Arbori de decizie: Privire de ansamblu

A. Noţiuni preliminare

– partiţie a unei mulţimi;

– entropie (definiţie: TM, pag. 57): pr. 2a, pr. 34a;

– entropie condiţională specifică: pr. 14a;

– entropie condiţională medie: pr. 2cd, ...;

– câştig de informaţie (definiţie: TM, pag. 58): pr. 2cd, pr. 34c, pr. 36a;

– arbori de decizie, văzuţi ca structură de date: pr. 1, pr. 6b, pr. 28
şi, respectiv, ca program în logica propoziţiilor: pr. 2e, pr. 33bc;
expresivitatea arborilor de decizie cu privire la funcţii boolene: pr. 29;

– spaţiu de versiuni pentru un concept (de învăţat): pr. 1, pr. 28, pr. 32;

– zone de decizie şi graniţe de separare/decizie pentru arbori de decizie cu variabile continue: pr. 9.

B. Algoritmul ID3 şi variante

B1. Algoritmul ID3

– pseudo-cod: TM, pag. 56;

– bias-ul inductiv: TM, pag. 63-64;

– exemple simple de aplicare: pr. 2, pr. 32, pr. 34;
în prezenţa „zgomotelor“: pr. 36;

– ID3 ca algoritm *per se*:

- de tip divide-et-impera, recursiv,
greedy $\Rightarrow$ nu garantează obţinerea soluţiei optime:
pr. 4, pr. 21a, pr. 33 (vs. pr. 32b);
- 1-step look-ahead
- complexitate de timp (vezi 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, m este numărul de exemple

– ID3 ca algoritm ML:

- consistent cu datele de antrenament (în absenţa „zgomotelor“): pr. 4
- algoritm de învăţare de tip "eager"
- analiza erorilor:
la antrenare: pr. 5, pr. 6a, pr. 9a, pr. 38;
la validare
la n-fold cross-validare
la cross-validare leave-one-out (CVLOO): pr. 9b, pr. 40bc;
- overfitting: pr. 9, pr. 21bc, pr. 40, pr. 57b.

B2. Extensii / variante ale algoritmului ID3

- **atribute cu valori continue: pr. 9, pr. 14c, pr. 40, pr. 41; ch. 4, pr. 11b;**
  3-way splitting (sau: n-way splitting): pr. 12, pr. 42
  alte variante de partiţionare: pr. 43
- **atribute discrete cu multe valori: pr. 13**
- **atribute cu valori nespecificate pentru unele instanţe;**
- **atribute cu diferite costuri asociate: pr. 14**
- **reducerea caracterului "eager" al învăţării: pr. 16**
- **reducerea caracterului "greedy" al învăţării:**
  IG cu "2-step look-ahead": pr. 17, pr. 18
  variante de tip "look-ahead" specifice atributelor continue: pr. 44
- **folosirea altor măsuri de impuritate în locul câştigului de informaţie:**
  Gini Impurity, Misclassification Impurity: pr. 15
- **reducerea overfitting-ului:**
  reduced-error pruning (folosind un set de date de validare):
  TM, pag. 69-71; A. Cornuéjols, L. Miclet, 2nd ed., pag. 418-421;
  rule post-pruning: TM, pag. 71-72;
  top-down vs. bottom-up pruning: pr. 19, pr. 45;
  pruning folosind testul statistic χ^2: pr. 20, pr. 46

C. Proprietăţi numerice / calitative ale arborelui ID3

- **eroarea la antrenare produsă de algoritmul ID3 pe orice set de date consistente este 0;**
- **arborele ID3 nu este neapărat optimal (ca nr. de noduri/niveluri): pr. 4, pr. 33**
- **influenţa atributelor identice şi, respectiv, a instanţelor multiple asupra arborelui ID3: pr. 7;**
- **o margine superioară pentru numărul de noduri frunză din arborele ID3, în funcţie de numărul de exemple şi de numărul de atribute: pr. 8;**
- **o margine superioară pentru eroarea la antrenare, în funcţie de numărul de valori ale variabilei de ieşire): pr. 6b;**
- **o margine superioară pentru adâncimea arborelui ID3 când atributele de intrare sunt continue, iar datele de antrenament sunt (ne)separabile liniar: pr. 10;**
- **o aproximare simplă a numărului de instanţe greşit clasificate din totalul de M instanţe care au fost asignate la un nod frunză, cu ajutorul entropei (H) nodului respectiv: pr. 37.**

