RECUNOAȘTEREA formelor

Radu-Daniel Vatavu

Serialul despre recunoașterea formelor continuă cu prezentarea unei tehnici des folosite în acest domeniu, și anume învățarea supravegheată. Vă vom prezenta câteva dintre regulile de bază, și anume regula celui mai apropiat vecin, cea a celor mai apropiați k vecini, cea a lui Bayes etc.

În procesul de învățare supravegheată se presupune faptul că există un set de învățare pe baza căruia se poate construi un model de clasificator care va fi folosit în procesul de recunoaștere. Fie F mulțimea formelor furnizate sistemului, C mulțimea claselor și S setul de învățare:

$$S = \{(x_i, c_j) / x_i \in F, c_j \in C\}$$

$$|F| = p, |C| = m,$$

unde p și m reprezintă numărul de elemente ale mulțimii F, respectiv C.

Mulțimea formelor F poate fi partiționată în m clase:

$$F = \{F_0, F_1, \ldots, F_{m-1}\}, F_i \cap F_j = \varnothing,$$

$$\forall i, j \in \{0,1,\ldots,m-1\}, i \neq j, F = \bigcup_{j=0}^{m-1} F_j .$$

O clasă F_i se numește ***omogenă*** dacă este îndeplinită condiția:

$$d(x_1, x_2) \leq d(x_1, y), d(x_1, x_2) \leq d(x_2, y),$$
$$\forall x_1, x_2 \in F_i, \forall y \notin F_i,$$

unde d desemnează o distanță.

Pentru formele din aceleași clase F_i se poate calcula o formă "medie" care reprezintă centrul de greutate al fiecărei clase:

$$m_i = \frac{1}{n_i} \sum_{j=0}^{n_i - 1} x_j^i$$

unde prin n_i s-a notat numărul formelor din clasa F_i, iar x_j^i este forma de indice j care aparține clasei i ($j = 0, ..., n_i - 1$).

Regula celui mai apropiat vecin

Cea mai intuitivă metodă de clasificare a unei noi forme x, plecând de la o mulțime de clase și o distanță d, este reprezentată de regula celui mai apropiat vecin (***NN*** - ***N****earest* ***N****eighbour*). Aceasta constă în calcularea distanțelor dintre forma x și fiecare formă x_i ($i = 0, 1, ..., p - 1$), care aparțin mulțimii F, luându-se următoarea decizie:

$$x \in F_i \Leftrightarrow \exists\, y \in F_i \text{ a.î. } \min\{d(x, x_i) / x_i \in F\} = d(x, y)$$

Deci, se va considera că noua formă aparține clasei din care face parte cel mai apropiat vecin, pentru o anumită distanță d.

Trebuie notat faptul că această regulă folosește ca informație privind clasificarea numai cel mai apropiat vecin, ignorând pur și simplu celelalte forme (respectiv distribuția lor în alte clase). Ca urmare, rezultatul obținut s-ar putea să nu fie întotdeauna corect, fie datorită prezenței unui anumit nivel de zgomot suprapus peste valorile caracteristicilor, fie datorită faptului că forma y a fost greșit clasificată. Unele dezavantaje prezentate de această regulă pot fi eliminate prin fundamentarea clasificării pe apartenența la clase a mai multor vecini.

Regula celor mai apropiați k vecini

Regula ***KNN*** (***K*** *Nearest* ***N****eighbours*) ia în considerare cei mai apropiați k vecini ai formei x, decizia fundamentându-se astfel: forma x aparține clasei din care fac parte cei mai mulți dintre cei k vecini.

Numărul k trebuie ales astfel încât să fie suficient de mare pentru a minimiza probabilitatea unei clasificări greșite și suficient de mic (în raport cu numărul p al formelor) astfel încât cei k vecini să fie într-adevăr "aproape" de x pentru a asigura o estimare corectă a clasei.

Această metodă este des utilizată datorită avantajelor pe care le prezintă: nu sunt necesare informații privind probabilitatea de apartenență a unei forme la o anumită clasă, este ușor de implementat și prezintă o probabilitate mică de eroare.

