SISTEME CU IA

RNA PENTRU CLASIFICARE SI RECUNOASTERE

Contextul general

Clasificarea - procesul de învățare a asemănărilor şi deosebirilor dintre instanțele unor obiecte dintr-o populație de obiecte neidentice.

Contextul general

DOUA etape

Clasificarea - sistemul învață caracteristicile generale ale claselor pe baza caracteristicilor specifice instanțelor .

Recunoaşterea - sistemul suprapune caracteristicile unei instanțe peste caracteristicile claselor cunoscute şi identifică clasa căreia aparține.

Contextul general

DOUA tipuri de invatare

Invatare supravegheata – se aplica atunci cand pentru fiecare model de intrare (instanta a obiectelor clasificate) se cunoaste apriori clasa la care se asociaza.

Invatare nesupravegheata- se aplica atunci cand nu se cunoaate in prealabil clasa la care se asociaza fiecare model de intrare.

Masuri de similaritate / disimilaritate

ABORDARE

Se porneste de la un set de vectori de intrare X = { x(1), x(2), ... , x(M) }, cu structura x(m) = (x1(m), x2(m), ... , xN(m)).

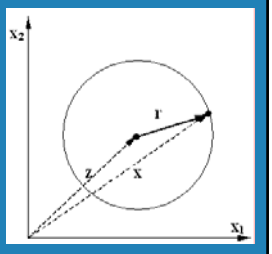
Se doreste separe a K clase – notate X1, X2, ... , XK – fiecare caracterizată de un prototip zk (k = 1,...,K) .

In final, pe baza celor K prototipuri zk se asociaza fiecare vector x(m) la una din cele K clase.

Masuri de similaritate / disimilaritate

Distanta Euclidiana – norma 2

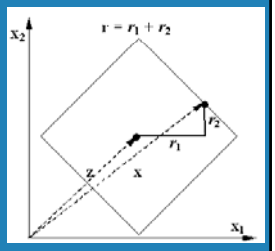
$$\| \mathbf{x} - \mathbf{z}^k \|_2 = \sqrt{\sum_{n=1}^{N} (x_n - z_n^k)^2}$$



Masuri de similaritate / disimilaritate

Distanta dupa norma 1

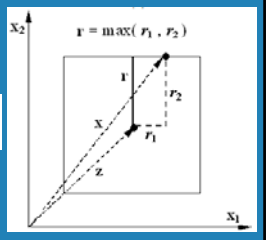
$$\| \mathbf{x} - \mathbf{z}^k \|_1 = \sum_{n=1}^{N} \left| x_n - z_n^k \right|$$



Masuri de similaritate / disimilaritate

Distanta dupa norma ∞

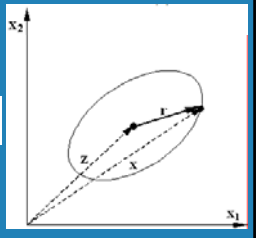
$$\| \mathbf{x} - \mathbf{z}^k \|_\infty = \max_{n=1,\ldots,N} \left(\left| x_n - z_n^k \right| \right)$$



Masuri de similaritate / disimilaritate

Distanta dupa ponderata (Mahalanobis)

$$\| \mathbf{x} - \mu \|_M = [(\mathbf{x} - \mu)^T \cdot C^{-1} \cdot (\mathbf{x} - \mu)]^{-1/2}$$



Masuri de similaritate / disimilaritate

Distanta dupa norma Hamming

$$\| \mathbf{x} - \mathbf{z} \|_H = \sum_{n=1}^{N} |x_n - z_n|$$

In cazul reprezentarii binare, distanta Hamming masoara numarul de componente diferite ale vecorilor x si z.

Pentru: x = (1101) şi z = (0110) se obtine:

$$\| x - z \|_H = |1-0| + |1-1| + |0-1| + |1-0| = 3$$

Algoritmi / Modele de clasificare

1. Algortimul celor mai apropiati K vecini
2. Algoritmul C-medii
3. Algoritmul ISODATA
4. Retele Kohonen (cu autoorganizare)
   - 4.1. Retele VQ – Vector Quantization
   - 4.2. Retele LVQ – Learning Vector Quantization
   - 4.3. Retele SOFM – Self Organizing Feature Maps

Algoritmul celor mai apropiati K vecini

Principiu

Se realizeaza selectarea dintre vectorii deja clasificați a primilor *K* (valoarea lui *K* este precizată a priori), cei mai apropiați de vectorul curent, iar acesta din urmă este asociat clasei dominante asociată celor *K* vectori de referință.

