eliminates the overlapping issue.

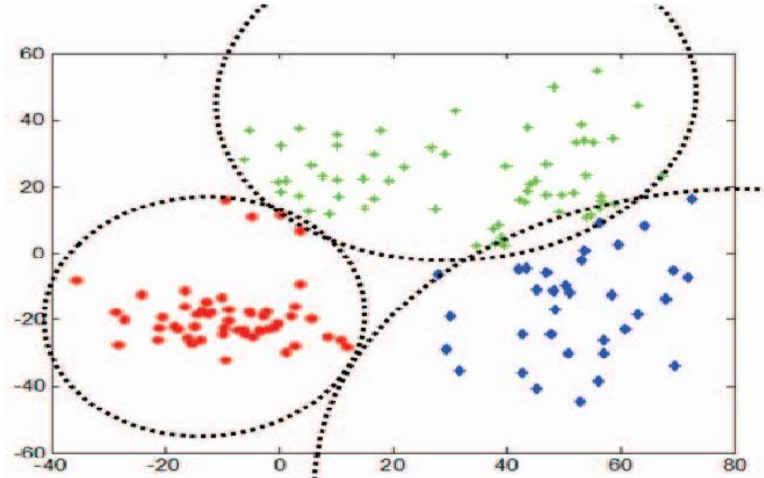


Fig 5 Example clustering by K-means

ΕΡΓΑΣΙΑ 3

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΚΑΡΑΠΕΠΕΡΑ ΕΛΠΙΔΑ | 57423 | 2020

ΑΣΚΗΣΗ 3.1

3.1.α

```
function [output] = g(d, m_i, Sigma_i, P_w_i)
    x      = sym('x',[size(m_i) 1],'real');
    output = -1/2*(x-m_i)'*inv(Sigma_i)*(x-m_i)-d/2*log(2*pi)-
1/2*log(det(Sigma_i))+log(P_w_i);
end
```

3.1.β

```
function [output] = euclidean_dist(x1, x2)
    output = norm(x1-x2);
end
```

3.1.γ

Η mahalanobis distance είναι το μέτρο της απόστασης ενός σημείου από μία κατανομή.

```
function [output] = mahalanobis_dist(x, m, Sigma)
    output = sqrt((x-m)'*inv(Sigma)*(x-m));
end
```

ΑΣΚΗΣΗ 3.2

3.2.1

```
disp('3.2.1 starting...');

[m_1_1 , Sigma_1_1] = Gaussian_ML_estimate(SAMPLES(:,1)');
syms g_1(x);
g_1_1(x)  = g(1, m_1_1, Sigma_1_1, P_w_1);

[m_1_2 , Sigma_1_2] = Gaussian_ML_estimate(SAMPLES(:,4)');
syms g_2(x);
g_1_2(x)  = g(1, m_1_2, Sigma_1_2, P_w_2);

g_1_3     = 0; %Since p=0;

d_1(x)    = g_1_1(x) - g_1_2(x);
w1_0      = double(solve(d_1(x)==0));

disp(['Separation points of w1, w2 using only x1: ',num2str(w1_0(1)),' ,
',num2str(w1_0(2))]);
```

```
function [m_hat,S_hat]=Gaussian_ML_estimate(X)
    [l,N]=size(X);
    m_hat=(1/N)*sum(X')';
    S_hat=zeros(l);
    for k=1:N
        S_hat=S_hat+(X(:,k)-m_hat)*(X(:,k)-m_hat)';
    end
S_hat=(1/N)*S_hat;
```

Αρχικά βρίσκω τις συναρτήσεις διάκρισης (g_1_1(x), g_1_2(x)) και τη διαφορά τους (d(x)).

Τα σημεία στα οποία η διαφορά d των 2 συναρτήσεων είναι 0, θα είναι και τα σημεία διάκρισης, όπου οι πιθανότητες να ανήκει το στοιχείο στο ω1 ή το ω2 είναι ίδιες.

