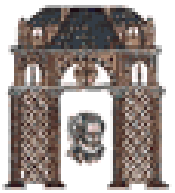


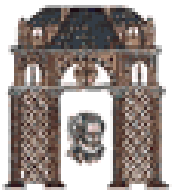
Αναγνώριση Προτύπων

Μη παραμετρικές τεχνικές – Αριθμητικά Παραδείγματα

(Non Parametric Techniques)

Καθηγητής Χριστόδουλος Χαμζάς

Τα περιεχόμενο της παρουσίασης βασίζεται στο βιβλίο: "Introduction to Pattern Recognition A Matlab Approach", S. Theodoridis, A. Pikrakis, K. Koutroubas, D. Caboyras, Academic Press, 2010



ΠΑΡΑΘΥΡΑ PARZEN

Για να υπολογίσουμε την pdf pn(x) στο σημείο **x** προσθέτουμε το

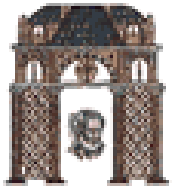
$$\varphi\left(\frac{\mathbf{x}-\mathbf{x_n}}{h_n}\right)$$

για όλα τα N σημεία $\mathbf{x_n}$ και το κανονικοποιούμε διαιρώντας με το N V_N δηλαδή

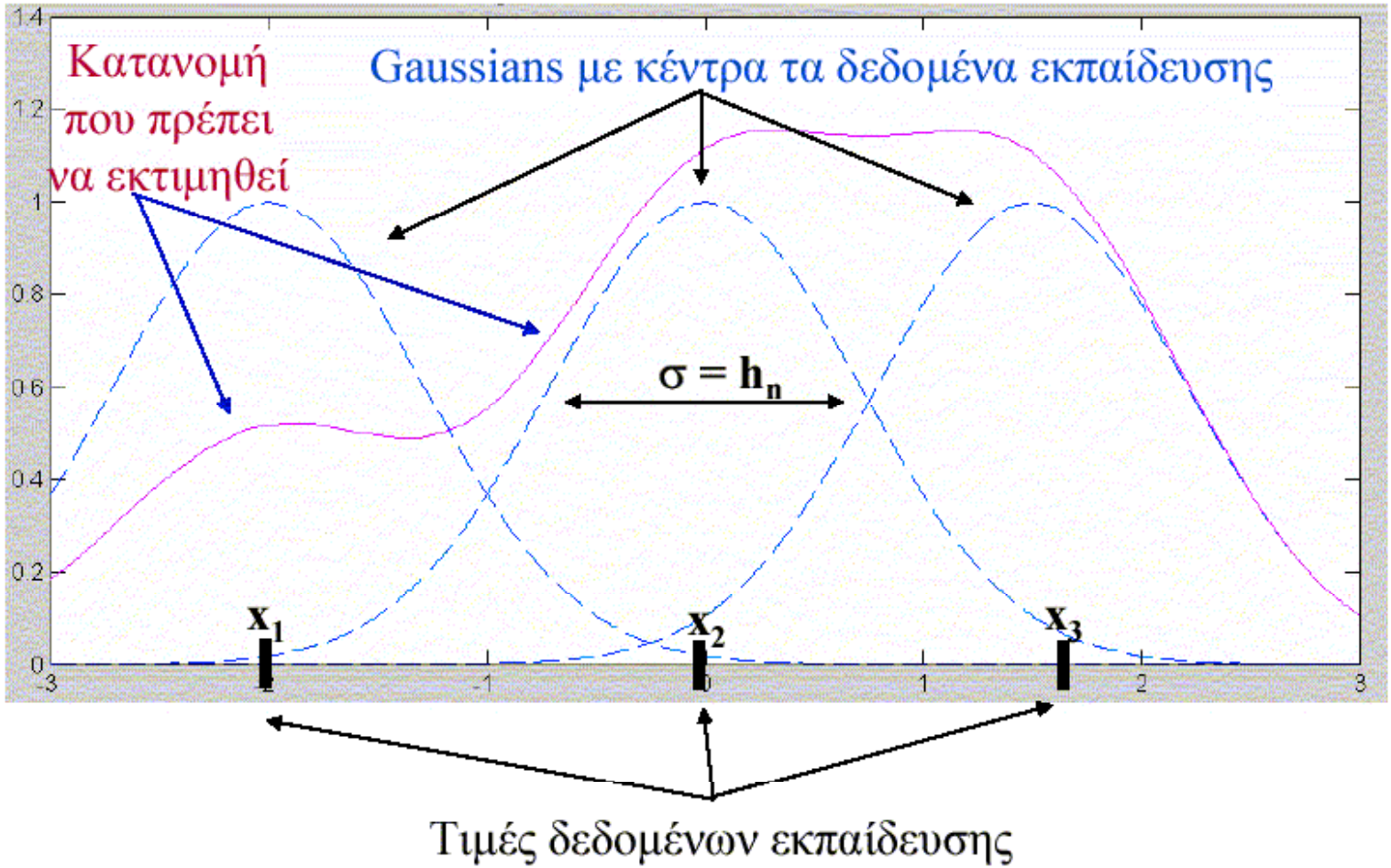
$$p_N(x)=\frac{1}{NV_N}\sum_{i=1}^{N}\varphi\left(\frac{\mathbf{x}-\mathbf{x_i}}{h_N}\right)$$

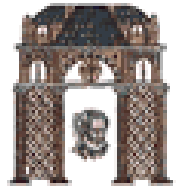
φ(.) είναι η συνάρτηση kernel και συνήθως είναι η πολυδιάστατη κανονική με $V_N = (h_N)^d$, δηλαδή,

$$p_N(x)=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}h_N^d}\exp\left(-\frac{(\mathbf{x}-\mathbf{x_i})^T(\mathbf{x}-\mathbf{x_i})}{2h_N^2}\right)$$



Παράθυρα Parzen





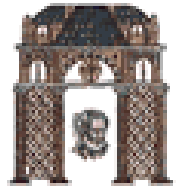
Parzen Παράθυρα

Example 1.7.1. Generate $N = 1000$ data points lying in the real axis, $x_i \in R$, $i = 1, 2, \ldots, N$, from the following pdf, and plot $p(x)$:

$$p(x) = \frac{1}{3}\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{x^2}{2\sigma_1^2}\right) + \frac{2}{3}\frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{(x-2)^2}{2\sigma_2^2}\right)$$

where $\sigma_1 = \sigma_2 = 0.2$.

Use the Parzen windows approximation with normal kernel, with $h = 0.1$, and plot the obtained estimate.

