

Cluster

Numim clasă (grup, cluster) o mulțime de obiecte (elemente) similare între ele și nesimilare obiectelor din alte clase.

Prin clasificare se înțelege gruparea unor entități (observații, obiecte etc.) în clase (grupuri) de entități similare. Atunci când gruparea este efectuată manual, cel

care o efectuează operează cu judecăți de similaritate, asemănare, apropiere. Acest tip de raționament este formalizat și în metodele automate.

Există, în esență, două tipuri de clasificare automată:

1. predictivă: se atribuie o observație la un grup pornind de la reguli de clasificare derivate din observații clasificate în prealabil.
2. descriptivă: se grupează obiectele pe baza similarității lor, nu este cunoscută o grupare prealabilă.

Clasificare predictivă

Considerăm situația clasificării propriu-zise, adică sunt cunoscute n obiecte prin atributele lor, inclusiv apartenența la clase, și se dorește clasarea unei noi observații.

Există mai multe metode de calcul a similarității între doi vectori. Printre cele mai cunoscute și folosite metode este calculul distanței Euclidiene.

O metrică sau “distanță” între doi vectori $D(\cdot, \cdot)$ trebuie să aibă patru proprietăți: pentru vectorii a, b și c :

non – negativitate: $D(a, b) \geq 0$

reflexivitate: $D(a, b) = 0$ dacă și numai dacă $a = b$

simetrie: $D(a, b) = D(b, a)$

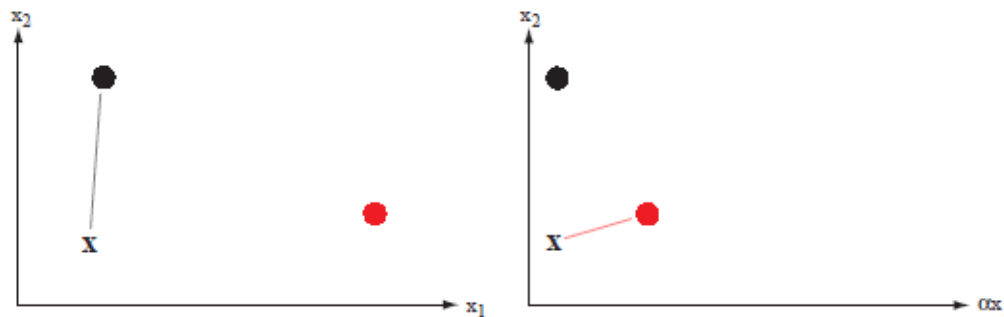
inegalitatea

triunghiului: $D(a, b) + D(b, c) \geq D(a, c)$ (suma laturilor AB și BC este \geq sau cel puțin egală cu lungimea celei de – a treia laturi AC)

Este ușor de verificat că distanța Euclidiană respectă aceste proprietăți.

Chiar mai mult, dacă fiecare coordonată este înmulțită cu o constantă arbitrară, și funcția rezultată îndeplinește cele 4 proprietăți, deși rezultatele aplicării algoritmului “Nearest-Neighbor” pot să fie în acest caz eronate.

În spațiul original, prototipul negru este cel mai apropiat vecin; în figura din dreapta, axa x a fost redimensionată cu un factor $1/3$; acum cel mai apropiat prototip este cel roșu.



Metoda celor mai apropiați k vecini (k - nearest neighbours)

Este o metodă utilizată pentru clasificare bazată pe o funcție de metrică sau “distanță” (funcție de similaritate) între doi vectori.

Se determină k obiecte cele mai apropiate de noua observație (k vecini, cei mai apropiați de înregistrarea ce se vrea a fi clasificată).

Metoda are nevoie de:

- setul de înregistrări cu clase cunoscute
- o metrică(distanță, funcție de similaritate) care calculează distanța dintre două înregistrări, pe baza valorilor atributelor
- valoarea k -numărul de vecini cei mai apropiați care sunt considerați

Pentru clasificarea unei înregistrări (pentru stabilirea clasei noului obiect):

- se calculează distanța către alte înregistrări din setul de antrenare
- se identifică cei mai apropiați k vecini

-se folosesc etichetele de clasă ale acestor k vecini pentru a estima clasa asociată înregistrării de test, fie prin:

- Vot majoritar –noul obiect este clasat în clasa la care aparțin cei mai mulți dintre cei k vecini (care dispun fiecare de un vot întreg), fie prin:

- Vot invers proporțional distanței –similar votului majoritar, dar fiecare dintre cei k vecini apropiați dispune de o fracțiune de vot, egală cu inversul distanței la noul obiect (obiectele mai apropiate contribuie mai mult la decizie).

