UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

PRIMJENA NADZIRANOG I NENADZIRANOG UČENJA U MATLAB-U

1. *Nadzirano učenje -> klasifikacija, regresija (binarna klasifikacija, višeklasna, algoritmi mašinskog učenja kao što su kNN, SVM, linearna regresija, logistička regresija itd.).*
2. *Nenadzirano učenje -> klastering.*

Iris je poznati skup podataka koji se koristi u mašinskom učenju i statistici. Sadrži mjerenja cvjetova jedne biljke koja se zove Iris (to je vrsta cvijeta). Cvijet se sastoji od više dijelova, kod irisa, vanjski listovi cvijeta se zovu: **Čašični listovi** i **Latice**. U skupu podataka nalaze se tri vrste cvijeta Iris:

1. Iris Setosa
2. Iris Versicolor
3. Iris Virginica

Vrsta cvijeta	**Kratki opis**
Iris Setosa	Manji, nježniji cvijet
Iris Versicolor	Srednje veličine
Iris Virginica	Najveći cvijet

Primjer1

Algoritam: Logistička regresija (binarna klasifikacija)

Logistička regresija je algoritam nadziranog učenja koji se koristi za binarne klasifikacijske probleme, odnosno kada želimo odlučiti da li neki uzorak pripada klasi *0* ili klasi *1*.

Za razliku od linearne regresije, koja daje kontinuiranu numeričku vrijednost, logistička regresija daje vjerovatnoću pripadanja nekoj klasi. Zbog toga koristi sigmoid funkciju:

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Ova funkcija mapira bilo koju vrijednost u opseg (0, 1). Ako je $h(x) > 0.5 \rightarrow$ uzorak se klasificira kao klasa 1, u suprotnom kao klasa 0. Model se trenira tako što se parametri θ optimizuju tako da minimiziraju logističku (log-loss) funkciju greške:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} [y^{(i)} \log (h(x^{(i)})) + (1 - y^{(i)}) \log (1 - h(x^{(i)}))]$$

Za optimizaciju se najčešće koristi:

- Gradijentni spust (ZA VELIKI SKUP PODATAKA)
  Gradijentni spust je iterativna metoda optimizacije koja se zasniva na pomjeranju parametara u smjeru negativnog gradijenta funkcije greške. Drugim riječima, algoritam traži smjer u kojem greška najbrže opada i pomjera parametre korak po korak: $\theta := \theta - \alpha \nabla J(\theta)$

Simbol	**Značenje**
$\nabla J(\theta)$	gradijent funkcije greške
α	stopa učenja (learning rate), koja određuje veličinu koraka

- Newton-Raphson metoda (MALI I SREDNJI SKUP PODATAKA)
  Newton-Raphson je brža optimizaciona metoda koja koristi drugi izvod (Hessovu matricu) kako bi izračunala direktniji korak prema minimumu. Ažuriranje parametara se vrši formulom:

$$\theta := \theta - H^{-1}\nabla J(\theta)$$

Simbol	Značenje
H	Hessova matrica (matrica drugih izvoda), daje zakrivljenost funkcije greške
$\nabla J(\theta)$	gradijent funkcije greške

Koristiti Iris skup podataka sa dvije klase cvijeta, Iris Setosa i Iris Versicolor, za ulazne podatke uzeti dužinu i širinu čašice. Primijeniti logističku regresiju i naučiti model da klasifikuje cvijet u jednu od te dvije klase. Ako je h(x) > 0.5, klasa 1, inače je klasa 0.

```
% 1) Binarna klasifikacija - Logistička regresija
load fisheriris
X = meas(1:100,1:2);          % biramo 100 primjera (2
klase)
y = species(1:100);

% Pretvaranje u binarnu oznaku (1 = setosa, 0 =
versicolor)
y = strcmp(y,'setosa');

% Dodavanje kolone jedinica (bias)
Xb = [ones(size(X,1),1) X];

% Inicijalizacija parametara
theta = zeros(3,1);

% Trening Newton-Raphson metodom
for i = 1:50
    z = Xb*theta;
    h = 1./(1+exp(-z));
    grad = Xb'*(h-y)/length(y);
    H = Xb' * diag(h.*(1-h)) * Xb / length(y);
    theta = theta - H\grad;
end

% Test i tačnost
y_pred = h > 0.5;
accuracy = mean(y_pred == y)*100;
disp(['Tačnost logističke regresije: ' num2str(accuracy)
'%'])
```

