Projektni zadatak iz prepoznavanja oblika 2022-2023

Mentor: Natalija Đorđević

Student: Ružić Nikola 0175/2019

Inovativni sistem klasifikacije za igru papir, kamen, makaze

Ulazni podatak za algoritam klasifikacije je RGB slika oblika 200x200x3. Zatim treba formirati obeležja za detekciju i klasifikaciju slika. Projektant algoritma se odlučio za sledeća dva obeležja:

1. binarizaciona maska
2. YCbCr maska

YCbCr je šema kodiranja boja koja se koristi za prikaz boja u digitalnim slikama i video sistemima. Ova šema razdvaja informacije o boji (krominancu) od informacija o osvetljenju (luminanci) slike. YCbCr model boja se dobija iz RGB (Crvena, Zelena, Plava) modela boja, ali predstavlja boje na drugačiji način.

U YCbCr modelu boja, "Y" komponenta predstavlja luminancu ili svetlinu boje, dok "Cb" i "Cr" komponente predstavljaju krominancu ili informacije o razlikama u boji.

Vrednosti oba obeležja su normalizovana. Prvo obeležje je da radi grubu detekciju ivica na slici, dok je drugo obeležje dobro za filtriranje pozadine i kasnije klasifikaciju pomoću parametarskog klasifikatora.

```
clear
close all
clc

dir1 = '/MATLAB Drive/Zadatak1/papir';
dir2 = '/MATLAB Drive/Zadatak1/kamen';
dir3 = '/MATLAB Drive/Zadatak1/makaze';

% Create an empty structure array or cell array
imageData = struct();  % Or use "imageData = {};" for a cell array

% Import images from directory 1
fileList1 = dir(fullfile(dir1, '*.png'));
for i = 1:numel(fileList1)
    imagePath = fullfile(dir1, fileList1(i).name);
    imageData.dir1(i).name = fileList1(i).name;
    imageData.dir1(i).image = imread(imagePath);
end

% Import images from directory 2
fileList2 = dir(fullfile(dir2, '*.png'));
for i = 1:numel(fileList2)
    imagePath = fullfile(dir2, fileList2(i).name);
    imageData.dir2(i).name = fileList2(i).name;
    imageData.dir2(i).image = imread(imagePath);
end

% Import images from directory 3
fileList3 = dir(fullfile(dir3, '*.png'));
for i = 1:numel(fileList3)
    imagePath = fullfile(dir3, fileList3(i).name);
```

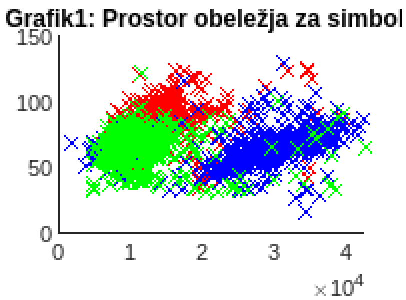
```
    imageData.dir3(i).name = fileList3(i).name;
    imageData.dir3(i).image = imread(imagePath);
end


papir_cnt = 0;
kamen_cnt = 0;
makaze_cnt = 0;

papir_ob = zeros(2,712);
kamen_ob = zeros(2,712);
makaze_ob = zeros(2,712);
```

Nakon frotmatiranja podataka, prelazi se na izraccunavanje obelezija za date slike.

```
for i = 3:714
    X1 = imageData.dir1(i-2).image;
    X2 = imageData.dir2(i-2).image;
    X3 = imageData.dir3(i-2).image;

    X1 = X1(:,20:end,:);
    X3 = X3(:,20:end,:);

    papir_ob(1,i-2) = ob1(X1);
    kamen_ob(1,i-2) = ob1(X2);
    makaze_ob(1,i-2) = ob1(X3);

    papir_ob(2,i-2) = ob2(X1);
    kamen_ob(2,i-2) = ob2(X2);
    makaze_ob(2,i-2) = ob2(X3);

end
```

Na slici se moze videti da su odabrana dva obeležija dovoljno dobro definisala i okarakterisala problem klasifikacije znakova. S obzirom da bolju separacija(a samim tim i klasikacija) postoji između znakova kamen i papir; izabrani su za projektovanje linearnog parametarskog klasifikatora.

```
figure
hold on
scatter(papir_ob(1,1:542),papir_ob(2,1:542),'rx')
scatter(kamen_ob(1,1:542),kamen_ob(2,1:542),'bx')
scatter(makaze_ob(1,1:542),makaze_ob(2,1:542),'gx')
title("Grafik1: Prostor obeležja za simbole")
hold off
```



Projekatant je dao sebi na slobodu da izabere linearni parametarski klasifikator, s obzirom na raspored podataka koji se vidi na grafiku1.

Konfuziona matrica

```
Y3 = papir_ob';
Y1 = kamen_ob';
Y2 = makaze_ob';

Mk = zeros(3, 3);

M_p = mean(Y1)'; S_p = cov(Y1);
M_k = mean(Y2)'; S_k = cov(Y2);
M_m = mean(Y3)'; S_m = cov(Y3);

klase = ['P','K','M'];


for k=1:3
    if k == 1
        Y = Y1;
    elseif k == 2
        Y = Y2;
    elseif k == 3
        Y = Y3;
    end
    for i=543:712
        x=Y(i,:)';
        fp=1/(2*pi*det(S_p)^1.5)*exp(-0.5*(x-M_p)'*inv(S_p)*(x-M_p));
        fk=1/(2*pi*det(S_k)^1.5)*exp(-0.5*(x-M_k)'*inv(S_k)*(x-M_k));
        fm=1/(2*pi*det(S_m)^1.5)*exp(-0.5*(x-M_m)'*inv(S_m)*(x-M_m));

        m=max([fp,fk,fm]);
        if m==fp
            Mk(k,1)=Mk(k,1)+1;
        elseif m==fk
            Mk(k,2)=Mk(k,2)+1;
```

```
        elseif m==fm
            Mk(k,3)=Mk(k,3)+1;
        end
    end
end
disp('Konfuziona matrica na test skupu:')
```

```
Konfuziona matrica na test skupu:
```

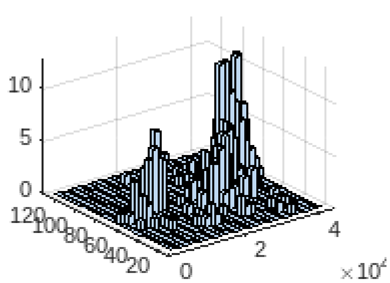
```
disp(Mk/(712-543+1)*100);
```

```
   70.5882   25.8824    3.5294
    5.8824   75.2941   18.8235
    4.7059   15.2941   80.0000
```

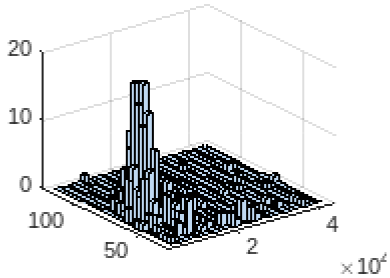
Ukoliko se pogledaju rezultati konfuzione matrice moze se zakljuciti da su rezultati prilično zadovoljavajući, doduše sa nekim poboljšanjem obeležja je moguće dobiti bolje rezultate.

Histogram

```
Y1 = Y1';
Y2 = Y2';
K1 = zeros(2,542);
K2 = zeros(2,542);

K1(1,:) = Y1(1,1:542);
K1(2,:) = Y1(2,1:542);

K2(1,:) = Y2(1,1:542);
K2(2,:) = Y2(2,1:542);

figure()
hist3(K1','Nbins', [30 30]);
```




```
figure();
hist3(K2','Nbins', [30 30]);
```




```
K1 = K1';
K2 = K2';

M1_est = mean(K1)';
M2_est = mean(K2)';
S1_est = cov(K1);
S2_est = cov(K2);
```

Histogrami govore gde je koncentracija merenja najveća, kad bi uzeli jednu unimodalnu(makaze) i jednu bimodalnu gausovu raspodelu(kamen), mogli bi lepo da izmodelujemo model verovatnoće ovih merenja. Doduše makaze imaju neko malo veće odstupanje od gausa, ali nedovoljno da načini model nevalidnim.

Parametarska klasifikacija

Kriterijumska funkcija linearnog klasifikatora je definisana:

$$h(X) = V_0 + \mathrm{V}^T X$$

Ukoliko se bolje pogleda kriterijumska funkcija, ona predstavlja projekciju vektora X na vektor V. Definišući kriterijumski funkciju koja zavisi od srednje vrednosti projekcije vektora X na vektor V, kao i varijanse takve transformacije. Optimizacijom te kriterijumske funkcije se dobijaju sledeći izrazi(imajući u vidu da su klase generisane gausovskom raspodelom):

$$s = \frac{-\eta_1/{\sigma_1}^2}{-\eta_1/{\sigma_1}^2 + \eta_2/{\sigma_2}^2}$$

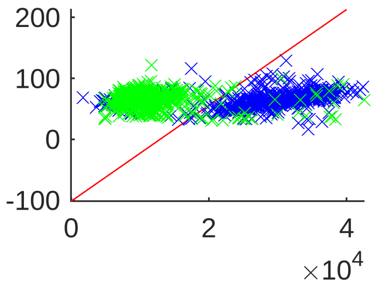
$$V = [s\Sigma_1 + (1-s)\Sigma_2]^{-1}(M_2 - M_1)$$

$$V_0 = -\frac{s{\sigma_1}^2\mathrm{V}^T M_2 + (1-s){\sigma_2}^2\mathrm{V}^T M_1}{s{\sigma_1}^2 + (1-s)s{\sigma_2}^2}$$

```
[V_optim, v0_optim] = lin(K1,K2)
```

```
V_optim = 2×1
   -0.0002
    0.0285
v0_optim = 2.8935
```

```
figure(1)
hold on
X1_lin=-2:0.01:40000;
X2_lin=-(v0_optim+V_optim(1)*X1_lin)/V_optim(2);

plot(X1_lin,X2_lin,'r');
scatter(kamen_ob(1,1:542),kamen_ob(2,1:542),'bx')
scatter(makaze_ob(1,1:542),makaze_ob(2,1:542),'gx')
hold off
```




```
```

Zadatak 2, Hipoteze

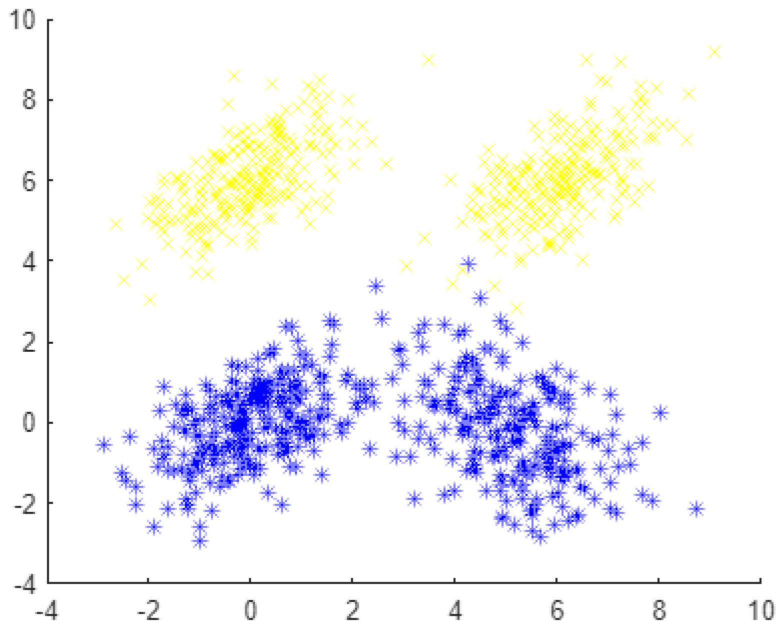
U daljem izveštaju će biti prikazani eksperimentalno dobijeni odbirci dveju dvodimenzionalnih bimodalnih gausovskih klasa, kao i klasifikacionih kriva Bayesovog testa minimalne greške, klasifikatora minimalne cene i Neuman Pearsonov klasifikatora.

Takođe će biti skicirana zavisnost broja odbirka od usvojene verovatnoće grešaka prvog/drugog tipa kod Waldovog sekvencionalnog testa.

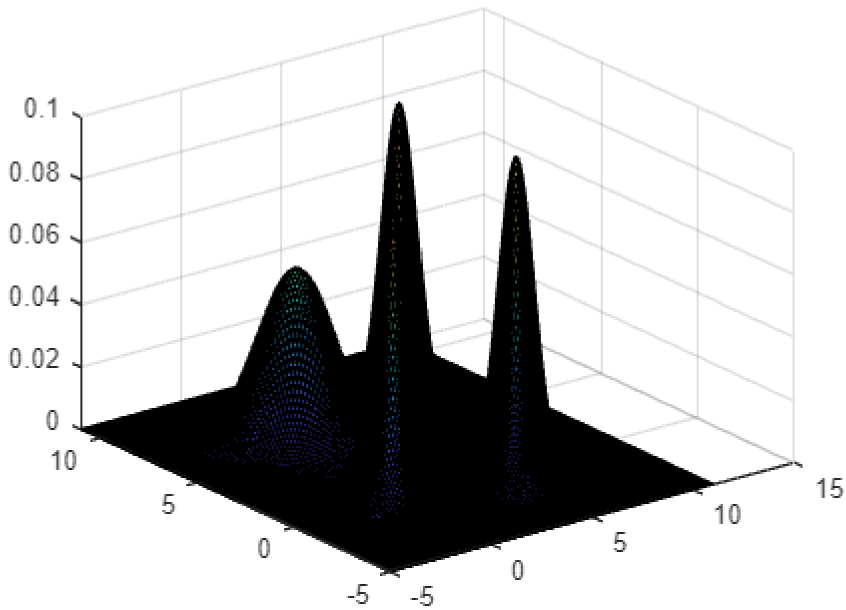
Klase

```
clc; clear; close all;
N=500;

M11 = [0 0]'; s11 = [1 0.5; 0.5 1];
M12 = [5 0]'; s12 = [1.5 -0.7; -0.7 1.5];
M21 = [6 6]'; s21 = [1 0.6; 0.6 1];
M22 = [0 6]'; s22 = [1 0.6; 0.6 1];

K1prva = mvnrnd(M11,s11,N);
K1druga = mvnrnd(M12,s12,N);
K2prva = mvnrnd(M21,s21,N);
K2druga = mvnrnd(M22,s22,N);

pom1 = rand(N,1);
K1 = (pom1<=0.5).*K1prva + (pom1>0.5).*K1druga;
pom2 = rand(N,1);
K2 = (pom2<=0.5).*K2prva + (pom2>0.5).*K2druga;

figure(1)
hold on
scatter(K1(:,1), K1(:,2),'blue', '*')
hold on
scatter(K2(:,1), K2(:,2),'yellow',"x")
hold off


%Teorijske fgv i histogram

x=-5:0.1:11;
y=-5:0.1:11;

f1 = zeros(length(x), length(y));
f2 = zeros(length(x), length(y));

const11 = 1/(2*pi*det(s11)^0.5);
const12 = 1/(2*pi*det(s12)^0.5);
const21 = 1/(2*pi*det(s21)^0.5);
const22 = 1/(2*pi*det(s22)^0.5);
```

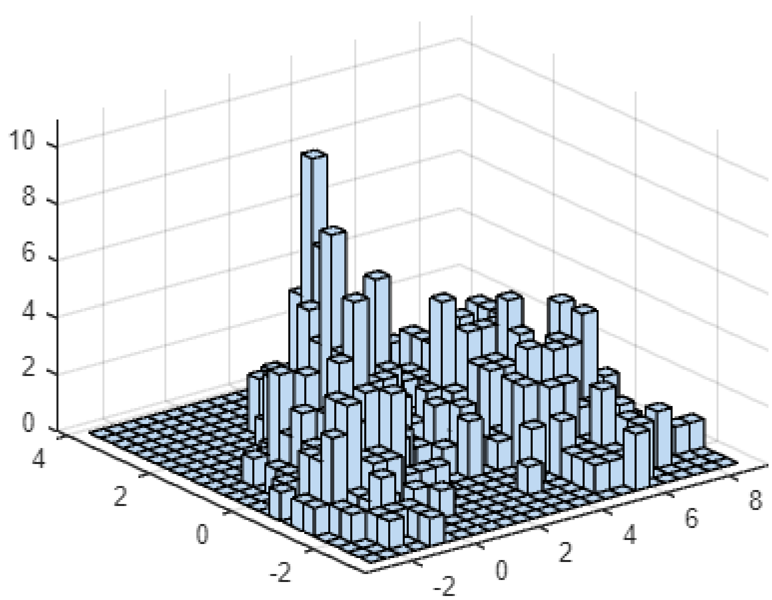
```
for i = 1:length(x)
    for j = 1:length(y)
        X = [x(i) y(j)]';
        f11 = const11* exp(-0.5*(X-M11)'*inv(s11)*(X-M11));
        f12 = const12* exp(-0.5*(X-M12)'*inv(s12)*(X-M12));
        f1(i,j) = 0.5*f11 + 0.5*f12;
        f21 = const21* exp(-0.5*(X-M21)'*inv(s21)*(X-M21));
        f22= const22* exp(-0.5*(X-M22)'*inv(s22)*(X-M22));
        f2(i,j) = 0.5*f21 + 0.5*f22;
    end
end

hold off
```



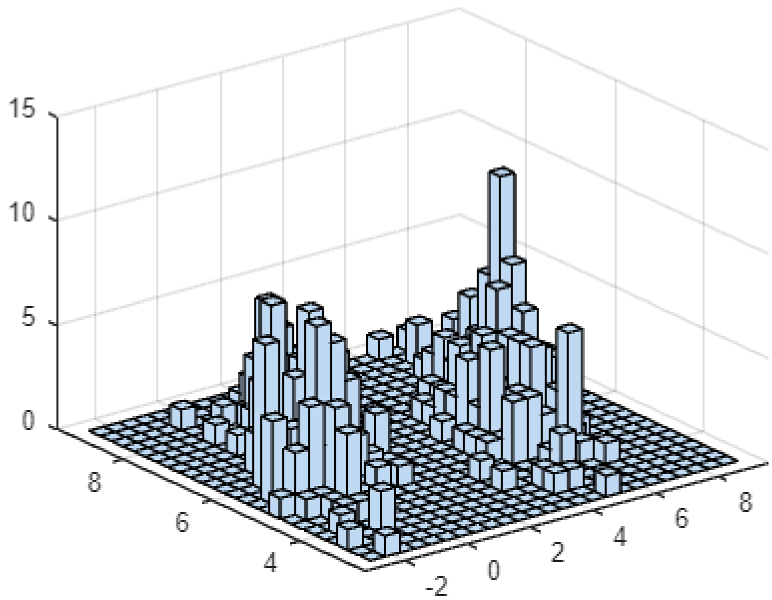

```
figure(2)
surf(x,y,f1);
hold on
surf(x,y,f2);
hold off
```




```
figure(3)
hist3([K1(:,1),K1(:,2)], 'nbins',[1,1]*25);
hold off
```




```
figure(4)
hist3([K2(:,1),K2(:,2)], 'nbins',[1,1]*25);
hold off
```



Bayesov test minimalne greške

Bayesov test minimalne greške se zasniva uslovnim verovatnoćama pripadnosti dobijenog merenja između dveju klasa. To su verovatnoće koje bi čitali pod uslovom X, verovatnoća da je u pitanju klasa *i.* Ukoliko je uslovna verovatnoće prve klase veća od druge, to znači da odbirak pripada prvoj klasi i obrnuto.

