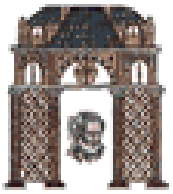
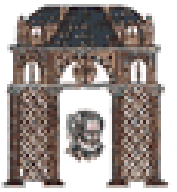


Αναγνώριση Προτύπων - Ασκήσεις

Γραμμικές Συναρτήσεις Διάκρισης
(Linear Discriminant Functions)

Χριστόδουλος Χαμζάς

Τα περιεχόμενα των παρουσιάσεων προέρχονται από τις παρουσιάσεις του αντίστοιχου διδακτέου μαθήματος του καθ. Παναγιώτη Τσακαλίδη, Τμ. Επιστήμης Υπολογιστών, Παν. Κρήτης και του καθ. Σέργιου Θεοδωρίδη, Τμήμα Πληροφορικής, Πανεπιστήμιο Αθηνών. Βασίζεται στα βιβλία: "Pattern Classification", R.O. Duda, P.E. Hart, D.G. Stork, Wiley, 2nd Ed., 2001 και S. Theodoridis, K. Koutroumbas, Pattern Recognition, 3rd Edition, Academic Press, 2006



Ο Αλγόριθμος Batch Perceptron

- Το διάνυσμα κλίσεων είναι: $$\nabla J_p(\mathbf{a}) = \left[\frac{\partial J_p}{\partial a_i}\right] = \sum_{y \in Y} -\mathbf{y}$$

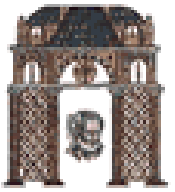
- Αναδρομική σχέση: $$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{y \in Y_k} \mathbf{y}$$

όπου Y_k είναι το σύνολο των δειγμάτων που έχουν ταξινομηθεί λάθος από το $\boldsymbol{a}_k$.

- Το επόμενο διάνυσμα βάρους (δ.β.) προκύπτει ως το άθροισμα του τρέχοντος δ.β. και ενός πολλαπλασίου του αθροίσματος των λάθος ταξινομημένων δειγμάτων.

Αλγόριθμος 3. Batch Perceptron

```
1 begin initialize a, κριτήριο θ, η(0)>0, k=0
2    do  k ← k+1
3        a ← a + η(k) Σ_{y∈Y_k} y
4    until |η(k) Σ_{y∈Y_k} y| < θ
5    return a
6 end
```



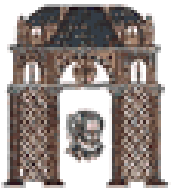
Batch Perceptron

$$w(k+1) = w(k) - \rho_k \sum_{y \in Y} y\,, \quad y = \begin{cases} x \text{ εάν } x \in \omega_1 \\ -x \text{ εάν } x \in \omega_2 \end{cases}$$

```
function [w,iter,mis_clas]=perce_with_plot(X,y,w_ini,rho)
%from  function [w,iter,mis_clas]=perce(X,y,w_ini,rho)
[l,N]=size(X);
max_iter=20000; % Maximum allowable number of iterations

w=w_ini;        % Initilaization of the parameter vector
iter=0;         % Iteration counter

mis_clas=N;     % Number of misclassfied vectors

while(mis_clas>0)&&(iter<max_iter)
   iter=iter+1;
   mis_clas=0;

   gradi=zeros(l,1); % Computation of the "gradient" term
   err=0;                          % evaluates the error CC
   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         gradi=gradi+rho*(-y(i)*X(:,i));
         err=err+w'*(y(i)*X(:,i));
                  %CC
      end
   end
   err   %prints the values of error CC
   if(iter==1)
      fprintf('\n First Iteration: # Misclassified points = %g
       \n',mis_clas);
   end
                                                  %plots the line CC
   XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL);
   w=w-rho*gradi; % Updating the parameter vector
end
```

FUNCTION
[w,iter,mis_clas]=perce(X,y,w_ini,rho)
Separates the vectors of two classes contained in a data set X with a hyperplane, which is iteratively adjusted via the perceptron learning rule. Note that the updating of the parameter vector at the t-th iteration is carried out after all the data vectors have been processed by the algorithm.
NOTE: In this implementation, the learning rate is chosen to be constant.

INPUT ARGUMENTS:
X: lxN matrix whose columns are the data vectors to be classfied.
y: N-dimensional vector whose i-th component contains the label of the class where the i-th data vector belongs (+1 or -1).
w_ini: l-dimensional vector, which is the initial estimate of the parameter vector that corresponds to the separating hyperplane.
rho: the learning rate parameter for the perceptron algorithm.
OUTPUT ARGUMENTS:
w: the final estimate of the parameter vector.
iter: the number of iterations required for the convergence of the algorithm.
mis_clas: number of misclassified data vectors.



Fixed Increment Single-Sample Perceptron

- Αντί να δοκιμάζουμε το διάνυσμα βαρών ***a(k)*** σε όλα τα δείγματα και να το διορθώνουμε βάσει του συνόλου Y_k των λάθος ταξινομημένων δειγμάτων, χρησιμοποιούμε τα δείγματα ένα κάθε φορά και ανάλογα με την ταξινόμηση του ανανεώνουμε ή όχι το διάνυσμα βαρών.
- Αν επιπλέον, χρησιμοποιήσουμε ένα σταθερό βήμα *η(k)*, τότε προκύπτει ο αλγόριθμος:

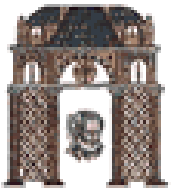
Αλγόριθμος 4. Fixed-Increment Single-Sample Perceptron

```
1 begin initialize a, k=0
2   do k ← (k+1) mod n
3     If y^k is misclassified by a, then a ← a + y^k
4   until all patterns properly classified
5   return a
6 end
```

Κυκλική Σειρά Δεδομένων (με πράσινο υποδηλώνονται τα λάθος ταξινομημένα δείγματα):

y_1 y_2 y_3 y_4 y_1 y_2 y_3 y_4 y_1 y_2 y_3 y_4

➔ y^1 y^2 y^3 y^0 $y^1 \ldots = y_2$ y_1 y_3 y_2 y_4



Single step perceptron

$$y = \begin{cases} x \text{ εάν } x \in \omega_1 \\ -x \text{ εάν } x \in \omega_2 \end{cases}$$

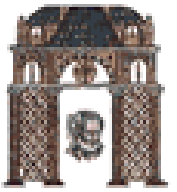
$$w(k+1) = \begin{cases} w(k) - \rho_k y_k & \text{εάν } w^T(k) y_k \le 0 \\ w(k) & \alpha\lambda\lambda o\iota\acute{\omega}\varsigma \end{cases}$$

```
[w,iter,mis_clas]=perce_online(X,y,w_ini,rho)

[l,N]=size(X);
max_iter=10000000; % Maximum allowable number of iterations

w=w_ini;        % Initilaization of the parameter vector
iter=0;         % Iteration counter

mis_clas=N;     % Number of misclassfied vectors

while(mis_clas>0)&&(iter<max_iter)
   mis_clas=0;

   for i=1:N
      if((X(:,i)'*w)*y(i)<0)
         mis_clas=mis_clas+1;
         w=w+rho*y(i)*X(:,i);  % Updating the parameter vector
      end
      iter=iter+1;
   end

   if(iter==1)
      mis_clas
   end
end
```

FUNCTION
[w,iter,mis_clas]=perce_online(X,y,w_ini,rho)
Separates the vectors of two classes contained in a data set X with a hyperplane, which is iteratively adjusted via the online perceptron learning rule. Note that the parameter vector is updated after the presentation of each data vector.
NOTE: In this implementation, the learning rate is chosen to be constant.

INPUT ARGUMENTS:
X: lxN dimensional matrix whose columns are the data vectors to be classfied.
y: N-dimensional vector whose i-th component contains the label of the class where the i-th data vector belongs (+1 or -1).
w_ini: l-dimensional vector which is the initial estimate of the parameter vector that corresponds to the separating hyperplane.
rho: the learning rate parameter for the perceptron algorithm.