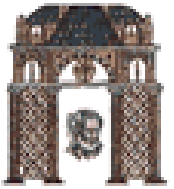


Solution. The pdf is actually a Gaussian mixture model. Use the function *generate_gauss_classes* to generate the required data set, typing

```
m=[0; 2]';
S(:,:,1)=[0.2];

S(:,:,2)=[0.2];
P=[1/3 2/3];
N=1000;
randn('seed',0);
[X]=generate_gauss_classes(m,S,P,N);
```

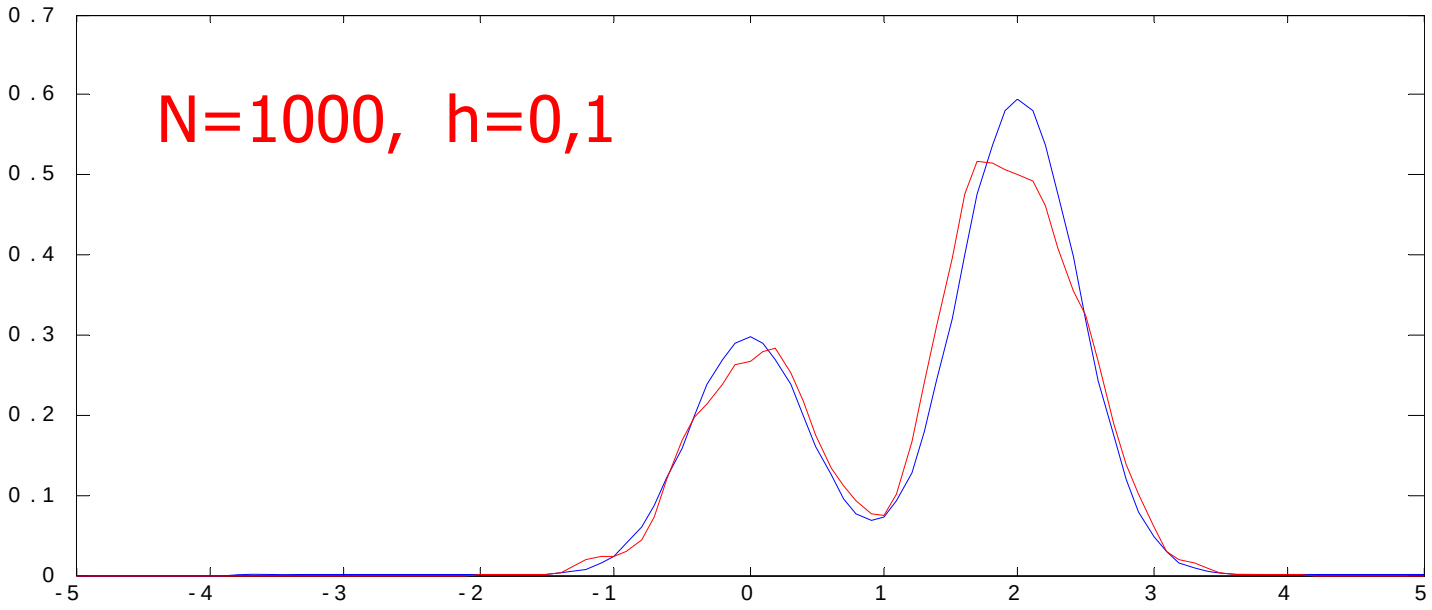
Step 1. To plot the pdf, assume $x \in [-5, 5]$ and type

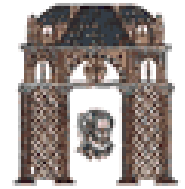
```
x=-5:0.1:5;
pdfx=(1/3)*(1/sqrt(2*pi*0.2))*exp(-(x.^2)/0.4)
     +(2/3)*(1/sqrt(2*pi*0.2))*exp(-((x-2).^2)/0.4);
plot(x,pdfx); hold;
```

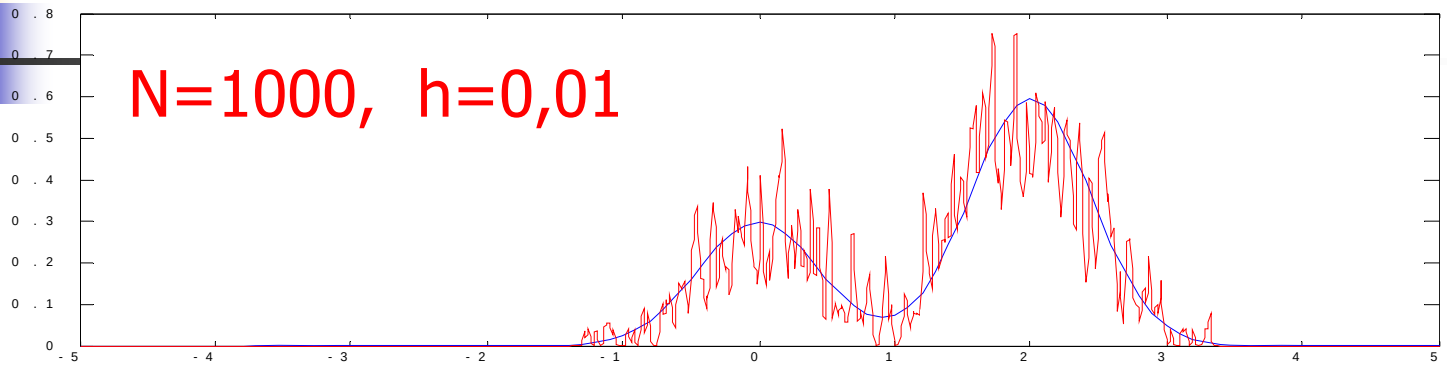


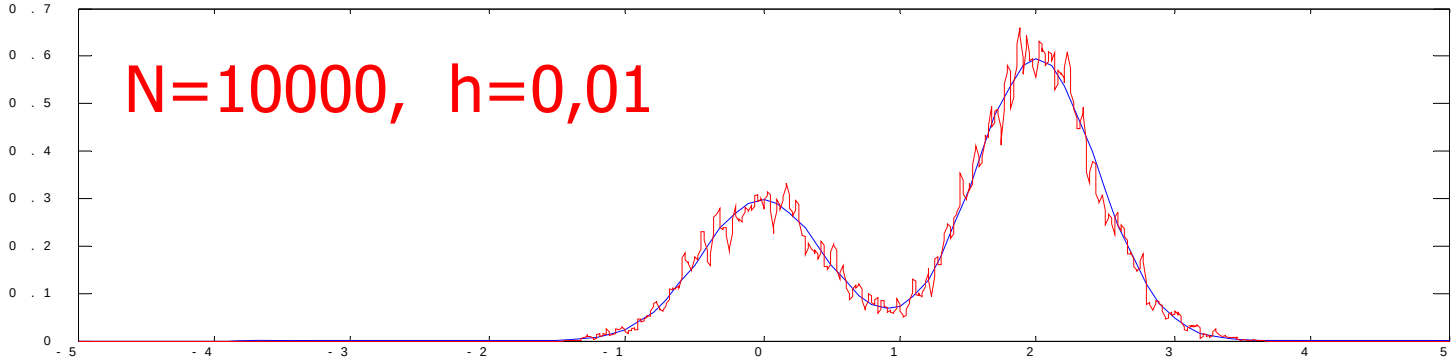
Step 2. To compute and plot the approximation of the pdf for $h = 0.1$ and $x \in [-5, 5]$, use function *Parzen_gauss_kernel* as follows:

