GNB vs GJB: ex. 16.**
- **Proprietăţi:**

[12] Vedeţi şi ex. 38 de la cap. *Estimarea parametrilor; metode de regresie.*

- **(P0′) presupunem că variabila de ieşire este booleană, i.e. ia valorile 0 sau 1; dacă pentru orice atribut de intrare, variabilele condiţionale $X_i|Y = 0$ şi $X_i|Y = 1$ au distribuţii gaussiene de varianţe 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: ex. 17, ex. 36.a;**
- **(P1′) similar, presupunem că variabila de ieşire este booleană;**
  dacă variabilele de intrare (notaţie: $X = (X_1, \ldots, X_d)$) au distribuţiile [corelate] condiţionale $X|Y = 0$ şi $X|Y = 1$ de tip gaussian [multi-variat], cu matricele de covarianţă egale ($\Sigma_0 = \Sigma_1$), atunci regula de decizie a algoritmului "full" / Joint Gaussian Bayes este şi ea echivalentă (ca formă) cu cea a regresiei logistice, deci separarea realizată este tot de formă liniară: ex. 18;
- **(P2′) când variabilele de intrare satisfac condiţii mixte de tip (P0′) sau (P7), atunci concluzia – separare liniară – se menţine: ex. 39.b;**
- **(P3′) dacă în condiţiile de la propoziţiile (P0′)-(P2′) presupoziţia de independenţă condiţională este satisfăcută, iar numărul de instanţe de antrenament tinde la infinit, atunci rezultatul de clasificare obţinut de către algoritmul Bayes Naiv gaussian este identic cu cel al regresiei logistice: ex. 19.a.**

  Atunci când presupoziţia de independenţă condiţională nu este satisfăcută, iar numărul de instanţe de antrenament tinde la infinit, regresia logistică se comportă mai bine decât algoritmul Bayes Naiv [gaussian]: ex. 19.b;
- **(P4′) nu există o corespondenţă 1-la-1 între parametrii calculaţi de regresia logistică şi între parametrii calculaţi de algoritmul Bayes Naiv [gaussian]: ex. 20.a;**
- **(P5′) atunci când varianţele distribuţiilor gaussiene care corespund probabilităţilor condiţionale $P(X_i|Y = k)$ depind şi de eticheta k, separatorul decizional determinat de algoritmul Bayes Naiv gaussian nu mai are forma regresiei logistice: ex. 38 (similar, pentru algoritmul Bayes Corelat gaussian, atunci când $\Sigma_0 \neq \Sigma_1$: ex 37);**
- **(P6′) parametrii algoritmului Bayes Corelat gaussian se pot estima în timp liniar în raport cu numărul de instanţe din setul de date de antrenament: ex. 20.b.**

5. Învăţare bazată pe memorare

Sumar

Noţiuni preliminare

– **măsuri de distanţă, măsuri de similaritate: ex. 2;**

– **normă într-un spaţiu vectorial; [măsura de] distanţă 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 împrieteneşti, ca să-ţi spun cine eşti“): ex. 15.a;**

– **exemple (simple) de aplicare: ex. 1, ex. 2, ex. 15.b-d;**

– **complexitate de spaţiu: $\mathcal{O}(d\,n)$**
complexitate de timp:

la antrenare: $\mathcal{O}(d\,n)$
la testare: $\mathcal{O}(d\,n\log n)$
[LC: $\mathcal{O}(d\,n\,k\log k)$ pt. $k > 1$ (worst case) şi $\mathcal{O}(d\,n)$ 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, Willey, pag. 163-173;**

– **k-NN ca algoritm ML “lazy” (vs. “eager”):**
suprafeţe de decizie şi graniţe 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;**
- **variaţia numărului de erori (la antrenare şi respectiv testare) în funcţie de valorile lui k: ex. 22, ex. 23.ab;**
  k-NN ca metodă ne-parametrică; alegerea lui k: CV: ex. 23.c;
- **CVLOO: ex. 3, ex. 12.b, ex. 16, ex. 24.a, ex. 20.b;**
- **sensibilitatea / robusteţea la „zgomote“: ex. 5, ex. 15;**
- **eroarea asimptotică: ex. 10, ex. 25.**

– **efectul trăsăturilor redundante sau irelevante;**

– **alegerea valorii convenabile pentru k: ex. 21.**

Proprietăţi ale algoritmului k-NN

- **(P0) output-ul algoritmului k-NN pentru o instanţă oarecare de test x_q depinde de valoarea lui k: ex. 1;**

- **(P1) pe seturi de date de antrenament *consistente*, eroarea la antrenare produsă de algoritmul 1-NN este 0: ex. 2, ex. 12.a;**
- **(P2) output-ul algoritmului k-NN, precum şi suprafeţele de decizie şi separatorii decizionali depind de *măsura de distanţă* folosită: ex. 7;**
- **(P3) „blestemul marilor dimensiuni“ (engl., the curse of dimensionality): în anumite condiţii, numărul de instanţe de antrenament necesare pentru a avea un *cel mai apropiat vecin* situat la distanţă *rezonabilă* faţă de instanţa de test x_q creşte exponenţial în funcţie de numărul de atribute folosite: ex. 9;**
- **(P4) în anumite condiţii, rata medie a *erorii asimptotice* a algoritmului 1-NN este mărginită superior de dublul ratei medii a erorii algoritmului Bayes Corelat: ex. 10, ex. 25.**

Comparaţii cu alţi 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 distanţă (decât distanţa euclidiană): ex. 7;**

– **k-NN cu *ponderarea distanţelor* (engl., distance-weighted k-NN): cartea ML, pag. 236-238 (formulele 8.2, 8.3, 8.4);**[13]

– **algoritmul lui Shepard: ex.8.**

Alte metode de tip IBL

– **reţele RBF: cartea ML, pag. 238-240;**

– **raţionare bazată pe cazuri (engl., case-based reasoning): cartea ML, pag. 240-244.**

[13] Sectiunea 8.3 din cartea ML (pag. 236-238) se referă la regresia [liniară] local-ponderată ca o formă mai generală de aproximare a [valorilor] funcţiilor, în raport cu cele calculate de către algoritmul k-NN atunci când se foloseşte ponderarea distanţelor.

