LAPI – Laboratorul de Analiza şi Prelucrarea Imaginilor

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Universitatea POLITEHNICA din Bucureşti

Tehnici de analiză și clasificare automată a informației

Conf. dr. ing. Bogdan IONESCU
http://imag.pub.ro/~bionescu

Bucureşti, 2015

Plan Curs

M1. Introducere (concept, aplicații)

M2. Prelucrarea și reprezentarea datelor de intrare

M3. Tehnici de clasificare ne-supervizată (“clustering”)

M4. Tehnici de clasificare supervizată (“classification”)

M5. Evaluarea performanței clasificatorilor

> M4. Tehnici de clasificare supervizată (“classification”)

4.1. [Introducere]

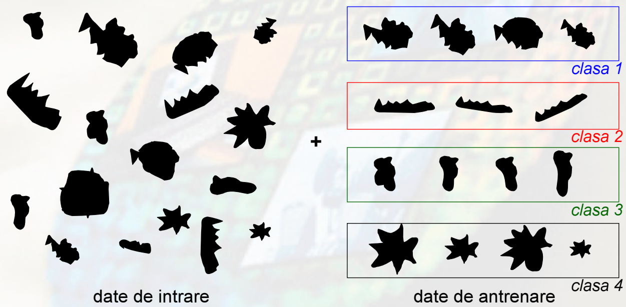
4.2. [k-NN]

4.3. [Support Vector Machines]

4.4. [Arbori de decizie]

Clasificare supervizată (classification) - principiu

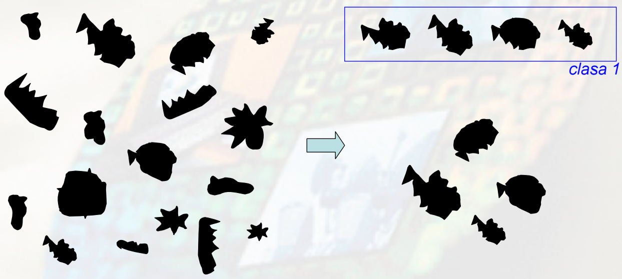
classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);



date de intrare date de antrenare

Clasificare supervizată (classification) – principiu (cont.)

classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);



date de intrare partiționare

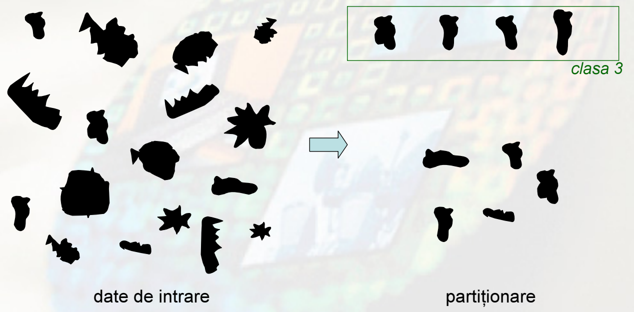
Clasificare supervizată (classification) – principiu (cont.)

classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);

date de intrare partiționare

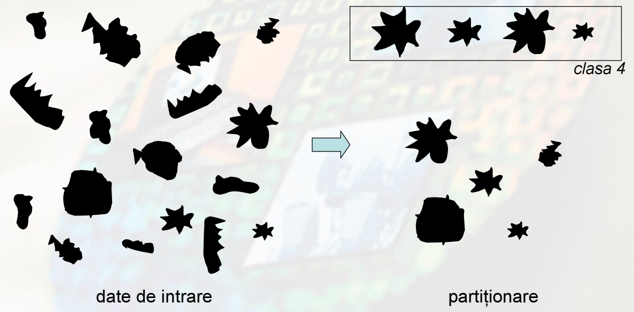
Clasificare supervizată (classification) - principiu (cont.)

classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);



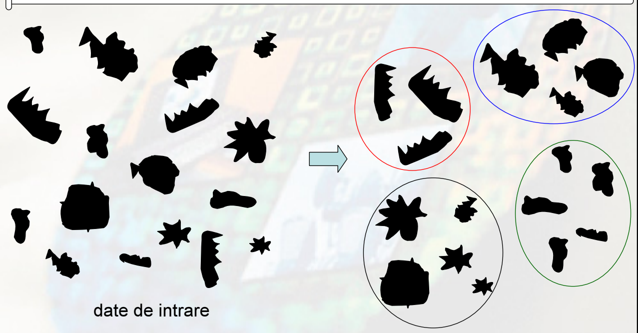
Clasificare supervizată (classification) - principiu (cont.)

classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);



Clasificare supervizată (classification) - principiu (cont.)

classification = partiționarea datelor de intrare în mulțimi similare pe baza unor exemple a priori de astfel de partiții (date de antrenare);



k-NN

datele de intrare sunt clasificate pe baza unui vot majoritar cu privire la clasa de apartenență a celor mai apropiați k vecini;

> date de intrare:

- date de antrenare:
$[Y_1 \in c_1, Y_2 \in c_2, \ldots, Y_m \in c_m]$, $Y_j=[y_{j,1},\ldots,y_{j,n}]$, $c_j \in \{1,\ldots,C\}$

- date de clasificat:
$X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$;

> algoritm:

p1. se alege o valoare pentru k (ex. 1, 3, 5 etc);

> antrenare:
p2. sunt stocate datele etichetate (lazy classifier);

k-NN (cont.)

