Μηχανική Μάθηση

1ο Σετ Ασκήσεων

ΤΜΗΜΑ: Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΟΝΟΜΑ: ΚΩΝΣΤΑΝΤΙΝΟΣ
ΕΠΩΝΥΜΟ: ΛΑΓΑΡΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1053705
ΕΤΟΣ ΦΟΙΤΗΣΗΣ: 6ο

Πρόβλημα 1

Α) Όπως έχουμε δείξει το βέλτιστο τεστ κατά Bayes(τεστ του λόγου πιθανοφάνιας) *που ελαχιστοποιεί την πιθανότητα σφάλματος είναι* :
για κάθε υλοποίηση του τυχαίου διανύσματος X=[χ1,χ2] , να πηγαίνουμε να υπολογίζουμε τον λόγο πιθανοφάνιας $\frac{\text{f1(x1,x2)}}{\text{f0(x1,x2)}}$ και να το συγκρίνουμε με τον λόγο $\frac{P(\text{H0})}{\text{P(H1)}}$
Εδώ οι εκ των προτέρων πιθανότητες, $P(H_0)$ και $P(H_1)$, είναι ίσες μεταξύ και ίσες με την μονάδα.

Επομένως για δεδομένη τιμή του X = [χ1,χ2]
όταν $\frac{\text{f1(x1,x2)}}{\text{f0(x1,x2)}} > 1$, ψηφίζω υπέρ του H_1 (δηλαδή ότι το X προέρχεται από την υπόθεση H1)

όταν $\frac{\text{f1(x1,x2)}}{\text{f0(x1,x2)}} < 1$, ψηφίζω υπέρ του H_0

ενώ όταν $\frac{\text{f1(x1,x2)}}{\text{f0(x1,x2)}} = 1$, τότε αποφασίζω στην τύχη .

Στο συγκεκριμένο παράδειγμα οι πυκνότητες πιθανότητας είναι :

$f_0(x_1,x_2) = f_0(x_1)*f_0(x_2)$, (αφού x1 και x2 είναι ανεξάρτητα μεταξύ τους)

Άρα $f_1(x_1,x_2) = \frac{e^{-\frac{1}{2}(x_1-1)^2}+e^{-\frac{1}{2}(x_1+1)^2}}{2\sqrt{2\pi}} \cdot \frac{e^{-\frac{1}{2}(x_2-1)^2}+e^{-\frac{1}{2}(x_2+1)^2}}{2\sqrt{2\pi}}$

και $f_0(x_1,x_2) = \frac{e^{-\frac{x_1{}^2}{2}}}{\sqrt{2\pi}} \cdot \frac{e^{-\frac{x_2{}^2}{2}}}{\sqrt{2\pi}}$

Β)
Δημιουργούμε 10^6 ζευγάρια (χ1,χ2) από την κάθε υπόθεση και εκτελούμε το τεστ κατά Bayes.

Κάνοντας στην προσημείωση στην Matlab :

Το ποσοστό των φορών που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H0 = 0.4235
Το ποσοστό των φορών που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H1 = 0.2813

Και η **αθροιστική πιθανότητα σφάλματος = 0.3524 , κατά Bayes**

Αυτή είναι η πιθανότητα σφάλματος του τρόπου απόφασης κατά Bayes που προσέγγισε η προσομοίωση μας (αρκετά καλή προσέγγιση αφού τα δεδομένα μας είναι 10^6 για κάθε υπόθεση)

Ξέρουμε πως αυτός ο τρόπος απόφασης είναι και ο καλύτερος δυνατός.

Γ)
Ακολουθώντας πιστά το υλικό που μας δόθηκε (υπολογισμός και ανανέωση παραμέτρων του δικτύου, κανονικοποίηση Adams) υλοποιήθηκαν νευρονικά δίκτυα διαστάσεων
2 x 20 x 1 και εκπαιδεύτηκαν πάνω σε 200 ζευγάρια (χ1,χ2) από κάθε υπόθεση.

Τα νευρονικά δίκτυα εκπαιδεύτηκαν με την μέθοδο Cross Entropy , όπου $\varphi(z) = -\log(1-z)$ και $\psi(z) = -\log(z)$ καθώς και με την μέθοδο Exponential , όπου $\varphi(z) = e^{0.5z}$ $\psi(z) = e^{-0.5z}$.

Το κόστος του προβλήματος βελτιστοποίησης που θέλουμε να ελαχιστοποιήσουμε είναι το

$$J(u(x,\theta)) = \Phi(u(x_0^1, x_0^2, \theta)) + \Psi(u(x_1^1, x_1^2, \theta))$$

Και ο αλγόριθμος που χρησιμοποιούμε για να βρούμε τις παραμέτρους οι οποίες ελαχιστοποιούνε την συνάρτηση κόστους μας είναι ο stochastic gradient descent :

$$\theta t = \theta t-1 - \mu * \nabla_\theta J(\theta t)$$

Επίσης κατά την εκπαίδευση χρησιμοποιήθηκε και η κανονικοποίηση κατά Adams η οποία υπολογίζει την ισχύ κάθε στοιχείου της κλίσης, και κανονικοποιεί τα στοιχεία της κλίσης με την αντίστοιχη ισχύ τους :

$$[\mathbf{A}_k \mathcal{B}_k]_t = [\mathbf{A}_k \mathcal{B}_k]_{t-1} - \mu \nabla_{[\mathbf{A}_k \mathcal{B}_k]_{t-1}} \phi(Y_t) ./ \sqrt{c + \mathbf{P}_{[\mathbf{A}_k \mathcal{B}_k]}(t)}$$

Όπου:

$$\mathbf{P}_{[\mathbf{A}_k \mathcal{B}_k]}(t) = (1-\lambda)\mathbf{P}_{[\mathbf{A}_k \mathcal{B}_k]}(t-1) + \lambda \left(\nabla_{[\mathbf{A}_k \mathcal{B}_k]_{t-1}} \phi(Y_t)\right)^{(.2)}$$

Και αρχικοποιούμε με :

$$\mathbf{P}_{[\mathbf{A}_k \mathcal{B}_k]}(1) = \left(\nabla_{[\mathbf{A}_k \mathcal{B}_k]_0} \phi(Y_1)\right)^{(.2)}$$

με $\lambda << 1$, και c το θέτουμε ένα μικρό $\varepsilon > 0$ ώστε να αποφύγουμε διαίρεση με το 0.
Οπότε το c το έθεσα ίσο με 10^{-310}.
($A_κ$ και $B_κ$ είναι τα βάροι και offset του δικτύου δηλαδή οι παράμετροι)

Αν δεν εφαρμόσουμε την κανονικοποίηση Adams η κλίση της συνάρτησης κόστους ως προς τις παραμέτρους θα εμφάνιζε αρκετά μεγάλη διαφορά για τις διαφορετικές παραμέτρους του δικτύου, κάτι το οποίο σημαίνει ότι το νευρονικό δίκτυο διορθώνει κατά την εκπαίδευση του κάποιες παραμέτρους περισσότερο από άλλες.
Έχει φανεί στην πράξη ότι κανονικοποιώντας τις παραγώγους έχουμε πιο ομαλές λύσεις και γρηγορότερη σύγκλιση.