Τα αποτελέσματα φαίνονται παρακάτω:

```
3.2.1 starting...

Separation points of w1, w2 using only x1: -4.8438 ,  4.0958
```

3.2.2

```
correct1_1 = 0;
correct1_2 = 0;

for(i=1:10)
    x1 = SAMPLES(i,1);
%     x2 = SAMPLES(i,2);
    if(subs(d_1)>=0)
        correct1_1 = correct1_1 + 1;
    end
    x1 = SAMPLES(i,4);
%     x2 = SAMPLES(i,5);
    if(subs(d_1)<=0)
        correct1_2 = correct1_2 + 1;
    end
end

error1_1    = (10-correct1_1)/10;
error1_2    = (10-correct1_2)/10;

disp(['Error of w1: ',num2str(error1_1)]);
disp(['Error of w2: ',num2str(error1_2)]);
```

Υπολογίζω το ποσοστό των σημείων που δεν ταξινομήθηκαν σωστά. Τα αποτελέσματα είναι:

```
Error of w1: 0.4
Error of w2: 0.3
```

3.2.3

```
disp("3.2.3 starting...");

[m_3_1 , Sigma_3_1] = Gaussian_ML_estimate(SAMPLES(:,1:2)');
syms g_3_1(x);
g_3_1(x)   = g(2, m_3_1, Sigma_3_1, P_w_1);

[m_3_2 , Sigma_3_2] = Gaussian_ML_estimate(SAMPLES(:,4:5)');
syms g_3_2(x);
g_3_2(x)   = g(2, m_3_2, Sigma_3_2, P_w_2);

d_3(x)     = g_3_1(x) - g_3_2(x);

w3_0       = vpa(solve(d_3(x)==0));

disp(['Separation functions of w1, w2 using x1 and x2:']);
fprintf('%s\n', w3_0(1));
fprintf('%s\n', w3_0(2));

correct3_1 = 0;
correct3_2 = 0;

for(i=1:10)
    x1 = SAMPLES(i,1);
    x2 = SAMPLES(i,2);
    if(subs(d_3)>=0)
        correct3_1 = correct3_1 + 1;
    end
```

```
    x1 = SAMPLES(i,4);
    x2 = SAMPLES(i,5);
    if(subs(d_3)<=0)
        correct3_2 = correct3_2 + 1;
    end
end

error3_1   = (10-correct3_1)/10;
error3_2   = (10-correct3_2)/10;

disp(['Error of w1: ',num2str(error3_1)]);
disp(['Error of w2: ',num2str(error3_2)]);
```

Τα αποτελέσματα είναι:

```
3.2.3 starting...

Separation functions of w1, w2 using x1 and x2:

0.39982933971516607930982750522561*x2 -
14.370942479501355360101463844241*(0.00080322688802338831106706812604854*x2^2 -
0.010390708700163045588616069932962*x2 + 0.053211017178063227570119588464343)^(1/2) +
0.7236829743287184259175501896397

0.39982933971516607930982750522561*x2 +
14.370942479501355360101463844241*(0.00080322688802338831106706812604854*x2^2 -
0.010390708700163045588616069932962*x2 + 0.053211017178063227570119588464343)^(1/2) +
0.7236829743287184259175501896397

Error of w1: 0.5
Error of w2: 0.3
```