OUTPUT ARGUMENTS:
w: the final estimate of the parameter vector.
iter: the number of iterations required for the convergence of the algorithm.
mis_clas: number of misclassified data vectors.



Variable-Increment Perceptron with Margin

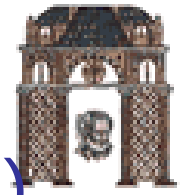
- Χρησιμοποιούμε τα δείγματα ένα κάθε φορά και διορθώνουμε το διάνυσμα βαρών $\boldsymbol{a}(k)$ όταν το εσωτερικό γινόμενό του με το δείγμα $\boldsymbol{y}^k$ είναι μικρότερο από κάποιο προκαθορισμένο θετικό όριο b: $\boldsymbol{a}(k)\boldsymbol{y}^k < b$.
- Αν επιπλέον, χρησιμοποιήσουμε ένα μεταβαλλόμενο βήμα $\eta(k)$, τότε προκύπτει ο αλγόριθμος:

Αλγόριθμος 5. Variable-Increment Perceptron w. Margin

```
1 begin initialize a, threshold θ, margin b, η(0), k=0
2   do k ← (k+1) mod n
3     if a^t y^k <b, then a ← a + η(k) y^k
4   until a^t y^k >b for all k
5   return a
6 end
```

Συνθήκες Σύγκλισης:

$$\eta(k) \geq 0, \quad \lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty, \quad \lim_{m\to\infty} \frac{\sum_{k=1}^{m} \eta^2(k)}{\left(\sum_{k=1}^{m} \eta(k)\right)^2} = 0$$



Μέθοδοι Χαλάρωσης (Relaxation Procedures)

- Συνάρτηση κριτηρίου:

$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in Y}\frac{\left(\mathbf{a}^t\mathbf{y}-b\right)^2}{\|\mathbf{y}\|^2}$$

όπου $Y(\mathbf{a})$ είναι το σύνολο των δειγμάτων για τα οποία $\mathbf{a}^t\mathbf{y}<b$.

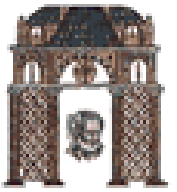
- Αν το $Y(\mathbf{a})$ είναι κενό, τότε $J_r(\mathbf{a})=0$. Η $J_r(\mathbf{a})$ δεν είναι ποτέ αρνητική και μηδενίζεται αν και μόνο αν $\mathbf{a}^t\mathbf{y}>b$ για όλα τα δείγματα εκπαίδευσης.

- Το διάνυσμα κλίσεων είναι:

$$\nabla J_r(\mathbf{a}) = \left[\frac{\partial J_r}{\partial a_i}\right] = \sum_{y\in Y}\frac{\mathbf{a}^t\mathbf{y}-b}{\|\mathbf{y}\|^2}\mathbf{y}$$

- Αναδρομική σχέση:

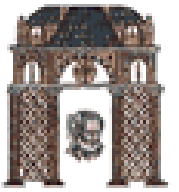
$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\sum_{y\in Y_k}\frac{b-\mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$$



Batch Relaxation with Margin

Αλγόριθμος 6. Batch Relaxation with Margin

```
1  begin initialize a, margin b, η(0), k=0
2    do k ← (k+1) mod n
3      Y_k={}
4      j=0
5      do j ← j+1
6        if a^t y^j <b, then append y^j to Y_k
7      until j=n
8      a ← a+η(k) Σ_{y∈Y} (b−a^t y)/‖y‖^2 y
9    until Y_k={}
10   return a
11 end
```



Duda Matlab Software

Στο λογισμικό των Stork and Yom-Tov που αντιστοιχεί στο βιβλίο του Duda χρησιμοποιείται η ονομασία

patterns: Είναι ένα **d** by **n** πινακας που περιέχει τα δεδομένα, όπου **d** είναι η διάσταση των δεδομένων και **n** ο συνολικός αριθμός τους

targets: είναι ένα **1** by **n** διάνυσμα που περιέχει τις κατηγορίες στις οποίες ανήκουν τα δεδομένα που είναι στον πίνακα pattern

Παράδειγμα: patterns=[0.1 0.2 2.1 3.2 -0.0; 2.1 3.2 1.2 5.1 3.1; 0.2 0.4 0.5 1.3 2.2]

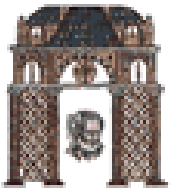
targets=[1 1 0 1 1 0]

είναι 5 δεδομένα με 3 συνιστώσες και τα οποία κατηγοριοποιούνται σε 2 κλάσεις

Στο λογισμικό του Stork and Yom-Tov τα δεδομένα αυξάνονται κατά μία διάσταση μέσα στην συνάρτηση, καθώς και το αρχικό διάνυσμα βαρών λαμβάνεται ο μέσος όρος των δεδομένων ενώ οι κλάσεις είναι οι 0 και 1 (Πικράκης -1 και 1)

Ο Πικράκης όταν δημιουργεί το Y δεν αλλάζει πρόσημα αλλά βάζει σε -1 το label της κλάσης.

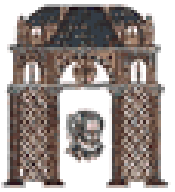
Επίσης Πικράκης Y^T=Y Duda.



Batch Relaxation with Margin

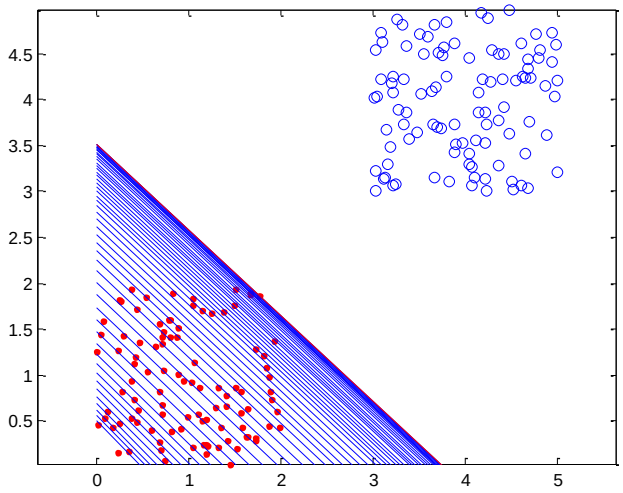
```
function [test_targets, a] = Relaxation_BM(train_patterns,
train_targets, test_patterns, params)

% From DUDA, Hart, Stork
%Classify using the batch relaxation with margin algorithm
% Inputs:
%   train_patterns  - Train patterns
%   train_targets   - Train targets
%   test_patterns   - Test  patterns
%   params          - [Max iter, Margin, Convergence rate]
%
% Outputs
%   test_targets    - Predicted targets
%   a               - Classifier weights
%
% NOTE: Works for only two classes.

[c, n]              = size(train_patterns);
[Max_iter, b, eta]  = process_params(params);
% generate Y data by aygmenting the X data (patterns) by 1 /CC
y                   = [train_patterns ; ones(1,n)];
train_zero          = find(train_targets == 0);

%Preprocessing
processed_patterns = y;
processed_patterns(:,train_zero) = -processed_patterns(:,train_zero);

%Initial weights
a                   = sum(processed_patterns')';
iter                = 0;
Yk                  = [1];
```

```
while (~isempty(Yk) & (iter < Max_iter))
  iter = iter + 1;

  %If a'y_j <= b then append y_j to Yk
  Yk = [];
  for k = 1:n,
   if (a'*processed_patterns(:,k) <= b),
       Yk = [Yk k];
    end
  end

  if isempty(Yk),
    break
  end

  % a <- a + eta*sum((b-w'*Yk)/||Yk||*Yk)
  grad        = (b-a'*y(:,Yk))./sum(y(:,Yk).^2);
  update      = sum(((ones(c+1,1)*grad).*y(:,Yk))')';
  a           = a + eta * update;
end

if (iter == Max_iter),
  disp(['Maximum iteration (' num2str(Max_iter) ') reached']);
end

%Classify test patterns

test_targets = a'*[test_patterns; ones(1, size(test_patterns,2))] > 0;
```