6. Clusterizare

Sumar

6.0 Noţiuni de bază

- instanţă neetichetată vs. instanţă etichetată (exemplu de antrenament);
- învăţare nesupervizată (clusterizare) vs. învăţare supervizată (clasificare);
- [funcţie / măsură de] *distanţă* definită pe $\mathbb{R}^d \times \mathbb{R}^d$: ex. 2 de la capitolul *Învăţare bazată pe memorare*;
- cluster / grup / grupare / bin (engl.) vs. clasă;
- tipuri de clusterizare: ierarhică vs. neierarhică;
- tipuri de ierarhii: ierarhii (arbori de clusterizare, dendrograme) obişnuite vs. ierarhii plate (engl., flat hierarchies);
  exemple: ex. 1.a şi respectiv ex. 1.b, ex. 6.a;
- tipuri de apartenenţă a unei instanţe la un cluster: hard vs. soft (ultima numai pt. clusterizare ne-ierarhică).

6.1. Clusterizare ierarhică

6.1.1. Noţiuni specifice

- [funcţie de] *similaritate* între clustere, definită pe baza [extinderii] noţiunii de distanţă la $\mathcal{P}(X) \times \mathcal{P}(X)$, unde $X \subset \mathbb{R}^d$ este mulţimea de instanţe, iar $\mathcal{P}(X)$ este mulţimea părţilor lui X;
  tipuri de [funcţii de] similaritate:

  "single-linkage":[14] $d(A,B) = \min\{d(x,y) | x \in A, y \in B\}$
  "complete-linkage":[15] $d(A,B) = \max\{d(x,y) | x \in A, y \in B\}$
  "average-linkage": $d(A,B) = \frac{1}{|A|\,|B|} \sum_{x \in A, y \in B} d(x,y)$
  metrica lui Ward: ex. 30, 31.

  În general, putem considera $sim(A,B) = 1/(1+d(A,B))$ sau chiar $sim(A,B) = 1/d(A,B)$ când ne referim doar la clustere non-singleton;
  proprietate / restricţie: $sim(A \cup B, C) \leq \min\{sim(A,C), sim(B,C)\}$ pentru orice clustere A, B selectate de algoritmul de clusterizare ierarhică la un pas oarecare [al algoritmului de clusterizare ierarhică] şi orice alt cluster C;
- [funcţie de] *coeziune* [internă] a unui cluster (sau: între elementele / instanţele dintr-un cluster);
  exemplu (pentru clustere non-singleton):
  $\mathrm{coh}(A) = \left(\frac{1}{\mathrm{C}^2_{|A|}} \sum_{x,y \in A} d(x,y) \right)^{-1} = \frac{\mathrm{C}^2_{|A|}}{\sum_{x,y \in A} d(x,y)}$.

[14] Sau: nearest-neighbour.
[15] Sau: furthest-neighbour.

6.1.2. 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 şi "greedy"; rezultatele (ierarhiile) obţinute nu sunt determinate neapărat în mod unic: ex. 3.b;

– exemple de aplicare: ex. 1-5, ex. 25-29 (pentru bottom-up), respectiv ex. 6 (pentru top-down);

– implementări: ex. 33, ex. 31, ex. 34.

6.1.3 Proprietăţi

– (P0) clusterizarea folosind similaritate de tip "single-linkage" are tendinţa să creeze clustere alungite; invers, folosind similaritate "complete-linkage" sau "average-linkage", se formează clustere de formă mai degrabă sferică: ex. 5 şi ex. 28;

– (P1) numărul maxim de niveluri dintr-o dendrogramă (văzută ca arbore în sensul teoriei grafurilor) este $n-1$, unde n este numărul de instanţe de clusterizat: ex. 4.a; numărul minim de niveluri: $\lceil \log_2 n \rceil$; ex. 4.b;

– (P2) există o anumită corespondenţă între clusterizare ierarhică cu similaritate de tip

  • "single-linkage" şi aflarea *arborelui* [*de acoperire*] *de cost minim* dintr-un graf: ex. 6;

  • "complete-linkage" şi aflarea unei *clici* (subgraf maximal complet) dintr-un graf (vedeţi Manning & Schütze, *op. cit.*, pag. 506-507);

– (P3) algoritmul de clusterizare aglomerativă la al cărui pseudo-cod am făcut referire mai sus are complexitate $\mathcal{O}(n^3)$: ex. 25; atunci când se foloseşte single-linkage sau complete-linkage, există însă versiuni / algoritmi de complexitate $\mathcal{O}(n^2)$: SLINK (1973) şi respectiv CLINK (1976);

– la clusterizare ierarhică aglomerativă cu similaritate "average-linkage":

  (P4) dacă se foloseşte ca măsură de similaritate între 2 instanţe cosinusul unghiului dintre vectorii care reprezintă instanţele şi se „normalizează" aceşti 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 şi calculul „distanţei" dintre două clustere se pot face în timp constant: ex. 32.

6.2. Clusterizare neierarhică,

folosind asignare "hard" a instanţelor la clustere

6.2.1 Noţiuni specifice

– centroid (centru de greutate) al unui cluster,
K-partiţie, K-configuraţie [iniţială] a centroizilor: ex. 11;

– o funcţie de evaluare a „calităţii“ clusterelor (sau: funcţie 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-partiţie, μ este K-configuraţie de centroizi, iar $\mu_{C(x_i)}$ este centroidul cel mai apropiat de x_i: ex. 12.

6.2.2 Algoritmul K-means

– pseudo-cod (o versiune [mai] generală): Manning & Schütze, *op. cit.*, pag. 516; alternativ, vedeţi enunţul ex. 12 (sau, echivalent, folosind variabile-indicator: ex. 39);
exemple de aplicare: ex. 7-11, ex. 15.a, ex.19.a, ex. 20.a, ex. 35, ex. 36.

– exemple de *euristici pentru iniţializarea centroizilor*:
iniţializare arbitrară / random în $\mathbb{R}^d$ sau în $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^d$ (setul de date de clusterizat);
aplicare în prealabil a unui algoritm de clusterizare ierarhică;
folosind o anumită distribuţie probabilistă definită pe X: K-means++ (David Arthur, Sergei Vassilvitskii, 2007): ex. 43.