Command Window

```
>> primjer1
Tačnost logističke regresije: 100%
```

Primjer2

k-Nearest Neighbors (kNN) je algoritam nadziranog učenja koji se koristi za klasifikaciju i regresiju, ali se najčešće primjenjuje u klasifikaciji. Osnovna ideja algoritma je da uzorak pripada istoj klasi kao i njegovi najbliži susjedi u prostoru osobina (feature space).

Princip rada

Kada se pojavi novi uzorak koji treba klasifikovati:

1. Izračunaju se udaljenosti između tog uzorka i svih uzoraka u trening skupu.
2. Odabere se k najbližih uzoraka (susjeda).
3. Gleda se kojoj klasi pripada većina tih susjeda.
4. Uzorak se klasifikuje u tu najčešću klasu.

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2}$$

Najčešće se koristi Euklidska udaljenost:

gdje su:

- $x$novi uzorak,
- $x_i$uzorci iz trening skupa,
- $n$broj osobina.

Parametar k

Broj susjeda $k$jako utiče na tačnost algoritma:

Vrijednost k	**Posljedica**
Premali k (npr. k=1)	Model postaje osjetljiv na šum, može preučno učiti (overfitting).
Prevelik k	Model previše „izravnava“ podatke, može pogrešno klasifikovati (underfitting).

Zato se kobično bira eksperimentalno (npr. k = 3, 5, 7).

Koristiti sve tri klse Iris cvijeta i primijeniti kNN algoritam.

Algoritam kNN traži najbliže primjere u trening skupu i gleda kojoj klasi većina pripada.

```
% 2) Višeklasna klasifikacija - kNN
load fisheriris
X = meas(:,1:2);
y = categorical(species);    % << Ovdje dodano

% Podjela na train / test skup
idx = randperm(length(y));
train = idx(1:100);
test = idx(101:end);

Xtrain = X(train,:);
ytrain = y(train);
```

```
Xtest = X(test,:);
ytest = y(test);

k = 5;
y_pred = categorical(strings(length(ytest),1)); % pravimo prazan categorical vektor

for i = 1:length(Xtest)
    dist = sum((Xtrain - Xtest(i,:)).^2,2);
    [~, ind] = sort(dist);
    y_pred(i) = mode(ytrain(ind(1:k)));  % sada radi jer su kategorije
end

accuracy = mean(y_pred == ytest)*100;
disp(['kNN tačnost: ' num2str(accuracy) '%'])
```

```
>> primjer2
kNN tačnost: 76%
```

Primjer3

Linearna regresija je algoritam nadziranog učenja koji se koristi za rješavanje regresijskih problema, odnosno za predviđanje kontinuirane numeričke vrijednosti. Cilj algoritma je da pronađe linearnu vezu između ulazne varijable (nezavisne) i izlazne varijable (zavisne).

Najjednostavniji model linearne regresije može se zapisati kao:

$$y = \theta_0 + \theta_1 x$$

gdje je:

Oznaka	Značenje
y	izlaz (predviđena vrijednost)
x	ulaz (osobina / feature)
θ_0	slobodni član (intercept)
θ_1	koeficijent nagiba (weight)

Koeficijent θ_1 određuje kako se izlaz mijenja kada se ulaz promijeni.

Cilj algoritma

Pronaći parametre θ_0 i θ_1 tako da se linija najbolje prilagodi podacima.