Παράδειγμα Batch Perceptron

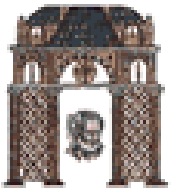
```
% Example 2.2.1 "Introduction to Pattern Recognition: A MATLAB Approach"  S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras

close('all');
clear

rand('seed',1);

% Generate the dataset X1 as well as the vector containing the class labels of
% the points in X1
N=[100 100]; % 100 vectors per class
l=2; % Dimensionality of the input space

x=[3 3]';
% x=[2 2]'; for X2
% x=[0 2]'; for X3
% x=[1 1]'; for X4

X1=[2*rand(l,N(1)) 2*rand(l,N(2))+x*ones(1,N(2))];
X1=[X1; ones(1,sum(N))];
y1=[-ones(1,N(1)) ones(1,N(2))];

% 1. Plot X1, where points of different classes are denoted by different colors,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal
hold on; axis(axis);
% 2. Run the perceptron algorithm for X1 with learning parameter 0.01
rho=0.02; % Learning rate
w_ini=[1 1 -.5]';
[w,iter,mis_clas]=perce(X1,y1,w_ini,rho)                % this is the basic routine
%  [w,iter,mis_clas]=perce_with_plot(X1,y1,w_ini,rho)  %to see all lines
 XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL, 'r'); hold off ;
```



%CC

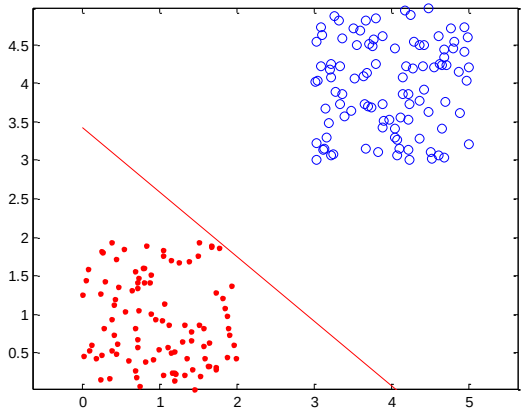
%plot the final line CC

% Ξεκινήστε από διαφορετικό αρχικό σημείο, με διαφορετικά δεδομένα, με διαφορετικές παραμέτρους και σχολιάστε τα αποτελέσματα



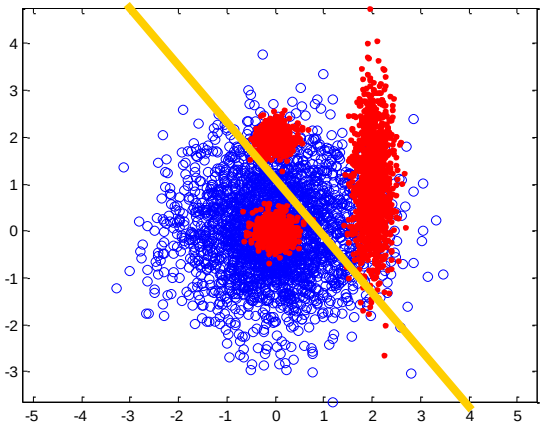
Single step perceptron

```
% .....................................
% 1. Plot X1, where points of different classes are denoted by different colors,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal
hold on; axis(axis);                                                    %CC
% 2. Run the perceptron algorithm for X1 with learning parameter 0.02
rho=0.02; % Learning rate
w_ini=[1 1 -.5]';
[w,iter,mis_clas]=perce_online(X1,y1,w_ini,rho)
 XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL, 'r'); hold off ;     %CC
```



$W = [0.2733 \quad 0.3275 \quad -1.1200]^T$
Iter=400, err=0

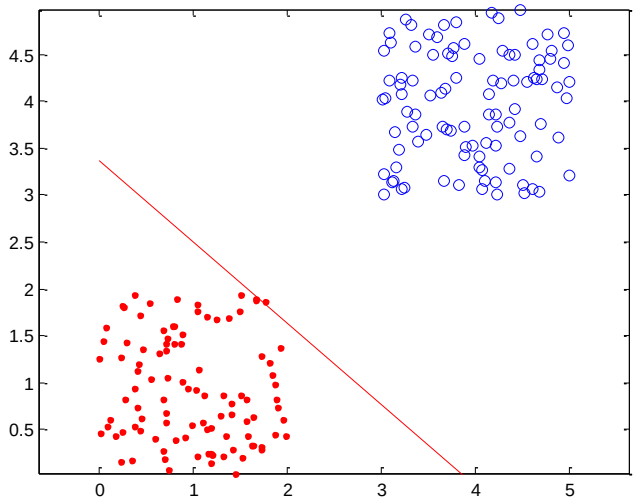
```
% We load data from Duda

load clouds; [c n]=size(patterns);
X1=[patterns; ones(1, n)];                  % converts to 3d by adding 1
y1=2*targets-1;                             % set the classes to 1 and -1 from 1 and 0

figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal
hold on; axis(axis);                                                    %CC
% 2. Run the perceptron algorithm for X1 with learning parameter 0.01
rho=0.01; % Learning rate
w_ini=[1 1 -.5]';
[w,iter,mis_clas]=perce_online(X1,y1,w_ini,rho)
 XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL, 'r'); hold off ;     %CC
perror=mis_clas/n
```

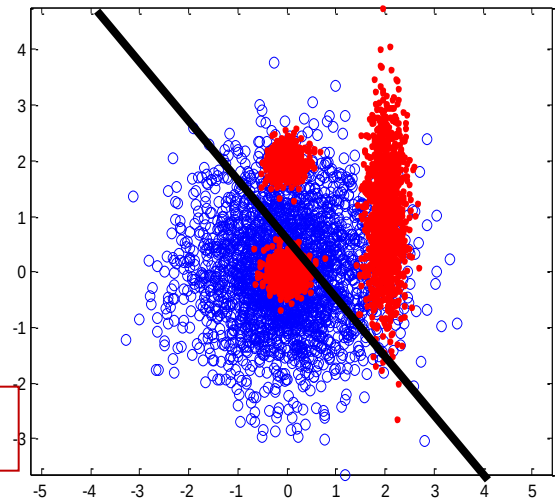


$W = [-0.0113 \quad -0.0092 \quad 0.010]^T$
Iter=10^7, no converge, perr=0.35

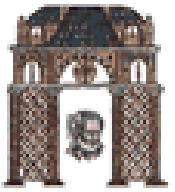
Batch relaxation with margin

```
% Example 2.2.1a
% Simulates the Duda Batch relaxation with margin  &5.6.3
 close('all');  clear
 rand('seed',1);
% Generate the dataset X1 as well as the vector containing the class labels of the points in X1
N=[100 100]; % 100 vectors per class
I=2; % Dimensionality of the input space
 x=[3 3]';
% x=[2 2]'; for X2  % x=[0 2]'; for X3  % x=[1 1]'; for X4

X1=[2*rand(I,N(1)) 2*rand(I,N(2))+x*ones(1,N(2))];
y1=[zeros(1,N(1)) ones(1,N(2))];  %assigns 0 and 1 for the classes

% or read CLOUDS
% load clouds; X1=patterns; y1=targets;

% 1. Plot X1, where points of different classes are denoted by different colors,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==0),X1(2,y1==0),'r.')
figure(1), axis equal
hold on; axis(axis);
% 2. Run the perceptron algorithm for X1 with learning parameter 0.01
params=[5 0.1 0.001]; %   params=[Max iter, Margin, Convergence rate]

[test_targets, w] = Relaxation_BM(X1, y1, X1, params); w

XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL, 'y'); hold off ;
[c n]=size(test_targets); perror=sum(abs(test_targets-y1))/n
```