În figura 1 este prezentată diferența dintre regulile *NN* și *KNN*, considerându-se un caz ipotetic de clasificare a unei forme necunoscute la una dintre cele două clase. S-a presupus că numărul caracteristicilor utilizate este $n = 2$ pentru a se facilita reprezentarea formelor în spațiul caracteristicilor.

Dezavantajul acestor două metode constă în faptul că, de fiecare dată când se dorește clasificarea unei noi forme,

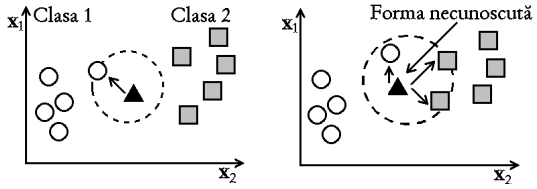


Figura 1: a) forma este atribuită clasei 1;
b) forma este atribuită clasei 2.

este necesară calcularea a *p* distanțe, ceea ce poate determina un timp de calcul ridicat.

Funcții discriminant

În spațiul caracteristicilor, fiecărei forme îi corespunde un punct având drept coordonate valorile celor *n* caracteristici:

$$x = (x_0, x_1, \ldots, x_{n-1})$$

Alegerea corespunzătoare a caracteristicilor va determina reliefarea similarității dintre forme prin apropierea dintre punctele din spațiul caracteristicilor. După cum se observă în figura 1, formele din fiecare clasă sunt grupate. Acest lucru determină posibilitatea separării claselor printr-o curbă discriminant (sau o hipersuprafață discriminant într-un plan *n*-dimensional).

Dacă există hipersuprafețe discriminant care să separe planul caracteristicilor astfel încât formele care aparțin aceleiași clase să se găsească în aceeași regiune, atunci clasele se numesc ***separabile***. Dacă hipersuprafețele discriminant sunt hiperplane, atunci clasele se numesc ***liniar separabile***.

Dacă mulțimea claselor are cardinalul *m*, hipersuprafețele sunt definite de *m* funcții $g_i(x)$, i=0, 1, ..., m - 1 numite ***funcții discriminant***. Alegerea acestor funcții trebuie să se facă astfel încât să fie îndeplinită condiția:

$$g_i(x) > g_j(x), \forall j = 0, \ldots, m-1,\ j \neq i\ ,$$

pentru orice formă *x* aparținând clasei *i*.

Ca urmare, decizia de apartenență a unei forme necunoscute *x* la o clasă se va fundamenta astfel:

$$x \in F_j \Leftrightarrow g_j(x) = \max\{g_i(x) / i = 0, \ldots, m-1\} \qquad (*).$$

Regiunile de decizie sunt separate de hipersuprafețele discriminant. Hipersuprafața de separație dintre clasele *i* și *j* este dată de ecuația:

$$g_{ij} = g_i(x) - g_j(x) = 0$$

și conține toate punctele (formele) *x* aflate la distanță egală față de clasele *i* și *j*.

În cazul în care clasele sunt liniar separabile, funcțiile discriminant vor fi funcții liniare de tipul:

$$g(x) = a_0 x_0 + a_1 x_1 + \ldots + a_{n-1} x_{n-1} + a_n,$$
$$a_i \in \Re,\ x = (x_0, x_1, \ldots, x_{n-1})$$

sau

$$g(x) = a \cdot x^T + a_n,\ a = (a_0, a_1, \ldots, a_{n-1}),\ x^T = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Pentru a determina funcțiile discriminant g_i, se pot folosi vectorii formă medie ai fiecărei clase (m_i) în baza următoarei definiții:

$$g_i(x) = F\{d(x, m_i)\},\ i = 0, \ldots, m-1,$$

unde *d* este o distanță iar *F* este o funcție descrescătoare.

Ca urmare, forma *x* va aparține clasei pentru care distanța de la *x* la forma medie a clasei este minimă (asemănător cu regula *NN*).