•Matricea de proximitate

Elementele matricii reprezintă proximitățile dintre obiectele x și y . Proximitatea poate fi o similaritate (asemănare), cum ar fi coeficientul de corelație, sau o disociere (depărtare, diferențiere), cum ar fi distanța euclidiană.



• Calculul distanței: o alegere populară este distanța Euclidiană: pentru $\mathbf{x} = (x_1, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2)$$

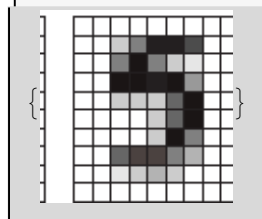
unde x și y reprezintă cei doi vectori/cele două înregistrări pentru care se calculează distanța, iar n reprezintă numărul de trăsături caracteristice.

Funcțiile de clasificare sunt estimate pe baza distanțelor dintre observații. Distanțele se pot calcula ca distanțe euclidiene, dar, din păcate variabile măsurate pe scale diferite, de ordine de mărime diferite, pot afecta foarte mult distanțele euclidiene.

Exemplu: Problema legată de “shift-area” la dreapta a cifrei 5, în încercarea de a identifica cifra 5:

```
i =  5; subimage = ;
```

```
Nearest[Flatten@ImagePartition[i, 100], subimage]
```



Clasificare descriptivă

Clasificarea descriptivă(cluster analysis) se referă la metodele utilizate pentru a identifica într-o mulțime de obiecte grupurile de obiecte similare.

Tabloul de date este amorf: nu există structurare a priori (dependențe funcționale, relații, clasificări cunoscute).

Această caracteristică este cea care ne depărtează de descrierea predictivă(unde se presupunea existența unei structurări necesare în etapa de training).

Drept rezultat al clasificării descriptive se obțin grupurile de elemente, clasele identificate.

Metodele de clasificare sunt de natură algoritmică: clasele apar ca urmare a unei suite de operații efectuate recursiv sau repetitiv.

Matricea de pattern-uri

Liniile sunt obiecte, iar coloanele sunt atribute (trasaturi, variabile); m obiecte și n atribute/trasaturi vor furniza o matrice de tip $m \times n$.

➤ Datele se reprezintă sub forma de vectori N -dimensionali

obiectul (exemplarul) i : $x_i = [x_{i1}, x_{i2}, \dots, x_{iN}]$, $x_i \in R^N$, $i = 1, \dots, M$

N – numărul de trăsături ale fiecarui obiect (dimensiunea trasaturilor)

M – numărul de obiecte (dimensiunea setului de date)

$$X = \begin{matrix} & \xrightarrow{\text{trasaturi, } j} & \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} & \begin{matrix} \downarrow \text{o} \\ \text{b} \\ \text{i} \\ \text{e} \\ \text{c} \\ \text{t} \\ \text{e} \\ \downarrow i \end{matrix} & \end{matrix}$$

Obiectivul gruparii este de a gasi

a) K vectori – **centre** grupurilor

$$c_K = [x_{k1}, x_{k2}, \dots, x_{kN}], \quad k = 1, \dots, K$$

b) Matricea **gradelor de apartenență**, cu dimensiunile $M \times K$ (M linii, K coloane)

Metode fuzzy

În afară de metodele deterministe, au fost dezvoltate și metode de clasificare fuzzy. Printr-o metodă fuzzy se obțin, pentru fiecare obiect, probabilitățile ca obiectul să aparțină fiecăruia dintre clustere. Rezultatul este conținut în matricea de apartenență care oferă probabilitățile apartenenței elementelor la clase. Partiționarea fuzzy se realizează iterativ (optimizând implicit funcția obiectiv) prin actualizarea la fiecare pas a matricei de apartenență și a centrelor clusterelor. Rezultă că problema esențială în determinarea (identificarea) clusterelor este cea a specificării proximității (apropierii, similarității) și cum se determină aceasta. Este evident că proximitatea este o noțiune dependentă de problema reală cercetată.

Metodele de clasificare fuzzy utilizează alegeri euristice ale “calității de membru” al unui cluster și regulile combinate euristice pentru a obține funcții discriminatorii. Astfel de tehnici se limitează la cazurile în care există foarte puține date și puține caracteristici.