W = [-1,742 -1.5957 1.000]^T
perr=0.29



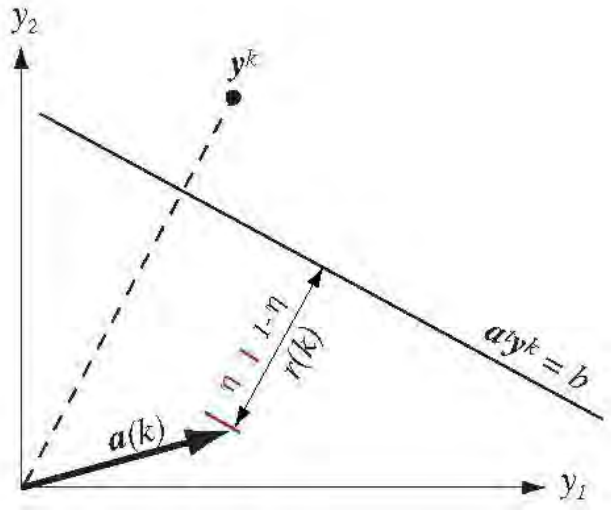
Single-Sample Relaxation with Margin

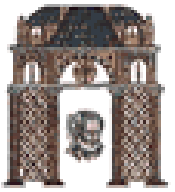
Αλγόριθμος 7. Single-Sample Relaxation with Margin

```
1 begin initialize a, margin b, η(0), k=0
2   do k ← (k+1) mod n
3       if a^t y^k < b, then  a ← a + η(k) (b − a^t y^k) / ||y^k||^2 y^k
4   until a^t y^k > b for all y^k
5   return a
6 end
```

Σε κάθε βήμα, το διάνυσμα βαρών ***a(k)***, μετατοπίζεται προς το υπερεπίπεδο $\mathbf{a}^t\mathbf{y}^k=b$ κατά ένα ποσοστό, *η(k)*, της απόστασής του, *r(k)*, από αυτό.

η(k)<1 → underrelaxation
η(k)>1 → overrelaxation
0<η(k)<2 για σύγκλιση





Μη Διαχωρίσιμες Κατηγορίες

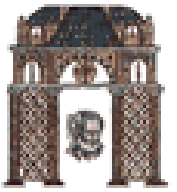
- Οι προηγούμενοι αλγόριθμοι βασίζονται στην υπόθεση ότι οι κατηγορίες είναι γραμμικά διαχωρίσιμες.
- Αλλά ακόμα και εάν το δείγμα εκπαίδευσης είναι γραμμικά διαχωρίσιμο αυτό δεν εγγυάται καλή συμπεριφορά στην πραγματικότητα

ΠΩΣ ΘΑ ΣΥΜΠΕΡΙΦΕΡΟΝΤΑΙ ΣΕ ΜΗ ΓΡΑΜΜΙΚΑ ΔΙΑΧΩΡΙΣΙΜΕΣ ΚΑΤΗΓΟΡΙΕΣ

Ας βάλλουμε όλα τα δεδομένα με ένα margin $a^T y_i - b_i > 0$

και ας προσπαθήσουμε να ελαχιστοποιήσουμε μια συνάρτηση κριτηρίου που βασίζεται στα ελάχιστα τετράγωνα ->

Μέθοδο Ελαχίστων Τετραγώνων



Μέθοδοι Ελαχίστων Τετραγώνων (Minimum Square Error – MSE)

- Στόχος: Καλή απόδοση τόσο στις γραμμικά διαχωρίσιμες όσο και στις μη γραμμικά διαχωρίσιμες περιπτώσεις.
- Πώς: Κριτήριο που περιλαμβάνει όλα τα πρότυπα. Επίσης,
  - Πρίν: Βρες $\boldsymbol{a}$ έτσι ώστε $\boldsymbol{a}^t\boldsymbol{y}_i>0$ για κάθε πρότυπο $\boldsymbol{y}_i$.
  - Τώρα: Βρες $\boldsymbol{a}$ έτσι ώστε $\boldsymbol{a}^t\boldsymbol{y}_i=b_i$ για κάθε πρότυπο $\boldsymbol{y}_i$. (b_i θετικές σταθερές).
- Δηλαδή; Μετατρέπουμε το πρόβλημα επίλυσης ενός συνόλου γραμμικών ανισοτήτων σε πρόβλημα επίλυσης γραμμικών εξισώσεων.
- Συμβολισμοί:

$$\mathbf{Y}_{n\times\hat{d}}=\begin{bmatrix}\mathbf{y}_1^t\\ \mathbf{y}_2^t\\ \vdots\\ \mathbf{y}_n^t\end{bmatrix}$$

Πίνακας Προτύπων

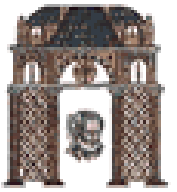
$$\mathbf{b}_{n\times 1}=\begin{bmatrix}b_1\\ b_2\\ \vdots\\ b_n\end{bmatrix}$$

Διάνυσμα θετικών παραμέτρων

$$\mathbf{a}_{\hat{d}\times 1}=\begin{bmatrix}a_0\\ a_1\\ \vdots\\ a_d\end{bmatrix}$$

Διάνυσμα βαρών

- Πρόβλημα: Εύρεση $\boldsymbol{a}$ έτσι ώστε: $\boldsymbol{Ya}=\boldsymbol{b}$.



Μέθοδοι Ελαχίστων Τετραγώνων (Minimum Square Error – MSE)

- Συνήθως $n>d+1$, περισσότερα πρότυπα από διαστάσεις → σύστημα υπερπροσδιορισμένο, δεν έχει ακριβή λύση.
- Επομένως; Ελαχιστοποίηση του τετραγώνου του μήκους του διανύσματος σφάλματος, $\boldsymbol{e}=\boldsymbol{Ya}-\boldsymbol{b}$:

$$J_s(\mathbf{a}) = \|\mathbf{Ya}-\mathbf{b}\|^2 = \sum_{i=1}^{n}\left(\mathbf{a}^t\mathbf{y}_i - b_i\right)^2$$

- Κλασσικό Πρόβλημα:

$$\nabla J_s(\mathbf{a}) = \mathbf{0} \Rightarrow \mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}$$

Κανονικές Εξισώσεις
Normal Equations

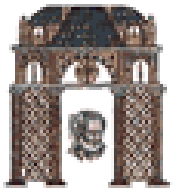
- Αν ο $\boldsymbol{Y}^t\boldsymbol{Y}$ είναι ομαλός,

$$\mathbf{a} = \underbrace{\left(\mathbf{Y}^t\mathbf{Y}\right)^{-1}\mathbf{Y}^t}_{\text{ψευδοαντίστροφος}}\ \mathbf{b}$$

Εάν ο $\boldsymbol{Y}^T\boldsymbol{Y}$ δεν είναι ομαλός, δηλ. $\|\boldsymbol{Y}^T\boldsymbol{Y}\|=0$, ήσηε προζθέηοσμε μία μικρή ζηαθερά **e** ζηην διαγώνιο και έτοσμε

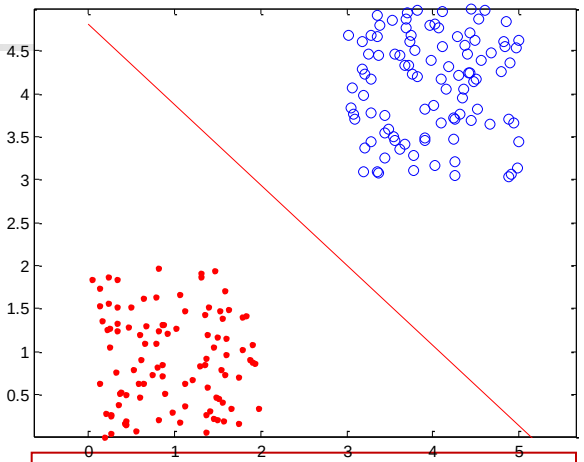
$$\mathbf{a} = \left(\mathbf{Y}^{\mathbf{T}}\mathbf{Y} + e\mathbf{I}\right)^{-1}\mathbf{Y}^{\mathbf{T}}\mathbf{b}$$

```
%MATLAB
[M,N]=size(Y);
a=inv(Y'*Y+e*eye(M))*(Y'*b);
```