Pentru cazul distanței euclidiene, distanța dintre forma *x* și forma medie m_i poate fi rescrisă:

$$d^2(x, m_i) = (x - m_i) \cdot (x - m_i)^T = (x - m_i) \cdot (x^T - m_i^T) =$$
$$= x \cdot x^T - x \cdot m_i^T - m_i \cdot x^T + m_i \cdot m_i^T =$$
$$= x \cdot x^T - 2 \cdot m_i \cdot x^T + m_i \cdot m_i^T =$$
$$= -2 \cdot \left[m_i \cdot x^T + 0.5 \cdot m_i \cdot m_i^T\right] + x \cdot x^T \quad .$$

Având în vedere faptul că pentru un anumit *x*, factorul $x \cdot x^T$ este constant pentru fiecare i = 0, ..., m-1, funcțiile discriminant pot fi alese:

$$g_i(x) = m_i \cdot x^T + 0.5 \cdot m_i \cdot m_i^T\ .$$

Astfel, conform criteriului (∗) definit anterior, forma necunoscută *x* va fi clasificată în clasa *j* pentru care $g_j(x)$ are valoarea maximă sau, altfel spus, distanța $d(x, m_j)$ are valoarea minimă.

Procesul de învățare constă în determinarea coeficienților funcțiilor discriminant $g_i(x)$.

Pe baza considerațiilor de mai sus se obține algoritmul de clasificare care este prezentat în continuare. Presupunem că funcțiile discriminant sunt deja calculate conform metodei prezentate, iar matricea *g*, de ordin $m \cdot (n + 1)$, conține coeficienții. Mulțimea *C* a claselor este reprezentată de vectorul `clasa`. Forma necunoscută a fost notată cu *x*.

```
clasa_x ← -1
max ← 0
pentru i ← 0, m - 1 execută
   temp ← 0
   pentru j ← 0, n - 1 execută
      temp ← temp + g_ij * x_j
   sfârșit pentru
   temp ← temp + g_in
   dacă temp > max atunci
      max ← temp
      clasa_x ← j
   sfârșit dacă
sfârșit pentru
scrie clasa[clasa_x]
```

În figura 2 este prezentat un caz ipotetic care cuprinde trei clase și indică suprafețele de separație dintre acestea. Fiecare formă este descrisă de două caracteristici x_1 și x_2.

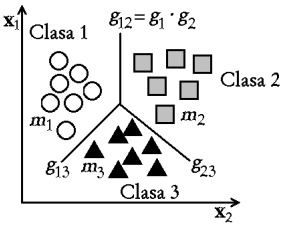


Figura 2

Dacă un astfel de clasificator, bazat pe distanța minimă față de centrul de greutate al fiecărei clase, oferă performanțele așteptate, nu există motive pentru utilizarea unor metode mai complicate (cum ar fi rețelele neuronale). Însă, există anumite situații în care rezultatul returnat de clasifi-

cator nu este cel corect. Câteva dintre acestea, precum și posibile soluții, sunt prezentate în tabelul 1. Au fost luate în considerare două clase și două caracteristici.

Rețele neuronale

După cum rezultă din tabelul 1, în anumite situații, suprafețele de separație dintre clase nu sunt liniare, ci devin chiar foarte complexe. În această situație, un clasificator de tipul celui prezentat mai sus nu poate oferi performanțele așteptate. În astfel de condiții, realizarea unui clasificator care să asigure rezultatele corespunzătoare se poate face apelând la rețelele neuronale. În continuare vor fi amintite modelele cunoscute ale *perceptronului* simplu și multistrat, prezentându-se și algoritmii de învățare pentru aceste modele.

Perceptronul este prezentat structural în figura 3, ieșirea sa y având valoarea:

$$y = f\left(\sum_{j=0}^{n-1} w_j \cdot x_j + w_n\right),$$

unde funcția de activare poate fi funcția prag simplă:

$$f(a)=\begin{cases} 1 \text{ pentru } a \geq 0 \\ -1 \text{ pentru } a < 0 \end{cases}.$$

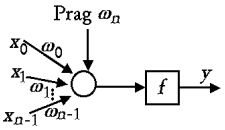


Figura 3

Perceptronul simplu este echivalent cu un clasificator liniar care face distincția dintre două clase, ponderile ω_j reprezentând coeficienții hiperplanului de separație. Astfel, forma x de intrare va fi clasificată în clasa 1 dacă y = 1 sau în clasa 2 dacă y = -1. Deci, pentru o clasificare corectă avem:

$$y \cdot z > 0, \quad \text{unde } z = \sum_{j=0}^{n-1} w_j \cdot x_j + w_n.$$

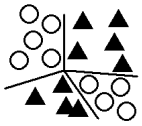
În cazul în care ieșirea perceptronului nu respectă valoarea clasei din setul de învățare, valorile ponderilor ω_j vor fi modificate corespunzător:

$$w_j = w_j + \alpha \cdot x_j, \; j = 0, \ldots, n-1,$$

unde α este o constantă de corecție pozitivă.

