CLASIFICAREA NESUPERVIZATĂ FUZZY

1. Introducere

Algoritmii de clasificare nesupervizată sunt utilizați în mod considerabil, nu numai pentru organizarea și categorisirea datelor, ci și pentru compresia datelor si construcția de modele. În cadrul acestui capitol sunt prezentați patru dintre cei mai reprezentativi algoritmi de clusterizare off-line utilizați în conjuncție cu rețele neurale cu funcții radiale și modelare fuzzy: C-Means (sau K-Means), C-Means fuzzy, clasificarea folosind metoda Mountain si clusterizarea prin îndepărtare.

Clasificarea împarte setul de date în mai multe grupuri, de așa manieră ca similaritatea în cadrul unui grup să fie mai mare ca similaritatea între grupuri. Obținerea unei astfel de partiționări necesită măsuri de similaritate ce primesc ca intrări doi vectori și returnează o valoare care le reflectă similaritatea. Deoarece majoritatea măsurilor de similaritate sunt sensibile la gama de valori a mărimilor de intrare, fiecare variabilă de intrare va trebui normalizată pentru a se încadra în intervalul $[0, 1]$. În continuare, setul de date se consideră a fi normalizat la hipercubul unitate.

Tehnicile de clasificare sunt folosite împreună cu rețelele neurale cu funcții radiale sau modelare fuzzy pentru a determina locațiile inițiale pentru funcțiile radiale sau regulile fuzzy if-then. Pentru aceasta, tehnicile de clasificare sunt validate pe baza următoarelor ipoteze:

1) Intrările similare trebuie să producă ieșiri similare în cadrul sistemului vizat;
2) Perechile similare de tip intrare-ieșire sunt grupate în cadrul clusterelor în setul de date de antrenare.

Prima ipoteză specifică faptul că sistemul vizat trebuie să modeleze intrările și ieșirile într-o manieră omogenă; lucru cunoscut pentru sistemele reale. Cea de-a doua ipoteză necesită ca setul de date să aibă o anumită distribuție specifică; însă nu este întotdeauna necesar.

De aceea, tehnicile de clasificare folosite în identificarea structurilor neurale sau în modelarea de tip fuzzy sunt euristice, iar existența unui set de date la care aceste tehnici să nu poată fi aplicate cu succes nu este o noutate.

2. K-medii (K-Means)

K-Medii cunoscut și sub denumirea C-Medii a fost aplicată într-o varietate de domenii, inclusiv în procesarea de imagini sau compresia de date, preprocesarea datelor pentru modelarea sistemelor cu rețele cu funcții radiale și descompunerea taskurilor în cadrul arhitecturilor de rețele neurale heterogene.

Algoritmul K-Means împarte o colecție de n vectori $x_j, j = 1, \dots, n$ în c grupuri $G_i, i = 1, \dots, c$ și găsește un centru în cadrul fiecărui grup astfel încât o funcție cost (sau o funcție obiectiv)

reprezentând măsura de disimilaritate din cadrul grupului respectiv să fie minimizată. Atunci când este folosita norma Euclidiană, funcția cost poate fi definită dupa cum urmează:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k, x_k \in G_i} \|x_k - c_i\|^2 \right) \quad (1)$$

unde $J_i = \sum_{k, x_k \in G_i} \|x_k - c_i\|^2$) este funcția cost din cadrul grupului i. Totuși, valoarea lui J_i depinde de proprietățile geometrice ale lui G_i precum și de locația lui c_i.

De obicei, unui vector x_k i se poate aplica o funcție generică $d(x_k, c_i)$ în cadrul grupului i; iar funcția cost globală poate fi exprimată ca:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k, x_k \in G_i} d(x_k - c_i) \right) \quad (2)$$

Pentru simplitate, se utilizează ca măsură a disimilarității norma Euclidiană și funcția cost globală detaliată în (1).

