Κεφάλαιο 10

Πρόβλημα 10

10.1 A. Find MSE and maximum stable learning rate

10.1.1 Find MSE index

Since the two reference patterns occur with equal probability, $Prob_1 = Prob_2 = \frac{1}{2} = 0.5$.

$$F(x) = c - 2x^T h + x^T Rx \tag{10.1}$$

όπου

$$c = E[t^2], h = E[tz], R = E[zz^T] \tag{10.2}$$

c = E[t^2] = $(1)^2 \cdot 0.5 + (-1)^2 \cdot 0.5 = 1$

h = E[tz] = $0.5 \cdot 1 \cdot \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix} + 0.5 \cdot (-1) \cdot \begin{vmatrix} 1 \\ -1 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix}$

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

Επομένως,

$$F(x) = c - 2x^T h + x^T Rx = 1 - 2\begin{vmatrix} w_{1,1} & w_{1,2} & b_1 \end{vmatrix}\begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix} + \begin{vmatrix} w_{1,1} & w_{12} & b_1 \end{vmatrix}\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}\begin{vmatrix} w_{1,1} \\ w_{1,2} \\ b_1 \end{vmatrix} = 1 - 2w_{1,2} + (w_{1,1} + b_1)^2 + w_{1,2}^2 \tag{10.3}$$

10.1.2 Maximum stable learning rate

In order to find the maximum stable learning rate for the LMS algorithm, we need to compute the eigenvalues of R. For R = $\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}$, according to MATLAB script, eigenvalues are $\lambda_1 = 0$, $\lambda_2 = 2$ and $\lambda_3 = 4$. So $\lambda_{max} = 4$.

The maximum stable learning rate should satisfy :

$$a < \frac{2}{\lambda_{max}} = \frac{2}{4} = 0.5 \tag{10.4}$$

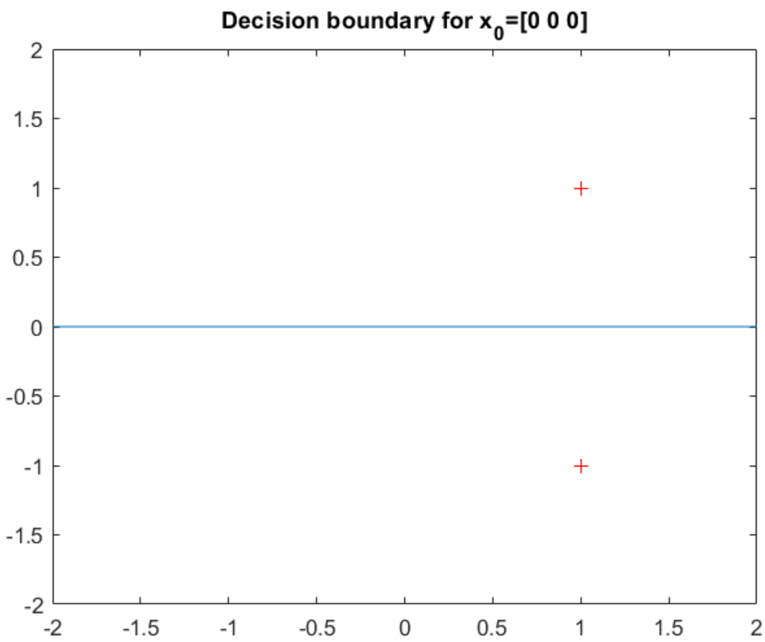
So maximum stable learning rate is $a_{max} = 0.5$.

10.1.3 MATLAB Code

```
R=2*[1 0 1;0 1 0;1 0 1];
eig (R)
```

10.2 B. LMS algorithm implementation on MATLAB

Taking 40 steps of the algorithm for a stable learning rate $a = 0.1$ and using the zero vector as the initial guess. Output is $w_{1,1} = 0$, $w_{1,2} = 1$ and $b_1 = -9.3370e-07$.