> algoritm (cont.):

> clasificare:
p3. pentru fiecare instanță de clasificat, X_i, se calculează distanța către toate datele de antrenare, Y_j, $j=1,\ldots,m$;

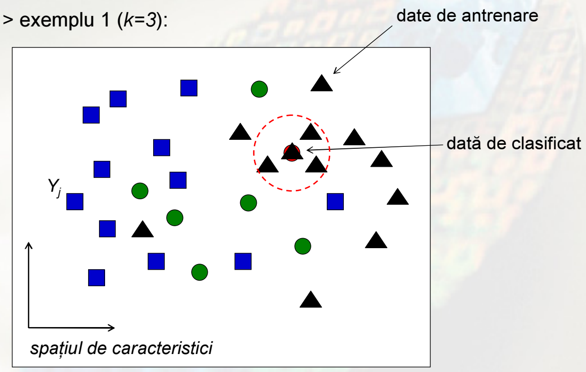
p4. se determină cele mai apropiate k date de antrenare;

p5. instanța X_i este clasificată ca aparținând clasei predominante din k date deja etichetate (*vot majoritar*);

p6. procesul se repetă până când sunt clasificate toate instanțele de intrare.

k-NN (cont.)

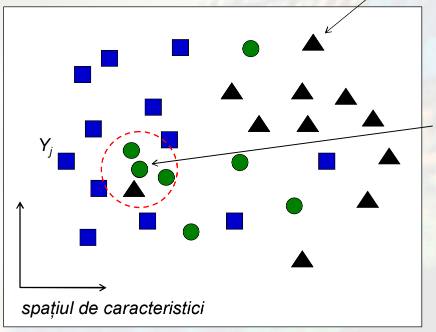
> exemplu 1 (k=3):



k-NN (cont.)

> exemplu 1 (*k=3*; cont.):

date de antrenare

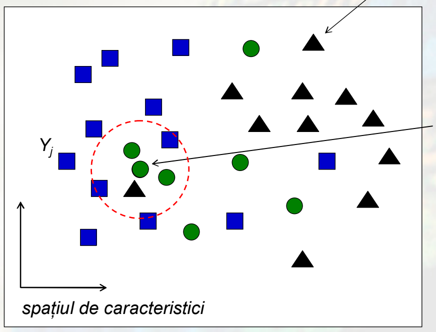


dată de clasificat

k-NN (cont.)

> exemplu 2 (*k=5*):

date de antrenare



dată de clasificat

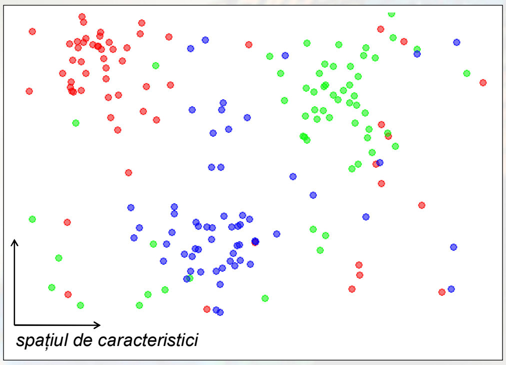
> Obs.: număr egal pentru albastru și verde?

[sursă imagine Wikipedia]

k-NN (cont.)

> exemplu 3:

date de antrenare

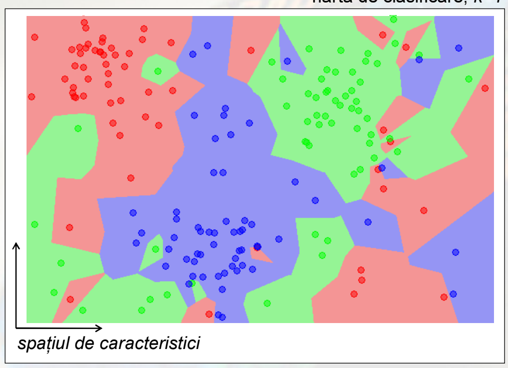


[sursă imagine Wikipedia]

k-NN (cont.)

> exemplu 3 (*cont.*):

harta de clasificare, *k=1*

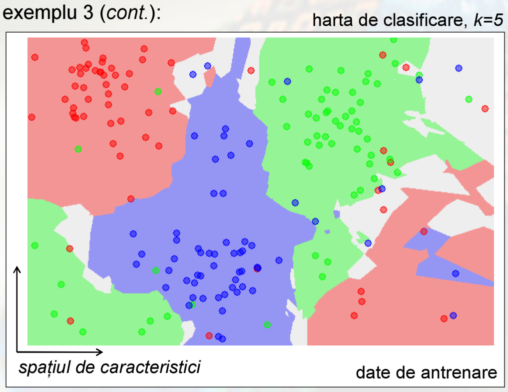


[sursă imagine Wikipedia]

k-NN (cont.)

> exemplu 3 (*cont.*):

harta de clasificare, *k=5*



date de antrenare

k-NN (cont.)

> alegere valoare *k*;

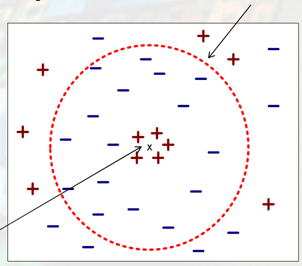
- *ce se întâmplă dacă k este prea mic?*
[alegerea clasei este sensibilă la "zgomot" – clase ne-majoritare pot influența decizia]

- *ce se întâmplă dacă k este prea mare?*
[alegerea clasei poate fi influențată de o altă clasă predominantă dar care este la o distanță semnificativă]

- "rule of thumb"

$$k < \sqrt{m}$$

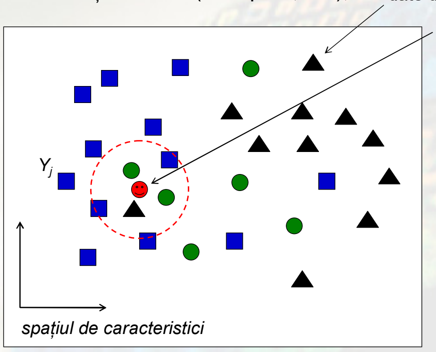
vecinătate



data de clasificat

k-NN (cont.)