$\mathrm{q1}(X) > \mathrm{q2}(X)$, pripada klasi 1
$\mathrm{q1}(X) < \mathrm{q2}(X)$, pripada klasi 2

Sad je potrebno izvesti formulu(hipotezu) uz Bayesovu teoremu uslovne verovatnoće, koja na osnovu definisanih gustina verovatnoće donosi odluku.

$\mathrm{P1f1}(X) > \mathrm{P2f2}(X)$, pripada klasi 1
$\mathrm{P2f1}(X) < \mathrm{P2f2}(X)$, pripada klasi 2
$- - - - - - - - - - - - - - - - -$

$h(x) = -\ln\left(\frac{\mathrm{f1}}{\mathrm{f2}}\right) < \ln\left(\frac{\mathrm{P1}}{\mathrm{P2}}\right)$, pripada klasi 1

$h(x) = -\ln\left(\frac{\mathrm{f1}}{\mathrm{f2}}\right) > \ln\left(\frac{\mathrm{P1}}{\mathrm{P2}}\right)$, pripada klasi 2

```
h = - log(f1./f2);
```

```
figure()
hold on
scatter(K1(:,1), K1(:,2),'blue', '*')
hold on
scatter(K2(:,1), K2(:,2),'yellow',"x")
hold on
contour(x,y,h',[0,0],'g','LineWidth',1.5)
hold off



% Eksperimentalna greska
% (pristup konfuzione matrice)

% Spajamo odbirke u vektor, njihove stvarne klase znamo, a klasifikator,
% daje predikciju

K1 = K1';
K2 = K2';

Xs = [K1,K2];

X_true = [ones(1,N), ones(1,N)*2]; %stvarne klase
X_pred = zeros(size(X_true));


for i = 1:length(Xs)
   X = Xs(:,i);
   f11 = const11* exp(-0.5*(X-M11)'*inv(s11)*(X-M11));
   f12 = const12* exp(-0.5*(X-M12)'*inv(s12)*(X-M12));
   f1 = 0.5*f11 + 0.5*f12;
   f21 = const21* exp(-0.5*(X-M21)'*inv(s21)*(X-M21));
   f22= const22* exp(-0.5*(X-M22)'*inv(s22)*(X-M22));
   f2= 0.5*f21 + 0.5*f22;

   if (f1 > f2)
       X_pred(i) = 1;
   else
       X_pred(i) = 2;
   end

end



C = confusionmat(X_true,X_pred);

err1 = C(1,2)/sum(C(1,:)); %procenat pogresno klasifikovanih iz K1
err2 = C(2,1)/sum(C(2,:)); %procenat pogresno klasifikovanih iz K2
```

```
err1
```

```
err1 = 0.0020
```

```
err2
```

```
err2 = 0.0020
```

Bayesova greška se teorijski definiše na sledeći način

$\varepsilon = E\{r(X)\}$

Gde $r(X)$ predstavlja rizik. Imajući u vidu ideju bayesovog klasifikatora, intuitivno možemo razmišljati o riziku kao mesto na hiperravni oblika X za koje je uslovna verovatnoća dveju klasa minimalna. Dakle,

$r(X) = \min[\mathrm{q1}(X), \mathrm{q2}(X)]$

Zatim važi

$$\varepsilon = E\{r(X)\} \ = E\{\min[\mathrm{q1}(X), \mathrm{q2}(X)]\} = \int \min[\mathrm{q1}(X), \mathrm{q2}(X)] f(X)\mathrm{dx}$$

$$= \int \min\left[\mathrm{P1} * \frac{\mathrm{f1}(X)}{f(x)}, \mathrm{P2} * \frac{\mathrm{f2}(X)}{f(x)}\right] f(X)\mathrm{dx} = \int \min[\mathrm{P1} * \mathrm{f1}(X), \mathrm{P2} * \mathrm{f2}(X)]\mathrm{dx} =$$

$$= \mathrm{P1} * \int_{\mathrm{L2}} \mathrm{f1}(X)\mathrm{dx} + \mathrm{P2} * \int_{\mathrm{L1}} \mathrm{f2}(X)\mathrm{dx} = \mathrm{P1} * \varepsilon 1 + \mathrm{P2} * \varepsilon 2$$

```
e1 = 0;
e2 = 0;

for i = 1:length(x)
    for j = 1:length(y)
        X = [x(i) y(j)]';
        f11 = const11* exp(-0.5*(X-M11)'*inv(s11)*(X-M11));
        f12 = const12* exp(-0.5*(X-M12)'*inv(s12)*(X-M12));
        f1(i,j) = 0.5*f11 + 0.5*f12;
        f21 = const21* exp(-0.5*(X-M21)'*inv(s21)*(X-M21));
        f22= const22* exp(-0.5*(X-M22)'*inv(s22)*(X-M22));
        f2(i,j) = 0.5*f21 + 0.4*f22;

        h = -log(f1(i,j)/f2(i,j));

        if h<0
            e2 = e2 + 0.1*0.1*f2(i,j);
        else
            e1 = e1 + 0.1*0.1*f1(i,j);
        end
    end
end

e1
```

```
e1 = 0.0015
```

```
e2
```

```
e2 = 0.0016
```

Razlika teorijski i eksperimentalno dobijene greške

```
e1 - err1
```

```
ans = -5.0570e-04
```

```
e2 - err2
```

```
ans = -4.3453e-04
```

Klasifikator minimalne cene

Klasifikator minimalne cene, ili bajesovo pravilo odlučivanja minimalne cene se oslanja na otežinjene uslovne verovatnoće pojavljivanja oblika. I to na sledeći način:

$\mathrm{cij\ cena\ odluke}\ X \in \omega i, \mathrm{kada\ je\ zapravo}\ \omega j$

Formira se rizik, koji je sad otežinjena suma tačno i loše klasifikovanih klasa. Dakle,

$\mathrm{ri}(X) = \mathrm{ci1q1}(X) * \mathrm{ci2q2}(X)$

Pa pravilo odlučivanja postaje

$\mathrm{r1}(X) < \mathrm{r2}(X), \mathrm{klasa\ 1}$
$\mathrm{r2}(X) > \mathrm{r1}(X), \mathrm{klasa\ 2}$

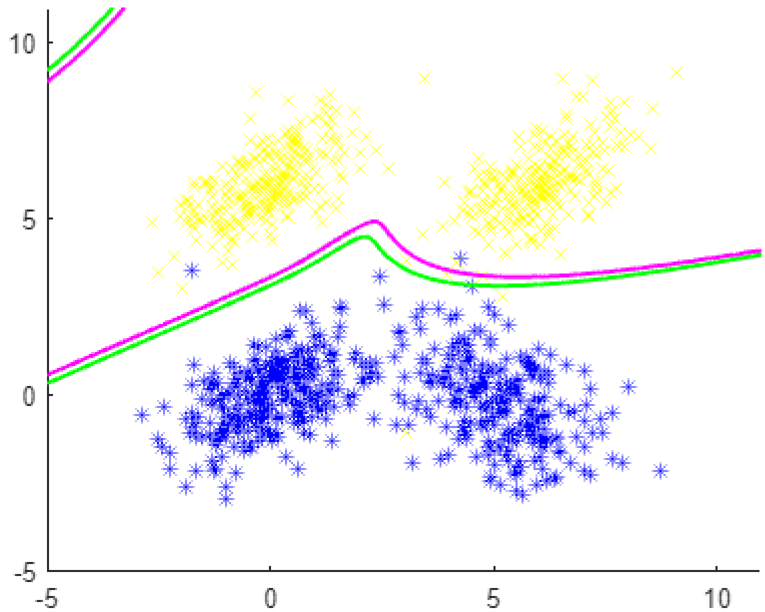
Tj gledamo klasu koja ima manji rizik. Na sličan način kao i za Bayesov klasifikator minimizujemo rizik. Dodavanjem određenih transformacija tako da izraz ukupnog rizika zavisi samo od jedne od oblasti klasa, i pomeranjem granica tako da minimizujemo sam taj rizik dobija se sledeći izraz.

$$\frac{\mathrm{f1}}{\mathrm{f2}} > \frac{(\mathrm{c12} - \mathrm{c22})}{(\mathrm{c21} - \mathrm{c11})}\frac{\mathrm{P2}}{\mathrm{P1}}, \mathrm{klasa\ 1}$$
$$\frac{\mathrm{f1}}{\mathrm{f2}} < \frac{(\mathrm{c12} - \mathrm{c22})}{(\mathrm{c21} - \mathrm{c11})}\frac{\mathrm{P2}}{\mathrm{P1}}, \mathrm{klasa\ 2}$$

U daljem rešavanju zadatka, potrebno je penalizovati više pogrešne klasifikacije odbiraka iz prve klase. Dakle c21 treba da bude vece.

```
c11 = 0; c22 = 0; % ne penalizujemo tacnu klasifikaciju

c21 = 5;

c12 = 1;
```

```
t = (c12 - c22)/(c21 - c11);
k = f1./f2;

figure(1)
hold all
scatter(K1(:,1), K1(:,2),'blue', '*')
scatter(K2(:,1), K2(:,2),'yellow',"x")

contour(x,y,k',[t,t],'m','LineWidth',1.5)
hold off
```



Neuman-Pearson-ov klasifikator

Nojmanov test hipoteze predstavlja još jedno od gore predstavljenih problema testiranja hipoteza. On ne teži da minimizuje ukupni rizik(grešku) klasifikacije obe klase(kao prethodne metode), već drži grešku klasifikacije jedne klase konstantnom, dok minimizuje grešku klasifikacije druge klase(koliko god je moguće). To se postiže korišćenjem Lagranžovog multiplikatora, gde je constraint $\varepsilon 2 = \varepsilon 0$.

$$r = \varepsilon 1 + \mu(\varepsilon 2 - \varepsilon 0)$$

$$r = \int_{\mathrm{L2}} \mathrm{f1}(X)\mathrm{dX} + \mu\left(\int_{\mathrm{L1}} \mathrm{f2}(X)\mathrm{dX} - \varepsilon 0\right)$$

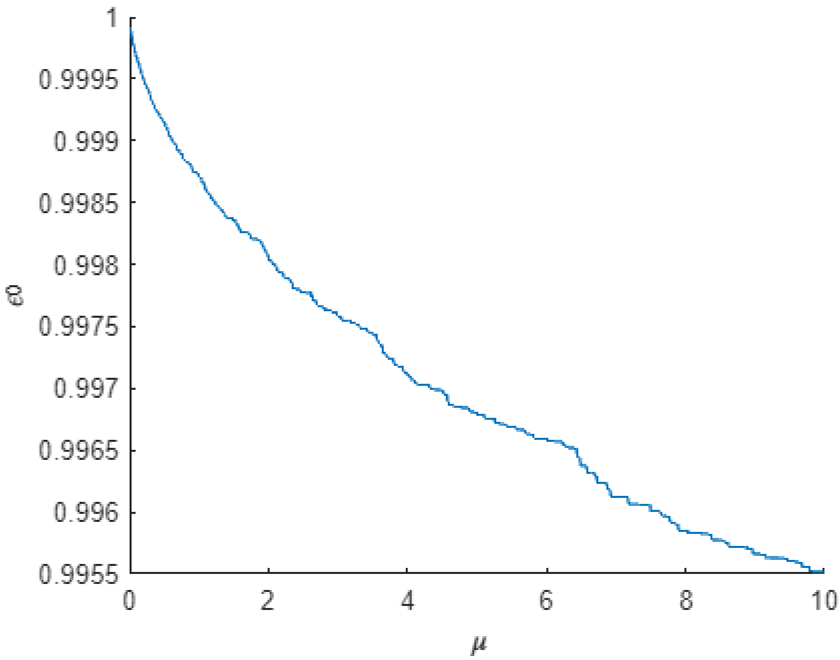
Na sličan način kao i ranije se rizik svodi na zavisnot od jedne oblasti, i pomeranjem te pblasti se vrši minimizacija rizika. Iz čega sledi:

$$\frac{f1}{f2} > \mu, \text{klasa } 1$$

$$\frac{f1}{f2} < \mu, \text{klasa } 2$$

S obzirom da se u zadatku traži realizacija klasifikatora koji ne sme da greši u klasifikaciji jedne klase(u našem slučaju prve), odabraćemo da držimo konstantom grešku klasifikacije prve klase i to da ona bude jednaka nuli. Lagranžov multiplikator se dobija numeričkim pristupom. Računa se konstantna greška klase variranjem lagranđovog multiplikatora. Za približnu vrednost greške se dobija vrednost multiplikatora.

```
h = -log(f1./f2);

br = 0;
for mi = 0.01:0.01:10
   br = br + 1;
   Eps1(br) = 0;
   for i = 1:length(x) - 1
       for j = 1:length(y) - 1
           if (h(i,j) < -log(mi))
               %Racuna rednju vrednost 4 temena kvadratica kojim
               %aproksimiramo integraciju fgv1 za greske klasifikacije
               Eps1(br) = Eps1(br) + 0.1*0.1*(f1(i,j) + f1(i+1,j) +
f1(i,j+1) + f1(i+1,j+1))/4;
           end


       end

   end

end

figure();
hold all;
plot(0.01:0.01:10,Eps1);
xlabel('\mu');
ylabel('\epsilon_0')
```



Waldov sekvencijalni test

Waldov sekvencijalni test je vrsta sekvencionalnog testa koji ne fiksira unapred broj odbiraka koji se koriste za testiranje hipoteze sm. Waldov sekvencijalni test funkcioniše na sledeći način:

$$s_m \le a, \text{klasa } 1$$
$$a < s_m < b, \text{uzmi sledeći odbirak}$$
$$s_m > b, \text{klasa } 2$$
$$s_m = \sum_{i=1}^{m} -\ln\left(\frac{\mathrm{f1(Xi)}}{\mathrm{f2(Xi)}}\right)$$

Gornja i donja granica waldovog testa se dobijaju na sledeći način:

$$a = -\log((1 - \mathrm{e1})/\mathrm{e2})$$
$$b = -\log(\mathrm{e1}/(1 - \mathrm{e2}))$$

```
test = 0;
e1 = 10^-4;
e2 = 10^-5;
m=0
```

```
m = 0
```

```
a = -log((1-e1)/e2)
```

```
a = -11.5128
```

```
b = -log(e1/(1-e2))
```

```
b = 9.2103
```

```
x=3:0.1:7;
y=4:0.1:5;

for i = 1:length(x)
    for j = 1:length(y)
        X = [x(i) y(j)]';

        test = test - log(fun1(X,M12,M11,s12,s11)/fun2(X,M21,M22,s21,s22));

        if test < a
            break
        end
        if test > b
            break
        end
        m = m+1;
    end
end

m
```

```
m = 5
```