D. Alte metode de învăţare automată bazate pe arbori

- **metode de învăţare automată de tip ansamblist bazate pe arbori de decizie:**
  boosting (AdaBoost): pr. 22 (convergenţa erorii de antrenare), pr. 23, 48, 49, 50, 51, 53 (aplicare), pr. 24 (AdaBoost ca algoritm de *optimizare secvenţială* în raport cu funcţia de cost / „pierdere" negativ-exponenţială), pr. 25 (marginea

de votare), pr. 26 (o condiţie suficientă pentru γ-slab învăţabilitate, bazată pe marginea de votare), pr. 52 (selecţarea trăsăturilor folosind AdaBoost; aplicare la clasificarea de documente)), pr. 54 (o proprietate interesantă: orice mulţime de instanţe distincte [şi etichetate] din $\mathbb{R}$ este γ-slab învăţabilă cu ajutorul compaşilor de decizie), pr. 55 (o variantă generalizată a algoritmului AdaBoost), pr. 22 şi 23 (adevărat / fals); Bagging, Random Forests;

– **arbori de regresie (CART).**

Clasificare bayesiană: Privire de ansamblu

A. Noţiuni preliminare (vezi [şi] cap. Probabilităţi şi statistică)

- **probabilităţi şi probabilităţi condiţionate;**
- **formula lui Bayes: pr. 4b;**
  cap. Probabilităţi şi statistică, pr. 6-7, 43-44;
- **independenţa [condiţională a] evenimentelor aleatoare:**
  cap. Probabilităţi şi statistică, pr. 5, 40, 41;
- **variabile aleatoare:**
  estimare de parametri în sensul verosimilităţii maxime (MLE): pr. 4a;
  (vezi şi cap. Estimarea probabilităţilor, pr. 5, 20)
- **distribuţii probabiliste corelate, marginale şi condiţionale: pr. 6, 8, 10, 27.**
  (vezi şi cap. Probabilităţi şi statistică, pr. 14, 15)
- **independenţa [condiţională a] variabilelor aleatoare: pr. 8, 10, 24, 27-33;**
  (vezi şi cap. Probabilităţi şi statistică, pr. 16, 53, 19, 20, 62)
- **ipoteze MAP vs. ipoteze ML:**
  formulare [ca soluţii la] probleme de optimizare: TM, pag. 156-157 (vezi pr. 13);
  exemplificare: pr. 2, 3, 21, 1, 32;
  exemplificare în cazul arborilor de decizie: pr. 12;
- **regresia logistică: vezi draftul capitolului suplimentar pentru cartea ML a lui T. Mitchell, *Generative and discriminative classifiers: Naive Bayes and logistic regression* (în special secţiunea 3);**
- **regresia logistică, chestiuni introductive: pr. 10 de la capitolul *Estimarea probabilităţilor*.**

B. Algoritmi

- **Algoritmul Bayes Naiv şi algoritmul Bayes Corelat:**[327]
  formulare ca probleme de optimizare: TM, pag. 167;
  pseudo-cod: vezi slide-uri;
  aplicare: pr. 4, 6, 7, 22, 23, 24;
- **aplicarea/adaptarea algoritmului Bayes Naiv pentru clasificare de texte:**[328]
  pr. 5, 25
  folosirea regulii "add-one" [a lui Laplace] pentru „netezirea" parametrilor: pr. 5, 26;
- **calculul ratei medii a erorilor pentru algoritmii Bayes Naiv şi Bayes Corelat: pr. 8, 9, 27, 28, 29, 33;**
- **zone de decizie şi graniţe de decizie (separatori):**
  evidenţierea grafică a „erorilor" clasificatorului Bayes Naiv în raport cu clasificatorul Bayes Corelat: pr. 10.

[327] La punctele B şi C considerăm (implicit) că toate variabilele de intrare sunt de tip Bernoulli sau, mai general, de tip categorial. La punctul D vom considera şi variabile de intrare de tip continuu, în genere de tip gaussian. Variabila de ieşire se consideră întotdeauna de tip Bernoulli/categorial.