- *Fuzzy c-means* (FCM) este o metoda de grupare a datelor în care fiecare obiect aparține unui grup într-un anumit grad, specificat de gradul de apartenență
- Metoda a fost introdusă de Jim Bezdek în 1981
- Este o metoda care arată **cum să se grupeze** obiectele ce populează un spațiu multidimensional *într-un număr specificat* de grupuri diferite.
- funcția *fcm* din Fuzzy Logic Toolbox pornește cu o **estimare inițială a centrelor grupurilor**, menite să marcheze locația medie a fiecărui grup.
- Estimarea inițială a centrelor este, cel mai probabil, incorectă.
- În plus, *fcm* atribuie inițial în mod aleator fiecărui obiect un grad de apartenență la fiecare grup
- Prin actualizarea iterativă a centrelor grupurilor și a gradelor de apartenență a tuturor obiectelor, *fcm* deplasează **iterativ** centrele în locațiile cele mai potrivite setului de date.
- Această iterare (optimizare) se bazează pe **minimizarea funcției obiectiv: suma ponderată a distanțelor fiecărui obiect la fiecare centru de grup, ponderile fiind gradele de apartenență a obiectelor la grupuri.**

Funcția obiectiv:

$$J = \sum_{k=1}^K \sum_{i=1}^M (\mu_{ik})^m \|x_i - c_k\|^2$$

m - constantă mai mare ca 1 (tipic: 2) ce indică gradul de nuanțare (fuzziness) a grupelor rezultate.

❖ **Minimizarea J** - suma distantelor pentru obiecte la centrele de grup, ponderate cu gradul de apartenență al obiectelor la grupuri.

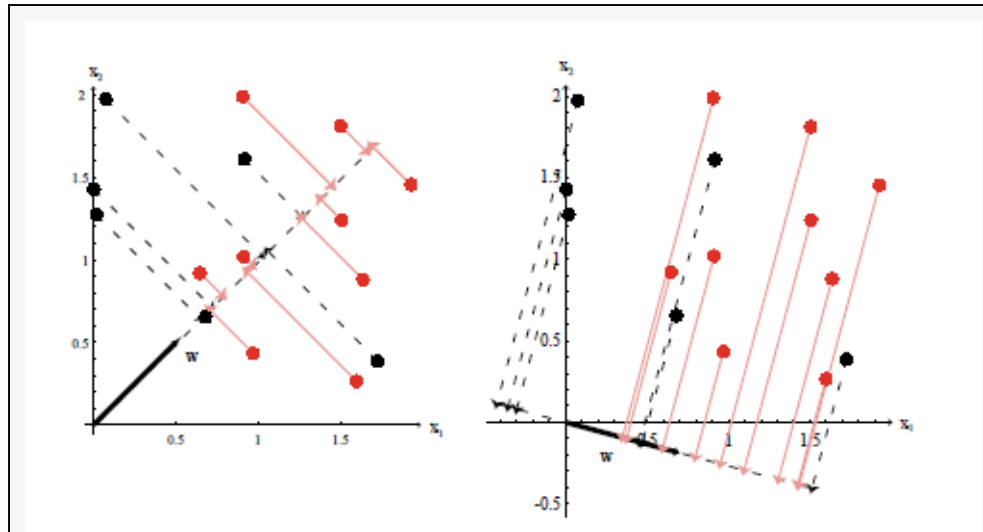
- Cu cât distanța este mai mică, cu atât ponderea (gradul de apartenență la acea clasă) va fi mai mare
- Cu cât distanța este mai mare, cu atât ponderea (gradul de apartenență la acea clasă) va fi mai mică

Gradele de apartenență:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^K \left(\frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{2/(m-1)}}$$

Analiza Discriminatorie Liniară (LDA) - Fisher Linear Discriminant

Una dintre problemele întâlnite în aplicarea tehnicilor statistice pentru problemele de pattern recognition a fost numită the “curse of dimensionality.” - “blestemul dimensionalității”. Procedurile care sunt administrabile analitic sau computațional în spații cu dimensiuni reduse pot deveni complet nepractice într-un spațiu de 50 sau 100 de dimensiuni.



Metodele fuzzy sunt deosebit de necorespunzătoare unor astfel de probleme de dimensiuni mari. Astfel, s-au dezvoltat diferite tehnici pentru reducerea dimensionalității spațiului caracteristic, cu speranța de a obține o problemă mai ușor de gestionat. Putem reduce dimensionalitatea de la dimensiunea d la o singură dimensiune dacă proiectăm datele d -dimensionale pe o linie. Desigur, chiar dacă probele s-au format bine separate, ca grupuri compacte în spațiu d , proiecția pe o linie arbitrară va produce, de obicei, un amestec confuz de eșantioane din toate clasele și, astfel, o performanță slabă de recunoaștere. Cu toate acestea, prin mutarea liniei, s-ar putea să găsim o orientare pentru care eșantioanele proiectate sunt bine separate.