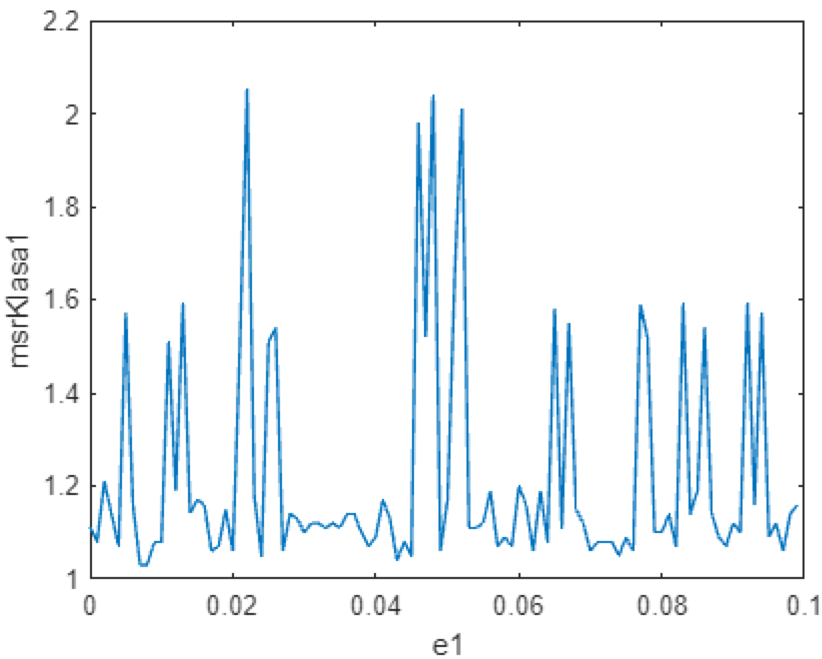
Sad je potrebno ispitati koji je prosečan broj merenja prilikom variranja grešaka prvog odnosno drugog tipa. Podrazumevajući da se jedna od tih grešaka drži konstantom.

```
X = K1;

plote1 = [];
plotmsr = [];

gres1 = 10^-5:0.001:10^-1;
e2 = 10^(-5);
% e1 = [10^(-7) 10^(-6) 10^(-5) 10^(-4) 10^(-3) 10^(-2) 10^(-1) 10^(0)];
e1 = gres1;
```

```
e1 = 1×100
    0.0000    0.0010    0.0020    0.0030    0.0040    0.0050    0.0060    0.0070 ···
```

```
for i = 1:length(e1)
    m=0;
    br=0;
    for j=1:100
        m = m + wald(X, e1(i), e2);
```

```
        br = br + 1;
    end
    msr = m/br;
    plote1 = [plote1 e1(i)];
    plotmsr = [plotmsr msr];
end

figure()
plot(plote1, plotmsr)
xlabel('e1')
ylabel('msrKlasa1')
```

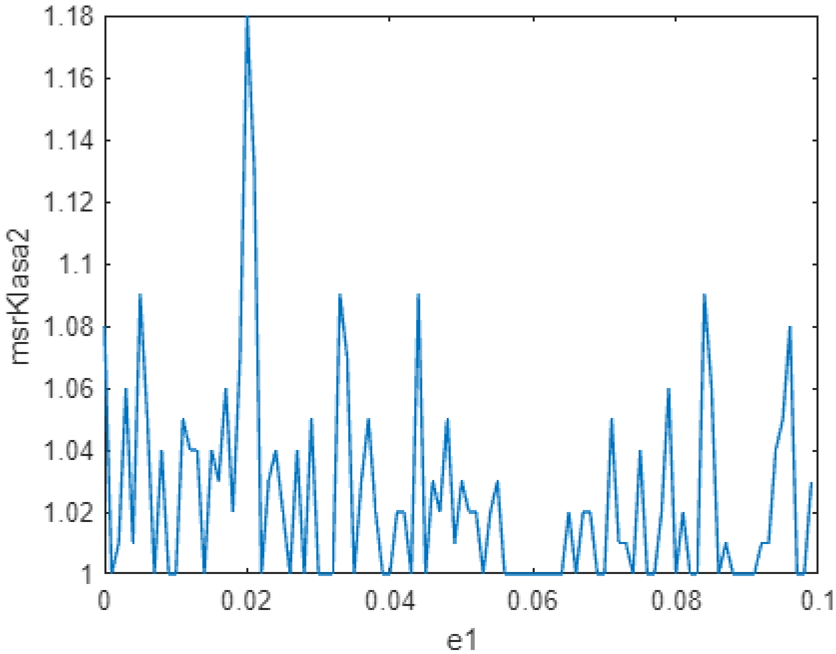



```
X = K2;

plote1 = [];
plotmsr = [];

gres1 = 10^-5:0.001:10^-1;
e2 = 10^(-5);
% e1 = [10^(-7) 10^(-6) 10^(-5) 10^(-4) 10^(-3) 10^(-2) 10^(-1) 10^(0)];
e1 = gres1;
```

```
e1 = 1×100
    0.0000    0.0010    0.0020    0.0030    0.0040    0.0050    0.0060    0.0070 ···
```

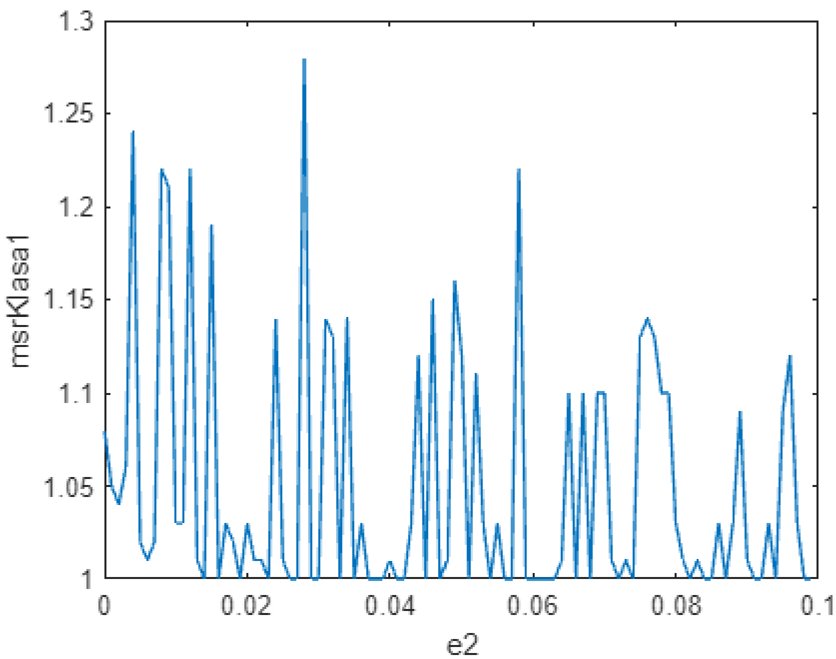
```
for i = 1:length(e1)
    m=0;
    br=0;
    for j=1:100
        m = m + wald(X, e1(i), e2);
```

```
        br = br + 1;
    end
    msr = m/br;
    plote1 = [plote1 e1(i)];
    plotmsr = [plotmsr msr];
end

figure()
plot(plote1, plotmsr)
xlabel('e1')
ylabel('msrKlasa2')
```



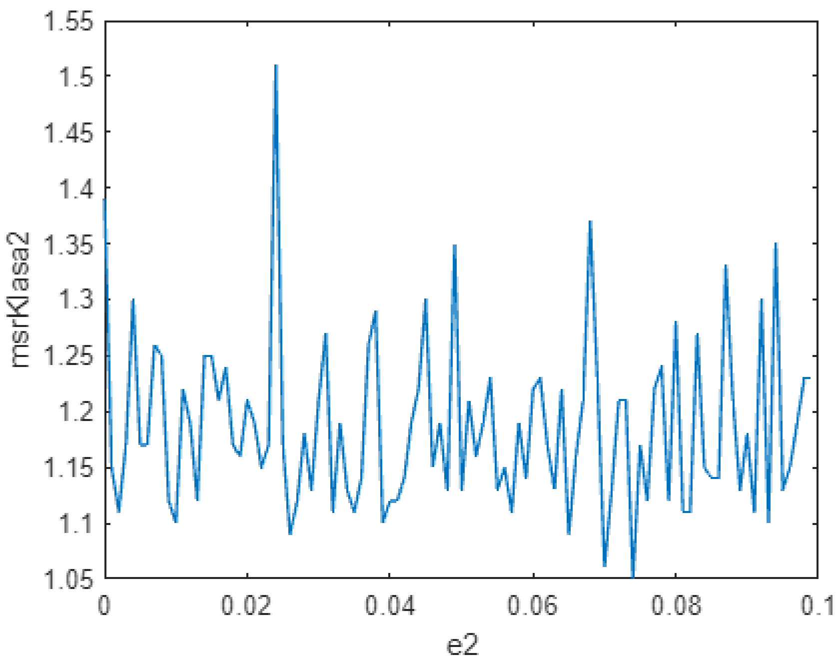

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X = K1;

plote2 = [];
plotmsr2 = [];


gres2 = 10^-5:0.001:10^-1;


e1 = 10^(-5);
% e2 = [10^(-7) 10^(-6) 10^(-5) 10^(-4) 10^(-3) 10^(-2) 10^(-1) 10^(0)];
e2 = gres2;
for i = 1:length(e2)
    m=0;
    br=0;
    for j=1:100
        m = m + wald(X, e1, e2(i));
```

```
        br = br + 1;
    end
    msr = m/br;
    plote2 = [plote2 e2(i)];
    plotmsr2 = [plotmsr2 msr];
end

figure()
plot(plote2, plotmsr2)
xlabel('e2')
ylabel('msrKlasa1')
```




```
X = K2;

plote2 = [];
plotmsr2 = [];


gres2 = 10^-5:0.001:10^-1;


e1 = 10^(-5);
% e2 = [10^(-7) 10^(-6) 10^(-5) 10^(-4) 10^(-3) 10^(-2) 10^(-1) 10^(0)];
e2 = gres2;
for i = 1:length(e2)
    m=0;
    br=0;
    for j=1:100
        m = m + wald(X, e1, e2(i));
```

```
        br = br + 1;
    end
    msr = m/br;
    plote2 = [plote2 e2(i)];
    plotmsr2 = [plotmsr2 msr];
end

figure()
plot(plote2, plotmsr2)
xlabel('e2')
ylabel('msrKlasa2')
```



Na osnovu grafika se može zaključiti da je osetljivost waldovog testa prilično velika za sva četiri slučaja. Projektant je zaključio da je to vrlo verovatno do toga što je uzeto 500 odbiraka u eksperimentu pa je i rezultat malo zašumljen. Pa je samim tim i prosečan broj odbiraka nizak(previše sličnih odbiraka). Kao i to da su klase prilično separabilne, pa test može veoma brzo da zaključi koja je klasa u pitanju, s obzirom da Waldow test minimizira srednju vrednost potrebnog broja merenja. E sad idalje ze može doći zaključiti da je uticaj verovatnoće grešaka za njhove respektivne klase, zanemarljiv. Kao i da je uticaj greške na primer drugog tipa, utiče na broj merenja koji je potreban za klasifikaciju prve klase, doduše ne dominatno.

Treći projektni zadatak

Potrebno je projektovati linearni klasifikator za tri separabilne dvodimenzionalne klase. Biće korišćene unimodalne gausovske raspodele. Kriterijumska funkcija linearnog klasifikatora je definisana:

$$h(X) = V_0 + \mathrm{V}^T X$$

Ukoliko se bolje pogleda kriterijumska funkcija, ona predstavlja projekciju vektora X na vektor V. Definišući kriterijumski funkciju koja zavisi od srednje vrednosti projekcije vektora X na vektor V, kao i varijanse takve transformacije. Optimizacijom te kriterijumske funkcije se dobijaju sledeći izrazi(imajući u vidu da su klase generisane gausovskom raspodelom):

$$s = \frac{-\eta_1/{\sigma_1}^2}{-\eta_1/{\sigma_1}^2 + \eta_2/{\sigma_2}^2}$$

$$V = [s\Sigma_1 + (1-s)\Sigma_2]^{-1}(M_2 - M_1)$$

$$V_0 = -\frac{s{\sigma_1}^2 \mathrm{V}^T M_2 + (1-s){\sigma_2}^2 \mathrm{V}^T M_1}{s{\sigma_1}^2 + (1-s)s{\sigma_2}^2}$$

Za implementaciju linearnih klasifikatora, koristiće se iterativni pristup resupstitucije:

1. Estimacija srednjih vrednosti i kovarijacionih matrica
2. Odredjivanje vektora V
3. Računanje pomoćne promenljive $y = \mathrm{V}^T X$
4. Brojenje pogrešno klasifikovanih odbiraka koji ne zadovoljavaju $y, \text{klasa } 1 < -V_0, y, \text{klasa } 2 > -V_0$, variranjem V_0, odabira se V_0 koji ima najmanje grešaka.
5. Varira se s, i uzima se ono za koje je broj grešaka najmanji

S obzirom da postoje tri klase, pristupiće se projektovanju deo po deo linearni klasifikatora. U konkretno slučaju sa tri gausovske separabilne klase, projektujemo tri linearna klasifikatora. Uzimamo ih kao "granicu" između bregova raspodela. Postoji mogućnost nastanka geometrijskog mesta tačaka koji neće moći biti klasifikovan. Uzimajući u obzir tri linearno isprojektovana klasifikatora $\mathrm{h12}(X), \mathrm{h13}(X), \mathrm{h23}(X)$.

$\text{if } h_{i1} > 0 \text{ and } h_{i2} > 0 \text{ and } h_{i3} > 0, \text{sledi klasa } i$

```
close all
clear all
clc



N=500;
rng(100);

M1=[1 ;1];  S1=[1 0; 0 1];
M2=[5;8]; S2=[1.2 0.4;0.4 1.2];
M3=[8;0]; S3=[1.2 0.4;0.4 1.2];
```

```
X1=mvnrnd(M1,S1,N)';
X2=mvnrnd(M2,S2,N)';
X3=mvnrnd(M3,S3,N)';

figure(1)
hold all
scatter(X1(1,:),X1(2,:),'ro');
scatter(X2(1,:),X2(2,:),'b*');
scatter(X3(1,:),X3(2,:),'yx');


[V_optim12, v0_optim12] = lin(X1,X2)
```

```
V_optim12 = 2×1
    1.5467
    5.5956
v0_optim12 = -26.2652
```

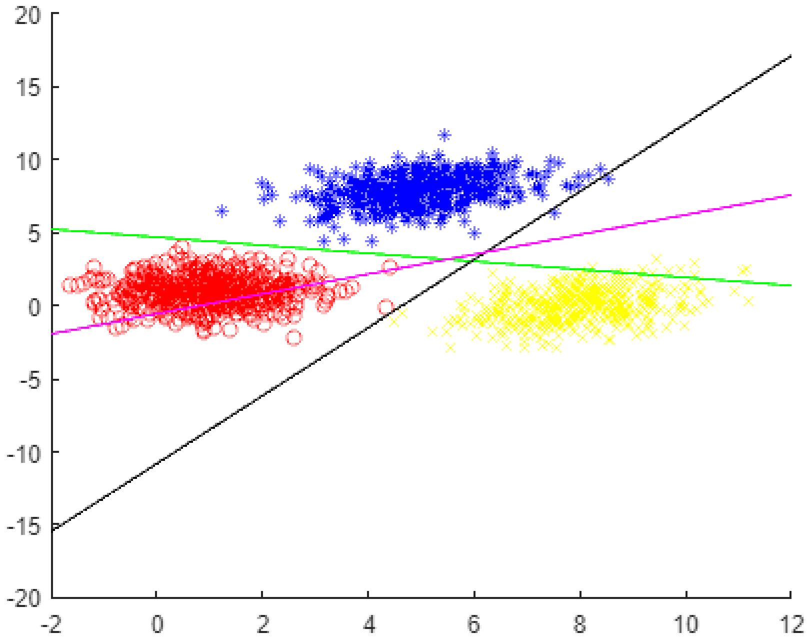
```
[V_optim13, v0_optim13] = lin(X1,X3)
```

```
V_optim13 = 2×1
    7.2359
   -3.1076
v0_optim13 = -33.6067
```

```
[V_optim23, v0_optim23] = lin(X2,X3)
```

```
V_optim23 = 2×1
    5.5337
   -8.1636
v0_optim23 = -4.5189
```

```
figure(1)
X1_lin=-2:0.01:12;
X2_lin12=-(v0_optim12+V_optim12(1)*X1_lin)/V_optim12(2);
X2_lin13=-(v0_optim13+V_optim13(1)*X1_lin)/V_optim13(2);
X2_lin23=-(v0_optim23+V_optim23(1)*X1_lin)/V_optim23(2);
plot(X1_lin,X2_lin12,'g');
plot(X1_lin,X2_lin13,'k');
plot(X1_lin,X2_lin23,'m');
```




```
%Generisanje test seta
%Povecao sam varijanse da bi bilo vise preklapanja izmedju klasa
M1t=[1 ;1];  S1t=[3 0; 0 3];
M2t=[5;8]; S2t=[2 0.36;0.36 2];
M3t=[8;0]; S3t=[3 0.2;0.2 3];