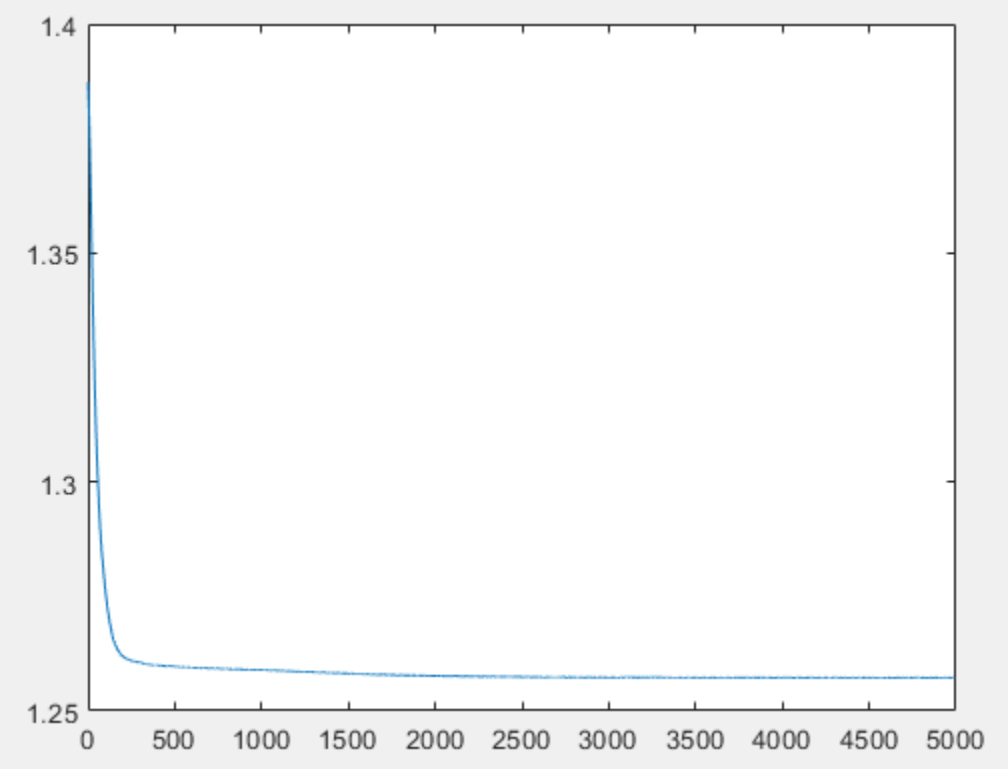
Η εκπαίδευση έγινε μέσα σε 5000 εποχές με τις παραμέτρους learning rate = $2*10^{-4}$
Και smoothing factor (1-λ) = 0.99
(παρόμοιες τιμές των μ και λ είδα να χρησιμοποιούνται και στο paper Training Neural Networks for Likelihood/Density Ratio Estimation George V. Moustakides and Kalliopi Basioti 2019 . Έκανα δοκιμές για διάφορες τιμές αλλά αυτές μου φάνηκαν να βγάζουν αρκετά ικανοποιητικά αποτελέσματα)

Η διαδικασία της εκπαίδευσης έγινε 10 φορές ώστε, λόγο των διαφορετικών τυχαίων αρχικών παραμέτρων του δικτύου στις οποίες φαίνεται να είναι πολύ ευαίσθητα τα νευρονικά δίκτυα, να επιλεχθούν οι παράμετροι που δίνουν το μικρότερο μέσο κόστος στο τέλος της εκπαίδευσης.

Τέλος, χρησιμοποιήθηκαν αυτά τα νευρονικά δίκτυα για να αποφασίσουν <u>αν τα ίδια δεδομένα</u> που εξέτασε και η μέθοδος του Bayes προέρχονται από την υπόθεση H0 ή H1 και υπολογίστηκε το ποσοστό σφάλματος που κάνουν.

Με την μέθοδο Cross Entropy :

Το μέσο κόστος ανά κάθε εποχή :



Εδώ η έξοδος του δικτύου είναι $\frac{r(x_1,x_2)}{1+r(x_1,x_2)}$, όπου $r(x_1,x_2) = \frac{f_1(x_1,x_2)}{f_0(x_1,x_2)}$

που είναι η εκ των υστέρων πιθανότητα.

Οπότε το ισοδύναμο τέστ τώρα είναι :

Αν έξοδος δικτύου > 0.5 , αποφασίζω υπέρ του H1
Αν έξοδος δικτύου < 0.5 , αποφασίζω υπέρ του H0
Αν έξοδος δικτύου = 0.5 , αποφασίζω στην τύχη

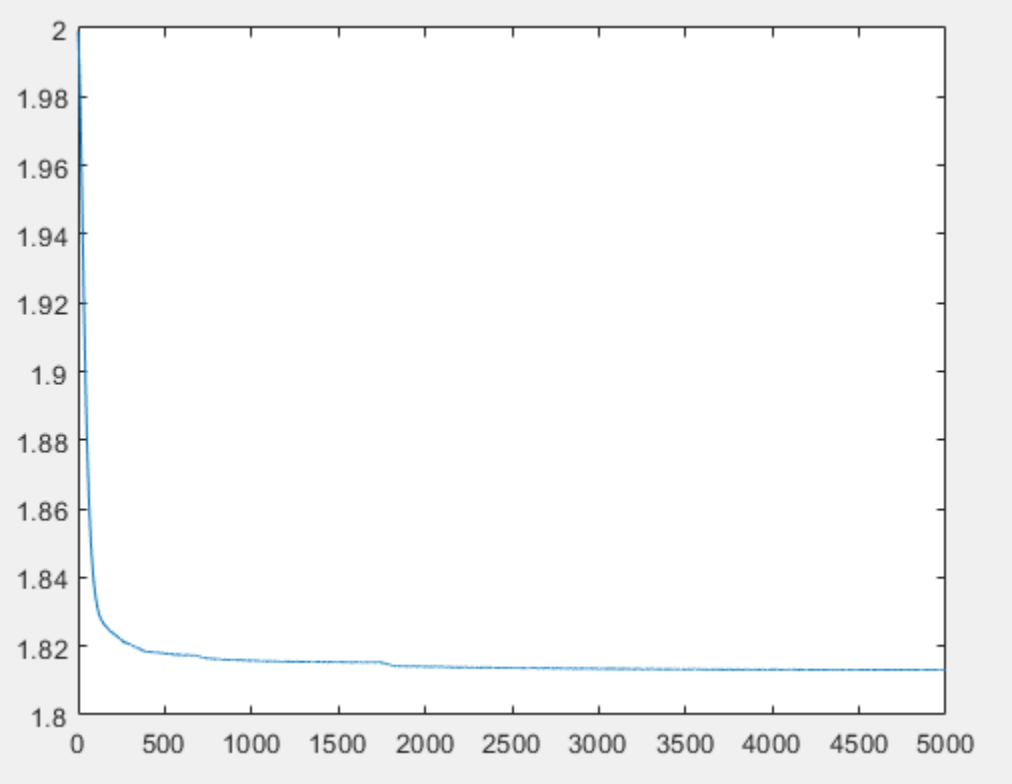
Τρέχοντας το παραπάνω τεστ πάνω στα $2*10^6$ δεδομένα που δημιουργήσαμε πριν
Παίρνουμε :

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H0 = 0.3024
Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H1 = 0.4336

Και η **αθροιστική πιθανότητα σφάλματος = 0.3680 , CrossEntropy**

Με την μέθοδο Exponential :

Το μέσο κόστος ανά κάθε εποχή :



Εδώ η έξοδος του δικτύου είναι $\log\left(r(x_1, x_2)\right)$, όπου $r(x_1, x_2) = \frac{f_1(x_1,x_2)}{f_0(x_1,x_2)}$
δηλαδή είναι ο λογάριθμος του λόγου πιθανοφάνιας.