– exemple de *criterii de oprire*:
după efectuarea unui număr maxim de iteraţii (fixat iniţial);
când componenţa clusterelor nu se mai modifică de la o iteraţie la alta;
când poziţiile centroizilor nu se mai modifică de la o iteraţie la alta;
când descreşterea valorii criteriului J_K de la o iteraţie 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*:
spaţiul de căutare este mulţimea tuturor K-partiţiilor care se pot forma pe dataset-ul de intrare;
(P0) întrucât acest spaţiu de căutare (deşi este finit) este exponenţial (K^n), K-means explorează doar parţial spaţiul de căutare, procedând *iterativ*: el pleacă de la o „soluţie“ (K-partiţie) aleasă eventual în mod arbitrar / aleatoriu şi o „îmbunătăţeşte“ la fiecare iteraţie;

(P1) soluţia găsită este dependentă de iniţializarea centroizilor: ex. 10;
(P1′) mai mult, chiar la o aceeaşi iniţializare, rezultatele pot diferi(!) dacă avem instanţe multiple / redundante, situate la egală distanţă de 2 centroizi la o iteraţie oarecare: ex. 12.b;
(P1″) rezultatele lui K-means sunt dependente [şi] de măsura de distanţă folosită: ex. 42.

K-means poate fi văzut şi ca *algoritm de optimizare* — vedeţi criteriul J_K de mai sus;
(P2) strategia de căutare / optimizare folosită de K-means este de tipul *descreştere pe coordonate* (engl., coordinate descent), i.e. descreştere 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. 40.b.

– ca algoritm de *învăţare automată*:
[urmat de] „generalizare“: o instanţă nouă x se asociază clusterului având

centroidul cel mai apropiat de x;
(P3) „graniţele“ de separare dintre [perechile de] clustere produse de K-means sunt [doar] liniare, [cel puţin] atunci când se foloseşte distanţa euclidiană: ex. 11.b;
(P3′) este însă posibil să se obţină separatori neliniari dacă se foloseşte o versiune „kernelizată“ a algoritmului K-means: ex. 44;
(P4) rezultatele lui K-means pot fi influenţate de prezenţa outlier-elor: ex. 10.

– chestiunea alegerii unei valori convenabile / „naturale” pentru K (pentru un dataset dat): ex. 38 (şi CMU, 2012f, E. Xing, A. Singh, HW3, ex. 1.de).
– adaptarea algoritmului K-means pentru cazul în care în locul distanţei euclidiene se foloseşte distanţa Manhattan: ex. 42;
– implementare: ex. 47.

6.2.3 Alte proprietăţi 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 mulţimea tuturor K-partiţiilor peste mulţimea de instanţe, $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^d$:
  ◦ (P5) pentru $K > 0$ fixat, $|\mathcal{P}_K| = K^n$, deci este finit, şi există $\underline{J_K} \overset{not.}{=} \min_C J_K(C, \mu_C)$;
  acest minimum ($\underline{J_K}$) se poate obţine prin explorarea exhaustivă a spaţiului $\mathcal{P}_K$, însă consumul de timp este prohibitiv în practică: ex. 12.b;
  ◦ (P6) valoarea 0 pentru $\underline{J}$ este atinsă, şi anume atunci când $K = n$, C este K-partiţia de clustere singleton $C_i = \{x_i\}$, iar $\mu_i = x_i$, pentru $i = 1, \ldots, n$ (ex. 38);
  ◦ (P7) $\underline{J_1} \geq \underline{J_2} \geq \ldots \geq \underline{J_{n-1}} \geq \underline{J_n} = 0$: ex. 13.

– (P8) dacă $d = 1$, deci $x_1, x_2, \ldots, x_n \in \mathbb{R}$,
  ◦ orice K-partiţie $(C_1, \ldots, C_K)$ pentru care se atinge $\underline{J_K}$ este de forma unei colecţii de „intervale“: $C_1 = \{x_1, \ldots, x_{i_1}\}$, $C_2 = \{x_{i_1+1}, \ldots, x_{i_2}\}$, …, $C_K = \{x_{i_{K-1}+1}$, …, $x_n\}$, cu $i_1 < i_2 < \ldots < i_{K-1} < i_K = n$;
  ◦ există un algoritm [de programare dinamică] de complexitate $\mathcal{O}(Kn^2)$ care calculează $\underline{J_K}$: ex. 40.

– în legătură cu J_K şi algoritmul K-means:
  ◦ (P9) $J_K(C^{t-1}, \mu^{t-1}) \geq J_K(C^t, \mu^t)$ la orice iteraţie ($t > 0$) a algoritmului K-means: ex. 12.a;
  ◦ (P9′) în consecinţă, dacă se impune restricţia ca la fiecare iteraţie 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 paşi;
  ◦ (P10) în vreme ce minimizează *coeziunea intra-clustere*, i.e. o variantă ponderată a „sumelor celor mai mici pătrate“ calculate pe clustere,

$$\sum_{k=1}^{K} \left(\frac{\sum_{i=1}^{n} \gamma_{ik}}{\sum_{i=1}^{n} \gamma_{ik}} \right) \|x_i - \mu_k\|^2,$$

unde $\gamma_{ik} = 1$ dacă x_i aparţine clusterului de centroid μ_k şi $\gamma_{ik} = 0$ în caz contrar, algoritmul K-means maximizează (în mod aproximativ!) o sumă ponderată a distanţelor 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 instanţelor $x_1, x_2, \ldots, x_n$ (ex. 39).