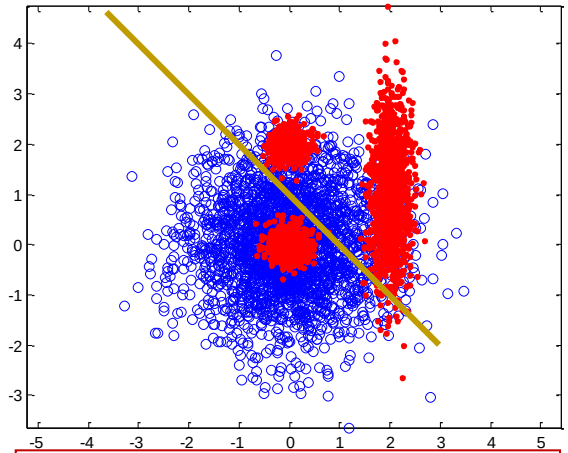


Χρησιμοποίησε τα προηγούμενα δεδομένα και διαχώρισέ τα με MSE

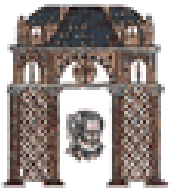
```
% Example 2.3.1cc By C. Chamzas for LSE
.....
N1=100; N2=100; N=N1+N2; % N1 vectors from class 1 and N2 vectors from class 2
l=2; % Dimensionality of the input space
x=[3 3]';

X1=[2*rand(l,N1) 2*rand(l,N2)+x*ones(1,N2)];
y1=[-ones(1,N1) ones(1,N2)];

% OR We load data from Duda (uncomment the 3 next lines
% load clouds; [c N]=size(patterns);
% X1=patterns;
% y1=2*targets-1;          % set the classes to 1 and -1 from 1 and 0

% 1. Plot X1, where points of different classes are denoted by different colors,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',...
X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal
hold on; axis(axis);

% 2. Augment the data vectors of X1 by adding 1 at the END
Y=[X1; ones(1,N)];
% instead of negating the Y data of class 1, we negate the constants b
b=y1;

% Compute the classification error of the LS classifier based on Y
[w]=SSErr(Y,b,0);
SSE_out=2*(w'*Y>0)-1;
err_SSE=sum(SSE_out.*y1<0)/N
%Plot the line
XL=[0 -w(3)/w(1)] ; YL=[-w(3)/w(2) 0]; plot(XL,YL, 'r'); hold off ;
```



$W = [-0.22\ -0.18\ 1.00]^T$
perr=0.



$W = [-0.98\ -0.99\ 1.00]^T$
perr=0.25

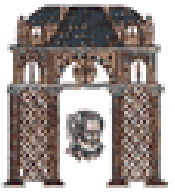


Widrow-Hoff (Least Mean Squares - LMS)

- Η $J_s(\boldsymbol{a})$ μπορεί να ελαχιστοποιηθεί μέσω αναδρομικών αλγορίθμων που δεν απαιτούν την αντιστροφή πινάκων.
- Διάνυσμα κλίσεων: $\nabla J_s(\mathbf{a}) = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$
- Βασική αναδρομική σχέση: $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{Y}^t(\mathbf{b} - \mathbf{Ya}(k))$
- Χρησιμοποιώντας ένα δείγμα σε κάθε βήμα, προκύπτει ο αλγόριθμος Widrow-Hoff (LMS):

Αλγόριθμος 8. Widrow-Hoff (LMS)

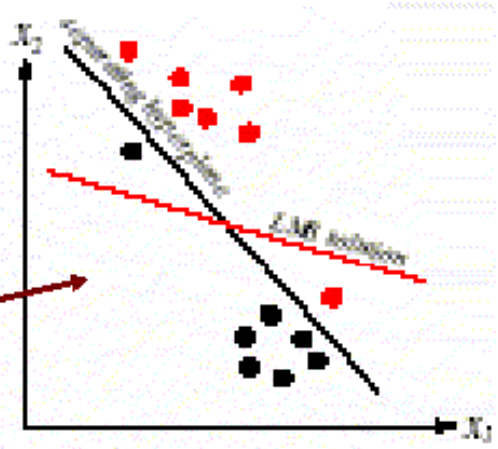
1 **begin initialize** a, b, κριτήριο θ, η(), k=0
2 **do** $k \leftarrow (k+1) \bmod n$
3 $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k$
4 **until** $|\eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k| < \theta$
5 **return** a
6 **end**



Μέθοδος Ho-Kashyap

- Το perceptron και οι τεχνικές χαλάρωσης βρίσκουν διαχωριστικά διανύσματα βαρών αν τα δείγματα είναι γραμμικά διαχωρίσιμα, αλλά δεν συγκλίνουν για μη διαχωρίσιμες κλάσεις.

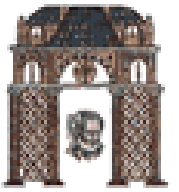
- Οι τεχνικές ελαχίστων τετραγώνων δίνουν πάντα ένα διάνυσμα λύσης (αυτό που ελαχιστοποιεί το $\|\mathbf{Ya}-\mathbf{b}\|^2$) το οποίο όμως δεν είναι απαραίτητα διαχωριστικό στην διαχωρίσιμη περίπτωση.



- Ο αλγόριθμος Ho-Kashyap λύνει αναδρομικά το εξής πρόβλημα ελαχιστοποίησης:

$$\min_{\mathbf{a},\mathbf{b}} J_s(\mathbf{a},\mathbf{b}) = \|\mathbf{Ya}-\mathbf{b}\|^2 \quad s.t. \quad \mathbf{b} > \mathbf{0}$$

- Είναι ένας αλγόριθμος που φροντίζει ώστε το $\boldsymbol{b}$ να μην συγκλίνει στο $\boldsymbol{0}$, θέτοντας όλες τις θετικές συνιστώσες του διανύσματος κλίσης $\nabla_b J_s$ ίσες με το μηδέν.

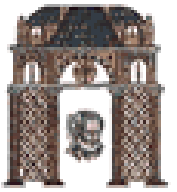


Αλγόριθμος Ho-Kashyap

Αλγόριθμος 9. Ho-Kashyap

```
1  begin initialize a, b, η()<1, threshold b_min, k_max
2    do k ← (k+1) mod n
3       e ← Ya-b
4       e+ ← (e+|e|)/2
5       b ← b + 2η(k)e+
6       a ← (YtY)-1Yb
7       if Abs(e)< b_min then return a, b and exit
8    until k= k_max
9    print "No solution found"
10 end
```

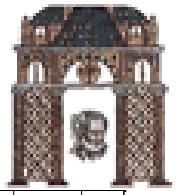
Βασικά βρίσκουμε πρώτα το gradient descent ως προς b και μετά εφαρμόζουμε ελάχιστα τετράγωνα