> îmbunătățire decizie (exemplu 2, k=5);



> număr egal de clase "majoritare" ?

> *îmbunătățire*: ponderarea contribuției vecinilor la votul majoritar:

$$w = \frac{1}{d(X_i, Y_j)^2}$$

k-NN (cont.)

> avantaje:

- poate fi aplicat datelor de orice tip de distribuție (ex. nu neapărat linear separabile);
- simplu și intuitiv;
- eficient când numărul de date de antrenare este foarte mare.

> dezavantaje:

- alegerea valorii lui *k*;
- complexitate matematică ridicată - nu există o etapă preliminară de antrenare (care poate fi realizată offline);
- pentru o bună performanță necesită un număr semnificativ de exemple (avantaj și dezavantaj).

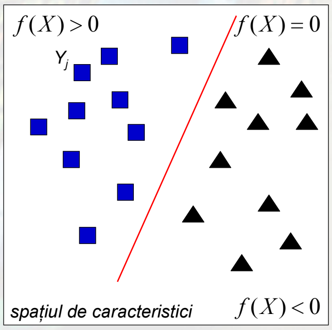
Support Vector Machines

Datele de intrare sunt împărțite în două clase prin optimizarea ecuației unui hiperplan astfel încât distanța la date este maximă;

> date de intrare:

- date de antrenare: $[Y_1 \in c_1, \ldots, Y_m \in c_m]$, $c_j \in \{+1,-1\}$, $Y_j=[y_{j,1},\ldots,y_{j,n}]$;
- date de clasificat: $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$;
- *clasificator liniar*: se determină o funcție liniară:

$$f(X) = \begin{cases} >0 & clasa(+1) \\ <0 & clasa(-1) \end{cases}$$



Support Vector Machines (cont.)

- *clasificator liniar*: se determină o funcție liniară (cont.):

$$f(X) = \begin{cases} >0 & clasa(+1) \\ <0 & clasa(-1) \end{cases}$$

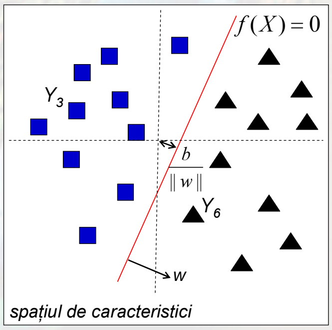
- funcția reprezintă ecuația unui hiperplan:

$$f(X) = w^T \cdot X + b$$

unde *w* și *b* reprezintă vectorul normal la hiperplan și respectiv decalajul față de origine;

$$f(Y_3) = w^T \cdot Y_3 + b > 0$$

$$f(Y_6) = w^T \cdot Y_6 + b < 0$$



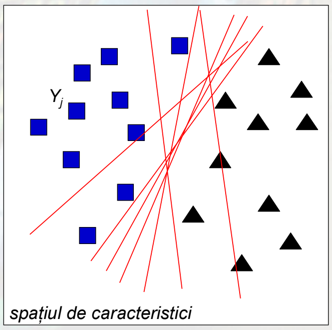
Support Vector Machines (cont.)

- *formulare clasificator*: având la dispoziție datele de antrenare, să se determine parametrii *w* și *b* ai hiperplanului care separă cel mai bine datele;
- clasificarea datelor noi se face prin:

$$f'(X_i) = \mathrm{sgn}(w^T \cdot X_i + b)$$

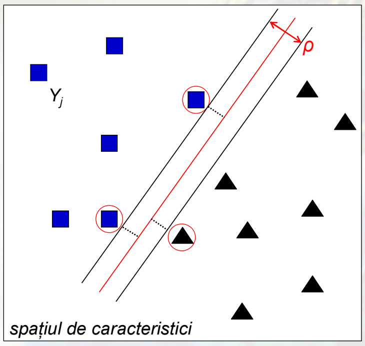
$$\mathrm{sgn}(x) = \begin{cases} 1 & x>0 \\ -1 & x<0 \end{cases}$$

- *cum determinăm hiperplanul care separă cel mai bine datele?*



Support Vector Machines (cont.)

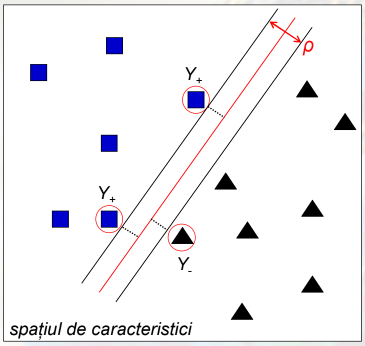
- alegere hiperplan;



- datele de antrenare cele mai apropiate de hiperplan se numesc *vectori suport*;
- distanța dintre vectorii suport definesc o margine *ρ*;
- *soluție*: separarea datelor se face cu hiperplanul ce **maximizează** pe *ρ*;

Support Vector Machines (cont.)

- alegere hiperplan (cont.);



- dacă:

$$w^T \cdot X + b = 0$$

atunci și:

$$c(w^T \cdot X + b) = 0$$

astfel, putem normaliza valorile astfel încât:

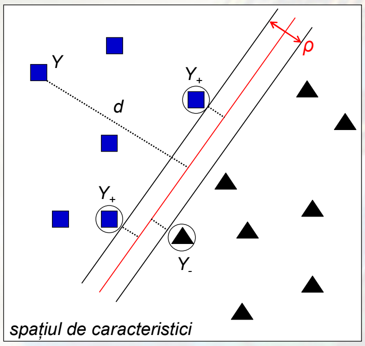
$$w^T \cdot Y_+ + b = +1$$

$$w^T \cdot Y_- + b = -1$$

unde Y_+ și respectiv Y_- sunt vectorii suport din planul + și respectiv -;

Support Vector Machines (cont.)