Algoritmul pentru învățarea perceptronului este prezentat în continuare:

Se aleg valori aleatoare mici pentru ω_j, `j = 0, ..., n`
Se alege o valoare pentru constanta α $(0 < \alpha \leq 1)$

Descriere Situație	Soluție posibilă
caracteristici alese necorespunzător 	Un clasificator liniar nu va putea niciodată separa cele două clase. O soluție ar fi construirea unor caracteristici mai bune, însă această operație nu se realizează întotdeauna ușor. Dacă se cunosc anumite informații despre distribuția formelor în clase și diferite probabilități condiționate de apartenență se pot aplica alte metode (de exemplu, *regula lui Bayes*).
caracteristici corelate 	Există posibilitatea ca două caracteristici să varieze împreună, influențate de un factor comun. Acest fenomen trebuie evitat (a se vedea discuția despre selecția caracteristicilor), dar nu întotdeauna corelația este remarcată. Astfel, o nouă formă x poate să fie mai apropiată de centrul clasei greșite. O soluție posibilă este utilizarea unei distanțe de tip *Mahalanobis*. O problemă similară apare dacă cele două caracteristici nu sunt scalate corespunzător (de exemplu, una este exprimată în centimetri, iar alta în kilometri).
suprafețe de separație dintre clase neliniare 	Suprafețele liniare generate de clasificatorii liniari s-ar putea să nu fie corespunzătoare pentru distingerea unor astfel de clase. Soluții posibile ar fi: redefinirea caracteristicilor, aplicarea distanței *Mahalanobis* sau cazul extrem: folosirea *rețelelor neuronale*.
existența subclaselor 	Se observă că avem cele patru subclase separate prin suprafețe liniare, dar nu putem spune același lucru despre cele două clase (este exemplul clasic al imposibilității implementării funcției *XOR* cu ajutorul perceptronului simplu). O soluție posibilă constă în folosirea unei metode specifice învățării nesupravegheate (cum ar fi metoda nucleelor dinamice) pentru împărțirea claselor în subclase. Rezultă astfel patru clase care într-o primă fază pot fi considerate distincte, iar apoi rezultatele sunt combinate cu ajutorul unei funcții *OR*.

Tabelul 1: Limitări ale funcțiilor discriminant liniare

repetă
  învățat ← 1
  pentru k ← 0, p-1 **execută**
    calculează z(k) *și* y(k) = f(z(k)) *pentru forma* x(k)
    dacă y(k)*z(k)<0 **atunci**
      modifică valorile ponderilor ω_j, j = 0, …, n
      învățat ← 0
    sfârșit dacă
  sfârșit pentru
până când învățat = 1

Perceptronul multistrat reprezintă o dezvoltare a perceptronului simplu, fiind organizat pe mai multe niveluri, având drept obiectiv realizarea unei funcții de decizie, care transformă vectori aparținând spațiului R^n în vectori m-dimensionali (din R^m). Structura unui perceptron multistrat complet conectat cu trei straturi, n intrări și m ieșiri este prezentată în figura 4 ($\omega_{ij}^{(in)}$ reprezintă ponderea legăturii dintre neuronul i al stratului de intrare și neuronul j al stratului ascuns, iar $\omega_{ik}^{(out)}$ ponderea legăturii dintre neuronul j al stratului ascuns și neuronul k aparținând stratului de ieșire).