Ο συνάρτηση του λογαρίθμου είναι γνησίως αύξουσα συνάρτηση (και προφανώς log(1) = 0) οπότε το ισοδύναμο τέστ τώρα είναι :

Αν έξοδος δικτύου > 0 , αποφασίζω υπέρ του H1
Αν έξοδος δικτύου < 0 , αποφασίζω υπέρ του H0
Αν έξοδος δικτύου = 0 , αποφασίζω στην τύχη

Τρέχοντας και το παραπάνω τεστ πάνω στα $2*10^6$ δεδομένα που δημιουργήσαμε πριν Παίρνουμε :

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H0 = 0.3478
Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του H1 = 0.2813

Και η **αθροιστική πιθανότητα σφάλματος = 0.3589 , Exponential**

Άρα τελικά έχουμε :

Μέθοδος	Ποσοστό **εσφαλμένων** αποφάσεων υπέρ του Ho	Ποσοστό **εσφαλμένων** αποφάσεων υπέρ του H1	Αθροιστικό Ποσοστό σφάλματος
CrossEntropy	30.24%	43.36%	**36.80%**
Exponential	34.78%	37.02%	**35.89%**
Bayes	42.35%	28.13%	**35.24%**

Οι άλλες 2 μέθοδοι αν και αρκετά κοντά στον Bayes έχουν μεγαλύτερη πιθανότητα σφάλματος από αυτόν.

Όπως ήταν αναμενόμενο **ο Bayes έχει την μικρότερη πιθανότητα σφάλματος.**

Πρόβλημα 2

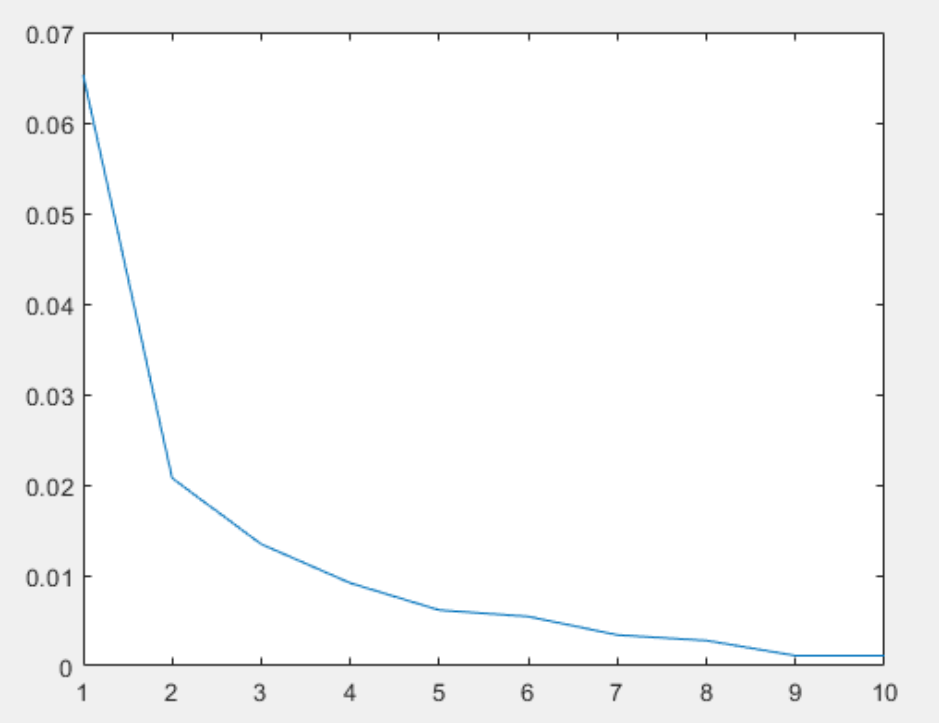
Ακολουθώντας την ίδια λογική του Προβλήματος 1, δημιουργήθηκαν 3 νευρονικά δίκτυα διαστάσεων 784 x 300 x 1 τα οποία εκπαιδεύτηκαν με τις μεθόδους Hinge , όπου $\varphi(z) = \max(1+z,0)$ και $\psi(z) = \max(1-z,0)$, Cross-entropy και Exponential αντίστοιχα.

Αυτά τα νευρονικά δίκτυα εκπαιδεύτηκαν πάνω σε δεδομένα της βιβλιοθήκης MNIST και πιο συγκεκριμένα σε εικόνες χειρόγραφων αριθμών (0 και 8) διάστασης 28 x 28 pixels (=784 είσοδοι).

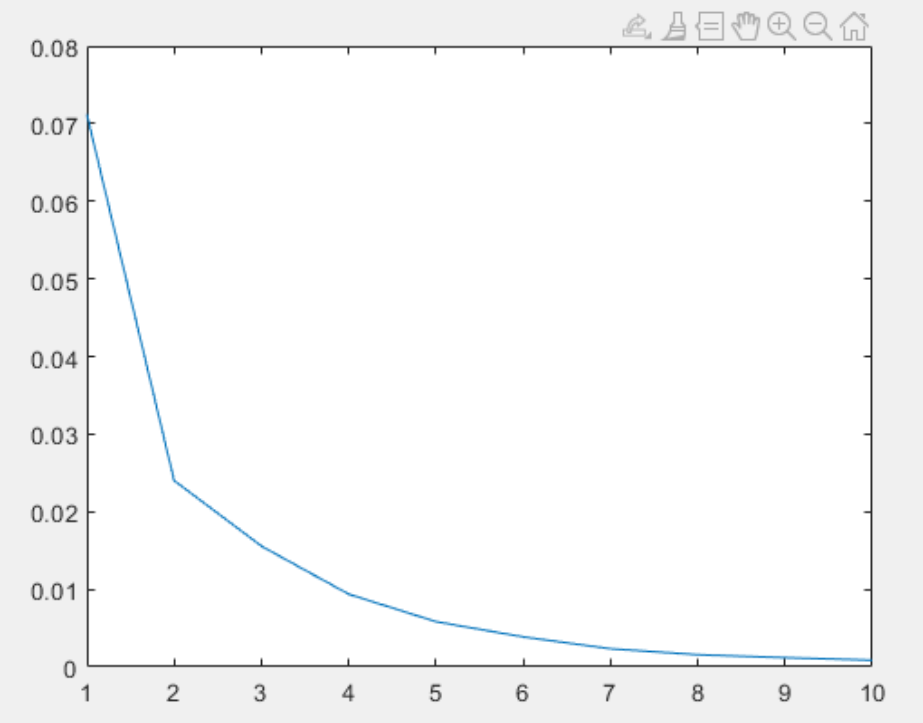
Για την εκπαίδευση χρησιμοποιήθηκαν 5500 εικόνες από την κάθε περίπτωση και ο υπολογισμός της πιθανότητας σφάλματος έγινε χρησιμοποιώντας 974 εικόνες από την κάθε περίπτωση. Επίσης έγιναν 10 εποχές για να συγκλίνουν οι αλγόριθμοι.