În general, vom atașa un obiect la unul din grupurile predeterminate pe baza observațiilor pe care le facem cu privire la acest obiect. Dacă presupunem că grupurile sunt separabile liniar, putem folosi modelul discriminantului liniar (LDA). Proprietatea de separabilitate liniară sugerează că grupurile pot fi separate printr-o combinație de trasaturi care descriu obiectele. Dacă avem doar două trasaturi, separatorii vor deveni drepte. Dacă avem trei trăsături, separatorul devine un plan, și așa mai departe.

In[]:= **EuclideanDistance**[{2.5, 4.3}, {2.1, 5.6}]

Out[]:= 1.36015

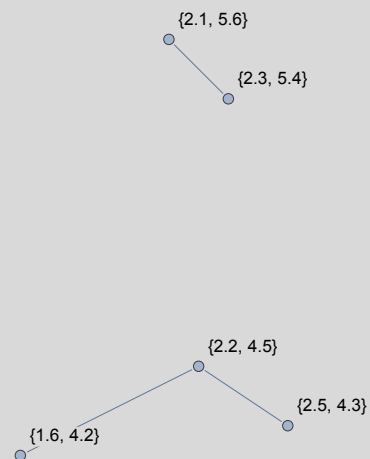
In[]:= **DistanceMatrix**[{{2.5, 4.3}, {2.1, 5.6}, {2.2, 4.5}, {1.6, 4.2}, {2.3, 5.4}}]

Out[]//MatrixForm=

$$\begin{pmatrix} 0. & 1.36015 & 0.360555 & 0.905539 & 1.11803 \\ 1.36015 & 0. & 1.10454 & 1.48661 & 0.282843 \\ 0.360555 & 1.10454 & 0. & 0.67082 & 0.905539 \\ 0.905539 & 1.48661 & 0.67082 & 0. & 1.38924 \\ 1.11803 & 0.282843 & 0.905539 & 1.38924 & 0. \end{pmatrix}$$

In[]:= **NearestNeighborGraph**[{{2.5, 4.3}, {2.1, 5.6}, {2.2, 4.5}, {1.6, 4.2}, {2.3, 5.4}}]

Out[]:=



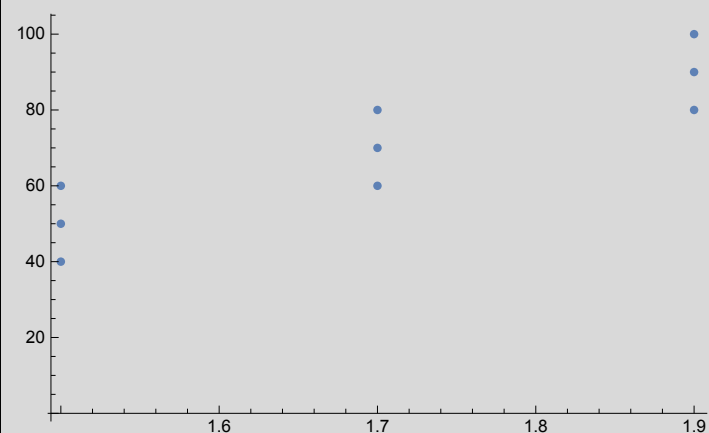
• Problemă legată de ordinul de mărime a datelor:

- avem două atribute: înălțimea și greutatea unor persoane
- înălțimea e măsurată în metri; intervalul poate fi de ex $[1.50\text{ m}, 2.00\text{ m}]$, deci cu o diferență de maxim 0.5
- greutatea se măsoară în kilograme; intervalul poate fi $[50\text{ kg}, 200\text{ kg}]$
- diferențele de greutate domină pe cele în înălțime; o diferență de 1 kg este mai mare decât orice diferență de înălțime, contribuind deci prea mult la calculul distanței

In[]:=

```
ListPlot[{{1.50, 40}, {1.50, 50}, {1.50, 60}, {1.70, 60}, {1.70, 70}, {1.70, 80}, {1.90, 80}, {1.90, 90}, {1.90, 100}}]
```

Out[]:=



Dar dacă folosim o funcție de similaritate f , de exemplu indicele IMC = greutate (kg) / înălțime(m) ², atunci:

In[]:= $f[h_, k_] = k / (h^2)$

Out[]:=
 $\frac{k}{h^2}$

In[]:= $f[1.50, 40]$

Out[]:= 17.7778

In[]:= `ListPlot[FindClusters[{{1.50, f[1.50, 40]}, {1.50, f[1.50, 50]}, {1.50, f[1.50, 60]}, {1.70, f[1.70, 60]}, {1.70, f[1.70, 70]}, {1.70, f[1.70, 80]}, {1.90, f[1.90, 80]}, {1.90, f[1.90, 90]}, {1.90, f[1.90, 100]}}, DistanceFunction -> EuclideanDistance]]`