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

C. Proprietăţi ale algoritmilor Bayes Naiv şi Bayes Corelat

– **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;**

– **numărul de parametri necesari de estimat din date: liniar pentru Bayes Naiv** $(2d+1)$ **şi exponenţial pentru Bayes Corelat** $(2^{d+1}-1)$**: pr. 7e, 23ab, 28;**

– **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]**

– **complexitatea de eşantionare: de ordin logaritmic pentru Bayes Naiv şi de ordin exponenţial pentru Bayes Corelat: pr. 11.**

– **(P0) echivalenţa regulei de decizie a algoritmului Bayes Naiv (când toate variabilele de intrare sunt de tip Bernoulli) regulei de decizie a *regresiei logistice* şi, în consecinţă, liniaritatea graniţelor de decizie.**

D. Algoritmii Bayes Naiv şi Bayes Corelat cu variabile de intrare de tip gaussian

Proprietăţi:

– **(P1) 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 (vezi pr. 34a, 15);**

Aplicare: G[N]B: 16, 38, G[J]B: 37, GNB vs G[J]B: 18.

E. Alte chestiuni

– **comparaţii între algoritmul Bayes Naiv şi alţi algoritmi de clasificare automată: pr. 31, 33;**

– **calculul [intervalului] „erorii reale" pornind de la „eroarea de eşantionare" (eroarea de test) pentru un clasificator oarecare: cap. Probabilităţi şi statistică, pr. 27.**

Învăţare bazată pe memorare: Privire de ansamblu

A. Noţiuni preliminare

– **măsuri de distanţă, măsuri de similaritate: pr. 2**

– **normă într-un spaţiu vectorial; [măsura de] distanţă indusă de către o normă: pr. 7**

B. Algoritmul k-NN şi variante

B1. Algoritmul k-NN

– **pseudo-cod: TM, pag. 232**

– **bias-ul inductiv: ... „Cine se aseamănă se adună“ (sau: „Spune-mi cu cine te împrieteneşti, ca să-ţi spun cine eşti“)**

– **exemple (simple) de aplicare: pr. 1, pr. 2**

– **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*****, pag. 163-173**

– k**-NN ca algoritm ML “lazy” (vs. “eager”):**
suprafeţe de decizie şi graniţe de decizie:
diagrame Voronoi pentru 1-NN: pr. 4, pr. 11, pr. 17, pr. 18, pr. 19;
Proprietate: suprafeţele de decizie şi graniţele de decizie depind de măsura de distanţă folosită: pr. 7

– **analiza erorilor:**

- **1-NN pe date consistente: eroarea la antrenare: 0**
- **variaţia numărului de erori (la testare şi respectiv testare) în funcţie de valorile lui** k**: pr. 20, pr. 21ab**
  k**-NN ca metodă ne-parametrică; alegerea lui** k**: CV: pr. 21c**
- **CVLOO: pr. 3, pr. 12, pr. 15, pr. 19**
- **sensibilitatea / robusteţea la „zgomote“: pr. 5**
- **eroarea asimptotică: pr. 10, pr. 23**

– **efectul trăsăturilor redundante sau irelevante**

– **„blestemul marilor dimensiuni“ (engl., the curse of dimensionality): pr. 9.**

B2. Variante ale algoritmului k-NN

– k**-NN folosind alte măsuri de distanţă (decât dist. euclidiană): pr. 7**

– k**-NN cu ponderarea distanţelor (engl., distance-weighted** k**-NN):**
TM, pag. 236-238 (form. 8.2, 8.3, 8.4)

– **algoritmul lui Shepard: pr.8**

C. Comparaţii cu alţi algoritmi

- **ID3: pr.11, pr. 13ab**
- **SVM: pr.12, pr. 13c**

D. Alte metode de tip IBL

- **regresie local-ponderată: TM, pag. 236-238**
- **reţele RBF: TM, pag. 238-240**
- **raţionare bazată pe cazuri (engl., case-based reasoning): TM, pag. 240-244**

Clusterizare: Privire de ansamblu

A. Noţiuni de bază

– **instanţă neetichetată vs. instanţă etichetată (exemplu de antrenament)**

– **învăţare nesupervizată (clusterizare) vs. învăţare supervizată (clasificare)**

– **cluster / grup / grupare / bin (engl.) vs. clasă**

– **[funcţie/măsură de] *distanţă* definită pe $\mathbb{R}^d \times \mathbb{R}^d$**