Αλγόριθμος Ho-Kashyap σε Matlab

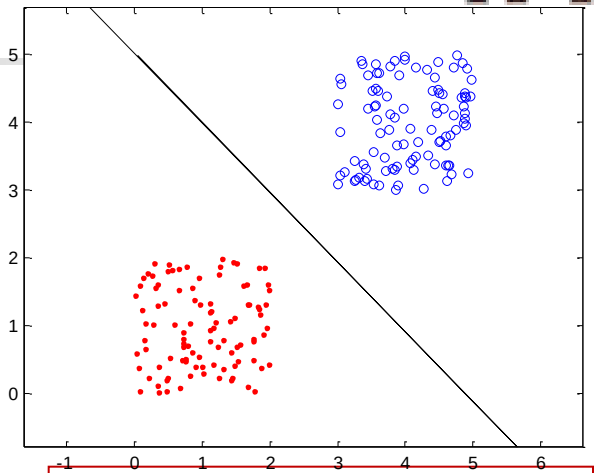
```
function [a, b] = Ho_Kashyap_cc(train_features, train_targets, type,
Max_iter, b_min, eta)

% Classify using the using the Ho-Kashyap algorithm
% Inputs:
%   features    - Train features
%   targets     - Train targets
%   Type(0 Basic/1 Modified), Maximum iteration, Convergence criterion,
Convergence rate
%
% Outputs
%   a           - Classifier weights
%   b           - Margin

[c, n]         = size(train_features);
train_class2   = find(train_targets == -1);

%Preprocessing (Needed so that b>0 for all features)
Y = train_features;
Y(:,train_class2) = -Y(:,train_class2);

b                = ones(1,n);
a                = pinv(Y')*b';
k                = 0;
e                = 1e3;
found              = 0;
```

```
while ((sum(abs(e) > b_min)>0) & (k < Max_iter) &(~found))
    k = k+1;
    e  = (Y' * a)' - b;
    e_plus  = 0.5*(e + abs(e));
    b = b + 2*eta*e_plus;

    if (type==0),
       a = pinv(Y')*b';
    else
       a = a + eta*pinv(Y')*e_plus';
    end
end

if (k == Max_iter),
   disp(['No solution found']);
else
   disp(['Did ' num2str(k) ' iterations'])
end
```

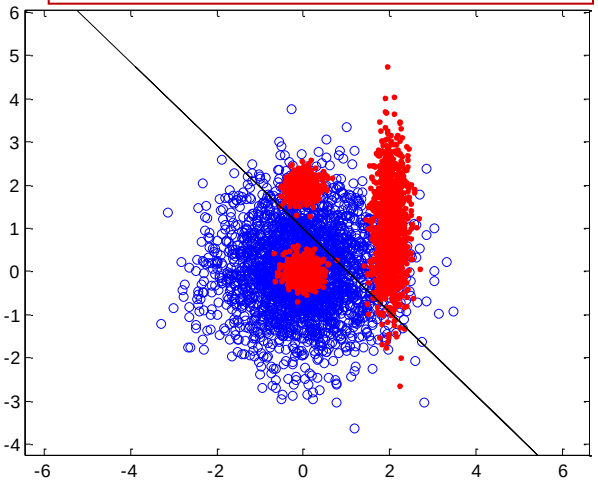


Παράδειγμα

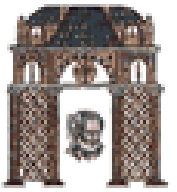
```
% Example ìå Ho-Kashyap  modified By C. Chamzas from Duda Software.
clear all;
N1=100; N2=100; N=N1+N2; % N1 vectors from class 1 and N2 vectors from class 2
l=2; % Dimensionality of the input space
x=[3 3]';

X1=[2*rand(l,N1) 2*rand(l,N2)+x*ones(1,N2)];
y1=[-ones(1,N1) ones(1,N2)];

% OR We load data from Duda (uncomment the 3 next lines
%----------------------
load clouds; [c N]=size(patterns);
X1=patterns; y1=2*targets-1;          % the classes are set also to 1 and -1
%----------------------
% 1. Plot X1, where points of different classes are denoted by different colors,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'bo',X1(1,y1==-1),X1(2,y1==-1),'r.')
figure(1), axis equal; hold on; XA=axis;

X1=[X1;ones(1,N)]; %increases the dimentionality of the training data by 1
[w,b]=Ho_Kashyap_cc(X1,y1, 0, 1000, 0.1, 0.01);

% Plots the discrimination line
XL=[ XA(1) XA(2) -(w(3)+w(2)*XA(3))/w(1) -(w(3)+w(2)*XA(4))/w(1)];
    YL=[-(w(3)+w(1)*XA(1))/w(2) -(w(3)+w(1)*XA(2))/w(2) XA(3) XA(4)];
    plot(XL,YL, 'k'); hold off

%prints the classification error
KH_out=2*(w'*X1>0)-1;
err_KH=sum(KH_out.*y1<0)/N
```



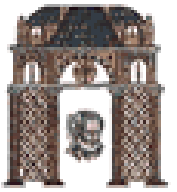
$W = [-0.21\ -0.20\ 1.00]^T$
perr=0.



$W = [-0.97\ -1.00\ 1.00]^T$
perr=0.25



- Έως τώρα είδαμε το πρόβλημα μόνο για 2 κατηγορίες. Τι μπορούμε να κάνουμε εάν έχουμε πολλές κατηγορίες προτύπων?



Ταξινόμηση Πολλαπλών Κλάσεων με MSE

➢Έστω $\mathbf{y_1, \ldots, y_n}$, πρότυπα από c>2 κλάσεις. Θέλουμε να βρούμε ένα ταξινομητή που να αποτελείται από γραμμικές συναρτήσεις διάκρισης $g_i(\mathbf{x}) = w_i\mathbf{x} + w_{i0}$ (μία για κάθε κλάση), ούτως ώστε εάν x ανήκει στην i κλάση τότε $g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i$. Συνεπώς στον επαυξημένο χώρο ψάχνουμε για τα βάρη $\mathbf{a_i}$ ώστε εάν y_k ανήκει στην i κλάση τότε

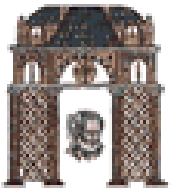
$$\mathbf{a_i} : \mathbf{a_i^t y}_\kappa > \mathbf{a_j^t y}_\kappa \quad \forall j \neq i$$

➢Για την επίλυση του προβλήματος συνεπώς υπολογίζουμε το βέλτιστο $\mathbf{a_i}$ ώστε να διαχωρίζει την κλάση i από όλες τις υπόλοιπες κλάσεις j≠i .

$$\mathbf{a}_\iota = \left(\mathbf{Y^T Y}\right)^{-1} \mathbf{b}_\iota$$

Y= [$\mathbf{y_1, \ldots, y_n}$] και $\mathbf{b_i = [b_{jk}]}$ k=1,…,n και b_{jk} =1 εάν $\mathbf{y_j}$ ανήκει στην κλάση ω_i και b_{jk} =-1 εάν $\mathbf{y_j}$ **δεν** ανήκει στην κλάση ω_i

➢Σημειώνεται ότι εάν δεν θέλουμε να αλλάξουμε το πρόσημο στα $\mathbf{y_j}$, μπορούμε να το αλλάξουμε στα αντίστοιχα $\mathbf{b_i}$



Ταξινόμηση Πολλαπλών Κλάσεων με MSE

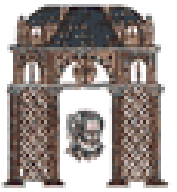
Παράδειγμα. Έστω πρόβλημα ταξινόμησης σε τέσσερεις ισοπίθανες κλάσεις $\omega_1,\omega_2,\omega_3,\omega_4$. Η κάθε κλάση μοντελοποιείται με Gaussian κατανομές μέσων τιμών $m_1=[1,1]^T$, $m_2=[5,10]^T$, $m_3=[10,4]^T$, $m_4=[10,10]^T$. Τα μητρώα συνδιασποράς είναι