6.3. Clusterizare neierarhică, folosind asignare "soft" a instanţelor la clustere

6.3.1 Noţiuni preliminare

- variabile aleatoare (discrete, resp. continue);
  media, varianţa şi co-varianţa variabilelor aleatoare;
- vector de variabile aleatoare; matrice de covarianţă pentru un astfel de vector; proprietăţi: matricea de covarianţă trebuie să fie în mod necesar simetrică şi pozitiv definită: ex. 18 de la capitolul de *Fundamente*;
- distribuţie (funcţie de densitate) de probabilitate (p.d.f.);
  parametri ai unei distribuţii de probabilitate;
  distribuţia gaussiană: cazurile uni- şi multi-variat;
- mixtură de distribuţii probabiliste:
  văzută ca o formă particulară de *combinaţie liniară* de distribuţii de probabilitate $\pi_1\Psi_1 + \pi_2\Psi_2 + \ldots + \pi_k\Psi_k$ (cu $\pi_i \geq 0$ şi $\sum_{i=1}^{k} \pi_i = 1$),
  definită [şi mai specific] scriind distribuţia $P(X)$ ca o sumă ponderată de probabilităţi condiţionate: $\sum_z P(X|Z)P(Z)$, unde X sunt variabilele „observabile", iar variabila Z (eventual multiplă) poate fi „neobservabilă" / „latentă" / „ascunsă";
  exemple: o mixtură de distribuţii categoriale, respectiv o mixtură de distribuţii Bernoulli: ex. 23 şi ex. 85 de la capitolul de *Fundamente*; o mixtură de distribuţii gaussiene multi-variate: ex. 89 de la capitolul de *Fundamente*; o mixtură de distribuţii oarecare: ex. 90 de la capitolul de *Fundamente*;
- funcţie de *verosimiliate* a unui set de date (D), în raport cu o distribuţie probabilistă dată: $L(\theta) = P(D|\theta)$, unde prin θ se notează parametrii respectivei distribuţii. Exemplificare: ex. 1.abd, ex. 2 de la capitolul *Estimarea parametrilor; metode de regresie*;
- MLE (Maximum Likelihood Estimation): estimarea [valorilor] parametrilor unei distribuţii probabiliste în sensul maximizării verosimilităţii datelor disponibile. Exemplificare: capitolul *Estimarea parametrilor; metode de regresie*, ex. 1-11, ex. 29-40. Aplicare în cazul distribuţiei gaussiene uni-variate: ex. 15.ab de la capitolul *Clasificare bayesiană*;
- analiza discriminativă gaussiană: ex. 38 de la capitolul *Estimarea parametrilor; metode de regresie*;
- Observaţie: Algoritmul EM este [sau, mai degrabă, poate fi folosit ca] o metodă de estimare a parametrilor unei mixturi de distribuţii probabiliste. *Alternativ*, pentru acelaşi obiectiv pot fi folosite alte metode, de exemplu *metoda gradientului ascendent*: ex. 59.

6.3.2 Algoritmul EM pentru clusterizare
prin estimarea parametrilor unui
model de mixturi de distribuţii gaussiene (EM/GMM)

– **pseudo-cod:**
cazul uni-dimensional, varianta când doar parametrul μ este lăsat liber: ex. 48 (cf. *Machine Learning*, Tom Mitchell, 1997, pag. 193); aplicare: ex. 15.b, ex. 16;
cazul uni-dimensional, varianta când toţi parametrii (π, μ şi σ) sunt lăsaţi liberi: ex. 17 (aplicare: ex. 15.c);
alte variante: ex. 49, ex. 50, ex. 51;
cazul multi-dimensional, varianta când toţi parametrii (π, μ şi Σ) sunt lăsaţi liberi: ex. 23; alte variante: ex. 52, ex. 54;
aplicarea algoritmului EM/GMM, cazul bivariat: ex. 19.b, ex. 20.b, ex. 21, ex. 22, ex. 55, ex. 56, ex. 57;

– **schema algoritmică EM: vedeţi Tom Mitchell, *Machine Learning* book, 1997, pag. 194-195;**

– **ca algoritm de *învăţare statistică*:**
algoritmul EM poate fi văzut ca o metodă de estimare a parametrilor (engl., parameter fitting);

– **ca algoritm *per se*:**

  ◦ ***algoritm iterativ*: pleacă de la o soluţie (instanţiere pentru parametri) aleasă eventual în mod arbitrar / aleatoriu şi o „îmbunătăţeşte“ la fiecare iteraţie. Soluţia găsită este dependentă de valorile iniţiale ale parametrilor;**

  ◦ ***algoritm de optimizare*:**

  în *esenţă* / *rezumat*, metoda de maximizare a funcţiei de *log-verosimilitate a datelor observabile* $\log P(X|\theta)$ **este maximizarea la fiecare iteraţie** t **a unei funcţii auxiliare** Q_t**, care constituie o margine inferioară a lui** $\log P(X|\theta)$**, şi anume media funcţiei de *log-verosimilitate a datelor complete* în raport cu distribuţia de probabilitate a *variabilelor neobservabile* la iteraţia** t**;**

  aşadar, la fiecare iteraţie t **se calculează funcţia „auxiliară“** $Q_t(\theta|\theta^{(t)})$**, care reprezintă media funcţiei de log-verosimilitate a datelor „complete“ (cele „observabile“ plus cele „neobservabile“), unde** $\theta^{(0)}$**, constând din valorile iniţiale ale parametrilor mixturii (**θ**), se alege în mod arbitrar, iar apoi** $\theta^{(t+1)} = \text{argmax}_\theta\, Q_t(\theta|\theta^{(t)})$**;**
  media reprezentată de funcţia Q_t **se calculează în funcţie de distribuţiile condiţionale ale variabilelor „neobservabile“** Z **în raport cu datele observabile** X **şi cu** $\theta^{(t)}$**;**

  (P0) Se poate demonstra că funcţia Q_t **constituie o *margine inferioară* pentru funcţia de log-verosimilitate a variabilelor „observabile“,** $\log P(X|\theta)$**: ex. 1 de la capitolul *Algoritmul EM*;**

  (P1) *Teorema* de corectitudine (vedeţi ex. 1 şi în special ex. 2 de la capitolul *Algoritmul EM*) pe de o parte garantează faptul că la fiecare iteraţie a algoritmului EM, log-verosimilitatea datelor „observabile“, $\log P(X|\theta^{(t)})$ **nu descreşte (ci fie creşte, fie rămâne neschimbată),**
  dar pe de altă parte nu garantează găsirea optimului global al funcţiei de log-verosimilitate a datelor „observabile“, $\log P(X|\theta)$**, ci eventual a unui optim local;**

– **ca algoritm de *învăţare automată*:**
algoritmul EM este o metodă de identificare / învăţare de ipoteze ML (Maximum Likelihood); vedeţi capitolul / secţiunea 6.4 din cartea *Machine Learning*;
învăţare în prezenţa unor variabile aleatoare ne-observabile(!);
[urmată eventual de] „generalizare“: o instanţă nouă x se asociază clusterului (i.e., distribuţiei) j pentru care se atinge $\max_{j'} P(X = x|h_{j'})P(h_{j'})$;

(P2) Rezultatele algoritmului EM depind (ca şi la K-means) de valorile atribuite parametrilor la iniţializare (ex. 15.c).

