**Μηχανική Μάθηση**

**1ο Σετ Ασκήσεων**

ΤΜΗΜΑ: Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΟΝΟΜΑ: ΚΩΝΣΤΑΝΤΙΝΟΣ

ΕΠΩΝΥΜΟ: ΛΑΓΑΡΟΣ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1053705

ΕΤΟΣ ΦΟΙΤΗΣΗΣ: 6o

**Πρόβλημα 1**

**Α)** Όπως έχουμε δείξει το βέλτιστο τεστ κατά Bayes(τεστ του λόγου πιθανοφάνιας)

για κάθε υλοποίηση του τυχαίου διανύσματος Χ=[χ1,χ2] , να πηγαίνουμε να υπολογίζουμε τον λόγο πιθανοφάνιας και να το συγκρίνουμε με τον λόγο

Εδώ οι εκ των προτέρων πιθανότητες, P(H0) και P(H1), είναι ίσες μεταξύ και ίσες με την μονάδα.

Επομένως για δεδομένη τιμή του Χ = [χ1,χ2]

όταν > 1 , ψηφίζω υπέρ του H1 (δηλαδή ότι το Χ προέρχεται από την υπόθεση Η1)

όταν , ψηφίζω υπέρ του H0

ενώ όταν = 1 , τότε αποφασίζω στην τύχη .

Στο συγκεκριμένο παράδειγμα οι πυκνότητες πιθανότητας είναι :

f0(x1,x2) = f0(x1)\*f0(x2) , (αφού x1 και x2 είναι ανεξάρτητα μεταξύ τους)

Άρα f1(x1,x2) =

και f0(x1,x2) =

**Β)**

Δημιουργούμε 106 ζευγάρια (χ1,χ2) από την κάθε υπόθεση και εκτελούμε το τεστ κατά Bayes.

Κάνοντας στην προσημείωση στην Matlab :

To ποσοστό των φορών που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η0 = 0.4235

Το ποσοστό των φορών που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η1 = 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 106 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 ϕ(z) = − log(1 − z)

και ψ(z) = − log(z) καθώς και με την μέθοδο Exponential , όπου ϕ(z) = e0.5z ψ(z) = e -0.5z .

Το κόστος του προβλήματος βελτιστοποίησης που θέλουμε να ελαχιστοποιήσουμε είναι το

J(u(x,θ)) = Φ(u(x01, x02, θ)) + Ψ(u(x11,, x12, θ))

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

θt = θt−1 − µ\*∇θJ(θt)

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 :



Όπου:



Και αρχικοποιούμε με :



με λ << 1 , και c το θέτουμε ένα μικρό ε > 0 ώστε να αποφύγουμε διαίρεση με το 0.

Οπότε το c το έθεσα ίσο με 10-310.

(Ακ και Βκ είναι τα βάροι και offset τoυ 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 αν τα ίδια δεδομένα που εξέτασε και η μέθοδος του Bayes προέρχονται από την υπόθεση Η0 ή Η1 και υπολογίστηκε το ποσοστό σφάλματος που κάνουν.

Με την μέθοδο Cross Entropy :

Το μέσο κόστος ανά κάθε εποχή :

A picture containing chart

Description automatically generated

Εδώ η έξοδος του δικτύου είναι , όπου

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\*106 δεδομένα που δημιουργήσαμε πριν

Παίρνουμε :

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η0 = 0.3024

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η1 = 0.4336

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

Με την μέθοδο Exponential :

Το μέσο κόστος ανά κάθε εποχή :

Chart

Description automatically generated

Εδώ η έξοδος του δικτύου είναι , όπου

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\*106 δεδομένα που δημιουργήσαμε πριν

Παίρνουμε :

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η0 = 0.3478

Οι φορές που η μέθοδος αποφάσισε **εσφαλμένα** υπέρ του Η1 = 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

ϕ(z) = max(1+z,0) και ψ(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 μ = 2\*10-4 και λ = 10-5

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

Chart, line chart

Description automatically generated

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

**Chart

Description automatically generated**

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

Chart, line chart

Description automatically generated

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 CrossEntropy 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 Exponential 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 Exponential 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 Hinge 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);