$S_1=[0.8\ 0.2;0.2\ 0.1]$, $S_2=[0.8\ 0.2;0.2\ 0.8]$, $S_3=[0.1\ 0.25;0.2\ 5,0.8]$, $S_3=[0.2\ 0.3;0.3\ 0.8]$,

Έχουμε N=10000 δεδομένα και θέλουμε να βρούμε τις συναρτήσεις διάκρισης

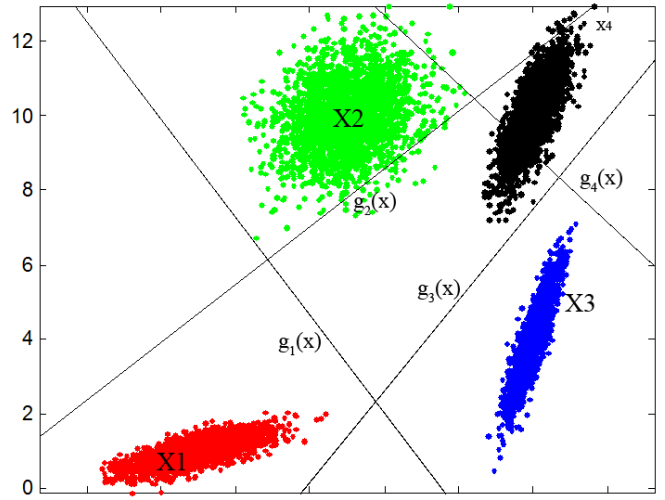
```
% Example 2.3.3 CC
close('all');
clear;

m=[1 1 ; 5 10 ; 10 4; 10 10]';
[l,c]=size(m);
S1=[0.8 0.2; 0.2 0.1]; S2=[0.8 0.2; 0.2 0.8];
S3=[0.1 0.25; 0.25 0.8]; S4=[0.2 0.3; 0.3 0.8];
S(:,:,1)=S1; S(:,:,2)=S2; S(:,:,3)=S3; S(:,:,4)=S4;
P=[1/4 1/4 1/4 1/4];

% 1. Generate X the training set
N1=10000;
randn('seed',0)
[X,y1]=generate_gauss_classes(m,S,P,N1);
[l,N1]=size(X);
X1=[X; ones(1,N1)];

% Plot X1 using different colors for points of different classes,
figure(1), plot(X1(1,y1==1),X1(2,y1==1),'r.',...
    X1(1,y1==2),X1(2,y1==2),'g.',...
    X1(1,y1==3),X1(2,y1==3),'b.',...
    X1(1,y1==4),X1(2,y1==4),'k.')
axis equal; hold on; axis(axis); A=axis;
```

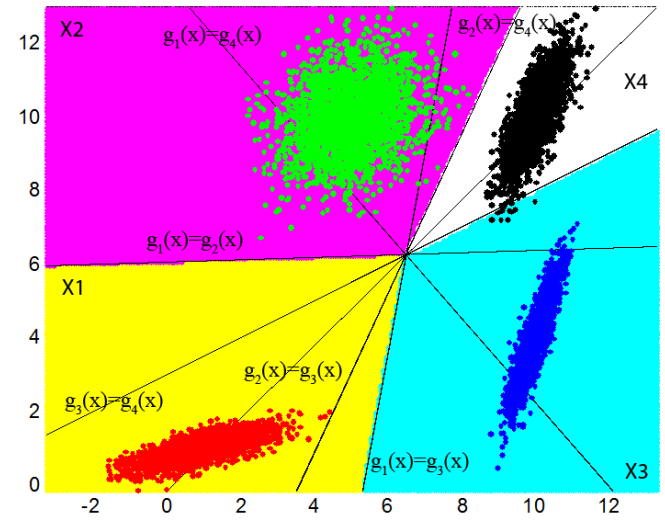
```
% Next, we define matrix z1, each column of which
% corresponds to a training point.
z1=zeros(c,N1);  z1=z1-1;  % set 1 to class i and -1 to all other classes
for i=1:N1
    z1(y1(i),i)=1;
end

% Estimate the parameter vectors of the c discriminant functions
w_all=[];
for i=1:c
    w=SSErr(X1,z1(i,:),0);
    w_all=[w_all w];
end
% Note: in w_all, the i-th column corresponds to the parameter vector
% of the i-th discriminant function.
% Now plots the gi(x)
for i=1:c
    w(:)=w_all(:,i);
    XL=[ A(1) A(2) -(w(3)+w(2)*A(3))/w(1) -(w(3)+w(2)*A(4))/w(1)];
    YL=[-(w(3)+w(1)*A(1))/w(2) -(w(3)+w(1)*A(2))/w(2) A(3) A(4)];
    plot(XL,YL, 'k');
end
% Compute the classification error using the training set X1
[vali,class_est]=max(w_all'*X1);
err=sum(class_est~=y1)/N1
```

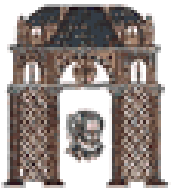


MLS ταξινόμηση

```
% plots segmended plane and gi(x)-gj(x)
% Generate XP as a set of points covering all the plane (NPxNP grid)
NP=250;
DX=(A(2)-A(1))/(NP-1); DY=(A(4)-A(3))/(NP-1);
XP1=[A(1):DX:A(2)];XP2=[A(3):DY:A(4)];
XP=[ones(1,NP*NP);ones(1,NP*NP)];
for i=1:NP
    for j=1:NP
        XP(:,(i-1)*NP+j)=[XP1(i);XP2(j)];
    end
end
XP=[XP;ones(1,NP*NP)];
[vali,class_est]=max(w_all'*XP);
% plots the segmented plane
figure(2), plot(XP(1,class_est==1),XP(2,class_est==1),'y.',...
    XP(1,class_est==2),XP(2,class_est==2),'m.',...
    XP(1,class_est==3),XP(2,class_est==3),'c.',...
    XP(1,class_est==4),XP(2,class_est==4),'w.'); axis(A);hold on;
% Now plots the gi(x)-gj(x)  i<j
for i=1:c
    for j=i+1:c
    w(:)=w_all(:,i)-w_all(:,j);
    XL=[ A(1) A(2) -(w(3)+w(2)*A(3))/w(1) -(w(3)+w(2)*A(4))/w(1)];
    YL=[-(w(3)+w(1)*A(1))/w(2) -(w(3)+w(1)*A(2))/w(2) A(3) A(4)];
    plot(XL,YL, 'k');
    end
end
plot(X1(1,y1==1),X1(2,y1==1),'r.',...
    X1(1,y1==2),X1(2,y1==2),'g.',...
    X1(1,y1==3),X1(2,y1==3),'b.',...
    X1(1,y1==4),X1(2,y1==4),'k.')
```







Πολλαπλές Κλάσεις: Δομή του Kesler

- Στόχος: Γραμμικός διαχωρισμός πολλαπλών κλάσεων:

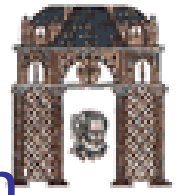
  Αν $y \sim \omega_1$, τότε $a_1^t y - a_j^t y > 0$ για κάθε $j=2,\ldots,c$.

- Αυτό το σύστημα των c-1 ανισοτήτων μπορεί να περιγραφεί ως εξής:
  Το $c\hat{d} - D$ διάνυσμα βαρών $\hat{\boldsymbol{\alpha}}$ ταξινομεί ορθά όλα τα c-1 $c\hat{d} - D$ πρότυπα $\eta_{12}, \eta_{13},\ldots, \eta_{1c}$: $\hat{\boldsymbol{\alpha}}^t \boldsymbol{\eta}_{1j} > 0 \quad \forall \quad j = 2,\ldots,c$ όπου:

$$\hat{\boldsymbol{\alpha}}_{c\hat{d}\times 1} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_c \end{bmatrix} \quad \boldsymbol{\eta}_{12} = \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{\eta}_{13} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ -\mathbf{y} \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad ,\cdots, \quad \boldsymbol{\eta}_{1c} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ -\mathbf{y} \end{bmatrix}$$

Δομή Kesler

- Γενικότερα: $\hat{\boldsymbol{\alpha}}^t \boldsymbol{\eta}_{ij} > 0 \quad \forall \quad j \neq i,$ όπου $\boldsymbol{\eta}_{ij} = \begin{bmatrix} \vdots \\ \mathbf{y} \\ \vdots \\ -\mathbf{y} \\ \vdots \end{bmatrix} \begin{matrix} \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{matrix}$



Ταξινόμηση Πολλαπλών Κλάσεων Perceptron

- Έστω $y_1,\ldots, y_n$ πρότυπα από c κλάσεις, γραμμικά διαχωρίσιμα. Έστω L_k μία γραμμική μηχανή $\boldsymbol{a}_1(k),\ldots, \boldsymbol{a}_c(k)$. Θέλουμε να κατασκευάσουμε μία ακολουθία γρ. μηχ. $L_1,\ldots,L_k,\ldots$ που να συγκλίνει σε μία διαχωριστική μηχανή L.