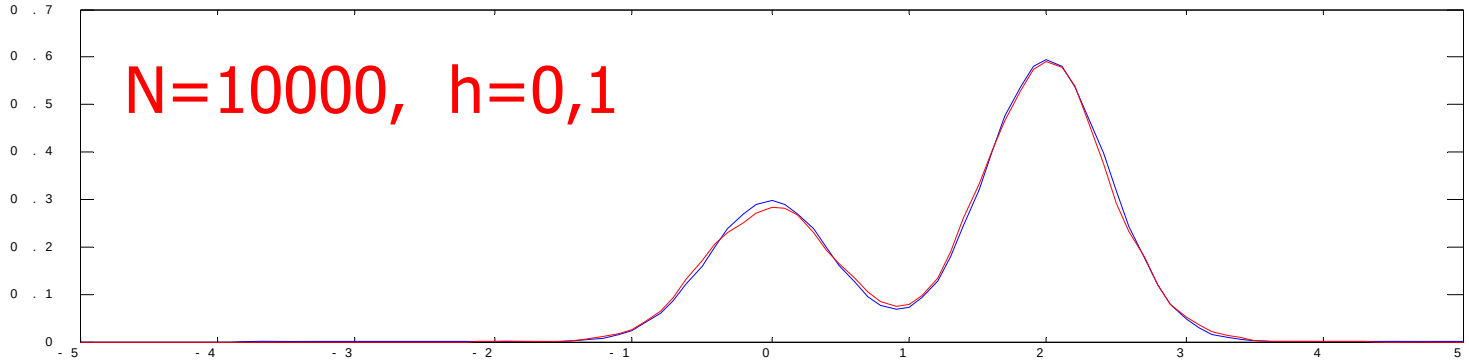
```
h=0.1;
pdfx_approx=Parzen_gauss_kernel(X,h,-5,5);
plot(-5:h:5,pdfx_approx,'r');
```






N=1000, h=0,01


N=10000, h=0,01


N=10000, h=0,1

Πιθανοτικά Νευρωνικά Δίκτυα
Probabilistic Neural Networks (PNN)

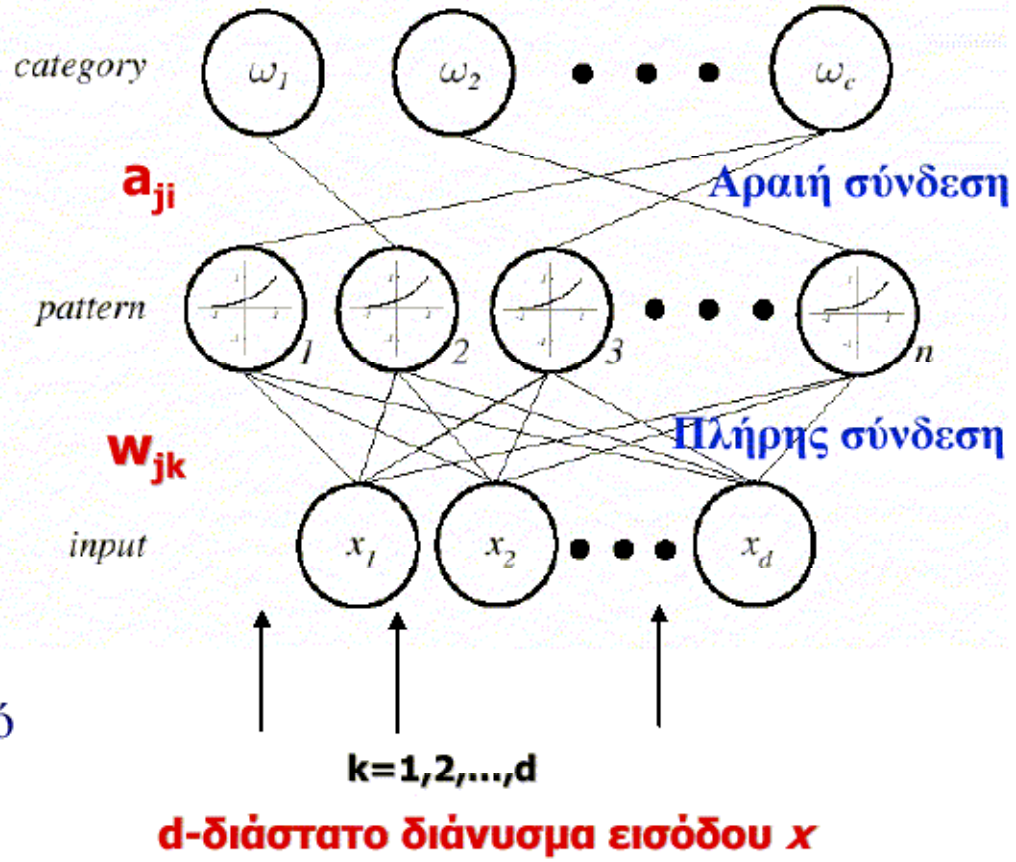
- Είσοδος: $\{x_k;\ k=1,\ldots,d\}$ d κόμβοι, καθένας αντιστοιχεί σε ένα χαρακτηριστικό.

- w_{jk}: βάρη που συνδέουν την k–στή είσοδο με τον j–στό κόμβο κρυφού επιπέδου (κόμβο προτύπου).

- Κρυφό επίπεδο: n κόμβοι, καθένας αντιστοιχεί σε ένα πρότυπο, δηλαδή δείγμα εκπαίδευσης, $j=1,2,\ldots,n$.

- Επίπεδο εξόδου: c κόμβοι, καθένας παριστά μια κλάση.

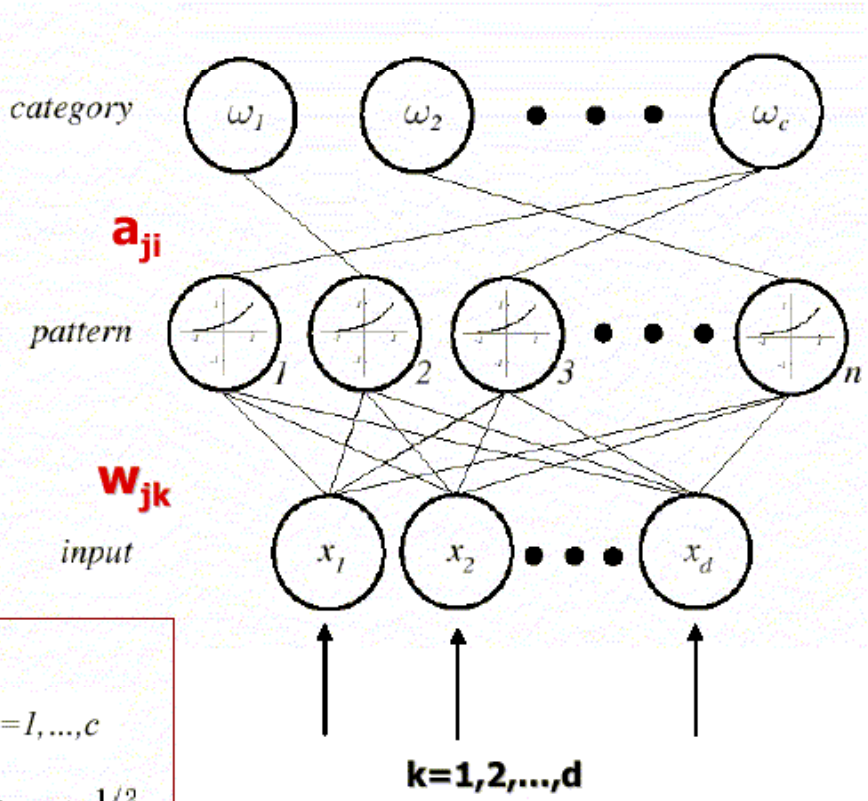
- a_{ji}: βάρη που συνδέουν j–στό κρυφό κόμβο με τον i–στό κόμβο εξόδου, $i=1,2,\ldots,c$



PNN - Εκπαίδευση

Εκπαίδευση

- Το *j-στό* δείγμα εκπαίδευσης (πρότυπο) κανονικοποιείται να έχει μέτρο μονάδα.
- Τοποθετείται στους κόμβους εισόδου.
- Τα βάρη w_{jk} ορίζονται ως $w_{jk}=x_{jk}$.
- Μία μοναδική σύνδεση με βάρος $a_{ji}=1$ γίνεται από τον πρώτο κρυφό κόμβο σε εκείνο τον κόμβο του επιπέδου εξόδου που αντιστοιχεί στην (γνωστή) κλάση του $\mathbf{x}_j$.