Εικόνα 10.1: *Decision boundary for* $x_0 = [000]$

10.2.1 MATLAB Code

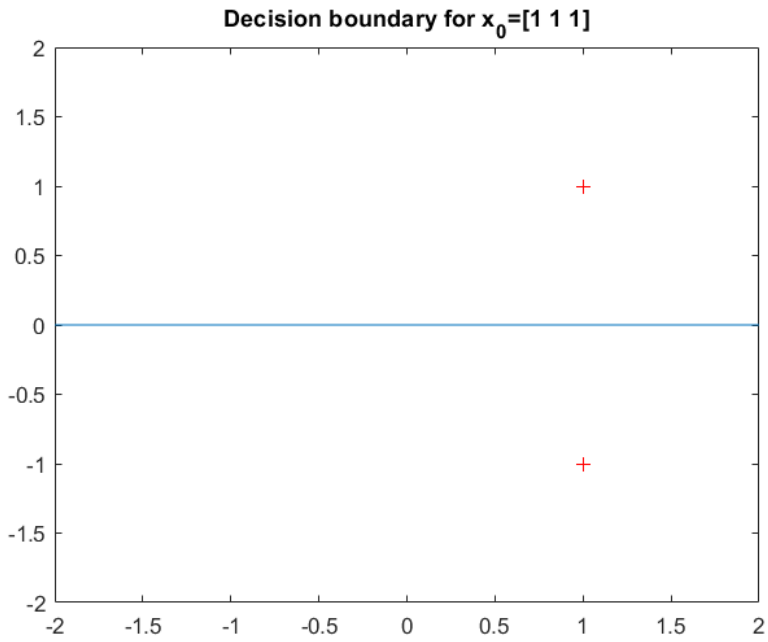
```
% CE418: Neuro-fuzzy Computing
%
%    Evangelos Stamos
%    02338
%    estamos@e-ce.uth.gr

% Problem-10 | B
%
```

```
% Training an ADALINE network with a bias
%
% Assuming that two reference patterns occur with equal probability
%
% Need to find weights[] and bias

clear

% Input data and set parameters
P = [1 1;1 -1];      % reference patterns
T = [1 -1];          % targets
alpha = 0.1;          % learning rate
W = [0;0];           % weights | Using the zero vector as the initial
    guess
b = 0;               % bias

% Training the ADALINE network with a bias

% LMS Algorithm Implementation

% Taking 40 = 20 x 2 steps of the algorithm for alpha = 0.1
for step = 1 : 20
        for i = 1 : 2
        a = purelin(W' * P(:,i) + b);
        e = T(i) - a;
      W = W + 2 * alpha * e * P(:,i);
      b = b + 2 * alpha * e;
   end
end
W
b

% Displaying reference patterns in graph
figure;
plot(P(1,1),P(2,1),'r+');    % Displaying p1
hold on;
plot(P(1,2),P(2,2),'r+');    % Displaying p2

% Sketching the final decision boundary
x = -2 : .1 : 2;
y =(-W(1,1)*x - b)/W(2,1);
plot(x,y);
axis([-2 2 -2 2]);
```

```
title('Decision boundary for x_0=[0 0 0]');
hold off;
```

10.3 C. Sketch the final decision boundary

Taking 40 steps of the algorithm for a stable learning rate $a = 0.1$ after setting the initial values of all parameters to 1. Output is $w_{1,1} = 0$, $w_{1,2} = 1$ and $b_1 = -2.7863e-07$.



Εικόνα 10.2: *Decision boundary for* $x_0 = [111]$

10.3.1 MATLAB Code

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

% Problem-10 | C
%
% Training an ADALINE network with a bias
%
% Assuming that two reference patterns occur with equal probability
%
% Need to find weights[] and bias

% Setting the initial values of all parameters to 1

clear

```

```
% Input data and set parameters

P = [1  1;1  -1];       % reference patterns
T = [1  -1];            % targets
alpha = 0.1;             % learning rate
W = [1;1];              % weights | Setting to 1
b = 1;                  % bias | Setting to 1

% Training the ADALINE network with a bias

% LMS Algorithm Implementation

% Taking 40 = 20 x 2 steps of the algorithm for alpha = 0.1
for step = 1 : 20
        for i = 1 : 2
        a = purelin(W' * P(:,i) + b);
        e = T(i) - a;
      W = W + 2 * alpha * e * P(:,i);
      b = b + 2 * alpha * e;
   end
end
W
b

% Displaying reference patterns in graph
figure;
plot(P(1,1),P(2,1),'r+');   % Displaying p1
hold on;
plot(P(1,2),P(2,2),'r+');   % Displaying p2

% Sketching the final decision boundary
x = -2 : .1 : 2;
y =(-W(1,1)*x - b)/W(2,1);
plot(x,y);
axis([-2 2 -2 2]);
title('Decision boundary for x_0=[0 0 0]');
hold off;
```

Πίνακας 10.1: *B and C final parameters*

Question	$w_{1,1}$	$w_{1,2}$	b_1
B	0	1	-9.3370e-07
C	0	1	-2.7863e-07

10.4 D. Compare the final parameters and the decision boundaries from (B) and (C)

The decision boundaries from (B) and (C) are the same, final parameters $w_{1,1}$ and $w_{1,2}$ are also the same, while b_1 is different. Since LMS algorithm tends to find a decision boundary as far from the patterns as possible, the resulting biases for each initial point are close to zero, so that the boundary falls almost halfway between the two vectors. Also, since the Hessian matrix has zero eigenvalue, there would exist a weak minimum (note that If the correlation matrix has some zero eigenvalues, the performance index will either have a weak minimum or no minimum depending on the vector d=-2h) .