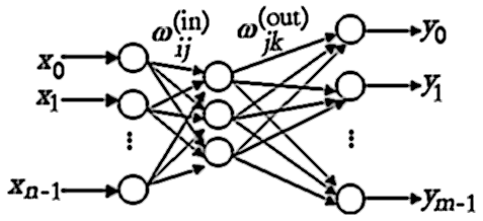


Figura 4

Unul din rezultatele legate de perceptronul multistrat și prin care se explică performanțele deosebite obținute prin utilizarea acestuia este dat de teorema lui ***Kolmogorov*** care afirmă că orice funcție continuă definită pe un cub n-dimensional (de exemplu $[0, 1]^n$) poate fi scrisă sub următoarea formă:

$$g(x_0, x_1, \ldots x_{n-1}) = \sum_{j=0}^{2n} \Psi_j \left(\sum_{i=0}^{n-1} \phi_{ij}(x_i) \right),$$

unde Ψ_j și ϕ_{ij} sunt funcții continue. Ca urmare, transformarea dintr-un cub n-dimensional $[0, 1]^n$ în spațiul R^m poate fi realizată doar printr-o rețea cu două straturi (fără a număra stratul de intrare) care necesită un strat ascuns cu cel puțin $2 \cdot n + 1$ neuroni. Amintind discuția referitoare la limitările clasificatorului liniar și complexitatea suprafețelor de separare, rezultă imediat aplicabilitatea perceptronului multistrat în discriminarea dintre clase. Trebuie precizat faptul că teorema lui *Kolmogorov* asigură doar existența funcțiilor, nu și forma acestora.

În continuare va fi schițat algoritmul de învățare cu propagarea înapoi a erorii pentru o rețea cu trei straturi. Sunt folosite următoarele notații:

- n - numărul intrărilor rețelei;
- h - numărul neuronilor din stratul ascuns;
- m - numărul ieșirilor rețelei;
- $w_{ij}^{(in)}$ - ponderea legăturii dintre neuronul i din stratul de intrare și neuronul j din stratul ascuns;
- $w_{jk}^{(out)}$ - ponderea legăturii dintre neuronul j al stratului ascuns și neuronul k al stratului de ieșire;
- f - funcția de activare a ieșirii fiecărui neuron; are forma: $f(a) = \frac{1}{1+e^{-a}}$;
- f' - derivata funcției de activare;
- p - numărul de elemente ale setului de învățare;
- x^s - vectorul de intrare (n-dimensional) cu indicele s din setul de învățare: $x^s = (x_0^s, x_1^s, \ldots, x_{n-1}^s)$;
- c^s - vectorul de ieșire așteptat (m-dimensional) când la intrare se aplică forma x^s: $c^s = (c_0^s, c_1^s, \ldots, c_{m-1}^s)$;
- y^s - vectorul de ieșire al rețelei (m-dimensional) când la intrare se aplică forma x^s: $y^s = (y_0^s, y_1^s, \ldots, y_{m-1}^s)$;
- a_j - valoarea de activare a neuronului j din stratul ascuns;
- b_j - ieșirea neuronului j din stratul ascuns $b_j = f(a_j)$;
- o_k - valoarea de activare a neuronului k din stratul de ieșire; evident, y_k^s (ieșirea neuronului k din stratul de ieșire) este $f(o_k)$;
- E_s - eroarea prezentată de ieșirile neuronilor corespunzători intrării x^s;
- E - eroarea medie pentru cele p forme ale intrării;
- *eps* - eroarea acceptată pentru învățarea setului de forme.

Algoritmul de învățare este prezentat în continuare:

Se aleg valori aleatoare mici pentru ponderile rețelei $w_{ij}^{(in)}$ *și* $w_{jk}^{(out)}$, i = 0, …, n - 1, j = 0, …, h - 1, k = 0, …, m - 1
Se alege valoarea constantei de instruire α ($0 < \alpha < 1$)
repetă
  pentru s ← 0, p - 1 **execută**
    Se calculează ieșirile pentru cele două straturi

$$b_j \leftarrow f(a_j),\ a_j \leftarrow \sum_{i=0}^{n-1} w_{ij}^{(in)} \cdot x_i^s,\ j = 0,\ldots,h-1$$

$$y_k^s = f(o_k),\ o_k = f\left(\sum_{j=0}^{h-1} w_{jk}^{(out)} \cdot b_j \right),\ k = 0,\ldots,m-1$$

    Se actualizează ponderile de ieșire $w_{jk}^{(out)}$

$$w_{jk}^{(out)} \leftarrow w_{jk}^{(out)} + \alpha \cdot (c_k^s - y_k^s)\, f'(o_k^s)\, f'(a_j)$$