(P3) Anumite valori atribuite iniţial parametrilor algoritmului EM pot provoca rularea la infinit a algoritmului, fără ca [la pasul M] valorile parametrilor să se modifice de la o iteraţie la alta: ex. 18.c;

(P4) Spre deosebire de cazul algoritmului K-means, suprafeţele / graniţele de separare create de algoritmul EM/GMM nu sunt în mod neapărat liniare (vedeţi de exemplu situaţiile întâlnite la rezolvarea ex. 15.c, pag. 556, sau la ex. 57.c şi ex. 58.c).

– **(P5) Comparativ cu algoritmul K-means, algoritmul EM/GMM este în general mai lent — mişcarea centroizilor poate explora într-o manieră mai fină spaţiul (vedeţi de exemplu ex. 19) —, iar din acest motiv el poate să obţină uneori rezultate mai bune / convenabile (vedeţi spre exemplu ex. 20);**
EM/GMM este mai robust la influenţa outlier-elor;

(P6) Apare un fenomen de “atracţie” reciprocă a mediilor gaussienelor (aceste medii fiind echivalentul centroizilor din algoritmul K-means), datorită faptului că fiecare instanţă aparţine (cu o anumită probabilitate) la fiecare cluster. Atracţia mediilor este cu atât mai puternică cu cât varianţele sunt mai mari. (Vedeţi spre exemplu ex. 15.b.)

6.3.3 Alte proprietăţi ale algoritmului EM/GMM

– **Pentru distribuţii gaussiene multi-variate:**
◦ (P7) în cazul cel mai general (deci când matricea Σ nu este neapărat diagonală), datele generate de acest tip de distribuţie se grupează în elipse (corpuri elipsoidale) cu axele de simetrie [desigur, perpendiculare, dar altfel] nerestricţionate.
◦ (P7′) dacă matricea de covarianţă Σ este diagonală, atunci distribuţia gaussiană respectivă este echivalentă cu un set / vector de variabile gaussiene uni-variate independente: ex. 28 de la capitolul de *Fundamente*;
◦ (P7″) dacă matricea Σ este de forma $\sigma^2 I$, unde I este matricea identitate, datele generate de respectiva distribuţie tind să se grupeze în sfere;
◦ (P7‴) dacă matricea Σ este diagonală (fără nicio altă restricţie), datele generate se grupează în elipse (sau: corpuri elipsoidale) având axele de simetrie paralele cu axele sistemului de coordonate;

– **(P8) Legătura dintre algoritmul K-means şi algoritmul EM/GMM (cazul multivariat):**
atunci când $\Sigma = \sigma^2 I$, iar $\sigma^2 \to 0$ (şi sunt satisfăcute încă două restricţii), algoritmul EM/GMM tinde să se comporte ca şi algoritmul K-means: ex. 54;

– **O legătură interesantă între algoritmul EM/GMM şi metoda gradientului ascendent, în cazul în care matricele de covarianţă sunt de forma $\sigma_k^2 I$: ex. 59;**

– **O legătură interesantă între clasificatorul Bayes Naiv gaussian şi algoritmul EM/GMM în cazul în care matricele de covarianţă sunt de forma $\sigma_k^2 I$: o *variantă semi-supervizată* a algoritmului EM/GMM: ex. 60.**

– ***Schema algoritmică EM*** **(vedeţi Tom Mitchell, *Machine Learning* book, 1997, pag. 194-195) are diverse variante şi aplicaţii:**

  • **calculul parametrilor pentru mixturi de diverse distribuţii [nu doar gaussiene]: vedeţi capitolul *Algoritmul EM*;**

  • **calculul parametrilor pentru gramatici probabiliste independente de context (engl., probabilistic context-free grammars, PCFG);**

  • **calculul parametrilor modelelor Markov ascunse (engl., hidden Markov models, HMM);**

  • **calculul parametrilor reţelelor bayesiene (engl., Bayes nets);**

  • **calculul parametrilor reţelelor de funcţii cu baza radială (engl., radial basis functions, RBF), o familie de reţele neuronale artificiale; etc.**

– **Proprietăţi pentru schema algoritmică EM: vedeţi cele menţionate mai sus în legătură cu algoritmul EM văzut ca *algoritm de optimizare*.**

7. Algoritmul EM

Sumar

Noţiuni preliminare

– **estimarea parametrilor unei distribuţii probabiliste în sensul verosimilităţii maxime (MLE) respectiv în sensul probabilităţii maxime a posteriori (MAP): vedeţi capitolul de *Fundamente*;**

– **tipuri / clase de distribuţii probabiliste: vedeţi capitolul de *Fundamente*;**

– **mixturi de distribuţii probabiliste: vedeţi capitolul de *Fundamente* şi capitolul de *Clusterizare*;**

– **metoda "coordinate ascent" pentru rezolvarea problemelor de optimizare: ex. 1;**

– **metoda multiplicatorilor lui Lagrange pentru rezolvarea problemelor de optimizare cu restricţii: ex. 5, ex. 7 şi ex. 15.**

Schema algoritmică EM

– **pseudo-cod: *Machine Learning*, Tom Mitchell, 1997, pag. 194-195;**

– **fundamentare teoretică: ex. 1 — pentru funcţia $F(q, \theta)$, care reprezintă o margine inferioară a funcţiei de log-verosimilitate a datelor complete, se face maximizarea prin metoda creţerii pe coordonate (engl., coordinate ascent) — şi ex. 2 (monotonia funcţiei de log-verosimilitate a datelor complete);**[16]

– **chestiuni metodologice (relativ la iniţializarea parametrilor): ex. 22.**

EM pentru modelarea de mixturi de distribuţii probabiliste

– **varianta generală: *An Introduction to Expectation-Maximization*, Dahua Lin;**

– **diverse instanţe ale acestei „variante":**

mixturi de distribuţii *Bernoulli*: ex. 14 şi ex. 4;

mixturi de distribuţii *categoriale*: ex. 5 şi ex. 15;

mixturi de distribuţii *Poisson*: ex. 18;

mixturi de distribuţii *Gamma*: ex. 19.