Za to se koristi funkcija greške, najčešće Srednja kvadratna greška (MSE):

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

gdje je:

- $h_\theta(x)$ izlaz modela,
- m broj podataka.

Najniža vrijednost ove funkcije znači najbolje moguće uklapanje linije sa podacima.

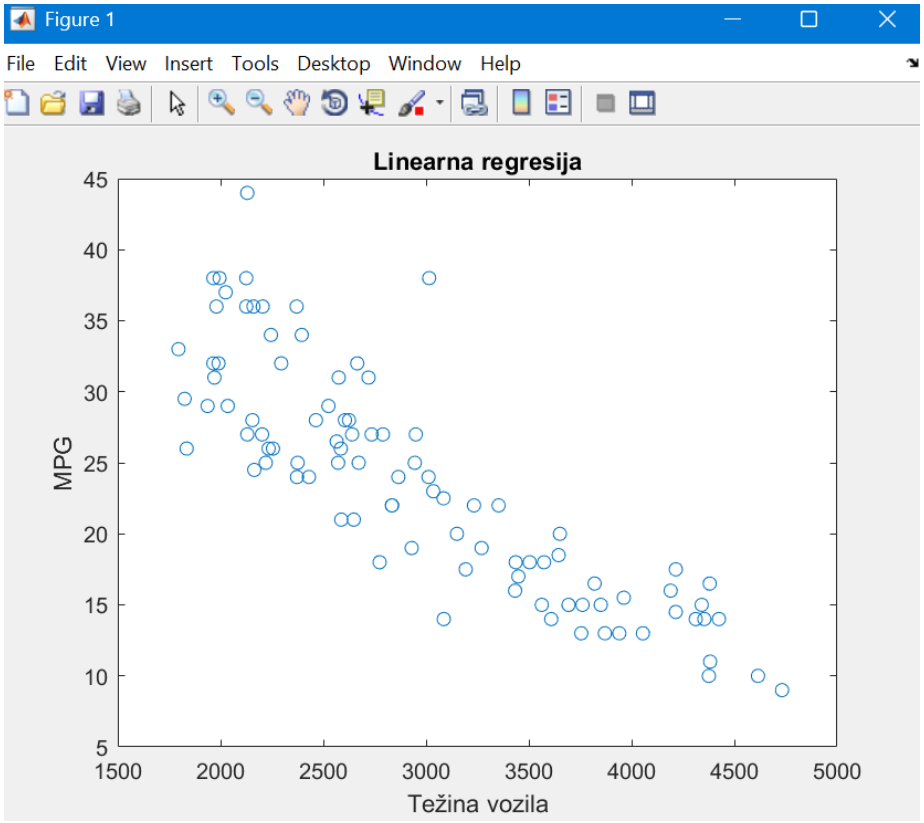
Koristiti carsmall skup podataka za modelovanje zavisnosti potrosnje goriva od težine vozila.

```
% 3) Linearna regresija
load carsmall
X = Weight;
y = MPG;

Xb = [ones(size(X)) X];
theta = (Xb' * Xb) \ (Xb' * y);

y_pred = Xb * theta;

figure
plot(X, y, 'o'); hold on
plot(X, y_pred);
xlabel('Težina vozila')
ylabel('MPG')
title('Linearna regresija')
```



Primjer4 – nenadzirano učenje

Poređenje metoda nenadziranog učenja

Metoda	*Osnovna ideja*	*Oblik klastera*	*Pripadnost*	*Gdje se koristi*
k-means	Centar = aritmetička sredina	Okrugli	Jedna klasa po tački	jednostavne grupe, brza analiza
GMM	Svaki klaster = Gaussova distribucija	Eliptični	Soft (vjerovatnoća)	statistička analiza, signali
Fuzzy c-means	Stepen pripadnosti svakom klasteru	Okrugli / eliptični	Fuzzy pripadnost	inteligentni sistemi, upravljanje energijom

---

K-means je algoritam nenadziranog učenja koji se koristi za klasterovanje podataka. Osnovna ideja je grupisati podatke u K grupa (klastera) tako da podaci unutar istog klastera budu što sličniji, a različiti klasteri budu što različitiji. Nema labela i nema unaprijed označenih klasa jer algoritam treba sam da otkriva strukturu u podacima.