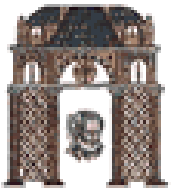
- Έστω $\boldsymbol{y}^k$ το k-στό δείγμα που ζητά διόρθωση (σωστή ταξινόμηση). Αν $\boldsymbol{y}^k \sim \omega_i$, σημαίνει ότι υπάρχει τουλάχιστον ένα $j \neq i$ για το οποίο $\boldsymbol{a}_i^t(k)\,\boldsymbol{y}^k < \boldsymbol{a}_j^t(k)\,\boldsymbol{y}^k$.

- Ο κανόνας διόρθωσης του L_k (perceptron με σταθερό μοναδιαίο βήμα) λέει:

$$\boldsymbol{\alpha}(k+1)=\boldsymbol{\alpha}(k)+\boldsymbol{\eta}_{ij}^k \quad \text{όπου} \quad \boldsymbol{\alpha}^t(k)\boldsymbol{\eta}_{ij}^k \le 0 \quad \text{με} \quad \boldsymbol{\alpha}(k)=\begin{bmatrix}\mathbf{a}_1(k)\\ \vdots \\ \mathbf{a}_c(k)\end{bmatrix} \quad \text{και} \quad \boldsymbol{\eta}_{ij}^k=\begin{bmatrix}\vdots \\ \mathbf{y}^k \\ \vdots \\ -\mathbf{y}^k \\ \vdots\end{bmatrix} \begin{matrix}\\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{matrix}$$

Δηλαδή:

$$\begin{aligned}\mathbf{a}_i(k+1)&=\mathbf{a}_i(k)+\mathbf{y}^k\\ \mathbf{a}_j(k+1)&=\mathbf{a}_j(k)-\mathbf{y}^k\\ \mathbf{a}_l(k+1)&=\mathbf{a}_l(k) \quad l\neq i \quad \text{και} \quad l\neq j\end{aligned}$$



Παράδειγμα με δομή Kesler

```
% Example Kesler CC
% It uses the Statistical Toolbox for Matlab
% http://cmp.felk.cvut.cz/cmp/software/stprtool/
% modified by chamzas
close('all'); clear;

m=[1 1 ; 5 10 ; 10 4; 10 10]';
[l,c]=size(m);
S1=[0.8 0.2; 0.2 0.1]; S2=[0.8 0.2; 0.2 0.8];
S3=[1.0 0.25; 0.25 1.8]; S4=[0.8 0.3; 0.3 1.8];
S(:,:,1)=S1; S(:,:,2)=S2; S(:,:,3)=S3; S(:,:,4)=S4;
P=[1/4 1/4 1/4 1/4];

% 1. Generate X1 the training set
N1=10000;
randn('seed',0)
[X,y1]=generate_gauss_classes(m,S,P,N1);
[l,N1]=size(X);
X1=[X; ones(1,N1)];

%--------- Multi class perceptron with KESLER structure -----------
data.X=X;  data.y=y1;  %puts data in format for STPRtool
options.tmax=100000; % max number of iterations
modelKESLER = mperceptron( data, options );
figure; ppatterns( data ); pboundary( modelKESLER );
% calculates the error on the trainning set
for i=1:l
    w_allKESLER(i,:)=modelKESLER.W(i,:);
end
w_allKESLER(l+1,:)=modelKESLER.b';
[vali,class_est]=max(w_allKESLER'*X1);
errKESLER=sum(class_est~=y1)/N1
```

Ένα χρήσιμο toolbook με υλοποιήσεις προγραμμάτων Στατιστικής και Αναγνώρισης Προτύπων είναι και το http://cmp.felk.cvut.cz/cmp/software/stprtool/

Τα δεδομένα δίνονται με την δομή data η οποία έχει το data.X με τα πρότυπα (patterns) και το data.y με τις κλάσεις (targets)

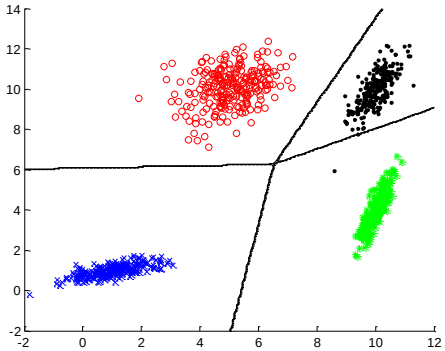
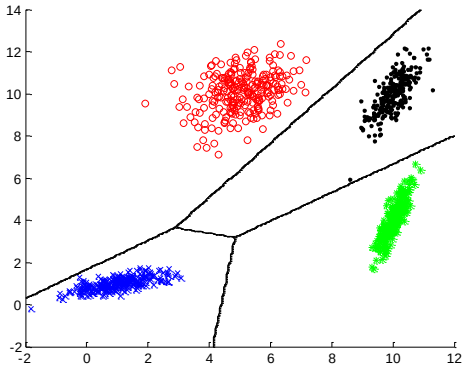
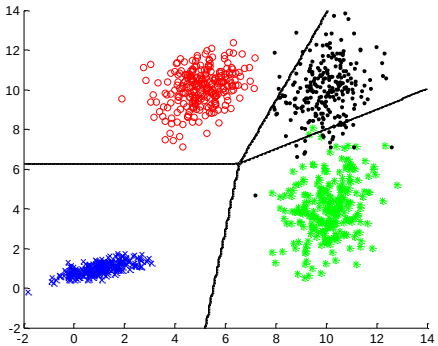
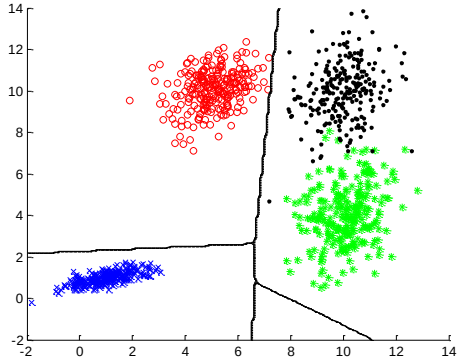
Στο eclass έχουν προστεθεί στο software του Πικράκη-Θεοδωρίδη και τα απαιτούμενα αρχεία από το παραπάνω toolbook καθώς και τα παραδείγματα των διαφανειών του μαθήματος.

Linear Discriminants for multi class problems
MSE versus Perceptron (Kesler)

Ο αλγόριθμος τύπου Perceptron με δομή Kelser χρειάζεται τα δεδομένα να είναι γραμμικά διαχωρίσιμα. Εάν δεν είναι τότε ο αλγόριθμος δεν συγκλίνει και η λύση που βρίσκει δεν είναι ικανοποιητική.

Οι αλγόριθμοι MSE βρίσκουν λύση σε κάθε περίπτωση, αλλά ακόμα και εάν οι κλάσεις είναι διαχωρίσιμες η λύση που βρίσκουν δεν είναι απαραίτητα βέλτιστη

Η προσέγγιση Ho-Kashyap προσπαθεί να συνδυάσει λύσεις Perceptron με MSE

	MSE	**KESLER**
Γραμμικά διαχωρίσιμες κλάσεις		
	err= 0.003 for the training set err= 0.0013 for a testing set	err= **0** for the training set err= 0.0007 for a testing set
ΜΗ Γραμμικά διαχωρίσιμες κλάσεις		
	err= 0.0300 for the training set err= 0.0250 for a testing set	err= 0.2500 for the training set err= 0.2520 for a testing set