Grupurile partiționate sunt definite de obicei de o matrice de apartenență binară U de dimensiune $c \times n$, în cadrul căreia, fiecare element u_{ij} este 1 dacă x_j aparține grupului i și 0 în caz contrar. Odată ce se determină centrul unui cluster, u_{ij} poate fi exprimat ca:

$$u_{ij} = \begin{cases} 1 & , dacă \ \|x_j - c_i\|^2 \leq \|x_j - c_k\|^2 \\ 0 & , în \ caz \ contrar \end{cases} \quad (3)$$

Reformulând, x_j aparține grupului i dacă c_i este cel mai apropiat centru din mulțimea centrilor. Dat fiind că un punct se poate regăsi într-un singur grup, matricea de apartenență U are următoarele proprietăți:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \dots, n \quad (4)$$

și

$$\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n \quad (5)$$

Pe de altă parte, dacă u_{ij} este fixat, atunci centrul optimal c_i care minimizează ecuația este media vectorilor din grupul i:

$$c_i = \frac{1}{|G_i|} \sum_{k, x_k \in G_i} x_k \quad (6)$$

unde $|G_i|$ este dimensiunea lui G_i sau $|G_i| = \sum_{j=1}^{n} u_{ij}$.

Pentru modul de operare în bloc, algoritmul K-Means este aplicat pe setul de date $x_i, i = 1, \dots, n$; algoritmul determină centrii c_i ai clusterelor precum și matricea de apartenență în mod iterativ aplicând pașii descriși în cele ce urmează:

Pasul 1 Se inițializează centrii clusterelor $c_i, i = 1, \dots, c$ prin selecția aleatoare de puncte din cadrul setului.

Pasul 2 Se determină matricea de apartenență U cu ajutorul ecuației (3).

Pasul 3 Se calculează funcția cost conform ecuației (1). Algoritmul de oprește atunci când

valoarea funcției cost este sub pragul de toleranță sau progresul înregistrat în raport cu iterația precedentă este sub o limită permisă.

Pasul 4 Se actualizează centrii clusterelor în conformitate cu (6). Se revine la Pasul 2.

Algoritmul este unul iterativ și nu garantează convergența către o soluție optimă. Performanța algoritmului K-Means depinde de pozițiile inițiale ale centrilor clusterelor, din acest motiv, se recomandă fie utilizarea unor metode frontale pentru selectarea unor centrii potriviți fie execuția succesivă a algoritmului de fiecare dată cu alt set de centri. În plus, algoritmul precedent este doar unul ilustrativ, fiind posibilă și o inițializare aleatoare a matricii de apartenență urmată de o procedură iterativă.

Algoritmul K-Means poate să opereze și în modul on-line, caz în care centrii clusterelor și grupările corespunzătoare sunt obținute prin mediere. De aceea, pentru un punct dat x, algoritmul găsește cel mai apropiat centru al unui cluster c_i și îl actualizează după formula:

$$\Delta c_i = \eta(x - c_i) \tag{7}$$

3. Clasificarea nesupervizată C-Medii fuzzy

Clasificarea C-Means fuzzy (sau FCM), cunoscută și sub denumirea de fuzzy ISODATA constă într-un algoritm de clasificare în cadrul căruia fiecare punct aparține unui cluster într-o anumită măsură, specificată printr-un grad de apartenență (o valoare între 0 și 1).

FCM împarte o colecție de n vectori $x_i, i = 1, \dots, n$ în c grupuri fuzzy și identifică centrii fiecărei grupări astfel încât o funcție cost, măsură a disimilarității dintre grupări să fie minimizată. Diferența majoră între FCM și C-Means este faptul că FCM utilizează partiții fuzzy astfel încat un anumit punct poate aparține mai multor grupuri, cu un grad de apartenență între 0 și 1. Pentru a ajusta introducerea partiției fuzzy, matricea de apartenență U va conține valori între 0 și 1. Oricum, normalizarea impune ca suma gradelor de apartenență pentru un set de date să fie unitară.

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \dots, n \tag{8}$$

Funcția cost sau funcția obiectiv va fi în acest caz o generalizare a funcției prezentate în (1).

$$J(U, c_1, \dots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{9}$$

unde u_{ij} este cuprins între 0 și 1; c_i este centrul grupului fuzzy i; $d_{ij} = \|c_i - x_j\|$ este norma Euclidiană între clusterul i și punctul j din setul de date; iar $m \in [1, \infty)$ este un exponent de ponderare.