Algoritam automatski traži centre grupa (tzv. centroidi) tako da minimizira varijansu unutar klastera. Svaka tačka se pridružuje onom centru kojem je najbliža.

Koraci algoritma K-means

1. Odaberi broj klastera K (npr. K = 3).
2. Nasumično inicijalizuj K centara (centroide).
3. Dodjela tačaka klasterima:
   Za svaki podatak izračunaj udaljenost do svakog centroida i dodijeli ga klasteru čiji je centroid najbliži.
4. Ažuriranje centroida:
   Za svaki klaster izračunaj novi centroid kao srednju vrijednost svih tačaka u tom klasteru.
5. Provjera konvergencije:
   Ako se centri više ne mijenjaju (ili promjena je vrlo mala), algoritam se zaustavlja.
   Inače, vrati se na korak 3.

Matematička formulacija

Centroid klastera C_j izračunava se kao:

$$\mu_j = \frac{1}{|\ C_j\ |} \sum_{x_i \in C_j} x_i$$

Cilj je minimizirati funkciju greške (sumu kvadratnih udaljenosti):

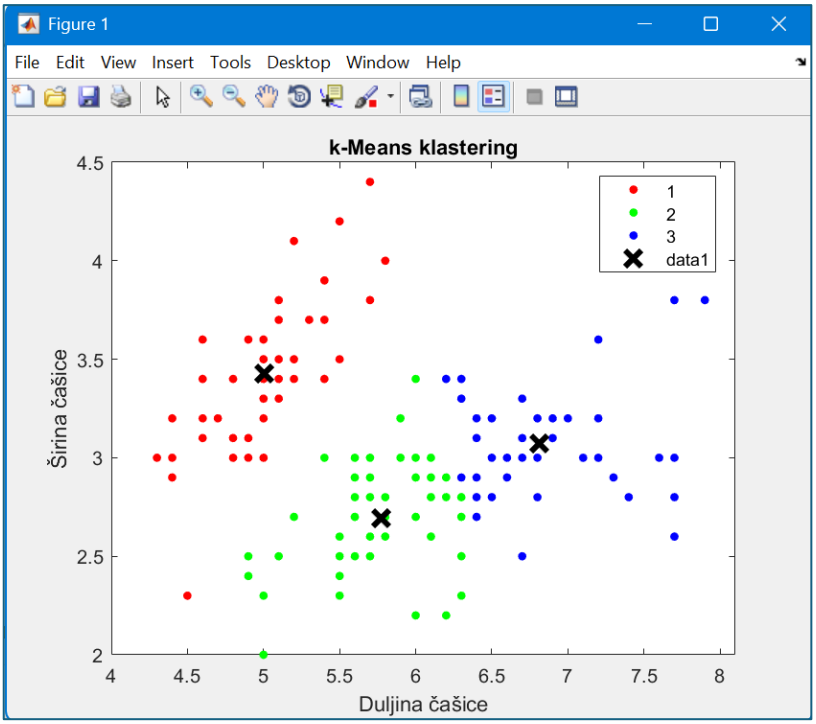
$$J = \sum_{j=1}^{K} \sum_{x_i \in C_j} \| x_i - \mu_j \|^2$$

Koristiti k-means za automatski grupisanje cvijeća u tri grupe.

```
% 4) K-Means klastering
load fisheriris
X = meas(:,1:2);

[idx, C] = kmeans(X,3);

figure
gscatter(X(:,1),X(:,2),idx)
hold on
plot(C(:,1), C(:,2), 'kx',
'MarkerSize',12,'LineWidth',3)
xlabel('Duljina čašice')
ylabel('Širina čašice')
title('k-Means klastering')
```