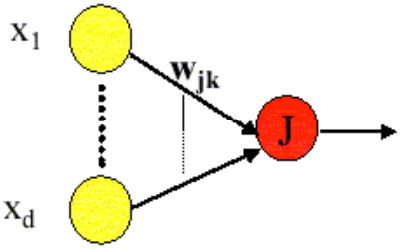
d-διάστατο διάνυσμα εισόδου ***x***

Algorithm 1 (PNN training)

1 **begin initialize** $j=0,\ a_{ji}=0$ for $j=1,...,n;\ i=1,...,c$
2 **do** $j \leftarrow j+1$
3 normalize : $x_{jk} \leftarrow x_{jk} / \left(\sum_{i}^{d} x_{ji}^2\right)^{1/2}$
4 train : $w_{jk} \leftarrow x_{jk}$
5 **if** $\mathbf{x} \in \omega_i$ **then** $a_{ji} \leftarrow 1$
6 **until** $j = n$

PNN – Ταξινόμηση

- Κάθε κόμβος προτύπου δημιουργεί το εσωτερικό γινόμενο του διανύσματος βαρών του και της κανονικοποιημένης εισόδου ***x*** για να υπολογίσει το $net_J = \mathbf{w}^t\mathbf{x}$, και να αποδώσει $e^{[(netJ-1)/\sigma 2]}$.



$$e^{\left((net_J - 1)/\sigma^2\right)}$$

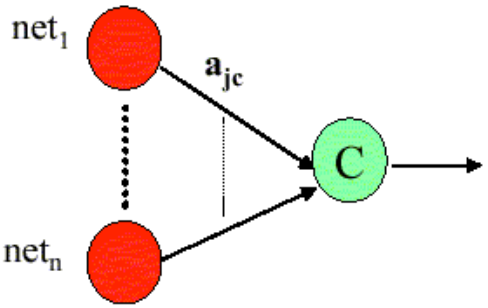
$$\varphi\left(\frac{\mathbf{x} - \mathbf{w}_k}{h_n}\right) \propto \overbrace{e^{-(\mathbf{x}-\mathbf{w}_k)^t(\mathbf{x}-\mathbf{w}_k)/2\sigma^2}}^{\text{desired Gaussian}}$$

$$= e^{-(\mathbf{x}^t\mathbf{x} + \mathbf{w}_k^t\mathbf{w}_k - 2\mathbf{x}^t\mathbf{w}_k)/2\sigma^2} = \underbrace{e^{(z_k-1)/\sigma^2}}$$

Πλάτος Γκαουσσιανού παραθύρου Parzen

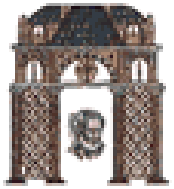
Συνάρτηση ενεργοποίησης

- Κάθε κόμβος κλάσης αθροίζει τα αποτελέσματα των κόμβων προτύπων που συνδέονται με αυτόν. Αυτό εξασφαλίζει ότι η ενεργοποίηση κάθε κλάσης παριστά την εκτίμηση σ.π.π. με κυκλικά συμμετρικό Gaussian παράθυρο Parzen με πίνακα συνδιασποράς $\sigma^2 \mathbf{I}_{d\times d}$, όπου ***I*** είναι ο μοναδιαίος πίνακας.



Algorithm 2 (PNN classification)

1 **begin initialize** $k = 0, \mathbf{x} =$ test pattern
2 **do** $k \leftarrow k + 1$
3 $z_k \leftarrow \mathbf{w}_k^t\mathbf{x}$
4 **if** $a_{ki} = 1$ **then** $g_i \leftarrow g_i + \exp[(z_k - 1)/\sigma^2]$
5 **until** $k = n$
6 **return** $class \leftarrow \arg\max_i g_i(\mathbf{x})$
7 **end**



Παράδειγμα ParzenPNN

function net = parzenPNNlearn(samples,classification,center)

samples=$[x_1\ x_2\ x_3\ .....\ x_N]^T$ Matrix Nxd with N samples with d components each.

Classification = $[\omega_1\omega_3\omega_3\omega_1....\omega_2]$ Classification of the N samples in C classes

Center=true

Example: N=5, d=2, c=2

$$\text{Samples} = \begin{bmatrix} 0 & 0{,}1 \\ 1.0 & 0{,}9 \\ 0{,}8 & 0{,}6 \\ 0{,}7 & 0{,}8 \\ 0{,}3 & 0{,}2 \end{bmatrix} \quad \text{classification} = [\text{abbba}]$$

Net : the probabilistic Parzen Neural Net

function [class,score,scores] = parzenPNNclassify(net,X,nonlin)

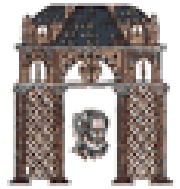
Net a valid parzenPNN

X sample to be classified with d componets

Nonlin = σ (default 2)

Example:

X=[0,3 0,1]

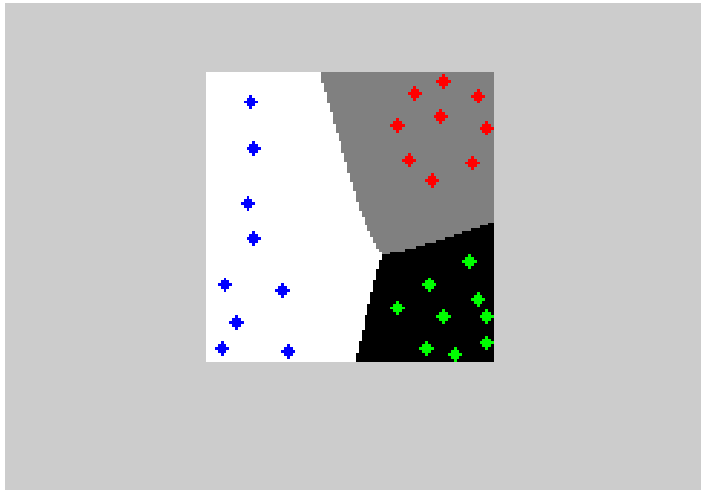


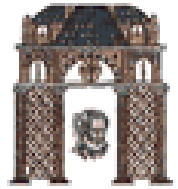
Εφαρμογή 1

```
close('all'); clear;
% To generate X, utilize the function generate_gauss_classes
m=[0 0 ; 0  3 ;  3 4]';
S1=0.8*eye(2);
S(:,:,1)=S1;S(:,:,2)=S1;S(:,:,3)=S1;
P=[1/3 1/3 1/3]';
N=10;
randn('seed',0);
[X,y]=generate_gauss_classes(m,S,P,N);
net=parzenPNNlearn(X,y);
class=parzenPNNclassify(net,[1 2]')
ans=1
[class score scores]=parzenPNNclassify(net, X)

y =     [1    1    1    2    2    2    3    3    3]
class = [1    1    1    2    1    2    3    3    3]
Score =[3.5497   4.5272   3.1704   3.0247   2.5215   2.7730   4.7555   3.5498   3.7502]

          [3.5497   2.6028   1.6534]
          [4.5272   2.8883   1.3262]
Scores = [ .....................................
          [1.4972   2.1230   3.5498]
          [1.6214   1.8528   3.7502]
```