Condițiile necesare pentru ca ecuația (9) să își atingă minimul pot fi găsite prin construirea unei noi funcții obiectiv $\bar{J}$ după cum urmează:

$$\bar{J}(U, c_1, \dots, c_c, \lambda_1, \dots, \lambda_n) = J(U, c_1, \dots, c_c) + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{c} u_{ij} - 1 \right) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 + \\ + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{c} u_{ij} - 1 \right) \tag{10}$$

unde $\lambda_j, j = 1, \dots n$ sunt multiplicatorii Lagrange pentru cele n constrângeri din ecuația (8). Prin dieferențierea lui $\bar{J}$ în raport cu argumentele sale, condițiile necesare ca (9) să își atingă minimul sunt:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_x}{\sum_{j=1}^{n} u_{ij}^m} \tag{11}$$

și

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{\frac{2}{m-1}}} \tag{12}$$

Algoritmul de clasificare C-Means fuzzy este o simplă procedură iterativă ce verifică condițiile necesare prezentate anterior. În modul de operare în lot, FCM determină centrii c_i ai clusterelor și matricea de apartenență U parcurgând următoarele etape:

Pasul 1 Se inițializează matricea de apartenență U cu valori aleatoare cu prinse între 0 și 1 astfel încât să fie satisfăcută relația (8).

Pasul 2 Se calculează cei c centrii fuzzy $c_i, i = 1, \dots, c$ utilizând relația (11).

Pasul 3 Se calculează funcția cost în conformitate cu ecuația (9). Algoritmul se oprește fie dacă valoarea funcției cost este sub un prag acceptat fie progresul înregistrat în raport cu iterația precedentă este sub o valoare permisă.

Pasul 4 Se calculează o nouă matrice U folosind ecuația (12) și se revine la Pasul 2.

Centrii clusterelor pot fi deasemenea inițializați înainte de aplicarea procedurii iterative. Nu există nici o garanție că FCM va converge către o soluție optimă. Performanțele sale depind de centrii inițiali ai clusterelor, din acest motiv, este permisă folosirea unui algoritm rapid pentru determinarea acestora sau este recomandată rularea succesivă a algoritmului FCM pentru seturi diferite de centrii.

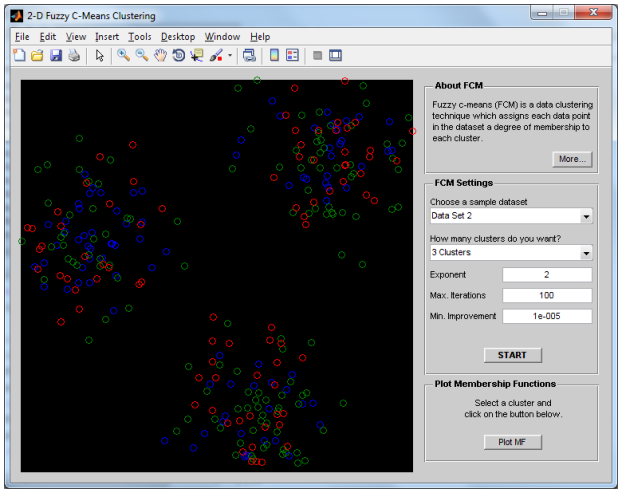

Figura 1. Programul demo pentru clasificarea C-Means fuzzy (Comanda MATLAB `fcmdemo`)

Figura 1. prezintă o aplicație demo a metodei C-Means fuzzy din toolbox-ul Fuzzy Logic Toolbox. Din interfața de configurare pot fi selectate: setul de date, numărul de clustere, exponentul de ponderare și diverse criterii de stop. La apăsarea butonului ”Start” se poate observa deplasarea centrilor clusterelor către pozițiile corecte. În special, prin intermediul etichetei ”Label Data” se poate observa evoluția fiecărui grup atunci când centrii se deplasează. La finalizarea procesului de clasificare, centrul unui cluster poate fi selectat, operație care va determina afișarea gradelor de apartenență a tuturor punctelor către centrul respectiv. Figura 2. ilustrează graficele MF aferente celor trei centri ai clusterelor. Gradele de apartenență sunt definite doar pentru locațiile punctelor din setul de date; iar graficele suprafețelor din figura 2. sunt obținute prin interpolare 2-D cu comanda MATLAB **griddata**.