Primjer 5

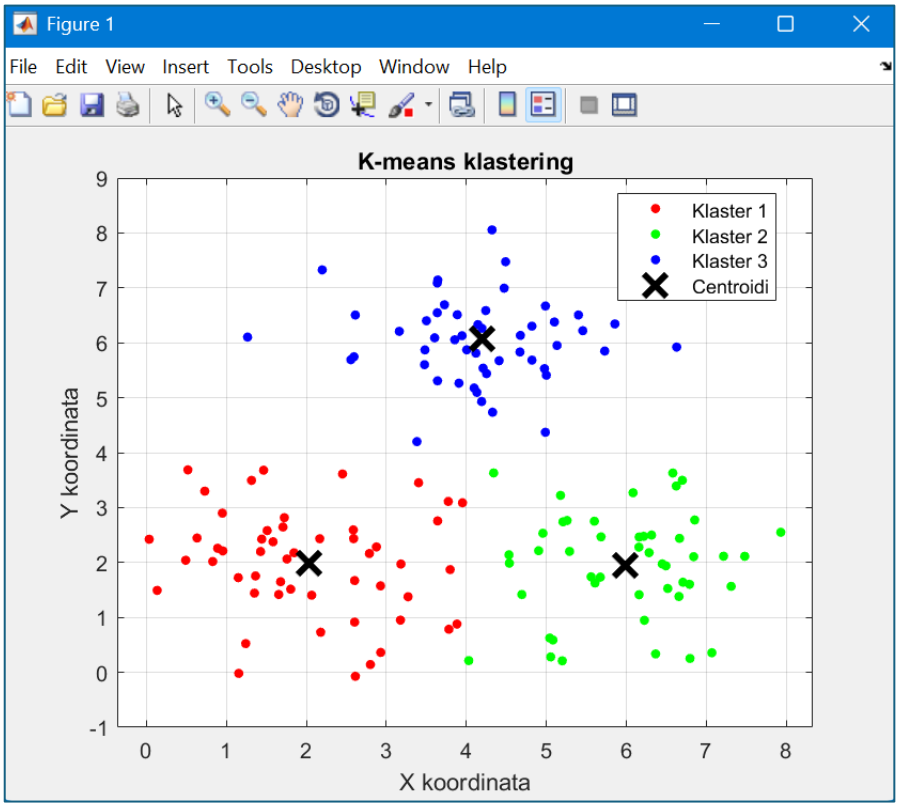
```
% K-means klastering primjer

% 1) Generišemo primjer podataka (3 različite grupe)
rng(1) % zbog ponovljivosti
group1 = randn(50,2) + [2 2];
group2 = randn(50,2) + [6 2];
group3 = randn(50,2) + [4 6];

% Spajamo sve dijelove u jednu matricu podataka
X = [group1; group2; group3];
```

```
% 2) Primjena k-means algoritma
k = 3; % broj klastera
[idx, centroids] = kmeans(X, k);

% 3) Vizualizacija rezultata
figure;
gscatter(X(:,1), X(:,2), idx, 'rgb');
hold on;
plot(centroids(:,1), centroids(:,2), 'kx', 'MarkerSize',
15, 'LineWidth', 3);
title('K-means klastering');
xlabel('X koordinata');
ylabel('Y koordinata');
legend('Klaster 1','Klaster 2','Klaster 3','Centroidi');
grid on;
```



ELBOW METODA – metoda za određivanje optimalnog broja klastera

K-means zahtijeva da unaprijed zadamo broj klastera **K**. Elbow metoda pomaže da odaberemo najbolji K. Mjeri se ukupna suma kvadrata udaljenosti tačaka od njihovih centara (tzv. *Within-Cluster Sum of Squares – WCSS*) za različite vrijednosti K.