– **tipuri de clusterizare: ierarhică vs. neierarhică**

– **tipuri de ierarhii: ierarhii (arbori de clusterizare, dendrograme) obişnuite vs. ierarhii plate (engl., flat hierarchies);**
exemple: pr. 1a şi respectiv pr. 1b, pr. 6a

– **tipuri de apartenenţă a unei instanţe la un cluster: hard vs. soft (ultima numai pt. clusterizare ne-ierarhică)**

B. Clusterizare ierarhică

B1. 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":[329] $d(A,B) = \min\{d(x,y) | x \in A, y \in B\}$

"complete-linkage":[330] $d(A,B) = \max\{d(x,y) | x \in A, y \in B\}$

"average-linkage": $d(A,B) = \frac{1}{|A|\ |B|} \sum_{x\in A, y\in B} d(x,y)$.

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 şi orice alt cluster C;

– **[funcţie de] *coeziune* [internă] a unui cluster (sau: între elementele / instanţele dintr-un cluster);**
exemplu (pentru clustere non-singleton):

$$\mathrm{coh}(A) = \left(\frac{1}{\mathrm{C}^2_{|A|}} \sum_{x,y\in A} d(x,y)\right)^{-1} = \frac{\mathrm{C}^2_{|A|}}{\sum_{x,y\in A} d(x,y)}$$

[329] Sau: nearest-neighbour.
[330] Sau: furthest-neighbour.

B2. 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 (vezi pr. 3b);

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

  implementări: pr. 54, 31, 55.

- **Proprietăţi:**

  ◦ 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 (vezi pr. 5 şi 28);

  ◦ 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 (vezi pr. 4a); numărul minim de niveluri: $\lceil \log_2 n \rceil$ (vezi pr. 4b).

  ◦ există o anumită corespondenţă între clusterizare ierarhică cu similaritate de tip

  – "single-linkage" şi aflarea *arborelui [de acoperire] de cost minim* dintr-un graf; vezi pr. 6;

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

  ◦ algoritmul de clusterizare aglomerativă la al cărui pseudo-cod am făcut referire mai sus are complexitate $\mathcal{O}(n^3)$ (vezi pr. 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":
  dacă se foloseşte ca măsură de similaritate între 2 instanţe cosinusul unghiului dintre vectorii care 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 32.

- **Alte tipuri de măsuri de similaritate:**
  ***metrica lui Ward*: pr. 30**

C. Clusterizare neierarhică...

C1. ...folosind asignare "hard" a instanţelor la clustere

Noţiuni specifice

– **centroid (centru de greutate) al unui cluster,**
K-partiţie, K-configuraţie [iniţială] a centroizilor (pr. 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 (pr. 12).**

Algoritmul K-means

- **pseudo-cod (o versiune [mai] generală): Manning & Schütze, *op. cit.*, pag. 516; alternativ, vezi enunţul pr. 12 (sau, echivalent, volosind variabile-indicator: pr. 37);**
exemple de aplicare: pr. 7-11, 15a, 19a, 20a, 33, 34;

- **exemple de *euristici pentru iniţializarea centroizilor*:**
iniţializare arbitrară/random în $\mathbb{R}^d$ sau în $\{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^d$ (setul de date de clusterizat);
aplicare în prealabil a unui algoritm de clusterizare ierarhică;
K-means++ (David Arthur, Sergei Vassilvitskii, 2007).

- ***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*:**
***algoritm iterativ*: pleacă de la o soluţie (K-partiţie) aleasă eventual în mod arbitrar/aleatoriu şi o „îmbunătăţeşte" la fiecare iteraţie;**
soluţia găsită este dependentă de iniţializarea centroizilor (vezi pr. 10);
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 (vezi pr. 12b);
K-means poate fi văzut şi ca *algoritm de optimizare* — vezi criteriul J_K de mai sus;
explorează doar parţial spaţiul de căutare [a minimului criteriului J_K];
algoritmul K-means nu garantează atingerea optimului global (i.e., minimul) criteriului J_K (vezi pr. 12b, 38b);
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 $J_K(C^t, \mu^t)$ (vezi pr. 12a).