Algoritmul celor mai apropiati K vecini

Initializari

Se porneşte de la setul de învăţare format din vectorii $\{x^{(1)}, x^{(2)}, \ldots, x^{(M)}\}$ şi de la numărul de clase C în care se face clasificarea. Atunci când unul din aceşti vectori $x^{(m)}$ este asociat unei clase c, se foloseşte notaţia:

$Clasa[m] = c$

După reordonarea aleatorie a vectorilor $x^{(m)}$ din setul de antrenare se consideră arbitrar că primii C vectori sunt incluşi fiecare în una din cele C clase.

Algoritmul celor mai apropiati K vecini

Clasificarea propriu-zisa

În continuare, pentru fiecare din restul vectorilor $x^{(C+1)}, x^{(C+2)}, \ldots, x^{(M)}$ se parcurg următorii paşi:

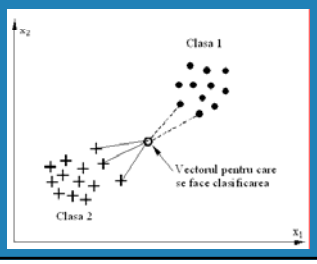
(a) se calculează distanţele faţă de vectorii deja clasificaţi;

(b) se ordonează crescător aceste distanţe şi se consideră vectorii corespunzători primelor K distanţe reordonate;

(c) pentru cei K vectori se determină clasa dominantă, iar vectorul curent se asociază acestei clase.

Algoritmul celor mai apropiati K vecini

Selectarea celor mai apropiaţi K vecini (cazul K = 5)



Algoritmul celor mai apropiati K vecini

Algoritmul

1. Date de intrare: modelele de antrenare, sub forma vectorilor de intrare $\mathbf{x}^{(m)}$, m = 1,..., M; parametrul k (numărul de vecini); numărul de clase C.
2. Reordonarea aleatorie a vectorilor de antrenare $\{\mathbf{x}^{(m)}\}$ m = 1,...,M.
3. Asocierea primelor C modele la cele C clase:
   for c = 1 to C do $Clasa(c) = c$
4. Asocierea claselor pentru restul modelelor de antrenare:
   for $m = C$ +1 to M do
   // Calculează distanțele față de vectorii deja clasificați
   for q = 1 to m - 1 do
   $d(q) = \| \mathbf{x}^{(m)} - \mathbf{x}^{(q)} \|$

Algoritmul celor mai apropiati K vecini

Algoritmul - continuare

// Ordonează crescător distanțele $d(q)$ prin reordonarea
// indecşilor q

for i = 1 to $m - 1$ do $lx(i) = i$
for i = 1 to $m - 2$ do
  for $j = i + 1$ to $m - 1$ do
    if $d(lx(j)) < d(lx(i))$ then $lx(j) \longleftrightarrow lx(i)$

Algoritmul celor mai apropiati K vecini

Algoritmul - continuare

// Calculează numărul de vectori asociați fiecărei clase
// pentru primele K distanțe $d(q)$, corespunzător
// indecşilor reordonați
$vmax = 0$;
for c = 1 to C do
  $v = 0$
  for j = 1 to k do
    if $Clasa(lx(j)) = c$ then $v = v + 1$
  if $v > vmax$ then
    $Clasa(m) = c$
    $vmax = v$
END.

Algoritmul celor mai apropiati K vecini

Observatii

(a) necesită precizarea a priori a numărului de clase ce urmează a fi separate;

(b) Rezultatele clasificarii sunt influentate de ordinea de prezentare a vectorilor;

(c) chiar principiul celor mai apropiați vecini compară vectorul curent cu vectori care, în majoritatea cazurilor, se găsesc către periferia zonelor

(d) nu stabileste vectori carateristici / prototipuri.