Εφαρμογή 2

Εάν τρέξουμε το parzenPNN_demo, μας επιτρέπει να βάλουμε 3 κλασεις σημείων 2 διαστάσεων, μετά μας επιτρέπει να προσθέσουμε και νέα δείγματα εκπαίδευσης και στο τέλος διαχωρίζει το επίπεδο στις περιοχές των κλάσεων





Εκτίμηση k_n-πλησιέστερων γειτόνων
k_n-Nearest neighbour (KNN)

Το πρόβλημα: Έχουμε N δεδομένα $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$ από μία άγνωστη pdf και θέλουμε να εκτιμήσουμε την τιμή p(**x**) της άγνωστη pdf για ένα **x**.

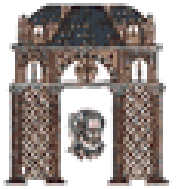
Ο Αλγόριθμος

1. Επιλέγουμε μία τιμή για το k καθώς και την συνάρτηση μέτρου που θα χρησιμοποιήσουμε για την απόσταση (Ευκλείδεια, Mahalanobis, Manhattan, ..)
2. Βρίσκουμε την απόσταση του **x** από όλα τα δεδομένα
3. Βρίσκουμε τα πλησιέστερα k δεδομένα στο **x**
4. Υπολογίζουμε τον όγκο V(x) που περικλείει τα k σημεία
5. Προσεγγίζουμε την pdf στο χ με $p(\mathbf{x}) \approx \frac{k}{NV(\mathbf{x})}$

Εάν έχουμε επιλέξει την Ευκλείδεια απόσταση στο d-διαστατο χώρο και η απόσταση από το μακρινότερο δείγμα είναι ρ τότε

V(x) = 2ρ για d=1, V(x) = πρ^2 για d=2, V(x) = (4/3)πρ^2 για d=3

ενώ για απόσταση Mahalanobis έχουμε υπερελλειψοειδή.



k πλησιέστεροι γείτονες

Example 1.8.1. Generate $N = 1000$ data points lying in the real axis, $x_i \in R$, $i = 1, 2, \ldots, N$, from the following pdf, and plot $p(x)$:

$$p(x) = \frac{1}{3}\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{x^2}{2\sigma_1^2}\right) + \frac{2}{3}\frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{(x-2)^2}{2\sigma_2^2}\right)$$

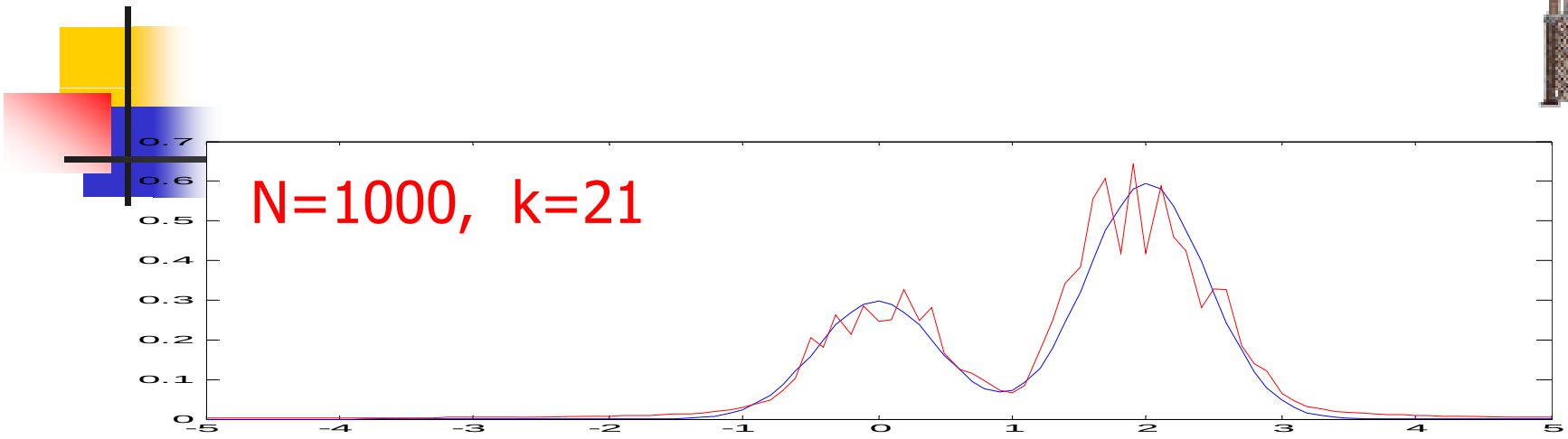
where $\sigma_1 = \sigma_2 = 0.2$.

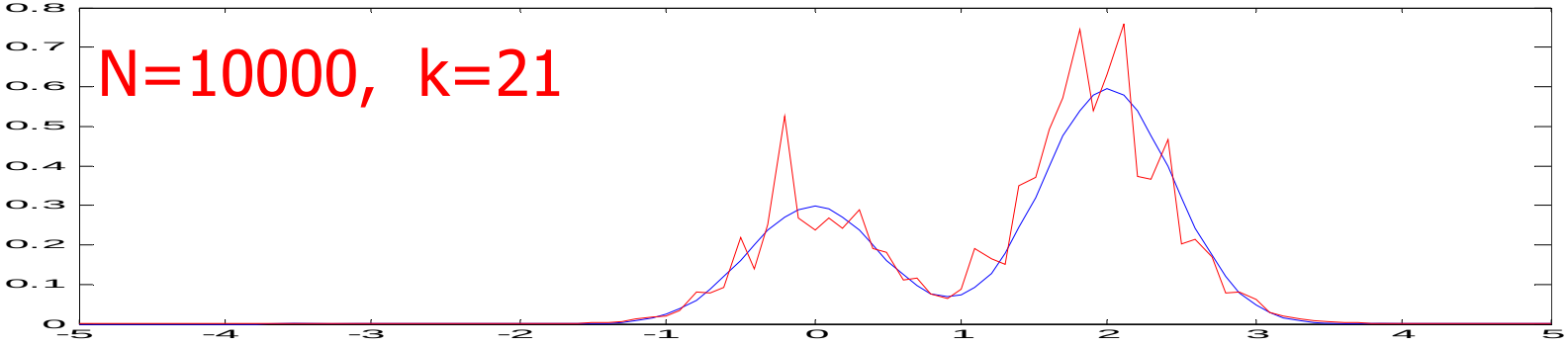
Use the k nearest neighborhood with k = 21, and plot the obtained estimate.