Εδώ ορίστηκαν $\mu = 2*10^{-4}$ και $\lambda = 10^{-5}$

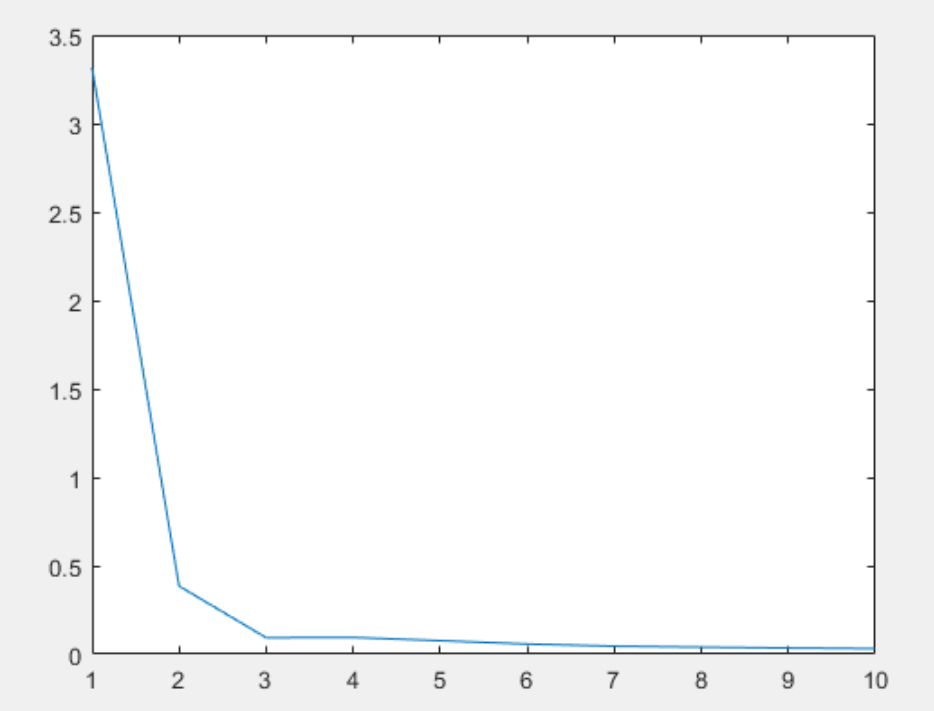
Το μέσο Κόστος της **Hinge** ανά εποχή :



Το μέσος Κόστος της **CrossEntropy** ανά εποχή :



Το μέσος Κόστος της **Exponential** ανά εποχή :



Αφού οι αλγόριθμοι σύγκλιναν τους έτρεξα πάνω στα 974 testing data από την κάθε περίπτωση (974 εικόνες “0” και 974 εικόνες “8”) και κατέληξα στα εξής ποσοστά σφάλματος :

Μέθοδος	Ποσοστό σφάλματος για τα 0	Ποσοστό σφάλματος για τα 8	Αθροιστικό ποσοστό σφάλματος
Cross Entropy	0.41%	0.62%	**0.51%**
Exponential	0.51%	0.41%	**0.46%**
Hinge	0.00%	0.51%	**0.26%**

Για την διεκπεραίωση της άσκησης γράφτηκε κώδικας σε Matlab

Για τις Ασκήσεις χρησιμοποιήθηκαν οι εξής **συναρτήσεις** :

```
function y = pdf0(x1,x2)
    y = (exp(-0.5*(x1^2 + x2^2)))/(2*pi);
end

function w = pdf1(x1,x2)
    w = (1/(8*pi))*(exp(-0.5*((x1+1)^2)) + exp(-0.5*((x1-1)^2)))*(exp(-
0.5*((x2+1)^2)) + exp(-0.5*((x2-1)^2)));
end

function y = Relu(u)
    y = max(0,u);
end
```

```
function y = Sigmoid(u)
    y = zeros(1,length(u));
    for i = 1:length(u)
        y(i) = 1 / ( 1 + exp(-u));
    end
    y = y';
end

function y = diff_Relu(u)
    y = zeros(1,length(u));
    for i = 1:length(u)
        if u(i) > 0
            y(i) = 1;
        elseif u(i) < 1
            y(i) = 0;
        end
    end
    y = y';
end

function y = diff_Sigmoid(u)
    y = zeros(1,length(u));
    for i = 1:length(u)
        y(i) = ( 1 - Sigmoid(u(i)) ) * Sigmoid(u(i));
    end
    y = y';
end

function y = Expon(u1,u2)
    y = exp(0.5*u1) + exp(-0.5*u2);
end

function y = Hinge(u1,u2)
    y = max(1+u1,0) + max(1-u2,0);
end


function y = CrossEntropy(u1,u2)
    y = -log(1-u1) -log(u2) ;
end

function y = diff_Phi_CrossEntropy(u)
    y = 1 / (1 - u);
end

function y = diff_Psi_CrossEntropy(u)
    y = - 1 / u;
end

function y = diff_Phi_expon(u)
    y = 0.5*exp(0.5*u);
end

function y = diff_Psi_expon(u)
```

```
    y = -0.5*exp(-0.5*u);
end

function y = Phi_CrossEntropy(u)
    y = - log(1 - u);
end

function y = Psi_CrossEntropy(u)
    y = - log(u);
end

function y = Phi_expon(u)
    y = exp(0.5*u);
end

function y = Psi_expon(u)
    y = exp(-0.5*u);
end

function y = Phi_Hinge(u)
    y = max(1+u,0);
end

function y = Psi_Hinge(u)
    y = max(1-u,0);
end
```