3.2.4

```
disp("3.2.4 starting...");

[m_4_1 , Sigma_4_1] = Gaussian_ML_estimate(SAMPLES(:,1:3)');
syms g_4_1(x);
g_4_1(x)   = g(3, m_4_1, Sigma_4_1, P_w_1);

[m_4_2 , Sigma_4_2] = Gaussian_ML_estimate(SAMPLES(:,4:6)');
syms g_4_2(x) x;
g_4_2(x)   = g(3, m_4_2, Sigma_4_2, P_w_2);

g_4_3      = 0; %Since p=0;

d_4(x)     = g_4_1(x) - g_4_2(x);

w4_0       = vpa(solve(d_4(x)==0));

disp(['Separation functions of w1, w2 using x1, x2 and x3:']);
fprintf('%s\n', w4_0(1));
fprintf('%s\n', w4_0(2));

correct4_1 = 0;
correct4_2 = 0;

for(i=1:10)
    x1 = SAMPLES(i,1);
    x2 = SAMPLES(i,2);
    x3 = SAMPLES(i,3);
```

```
    if(subs(d_4)>=0)
        correct4_1 = correct4_1 + 1;
    end
    x1 = SAMPLES(i,4);
    x2 = SAMPLES(i,5);
    x3 = SAMPLES(i,6);
    if(subs(d_4)<=0)
        correct4_2 = correct4_2 + 1;
    end
end


error4_1   = (10-correct4_1)/10;
error4_2   = (10-correct4_2)/10;

disp(['Error of w1: ',num2str(error4_1)]);
disp(['Error of w2: ',num2str(error4_2)]);
```

Τα αποτελέσματα είναι:

```
3.2.4 starting...

Separation functions of w1, w2 using x1, x2 and x3:

2.6436462342665080535117166627726*x3 - 0.22439225783518149203544159982281*x2 -
31.747704000860878111205317646307*(0.017498020687503072672966845851416*x3 -
0.0088225455322853068352764076878111*x2 - 0.0041959327631017480446393797284946*x2*x3 +
0.00085538337557245443631977815209313*x2^2 + 0.0089482701284014970379606731110067*x3^2 +
0.0091967210823408329470602545095078)^(1/2) + 2.8768619806235245114846236676642

2.6436462342665080535117166627726*x3 - 0.22439225783518149203544159982281*x2 +
31.747704000860878111205317646307*(0.017498020687503072672966845851416*x3 -
0.0088225455322853068352764076878111*x2 - 0.0041959327631017480446393797284946*x2*x3 +
0.00085538337557245443631977815209313*x2^2 + 0.0089482701284014970379606731110067*x3^2 +
0.0091967210823408329470602545095078)^(1/2) + 2.8768619806235245114846236676642

Error of w1: 0.2
Error of w2: 0.1
```

3.2.5

Τα αποτελέσματα που περιμένουμε είναι τα σφάλματα να μειώνονται όλο και περισσότερο με την προσθήκη επιπλέον χαρακτηριστικών.

Πράγματι, προσθέτοντας το χαρακτηριστικό x3 φαίνεται ότι τα σφάλματα, από 0.5 και 0.3 έγιναν 0.2 και 0.1 αντίστοιχα.

Κάτι τέτοιο όμως δε συνέβη όταν πρόσθεσα το χαρακτηριστικό x2. Αντίθετα το 1^o σφάλμα, από 0.4 έγινε 0.5, ενώ το 2^o παρέμεινε 0.3.

Η πιο πιθανή εξήγηση είναι ότι το x2 χαρακτηριστικό δεν περιέχει κάποια πληροφορία που να βοηθάει στη διάκριση των στοιχείων. Η μικρή αύξηση του σφάλματος που παρατηρήθηκε, ίσως να οφείλεται στο μικρό αριθμό δειγμάτων (10) που έχουμε, ο οποίος συνεπώς μας δίνει μικρή ακρίβεια. Όσο περισσότερο αυξάνονται τα χαρακτηριστικά, τόσο περισσότερο πρέπει να αυξάνονται και τα στοιχεία. Εάν για την καλή εκτίμηση των παραμέτρων μιας μονοδιάστατης pdf χρειάζονται N δείγματα, για την εκτίμηση της pdf σε d διαστάσεις χρειάζονται N^d δείγματα, σύμφωνα με το curse of dimentionality.