Domeniile de aplicare ale clasificării C-Means fuzzy sunt segmentarea imaginilor medicale și modelarea calitativă.

4. Clasificarea prin metoda Mountain

Clasificarea utilizând metoda Mountain este o abordare relativ simplă și eficientă a estimării centrilor clusterelor pe baza măsurii densității numită funcție mountain. Această metodă poate fi folosită pentru a obține valorile inițiale pentru centri necesare unui algoritm mai sofisticat, ca de exemplu FCM prezentat în secțiunea anterioară. Poate fi folosită deasemenea si ca o metodă independentă de aproximare a clasificării. Această metodă se bazează pe formarea vizuală a clusterelor dintr-un set de date.

Prima etapă implică construirea unui grid în spațiul setului de date, în cadrul căruia intersecțiile liniilor constituie condidații pentru centri, notați prin V. Un grid de calitate mai bună va crește atât numărul de potențiali centri cât și calculele necesare. Împărțirea se face cu spațiere egală, dar nu este obligatorie.

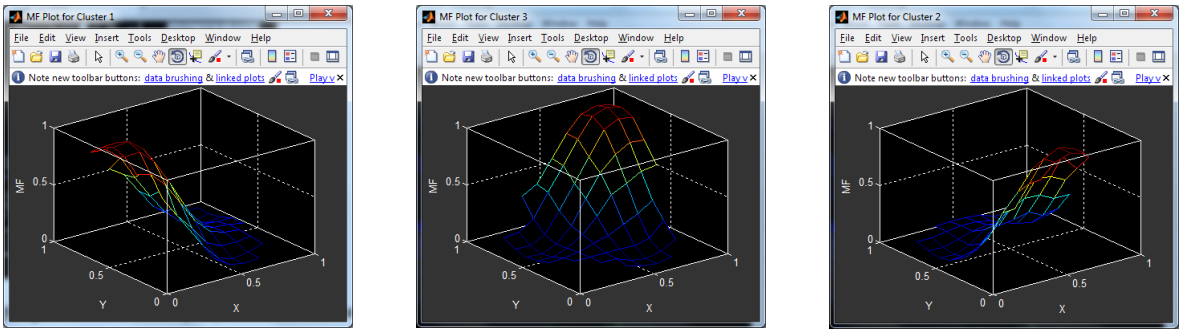

Figura 2. Graficele MF pentru aplicația demo prezentată în figura A.F.1.

Pot exista și griduri inegal spațiate pentru a reflecta cunoașterea a priori a distribuției datelor. Mai mult, dacă însuși setul de date este utilizat în locul gridului pentru selecția centrilor, atunci clasificarea va fi prin îndepărtare si va fi prezentată în secțiunea următoare.

Al doilea pas determină construirea funcției mountain ca o măsură a densității datelor. Înălțimea acesteia la un anumit punct $v \in V$ este egală cu:

$$m(v) = \sum_{i=1}^{N} e^{-\frac{\|v-x_i\|^2}{2\sigma^2}} \tag{13}$$

unde x_i este un punct de date și σ este o constantă specifică aplicației. Valoarea funcției m depinde de fiecare punct și invers proporțional de distanța dintre x_i și v. Funcția m poate fi văzută ca o măsură a densității datelor deoarece crește o dată cu numărul de puncte localizate în apropiere și scade dacă în vecinătate nu există puncte. Constanta σ determină atât valoarea maximă, cât și netezimea funcției rezultate; acest lucru fiind demonstrat în exemplul 1. Rezultatele clasificării sunt în mod obișnuit independente de valoarea parametrului σ atât timp cât dimensiunea setului de date este suficient de mare iar clasificarea se realizează corect.