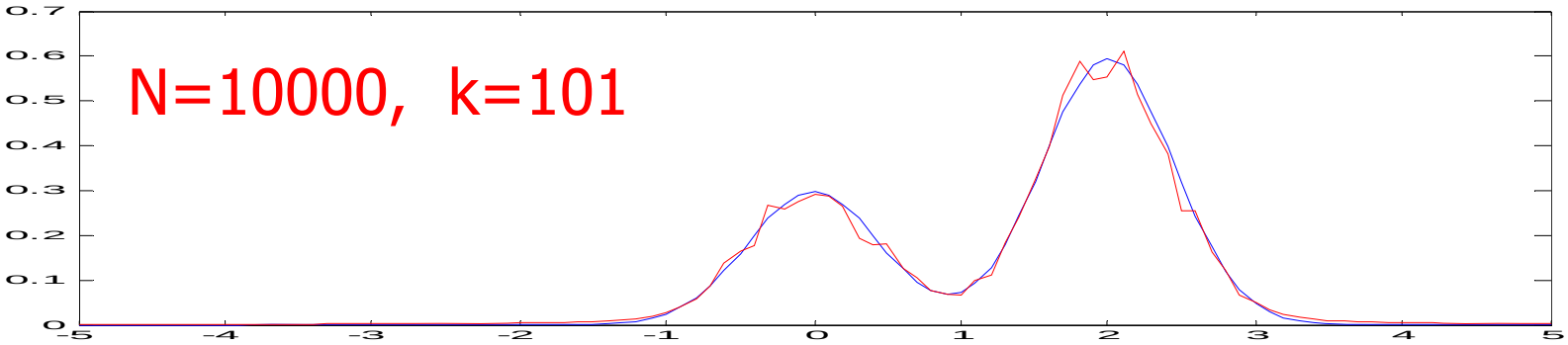
Solution. To generate the set X of the data vectors, work as in Example 1.7.1. Assuming that we are interested in approximating the pdf for $x \in [-5, 5]$ (as in Example 1.7.1), we use the function *knn_density_estimate*, typing

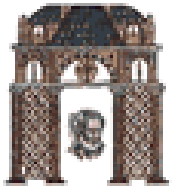
```
pdfx_approx=knn_density_estimate(X,21,-5,5,0.1);
plot(-5:0.1:5,pdfx_approx,'r');
```

■


N=1000, k=21

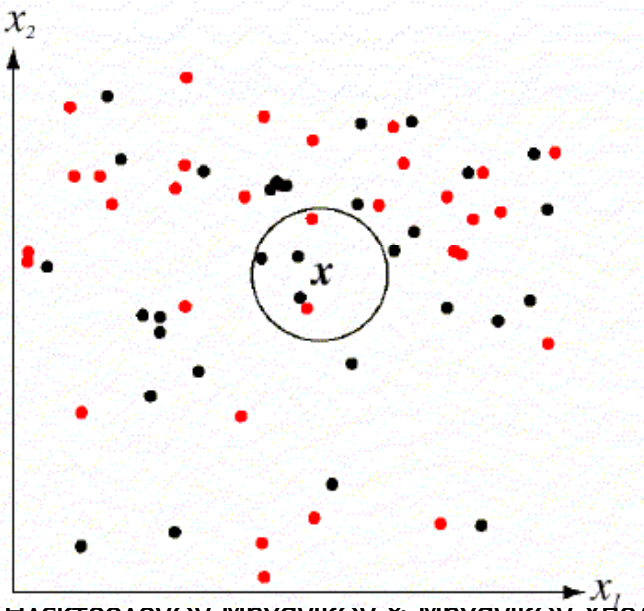

N=10000, k=21

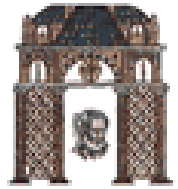

N=10000, k=101



Ταξινομητής KNN

- Για την ταξινόμηση ενός δεδομένου δείγματος $\boldsymbol{x}$,
  - Μεταξύ των n διανυσμάτων εκπαίδευσης, προσδιορίζουμε τους k πλησιέστερους γείτονες του ανεξάρτητα από την κλάση στην οποία ανήκουν, (όπου το k είναι περιττός για ταξινόμηση σε μία από δύο κλάσεις).
    Εάν έχουμε **c** κλάσεις τότε το **k** δεν πρέπει να είναι πολλαπλάσιο του **c.**
  - Προσδιορίζουμε πόσα από δείγματα (έστω k_i) ανήκουν στην τάξη i, $\sum_i k_i = k$
  - Ταξινομούμε το $\boldsymbol{x}$ στην κλάση με το μεγαλύτερο πλήθος k_i δειγμάτων!





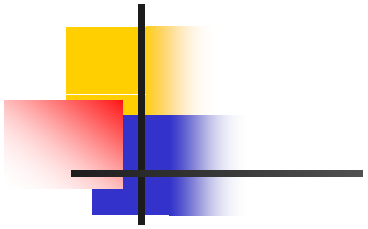
Παράδειγμα knn ταξινομητή

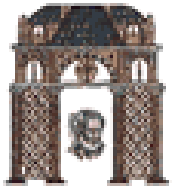
Example 1.10.1

1. Consider a 2-dimensional classification problem where the data vectors stem from two equiprobable classes, ω_1 and ω_2. The classes are modeled by Gaussian distributions with means $m_1 = [0,0]^T$, $m_2 = [1,2]^T$, and respective covariance matrices

$$S_1 = S_2 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

   Generate two data sets X_1 and X_2 consisting of 1000 and 5000 points, respectively.
2. Taking X_1 as the training set, classify the points in X_2 using the k-NN classifier, with $k = 3$ and adopting the squared Euclidean distance. Compute the classification error.





Solution

Step 1. To generate sets X_1 and X_2, type

```
m=[0 0; 1 2]';
S=[0.8 0.2;0.2 0.8];
S(:,:,1)=S;S(:,:,2)=S;
P=[1/2 1/2]'; N_1=1000;
randn('seed',0)
[X1,y1]=generate_gauss_classes(m,S,P,N_1);
N_2=5000;
randn('seed',100)
[X2,y2]=generate_gauss_classes(m,S,P,N_2);
```

Step 2. For the classification task, use function *k_nn_classifier* and type

```
k=3;
z=k_nn_classifier(X1,y1,k,X2);
```

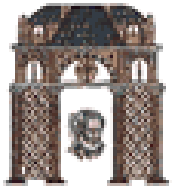
To compute the classification error, type

```
pr_err=sum(z~=y2)/length(y2)
```

Compute the optimum Classification error with a Bayes Classifier

z1=bayes_classifier(m,S,P,X2);
pr_err_Bayes=sum(z1~=y2)/length(y2)

Η πιθανότητα λάθους που παίρνουμε είναι **15,12%** ενώ το βέλτιστο σφάλμα για ένα Bayes ταξινομητή είναι **12,44%**



Μείωση Διαστάσεων

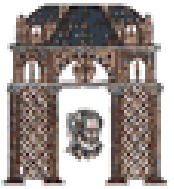
Μία μέθοδος για να μειώσουμε την υπολογιστική πολυπλοκότητα είναι να μειώσουμε τον αριθμό N των χαρακτηριστικών των δειγμάτων X

1. Μπορούμε να το πετύχουμε παραλείποντας στους υπολογισμούς μερικά χαρακτηριστικά
2. Είναι καλύτερο να δημιουργήσουμε νέα δείγματα **Y=AX**
   με τον γραμμικό συνδυασμό ώστε κάποια χαρακτηριστικά των Y να μην είναι πλέον σημαντικά για την αναπαράστασή τους.