- alegere hiperplan (cont.);



- cum determinăm valoarea lui ρ?

$$\rho = \frac{w^T \cdot Y_+ + b}{\| w \|} - \frac{w^T \cdot Y_- + b}{\| w \|} = \frac{2}{\| w \|}$$

- care este distanța de la un vector Y la hiperplan?

$$d = \frac{w^T \cdot Y + b}{\| w \|}$$

Support Vector Machines (cont.)

- clasificatorul rezultat (formulare matematică):

  - având *m* date de antrenare, $\{(Y_j, c_j)\}$, cu $j=1,\ldots,m$, $Y_j=[y_{j,1},\ldots,y_{j,n}]$ și $c_j \epsilon \{-1;+1\}$, acestea sunt separate de un hiperplan de margine ρ;

  - pentru fiecare set $\{(Y_j, c_j)\}$, avem:

$$w^T \cdot Y_j + b \le -\frac{\rho}{2} \quad \text{dacă } c_j = -1$$

$$w^T \cdot Y_j + b \ge \frac{\rho}{2} \quad \text{dacă } c_j = +1$$

$$\Rightarrow c_j(w^T \cdot Y_j + b) \ge \frac{\rho}{2}$$

Support Vector Machines (cont.)

- clasificatorul rezultat (formulare matematică; cont.):

  - pentru fiecare *vector suport*, Y_j^s, inegalitate devine egalitate:

$$c_j(w^T \cdot Y_j^s + b) = \frac{\rho}{2}$$

  - astfel, distanța de la Y_j^s la hiperplan devine:

$$d = \frac{c_j(w^T \cdot Y_j^s + b)}{\| w \|} = \frac{1}{\| w \|}$$

  - deci marginea ρ este:

$$\rho = \frac{2}{\| w \|}$$

Support Vector Machines (cont.)

$$\rho = \frac{2}{\| w \|}$$

- clasificatorul rezultat (formulare matematică; cont.):

  - în aceste condiții antrenarea clasificatorului poate fi formulată ca:

> să se determine *w* și *b* astfel încât să fie **maximizat** ρ, cu condiția că pentru toate datele de antrenare $\{(Y_j, c_j)\}$: $c_j(w^T \cdot Y_j + b) \ge 1$

normalizare la *ρ/2*

  - și mai departe reformulată ca (minimizare):

> să se determine *w* și *b* astfel încât să fie **minimizat** $\|w\|^2=w^Tw$, cu condiția că pentru toate $\{(Y_j, c_j)\}$: $c_j(w^T \cdot Y_j + b) \ge 1$

normalizare la *ρ/2*

= o problemă de optimizare pătratică (bine studiată în literatură);

Support Vector Machines (cont.)

- clasificatorul rezultat (formulare matematică; cont.):

> să se determine *w* și *b* astfel încât să fie **minimizat** $\|w\|^2=w^Tw$, cu condiția că pentru toate $\{(Y_j, c_j)\}$: $c_j(w^T \cdot Y_j + b) \ge 1$

normalizare la *ρ/2*

- soluție folosind multiplicatorii Lagrange:

> să se determine $\alpha_1,\ldots,\alpha_m$ astfel încât să maximizăm:
>
> $$\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j c_i c_j Y_i^T Y_j$$
>
> cu următoarele ipoteze:
>
> (1) $\sum_j \alpha_j c_j = 0$
>
> (2) $\alpha_i \ge 0$ pentru $\forall \alpha_i$

Support Vector Machines (cont.)

- clasificatorul rezultat (formulare matematică; cont.):

să se determine $\alpha_1,\ldots,\alpha_m$ astfel încât să maximizăm:

$$\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j c_i c_j Y_i^T Y_j$$

cu următoarele ipoteze:

(1) $\sum_j \alpha_j c_j = 0$

(2) $\alpha_i \geq 0$ pentru $\forall \alpha_i$

$$\Rightarrow w = \sum_j \alpha_j c_j Y_j$$

$$\Rightarrow b = c_k - \sum_j \alpha_j c_j Y_j^T Y_k \quad \forall \alpha_k > 0$$

Support Vector Machines (cont.)

- SVM liniar "hard margin":

$$w = \sum_j \alpha_j c_j Y_j$$

$$b = c_k - \sum_j \alpha_j c_j Y_j^T Y_k \quad \forall \alpha_k > 0$$

- fiecare valoare α non-nulă indică un vector suport;

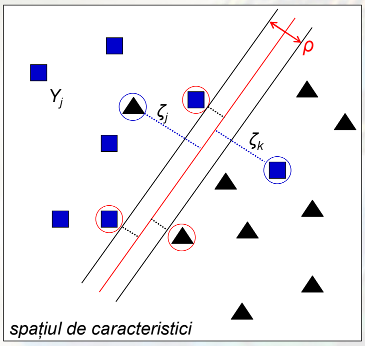
- clasificatorul este dat de:

$$f_{linSVM}(X_i) = \sum_j \alpha_j c_j Y_j^T X_i + b$$

unde $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$, sunt datele de clasificat;

Support Vector Machines (cont.)

- SVM "soft margin":



- ce putem face în această situație?

- o variantă este aceea de a adauga un set de variabile (soft) care să-mi permită să reduc clasificările greșite, dificile sau afectate de zgomot = "soft margin" SVM;

Support Vector Machines (cont.)