ΠΡΟΒΛΗΜΑ 1

α) Για τέστ του Bayes χρησιμοποιήθηκε ο παρακάτω κώδικας :

```
clc
clear all;
X0 = randn(10^6,2); % pdf0
X1 = zeros(10^6,2); % pdf1

for i = 1:10^6
    for j = 1:2
        if randn()>=0
            X1(i,j) = -1 + randn();
        else
            X1(i,j) = 1 + randn();
        end
    end
end
```

```
save("X0train","X0");
save("X1train","X1");

figure;
histogram(X0);
figure;
histogram(X1);

count1 = 0;
for i = 1:10^6
    if pdf1(X0(i,1),X0(i,2)) > pdf0(X0(i,1),X0(i,2))
        count1 = count1 +1;
    elseif pdf1(X0(i,1),X0(i,2)) == pdf0(X0(i,1),X0(i,2))
        if rand() > 0.5
            count1 = count1 + 1;
        end
    end
end

count2 = 0;
for i = 1:10^6
    if pdf1(X1(i,1),X1(i,2)) < pdf0(X1(i,1),X1(i,2))
        count2 = count2 +1;
    elseif pdf1(X1(i,1),X1(i,2)) == pdf0(X1(i,1),X1(i,2))
        if rand() > 0.5
            count2 = count2 + 1;
        end

    end
end

Bayes_test_error = 0.5*(count1/(10^6)) + 0.5*(count2/(10^6));
```

Γ) ***Για το τεστ με τα νευρονικα δίκτυα με <u>CrossEntropy</u> πάνω στα δεδομένα που κατηγοριοποίησε και ο Bayes :***

```
clc;
clear;
close all;

X0test = (load("X0train.mat","X0").X0)';
```

```
X1test = (load("X1train.mat","X1").X1)';

X0train = randn(2,200);  % train set made from Ho
X1train = zeros(2,200);   % train set made from H1

for i = 1:2
    for j = 1:200
        if randn()>=0
            X1train(i,j) = -1 + randn();
        else
            X1train(i,j) = 1 + randn();
        end
    end
end

% initialiazation of NN with dimention IxMxN "3-layers"

I = 2;
M = 20;
N = 1;

m = 2*0.0001; %learning rate
lamda = 0.00001; %smoothing factor moustakides has m= 2*1e-4 , and 1-lamda= 0.99
c = 1e-315;



% stohastic gradient descent -- learning

epochs = 5000;
Best_Cost = inf;

for d = 1:10
    A1 = (1/(M+I)) * randn(M,I);
    B1 = zeros(20,1);
    A2 = (1/(M*N)) * randn(N,M);
    B2 = 0;

    k=1;
    Cost = zeros(1,epochs*200);
    for j = 1:epochs % epochs
        for i = 1:200

        Z00 = X0train(:,i);
        Z10 = X1train(:,i);

        W01 = A1*Z00 + B1;
        Z01 = Relu(W01);
        W02 = A2*Z01 + B2;
        Y0 = Sigmoid(W02);

        W11 = A1*Z10 + B1;
        Z11 = Relu(W11);
        W12 = A2*Z11 + B2;
        Y1 = Sigmoid(W12); %
```

```
        Cost(k) = CrossEntropy(Y0,Y1);


        %derivative calculation
        u02 = diff_Phi_CrossEntropy(Y0);
        v02 = u02 .* diff_Sigmoid(W02);    %f2(x) = sigmoid , f2'(x) = diff_sigmoid
        u01 = (A2')*v02;
        v01 = u01 .* diff_Relu(W01);

        u12 = diff_Psi_CrossEntropy(Y1);
        v12 = u12 .* diff_Sigmoid(W12);
        u11 = (A2')*v12;
        v11 = u11 .* diff_Relu(W11);

        if (i == 1) && (j == 1)
            P2 = ((v02*[Z01' 1]) + (v12*[Z11' 1])) .^ 2;
        else
            P2 = (1 - lamda)*P2_old + lamda*power(((v02*[Z01' 1]) + (v12*[Z11'
1])),2);
        end

        P2_old = P2;

        if (i == 1) && (j == 1)
            P1 = ((v01*[Z00' 1]) + (v11*[Z10' 1])) .^ 2;
        else
            P1 = (1 - lamda)*P1_old + lamda*power(((v01*[Z00' 1]) + (v11*[Z10'
1])),2);
        end

        P1_old = P1;


        AB2 = [A2 B2] - m * (((v02*[Z01' 1]) + (v12*[Z11' 1])) ./ sqrt(c + P2)) ;
        A2 = AB2(1:20);
        B2 = AB2(1,21);

        AB1 = [A1 B1] - m * (((v01*[Z00' 1]) + (v11*[Z10' 1])) ./ sqrt(c + P1)) ;
        A1 = AB1(1:20,1:2);
        B1 = AB1(1:20,3);



        k=k+1;
        end
    end

    Cost_mean_itr = 0;
    for i = 1:200
        Cost_mean_itr = ( Cost_mean_itr + Cost(epochs*200 - 200 + i) );
    end

    Cost_mean_itr = Cost_mean_itr / 200 ;
```

```
    if Best_Cost > Cost_mean_itr
        Best_Cost = Cost_mean_itr;
        AA1 = A1;
        AA2 = A2;
        BB1 = B1;
        BB2 = B2;
        Final_Cost = Cost;
    end

end



% Cost_mean = zeros(1,epochs);
z=1;
Cost_mean = zeros(1,epochs);
for i = 0:200:length(Final_Cost) - 200
    for l = 0:199
        Cost_mean(z) = ( Cost_mean(z) + Final_Cost(i+l+1) );
    end
%     plot(Cost_mean(:)/200); drawnow
    z=z+1;
end



figure;
plot(Cost_mean/200);
figure;
plot(Cost);

% test of Neural Net
count1 = 0;
count2 = 0;

for i = 1:10^6
    Z00 = X0test(:,i);
    Z10 = X1test(:,i);

    W01 = AA1*Z00 + BB1;
    Z01 = Relu(W01);
    W02 = AA2*Z01 + BB2;
    Y0 = Sigmoid(W02);

    W11 = AA1*Z10 + BB1;
    Z11 = Relu(W11);
    W12 = AA2*Z11 + BB2;
    Y1 = Sigmoid(W12);

    if Y1 < 0.5
        count1 = count1 + 1;
    end

    if Y0 > 0.5
        count2 = count2 + 1;
```

```
    end

end

NN_CrossEntropy_error = 0.5*(count1/10^6) + 0.5*(count2/10^6);
```