    Se actualizează ponderile de intrare $w_{ij}^{(in)}$

$$w_{ij}^{(in)} \leftarrow w_{ij}^{(in)} + \alpha \cdot x_i^s \cdot f'(a_j) \sum_{k=0}^{m-1} w_{jk} \cdot (c_k^s - y_k^s)\, f'(o_k)$$

    Se calculează eroarea rețelei pentru intrarea x^s

$$E_s = \frac{1}{2} \sum_{k=0}^{m-1} (y_k^s - c_k^s)^2$$

  sfârșit pentru

Se calculează eroarea medie totală $E = \frac{\sum_{s=0}^{p-1} E_s}{p}$
până când E < eps

Practic, algoritmul constă în minimizarea erorii pentru fiecare formă aplicată la intrare prin actualizarea ponderilor de intrare și de ieșire folosind o metodă de tip gradient (care determină apariția derivatei funcției de activare *f*).

Regula lui Bayes

O altă abordare în cadrul contextului supravegheat al recunoașterii formelor apelează la teoria probabilităților. Fundamentarea deciziei de apartenență a unei forme *x* la o anumită clasă poate fi influențată de următoarele elemente: probabilitatea de apariție a unei anumite forme *x*, probabilitatea de apariție a unei forme *x* aparținând unei clase *j* etc.

Probabilitatea condiționată a evenimentului *A*, știind că un eveniment *B* a avut loc, se definește astfel:

$$\Pr\{A \mid B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}},$$

unde evenimentele *A* și *B* sunt considerate submulțimi ale unui spațiu de selecție *S*.

Teorema lui *Bayes* permite calculul probabilităților condiționate astfel:

$$\Pr\{A \mid B\} = \frac{\Pr\{A\} \cdot \Pr\{B \mid A\}}{\Pr\{B\}}.$$

Particularizând pentru problema recunoașterii formelor, fie:

- Pr{*x*} - probabilitatea de apariție a unei forme particulare *x*;
- Pr{*x* | *j*} - probabilitatea condiționată de apariție a unei anumite forme *x*, dată fiind clasa *j*;
- Pr{*j*} - probabilitatea de apariție a unei forme din clasa *j*;
- Pr{*j* | *x*} - probabilitatea condiționată de apartenență la clasa *j* (probabilitatea de apariție a clasei *j*) dată fiind forma *x*.

Aplicând teorema lui *Bayes* (fiind interesați de probabilitatea de apartenență a unei forme *x* la o clasă *j*), avem:

$$\Pr\{j \mid x\} = \frac{\Pr\{j\} \cdot \Pr\{x \mid j\}}{\Pr\{x\}}.$$

Ca urmare, se poate considera următoarea decizie de clasificare:

$$x \in F_j \Leftrightarrow \Pr\{j \mid x\} = \max\{\Pr\{k \mid x\} / k = 0,...,m-1\}.$$

Așadar, obținem:

$$\Pr\{j \mid x\} > \Pr\{k \mid x\} \Leftrightarrow \Pr\{j\} \cdot \Pr\{x \mid j\} > \Pr\{k\} \cdot \Pr\{x \mid k\}$$
$$\forall k = 0,...,m-1, k \neq j.$$

Se pot defini, de asemenea, funcții discriminant astfel:

$$g_j(x) = \Pr\{j\} \cdot \Pr\{x \mid j\},\ j = 0,...,m-1$$

decizia luându-se prin determinarea maximului acestor funcții, cum s-a arătat anterior.

Pentru această abordare sunt necesare două tipuri de informații:

- Pr{*j*} - care este raportul dintre numărul de forme n_j care aparțin clasei *j* și numărul total de forme din mulțimea de învățare;
- Pr{*x* | *j*} - care se determină cu ajutorul distribuției normale *Gauss*:

$$\Pr\{x \mid j\} = \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot e^{\frac{(x-m_j)(x-m_j)'}{2\sigma_j^2}}$$

unde σ_j reprezintă abaterea medie pătratică iar m_j vectorul formă mediu al clasei *j*.