- SVM "soft margin" (cont.):

- formulare "hard margin":

să se determine w și b astfel încât să fie **minimizat** $||w||^2=w^Tw$, cu condiția că pentru toate $\{(Y_j,c_j)\}$: $c_j(w^T\cdot Y_j + b) \geq 1$

normalizare la $\rho/2$

- formulare "soft margin":

să se determine w și b astfel încât să fie **minimizat** $w^T w + C\sum_j \zeta_j$

cu condiția că pentru toate $\{(Y_j,c_j)\}$: $c_j(w^T\cdot Y_j + b) \geq 1-\zeta_j, \zeta_j \geq 0$

normalizare la $\rho/2$

- parametrul C poate fi văzut ca o modalitate de a controla adaptarea excesivă la datele de antrenare ("overfitting"): compromis între maximizare margine și adaptare la date.

Support Vector Machines (cont.)

- SVM "soft margin" (cont.):

- soluție folosind multiplicatorii Lagrange (aceeași formulare):

să se determine $\alpha_1,\ldots,\alpha_m$ astfel încât să maximizăm:

$$\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j c_i c_j Y_i^T Y_j$$

cu următoarele ipoteze:

(1) $\sum_j \alpha_j c_j = 0$

(2) $\alpha_i \geq 0$ pentru $\forall \alpha_i$

$$\Rightarrow w = \sum_j \alpha_j c_j Y_j$$

$$\Rightarrow b = c_k(1-\zeta_k) - \sum_j \alpha_j c_j Y_j^T Y_k \quad \forall \alpha_k > 0$$

Support Vector Machines (cont.)

- SVM "soft margin" (cont.):

$$w = \sum_j \alpha_j c_j Y_j$$

$$b = c_k(1-\zeta_k) - \sum_j \alpha_j c_j Y_j^T Y_k \quad \forall \alpha_k > 0$$

- fiecare valoare α non-nulă indică un vector suport;

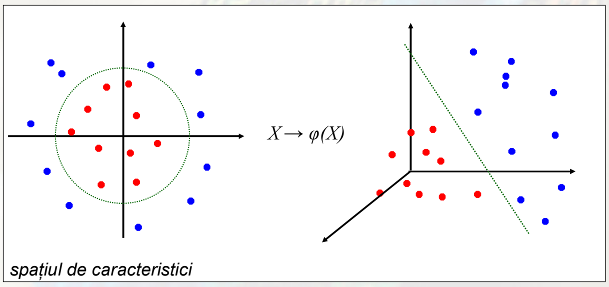
- clasificatorul este dat de:

$$f_{softSVM}(X_i) = \sum_j \alpha_j c_j Y_j^T X_i + b$$

unde $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$, sunt datele de clasificat;

Support Vector Machines (cont.)

- *ce se întâmplă dacă datele nu sunt liniar separabile?*
  [transformare a spațiului astfel încât separabilitatea să fie posibilă]



[sursă Machine Learning Group, Univ. of Texas]

Support Vector Machines (cont.)

- abordare neliniară;

$$\boxed{f_{linSVM}(X_i) = \sum_j \alpha_j c_j Y_j^T X_i + b}$$

unde $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$, sunt datele de clasificat;

produse X^TX

$$b = c_k - \sum_j \alpha_j c_j Y_j^T Y_k$$

- "*kernel trick*" (vezi și M3, k-means) - produsele X^TX sunt transformate printr-o funcție neliniară:

$$K(Y_j,Y_k) = Y_j^T Y_k \quad \Rightarrow \quad K(Y_j,Y_k) = \varphi(Y_j)^T \varphi(Y_k)$$

[liniar] [neliniar]

Support Vector Machines (cont.)

- abordare neliniară: "*kernel trick*" (cont.);
  - funcțiile nucleu trebuie să fie semi-pozitiv definite și simetrice;
  - exemple de funcții uzuale folosite pentru SVM:

$K(Y_j,Y_k) = Y_j^T Y_k$	*liniar*
$K(Y_j,Y_k) = (1 + Y_j^T Y_k)^p$	*polinomial*
$K(Y_j,Y_k) = e^{-\frac{\lVert Y_j - Y_k \rVert^2}{2\sigma^2}}$	*Gaussiană (Radial Basis Function)*
$K(Y_j,Y_k) = 1 - 2\sum_{i=1}^{n} \frac{(y_{j,i} - y_{k,i})^2}{(y_{j,i} + y_{k,i})}$	*Chi-Square*

Support Vector Machines (cont.)

- abordare neliniară: "*kernel trick*" (cont.);

să se determine $\alpha_1,\ldots, \alpha_m$ astfel încât să maximizăm:

$$\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j c_i c_j K(Y_i, Y_j)$$

cu următoarele ipoteze:

(1) $\sum_j \alpha_j c_j = 0$

(2) $\alpha_i \geq 0$ pentru $\forall \alpha_i$

- clasificatorul este dat de:

$$\boxed{f_{neliniarSVM}(X_i) = \sum_j \alpha_j c_j K(Y_j, X_i) + b}$$

unde $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$, sunt datele de clasificat;

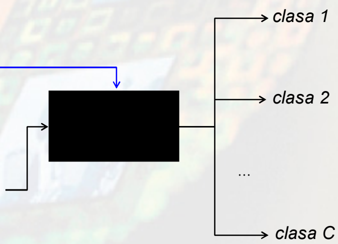
Support Vector Machines (cont.)

- *ce se întâmplă dacă datele de clasificat sunt multi-clasă?*
(SVM este nativ un clasificator binar)
  [formăm un clasificator multiclasă]

> date de intrare:

- date de antrenare:
  {(Y_j,c_j)}, $j=1,\ldots,m$,
  $Y_j=[y_{j,1},\ldots,y_{j,n}]$,
  $c_j \in \{1,\ldots,C\}$;

- date de clasificat:
  $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$;



Support Vector Machines (cont.)