Alte instanţe / aplicaţii ale schemei algoritmice EM

– **EM pentru estimarea unui parametru [de tip probabilitate] pentru o distribuţie discretă [în ocurenţă, o distribuţie *categorială*], în condiţiile existenţei unei variabile „neobservabile": ex. 3;**
similar pentru distribuţia *multinomială*: ex. 13;

– **EM pentru estimarea tuturor parametrilor unei distribuţii *categoriale*: ex. 12;**

– **EM pentru estimarea parametrului unei distribuţii *Poisson* în condiţiile în care o parte din valorile date lipsesc: ex. 10;**

[16] Legătura cu funcţia auxiliară de la iteraţia t: $Q(\theta \mid \theta^{(t)}) = \underbrace{F(P(z \mid x, \theta^{(t)}), \theta)}_{G_t(\theta)} - H[P(z \mid x, \theta^{(t)})]$.

- **EM pentru estimarea parametrilor a două distribuţii probabiliste atunci când se dau instanţe care sunt generate de *suma* celor două distribuţii:**
  distribuţii *exponenţiale*: ex. 8,
  distribuţii *gaussiene*: ex. 20;
- **algoritmul Bayes Naiv ne-supervizat, i.e. algoritmul EM pentru [modelare] de mixturi de distribuţii *categoriale multi-variate*, cu presupunerea de independenţă condiţională a atributelor de intrare în raport cu atributul de ieşire (eticheta): ex. 7 (varianta de asignare "soft" a instanţelor la cluster) şi ex. 17 (varianta "hard");**
- **EM pentru *estimarea probabilităţii de selecţie* a unei componente din cadrul unei mixturi [i.e., combinaţie liniară] de două distribuţii probabiliste oarecare: ex. 9;**
- **EM pentru *modelul mixturii domeniilor semantice* (engl., topic model) pentru clusterizare de *documente*: [ex. 6 şi] ex. 16;**
- **EM pentru *estimarea* [nu în sens MLE, cum a fost cazul până aici, ci] *în sens MAP*: ex. 20.**

8. Reţele neuronale artificiale

Sumar

Noţiuni preliminare

- funcţie matematică; compunere de funcţii reale;
  calculul valorii unei funcţii pentru anumite valori specificate pentru argumentele / variabilele ei;
- funcţie prag (sau, treaptă), funcţie liniară, funcţie sigmoidală (sau, logistică), funcţie sigmoidală generalizată;
  separabilitate liniară pentru o mulţime de puncte din $\mathbb{R}^d$;
- ecuaţii asociate dreptelor în plan / planelor în spaţiu / hiper-planelor în spaţiul $\mathbb{R}^d$;
  ecuaţia dreptei în plan care trece prin două puncte date;
  semnele asociate punctelor din semi-planele determinate de o dreaptă dată în plan;
- derivate ale funcţiilor elementare de variabilă reală; derivate parţiale
- vectori; operaţii cu vectori, în particular produsul scalar al vectorilor;
- metoda gradientului descendent (ca metoda de optimizare); avantaje şi dezavantaje; ex. 54de la cap. *Fundamente*, ex. 22, ex. 34, ex. 35.

Câteva noţiuni specifice

- *unităţi* neuronale artificiale (sau, *neuroni* artificiali, *perceptroni*);
  tipuri de neuroni artificiali: neuroni-prag, liniari, sigmoidali;
  componente ale unui neuron artificial: input, componenta de sumare, componenta / funcţia de activare, output;
  funcţia matematică reprezentată / calculată de un neuron artificial;
- *reţea* neuronală artificială; reţele de tip feed-forward;
  niveluri / straturi de neuroni, niveluri ascunse, niveluri de ieşire;
  ponderi asociate conexiunilor dintr-o reţea neuronală artificială;
  funcţia matematică reprezentată / calculată de o reţea neuronală artificială;
  graniţe şi zone de decizie determinate de o reţea neuronală artificială;
  funcţia de eroare / cost (engl., loss function).

Câteva proprietăţi relative la *expresivitatea* reţelelor neuronale artificiale

- (P0) Toate cele trei tipuri de neuroni artificiali (prag, liniar, sigmoidal) produc *separatori liniari.*

  Consecinţă: Conceptul XOR nu poate fi reprezentat / învăţat cu astfel de „dispozitive“ simple de clasificare.
- (P0′) Reţelele neuronale artificiale pot determina graniţe de decizie neliniare (şi, în consecinţă, pot reprezenta concepte precum XOR).

  Observaţie: Reţele de unităţi sigmoidale pot determina graniţe de decizie curbilinii: ex. 8.

– **(P1) Reţele de neuroni diferite (ca structură şi / sau tipuri de unităţi) pot să calculeze o aceeaşi funcţie: ex. 3 şi ex. 1.c vs. ex. 2.**

  (P1′) Dată o topologie de reţea neuronală (i.e., graf de unităţi neuronale al căror tip este lăsat nespecificat), este posibil ca plasând în noduri unităţi de un anumit tip să putem reprezenta / calcula o anumită funcţie, iar schimbând tipul unora dintre unităţi (sau al tuturor unităţilor), funcţia respectivă să nu mai potă fi calculată: ex. 4 vs. ex. 33.[17]

– **(P2) Orice unitate liniară situată pe un nivel ascuns poate fi „absorbită“ pe nivelul următor: ex. 32.**

– **(P3) Orice funcţie booleană poate fi reprezentată cu ajutorul unei reţele neuronale artificiale având doar două niveluri de perceptroni-prag: ex. 5.**

– **(P4) Orice funcţie definită pe un interval mărginit din $\mathbb{R}$, care este continuă în sens Lipschitz, poate fi aproximată oricât de bine cu ajutorul unei reţele neuronale care are un singur nivel ascuns: ex. 7.**

Algoritmi de antrenare a neuronilor artificiali folosind metoda gradientului descendent

– **algoritmul de antrenare a unităţii liniare: ex. 36;**
vedeţi T. Mitchell, *Machine Learning*, p. 93, justificare: p. 91-92; convergenţa: p. 95; exemplu de aplicare: ex. 10;
varianta incrementală a algoritmului de antrenare a unităţii liniare: cartea ML, p. 93-94; despre convergenţa acestei variante (ca aproximare a variantei precedente (“batch”): cartea ML, p. 93 jos;

– **algoritmul de antrenare a perceptronului-prag şi convergenţa: cartea ML, p. 88-89; exemplu de aplicare: ex. 11;**

– **algoritmul de antrenare a perceptronului sigmoidal şi justificarea sa teoretică: cartea ML, p. 95-97;**

– **algoritmul *Perceptron* al lui Rosenblatt; exemplu de aplicare: ex. 16, ex. 38;**

– **deducerea regulii de actualizare a ponderilor pentru tipuri particulare de perceptroni: ex. 12, ex. 24.a, ex. 37, 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 funcţii de cost / pierdere / penalizare (engl., loss function) decât semi-suma pătratelor erorilor: suma costurilor de tip log-sigmoidal, ex. 14 (pentru perceptronul liniar), o funcţie de tip cross-entropie, ex. 15 (pentru perceptronul sigmoidal).**

Perceptronul Rosenblatt şi rezultate de *convergenţă*

– **exemplu de aplicare [adică, învăţare cu perceptronul Rosenblatt]: ex. 16.**

– **câteva *proprietăţi* simple ale perceptronului Rosenblatt: ex. 17.**

[17] Problemele 1.d şi ex. 31 au în vedere o chestiune similară, însă pentru reţele cu topologii diferite: o anumită extensie a funcţiei XOR nu poate fi reprezentată pe reţele de neuroni-prag care au un singur nivel ascuns.