- **ca algoritm de învăţare automată:**
  [urmat de] „generalizare“: o instanţă nouă x se asociază clusterului având centroidul cel mai apropiat de x;
  „graniţele“ de separare dintre [perechile de] clustere produse de K-means sunt [doar] liniare, [cel puţin] atunci când se foloseşte distanţa euclidiană (vezi pr. 12.b);
  este însă posibil să se obţină separatori neliniari dacă se foloseşte o versiune „kernelizată“ a algoritmului K-means (vezi pr. 39);
  ca *euristică* pentru alegerea unei valori convenabile pentru K, vezi CMU, 2012 fall, E. Xing, A. Singh, HW3, pr. 1.
- **implementare: pr. 56.**

Proprietăţi

- **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, $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^d$:**
  ∘ valoarea 0 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 $i = 1, \ldots, n$ (pr. 36);
  ∘ 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 (vezi pr. 12a);
  ∘ $\underline{J}_1 \geq \underline{J}_2 \geq \ldots \geq \underline{J}_{n-1} \geq \underline{J}_n = 0$ (vezi pr. 13).
- **în legătură cu J_K şi algoritmul K-means:**
  ∘ $J_K(C^{t-1}, \mu^{t-1}) \geq J_K(C^t, \mu^t)$ la orice iteraţie ($t > 0$) a algoritmului K-means (vezi pr. 12a);
  ∘ 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;
  ∘ în vreme ce minimizează *coeziunea intra-clustere*, i.e. o sumă 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 aparţine clusterului de centroid μ_k şi $\gamma_{ik} = 0$ în caz contrar, algoritmul K-means maximizează (în mod aproximativ!) o măsură de distanţă între 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 $x_1, x_2, \ldots, x_n$ (pr. 37).
- **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“: $C_1 = \{x_1, \ldots, x_{i_1}\}$, $C_2 = \{x_{i_1+1}, \ldots, x_{i_2}\}$, ..., $C_{K-1} = \{x_{i_{K-1}+1}, \ldots, x_n\}$, cu $i_1 < i_2 < \ldots < i_{K-1} < i_K = n$;
  ∘ există un algoritm de complexitate $\mathcal{O}(Kn^2)$ care calculează $\underline{J}_K$ (vezi pr. 38).

- **algoritmul K-means poate fi „kernelizat“ (vezi pr. 39);**
  în consecinţă, putem obţine drept graniţe de separare între clustere [şi] suprafeţe non-liniare (spre deosebire de versiunea ne-kernelizată a algoritmului K-means, unde graniţele sunt doar liniare).

C2. ...folosind asignare “soft” a instanţelor la clustere

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 (vezi pr. 24 de la capitolul de *Probabilităţi şi statistică*);**
- **distribuţie (funcţie de densitate) de probabilitate (p.d.f.);**
  parametri ai unei distribuţii;
  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: pr. 56 de la capitolul *Probabiltăţi şi statistică*; o mixtură de distribuţii gaussiene: pr 16 de la capitolul *Clasificare bayesiană*.
- **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: pr. 16c; pr. 1 de la cap. *Estimarea probabiltăţilor***
- **MLE (Maximum Likelihood Estimation): estimarea [valorilor] parametrilor unei distribuţii probabiliste în sensul maximizării verosimilităţii datelor disponibile. Exemplificare: cap. *Estimarea probabilităţilor*, pr. 1-7, 11-21.**
- **Observaţie: Algoritmul EM este 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*.**

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

- **pseudo-cod:**
  cazul uni-dimensional, varianta când doar parametrul μ este lăsat liber: *Machine Learning*, Tom Mitchell, 1997, pag. 193; aplicare: pr. 16;
  cazul uni-dimensional, varianta când toţi parametrii (π, μ şi σ) sunt lăsaţi liberi: pr. 17; alte variante: pr. 43, 44;
  cazul multi-dimensional, varianta când toţi parametrii (π, μ şi Σ) sunt lăsaţi liberi: pr. 23;

– **schema algoritmică EM: *ML* book, pag. 195;**

– **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*:**
  la fiecare iteraţie t se calculează o funcţie „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 Q_t se calculează în funcţie de distribuţiile condiţionale ale variabilelor „neobservabile“ în raport cu datele observabile şi cu $\theta^{(t)}$;
  se poate demonstra că funcţia Q_t constituie o margine inferioară pentru funcţia de log-verosimilitate a variabilelor „observabile“, $\log P(X|\theta)$ (vezi pr. 1 de la capitolul *Algoritmul EM*);
  teorema de corectitudine / convergenţă (vezi problemele 1 şi în special 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;
  metoda de căutare a optimului / maximului funcţiei $\log P(X|\theta)$ este “coordinate ascent” (căutare pe coordonate, în mod alternant);

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

– **ca algoritm de *învăţare automată*:**
  algoritmul EM este o metodă de identificare/învăţare de ipoteze ML (Maximum Likelihood); vezi 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 $\max_{j'} P(X = x|h_{j'})P(h_{j'})$;
  spre deosebire de cazul algoritmului K-means, suprafeţele / graniţele de separare create de algoritmul EM/GMM nu sunt în mod neapărat liniare (vezi de exemplu o situaţie întâlnită la rezolvarea pr. 15.c, pag. 358, sau pr. 51.c).