- clasificare multi-clasă (cont.):

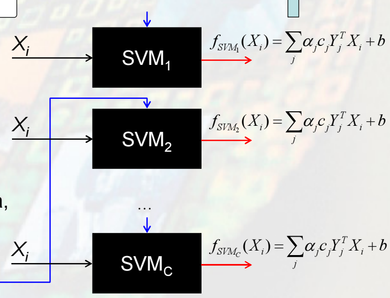
cum luăm decizia de clasificare?
$\text{argmax}_{c=1,\ldots,C}\{f_{SVM_c}(X_i)\}$

abordare 1 - **one-vs.-all**

- sunt creați C clasificatori SVM (câte unul pentru fiecare clasă);

- clasificatorii sunt antrenați cu datele de antrenare modificate ca, exemplu SVM_2:

  {(Y_j,+1)} *dacă* c_j=2
  {(Y_j,-1)} altfel



Support Vector Machines (cont.)

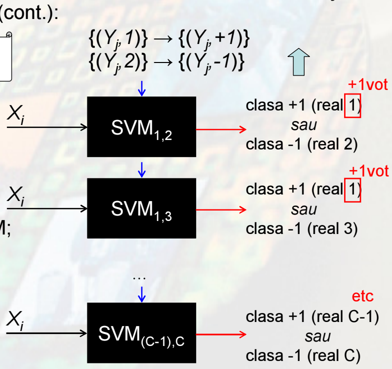
cum luăm decizia de clasificare? vot majoritar

- clasificare multi-clasă (cont.):

abordare 2 - **one-vs.-one**

- sunt creați C' clasificatori SVM, câte unul pentru fiecare combinație de două clase = $\frac{(C-1)C}{2}$ x SVM;

- clasificatorii sunt antrenați doar cu datele de antrenare aferente celor două clase;



Arbori de decizie (Decision Trees)

Datele sunt clasificate prin asocierea observațiilor (date de antrenare) cu o serie de concluzii privind valorile acestora (predicție), ceea ce conduce la o reprezentare arborescentă;

- punerea problemei, un exemplu simplu:

  - să presupunem că avem posibilitatea să realizăm patru activități de weekend:
    {“*cumpărături*”,“*film*”,“*tenis*”,“*nimic*”}
    (reprezintă clasele);

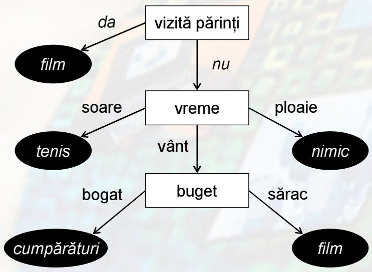
  - aceste activități depind de o serie de variabile:
    “*vreme*” є {“*vânt*”,“*ploaie*”,“*soare*”}
    “*buget*”є {“*bogat*”,“*sărac*”}
    “*vizită părinți*”є {“*da*”,“*nu*”}
    (reprezintă atributele datelor);

[Simon Colton, lectures, 2004]

Arbori de decizie (Decision Trees; cont.)

- punerea problemei, un exemplu simplu (cont.):

  - pe baza cunoștințelor actuale putem asocia valorile atributelor unor decizii de activități (clase): ~ etapă de antrenare;



nodurile arborelui reprezintă *atributele*;

ramurile arborelui reprezintă *relaționarea valorilor atributelor*;

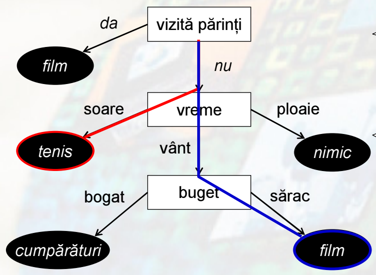
frunzele arborelui reprezintă *clasele*;

[Simon Colton, lectures, 2004]

Arbori de decizie (Decision Trees; cont.)

- punerea problemei, un exemplu simplu (cont.):

  - pentru o serie de valori noi ale atributelor, folosind arborele creat putem lua o decizie: ~ etapă de clasificare;



Sâmbătă 16 Mai:
“vizită părinți” = “nu”
“vreme” = “soare”

Duminică 17 Mai:
“vizită părinți” = “nu”
“vreme” = “vânt”
“buget” = “sărac”

[Simon Colton, lectures, 2004]

Arbori de decizie (Decision Trees; cont.)

- antrenarea arborelui (metoda ID3 - Iterative Dichotomiser 3);

  - *cum selectăm atributele care să fie asociate nodurilor și cum alegem ordinea (prioritatea) acestora?*

• *entropie*: având la dispoziție un sistem binar de clasificare și un set de exemple S în care $p+$ % exemple sunt clasificate în clasa1 (pozitive) și respectiv $p-$ % în clasa2 (negative) atunci:

$$entropy\,(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

> este o măsură a “purității” datelor pentru o colecție de exemple (puritate = datele sunt fie toate în clasă sau clasa este goală);

$$p \to 0 \Rightarrow \log_2(p)\ \text{mare, negativ;}\ \ p\log_2(p) \cong 0$$

$$p \to 1 \Rightarrow \log_2(p)\ \text{mic},\ \approx 0;\ p\log_2(p) \cong 0$$

Arbori de decizie (Decision Trees; cont.)

- antrenarea arborelui (metoda ID3; cont.);

• *entropie*: în cazul a C clase, unde setul de exemple S are p_i % exemple clasificate în clasa c_i, atunci:

$$entropy\,(S) = \sum_{i=1}^{C} -p_i \log_2(p_i)$$

• *câștig informațional* (“information gain”): pentru un atribut A, cu mulțimea de valori posibile $\{A\}$, notând cu S_a subsetul de exemple din S în care atributul A are valoarea a, atunci:

$$gain\,(S,A) = entropy\,(S) - \sum_{a\in\{A\}} \frac{|S_a|}{|S|}\, entropy\,(S_a)$$

unde operatorul |.| returnează numărul de elemente al unui set.