Al treilea pas implică selecția centrilor clusterelor prin fragmentarea secvențială a funcției m. Se găsește un punct dintre centrii candidați V care are valoarea maximă și se consideră a fi primul centru c_1. (În caz că există mai multe astfel de puncte, se va selecta aleator unul ca fiind centrul primului cluster.) Obținerea următorului centru necesită eliminarea efectului centrului recent identificat, care este de obicei înconjurat de un număr de puncte care au deasemenea valori mari ale funcției m. Această problemă este rezolvată prin ajustarea funcției m, prin scăderea unei funcții Gaussiene scalate centrate în c_1:

$$m_{new}(v) = m(v) - m(c_1)e^{-\frac{\|v-c_1\|^2}{2\beta^2}} \tag{14}$$

Cantitatea scăzută este invers proporțională cu distanța dintre v și centrul recent identificat c_1 și direct proporțional cu valoarea funcției în c_1. După ajustarea funcției, este selectat următorul centru dintre punctele din V care au valoarea maximă a funcției m. Procesul de revizuire a funcției m și de găsire a următorilor centri continuă până la descoperirea unui număr suficient de centri. Următorul exemplu clarifică procesul.

Exemplul 1. Metoda de clusterizare mountain pentru date bidimensionale.

Figura 3.a. prezintă un set de date bidimensionale și trei clustere ușor de observat. În orice caz, pentru seturi de date de mari dimensiuni, nu există metode eficiente de determinare vizuală a clusterelor. În cadrul acestui exemplu, metoda Mountain este utilizată pentru descoperirea centrilor clusterelor. Pentru a demonstra efectele lui σ, în figurile 3.b-d. sunt reprezentate funcțiile m pentru σ egal cu 0.02, 0.1 și 0.2. Cu certitudine σ influențează valoarea maximă și netezimea funcției m; așadar ar trebui selecționată cu grijă considerând atât dimensiunea setului de date cât și dimensiunea datelor.

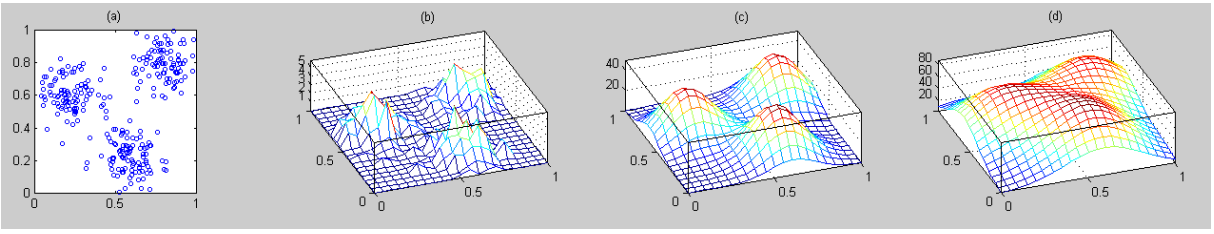

Figura 3. Construcția funcției m: a) pentru un set de date bidimensional și următoarele valori ale parametrului σ: (a) 0.02; (b) 0.1; și (c) 0.2. (Comanda MATLAB **`mount1`**).

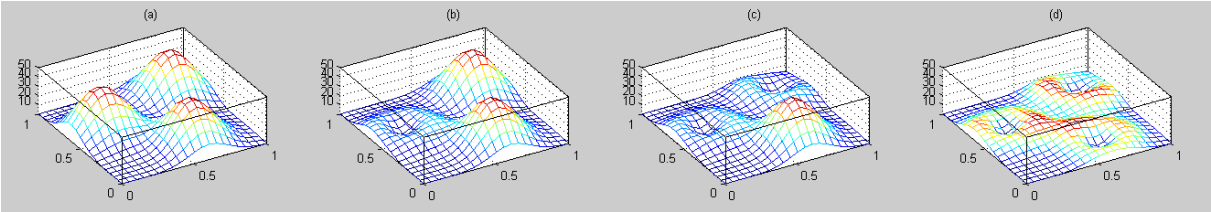

Figura 4. Ajustarea funcției m cu $\beta = 0.1$: (a) funcția m originală pentru $\sigma = 0.1$; (b) funcția m după prima ajustare; (c) funcția m după cea de-a doua ajustare; (d) funcția m după cea de-a treia ajustare. (Comanda MATLAB **mount2**).