Παράδειγμα: $X_1=(1, 5)$, $X_2=(2, 8)$, $X_3=(0, 2)$, $X_4=(-1, -1)$, $X_5=(0.2 , 2.6)$,

$$\mathbf{Y} = \mathbf{A}^T\mathbf{X} = \begin{bmatrix} 0.316 & 0.949 \\ -0.949 & 0.316 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 & -1 & 0.2 \\ 5 & 8 & 2 & -1 & 2.6 \end{bmatrix} = \begin{bmatrix} 5.06 & 8.22 & 1.90 & 1.26 & 2.53 \\ 0.63 & 0.63 & 0.63 & 0.63 & 0.63 \end{bmatrix}$$

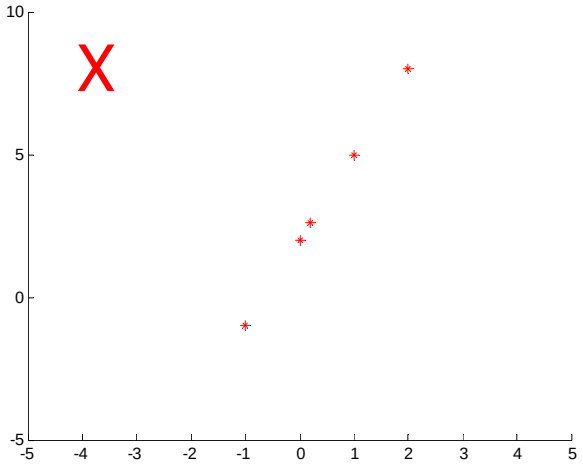
$\lambda_1 = 12.68 \quad \lambda_2 = 0.0$

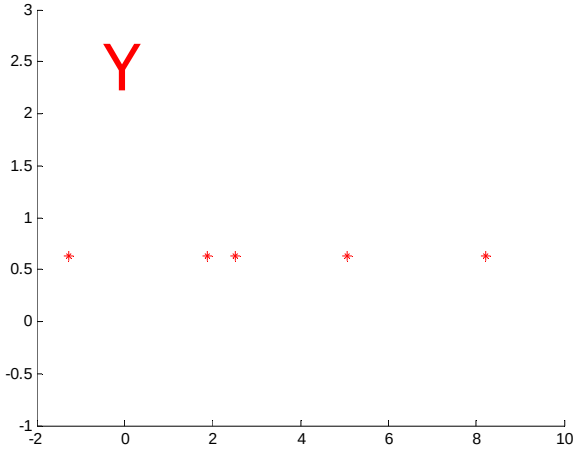


```
X=[1,5;2,8;0,2;-1,-1;0.2,2.6]';
[eigenval,eigenvec,explain,Y,mean_vec]=pca_fun(X,2);
figure(1), hold on;
axis([-5 5 -5 10]);
plot(X(1,:),X(2,:),'r*');
Y=eigenvec'*X;
figure(2), hold on;
axis([-2 10 -1 3]);
plot(Y(1,:),Y(2,:),'r*');
```

Άρα χρειαζόμαστε μόνο ένα χαρακτηριστικό το

$$\mathbf{y_1 = 0.316x_1 + 0.949x_2}$$

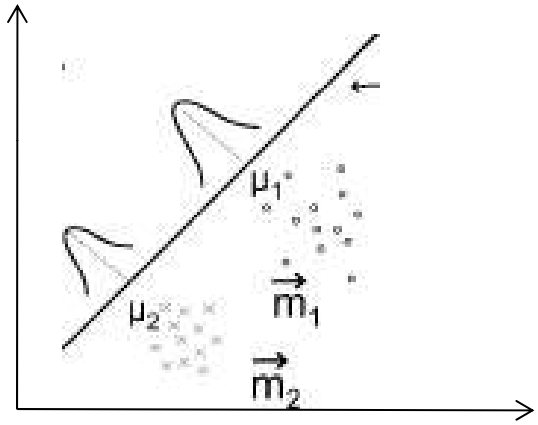




ΜΕΙΩΣΗ ΔΙΑΣΤΑΣΕΩΝ: PCA

Principal Component Analysis

- Έστω **x** σε D-διαστάσεις. Θέλουμε κάποιο μετασχηματισμό **A** να μας δώσει ένα **y=Ax** σε N-διαστάσεις, (N<D), ώστε να μπορούμε να αναπαραστήσουμε ικανοποιητικά το **y** σε λιγότερες διαστάσεις



Principal component analysis (PCA) is one of the most popular techniques for dimensionality reduction. Starting from an original set of l samples (features), which form the elements of a vector $x \in \mathcal{R}^l$, the goal is to apply a linear transformation to obtain a new set of samples:

$$y = A^T x$$

so that the components of y are uncorrelated. In a second stage, one chooses the most significant of these components. The steps are summarized here:

1. Estimate the covariance matrix S. Usually the mean value is assumed to be zero, $E[x] = 0$. In this case, the covariance and autocorrelation matrices coincide, $R = E[xx^T] = S$. If this is not the case, we subtract the mean. Recall that, given N feature vectors, $x_i \in \mathcal{R}^l$, $i = 1, 2, \ldots, N$, the autocorrelation matrix estimate is given by

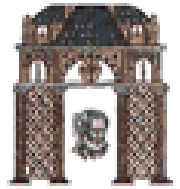
$$R \approx \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T \tag{3.1}$$

2. Perform the eigendecomposition of S and compute the l eigenvalues/eigenvectors, λ_i, $a_i \in \mathcal{R}^l$, $i = 0, 2, \ldots, l-1$.

3. Arrange the eigenvalues in descending order, $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{l-1}$.
4. Choose the m largest eigenvalues. Usually m is chosen so that the gap between λ_{m-1} and λ_m is large. Eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{m-1}$ are known as the *m principal components*.
5. Use the respective (column) eigenvectors a_i, $i = 0, 1, 2, \ldots, m-1$ to form the transformation matrix

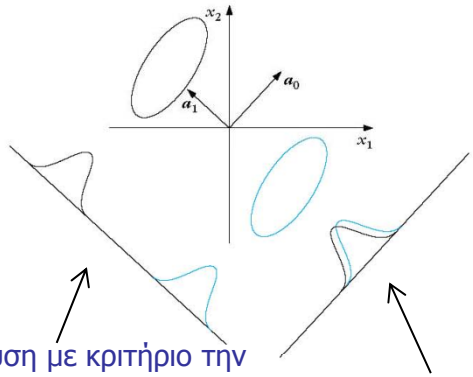
$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{m-1} \end{bmatrix}$$

6. Transform each l-dimensional vector x in the original space to an m-dimensional vector y via the transformation $y = A^T x$. In other words, the ith element $y(i)$ of y is the *projection* of x on a_i $\left(y(i) = a_i^T x\right)$.



ΜΕΙΩΣΗ ΔΙΑΣΤΑΣΕΩΝ: Τεχνική Fisher

- Έστω **y** σε D-διαστάσεις. Θέλουμε κάποιο μετασχηματισμό **w** να μας δώσει ένα **x=wy** σε N-διαστάσεις, ώστε να κάνουμε την κατηγοριοποίηση σε λιγότερες διαστάσεις



Ανάλυση με κριτήριο την μέγιστη διαχωριστικότητα (Fisher Eye)

Ανάλυση σε κύριες συνιστώσες (PCA)

Παράδειγμα D=2 και N=1

- Θέλουμε να διαλέξουμε το μετασχηματισμό w ώστε:
  1. Σε κάθε κατηγορία να έχουμε τα γεγονότα κοντά μεταξύ τους, και
  2. οι κατηγορίες να απέχουν το μέγιστο μεταξύ τους.
- Δηλαδή, θα θέλαμε να ισχύουν:
  1. οι μέσες τιμές να απέχουν το μέγιστο μεταξύ τους και
  2. οι διασπορές των δύο κατηγοριών να είναι μηδέν.