> este o măsură a reducerii entropiei datorată valorii lui A;

Arbori de decizie (Decision Trees; cont.)

- antrenarea arborelui (metoda ID3; cont.);

atribute

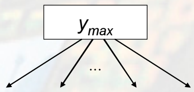
> date de intrare:

- date de antrenare S: $\{(Y_j, c_j)\}$, $j=1,\ldots,m$, $Y_j=[\boxed{y_{j,1},\ldots,y_{j,n}}]$, $c_j \in \{1,\ldots,C\}$;
- date de clasificat: $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,n$;

> algoritm:

> antrenare:

p1. atributul $y=y_{max}$ pentru care avem valoarea maximă a information gain, $gain(S,y)$, relativ la S este ales drept rădăcină;



p2. pentru fiecare valoare posibilă a lui y_{max} (din mulțimea $\{y_{max}\}$) creăm o ramură;

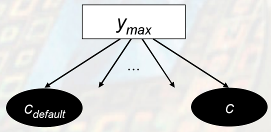
Arbori de decizie (Decision Trees; cont.)

- antrenarea arborelui (metoda ID3; cont.);

> algoritm (cont.):

> antrenare (cont.):

p3. pentru fiecare ramură calculăm S_v (cu v valoarea asociată ramurii);



p4. dacă S_v este mulțimea vidă atunci determinăm clasa $c_{default}$ care are cele mai multe exemple în setul de antrenare S; aceasta definește frunza ce închide această ramură;

p5. dacă S_v conține doar date dintr-o clasă c atunci definim cu această clasă frunza ce închide ramura;

Arbori de decizie (Decision Trees; cont.)

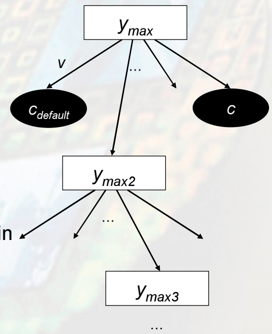
- antrenarea arborelui (metoda ID3; cont.);

> algoritm (cont.):

> antrenare (cont.):

p6. dacă S_v nu este conform p4 și p5 atunci:

- y_{max} este eliminat din lista potențialelor atribute care definesc noduri;
- determinăm atributul $y=y_{max2}$ pentru care avem information gain maxim relativ la S_v, $gain(S_v,y)$;
- acesta crează un nod nou;
- se repetă algoritmul cu pasul 2.



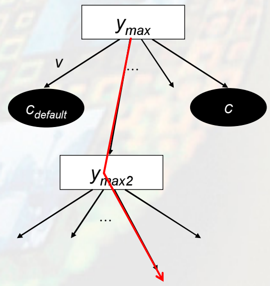
Arbori de decizie (Decision Trees; cont.)

- antrenarea arborelui (metoda ID3; cont.);

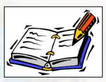
> algoritm (cont.):

> clasificare:

- pentru un vector X de intrare nu trebuie decât să parcurg arborele în funcție de valorile atributelor;
- clasa de apartenență a lui X este dată de frunza arborelui la care ajung în final;
- procesul de clasificare este imediat.



Arbori de decizie (Decision Trees; cont.)



> exemplu numeric:

- date de intrare (setul S):

nr.	vreme	vizită părinți	buget	decizie
#1	soare	da	bogat	film
#2	soare	nu	bogat	tenis
#3	vânt	da	bogat	film
#4	ploaie	da	sărac	film
#5	ploaie	nu	bogat	nimic
#6	ploaie	da	sărac	film
#7	vânt	nu	sărac	film
#8	vânt	nu	bogat	cumpărături
#9	vânt	da	bogat	film
#10	soare	nu	bogat	tenis

[Simon Colton, lectures, 2004]

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p1: alegere nod rădăcină:

$$entropy(S) = \sum_{i=1}^{C} -p_i \log_2(p_i)$$

$C \in$ {film, tenis, cumpărături, nimic}

$$entropy(S) = -p_{film}\log_2(p_{film}) - p_{tenis}\log_2(p_{tenis}) - $$
$$- p_{cumparaturi}\log_2(p_{cumparaturi}) - p_{nimic}\log_2(p_{nimic})$$

$$= -\frac{6}{10}\log_2\left(\frac{6}{10}\right) - \frac{2}{10}\log_2\left(\frac{2}{10}\right) - \frac{1}{10}\log_2\left(\frac{1}{10}\right)$$
$$-\frac{1}{10}\log_2\left(\frac{1}{10}\right) = 1.571$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

$A \in$ {vreme, vizită părinți, buget}

- p1: alegere nod rădăcină (cont.):

$$gain(S, A) = entropy(S) - \sum_{a \in \{A\}} \frac{|S_a|}{|S|} entropy(S_a)$$

$$gain(S, vreme) = 1.571 - \frac{3}{10} entropy(S_{soare}) - \frac{4}{10} entropy(S_{vant}) - \frac{3}{10} entropy(S_{ploaie})$$

$$entropy(S_{ploaie}) = -\frac{2}{3}\log_2\left(\frac{2}{3}\right) - 0 - 0 - \frac{1}{3}\log_2\left(\frac{1}{3}\right) = $$
$$= 0.918$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p1: alegere nod rădăcină (cont.):

$$gain(S, vreme) = 1.571 - \frac{3}{10} entropy(S_{soare}) - \frac{4}{10} entropy(S_{vant}) - \frac{3}{10} entropy(S_{ploaie})$$

$$entropy(S_{ploaie}) = 0.918$$
$$entropy(S_{vant}) = 0.811 \Rightarrow gain(S, vreme) = 0.7$$
$$entropy(S_{soare}) = 0.918$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p1: alegere nod rădăcină (cont.):

$$gain(S, vizita\ parinti) = 1.571 - \frac{5}{10} entropy(S_{da}) - \frac{5}{10} entropy(S_{nu})$$

$$entropy(S_{da}) = 0$$
$$entropy(S_{nu}) = 1.922 \Rightarrow gain(S, vizita\ parinti) = 0.61$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p1: alegere nod rădăcină (cont.):