X1t=mvnrnd(M1t,S1t,N)';
X2t=mvnrnd(M2t,S2t,N)';
X3t=mvnrnd(M3t,S3t,N)';



X_test = [X1t, X2t, X3t];
X_true = [ones(1,500) ones(1,500)*2 ones(1,500)*3];
X_pred = zeros(size(X_true));
```

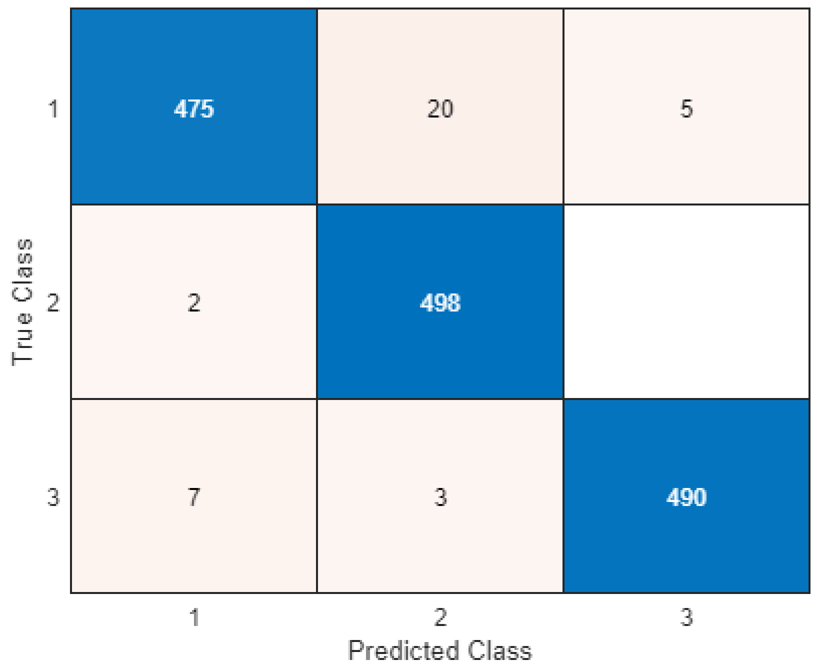
Napomena, treba iskomentarisati implementaciju gore pomenute deo po deo linearnog klasifikatora. Vrsi se tako sto se krene od karakteristicne tacke preseka linearnih klasifikatora. Deli se ravan podataka po jednoj od kordinata, pa se gleda gore izvedena vrednost druge koordinate i poredi da li je veca ili manja od dobijene koordinate, pa se na taj nacin ide dalje po ravni analogno za svaki uzeti linearni segment.

```
%Bilo bi idealno da se ovo nadje programski
 xpresek = 5.94
```

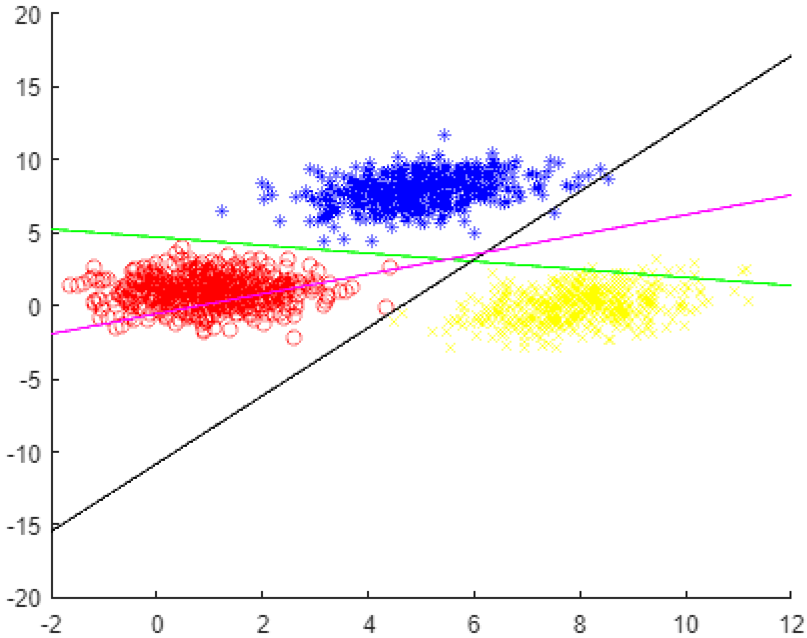
```
xpresek = 5.9400
```

```
for i = 1:length(X_true)
```

```
    X_cur = X_test(:,i);

    x = X_cur(1);
    y = X_cur(2);

    ylin12=-(v0_optim12+V_optim12(1)*x)/V_optim12(2);
    ylin13=-(v0_optim13+V_optim13(1)*x)/V_optim13(2);
    ylin23=-(v0_optim23+V_optim23(1)*x)/V_optim23(2);

    if x > xpresek
        %klasa2 ili klasa3
        if y > ylin23
            %klasa2
            X_pred(i) = 2;
        else
            %klasa3
            X_pred(i) = 3;
        end

    else
        %klasa 1/klasa2/klasa3
        if y > ylin13
            %klasa 1/2
            if y> ylin12
                %klasa 2
                X_pred(i) = 2;
            else
                %klasa 1
                X_pred(i) = 1;
            end

        else
            %klasa 3
           X_pred(i) = 3;
        end

    end
end

figure()
C = confusionmat(X_true,X_pred);
confusionchart(C)
```




```
figure()
hold all
scatter(X1(1,:),X1(2,:),'ro');
scatter(X2(1,:),X2(2,:),'b*');
scatter(X3(1,:),X3(2,:),'yx');
X1_lin=-2:0.01:12;
X2_lin12=-(v0_optim12+V_optim12(1)*X1_lin)/V_optim12(2);
X2_lin13=-(v0_optim13+V_optim13(1)*X1_lin)/V_optim13(2);
X2_lin23=-(v0_optim23+V_optim23(1)*X1_lin)/V_optim23(2);
plot(X1_lin,X2_lin12,'g');
plot(X1_lin,X2_lin13,'k');
plot(X1_lin,X2_lin23,'m');
```



Projektovanje linearnog klasifikatora na bazi željenog izlaza

Možemo objediniti odluke linearnog klasifikatora, ukoliko prepakujemo vektor X u vektor Z. Gde je svaki odbirak iz X koji pripada prvoj klasi biva pomnožen sa -1, odbirak koji pripada drugoj klasi sa +1, i sve se to stavlja u vektor Z. Zatim diskrimanciona funkcija postaje:

$$h(Z) = W^T Z > 0$$

Gde je za linearni klasifikator $W = \left[V_0\, V^T\right]$. Korišćenjem kriterijuma sume kvadrata razlike h(Z) i markera koji su postavljeni kao vrednost kojoj diskriminaciona funkcija treba da bude jednaka. Minimizacijom datog kriterijuma se dobija:

$$W = (U\mathrm{U}^T)^{-1} U\Gamma$$

Gde je:

$$U = [Z_1\, Z_2 \ldots Z_N],\ \Gamma = [\gamma(Z_1)\, \gamma(Z_2) \ldots \gamma(Z_N)]^T$$

U, vektor odbiraka Z, Γ vektor željenih izlaza. Dalje su predstavljeni dobijeni rezulatati kada se ne penalizuje nijedna klasa, kada se favorizuje crvena klasa i kada se penalizuje crvena klasa. Bitno je napomenuti da ovako izvedena analitčka relacija važi jedino za gore opisani kriterijum metode najmanjih kvadrata.

```
X_test = [X1, X2, X3];
X_true = [ones(1,500) ones(1,500)*2 ones(1,500)*3];
X_pred = zeros(size(X_true));



[V_optim12, v0_optim12] = izlaz(X1,X2,N,1,1)
```

```
V_optim12 = 1×2
    0.0908    0.2163
v0_optim12 = -1.2290
```

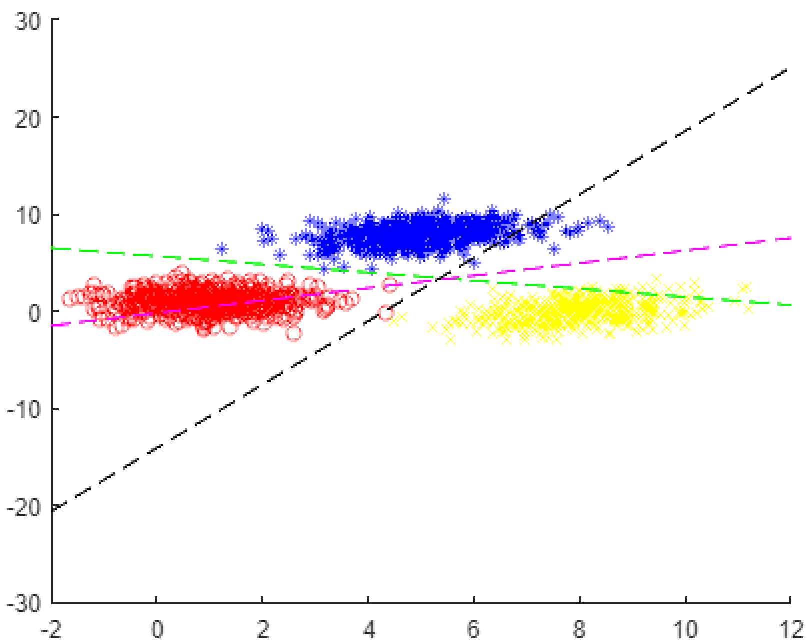
```
[V_optim13, v0_optim13] = izlaz(X1,X3,N,1,1)
```

```
V_optim13 = 1×2
    0.2553   -0.0782
v0_optim13 = -1.1009
```

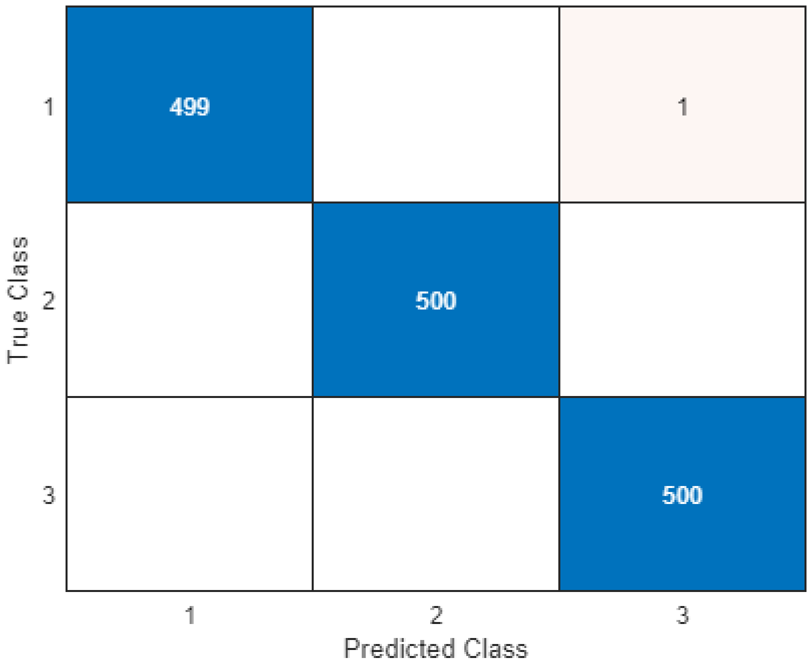
```
[V_optim23, v0_optim23] = izlaz(X2,X3,N,1,1)
```

```
V_optim23 = 1×2
    0.1254   -0.1949
v0_optim23 = -0.0349
```

```
figure()
hold all
scatter(X1(1,:),X1(2,:),'ro');
scatter(X2(1,:),X2(2,:),'b*');
scatter(X3(1,:),X3(2,:),'yx');

X1_lin=-2:0.01:12;
X2_lin12=-(v0_optim12+V_optim12(1)*X1_lin)/V_optim12(2);
X2_lin13=-(v0_optim13+V_optim13(1)*X1_lin)/V_optim13(2);
X2_lin23=-(v0_optim23+V_optim23(1)*X1_lin)/V_optim23(2);
plot(X1_lin,X2_lin12,'g--');
plot(X1_lin,X2_lin13,'k--');
plot(X1_lin,X2_lin23,'m--');
```



```
X_pred =
predict(X_test,X_true,v0_optim12,v0_optim13,v0_optim23,V_optim12,V_optim13,V_
optim23);
figure()
C = confusionmat(X_true,X_pred);
confusionchart(C)
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[V_optim12, v0_optim12] = izlaz(X1,X2,N,2,1)
```

```
V_optim12 = 1×2
    0.1361    0.3244
v0_optim12 = -2.3435
```

```
[V_optim13, v0_optim13] = izlaz(X1,X3,N,2,1)
```

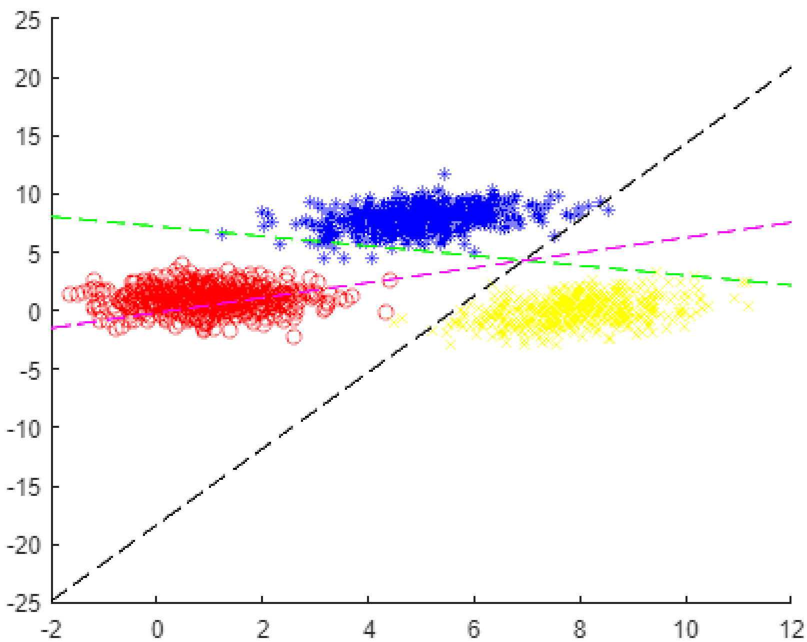
```
V_optim13 = 1×2
    0.3829   -0.1173
v0_optim13 = -2.1514
```

```
[V_optim23, v0_optim23] = izlaz(X2,X3,N,1,1)
```

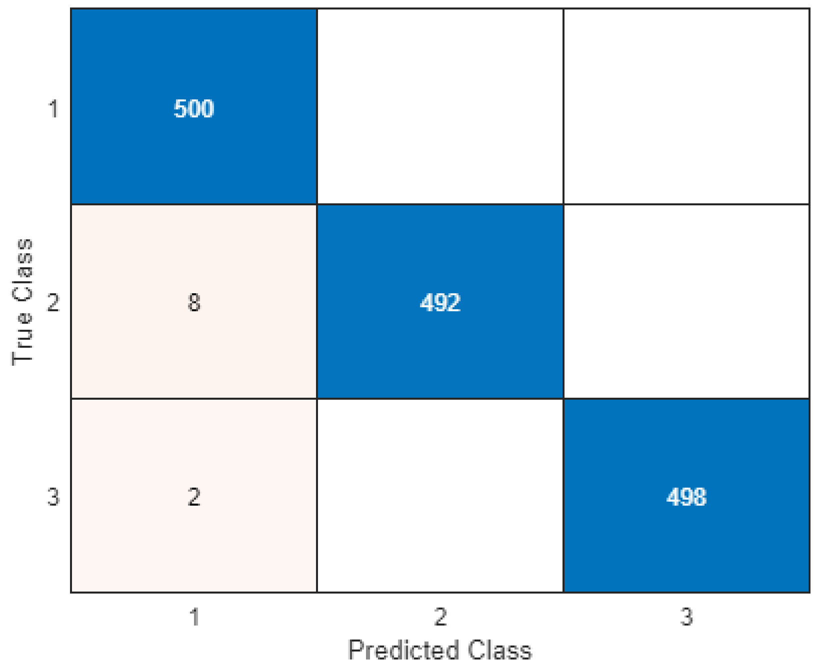
```
V_optim23 = 1×2
    0.1254   -0.1949
v0_optim23 = -0.0349
```

```



figure()
hold all
scatter(X1(1,:),X1(2,:),'ro');
```

```
scatter(X2(1,:),X2(2,:),'b*');
scatter(X3(1,:),X3(2,:),'yx');

X1_lin=-2:0.01:12;
X2_lin12=-(v0_optim12+V_optim12(1)*X1_lin)/V_optim12(2);
X2_lin13=-(v0_optim13+V_optim13(1)*X1_lin)/V_optim13(2);
X2_lin23=-(v0_optim23+V_optim23(1)*X1_lin)/V_optim23(2);
plot(X1_lin,X2_lin12,'g--');
plot(X1_lin,X2_lin13,'k--');
plot(X1_lin,X2_lin23,'m--');
```