Algoritmul C - medii

Principiu

Algoritmul c-medii introduce pentru prima dată noțiunea de **prototip** sau **vector-centru** sau **vector de codare**, care descrie în mod unitar o clasă. Astfel, pentru o clasă **c** , prototipul $z^{(c)}$ se calculează ca medie aritmetică a celor **n (c)** vectori ce au fost asociați clasei respective:

$$\mathbf{z}^{(c)} = \frac{1}{n(c)} \sum_{m=1}^{M} \mathbf{x}^{(m)}$$

$z^{(c)} = x^{(m)}$ $\forall$ **m**, cu proprietatea **Clasa (m) = c**

Algoritmul C - medii

Initializare

Se porneşte de la setul de antrenare $\{x^{(1)}, x^{(2)}, \dots, x^{(M)}\}$ şi de la numărul de clase $C < M$ care trebuie separate. După reordonarea aleatorie a setului de antrenare, se atribuie arbitrar primele **C** modele de antrenare celor **C** clase, vectorii $x^{(1)}, \dots, x^{(C)}$ devenind prototipurile $z^{(1)}, \dots, z^{(C)}$.

Algoritmul C - medii

Clasificarea propriu-zisa

Fiecare din restul de **M – C** modele de antrenare este asociat unei clase, pe baza distanțelor minime față de cele **C** prototipuri.

Se recalculeaza prototipurile claselor şi se fac noi asocieri.

Procesul se reia, până când – în două iterații succesive – prototipurile claselor nu se schimbă sau se schimbă într-o măsură nesemnificativă.

Algoritmul C - medii

Cunatificarea preciziei

Evaluarea dimensiunilor zonei acoperite de fiecare clasă – abaterea medie pătratică dintre prototipul clasei şi vectorii din setul de antrenare asociați acesteia:

$$\sigma_c^2 = \frac{1}{n(c)} \sum_{m=1}^{M} \|\mathbf{x}^{(m)} - \mathbf{z}^{(c)}\|^2 \qquad \forall\, m, \text{ cu proprietatea } Clasa(m) = c$$

Abaterea patratica totala:

$$\sigma_T^2 = \sum_{c=1}^{C} \sigma_c^2$$

reprezintă o măsură a preciziei de clasificare.

Algoritmul C - medii

Algoritmul

1. Date de intrare: modelele de antrenare, sub forma vectorilor de intrare $\mathbf{x}^{(m)}$, $m = 1,\ldots, M$; numărul de clase C.
2. Reordonarea aleatorie a vectorilor de antrenare $\{\mathbf{x}^{(m)}\}$ $m = 1,\ldots,M$.
3. Asocierea primelor C modele la cele C clase:

   for $c = 1$ to C do

   $\mathbf{z}^{(c)} = \mathbf{x}^{(c)}$; $n(c) = 0$

Algoritmul C - medii

Algoritmul - continuare

4. Asocierea fiecărui model de antrenare la clasa cu prototipul cel mai apropiat:

```
for m = 1 to M do
    Dmin = 10^6                    // Inițializarea distanței minime
    for c = 1 to C do
        if ||x^(m) – z^(c)|| < Dmin then
            Dmin = ||x^(m) – z^(c)||
            Cmin = c
    Clasa(m) = Cmin
    n(Cmin) = n(Cmin) + 1
```

Algoritmul C - medii

Algoritmul - continuare

5. Calculul noilor prototipuri pentru cele C clase:

```
for c = 1 to C do
    w^(c) = 0
    for m =1 to M do
        if Clasa(m) = c then
            w^(c) = w^(c) + x^(m)
    w^(c) = w^(c) / n(c)        // Prototipuri temporare
```

Algoritmul C - medii

Algoritmul - continuare

6. Calculul abaterilor pătratice:

```
σ_T^2 = 0
for c = 1 to C do
    σ_c^2 = 0
    for m = 1 to M do
        if Clasa(m) = c then
            σ_c^2 = σ_c^2 + ||x^(m) – w^(c)||^2
    σ_c^2 = σ_c^2 / n(c)
    σ_T^2 = σ_T^2 + σ_c^2
```

Algoritmul C - medii

Algoritmul - continuare

7. Criteriul de oprire:

dacă prototipurile s – au modificat nesemnificativ – adică

$$||\mathbf{z}^{(c)} - \mathbf{w}^{(c)}|| < \varepsilon \text{ pentru toate centrele } c = 1,\ldots, C$$

algoritmul se încheie. În caz contrar, se memorează noile prototipuri:
for $c = 1$ to C do $\mathbf{z}^{(c)} = \mathbf{w}^{(c)}$
și se revine la pasul 4.

Algoritmul C - medii

Observatii

(a) necesită precizarea a priori a numărului de clase ce urmează a fi separate;
(b) nu este garantata convregenta;
(c) rezultate mai bune decat algoritmul celo mai apropiati K vecini
(d) Da rezultate bune, in special in cazul claselor net separabile.