Κεφάλαιο 11

Πρόβλημα 11

11.1 A. Sketching contour plot of MSE performance index

11.1.1 Εύρεση MSE index

Since the vectors are equiprobable, $Prob_1 = Prob_2 = \frac{1}{2} = 0.5$.

$$F(x) = c - 2x^T h + x^T R x \tag{11.1}$$

όπου

$$c = E[t^2], h = E[tz], R = E[zz^T] \tag{11.2}$$

c = E[t^2]

h = E[tz] = $0.5\cdot(-1) \cdot \begin{vmatrix} 1 \\ 2 \end{vmatrix} + 0.5 \cdot 1 \cdot \begin{vmatrix} -2 \\ 1 \end{vmatrix} = \begin{vmatrix} -1.5 \\ 0.5 \end{vmatrix}$

R = E[zz^T] =$p_1 \cdot p_1{}^T \cdot Prob_1 + p_2 \cdot p_2{}^T \cdot Prob_2 = \begin{vmatrix} 1 & 2 \\ 2 & 4 \end{vmatrix} \cdot 0.5 + \begin{vmatrix} 4 & -2 \\ -2 & 1 \end{vmatrix} \cdot 0.5 = \begin{vmatrix} 2.5 & 0 \\ 0 & 2.5 \end{vmatrix}$

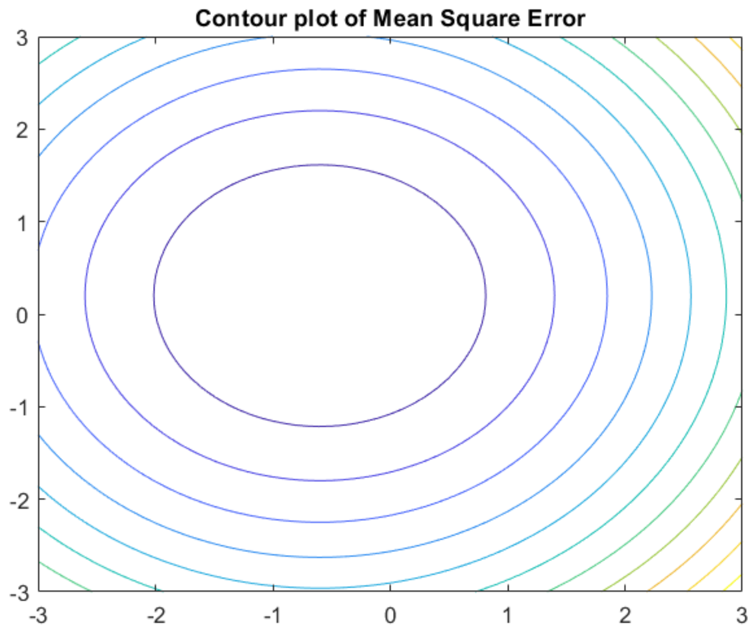
Επομένως,

$$F(x) = c - 2x^T h + x^T R x = 1 - 2 \begin{vmatrix} w_{1,1} & w_{1,2} \end{vmatrix} \begin{vmatrix} -1.5 \\ 0.5 \end{vmatrix} + \begin{vmatrix} w_{1,1} & w_{12} \end{vmatrix} \begin{vmatrix} 2.5 & 0 \\ 0 & 2.5 \end{vmatrix} \begin{vmatrix} w_{1,1} \\ w_{1,2} \end{vmatrix} = 1 + 3w_1 - w_2 + 2.5(w_1^2 + w_2^2) \tag{11.3}$$

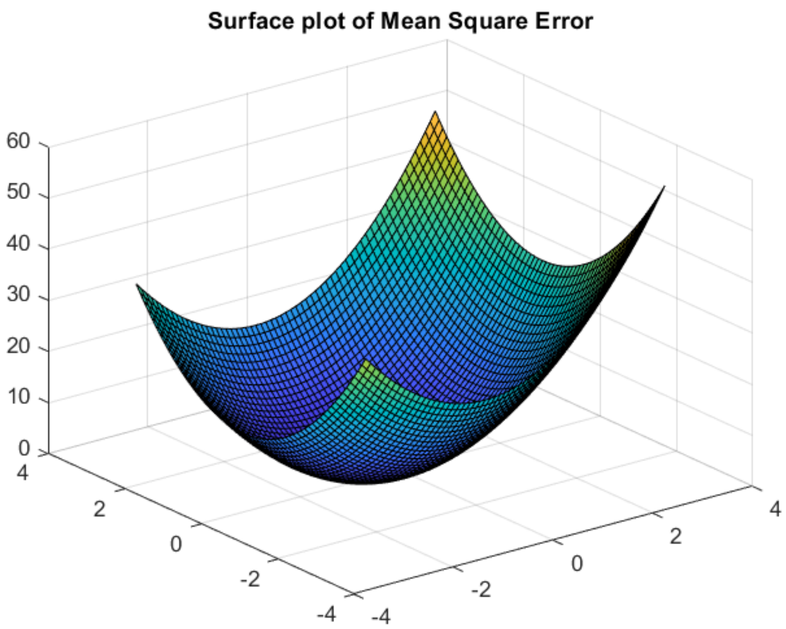
11.1.2 MATLAB Code

```
% CE418: Neuro−fuzzy Computing
%
%    Evangelos Stamos
%    02338
%    estamos@e−ce.uth.gr

% Problem−11 | A
%
% Sketch the contour plot of the mean square error performance index
%