Για το τεστ με τα νευρονικα δίκτυα με <u>Exponential</u> πάνω στα δεδομένα που κατηγοριοποίησε και ο Bayes :

```
clc;
clear;
close all;

X0test = (load("X0train.mat","X0").X0)';
X1test = (load("X1train.mat","X1").X1)';

X0train = randn(2,200);  % train set made from Ho
X1train = zeros(2,200);   % train set made from H1

for i = 1:2
    for j = 1:200
        if randn()>=0
            X1train(i,j) = -1 + randn();
        else
            X1train(i,j) = 1 + randn();
        end
    end
end

% initialiazation of NN with dimention IxMxN "3-layers"

I = 2;
M = 20;
N = 1;

m = 2*0.0001; %learning rate
lamda = 0.01; %smoothing factor moustakides has m= 2*1e-4 , and 1-lamda= 0.99
c = 1e-315;



% stohastic gradient descent -- learning

epochs = 5000;
Best_Cost = inf;

for d = 1:10
    A1 = (1/(M+I)) * randn(M,I);
    B1 = zeros(20,1);
    A2 = (1/(M*N)) * randn(N,M);
    B2 = 0;

    k=1;
```

```
    Cost = zeros(1,epochs*200);
    for j = 1:epochs % epochs
        for i = 1:200

        Z00 = X0train(:,i);
        Z10 = X1train(:,i);

        W01 = A1*Z00 + B1;
        Z01 = Relu(W01);
        W02 = A2*Z01 + B2;
        Y0 = W02;

        W11 = A1*Z10 + B1;
        Z11 = Relu(W11);
        W12 = A2*Z11 + B2;
        Y1 = W12; % no need for a non-linear activation function in output since its
the expontential

        Cost(k) = Expon(Y0,Y1);


        %derivative calculation
        u02 = diff_Phi_expon(Y0);
        v02 = u02 .* 1;    %f2(x) = x , f2'(x) = 1
        u01 = (A2')*v02;
        v01 = u01 .* diff_Relu(W01);

        u12 = diff_Psi_expon(Y1);
        v12 = u12 .* 1;
        u11 = (A2')*v12;
        v11 = u11 .* diff_Relu(W11);

        if (i == 1) && (j == 1)
            P2 = ((v02*[Z01' 1]) + (v12*[Z11' 1])) .^ 2;
        else
            P2 = (1 - lamda)*P2_old + lamda*power(((v02*[Z01' 1]) + (v12*[Z11'
1])),2);
        end

        P2_old = P2;

        if (i == 1) && (j == 1)
            P1 = ((v01*[Z00' 1]) + (v11*[Z10' 1])) .^ 2;
        else
            P1 = (1 - lamda)*P1_old + lamda*power(((v01*[Z00' 1]) + (v11*[Z10'
1])),2);
        end

        P1_old = P1;


        AB2 = [A2 B2] - m * (((v02*[Z01' 1]) + (v12*[Z11' 1])) ./ sqrt(c + P2)) ;
        A2 = AB2(1:20);
        B2 = AB2(1,21);
```

```
        AB1 = [A1 B1] - m * (((v01*[Z00' 1]) + (v11*[Z10' 1])) ./ sqrt(c + P1)) ;
        A1 = AB1(1:20,1:2);
        B1 = AB1(1:20,3);



        k=k+1;
        end
    end

    Cost_mean_itr = 0;
    for i = 1:200
        Cost_mean_itr = ( Cost_mean_itr + Cost(epochs*200 - 200 + i) );
    end

    Cost_mean_itr = Cost_mean_itr / 200 ;

    if Best_Cost > Cost_mean_itr
        Best_Cost = Cost_mean_itr;
        AA1 = A1;
        AA2 = A2;
        BB1 = B1;
        BB2 = B2;
        Final_Cost = Cost;
    end

end



z=1;
Cost_mean = zeros(1,epochs);
for i = 0:200:length(Final_Cost) - 200
    for l = 0:199
        Cost_mean(z) = ( Cost_mean(z) + Final_Cost(i+l+1) );
    end
    z=z+1;
end



figure;
plot(Cost_mean/200);
figure;
plot(Cost);

% test of Neural Net
count1 = 0;
count2 = 0;

for i = 1:10^6
    Z00 = X0test(:,i);
    Z10 = X1test(:,i);

    W01 = AA1*Z00 + BB1;
```

```
    Z01 = Relu(W01);
    W02 = AA2*Z01 + BB2;
    Y0 = W02;

    W11 = AA1*Z10 + BB1;
    Z11 = Relu(W11);
    W12 = AA2*Z11 + BB2;
    Y1 = W12;

    if Y1 < 0
        count1 = count1 + 1;
    end

    if Y0 > 0
        count2 = count2 + 1;
    end

end

NN_Expon_error = 0.5*(count1/10^6) + 0.5*(count2/10^6);
```

ΠΡΟΒΛΗΜΑ 2

Για τον Cross Entropy Classifier πάνω στα δεδομένα της MNIST

```
clc
clear;

% Mnist's whole data loading
train_data = load('mnist_train.csv');
test_data = load('mnist_test.csv');


%keeping only 0 and 8 train data ,both same number(5500)
X0_train = zeros(5500,784);
X8_train = zeros(5500,784);

count1 = 0;
count2 = 0;
for i = 1:60000
    if train_data(i,1) == 0 && count1 < 5500
        X0_train(count1+1,:) = train_data(i,2:785);
        count1 = count1 + 1;
    end
    if train_data(i,1) == 8 && count1 < 5500
        X8_train(count2+1,:) = train_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 11000 % 2*5500
        break
    end
```

```
end

%keeping only 0 and 8 test data ,both same number(974)
X0_test = zeros(974,784);
X8_test = zeros(974,784);

count1 = 0;
count2 = 0;
for i = 1:10000
    if test_data(i,1) == 0 && count1 < 974
        X0_test(count1+1,:) = test_data(i,2:785);
        count1 = count1 + 1;
    end
    if test_data(i,1) == 8 && count1 < 974
        X8_test(count2+1,:) = test_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 1948 % 2*974
        break;
    end

end

% normalization of imagevector values
X0_train = X0_train / 255;
X8_train = X8_train / 255;
X0_test = X0_test / 255;
X8_test = X8_test / 255;




% Neural Net training

% initialiazation of NN with dimention IxMxN "3-layers"
I = 784;
M = 300;
N = 1;

m = 2*0.0001; %learning rate
lamda = 0.00001; %smoothing factor moustakides has m= 2*1e-4 , and 1-lamda= 0.99
c = 1e-310;


% stohastic gradient descent -- learning

epochs = 10;
Best_Cost = inf;
Num_train = 5500; %number of train data = 5500
Num_test = 974;   %number of train data = 974


for d = 1:1
    A1 = (1/(M+I)) * randn(M,I);
```

```
    B1 = zeros(M,1);
    A2 = (1/(M*N)) * randn(N,M);
    B2 = 0;

    k=1;
    Cost = zeros(1,epochs*Num_train);
    for j = 1:epochs % epochs
        for i = 1:Num_train

        Z00 = X0_train(i,:)';
        Z10 = X8_train(i,:)';

        W01 = A1*Z00 + B1;
        Z01 = Relu(W01);
        W02 = A2*Z01 + B2;
        Y0 = Sigmoid(W02);

        W11 = A1*Z10 + B1;
        Z11 = Relu(W11);
        W12 = A2*Z11 + B2;
        Y1 = Sigmoid(W12); %

        Cost(k) = CrossEntropy(Y0,Y1);


        %derivative calculation
        u02 = diff_Phi_CrossEntropy(Y0);
        v02 = u02 .* diff_Sigmoid(W02);    %f2(x) = sigmoid , f2'(x) = diff_sigmoid
        u01 = (A2')*v02;
        v01 = u01 .* diff_Relu(W01);

        u12 = diff_Psi_CrossEntropy(Y1);
        v12 = u12 .* diff_Sigmoid(W12);
        u11 = (A2')*v12;
        v11 = u11 .* diff_Relu(W11);

        if (i == 1) && (j == 1)
            P2 = ((v02*[Z01' 1]) + (v12*[Z11' 1])) .^ 2;
        else
            P2 = (1 - lamda)*P2_old + lamda*power(((v02*[Z01' 1]) + (v12*[Z11'
1])),2);
        end

        P2_old = P2;

        if (i == 1) && (j == 1)
            P1 = ((v01*[Z00' 1]) + (v11*[Z10' 1])) .^ 2;
        else
            P1 = (1 - lamda)*P1_old + lamda*power(((v01*[Z00' 1]) + (v11*[Z10'
1])),2);
        end

        P1_old = P1;
```

```
        AB2 = [A2 B2] - m * (((v02*[Z01' 1]) + (v12*[Z11' 1])) ./ sqrt(c + P2)) ;
        A2 = AB2(1:M);
        B2 = AB2(1,M+N);

        AB1 = [A1 B1] - m * (((v01*[Z00' 1]) + (v11*[Z10' 1])) ./ sqrt(c + P1)) ;
        A1 = AB1(1:M,1:I);
        B1 = AB1(1:M,I+1);



        k=k+1;
        end
    end

    Cost_mean_itr = 0;
    for i = 1:Num_train
        Cost_mean_itr = ( Cost_mean_itr + Cost(epochs*Num_train - Num_train + i) );
    end

    Cost_mean_itr = Cost_mean_itr / Num_train ;

    if Best_Cost > Cost_mean_itr
        Best_Cost = Cost_mean_itr;
        AA1 = A1;
        AA2 = A2;
        BB1 = B1;
        BB2 = B2;
        Final_Cost = Cost;
    end

end



z=1;
Cost_mean = zeros(1,epochs);
for i = 0:Num_train:length(Final_Cost) - Num_train
    for l = 0:Num_train-1
        Cost_mean(z) = ( Cost_mean(z) + Final_Cost(i+l+1) );
    end
    z=z+1;
end



figure;
plot(Cost_mean/Num_train);
figure;
plot(Cost);

% test of Neural Net
count1 = 0;
count2 = 0;

for i = 1:Num_test
```

```
    Z00 = X0_test(i,:)';
    Z10 = X8_test(i,:)';

    W01 = AA1*Z00 + BB1;
    Z01 = Relu(W01);
    W02 = AA2*Z01 + BB2;
    Y0 = Sigmoid(W02);

    W11 = AA1*Z10 + BB1;
    Z11 = Relu(W11);
    W12 = AA2*Z11 + BB2;
    Y1 = Sigmoid(W12);

    if Y1 < 0.5
        count1 = count1 + 1;
    end

    if Y0 > 0.5
        count2 = count2 + 1;
    end

end
NN_CrossEntropy_error = 0.5*(count1/974) + 0.5*(count2/974);
```