3.2.6

```
disp("3.2.6 starting...");

P_w_1   = 0.8;
P_w_2   = 0.1;
P_w_3   = 0.1;

[m_6_1 , Sigma_6_1] = Gaussian_ML_estimate(SAMPLES(:,1:3)');
syms g_6_1(x);
g_6_1(x)  = g(3, m_6_1, Sigma_6_1, P_w_1);

[m_6_2 , Sigma_6_2] = Gaussian_ML_estimate(SAMPLES(:,4:6)');
syms g_6_2(x);
g_6_2(x)  = g(3, m_6_2, Sigma_6_2, P_w_2);

[m_6_3 , Sigma_6_3] = Gaussian_ML_estimate(SAMPLES(:,7:9)');
syms g_6_3(x);
g_6_3(x)  = g(3, m_6_3, Sigma_6_3, P_w_3);

d_6__1_2(x)    = g_6_1(x) - g_6_2(x);
d_6__1_3(x)    = g_6_1(x) - g_6_3(x);
d_6__2_3(x)    = g_6_2(x) - g_6_3(x);

w6_0__1_2      = vpa(solve(d_6__1_2(x)==0));  % Separation Functions for w1, w2
w6_0__1_3      = vpa(solve(d_6__1_3(x)==0));  % Separation Functions for w1, w3
w6_0__2_3      = vpa(solve(d_6__2_3(x)==0));  % Separation Functions for w2, w3

disp(['Separation functions of w1, w2 using x1, x2 and x3:']);
fprintf('%s\n', w6_0__1_2(1));
fprintf('%s\n', w6_0__1_2(2));

disp(['Separation functions of w1, w3 using x1, x2 and x3:']);
fprintf('%s\n', w6_0__1_3(1));
fprintf('%s\n', w6_0__1_3(2));

disp(['Separation functions of w2, w3 using x1, x2 and x3:']);
fprintf('%s\n', w6_0__2_3(1));
fprintf('%s\n', w6_0__2_3(2));
```

Τα αποτελέσματα είναι:

```
3.2.6 starting...

Separation functions of w1, w2 using x1, x2 and x3:

2.6436462342665080535117166627726*x3 - 0.22439225783518149203544159982281*x2 -
31.747704000860878111205317646307*(0.017498020687503072672966845851416*x3 -
0.0088225455322853068352764076878111*x2 - 0.0041959327631017480446393797284946*x2*x3 +
0.00085538337557245443631977815209313*x2^2 + 0.0089482701284014970379606731111067*x3^2 +
0.14019463775835930206961779789335)^(1/2) + 2.8768619806235245114846236676642

2.6436462342665080535117166627726*x3 - 0.22439225783518149203544159982281*x2 +
31.747704000860878111205317646307*(0.017498020687503072672966845851416*x3 -
0.0088225455322853068352764076878111*x2 - 0.0041959327631017480446393797284946*x2*x3 +
0.00085538337557245443631977815209313*x2^2 + 0.0089482701284014970379606731111067*x3^2 +
0.14019463775835930206961779789335)^(1/2) + 2.8768619806235245114846236676642
```

Separation functions of w1, w3 using x1, x2 and x3:

0.78170676515253050659735051274403*x2 + 0.094329933114501721602600936954 6*x3 -
1.6282068132641672822359560984145*(0.47830133076426836396393159648335*x2 +
0.004611286122864790862445340379496 8*x3 + 0.038620446974076774208987827387 84*x2*x3 -
0.065570608038837332086821317837803*x2^2 + 0.014912583974643239683016497939523*x3^2 -
1.0106973003355155511405460737 75)^(1/2) + 3.1080598585691785405765257783648

0.78170676515253050659735051274403*x2 + 0.0943299331145017216026009369546*x3 +
1.6282068132641672822359560984145*(0.47830133076426836396393159648335*x2 +
0.0046112861228647908624453403794968*x3 + 0.03862044697407677420898782738784*x2*x3 -
0.065570608038837332086821317837803*x2^2 + 0.014912583974643239683016497939523*x3^2 -
1.01069730033551555114054607377 5)^(1/2) + 3.1080598585691785405765257783648

Separation functions of w2, w3 using x1, x2 and x3:

0.73262533953481410944082353995084*x2 + 0.21869550305557530261989701935512*x3 -
1.5487765486783121806098430099132*(0.31298204284218172639464635747769*x2 +
0.38633641961727582977815120743272*x3 + 0.053827263136181485810846865998106*x2*x3 -
0.070981427797124074226440479260697*x2^2 + 0.073377994358862903511591649229 91*x3^2 +
1.8102237574599248427539136993921)^(1/2) + 3.0967811263667896654047732159042

0.73262533953481410944082353995084*x2 + 0.21869550305557530261989701935512*x3 +
1.5487765486783121806098430099132*(0.31298204284218172639464635747769*x2 +
0.38633641961727582977815120743272*x3 + 0.053827263136181485810846865998106*x2*x3 -
0.070981427797124074226440479260697*x2^2 + 0.07337799435886290351159164922991*x3^2 +
1.8102237574599248427539136993921)^(1/2) + 3.0967811263667896654047732159042

ΑΣΚΗΣΗ 3.3

3.3.1

```
disp("3.3 starting...");

a        = sym('a','real');
integral = integral(@(u) sin(pi*u),0,1);
A        = double(solve(a*integral==1));

disp('A = ',nut2str(A));
```

Το αποτέλεσμα:

3.3 starting...
A = 1.5708

3.3.2

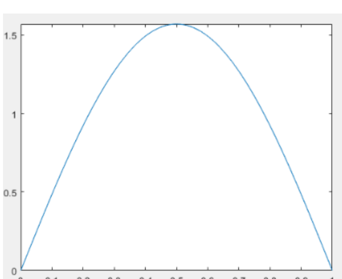
```
x = [1 1 0 1 0 1 1 1 0 1];

syms p0(u) p1(u) p5(u) p10(u);
p0(u)  = p(0, A, x);
p1(u)  = p(1, A, x);
p5(u)  = p(5, A, x);
```

```
x = [1 1 0 1 0 1 1 1 0 1];

syms p0(u) p1(u) p5(u) p10(u);
p0(u)  = p(0, A, x);
p1(u)  = p(1, A, x);
p5(u)  = p(5, A, x);
p10(u) = p(10, A, x);


disp('P(Theta|D0):');
disp(p0);
figure;
fplot(p0,[0 1]);
disp('P(Theta|D1):');
disp(p1);
figure;
fplot(p1,[0 1]);
disp('P(Theta|D5):');
disp(p5);
figure;
fplot(p5,[0 1]);
disp('P(Theta|D10):');
disp(p10);
figure;
fplot(p10,[0 1]);
```

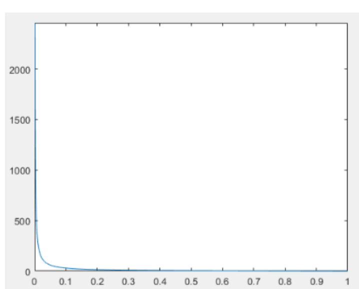
```
function [output] = p(n, A, x)
    syms u;
    if(n == 0)
        output = A*sin(pi*u);
    else
        output = u^n*(1-u)^(10-n)*p(0, A, x)/(int(u^n*(1-u)^(10-n)*p(0, A, x)));
    end
end
```

Τα αποτελέσματα φαίνονται παρακάτω:

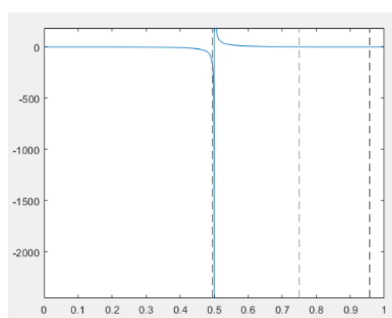
```
P(Theta|D0):
(pi*sin(pi*u))/2
symbolic function inputs: u

P(Theta|D1):
-(u*pi*sin(pi*u))/(2*((u*cos(pi*u))/2 - sin(pi*u)/(2*pi)))
symbolic function inputs: u

(Οι D5 και D10 είναι πολύ μεγάλες και δε τις γράφω στο ριπορτ. Τυπώνονται, ωστόσο, κανονικά τρέχοντας το αρχείο ergasia3.m)
```