Odată determinat σ (0.1 în cazul de față) și având construită funcția m, începe selecția clusterelor și revizuirea secvențială a funcției m. Acest proces este prezentat în figurile 4.a-d cu $\beta = 0.1$ pentru ecuația (14).

Clasificarea folosind metoda Mountain a fost aplicată în identificare sau în modelare de tip fuzzy pe un set de date de antrenare de tipul intarări și ieșiri dorite pentru găsirea centrilor (x_i, y_i) și apoi formarea unui model fuzzy Sugeno de ordin zero în cadrul căruia regula i este de forma:

$$IF\ X\ este\ aproape\ de\ x_i\ THEN\ Y\ este\ aproape\ de\ y_i. \quad (15)$$

Reformulată, regula are la bază centrul clusterului i identificat prin metoda Mountain. După determinarea acestei structuri, pot fi aplicate scheme de optimizare de tip backpropagation sau alte variante pentru a trece la identificarea parametrilor.

5. Clasificarea prin reducere

Metoda Mountain descrisă la secțiunea anterioară este relativ simplă și eficientă. Însă, în acest caz, calculele cresc în mod exponențial în funcție de dimensiunea problemei deoarece metoda trebuie să evalueze funcția m în toate punctele de pe grid. De exemplu, se consideră o problemă de clasificare cu patru variabile pentru care fiecare dimensiune are o rezoluție de 10 linii în cadrul gridului va determina un număr de 10^4 puncte din grid care vor trebui evaluate. O abordare alternativă o constituie clasificarea prin metoda reducerii, în cadrul căreia datele sunt considerate a fi candidați pentru centrii clusterelor. Prin utilizarea acestei metode, complexitatea computațională este direct proporțională cu numărul de centri și independentă de dimensiunea problemei considerate.

Se consideră o colecție de n puncte $\{x_1, \dots, x_n\}$ într-un spațiu M-dimensional. Fără a particulaiza, punctele respective se consideră a fi normalizate la hipercubul unitate. Deoarece fiecare punct constituie un candidat posibil pentru unul din centri, se definește o măsură a densității pentru fiecare punct ca fiind:

$$D_i = \sum_{j=1}^{n} e^{-\frac{\|x_i - x_j\|^2}{\left(\frac{r_a}{2}\right)^2}} \quad (16)$$

unde r_a este o constantă pozitivă. Astfel, un punct va avea o valoare mare a densității dacă în vecinătatea sa se va găsi un număr mare de puncte. Raza r_a definește o vecinătate; punctele aflate în afara acestei vecinătăți influențând foarte puțin funcția D.

După calcularea valorilor funcției densitate pentru fiecare punct în parte, punctul de densitate maximă va fi selectat ca centrul primului cluster. Fie x_{c_1} punctul selectat și D_{c_1} densitatea în punctul respectiv. În continuare, funcția densitate va fi revizuită pentru fiecare punct x_i după formula:

$$D_i = D_i - D_{c_1} \cdot e^{-\frac{\|x_i - x_{c_1}\|^2}{\left(\frac{r_b}{2}\right)^2}} \tag{17}$$

unde r_b este o constantă pozitivă. Astfel, punctele situate în vecinătatea primului centru x_{c_1} vor avea valori ale funcției densitate reduse semnificativ, lucru care le împiedică în a mai fi selectate în etapele următoare. Constanta r_b definește o vecinătate în care funcția densitate a fost redusă considerabil. În mod obișnuit $r_b > r_a$ pentru a preveni apariția unor clustere ale căror centri sunt apropiați spațial. Uzual se consideră $r_b = 1.5 r_a$.

După revizuirea valorilor funcției densitate pentru fiecare punct, este selectat următorul centru x_{c_2} după care procedura se repetă până când este descoperit un număr suficient de mare de clustere. Un criteriu de stop ceva mai sofisticat determină în mod automat numărul de clustere ce pot fi descoperite.