Για τον <u>Exponential</u> Classifier πάνω στα δεδομένα της MNIST

```
% Mnist's whole data loading
train_data = load('mnist_train.csv');
test_data = load('mnist_test.csv');


%keeping only 0 and 8 train data ,both same number(5500)
X0_train = zeros(5500,784);
X8_train = zeros(5500,784);

count1 = 0;
count2 = 0;
for i = 1:60000
    if train_data(i,1) == 0 && count1 < 5500
        X0_train(count1+1,:) = train_data(i,2:785);
        count1 = count1 + 1;
    end
    if train_data(i,1) == 8 && count1 < 5500
        X8_train(count2+1,:) = train_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 11000 % 2*5500
        break
    end

end
```

```
%keeping only 0 and 8 test data ,both same number(974)
X0_test = zeros(974,784);
X8_test = zeros(974,784);

count1 = 0;
count2 = 0;
for i = 1:10000
    if test_data(i,1) == 0 && count1 < 974
        X0_test(count1+1,:) = test_data(i,2:785);
        count1 = count1 + 1;
    end
    if test_data(i,1) == 8 && count1 < 974
        X8_test(count2+1,:) = test_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 1948 % 2*974
        break;
    end

end

% normalization of imagevector values
X0_train = X0_train / 255;
X8_train = X8_train / 255;
X0_test = X0_test / 255;
X8_test = X8_test / 255;




% Neural Net training

% initialiazation of NN with dimention IxMxN "3-layers"
I = 784;
M = 300;
N = 1;

m = 2*0.0001; %learning rate
lamda = 0.00001;
c = 1e-300;


% stohastic gradient descent -- learning

epochs = 10;
Best_Cost = inf;
Num_train = 5500; %number of train data = 5500
Num_test = 974;   %number of train data = 974


for d = 1:1
    A1 = (1/(M+I)) * randn(M,I);
    B1 = zeros(M,1);
    A2 = (1/(M*N)) * randn(N,M);
    B2 = 0;
```

```
    k=1;
    Cost = zeros(1,epochs*Num_train);
    for j = 1:epochs % epochs
        for i = 1:Num_train

        Z00 = X0_train(i,:)';
        Z10 = X8_train(i,:)';

        W01 = A1*Z00 + B1;
        Z01 = Relu(W01);
        W02 = A2*Z01 + B2;
        Y0 = W02;

        W11 = A1*Z10 + B1;
        Z11 = Relu(W11);
        W12 = A2*Z11 + B2;
        Y1 = W12; %

        Cost(k) = Expon(Y0,Y1);


        %derivative calculation
        u02 = diff_Phi_expon(Y0);
        v02 = u02 .* 1;    %f2(x) = x, f2'(x) = 1
        u01 = (A2')*v02;
        v01 = u01 .* diff_Relu(W01);

        u12 = diff_Psi_expon(Y1);
        v12 = u12 .* 1;
        u11 = (A2')*v12;
        v11 = u11 .* diff_Relu(W11);

        if (i == 1) && (j == 1)
            P2 = ((v02*[Z01' 1]) + (v12*[Z11' 1])) .^ 2;
        else
            P2 = (1 - lamda)*P2_old + lamda*power(((v02*[Z01' 1]) + (v12*[Z11'
1])),2);
        end

        P2_old = P2;

        if (i == 1) && (j == 1)
            P1 = ((v01*[Z00' 1]) + (v11*[Z10' 1])) .^ 2;
        else
            P1 = (1 - lamda)*P1_old + lamda*power(((v01*[Z00' 1]) + (v11*[Z10'
1])),2);
        end

        P1_old = P1;


        AB2 = [A2 B2] - m * (((v02*[Z01' 1]) + (v12*[Z11' 1])) ./ sqrt(c + P2)) ;
        A2 = AB2(1:M);
        B2 = AB2(1,M+N);
```

```
        AB1 = [A1 B1] - m * (((v01*[Z00' 1]) + (v11*[Z10' 1])) ./ sqrt(c + P1)) ;
        A1 = AB1(1:M,1:I);
        B1 = AB1(1:M,I+1);



        k=k+1;
        end
    end

    Cost_mean_itr = 0;
    for i = 1:Num_train
        Cost_mean_itr = ( Cost_mean_itr + Cost(epochs*Num_train - Num_train + i) );
    end

    Cost_mean_itr = Cost_mean_itr / Num_train ;

    if Best_Cost > Cost_mean_itr
        Best_Cost = Cost_mean_itr;
        AA1 = A1;
        AA2 = A2;
        BB1 = B1;
        BB2 = B2;
        Final_Cost = Cost;
    end

end



z=1;
Cost_mean = zeros(1,epochs);
for i = 0:Num_train:length(Final_Cost) - Num_train
    for l = 0:Num_train-1
        Cost_mean(z) = ( Cost_mean(z) + Final_Cost(i+l+1) );
    end
    z=z+1;
end



figure;
plot(Cost_mean/Num_train);
figure;
plot(Cost);

% test of Neural Net
count1 = 0;
count2 = 0;

for i = 1:Num_test
    Z00 = X0_test(i,:)';
    Z10 = X8_test(i,:)';
```

```
    W01 = AA1*Z00 + BB1;
    Z01 = Relu(W01);
    W02 = AA2*Z01 + BB2;
    Y0 = W02;

    W11 = AA1*Z10 + BB1;
    Z11 = Relu(W11);
    W12 = AA2*Z11 + BB2;
    Y1 = W12;

    if Y1 < 0
        count1 = count1 + 1;
    end

    if Y0 > 0
        count2 = count2 + 1;
    end

end
NN_Expon_error = 0.5*(count1/974) + 0.5*(count2/974);
```