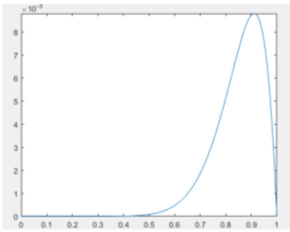
$P(\theta|D^0)$



$P(\theta|D^1)$



$P(\theta|D^5)$



$P(\theta|D^{10})$

3.3.3

ΑΣΚΗΣΗ 3.4

3.4.A

```
disp("3.4 starting...");

P           = [1/3 1/3 1/3];
m           = [[0; 0; 0] [1; 2; 2] [3; 3; 4]];

Sigma(:,:,1) = [0.8 0.2 0.1; 0.2 0.8 0.2; 0.1 0.2 0.8];
Sigma(:,:,2) = [0.8 0.2 0.1; 0.2 0.8 0.2; 0.1 0.2 0.8];
Sigma(:,:,3) = [0.8 0.2 0.1; 0.2 0.8 0.2; 0.1 0.2 0.8];

X           = [mvnrnd(m(:,1), Sigma(:,:,1), 3333); mvnrnd(m(:,2), Sigma(:,:,2), 3333);
mvnrnd(m(:,3), Sigma(:,:,3), 3334)]';
X1          = [mvnrnd(m(:,1), Sigma(:,:,1), 333); mvnrnd(m(:,2), Sigma(:,:,2), 333);
mvnrnd(m(:,3), Sigma(:,:,3), 334)]';
X1_correct  = [ones(1,333) 2*ones(1,333) 3*ones(1,334)];
```

3.4.B

```
euclidean   = euclidean_classifier(m, X1);
mahalanobis = mahalanobis_classifier(m,  Sigma, X1);
bayes       = bayes_classifier(m, Sigma, P, X1);

eucl_corr   = 0;
mah_corr    = 0;
bayes_corr  = 0;
for i=1:1000
    if(euclidean(i)==X1_correct(i))
        eucl_corr = eucl_corr + 1;
    end
    if(mahalanobis(i)==X1_correct(i))
        mah_corr = mah_corr + 1;
    end
    if(bayes(i)==X1_correct(i))
        bayes_corr = bayes_corr + 1;
    end
end
```

```
euclidean_error = (1000-eucl_corr)/1000;
mahalanobis_error = (1000-mah_corr)/1000;
bayes_error = (1000-bayes_corr)/1000;

disp('Euclidean error:');
disp(euclidean_error);
disp('Mahalanobis error:');
disp(mahalanobis_error);
disp('Bayes error:');
disp(bayes_error);
```

```
function [z]=euclidean_classifier(m,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        de(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
    end
    [num,z(i)]=min(de);
end
```

```
function z=mahalanobis_classifier(m,S,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        dm(j)=sqrt((X(:,i)-m(:,j))'*inv(S(:,:,j))*(X(:,i)-m(:,j)));
    end
    [num,z(i)]=min(dm);
end
```

```
function [z]=bayes_classifier(m,S,P,X)
[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        t(j)=P(j)*comp_gauss_dens_val(m(:,j),S(:,:,j),X(:,i));
    end
    [num,z(i)]=max(t);
end
```

```
function [z]=comp_gauss_dens_val(m,S,x)
[l,c]=size(m);
z=(1/( (2*pi)^(l/2)*det(S)^0.5) )*exp(-0.5*(x-m)'*inv(S)*(x-m));
```

Το αποτέλεσμα:

```
3.4 starting...
Euclidean error:
   0.1070

Mahalanobis error:
   0.1090

Bayes error:
```

```
0.1090
```

Παρατηρώ ότι η πιθανότητα λάθους των mahalanobis και Bayes είναι ίσες μεταξύ τους και μικρότερες από την Eucllidean.