La aplicarea clasificării prin reducere pe un set de date, fiecare din centrii identificați reprezintă un prototip reprezentativ pentru caracteristicile sistemului modelat. Acești centri vor fi folosiți convenabil pentru premisele regulilor fuzzy dintr-un model Sugeno de ordin zero sau ca funcții radiale într-o rețea neurală cu funcții radiale. De exemplu, se consideră centrul clusterului i în dimensiunea M. Acesta poate fi descompus în doi vectori p_i și q_i, unde p_i este partea de intrare și conține primele N elemente ale lui c_i și q_i reprezintă partea aferentă ieșirii și conține ultimele $M - N$ elemente ale lui c_i. Se dă un vector de intrare x, gradul de apartenență la o regulă fuzzy fiind dat de:

$$\mu_i = e^{-\frac{\|x - p_i\|^i}{\left(\frac{r_a}{2}\right)^2}} \tag{18}$$

Aceasta este definiția funcției radiale i în cazul abordării perspectivei modelării folosind rețele neurale cu funcți radiale. După ce au fost determinate funcțiile radiale, ponderile aferente ieșirilor rețelei pot fi determinate prin metroda celor mai mici pătrate. După aplicarea acestor proceduri, se poate obține o mai mare acuratețe prin utilizarea metodei gradientului sau a altor scheme de optimizare derivative avansate.

6. Concluzii

În cadrul acestui capitol au fost prezentate patru dintre cele mai cunoscute tehnici de clasificare off-line utilizate în conjuncție cu rețele neurale cu funcții radiale și modelare fuzzy. Aceste tehnici de clasificare constituie abordări de tip în lot în identificarea prototipurilor ce caracterizează

un set de date; aceste prototipuri fiind polosite ca centri pentru rețele neurale cu funcții radiale sau ca reguli fuzzy. Pentru compresia de date, aceste prototipuri sunt folosite în cuantizarea vectorilor.