- Ako je K premali → grupe su "loše", WCSS je velik.
- Kako se K povećava → WCSS opada.
- Ali nakon jednog trenutka smanjenje postane zanemarivo → tu je "lakat" (elbow).

Primjer 6

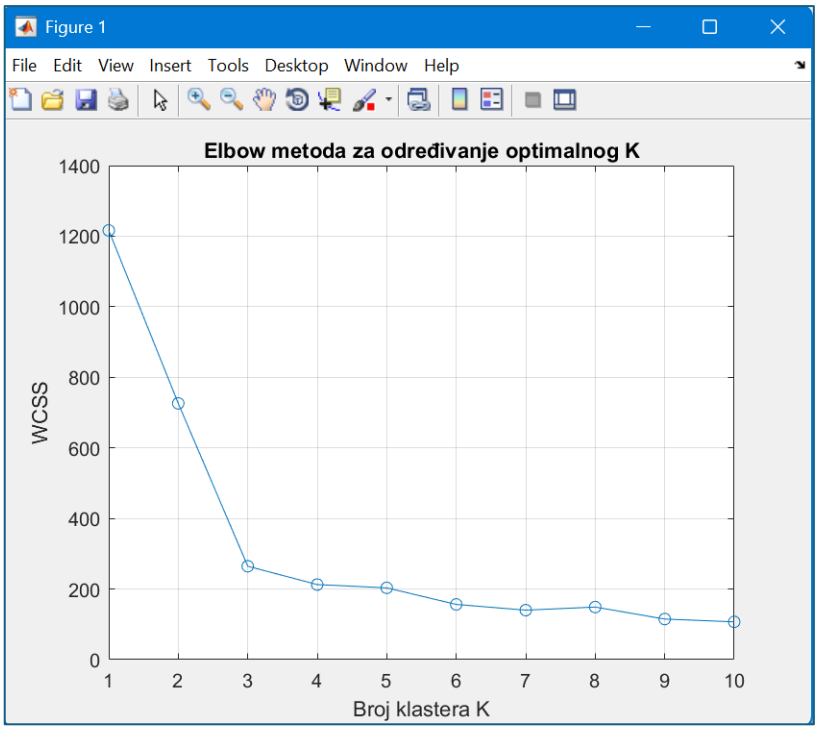
```
% Elbow metoda za određivanje optimalnog K

rng(1)
X = [randn(50,2)+[2 2];
     randn(50,2)+[6 2];
     randn(50,2)+[4 6]];

maxK = 10;
WCSS = zeros(maxK,1);

for k = 1:maxK
    [~,~,sumd] = kmeans(X, k);
    WCSS(k) = sum(sumd);
end

figure;
plot(1:maxK, WCSS, '-o');
xlabel('Broj klastera K');
ylabel('WCSS');
title('Elbow metoda za određivanje optimalnog K');
grid on;
```



K = 3

GAUSS klastering (GMM – Gaussian Mixture Model)

Ovdje klasteri nisu bazirani na sredini (kao k-Means), već se svaki klaster modelira kao Gaussova distribucija. Koristi se EM algoritam (Expectation-Maximization).