```
X_pred =
predict(X_test,X_true,v0_optim12,v0_optim13,v0_optim23,V_optim12,V_optim13,V_
optim23);
figure()
C = confusionmat(X_true,X_pred);
confusionchart(C)
```




```
[V_optim12, v0_optim12] = izlaz(X1,X2,N,1,2)
```

```
V_optim12 = 1×2
    0.1361    0.3244
v0_optim12 = -1.3435
```

```
[V_optim13, v0_optim13] = izlaz(X1,X3,N,1,2)
```

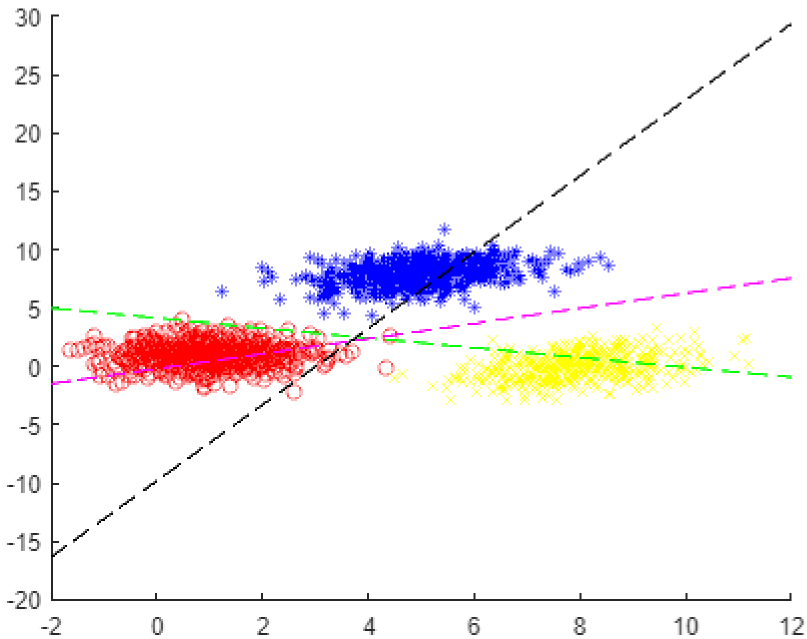
```
V_optim13 = 1×2
    0.3829   -0.1173
v0_optim13 = -1.1514
```

```
[V_optim23, v0_optim23] = izlaz(X2,X3,N,1,1)
```

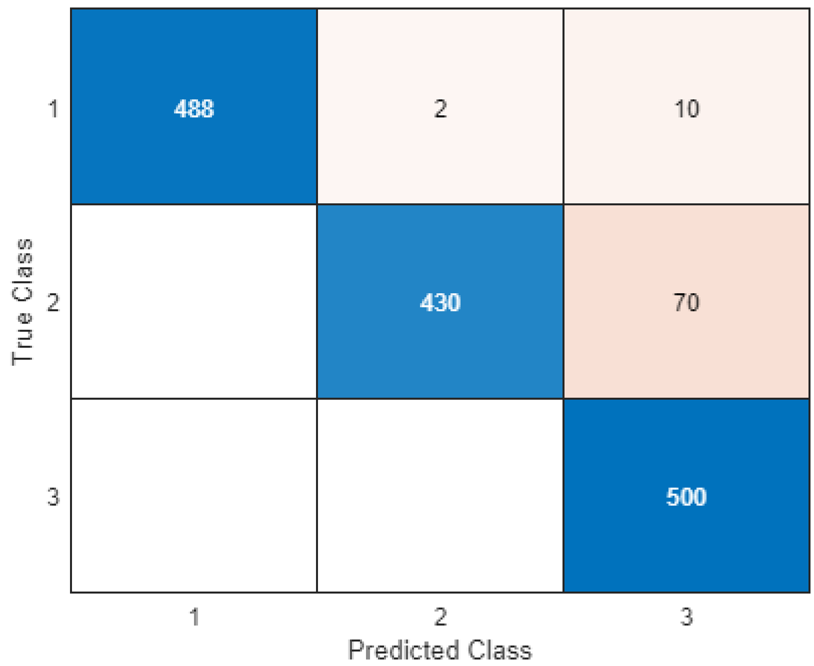
```
V_optim23 = 1×2
    0.1254   -0.1949
v0_optim23 = -0.0349
```

```
figure()
hold all
scatter(X1(1,:),X1(2,:),'ro');
scatter(X2(1,:),X2(2,:),'b*');
scatter(X3(1,:),X3(2,:),'yx');

X1_lin=-2:0.01:12;
X2_lin12=-(v0_optim12+V_optim12(1)*X1_lin)/V_optim12(2);
X2_lin13=-(v0_optim13+V_optim13(1)*X1_lin)/V_optim13(2);
X2_lin23=-(v0_optim23+V_optim23(1)*X1_lin)/V_optim23(2);
```

```
plot(X1_lin,X2_lin12,'g--');
plot(X1_lin,X2_lin13,'k--');
plot(X1_lin,X2_lin23,'m--');
```




```
X_pred =
predict(X_test,X_true,v0_optim12,v0_optim13,v0_optim23,V_optim12,V_optim13,V_
optim23);
figure()
C = confusionmat(X_true,X_pred);
confusionchart(C)
```



Dakle povećavanjem vrednosti željenog izlaza, penalizujemo vrednosti te klase za čije su veće vrednosti željenog izlaza. Što ima smisla s obzitom da će kriterijumska funkcija u tom slučaju imati veću vrednost. Takođe se vidi i pomeranje klasifikacionih prava u zavisnosti od toga da li favorizujemo crvenu klasu ili ne, što se dobija povećanjem koeficijenata u svakom od medjusobnih linearnih klasifikatora.

Projektovanje kvadratnog klasifikatora

Kada klase nisu linearno separabilne potrebno je uvesti nelinearne članove diskriminacione funkcije, u ovom slučaju kvadratnih. Diskriminaciona funkcija je definisana:

$$h(X) = X^T \mathrm{QX} + V^T X + V_0 < 0, \text{klasa 1}$$
$$h(X) = X^T \mathrm{QX} + V^T X + V_0 > 0, \text{klasa 2}$$

Zatim se linearizuje predstavljena relacija i predstavlja u obliku:

$$h(X) = \mathrm{AZ} + V_0$$

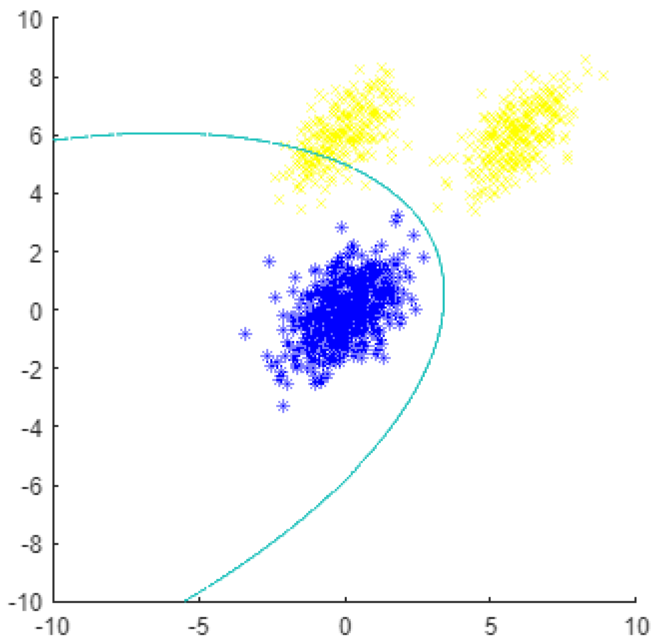
$$A = [sK_1 + (1-s)K_2]^{-1}(D_2 - D_1)$$

Gde je $Z = [Y^T \;\; X^T]^T$, K_1, K_2 – kovarijacione matrice vektora Z, D_1, D_2 – matematičkog očekivanja vektora Z. Dalje se V_0 i A dobijaju metodom željenog izlaza. Treba napomenuti da matrica A sadrži vrednosti koeficijenata uz kvadratne članove kao i linearne. Koristiće se kriterijumska funkcija najmanjeg kvadrata, kao i ista analitička relacija za težinski vektor W.

```
M11 = [0 0]'; s11 = [1 0.5; 0.5 1];
```

```
M12 = [5 0]'; s12 = [1.5 -0.7; -0.7 1.5];
M21 = [6 6]'; s21 = [1 0.6; 0.6 1];
M22 = [0 6]'; s22 = [1 0.6; 0.6 1];

K1prva = mvnrnd(M11,s11,N);

K2prva = mvnrnd(M21,s21,N);
K2druga = mvnrnd(M22,s22,N);

X = K1prva;
pom2 = rand(N,1);
Y = (pom2<=0.5).*K2prva + (pom2>0.5).*K2druga;

X = X';
Y = Y';

Gama=ones(2*N,1);

U=[-1*ones(1,N),ones(1,N);...
    -1*X,Y;...
    -1*(X(1,:).^2),(Y(1,:).^2);...
    -1*(X(2,:).^2),(Y(2,:).^2);...
    -2*(X(1,:).*X(2,:)),2*(Y(1,:).*Y(2,:))];


W=(U*U')^(-1)*U*Gama;

v0=W(1);
V1=W(2);
V2=W(3);
Q11=W(4);
Q22=W(5);
Q12=W(6);

x1=-10:0.01:10;
x2=-10:0.01:10;

h=zeros(length(x1),length(x2));

for i=1:length(x1)
    for j=1:length(x2)
        h(i,j)=v0+V1*x1(i)+V2*x2(j)+Q11*x1(i)^2+Q22*x2(j)^2+Q12*x1(i)*x2(j);
    end
end

figure()
hold all
scatter(X(1,:), X(2,:),'blue', '*')
scatter(Y(1,:), Y(2,:),'yellow',"x")
contour(x1,x2,h,[0 0]);
```

```
axis equal
```



Zadatak 4 - Klasterizacije

Primenićemo K-means algoritam za klasterizaciju nekoliko gausovskih linearno separabilnih klasa.

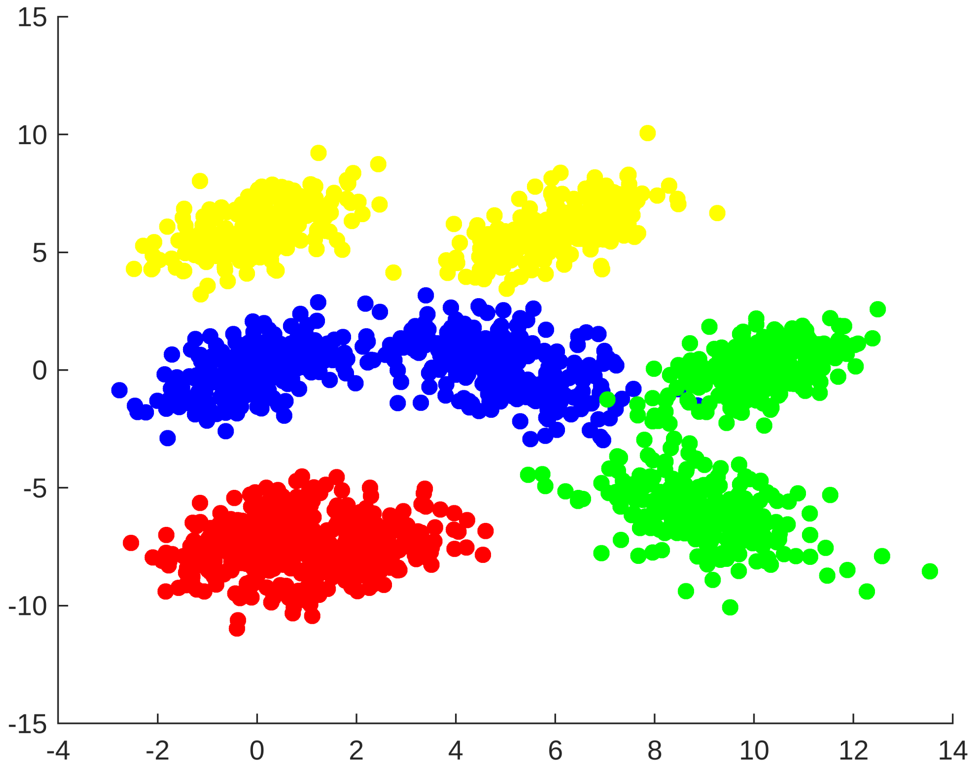
K means algoritam klasterizacije funkcioniše tako što za zadati broj klasa reklasifikacijom datih odbiraka optimizuje neku kriterijumsku funkciju. Ta kriterijumska funkcija se sazniva na kvadratu rastojanja unutar klasnih odbiraka i centra centriole koja formira klaster.

$$J = \frac{1}{N} \sum_{r=1}^{L} \sum_{j=1}^{\mathrm{Nr}} \| \mathrm{Xj}^{(r)} - M_r \|^2$$

Ponavljanjem procesa reklasifikacije, algoritam se završava ukoliko predjemo odredjeni broj iteracija ili su klasteri ostali nepromenjeni.

```
clc; clear; close all;

N=500;

M11 = [0 0]'; s11 = [1 0.5; 0.5 1];
M12 = [5 0]'; s12 = [1.5 -0.7; -0.7 1.5];
M21 = [6 6]'; s21 = [1 0.6; 0.6 1];
M22 = [0 6]'; s22 = [1 0.6; 0.6 1];

K1prva = mvnrnd(M11,s11,N);
K1druga = mvnrnd(M12,s12,N);
K2prva = mvnrnd(M21,s21,N);
K2druga = mvnrnd(M22,s22,N);

pom1 = rand(N,1);
A = (pom1<=0.5).*K1prva + (pom1>0.5).*K1druga;
pom2 = rand(N,1);
B = (pom2<=0.5).*K2prva + (pom2>0.5).*K2druga;

M31 = [10 0]'; s31 = [1 0.5; 0.5 1];
M32 = [9 -6]'; s32 = [1.5 -0.7; -0.7 1.5];
M41 = [0 -7]'; s41 = [1 0.6; 0.6 1];
M42 = [2 -8]'; s42 = [1 0.6; 0.6 1];

K1prva = mvnrnd(M31,s31,N);
K1druga = mvnrnd(M32,s32,N);
K2prva = mvnrnd(M41,s41,N);
K2druga = mvnrnd(M42,s42,N);

pom1 = rand(N,1);
C = (pom1<=0.5).*K1prva + (pom1>0.5).*K1druga;
pom2 = rand(N,1);
D = (pom2<=0.5).*K2prva + (pom2>0.5).*K2druga;

figure(1)
hold all
```

```
scatter(A(:,1), A(:,2),'blue', 'filled')
scatter(B(:,1), B(:,2),'yellow',"filled")
scatter(C(:,1), C(:,2),'green',"filled")
scatter(D(:,1), D(:,2),'red',"filled")
```



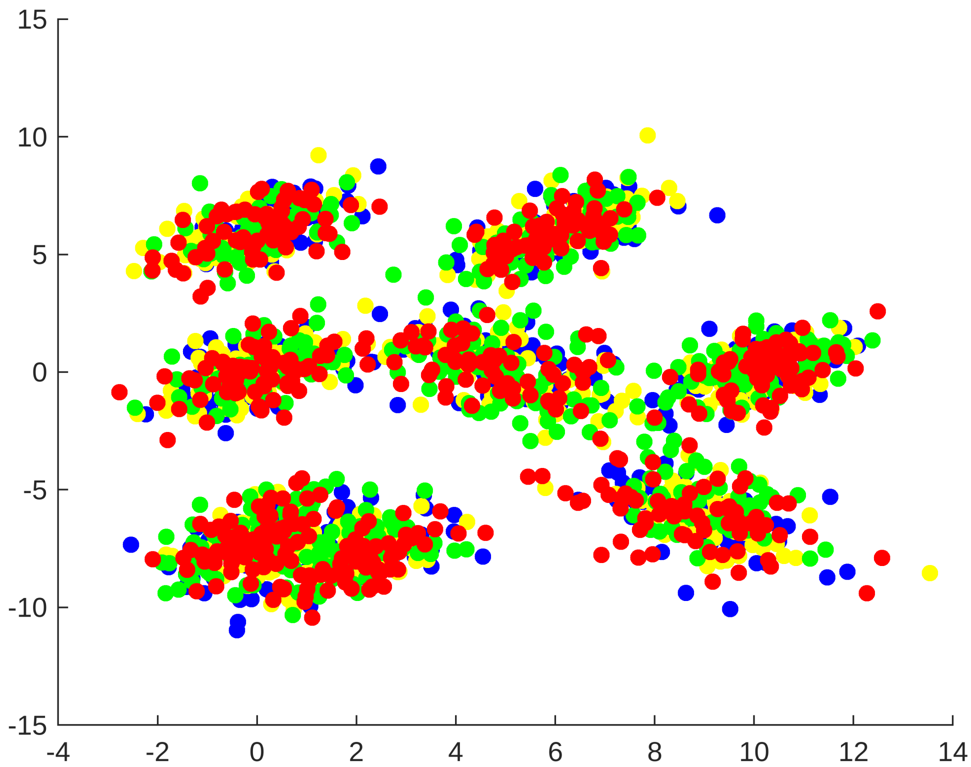