Algoritmul de învățare și clasificare cuprinde următoarele etape:

- calcularea vectorilor formă medii m_j și a abaterilor pătratice medii σ_j pentru fiecare clasă din setul de învățare (*j* = 0, ..., *m* - 1);
- calcularea probabilităților Pr{*x* | *j*} și Pr{*j*} pentru fiecare clasă *j*;
- forma nouă *x* este clasificată la clasa *j* dacă:

$$\Pr\{j\} \cdot \Pr\{x \mid j\} = \max\{\Pr\{k\} \cdot \Pr\{x \mid k\} / k = 0,...,m-1\}.$$

Validarea învățării

După implementarea unui model de clasificator în cadrul unui proces de învățare, acesta va trebui să fie capabil de a realiza funcția de generalizare, adică de a lua decizii corecte pentru alte forme decât cele care aparțin setului de învățare. Capacitatea unui clasificator de a generaliza procesul decizional este dată de mărimea erorii la ieșire. Această eroare poate fi realizată pe baza setului de învățare, calculându-se (eventual sub formă de medie pătratică) diferența dintre ieșirile clasificatorului și ieșirile dorite. Însă, eroarea nu ne va da nici o informație despre capacitatea de generalizare a clasificatorului ci doar despre cât de bine a reușit acesta să învețe setul de intrare.

O metodă simplă de a testa clasificatorul este de a împărți setul inițial de învățare în două subseturi: un subset va fi folosit pentru învățare, iar cel de-al doilea pentru testarea și calculul erorii (deci, a capacității de generalizare). Această metodă dă rezultate bune, însă nu trebuie neglijat subsetul de învățare în favoarea celui de testare deoarece aceasta conduce la scăderea performanțelor clasificatorului. O derivare a acestei metode constă în împărțirea setului inițial de învățare în *k* subseturi de mărime egală (de exemplu, *k* = 10) din care numai un subset este folosit pentru testare. Clasificatorul este construit de *k* ori, de fiecare dată folosindu-se un alt subset pentru testare și se păstrează acea formă a clasificatorului care minimizează eroarea la ieșire.

Considerații privind setul de învățare

Din cele prezentate până acum, reiese foarte clar că performanțele clasificatorului (indiferent cum a fost implementat - *NN*, *KNN*, *rețele neuronale* etc.) precum și capacitatea sa de a generaliza sunt influențate semnificativ de setul de învățare folosit.

Un set de învățare ideal cuprinde un număr minim de exemple necesare construirii unui clasificator capabil de a realiza funcția de generalizare în vederea obținerii unui rezultat corect. Deoarece în practică nu se dispune de un asemenea set ideal, se impune realizarea unor operații de "editare" a setului de învățare care poate avea ca scop: eliminarea cazurilor conflictuale (exemple aproape identice, dar a căror clase diferă semnificativ), exemplelor irelevante care, pur și simplu, măresc inutil dimensiunea setului etc. Această operație trebuie realizată astfel încât să nu se modifice

(sau să se modifice nesemnificativ) suprafețele de separație dintre clase deci, fără a afecta calitatea procesului de învățare. De asemenea, timpul necesar învățării (sau clasificării în cazul regulii *NN* și *KNN* unde este necesar calculul a *p* distanțe) se poate reduce considerabil.

În vederea reducerii dimensiunilor setului de învățare prin eliminarea exemplelor redundante, care nu influențează major performanța procesului de învățare, se pot folosi o serie de metode care se bazează pe ***grafurile de proximitate***.

Aceste grafuri de proximitate au ca noduri punctele din spațiul caracteristicilor asociate fiecărui exemplu aparținând setului de învățare. Două puncte (noduri) sunt legate printr-o muchie dacă sunt "apropiate" dintr-un anumit punct de vedere și dacă nu există alte exemple aflate într-o zonă "interzisă" determinată de cele două puncte inițiale. Aceste aspecte sunt clarificate în continuare, considerându-se două exemple.

Diagrama Voronoi

Diagrama Voronoi reprezintă o partiție a spațiului caracteristicilor în regiuni astfel încât toate punctele dintr-o regiune sunt mai apropiate (din punct de vedere al unei anumite distanțe) de un anumit nod decât de celelalte. *Triangularizarea Voronoi* se obține prin unirea prin muchii a nodurilor care fac parte din regiuni ce prezintă o margine comună (figura 5).

