

Μηχανική Μάθηση

2^ο Σετ Ασκήσεων

ΟΝΟΜΑ: Αλέξανδρος

ΕΠΩΝΥΜΟ: Καλέργης

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1066665

ΕΤΟΣ ΦΟΙΤΗΣΗΣ: 4ο

Πρόβλημα 2.1

Σε αυτό το Πρόβλημα μας δίνεται ένα generative μοντέλο για τη δημιουργία ενός “χειρόγραφου” αριθμητικού ψηφίου 8.

- Το αρχείο περιέχει δύο μήτρες A_1 , A_2 και δύο διανύσματα B_1 , B_2 . Η μήτρα A_1 είναι διάστασης 128×10 , η A_2 διάστασης 784×128 , το B_1 είναι 128×1 και το B_2 είναι 784×1 .
 - Το νευρωνικό δίκτυο είναι πλήρως συνδεδεμένο (fully connected) και στο πρώτο εσωτερικό επίπεδο εφαρμόζουμε την ReLU ενώ στην έξοδο την sigmoid.
 - Η είσοδος Z είναι διάστασης 10×1 και τα στοιχεία της είναι ανεξάρτητες υλοποιήσεις μιας Gaussian με μέση τιμή 0 και διασπορά 1.
 - Η έξοδος είναι διάνυσμα διάστασης 784×1 το οποίο χρησιμοποιούμε για να δημιουργήσουμε μια εικόνα 28×28 με ένα “χειρόγραφο” 8.

Οι εξισώσεις που υπολογίζουν την έξοδο είναι:

$$\begin{aligned} W_1 &= A_1 * Z + B_1 \\ Z_1 &= \max(W_1, 0) \quad (\text{ReLU}) \\ W_2 &= A_2 * Z_1 + B_2 \\ X &= 1/(1 + e^{W_2}) \quad (\text{Sigmoid}) \end{aligned} \quad (1)$$

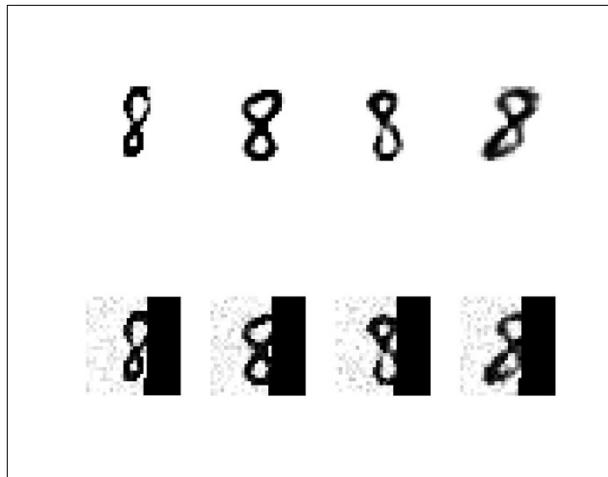
Αποτέλεσμα:

Πρόβλημα 2.2

Σε αυτό το Πρόβλημα έχουμε στη διάθεση μας δύο πίνακες X_i και X_n οι οποίοι είναι διάστασης 784×4 .

- Ο X_i περιέχει τέσσερις εικόνες (μία σε κάθε στήλη του) από τον αριθμό οκτώ και ο X_n περιέχει τις ίδιες εικόνες στις οποίες έχει προστεθεί θόρυβος.
- Θεωρούμε ότι δεν μπορούμε να χρησιμοποιήσουμε και τα 784 σημεία κάθε στήλης του X_n άλλα κάθε φορά κρατάμε $N = 500, 400, 350$ και 300 σημεία και προσπαθούμε να ανακατασκευάσουμε την εικόνα που προκύπτει.

Τα αποτελέσματα φαίνονται στη παρακάτω εικόνα:



Πιο συγκεκριμένα θεωρούμε ότι κάθε στήλη Y του X_n που μπορούμε να χρησιμοποιήσουμε (δηλαδή τα N σημεία) έχει προέλθει από έναν μετασχηματισμό μιας στήλης X του X_i της μορφής:

$$Y = TX + W$$

Όπου W είναι ο προστιθέμενος θόρυβος και T είναι η μήτρα μετασχηματισμού που έχει διαστάσεις $N \times 784$ και έχει τη μορφή:

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}$$

Για να λύσουμε το πρόβλημα της ανακατασκευής ψάχνουμε εκείνο το \hat{Z} για το οποίο εάν υπολογίσουμε το $\hat{X} = \hat{G}(Z)$ το \hat{X} θα προσεγγίζει το ζητούμενο X .

Για τα Προβλήματα 2.2 και 2.3 θα χρειαστεί να εφαρμόσουμε τον αλγόριθμο Gradient Descent για να ελαχιστοποιήσουμε το κόστος $J(Z)$:

$$J(Z) = M \log \left(\|T\hat{G}(Z) - Xn\|^2 \right) + \|Z\|^2$$

Δηλαδή: $Z_t = Z_{t-1} - \mu \nabla_{Z_{t-1}} \{ M \log \left(\|T\hat{G}(Z_{t-1}) - Xn\|^2 \right) + \|Z_{t-1}\|^2 \}$, όπου μ το learning rate.