– **comparativ cu algoritmul K-means:**
  algoritmul EM/GMM este în general mai lent — mişcarea centroizilor poate explora într-o manieră mai fină spaţiul (vezi de exemplu pr. 19) —, iar din acest motiv poate să obţină uneori rezultate mai bune / convenabile (vezi spre exemplu pr. 20), şi este mai robust la influenţa outlier-elor;
  apare un fenomen de “atracţie” reciprocă a mediilor gaussienelor (aceste medii fiind echivalentul centroizilor din algoritmul K-means), dotorită faptului că fiecare instanţă aparţine (cu o anumită probabilitate) la fiecare cluster (vezi spre exemplu pr. 15.b).

– ***schema algoritmică EM*** **(vezi Tom Mitchell, *Machine Learning* book, 1997, pag. 195) are diverse variante/aplicaţii:**
  • calculul parametrilor pentru mixturi de diverse distribuţii [nu doar gaussiene]: vezi 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 distribuţii gaussiene multi-variate:**
◦ dacă matricea de covarianţă Σ este diagonală, atunci distribuţia gaussiană respectivă este echivalentă cu un set/vector de variabile gaussiene uni-variate independente (vezi pr. 25 de la capitolul *Probabilităţi şi statistică*);
◦ 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;
◦ 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;
◦ 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.

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

– **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 (vezi pr. 47);

– **O legătură interesantă între clasificatorul Bayes Naiv gaussian şi algoritmul EM/GMM când matricele de covarianţă sunt diagonale:**
o variantă semi-supervizată a algoritmului EM/GMM. (pr. 53).

Algoritmul EM: Privire de ansamblu

Noţiuni preliminare

- estimarea parametrilor unei distribuţii probabiliste în sensul verosimilităţii maxime (MLE) respectiv în sensul probabilităţii maxime a posteriori (MAP): vezi cap. *Probabilităţi şi statistică*;
- tipuri/clase de distribuţii probabiliste vezi cap. *Probabilităţi şi statistică*;
- mixturi de distribuţii probabiliste vezi cap. *Probabilităţi şi statistică* şi cap. *Clusterizare*;
- metoda "coordinate ascent" pentru rezolvarea problemelor de optimizare: pr. 1;
- metoda multiplicatorilor lui Lagrange pentru rezolvarea problemelor de optimizare cu restricţii: pr. 5, 7 şi 15.

Schema algoritmică EM

- pseudo-cod: *Machine Learning*, Tom Mitchell, 1997, pag. 194-195;
- fundamentare teoretică: pr. 1 şi 2;
- chestiuni metodologice (relativ la iniţializarea parametrilor): pr. 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 (pr. 14, 4), mixturi [de distribuţii] categoriale (pr. 5, 15)mixturi [de distribuţii] Poisson (pr. 18), mixturi [de distribuţii] Gamma (pr. 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": pr. 3. Similar pentru distribuţia multinomială: pr. 13;
- EM pentru estimarea tuturor parametrilor unei distribuţii categoriale: pr. 12;
- EM pentru estimarea parametrului unei distribuţii Poisson în condiţiile în care o parte din valorile date lipsesc (pr. 10);
- EM pentru estimarea parametrilor a doă distribuţii probabiliste atunci când se dau instanţe care sunt generate de suma celor două distribuţii: distribuţii exponenţiale (pr. 8), sau distribuţii gaussiene (pr. 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): pr. 7 (varianta de asignare "soft" a instanţelor la cluster) şi pr. 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: pr. 9;**
- **EM pentru modelul [mixturii] domeniilor semantice (engl., topic model) pentru clusterizare de documente: pr. [6 şi] 16.**
- **EM pentru estimarea [nu în sens MLE, cum a fost cazul până aici, ci] în sens MAP: pr. 20.**

Reţele neuronale artificiale: Privire de ansamblu

A1. 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 $(\cdot)$;**
- **metoda gradientului descendent (ca metoda de optimizare); avantaje şi dezavantaje; pr. 10, 23, 35, 36.**

A2. [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 conxiunilor 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).