– **rezultate de convergenţă de tip "mistake bound" pentru [algoritmul de antrenare pentru] perceptronul-prag [în varianta] Rosenblatt: ex. 18, ex. 39;**
pentru perceptronul-prag (clasic): ex. 41;
învăţare online cu perceptronul-prag de tip Rosenblatt: ex. 40;

– ***Perceptronul* kernel-izat [dual]: ex. 23; particularizare pentru cazul nucleului RBF: ex. 49.**

Antrenarea reţelelor neuronale artificiale: algoritmul de *retro-propagare* pentru reţele feed-forward

– **T. Mitchell, *Machine Learning*, p. 98: pseudo-cod pentru reţele cu unităţi de tip sigmoidal, cu 2 niveluri, dintre care unul ascuns; pentru deducerea regulilor de actualizare a ponderilor; în cazul mai general al reţelelor feed-forward (de unităţi sigmoidale) cu oricâte niveluri, vedeţi p. 101-103;**

  ex. 19: deducerea regulilor de actualizare a ponderilor în cazul reţelelor cu 2 niveluri, având însă unităţi cu funcţie de activare oarecare (derivabilă);

– **aplicare: ex. 20, ex. 42, ex. 43;**

– **prevenirea overfitting-ului:**
folosirea unei componente de tip „moment" în expresia regulilor de actualizare a ponderilor: ex. 45;
regularizare: introducerea unei componente suplimentare în funcţia de optimizat: ex. 21;

– **cazul folosirii unei funcţii de activare de tip tangentă hiperbolică: ex. 44;**

– **cazul folosirii unei funcţii de cost / penalizare / eroare de tip cross-entropie: ex. 47;**

– **execuţia manuală a unei iteraţii a algoritmului de retro-propagare în cazul unei reţele neuronale simple, având un singur nivel ascuns, cu unităţi ce folosesc funcţia de activare ReL: ex. 48.**

Reţele neuronale profunde — câteva chestiuni introductive

– **analiza convexităţii unor funcţii de cost folosite în învăţarea profundă: ex. 54;**

– **fenomenul de „dispariţie" a gradientului [în cazul aplicării algoritmului de retro-propagare] pentru reţele neuronale profunde (engl., deep neural networks) care folosesc funcţia de activare sigmoidală: ex. 26;**

– **determinarea numărului de parametri şi de conexiuni din reţeaua neuronală convolutivă LeNet: ex. 27;**

– **determinarea mărimii hărţii de trăsături de pe un anumit nivel, precum şi a numărului de operaţii în virgulă mobilă (FLOPs) executate la procesarea forward într-o reţea neuronală convolutivă: ex. 55.**

9. Maşini cu vectori-suport

Sumar

Noţiuni preliminare

- elemente [simple] de *calcul vectorial*; proprietăţi elementare ale produsului scalar al vectorilor din $\mathbb{R}^n$, norma euclidiană (L_2) şi norma L_1 în $\mathbb{R}^n$: ex. 1, ex. 34;
- elemente [simple] de *geometrie analitică*: ecuaţia unei drepte din planul euclidian, ecuaţia unui plan din $\mathbb{R}^3$, ecuaţia unui hiper-plan din $\mathbb{R}^n$; ecuaţia 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;
- distanţa (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;
- proprietăţi de bază din *calculul matriceal*;
- calculul *derivatelor parţiale* [pentru funcţii obţinute prin compuneri de funcţii elementare];
- *metoda lui Lagrange* pentru rezolvarea problemelor de *optimizare convexă cu restricţii*:
  ex. 34,
  ex. din https://www.cs.helsinki.fi/u/jkivinen/teaching/sumale/Spring2014/kkt_example.pdf,
  ex. 56, ex. 57 de la capitolul de *Fundamente*;
- *separabilitate* liniară, separator optimal, *margine* geometrică: ex. 37, ex. 6.abc, ex. 9.

SVM cu margine "hard"

- (•) deducerea *formei primale* pentru problema SVM (cu margine "hard"), pornind de la principiul maximizarii marginii geometrice: ex. 2; [18]
- (P0) o formă [simplă] echivalentă cu *forma primală* a *problemei de optimizare SVM*: ex. 7;
- *exemplificarea* identificării *separatorului optimal* şi a *vectorilor-suport*, pornind de la condiţiile din forma primală: ex. 3-5, ex. 35, ex. 37, ex. 38 şi CMU, 2004 fall, T. Mitchell, Z. Bar-Joseph, HW4, ex. 4.1-4;
- calcularea *erorii* la cross-validare "leave-one-out" atunci când se foloseşte o SVM liniară cu margine "hard": ex. 4.d, ex. 36;
- exemple de [funcţii de] *mapare a atributelor*, cu scopul de a obţine 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] spaţiu de trăsături; identificarea separatorului neliniar din spaţiul iniţial [de trăsături]: ex. 9.de, ex. 39.ce, ex. 40.b-e;
- (•) deducerea *formei duale* pentru problema SVM cu margine "hard": ex. 10;

[18] Vedeţi şi Andrew Ng (Stanford), Lecture Notes, part V, section 3.