```
pom = rand(1,4*N);

X1 = []; X2 = []; X3 = []; X4 = [];
A = A';
B = B';
C = C';
D = D';
```

Početna klasterizacija, na nasumičan način delimo odbirke po klasama

```
for i=1:N
    if pom(i)<0.25
        X1 = [X1 A(:,i)];
    else
        if and(pom(i)>=0.25, pom(i)<0.5)
            X2 = [X2 A(:,i)];
        else
            if and(pom(i)>=0.5, pom(i)<0.75)
                X3 = [X3 A(:,i)];
```

```
            else
                X4 = [X4 A(:,i)];
            end
        end
    end
end

for i=1:N
    if pom(i)<0.25
        X1 = [X1 B(:,i)];
    else
        if and(pom(i)>=0.25, pom(i)<0.5)
            X2 = [X2 B(:,i)];
        else
            if and(pom(i)>=0.5, pom(i)<0.75)
                X3 = [X3 B(:,i)];
            else
                X4 = [X4 B(:,i)];
            end
        end
    end
end

for i=1:N
    if pom(i)<0.25
        X1 = [X1 C(:,i)];
    else
        if and(pom(i)>=0.25, pom(i)<0.5)
            X2 = [X2 C(:,i)];
        else
            if and(pom(i)>=0.5, pom(i)<0.75)
                X3 = [X3 C(:,i)];
            else
                X4 = [X4 C(:,i)];
            end
        end
    end
end

for i=1:N
    if pom(i)<0.25
        X1 = [X1 D(:,i)];
    else
        if and(pom(i)>=0.25, pom(i)<0.5)
            X2 = [X2 D(:,i)];
        else
            if and(pom(i)>=0.5, pom(i)<0.75)
                X3 = [X3 D(:,i)];
            else
                X4 = [X4 D(:,i)];
```

```
            end
        end
    end
end

figure(2)
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```




```
%ideja za variranje pocetne klasterizacije, menjaj sanse za nasumicno
%dodeljivanje

N1 = length(X1(1,:));
N2 = length(X2(1,:));
N3 = length(X3(1,:));
N4 = length(X4(1,:));
```

Računanje srednje vrednosti

```
M1 = mean(X1,2);
```

```
M2 = mean(X2,2);
M3 = mean(X3,2);
M4 = mean(X4,2);
```

Implementacija algoritma. Kreće se kroz odbirak svake klase i tražimo minimalno rastojanje od centara centriola. Na taj način se minimizuje gore predstavljen kriterijum.

```
lmax = 100;
l = 1;
reklas = 1;

while (l < lmax) && reklas
    X1pom = [];
    X2pom = [];
    X3pom = [];
    X4pom = [];
    reklas = 0;

    for i=1:N1
        d1 = sum((X1(:,i)-M1).^2);
        d2 = sum((X1(:,i)-M2).^2);
        d3 = sum((X1(:,i)-M3).^2);
        d4 = sum((X1(:,i)-M4).^2);

        [~,dind]= min([d1,d2,d3,d4]);

        if dind == 2
            X2pom = [X2pom X1(:,i)];
            reklas = 1;
        end

        if dind == 3
            X3pom = [X3pom X1(:,i)];
            reklas = 1;
        end

        if dind == 4
            X4pom = [X4pom X1(:,i)];
            reklas = 1;
        end

        if dind == 1
            X1pom = [X1pom X1(:,i)];

        end

    end

    for i=1:N2
```

```
        d1 = sum((X2(:,i)-M1).^2);
        d2 = sum((X2(:,i)-M2).^2);
        d3 = sum((X2(:,i)-M3).^2);
        d4 = sum((X2(:,i)-M4).^2);

        [~,dind]= min([d1,d2,d3,d4]);

        if dind == 2
            X2pom = [X2pom X2(:,i)];
        end

        if dind == 3
            X3pom = [X3pom X2(:,i)];
            reklas = 1;
        end

        if dind == 4
            X4pom = [X4pom X2(:,i)];
            reklas = 1;
        end

        if dind == 1
            X1pom = [X1pom X2(:,i)];
            reklas = 1;

        end

    end


    for i=1:N3
        d1 = sum((X3(:,i)-M1).^2);
        d2 = sum((X3(:,i)-M2).^2);
        d3 = sum((X3(:,i)-M3).^2);
        d4 = sum((X3(:,i)-M4).^2);

        [~,dind]= min([d1,d2,d3,d4]);

        if dind == 2
            X2pom = [X2pom X3(:,i)];
            reklas = 1;
        end

        if dind == 3
            X3pom = [X3pom X3(:,i)];

        end

        if dind == 4
            X4pom = [X4pom X3(:,i)];
```

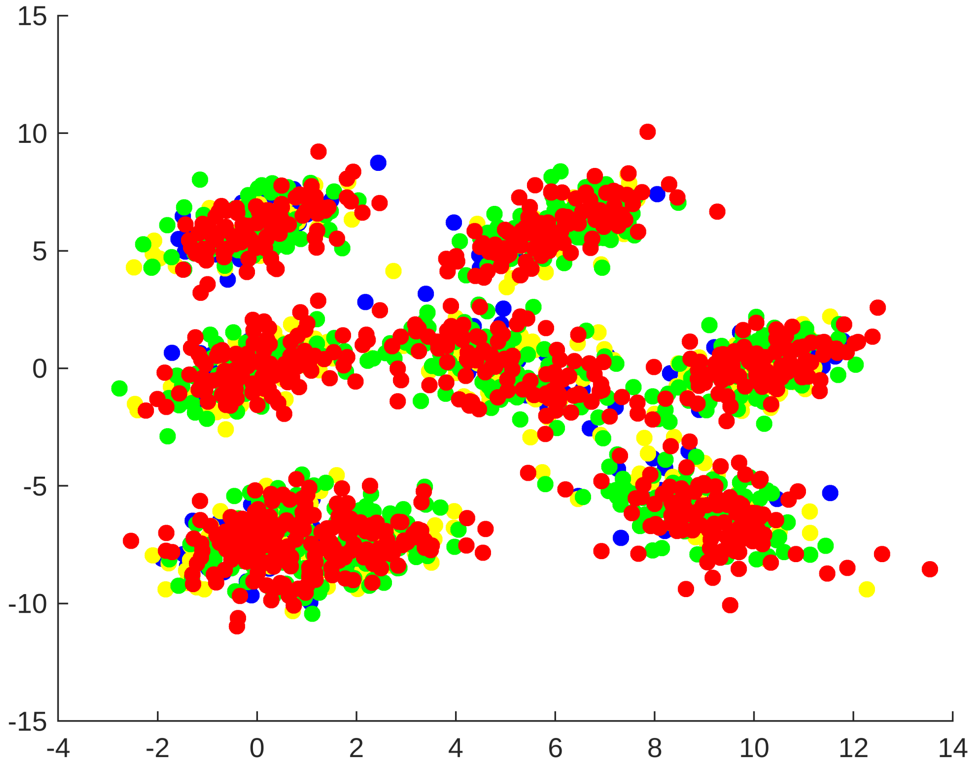
```
            reklas = 1;
        end

        if dind == 1
            X1pom = [X1pom X3(:,i)];
            reklas = 1;

        end

    end

    for i=1:N4
        d1 = sum((X4(:,i)-M1).^2);
        d2 = sum((X4(:,i)-M2).^2);
        d3 = sum((X4(:,i)-M3).^2);
        d4 = sum((X4(:,i)-M4).^2);

        [~,dind]= min([d1,d2,d3,d4]);

        if dind == 2
            X2pom = [X2pom X4(:,i)];
            reklas = 1;
        end

        if dind == 3
            X3pom = [X3pom X4(:,i)];
            reklas = 1;
        end

        if dind == 4
            X4pom = [X4pom X4(:,i)];

        end

        if dind == 1
            X1pom = [X1pom X4(:,i)];
            reklas = 1;
        end

    end
    clear X1 X2 X3 X4
    X1 = X1pom;
    X2 = X2pom;
    X3 = X3pom;
    X4 = X4pom;

    figure()
%     hold all
%     scatter(X1(1,:), X1(2,:),'blue', 'filled')
%     scatter(X2(1,:), X2(2,:),'yellow',"filled")
```

```
%     scatter(X3(1,:), X3(2,:),'green',"filled")
%     scatter(X4(1,:), X4(2,:),'red',"filled")
%     title(['Iteracija broj' num2str(l)]);

    if  isempty(X1)
        N1 = 0;
    else
        M1 = mean(X1,2);
        N1 = length(X1(1,:));
    end

    if  isempty(X2)
        N2 = 0;
    else
        M2 = mean(X2,2);
        N2 = length(X2(1,:));
    end

    if  isempty(X3)
        N3 = 0;
    else
        M3 = mean(X3,2);
        N3 = length(X3(1,:));
    end

    if  isempty(X4)
        N4 = 0;
    else
        M4 = mean(X4,2);
        N4 = length(X4(1,:));
    end
    l = l+1;


end
```

Analiza uticaja početne podele na broj iteracija klasterizacije

Postramo uticaj klasterizacije kada u početnom rasporedu damo veci broj zelene i crvene klase.

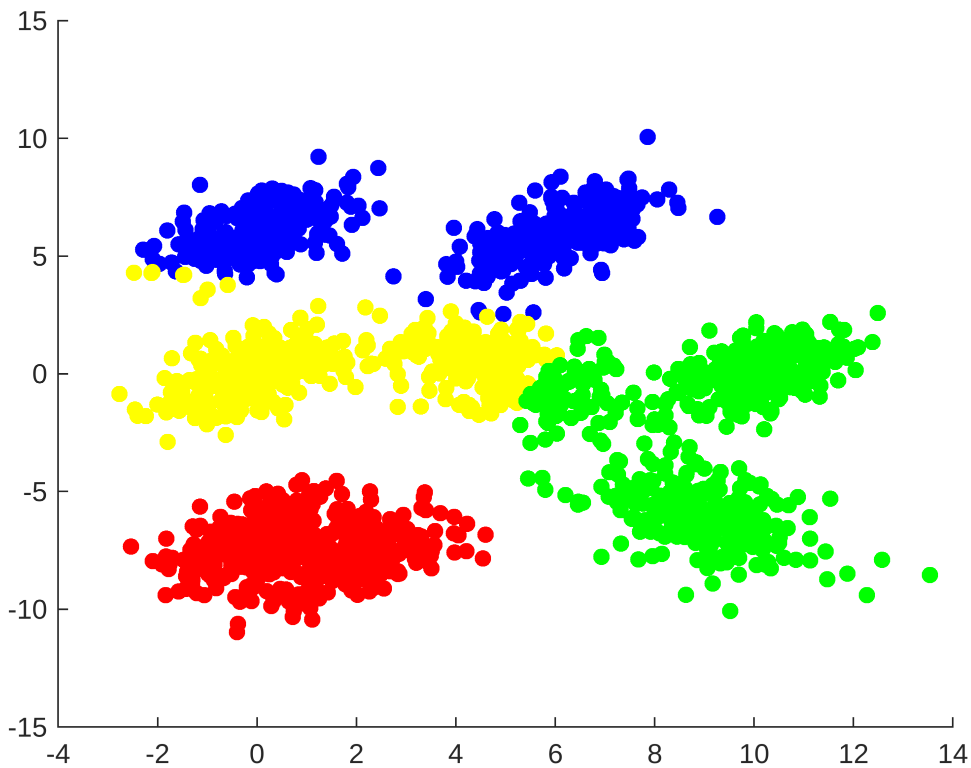
```
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.1,0.3,0.6);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



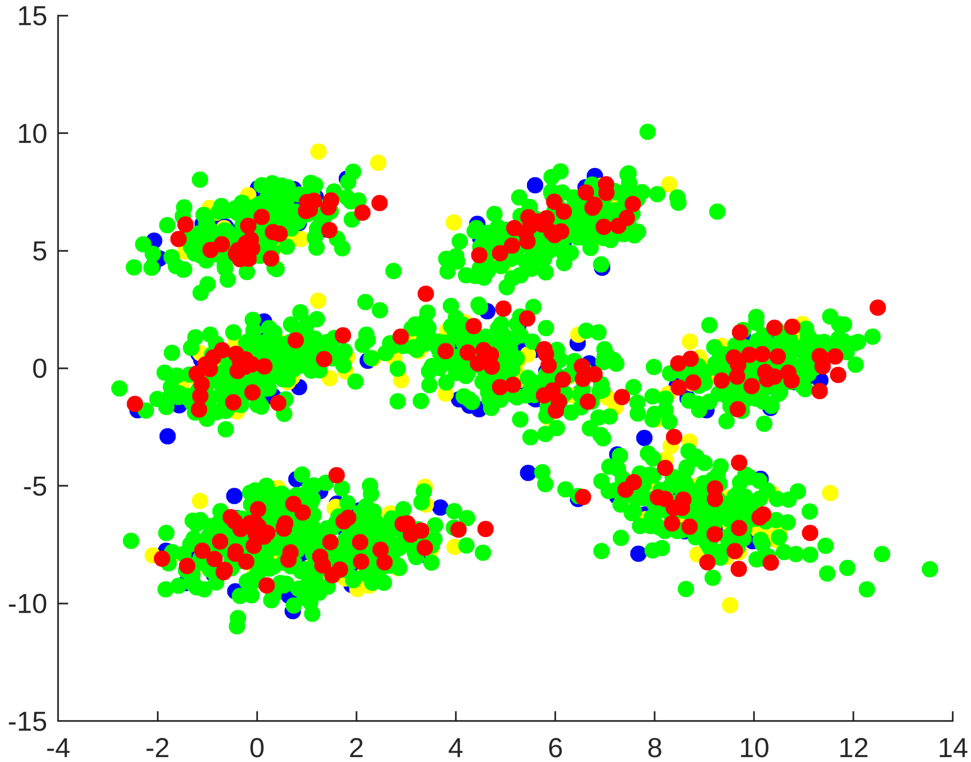

```
[X1,X2,X3,X4,briter] = Km(X1,X2,X3,X4,100);
```

Rezultat algoritma gde je crvena dominantna

```
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



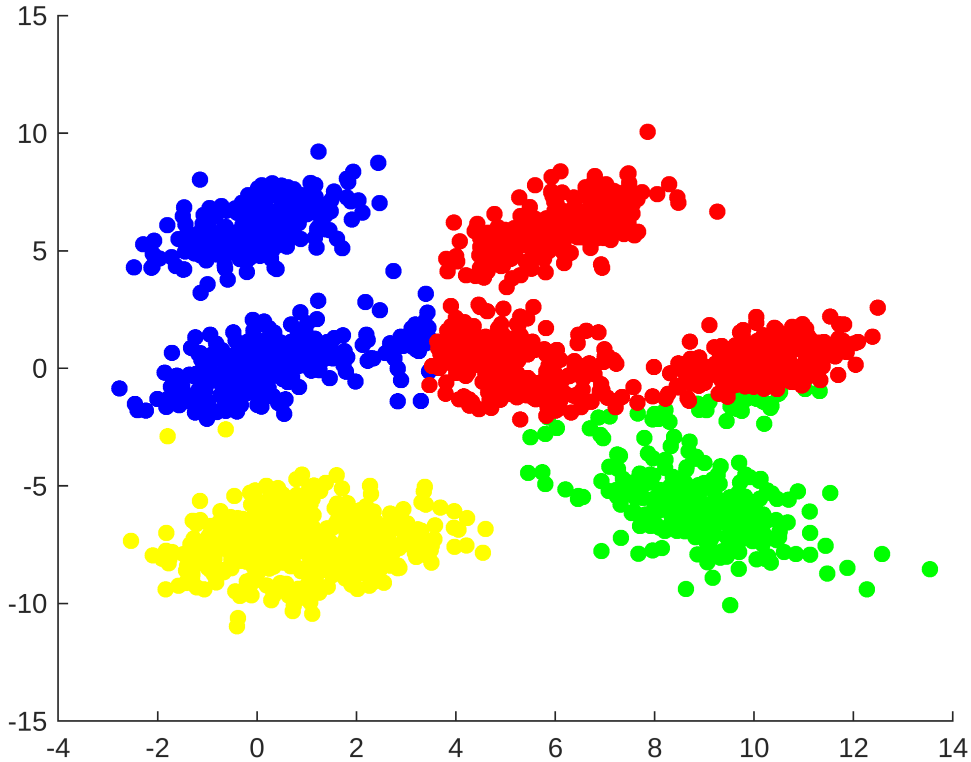

```
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.1,0.2,0.9);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```