3.4.Γ

```
disp("3.4.C starting...");

[m_hat(:,1), Sigma_hat(:,:,1)] = Gaussian_ML_estimate(X(:,1:3333));
[m_hat(:,2), Sigma_hat(:,:,2)] = Gaussian_ML_estimate(X(:,3334:6666));
[m_hat(:,3), Sigma_hat(:,:,3)] = Gaussian_ML_estimate(X(:,6667:10000));

euclidean   = euclidean_classifier(m_hat, X1);
mahalanobis = mahalanobis_classifier(m_hat,  Sigma_hat, X1);
bayes       = bayes_classifier(m_hat, Sigma_hat, P, X1);

eucl_corr   = 0;
mah_corr    = 0;
bayes_corr  = 0;

for i=1:1000
    if(euclidean(i)==X1_correct(i))
        eucl_corr = eucl_corr + 1;
    end
    if(mahalanobis(i)==X1_correct(i))
        mah_corr = mah_corr + 1;
    end
    if(bayes(i)==X1_correct(i))
        bayes_corr = bayes_corr + 1;
    end
end

euclidean_error = (1000-eucl_corr)/1000;
mahalanobis_error = (1000-mah_corr)/1000;
bayes_error = (1000-bayes_corr)/1000;

disp('Euclidean error:');
disp(euclidean_error);
disp('Mahalanobis error:');
disp(mahalanobis_error);
disp('Bayes error:');
disp(bayes_error);
```

Το αποτέλεσμα:

```
3.4.C starting...
Euclidean error:
  0.1060

Mahalanobis error:
  0.1010

Bayes error:
  0.1000
```

Επιλέγω τον ταξινομητή Bayes που φαίνεται να έχει το μικρότερο σφάλμα.

3.4.Δ

```
disp("3.4.D starting...");

P           = [1/6 1/6 2/3];
m           = [[0; 0; 0] [1; 2; 2] [3; 3; 4]];

Sigma(:,:,1) = [0.8 0.2 0.1; 0.2 0.8 0.2; 0.1 0.2 0.8];
Sigma(:,:,2) = [0.6 0.2 0.01; 0.2 0.8 0.01; 0.01 0.01 0.6];
Sigma(:,:,3) = [0.6 0.1 0.1; 0.1 0.6 0.1; 0.1 0.1 0.6];

X           = [mvnrnd(m(:,1), Sigma(:,:,1), 3333); mvnrnd(m(:,2), Sigma(:,:,2), 3333);
mvnrnd(m(:,3), Sigma(:,:,3), 3334)]';
X1          = [mvnrnd(m(:,1), Sigma(:,:,1), 333); mvnrnd(m(:,2), Sigma(:,:,2), 333);
mvnrnd(m(:,3), Sigma(:,:,3), 334)]';
X1_correct  = [ones(1,333) 2*ones(1,333) 3*ones(1,334)];

euclidean   = euclidean_classifier(m, X1);
mahalanobis = mahalanobis_classifier(m,  Sigma, X1);
bayes       = bayes_classifier(m, Sigma, P, X1);

eucl_corr   = 0;
mah_corr    = 0;
bayes_corr  = 0;
for i=1:1000
    if(euclidean(i)==X1_correct(i))
        eucl_corr = eucl_corr + 1;
    end
    if(mahalanobis(i)==X1_correct(i))
        mah_corr = mah_corr + 1;
    end
    if(bayes(i)==X1_correct(i))
        bayes_corr = bayes_corr + 1;
    end
end

euclidean_error = (1000-eucl_corr)/1000;
mahalanobis_error = (1000-mah_corr)/1000;
bayes_error = (1000-bayes_corr)/1000;

disp('Euclidean error:');
disp(euclidean_error);
disp('Mahalanobis error:');
disp(mahalanobis_error);
disp('Bayes error:');
disp(bayes_error);
```

Το αποτέλεσμα:

```
Euclidean error:
  0.0610

Mahalanobis error:
  0.0620

Bayes error:
  0.0600
```

Τα σφάλματα φαίνεται να είναι πολύ μικρότερα από τα προηγούμενα. Επίσης, πλέον ο ταξινομητής Bayes φαίνεται να έχει την καλύτερη απόδοση.