```

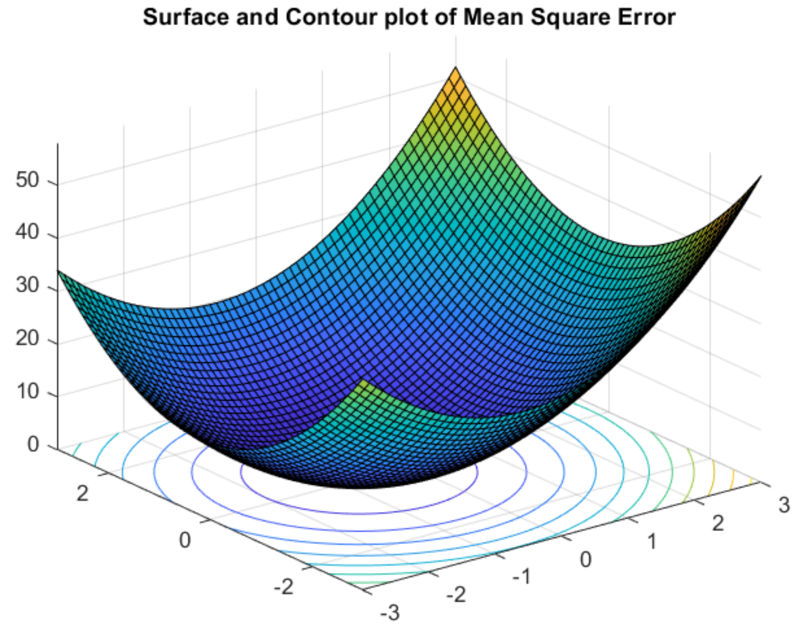
```
clear
[W1,W2] = meshgrid(-3 : .1 : 3);
F = 1 + 3 * W1 - W2 + 2.5 * (W1.^2 + W2.^2);
surf(W1,W2,F)
title('Surface plot of Mean Square Error');
```



Εικόνα 11.1: *Contour plot of Mean Square Error Performance Index*



Εικόνα 11.2: *Surface plot of Mean Square Error Performance Index*



Εικόνα 11.3: *Surface and Contour plot of Mean Square Error Performance Index*

11.2 B. Sketch the optimal decision boundary

We need to find the optimal weights :

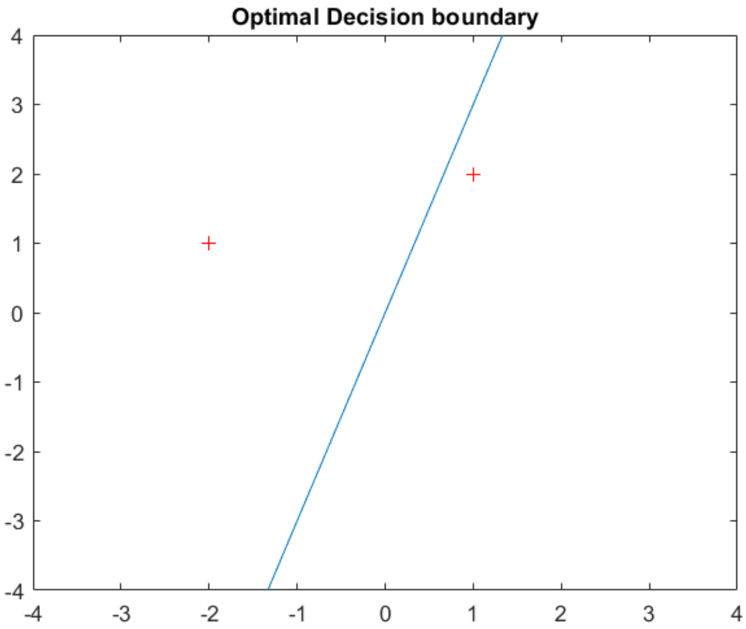
$$w^* = R^{-1} \cdot h = \begin{vmatrix} 2.5 & 0 \\ 0 & 2.5 \end{vmatrix}^{-} \cdot \begin{vmatrix} -1.5 \\ 0.5 \end{vmatrix} = \begin{vmatrix} 0.4 & 0 \\ 0 & 0.4 \end{vmatrix} \cdot \begin{vmatrix} -1.5 \\ 0.5 \end{vmatrix} = \begin{vmatrix} -0.6 \\ 0.2 \end{vmatrix} \tag{11.4}$$

So optimal weights are $w_1 = -0.6$ and $w_2 = 0.2$.