A3. Câteva *proprietăţi* de *expresivitate* ale 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 (vezi pr. 8).

- **(P1) Reţele de neuroni diferite (ca structură şi / sau tipuri de unităţi) pot să calculeze o aceeaşi funcţie. (Vezi pr. 3 şi pr. 1.c vs. pr. 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. (Vezi pr. 4 vs. pr. 34.[331])

- **(P2) Orice unitate liniară situată pe un nivel ascuns poate fi „absorbită“ pe nivelul următor (pr. 33).**
- **(P3) Orice funcţie booleană poate fi reprezentată cu ajutorul unei reţele neuronale artificiale având doar două niveluri de perceptroni-prag (pr. 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 (pr. 7).**

B. Algoritmi de antrenare a neuronilor artificiali folosind metoda gradientului descendent

- **algoritmul de antrenare a unităţii liniare: pr. 37**
  vezi T. Mitchell, *Machine Learning*, p. 93, justificare: p. 91-92; convergenţa: p. 95; exemplu de aplicare: pr. 11
  varianta incrementală a algoritmului de antrenare a unităţii liniare: TM, ML book, p. 93-94; despre convergenţa acestei variante (ca aproximare a variantei precedente (“batch”): TM, ML book, p. 93 jos;
- **algoritmul de antrenare a perceptronului-prag şi convergenţa: TM, ML book, p. 88-89; exemplu de aplicare: pr. 12;**
- **algoritmul de antrenare a perceptronului sigmoidal şi justificarea sa teoretică: TM, ML book, p. 95-97;**
- **algoritmul *Perceptron* (!) al lui Rosenblatt; exemplu de aplicare: 17, 39;**
- **deducerea regulii de actualizare a ponderilor pentru tipuri particulare de perceptroni: pr. 13, 25.a, 38, 14.a;**
- **o justificare probabilistă (gen ipoteză de tip *maximum likelihood*) pentru minimizarea sumei pătratelor erorilor [la deducerea regulii de antrenare] pentru perceptronul liniar: pr 14.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, pr. 15 (pentru perceptronul liniar), o funcţie de tip cross-entropie, pr. 16 (pentru perceptronul sigmoidal).**

B′. Perceptronul Rosenblatt şi rezultate de convergenţă

- **exemplu de aplicare [adică, învăţare cu perceptronul Rosenblatt]: pr. 17.**

[331] Problemele 1.d şi pr. 32 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.

– **câteva *proprietăţi* simple ale perceptronului Rosenblatt: pr. 18.**

– **rezultate de convergenţă de tip "mistake bound" pentru [algoritmul de antrenare pentru] perceptronul-prag [în varianta] Rosenblatt: pr. 19, 40;**
pentru perceptronul-prag (clasic): pr. 42;
învăţare online cu perceptronul-prag de tip Rosenblatt: pr. 41;

– ***Perceptronul* kernelizat [dual]: pr. 24; particularizare pentru cazul nucleului RBF: pr. 50.**

C. Antrenarea reţelelor neuronale artificiale: algoritmul de retro-propagare pentru reţelel 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 reţelelor al feed-forward (de unităţi sigmoidale) cu oricâte niveluri, vezi p. 101-103);**

pr. 20: deducerea regulilor de actualizare a ponderilor în cazul reţelelor cu 2 niveluri, având însă unităţi cu funcţie de activare oarecare (derivabilă);

– **aplicare: pr. 21, 43, 44;**

– **prevenirea overfitting-ului:**
folosirea unei componente de tip „moment" în expresia regulilor de actualizare a ponderilor: pr. 46;
regularizare: introducerea unei componente suplimentare în funţia de optimizat: pr. 22;

– **cazul folosirii unei funcţii de activare de tip tangentă hiperbolică: pr. 45;**

– **cazul folosirii unei funcţii de cost/penalizare/eroare de tip cross-entropie: pr. 48;**

– **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: pr. 27;**
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: pr. 49.

D. Reţele neuronale profunde:

chestiuni introductive: pr. 28 şi pr. 55.