7. Anexă

```
% mount1.m

data_n = 100;
% === cluster 1
c1 = 0.6 + j*0.2;
data1 = c1 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === cluster 2
c2 = 0.2 + j*0.6;
data2 = c2 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === cluster 3
c3 = 0.8 + j*0.8;
data3 = c3 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === final data
data = [data1; data2; data3];
tmp1 = (real(data)<0) | (real(data)>1);
tmp2 = (imag(data)<0) | (imag(data)>1);
index = tmp1 | tmp2;

data(find(index == 1), :) = [];

subplot(3,4,1);
h = plot(data, 'o');
set(h, 'markersize', 3);
axis equal; axis square;
axis([0 1 0 1]);
title('(a)');

point_n = 21;
x = linspace(0, 1, point_n);
y = linspace(0, 1, point_n);
[xx, yy] = meshgrid(x, y);
grid_point = xx(:) + j*yy(:);

grid_n = size(grid_point, 1);
data_n = size(data, 1);

tmp1 = grid_point*ones(1, data_n);
tmp2 = ones(grid_n, 1)*data.';
tmp = tmp1 - tmp2;
dist = sqrt(real(tmp).^2 + imag(tmp).^2);

subplot(3,4,2);
sigma = 0.02;
tmp = exp(-dist.*dist/(2*sigma^2));
zz = reshape(sum(tmp'), point_n, point_n);
mesh(xx, yy, zz);
view([-20 60]);
set(gca, 'box', 'on');
title('(b)');
axis([-inf inf -inf inf -inf inf]);

subplot(3,4,3);
sigma = 0.1;
tmp = exp(-dist.*dist/(2*sigma^2));
zz = reshape(sum(tmp'), point_n, point_n);
mesh(xx, yy, zz);
view([-20 60]);
set(gca, 'box', 'on');
title('(c)');
axis([-inf inf -inf inf -inf inf]);

subplot(3,4,4);
sigma = 0.2;
tmp = exp(-dist.*dist/(2*sigma^2));
zz = reshape(sum(tmp'), point_n, point_n);
```

```
mesh(xx, yy, zz);
view([-20 60]);
set(gca, 'box', 'on');
title('(d)');
axis([-inf inf -inf inf -inf inf]);
```

```
% mount2.m

data_n = 100;
% === cluster 1
c1 = 0.6 + j*0.2;
data1 = c1 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === cluster 2
c2 = 0.2 + j*0.6;
data2 = c2 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === cluster 3
c3 = 0.8 + j*0.8;
data3 = c3 + (randn(data_n,1) + j*randn(data_n,1))/10;
% === final data
data = [data1; data2; data3];
tmp1 = (real(data)<0) | (real(data)>1);
tmp2 = (imag(data)<0) | (imag(data)>1);
index = tmp1 | tmp2;

data(find(index == 1), :) = [];

point_n = 21;
x = linspace(0, 1, point_n);
y = linspace(0, 1, point_n);
[xx, yy] = meshgrid(x, y);
grid_point = xx(:) + j*yy(:);

grid_n = size(grid_point, 1);
data_n = size(data, 1);

tmp1 = grid_point*ones(1, data_n);
tmp2 = ones(grid_n, 1)*data.';
tmp = tmp1 - tmp2;
dist = sqrt(real(tmp).^2 + imag(tmp).^2);

subplot(3,4,1);
sigma = 0.1;
mountain = sum(exp(-dist.*dist/(2*sigma^2))')';
zz = reshape(mountain, point_n, point_n);
mesh(xx, yy, zz);
view([-30 60]);
set(gca, 'box', 'on');
title('(a)');
max_mountain = max(mountain);
min_mountain = min(mountain);
axis([-inf inf -inf inf min_mountain max_mountain]);

subplot(3,4,2);
[max_height, max_index] = max(mountain);
beta = 0.1;
tmp = grid_point - grid_point(max_index);
dist1 = sqrt(real(tmp).^2 + imag(tmp).^2);
to_be_removed = max_height*exp(-dist1.*dist1/(2*beta^2));
mountain = mountain - to_be_removed;
zz = reshape(mountain, point_n, point_n);
mesh(xx, yy, zz);
view([-30 60]);
set(gca, 'box', 'on');
title('(b)');
axis([-inf inf -inf inf min_mountain max_mountain]);

subplot(3,4,3);
[max_height, max_index] = max(mountain);
beta = 0.1;
tmp = grid_point - grid_point(max_index);
dist1 = sqrt(real(tmp).^2 + imag(tmp).^2);
to_be_removed = max_height*exp(-dist1.*dist1/(2*beta^2));
mountain = mountain - to_be_removed;
zz = reshape(mountain, point_n, point_n);
```

```
mesh(xx, yy, zz);
view([-30 60]);
set(gca, 'box', 'on');
title('(c)');
axis([-inf inf -inf inf min_mountain max_mountain]);

subplot(3,4,4);
[max_height, max_index] = max(mountain);
beta = 0.1;
tmp = grid_point - grid_point(max_index);
dist1 = sqrt(real(tmp).^2 + imag(tmp).^2);
to_be_removed = max_height*exp(-dist1.*dist1/(2*beta^2));
mountain = mountain - to_be_removed;
zz = reshape(mountain, point_n, point_n);
mesh(xx, yy, zz);
view([-30 60]);
set(gca, 'box', 'on');
title('(d)');
axis([-inf inf -inf inf min_mountain max_mountain]);
```

8. Probleme propuse

Problema 1.

Se consideră consumul de energie termică dintr-o scară de bloc pe perioadă de o lună de zile. Se monitorizează cantitatea de căldură măsurată Qm, cantitatea de căldură transferată prin pereții apartamentului către exteriorul blocului Qtrext, cantitatea de căldură trnasferată prin pereții apartamentului către casa scării Qtrint, cantitatea de căldură care se pierde prin sistemul de ventilație Qv și cantitatea de căldură provenită din energia solară Gs. Datele numerice sunt stocate în fișierul data1.

Să se scrie codul matlab pentru clasificarea nesupervizată utilizând metoda fuzzy c-means a apartamentelor din bloc în trei clase după consumul de energie termică.