```
[X1,X2,X3,X4,briter] = Km(X1,X2,X3,X4,100);
```

Rezultat algoritma gde je zelena dominantna

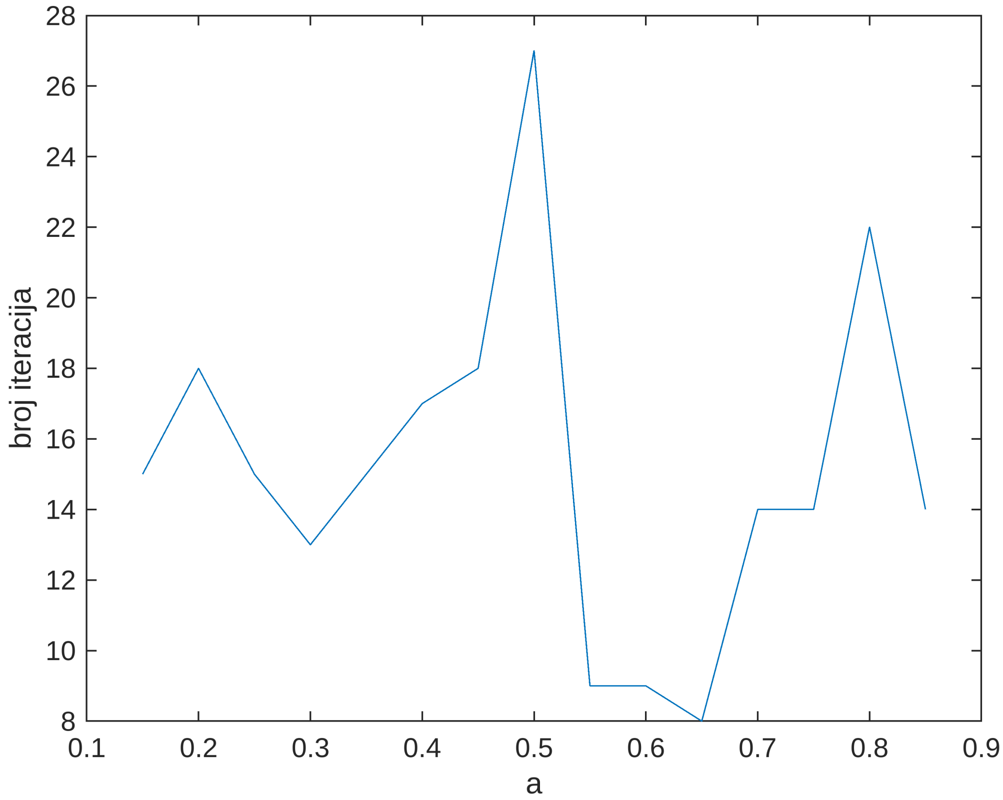
```
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



Gledajući dobijene rezultate može se zaključiti da je algoritam relativno neosetljiv na početnu podelu. Različita početna podela daje različite klastere. Naime, ukoliko uzmemo klasu koja je izdvojenija od drugih, i nju postavimo da bude dominantna u početnom sampl-u, brže će se završiti klasterizacija. Drugim rečima ukoliko imamo izdvojeniju klasu i nju over samplujemo algoritam će se brže završiti. Sad cemo varirati sempling jedne klase i gledati broj iteracija klasterizacije.

```
iters = [];
vred = [];
for a=0.15:0.05:0.85
    [X1,X2,X3,X4] = podela(A,B,C,D,N,0.1,a,0.9);
    [~,~,~,~,iterNum] = Km(X1,X2,X3,X4,100);
    iters = [iters, iterNum];
    vred = [vred a];
end
```

```
figure()
plot(vred,iters)
xlabel('a')
ylabel('broj iteracija')
```



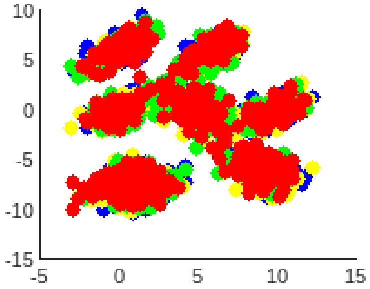
Ako se pogleda zavisnost broja iteracija od proporcije broja odbiraka u početnoj inicijalizaciji, kao i prethodne grafike klasterizacije moze se zaključiti:

1. Algoritam je neosetljiv na stohastičku početnu podelu.
2. Prosečan broj iteracija je negde oko 15, što nam govori da je ovaj metod u proseku relativno ne zahtevan za kompjutaciju.

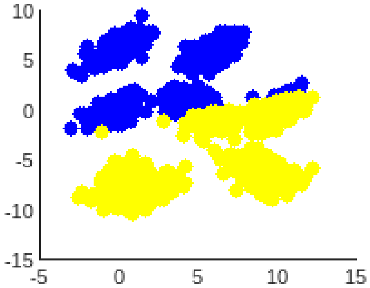
Dalje predstavljamo rezultate klasifikacije kada se uzima broj klasa 2,3,4.

Dve klase:

```
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.25,0.5,0.75);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```

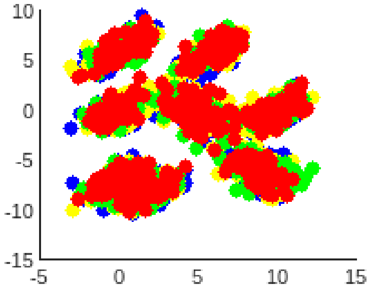



```
[X1,X2,briter] = Km2(X1,X2,100);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
```



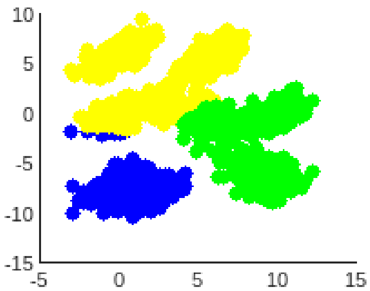
Tri klase:

```
%tri klase
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.25,0.5,0.75);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



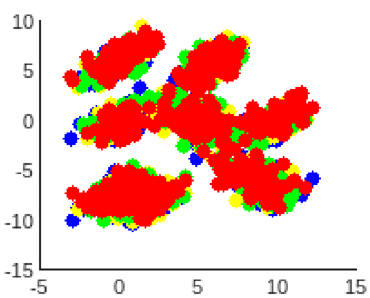

```
[X1,X2,X3,briter] = Km3(X1,X2,X3,100);
figure()
hold all
```

```
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
```

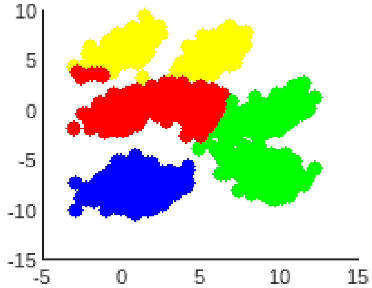


Četiri klase:

```
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.25,0.5,0.75);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```




```
[X1,X2,X3,X4,briter] = Km(X1,X2,X3,X4,100);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



Možemo videti da je najbolje poklapanje kada je kriterijum minimalan, tj kada se klasifikacija vrši sa četiri klase. U narednim razmatranjima sa klasterizacijom maksimalne verodostojnosti nije vršena analiza osetljivosti klusterizacija za apriorni broj klasa jer je ponasanje slično kao i za Cmeans algoritam. Eventualno bi videli bolju konvergenciju ka teorijskom minimumu kriterijuma, ali sama zavinost konvergencije algoritma zavisi na isti način kao i kod Cmeans algoritma. Biće predstavljene analize za incijalnu klasterizaciju, kao i grafik zavisnosti broja iteracija od proporcija inicijalne klasterizacije.

Klasterizacija metodom maksimalne verodostojnosti

Dalje je primenjen metod klasterizacije na bazi maksimalne verodnostojnosti. Pretpostavka ove metode jeste da su funkcije gustine verovatnoće oblika iz pojedinih klasa Gausovske i stoga da je združena funkcija gustine zapravo mešavina tj zbir Gausovih raspodela. S toga se problem svodi na maksimizaciju funkcije:

$$\prod_{j=1}^{N} f(X_j)$$

Po parametrima apriorne verovatnoće, srednje vrednosti i kovarijacione matrice. Uz uslov normalizacije dobija se sledeća kriterijumska funkcija:

$$J = \sum_{j=1}^{N} \ln(f(X_j)) - \eta\left(\sum_{i=1}^{L} P_i - 1\right)$$

Procedurom tipičnom za optimizacione probleme Lagranžovim multiplikatorom traženjem parcijalnih izvoda dobijaju se sledeći izrazi:

$$M_i = \frac{1}{N_i}\sum_{j=1}^{N} q_i(X_j)X_j$$

$$\Sigma_i = \frac{1}{N_i}\sum_{j=1}^{N} q_i(X_j)(X_j - M_i)(X_j - M_i)^T$$

Uz to da je poznato da važi:

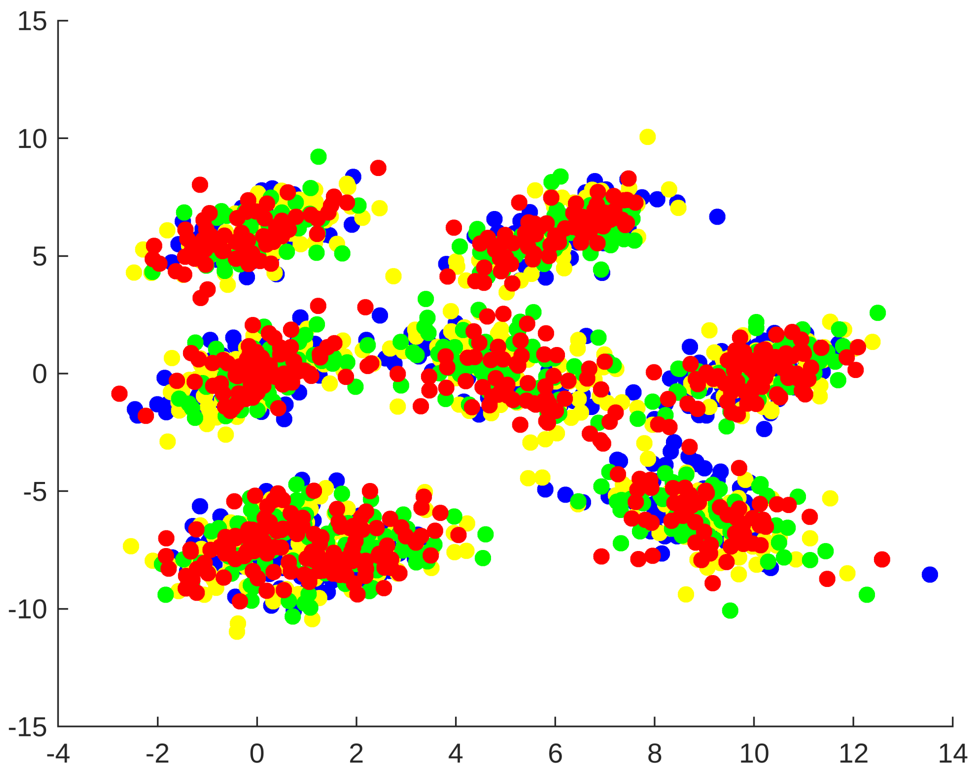
$$q_i(X_j) = \frac{P_i f_i(X_j)}{\sum_{k=1}^{L} P_k f_k(X_j)}$$

Ovi izrazi predstavljaju srž mehanizma koji pokreće metodu maskimalne verodostojnosti. S obzirom da iterativno vršimo iterativno *popravke* gore navedenih parametara, prema formulama koje važe ukoliko je gornja kriterijumska funkcija maksimalna. Intuitivno se može očekivati da je ova metoda osetljivija na početnu klasterizaciju s obzirom da se oslanja na kovarijansu datih klasa.

Isto tako, kao što je parametarska granica između klasa kod Cmean metode više bisektora, ovde su klasifikacione granice krive drugog reda, što znači da imaju potencijal da vrše klasterizacije i nelinearno separabilnih klasa sve dok važi pretpostavka da se njihova raspodela može modelovati gausovski. Očekuje se da će algoritam pravilno izvršiti klasifikaciju kod slučaja nelinearno separabilnih klasa.

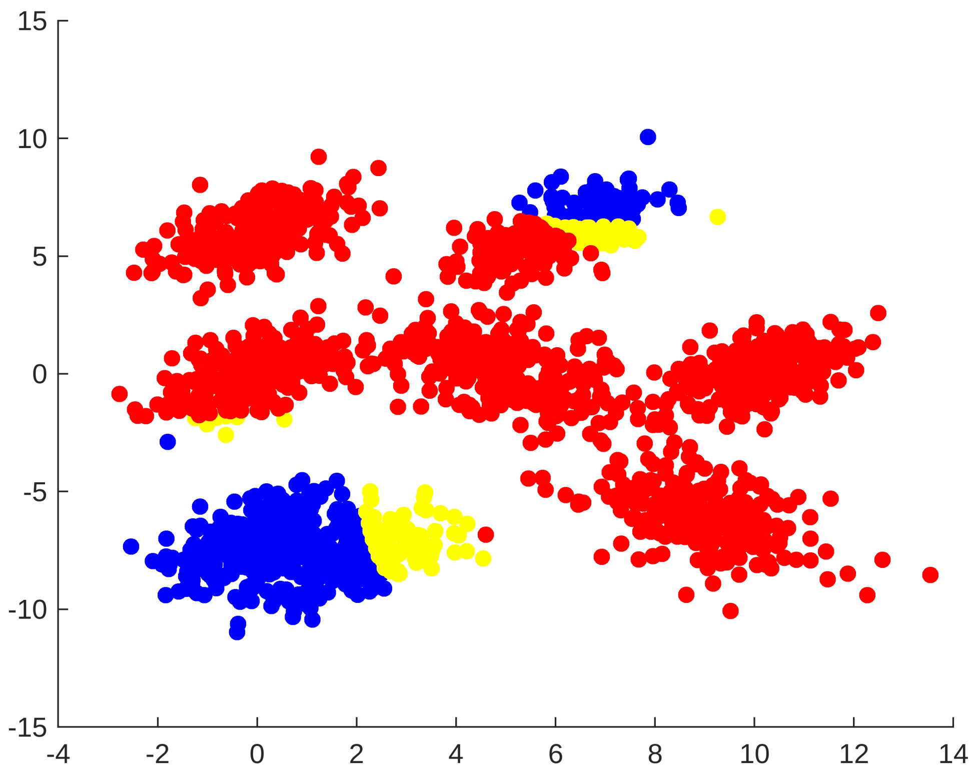
Klasterizacija linearno separabilnih klasa

```
[X1,X2,X3,X4] = podela(A,B,C,D,N,0.25,0.5,0.75);
figure()
hold all
scatter(X1(1,:), X1(2,:),'blue', 'filled')
scatter(X2(1,:), X2(2,:),'yellow',"filled")
scatter(X3(1,:), X3(2,:),'green',"filled")
scatter(X4(1,:), X4(2,:),'red',"filled")
```



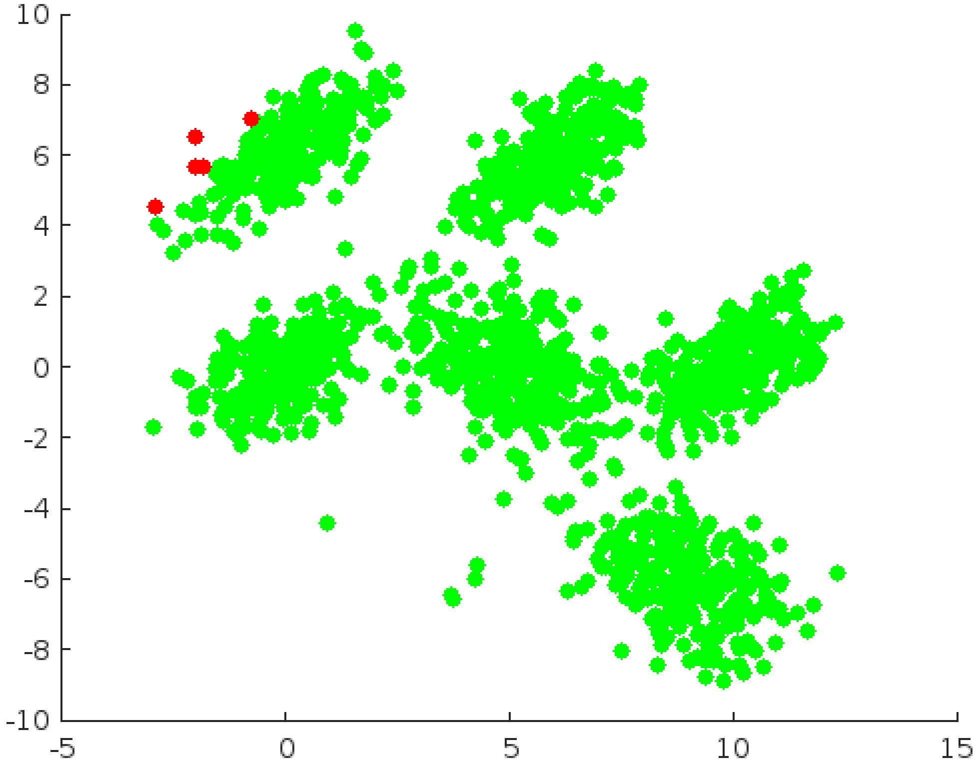

```
[qp1, qp2, qp3, qp4, K, briter] = MaxL(X1, X2, X3, X4, N);
```

```
X1r = [];
X2r = [];
X3r = [];
X4r = [];


for i = 1:4*N
    [M,I] = max([qp1(i),qp2(i),qp3(i), qp4(i)]);

    if  I == 1
        X1r = [X1r K(:,i)];
    elseif I == 2
        X2r = [X2r K(:,i)];
    elseif I == 3
        X3r = [X3r K(:,i)];
    elseif I == 4
        X4r = [X4r K(:,i)];
    end
end


figure()
hold all
if ~isempty(X1r)
    scatter(X1r(1,:), X1r(2,:),'blue', 'filled')
end
if ~isempty(X2r)
    scatter(X2r(1,:), X2r(2,:),'yellow',"filled")
end
if ~isempty(X3r)
    scatter(X3r(1,:), X3r(2,:),'green',"filled")
end
if ~isempty(X4r)
    scatter(X4r(1,:), X4r(2,:),'red',"filled")
end
hold off
```