Considerând regula *NN*, când o formă necunoscută "cade" într-o anumită regiune, ea va fi clasificată în clasa nodului regiunii respective.

Suprafața de separare dintre cele două clase este alcătuită din segmente reprezentate de margini ale *regiunilor Voronoi*. Ca urmare, nodurile care nu au contribuit la realizarea suprafeței de separare sunt redundante și pot fi eliminate.

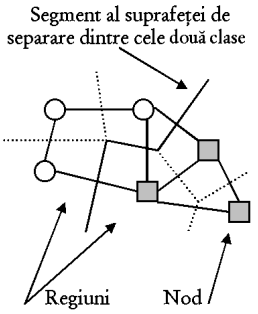


Figura 5

Determinarea triangularizării *Voronoi* se poate realiza folosind un algoritm de tip *divide et impera*, împărțind succesiv setul de învățare în două subseturi de dimensiuni egale, determinând triangularizarea pentru cele două subseturi și realizând reuniunea acestora. În cazurile particulare a două sau trei noduri, regiunile *Voronoi* sunt delimitate de mediatoarele (hiperplanele mediatoare pentru spații cu mai multe dimensiuni) ale segmentelor care unesc cele două, respectiv trei puncte.

Graful Gabriel

În cazul ***grafului Gabriel***, două noduri *A* și *B* sunt legate printr-o muchie dacă sfera determinată de diametrul *AB* nu conține alte noduri. În exemplul din figura 6, nodurile *A* și *B* sunt *vecini Gabriel* în timp ce nodurile *A* și *C* nu îndeplinesc această proprietate. Nodurile ale căror vecini din *graful Gabriel* care aparțin aceleiași clase pot fi eliminate deoarece nu contribuie la alcătuirea suprafeței de separare.

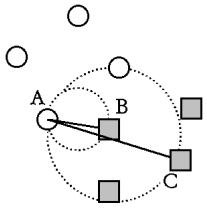


Figura 6

Întotdeauna, setul editat cu ajutorul *grafului Gabriel* este mai mic decât cel obținut cu *diagrama Voronoi* datorită faptului că *graful Gabriel* este un subgraf al *triangularizării Voronoi*.

Algoritmii de editare a setului de învățare sunt identici pentru cele două abordări și constau în următorii pași:

- se determină graful de proximitate (triangularizarea *Voronoi* sau graful *Gabriel*) pentru mulțimea exemplelor din setul de învățare;
- se parcurg nodurile grafului și se marchează acele noduri pentru care toți vecinii aparțin aceleiași clase;
- se șterg nodurile marcate.

Se observă că ordinea de parcurgere a grafului nu contează întrucât toate nodurile marcate sunt șterse în același timp, și anume în cadrul celui de-al treilea pas.

Radu-Daniel Vatavu este student în anul IV la Universitatea Ștefan cel Mare din Suceava. Poate fi contactat prin e-mail la raduvro@yahoo.com.

Un nou Athlon XP

Pe data de 1 octombrie 2002 compania Advanced Micro Devices (AMD) a lansat pe piață o versiune îmbunătățită a procesoarelor Athlon XP, destinate sistemelor desktop.

Cea mai importantă noutate introdusă este magistrala de 333 MHz, folosită în premieră în industria procesoarelor.

Au fost lansate două modele ale noului procesor, numite Athlon XP 2700+ și Athlon XP 2800+. Indicativul de performanță sugerează faptul că cel mai rapid procesor AMD este echivalent cu un Intel Pentium 4 la 2,8 GHz, chiar dacă frecvența de funcționare este doar cu puțin mai mare decât 2 GHz.

Din nefericire pentru AMD, noile procesoare sunt disponibile în număr limitat, iar utilizatorii obișnuiți le vor putea achiziționa doar la începutul anului 2003.

Următorul pas important al companiei AMD va fi făcut în cursul anului 2003 când se vor lansa primele procesoare care folosesc o arhitectură pe 64 de biți. O denumire probabilă a viitorului procesor este Athlon DT, iar indicativul de performanță va fi de peste 3000+.