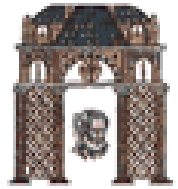
Τα παραπάνω μεταφράζονται ως εξής

$$\left|\mu_1 - \mu_2\right|^2 = \max$$

$$\sigma_1^2 + \sigma_2^2 = \min$$

Και ο Fisher πρότεινε την

$$J(w) = \frac{\left|\mu_1 - \mu_2\right|^2}{\sigma_1^2 + \sigma_2^2} = \max$$

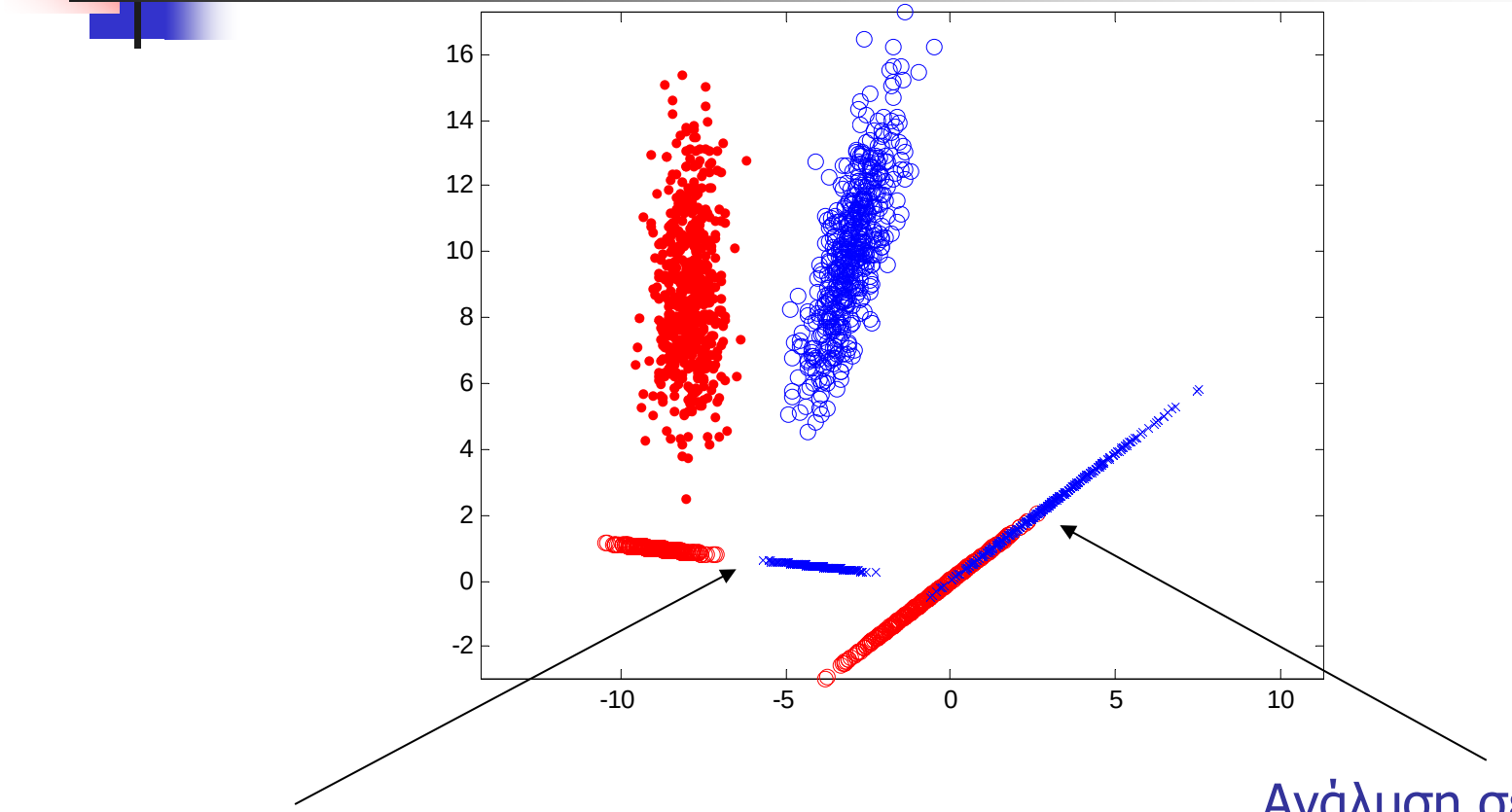


Σύγκριση PCA και FISHER

```
% Example 3.4.1 (modified by Chamzas)
% "Introduction to Pattern Recognition: A MATLAB Approach"
% S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras
 close('all'); clear;
randn('seed',0)
S1=[.3 .2; .2 5]; S2=[.5 1.2; 1.2 5];
[l,l]=size(S1);
mv=[-8 9; -3 10]';
N=500;
X=[mvnrnd(mv(:,1),S1,N); mvnrnd(mv(:,2),S2,N)]';
y=[ones(1,N), 2*ones(1,N)];
 % Plot the dataset
figure(1), plot(X(1,y==1),X(2,y==1),'r.',X(1,y==2),X(2,y==2),'bo')
figure(1), axis equal
```

```
% FISHER EYE ANALYSIS
% Estimate the mean vectors of each class using the available samples
mv_est(:,1)=mean(X(:,y==1)')';
mv_est(:,2)=mean(X(:,y==2)')';

% Compute the within scatter matrix
[Sw,Sb,Sm]=scatter_mat(X,y);
w=inv(Sw)*(mv_est(:,1)-mv_est(:,2));

% Compute the projections
t1=w'*X(:,y==1); t2=w'*X(:,y==2);
X_proj1=[t1;t1].*((w/(w'*w))*ones(1,length(t1)));
X_proj2=[t2;t2].*((w/(w'*w))*ones(1,length(t2)));
X_proj=[X_proj1 X_proj2];

%Plot the projections on the FISHER axis
figure(1), hold on
plot(X_proj(1,y==1),X_proj(2,y==1),'ro',X_proj(1,y==2),X_proj(2,y==2),'bx')
```

```
% Apply PCA on X
[eigenval,eigenvec,explained,Y,mean_vec]=pca_fun(X,1);
w=eigenvec;
% Compute the projections on PCA direction
t1=w'*X(:,y==1); t2=w'*X(:,y==2);
X_proj1=[t1;t1].*((w/(w'*w))*ones(1,length(t1)));
X_proj2=[t2;t2].*((w/(w'*w))*ones(1,length(t2)));
X_proj=[X_proj1 X_proj2];

%Plot the projections on the PCA axis
figure(1), hold on
plot(X_proj(1,y==1),X_proj(2,y==1),'r0',X_proj(1,y==2),X_proj(2,y==2),'bx')
```



Ανάλυση με κριτήριο την μέγιστη διαχωριστικότητα (Fisher Eye)

Ανάλυση σε κύριες συνιστώσες (PCA)