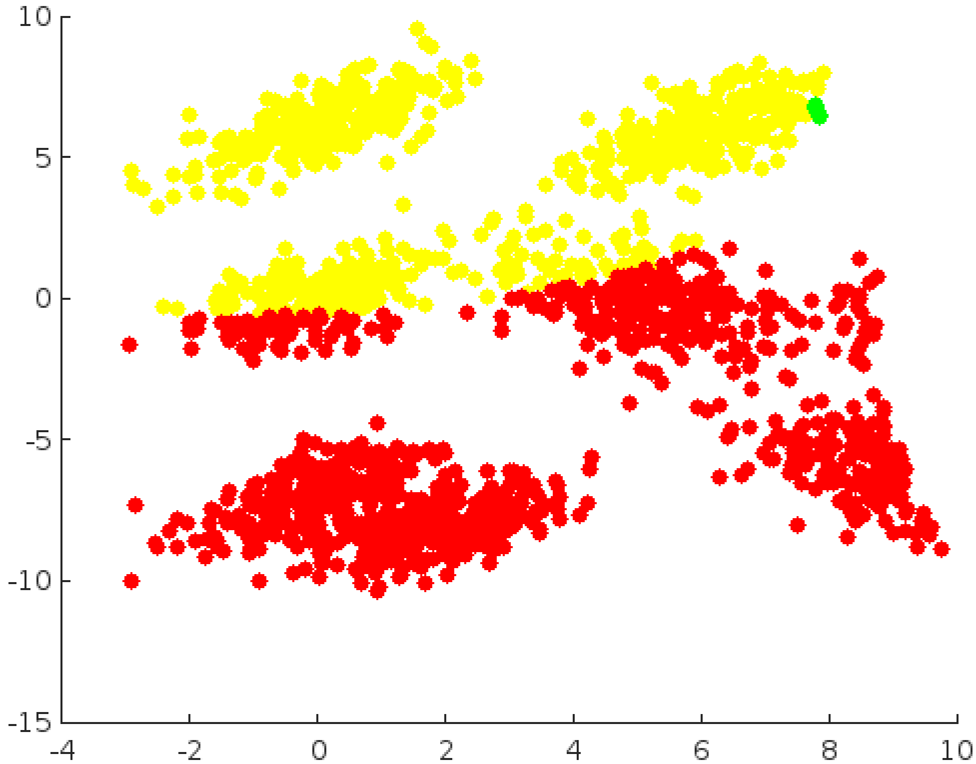


Može se zaključiti da metoda maksimalne verodostojnosti i jeste dosta osetljiva na početne inicijalizacije kao što se može videti na naredna dva grafika:

```
fig1 = openfig('iter1.fig','visible');
```

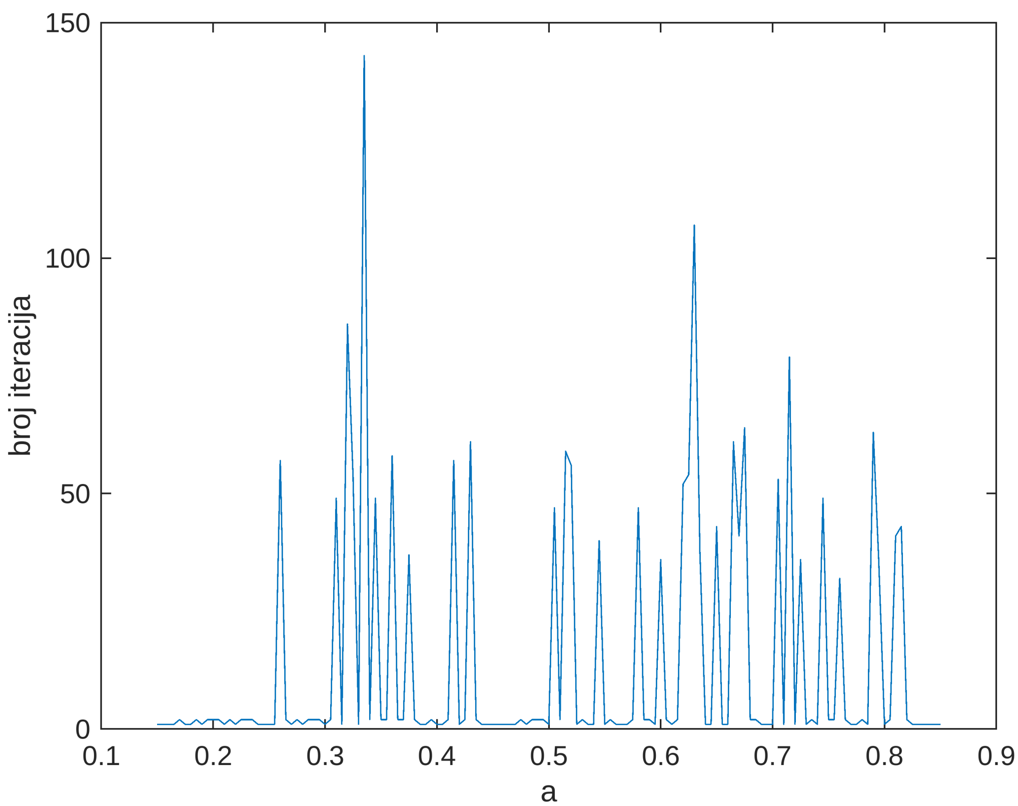



```
fig2 = openfig('iter2.fig','visible');
```



```
iters = [];
vred = [];
for a=0.15:0.005:0.85
    [X1,X2,X3,X4] = podela(A,B,C,D,N,0.1,a,0.9);
    [~,~,~,~,iterNum] = MaxL(X1,X2,X3,X4,100);
    iters = [iters, iterNum];
    vred = [vred a];
end
```

```
figure()
plot(vred,iters)
xlabel('a')
ylabel('broj iteracija')
```



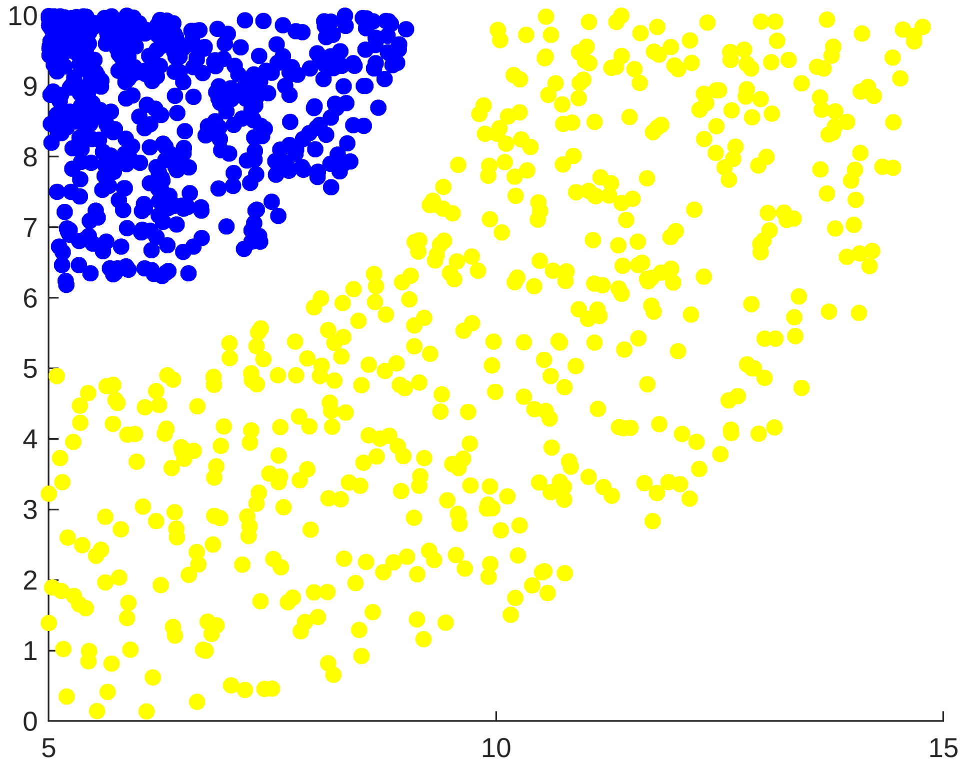
Ako se pogleda zavisnost broja iteracija od proporcije broja odbiraka u početnoj inicijalizaciji moze se zaključiti:

1. Jako je šumovit grafik, što nam govori da je osetljivost na početnu podelu velika.

2. Postoje dva pika u broju iteracija negde na 0.4 i 0.6. Poreklo dva peaka je usred implementacije same proporcije koja je demonstrirala zavisnost broja iteracija od početne podele. Minimalan broj peakova je kod slučaja gde postoji ili veliki broj odbiraka jedne klase, ili u slučaju kada su klase relativno jednako raspoređene.
3. Prosečan broj iteracija je negde oko 35, što nam govori da je ovaj metod u proseku zahtevniji od Cmeans-a.

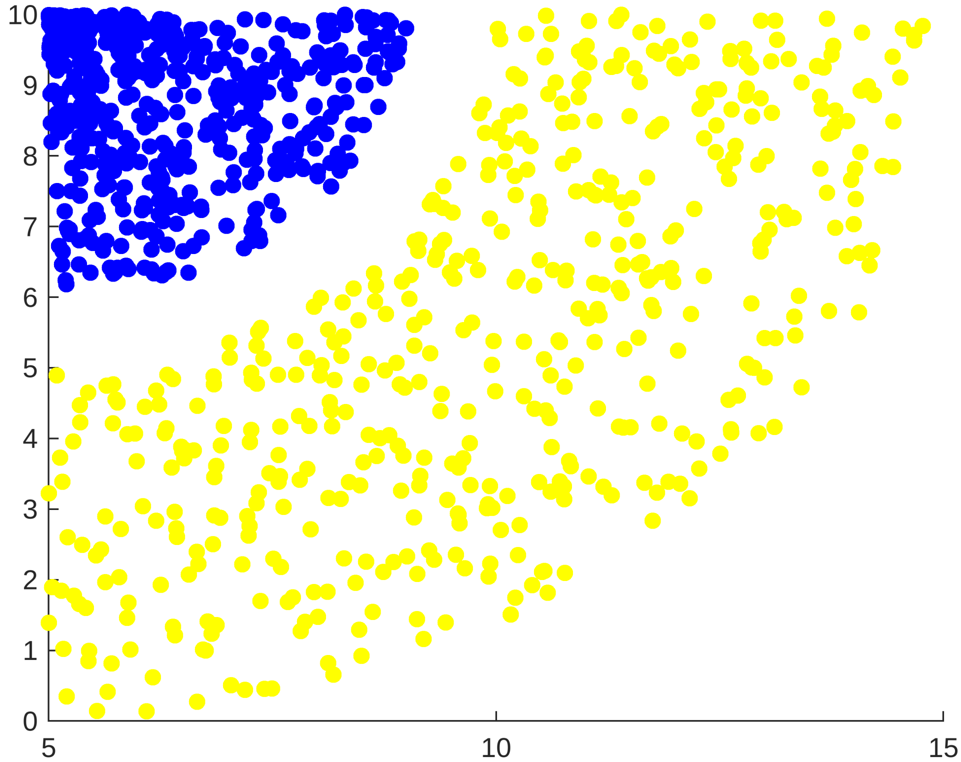
Klasterizacija linearno neseparabilnih klasa

```
N = 500;

C = [5;10];
R = 4*rand(1,N);
Teta = (-pi/2)*rand(1,N);
X1 = [R.*cos(Teta); R.*sin(Teta)];
X1 = X1 + C*ones(1,N);

C = [5;10];
R = 10-5*rand(1,N);
Teta = (-pi/2)*rand(1,N);
X2 = [R.*cos(Teta); R.*sin(Teta)];
X2 = X2 + C*ones(1,N);

figure()
hold all
if ~isempty(X1)
    scatter(X1(1,:), X1(2,:),'blue', 'filled')
end
if ~isempty(X2)
    scatter(X2(1,:), X2(2,:),'yellow',"filled")
end
hold off
```



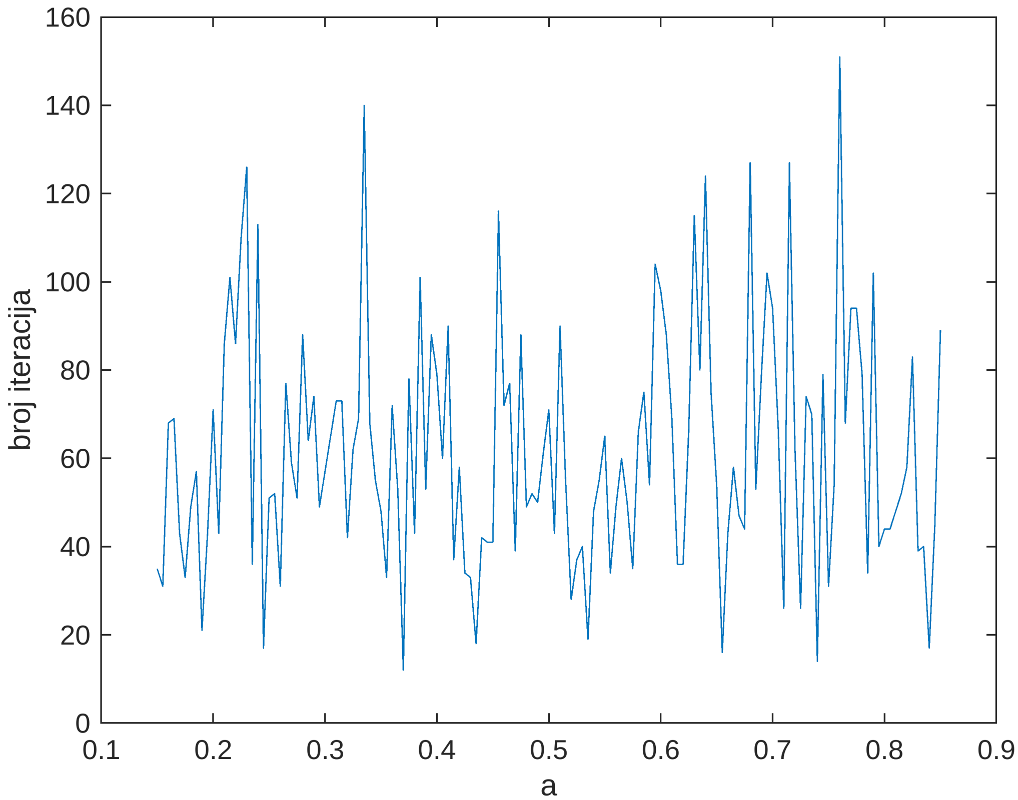

```
[qp1, qp2, K, briter] = Max2L(X1, X2, N);

X1r = [];
X2r = [];

for i = 1:2*N
    [M,I] = max([qp1(i),qp2(i)]);

    if  I == 1
        X1r = [X1r K(:,i)];
    elseif I == 2
        X2r = [X2r K(:,i)];
    end
end


figure()
hold all
if ~isempty(X1r)
    scatter(X1r(1,:), X1r(2,:),'blue', 'filled')
end
if ~isempty(X2r)
    scatter(X2r(1,:), X2r(2,:),'yellow',"filled")
end
```

```
hold off
```




```
iters = [];
vred = [];
for a=0.15:0.005:0.85

    C = [5;10];
    R = 4*rand(1,N);
    Teta = (-pi/2)*rand(1,N);
    X1 = [R.*cos(Teta); R.*sin(Teta)];
    X1 = X1 + C*ones(1,N);

    C = [5;10];
    R = 10-5*rand(1,N);
    Teta = (-pi/2)*rand(1,N);
    X2 = [R.*cos(Teta); R.*sin(Teta)];
    X2 = X2 + C*ones(1,N);

    [~,~,~,iterNum] = Max2L(X1,X2,100);
    iters = [iters, iterNum];
    vred = [vred a];
end
```

```
figure()
plot(vred,iters)
xlabel('a')
ylabel('broj iteracija')
```



Iz dobijenih rezultata može se zaključiti da MaxLikelihood može da se primeni i na slučajeve gde naočigled raspored nije gausovski, i da metoda klasterizuje takve slučaje kako treba. Svi raniji zaključci važe i ovde, s tim što je broj iteracija dosta veći u proseku i iznosi negde oko 70. Izgleda da uticaj stohastičnosti kod metode opada sa brojem klasa koje treba klasifikovati.