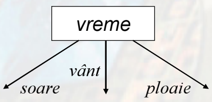
$$gain(S, buget) = 1.571 - \frac{7}{10} entropy(S_{bogat}) - \frac{3}{10} entropy(S_{sarac})$$

$$entropy(S_{bogat}) = 1.842$$
$$entropy(S_{sarac}) = 0 \Rightarrow gain(S, buget) = 0.2816$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p1: alegere nod rădăcină (cont.):



$$\boxed{gain(S, vreme) = 0.7}$$
$$gain(S, vizita\ parinti) = 0.61$$
$$gain(S, buget) = 0.2816$$

- p2: creăm ramurile pentru nodul *vreme*;

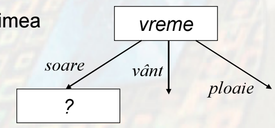
- p3: calculăm:

$$S_{ploaie} = \{\#4, \#5, \#6\}, |S_{ploaie}| = 3;$$
$$S_{soare} = \{\#1, \#2, \#10\}, |S_{soare}| = 3;$$
$$S_{vant} = \{\#3, \#7, \#8, \#9\}, |S_{vant}| = 4.$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p4: S_{ploaie}, S_{soare}, $S_{vânt}$ sunt mulțimea vidă? nu;



- p5: S_{ploaie}, S_{soare}, $S_{vânt}$ conțin date doar dintr-o clasă? nu;

- p6: determinăm atributul (altul decât *vreme*) pentru care obținem information gain maxim:

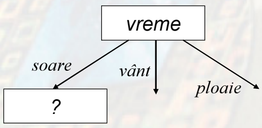
$$entropy(S_{soare}) = -p_{film}\log_2(p_{film}) - p_{tenis}\log_2(p_{tenis}) - p_{cumparaturi}\log_2(p_{cumparaturi}) - p_{nimic}\log_2(p_{nimic})$$

- valorile sunt calculate pe S_{soare} doar;

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p6: determinăm atributul pentru care obținem information gain maxim (cont.):



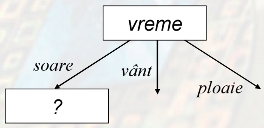
$$entropy\,(S_{soare}) =$$
$$-\,p_{film}\log_2(p_{film}) - p_{tenis}\log_2(p_{tenis}) -$$
$$-\,p_{cumparatur\,i}\log_2(p_{cumparatur\,i}) - p_{nimic}\log_2(p_{nimic})$$

$$entropy\,(S_{soare}) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) - 0 - 0$$
$$= 0.918$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p6: determinăm atributul pentru care obținem information gain maxim (cont.):



$$gain(S_{soare}, vizita\ \ parinti) =$$
$$0.918 - \frac{1}{3}entropy\,(S_{da}) - \frac{2}{3}entropy\,(S_{nu})$$

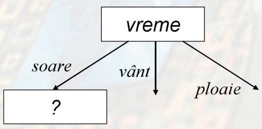
$$entropy\,(S_{da}) = 0$$
$$entropy\,(S_{nu}) = 0$$

$\Rightarrow$ $gain(S_{soare}, vizita\ \ parinti) = 0.918$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p6: determinăm atributul pentru care obținem information gain maxim (cont.):



$$gain(S_{soare}, buget) =$$
$$0.918 - \frac{3}{3}entropy\,(S_{bogat}) - \frac{0}{3}entropy\,(S_{sarac})$$

$$entropy\,(S_{bogat}) = 0.918$$
$$entropy\,(S_{sarac}) = 0$$

$\Rightarrow$ $gain(S_{soare}, buget) = 0$

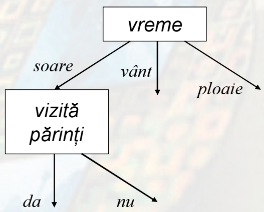
Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p6: determinăm atributul pentru care obținem information gain maxim (cont.):

$\boxed{gain(S_{soare}, vizita\ \ parinti) = 0.918}$

$gain(S_{soare}, buget) = 0$



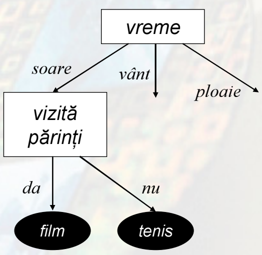
- p2: creăm ramurile pentru nodul *vizită părinți*;
- p3: calculăm:

$$S_{da \cap soare} = \{\#1\}, |S_{da}| = 1;$$
$$S_{nu \cap soare} = \{\#2, \#10\}, |S_{nu}| = 2.$$

Arbori de decizie (Decision Trees; cont.)

> exemplu numeric (cont.):

- p4: S_{da}, S_{nu} sunt mulțimea vidă? nu;
- p5: S_{da}, S_{nu} conțin date doar dintr-o clasă?
  - da, S_{da} este in categoria *film*;
  - da, S_{nu} este in categoria *tenis*;



- ... pentru valorile *vânt* și *ploaie* se procedează similar ...

> Sfârşit M4