Primjer 7

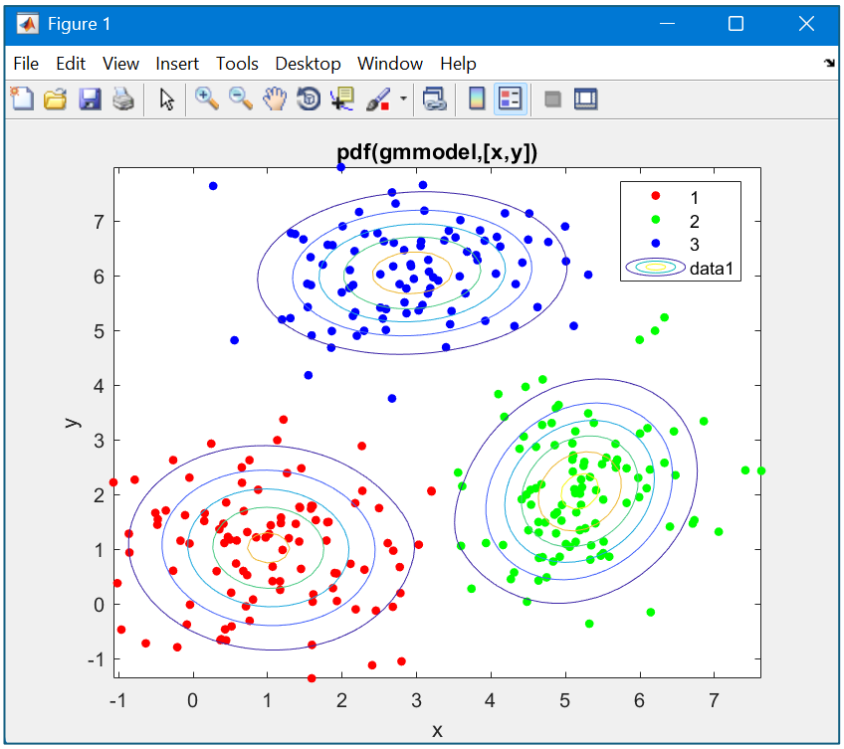
```
% Gaussian Mixture Model klastering

rng(1)
X = [randn(100,2)+[1 1];
     randn(100,2)+[5 2];
     randn(100,2)+[3 6]];

gmmodel = fitgmdist(X, 3);    % 3 klastera
idx = cluster(gmmodel, X);    % dodjela tačaka klasterima

figure;
gscatter(X(:,1), X(:,2), idx);
title('GMM klasterovanje (Gaussian Mixture Model)');
hold on;

% crtanje kontura
f = @(x,y) pdf(gmmodel,[x y]);
ezcontour(f, [min(X(:,1)) max(X(:,1))], [min(X(:,2))
max(X(:,2))]);
```



Fuzzy c-means (Fuzzy Logika klastering)

Za razliku od k-means gdje svaka tačka pripada samo jednom klasteru, u fuzzy c-means svaka tačka ima stepen pripadnosti svakom klasteru (npr. 0.2, 0.7, 0.1). Koristi se kod inteligentnih sistema, neuronskih mreža, upravljanja energijom...

Primjer 8

```
% Fuzzy C-Means klastering

rng(1)
X = [randn(50,2)+[2 2];
     randn(50,2)+[6 2];
     randn(50,2)+[4 6]];

% fuzzy c-means
[center, U] = fcm(X, 3);  % U = matrica pripadnosti

% Dodjela klastera prema najvećoj pripadnosti
[~, idx] = max(U);

figure;
gscatter(X(:,1), X(:,2), idx);
hold on;
plot(center(:,1), center(:,2), 'kx', 'MarkerSize', 15,
'LineWidth', 3);
title('Fuzzy c-means klasterovanje');
xlabel('X'); ylabel('Y');
grid on;
```

```
>> primjer8
Iteration count = 1, obj. fcn = 518.699270
Iteration count = 2, obj. fcn = 396.905090
Iteration count = 3, obj. fcn = 376.562145
Iteration count = 4, obj. fcn = 331.826143
Iteration count = 5, obj. fcn = 254.244822
Iteration count = 6, obj. fcn = 201.824826
Iteration count = 7, obj. fcn = 195.237961
Iteration count = 8, obj. fcn = 194.789717
Iteration count = 9, obj. fcn = 194.755255
Iteration count = 10, obj. fcn = 194.751797
Iteration count = 11, obj. fcn = 194.751360
Iteration count = 12, obj. fcn = 194.751295
Iteration count = 13, obj. fcn = 194.751284
Iteration count = 14, obj. fcn = 194.751282
```