It is clear, from image 11.4 , that optimal decision boundary separates the patterns into the appropriate categories .

11.2.1 MATLAB Code

```
% CE418:  Neuro-fuzzy  Computing
%
%    Evangelos  Stamos
%    02338
%    estamos@e-ce.uth.gr

% Problem-11  |  B
%
% Training  an  ADALINE  network  without  a  bias
%
% The  vectors  are  equiprobable
%
% Sketch  the  optimal  decision  boundary.
```



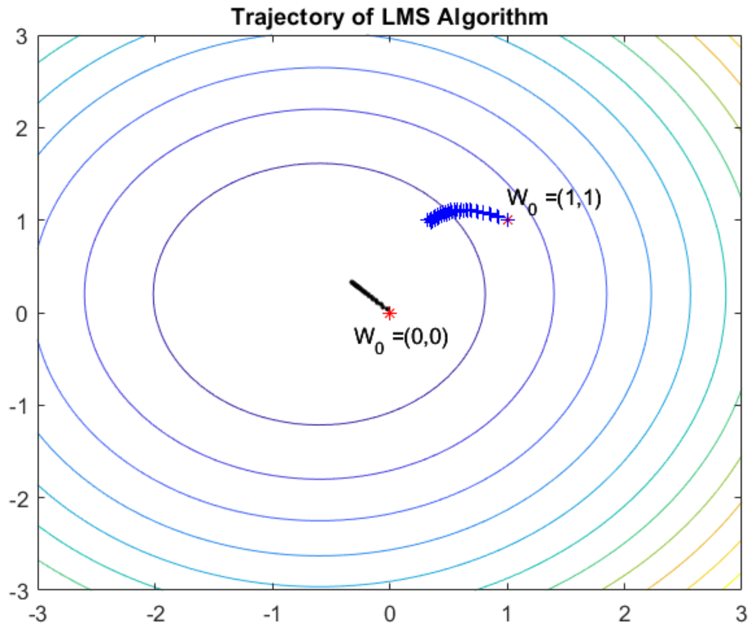
Εικόνα 11.4: *Optimal Decision Boundary*

```
%

clear

% Input data and set parameters
P = [1 -2;2 1];            % reference patterns
W = [-0.6 0.2];            % Optimal weights
figure;
plot(P(1,1),P(2,1),'r+');
hold on;
plot(P(1,2),P(2,2),'r+');

%Decision Boundary
x = -4 : .1 : 4;
y =(-W(1)*x )/W(2);
plot(x,y);
axis([-4 4 -4 4]);
title('Optimal Decision boundary');
hold off;
```

11.3 B. Sketch the trajectory of the LMS algorithm

11.3.1 MATLAB Code

```
% CE418: Neuro-fuzzy Computing
```



Εικόνα 11.5: *Trajectory of the LMS algorithm on contour plot*

```
%
%    Evangelos Stamos
%    02338
%    estamos@e-ce.uth.gr

% Problem-11 | C
%
% Training an ADALINE network without a bias
%
% The vectors are equiprobable
%
% Sketch the trajectory of the LMS algorithm on your contour plot
% Assuming a very small learning rate | alpha = 0.01
% Starting with initial weights W(0) = [0 1]

clear
[W1,W2] = meshgrid(-3 : .1 : 3);
F = 1 + 3 * W1 - W2 + 2.5 * (W1.^2 + W2.^2);
contour(W1,W2,F)
title('Trajectory of LMS Algorithm');
hold on;

% Input data and set parameters
P = [1 -2;-2 1];     % reference patterns
```

```
T = [-1  1];           % targets
alpha = 0.01;          % learning rate
W = [0;1];             % weights

% Training the ADALINE network without a bias

% LMS Algorithm Implementation

% Taking 40 = 20 x 2 steps of the algorithm for alpha = 0.01
%Initialize data

W1 = [0;0];
W2 = [1;1];
for k = 1 : 2
    if (k == 1)
        W = W1;
    else
        W = W2;
    end
    plot(W(1), W(2), 'r*')
    text(-0.3,-0.3,'W_0 =(0,0)');
    text(1,1.2,'W_0 =(1,1)');

        % Training the network
        for step = 1 : 20
        for i = 1 : 2
        a = purelin(W' * P(:,i));
        e = T(i) - a;
        W = W + 2 * alpha * e * P(:,i);
          if (k == 1)
              plot(W(1), W(2), 'k.')
              W1 = W;
          else
              plot(W(1), W(2), 'b+')
              W2 = W;
          end
        end
        end
end
W1
W2
hold off;
```