Για τον <u>Hinge</u> Classifier πάνω στα δεδομένα της MNIST

```
clc
clear;

% Mnist's whole data loading
train_data = load('mnist_train.csv');
test_data = load('mnist_test.csv');


%keeping only 0 and 8 train data ,both same number(5500)
X0_train = zeros(5500,784);
X8_train = zeros(5500,784);

count1 = 0;
count2 = 0;
for i = 1:60000
    if train_data(i,1) == 0 && count1 < 5500
        X0_train(count1+1,:) = train_data(i,2:785);
        count1 = count1 + 1;
    end
    if train_data(i,1) == 8 && count1 < 5500
        X8_train(count2+1,:) = train_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 11000 % 2*5500
        break
    end
```

```
end

%keeping only 0 and 8 test data ,both same number(974)
X0_test = zeros(974,784);
X8_test = zeros(974,784);

count1 = 0;
count2 = 0;
for i = 1:10000
    if test_data(i,1) == 0 && count1 < 974
        X0_test(count1+1,:) = test_data(i,2:785);
        count1 = count1 + 1;
    end
    if test_data(i,1) == 8 && count1 < 974
        X8_test(count2+1,:) = test_data(i,2:785);
        count2 = count2 + 1;
    end
    if count1 + count2 == 1948 % 2*974
        break;
    end

end

% normalization of imagevector values
X0_train = X0_train / 255;
X8_train = X8_train / 255;
X0_test = X0_test / 255;
X8_test = X8_test / 255;




% Neural Net training

% initialiazation of NN with dimention IxMxN "3-layers"
I = 784;
M = 300;
N = 1;

m = 2*0.0001; %learning rate
lamda = 0.00001; %smoothing factor
c = 1e-310;


% stohastic gradient descent -- learning

epochs = 10;
Best_Cost = inf;
Num_train = 5500; %number of train data = 5500
Num_test = 974;   %number of train data = 974


for d = 1:1
    A1 = (1/(M+I)) * randn(M,I);
    B1 = zeros(M,1);
```

```
    A2 = (1/(M*N)) * randn(N,M);
    B2 = 0;

    k=1;
    Cost = zeros(1,epochs*Num_train);
    for j = 1:epochs % epochs
        for i = 1:Num_train

        Z00 = X0_train(i,:)';
        Z10 = X8_train(i,:)';

        W01 = A1*Z00 + B1;
        Z01 = Relu(W01);
        W02 = A2*Z01 + B2;
        Y0 = W02;

        W11 = A1*Z10 + B1;
        Z11 = Relu(W11);
        W12 = A2*Z11 + B2;
        Y1 = W12; %

        Cost(k) = Hinge(Y0,Y1);


        %derivative calculation
        u02 = heaviside(Y0+1);
        v02 = u02 .* 1;    %f2(x) = x , f2'(x) = 1
        u01 = (A2')*v02;
        v01 = u01 .* diff_Relu(W01);

        u12 = -heaviside(-Y1+1); % careful at Hinge's Derivative!
        v12 = u12 .* 1;
        u11 = (A2')*v12;
        v11 = u11 .* diff_Relu(W11);

        if (i == 1) && (j == 1)
            P2 = ((v02*[Z01' 1]) + (v12*[Z11' 1])) .^ 2;
        else
            P2 = (1 - lamda)*P2_old + lamda*power(((v02*[Z01' 1]) + (v12*[Z11'
1])),2);
        end

        P2_old = P2;

        if (i == 1) && (j == 1)
            P1 = ((v01*[Z00' 1]) + (v11*[Z10' 1])) .^ 2;
        else
            P1 = (1 - lamda)*P1_old + lamda*power(((v01*[Z00' 1]) + (v11*[Z10'
1])),2);
        end

        P1_old = P1;


        AB2 = [A2 B2] - m * (((v02*[Z01' 1]) + (v12*[Z11' 1])) ./ sqrt(c + P2)) ;
```

```
        A2 = AB2(1:M);
        B2 = AB2(1,M+N);

        AB1 = [A1 B1] - m * (((v01*[Z00' 1]) + (v11*[Z10' 1])) ./ sqrt(c + P1)) ;
        A1 = AB1(1:M,1:I);
        B1 = AB1(1:M,I+1);



        k=k+1;
        end
    end

    Cost_mean_itr = 0;
    for i = 1:Num_train
        Cost_mean_itr = ( Cost_mean_itr + Cost(epochs*Num_train - Num_train + i) );
    end

    Cost_mean_itr = Cost_mean_itr / Num_train ;

    if Best_Cost > Cost_mean_itr
        Best_Cost = Cost_mean_itr;
        AA1 = A1;
        AA2 = A2;
        BB1 = B1;
        BB2 = B2;
        Final_Cost = Cost;
    end

end



z=1;
Cost_mean = zeros(1,epochs);
for i = 0:Num_train:length(Final_Cost) - Num_train
    for l = 0:Num_train-1
        Cost_mean(z) = ( Cost_mean(z) + Final_Cost(i+l+1) );
    end
    z=z+1;
end



figure;
plot(Cost_mean/Num_train);
figure;
plot(Cost);

% test of Neural Net
count1 = 0;
count2 = 0;

for i = 1:Num_test
    Z00 = X0_test(i,:)';
```

```
    Z10 = X8_test(i,:)';

    W01 = AA1*Z00 + BB1;
    Z01 = Relu(W01);
    W02 = AA2*Z01 + BB2;
    Y0 = W02;

    W11 = AA1*Z10 + BB1;
    Z11 = Relu(W11);
    W12 = AA2*Z11 + BB2;
    Y1 = W12;

    if Y1 < 0
        count1 = count1 + 1;
    end

    if Y0 > 0
        count2 = count2 + 1;
    end

end
NN_Hinge_error = 0.5*(count1/974) + 0.5*(count2/974);
```