– ***exemplificarea*** **a două modalităţi de găsire a formei duale a problemei SVM cu margine "hard" pentru învăţarea unui concept [reprezentat de un set de date de antrenament neseparabil în spaţiul „iniţial" de trăsături] folosind o funcţie de mapare Φ dată: prin optimizare directă (ex. 11, ex. 41), respectiv prin folosirea relaţiilor de legătură cu soluţia problemei primale: ex. 12;**

– **(P1) efectul *multiplicării valorilor atributelor* cu o constantă pozitivă asupra separatorului obţinut de SVM): ex. 27.a;**[19]

– **vedeţi proprietăţile (P4) şi (P6) enunţate mai jos (la secţiunea despre C-SVM), care sunt valabile şi în cazul SVM [cu margine "hard"];**

– **(P2) efectul unui *atribut irelevant* — în sensul că nu afectează satisfacerea *restricţiilor* de separabilitate liniară a datelor de antrenament şi, în plus, nu măreşte marginea de separare — asupra rezultatelor clasificatorului SVM (şi respectiv C-SVM): ex. 48, ex. 56;**

– ***comparaţii*** **între SVM [având, eventual, diferite funcţii-nucleu] şi alţi clasificatori: ex. 56, ex. 44, ex. 45; vedeţi şi ex. 12 de la capitolul *Învăţare bazată pe memorare*.**

SVM cu margine "soft" (C-SVM):

– **(•) deducerea *formei duale* pentru problema SVM cu margine "soft" (C-SVM): ex. 13;**

– **(P3) exprimarea / calcularea distanţei 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*** **noţiunilor de bază: ex. 46, ex. 47 şi CMU, 2008 fall, Eric Xing, final, ex. 2.2;**
un *exemplu* de calculare a valorii optime pentru funcţia obiectiv a problemei de optimizare C-SVM: ex. 14.b;

– **exemplificarea poziţionării separatorului optimal determinat de C-SVM (pentru diferite valori ale parametrului C), în prezenţa unui outlier: ex. 16;**

– **exemplificarea efectului pe care îl are creşterea valorii parametrului de „destindere" C [asupra marginii şi asupra excepţiilor la clasificare]: ex. 17, CMU, 2010 fall, Aarti Singh, HW3, ex. 3.2;**

– **un exemplu de situaţie în care forma duală a problemei de optimizare C-SVM are soluţie unică, dar forma sa primală *nu* are soluţie unică: ex. 18;**

– **(P4) o proprietate pentru C-SVM (dar şi pentru SVM): ex. 15**
Dacă în setul de date de antrenament două trăsături (engl., features) sunt duplicate ($x_{ij} = x_{ik}$ pentru $i = 1, \ldots, m$), atunci ele vor primi ponderi identice ($\bar{w}_j = \bar{w}_k$) în soluţia 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;**

$$\text{err}_{train}(\text{C-SVM}) \leq \frac{1}{m} \sum_i \xi_i$$

[19] Rezultatul *nu* se menţine şi în cazul C-SVM: ex. 27.b.

– **(P6) o proprietate pentru C-SVM (dar şi pentru SVM):**
La CVLOO numai vectorii-suport pot fi (eventual!) clasificaţi eronat: ex. 20; aşadar avem [şi] o *margine superioară* pentru numărul de *erori* comise la *CVLOO* de către C-SVM:

$$\text{err}_{CVLOO}(\text{C-SVM}) \leq \frac{\#SVs}{m}$$

– **(P7) chestiuni legate de *complexitatea computaţională* 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 restricţii asupra variabilelor, dar în care se foloseşte funcţia de pierdere / cost *hinge*: ex. 21;**
exemplificare / aplicare (şi comparare cu regresia logistică): ex. 50;

– **(•) algoritmul SMO (Sequential Minimal Optimization): deducerea relaţiilor de actualizare a variabilelor Lagrange;**
***exemple* de aplicare a algoritmului SMO simplificat: ex. 23, ex. 51;**

– **o *comparaţie* asupra efectului *atributelor irelevante* (aici, în sensul că odată eliminate / adăugate, n-ar trebui să afecteze rezultatele clasificării) asupra clasificatorilor 1-NN şi C-SVM: ex. 56;**

SVM / C-SVM şi funcţiile-nucleu — câteva proprietăţi

– ***exemplificarea* corespondenţei dintre forma (primală sau duală) a problemei C-SVM şi alegerea valorii parametrului de „destindere" C şi a *funcţiei-nucleu* pe de o parte şi alura şi poziţionarea 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 poziţiei separatorului optimal (în raport cu *funcţia-nucleu* folosită): ex. 42;**

– **C-SVM: condiţii suficiente asupra parametrului de „destindere" C şi asupra valorilor funcţiei-nucleu pentru ca toate instanţele de antrenament să fie vectori-suport: ex. 25;**

– **SVM cu nucleu RBF: câteva proprietăţi remarcabile**

  ◦ **pentru SVM pe un set de date [separabil liniar în spaţiul de trăsături] instanţe foarte depărtate de separatorul optimal pot fi vectori-suport: ex. 43;**

  ◦ **(P8) pentru orice set de instanţe distincte şi pentru orice etichetare a acestora, există o valoare a hiper-parametrului nucleului RBF (σ) astfel încât SVM obţine la antrenare eroare 0: ex. 26. Rezultatul *nu* este valabil şi pentru C-SVM;**

  ◦ **(P9) pentru orice set de instanţe distincte, pentru orice etichetare a acestora şi pentru *orice* valoare a hiper-parametrului nucleului RBF (σ), problema de tip SVM care impune ca toate instanţele să fie corect clasificate şi la distanţa $1/\|w\|$ faţă de separatorul optimal are soluţie: ex. 58;**

– **avantaje şi dezavantaje ale folosirii metodelor de clasificare liniară de tipul SVM, Perceptron etc. şi versiunile lor kernel-izate: ex. 55.**

– **chestiuni recapitulative: ex. 28, ex. 27, ex. 66, ex. 57.**

Alte probleme [de optimizare] de tip SVM

- **SVM pentru clasificare n-ară (SVM multi-class): 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 soluţiile problemei *one-class SVM*, versiunea *Max Margin*, cu margine "hard" şi respectiv cele ale problemei SVM (cu şi 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 condiţie suficientă pentru ca variantele cu margine "hard" pentru cele două tipuri de probleme de optimizare *one-class SVM*, şi anume *Max Margin* şi *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 funcţie de cost / pierdere ε-*senzitivă*: ex. 33 (cu margine "hard"), ex. 64 (cu margine "soft" şi (echivalent) cu funcţie de cost ε-senzitivă); exemplificare / aplicare: ex. 63;**

– **teorema de reprezentare: ex. 65.**