Θέλουμε λοιπόν να βρούμε την παράγωγο της συνάρτησης κόστους $J(Z)$ ως προς Z . Άρα θέλουμε να υπολογίσουμε την παράσταση:

$$\nabla_Z J(Z) = \nabla_Z M \log \left(\|T\hat{G}(Z) - Xn\|^2 \right) + 2Z$$

Αρκεί λοιπόν να υπολογίσουμε τον όρο: $\nabla_Z M \log \left(\|T\hat{G}(Z) - Xn\|^2 \right)$

Θεωρούμε: $\Phi(\hat{X}) = M \log \left(\|T\hat{G}(Z) - Xn\|^2 \right) = \Phi(Z)$, άρα αρκεί να υπολογίσουμε τον όρο: $\nabla_Z \Phi(\hat{X})$

Γνωρίζουμε τις πράξεις του Νευρωνικού Δικτύου μας από την (1), άρα υπολογίζουμε την $\nabla_Z \Phi(\hat{X})$ ως εξής:

$$U_2 = \nabla_{\hat{X}} \Phi(\hat{X}) = \frac{2T^T(T\hat{X} - X_n)}{\|T\hat{X} - X_n\|^2}$$

$$V_2 = U_2 \cdot * f_2'(W2)$$

$$U_1 = A_2^T V_2$$

$$V_1 = U_1 \cdot * f_1'(W1)$$

$$U_0 = A_1^T V_1 = \nabla_Z \Phi(\hat{X})$$

Όπου:

$$f_1'(W1) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$f_2'(W2) = \frac{-e^x}{(1 + e^x)^2}$$

Μέσω της μεθόδου ADAM μπορούμε να κανονικοποιήσουμε τη παράγωγο ώστε να έχουμε πιο ομαλή σύγκληση βρίσκοντας την ενέργεια:

$$P_t = (1 - \lambda)P_{t-1} + \lambda\Psi(Z_{t-1})^2$$

$$\text{Όπου } \Psi(Z_{t-1}) = \nabla_{Z_{t-1}}\Phi(Z_{t-1}) + 2Z_{t-1}$$

Επομένως ο αλγόριθμος Gradient Descent γίνεται:

$$Z_t = Z_{t-1} - \mu \frac{\Psi(Z_{t-1})}{\sqrt{10^{-6} + P_{t-1}}}$$

Στον αλγόριθμό μας έχουμε επιλέξει ως αρχικό Z ένα διάνυσμα 10×1 που η επιλογή των στοιχείων γίνεται μέσω της ομοιόμορφης κατανομής $N(0,1)$. Η επιλογή αυτή στη MatLab γίνεται με την εντολή `randn()`.

Τρέχουμε τον Αλγόριθμό μας 7 φορές με διαφορετικά αρχικά διανύσματα εισόδου Z και επιλέγουμε εκείνο για το οποίο συγκλίνει καλύτερα. Η επιλογή των Z γίνεται ξεχωριστά για κάθε στήλη του Xn .

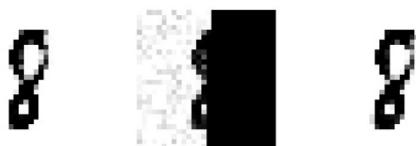
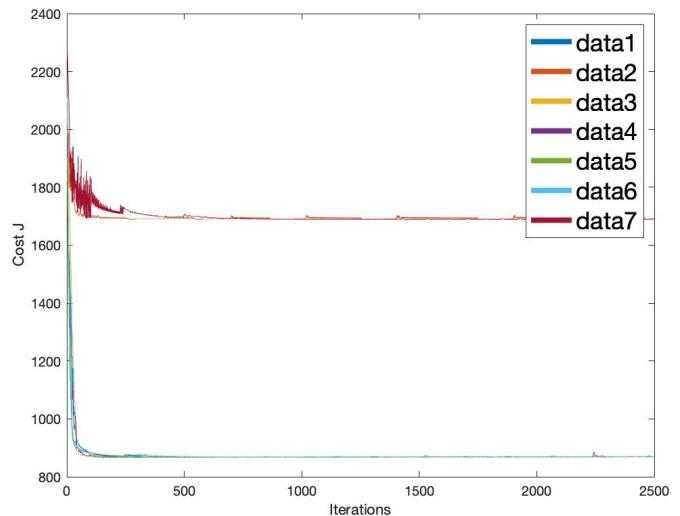
Τα παρακάτω σχήματα δείχνουν τη σύγκλιση του αλγορίθμου για τα διάφορα Z και για διαφορετικές τιμές N .

1^ο «χειρόγραφο» οκτώ:

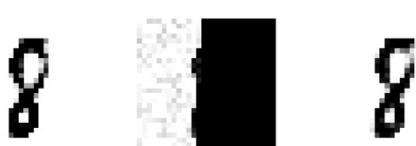
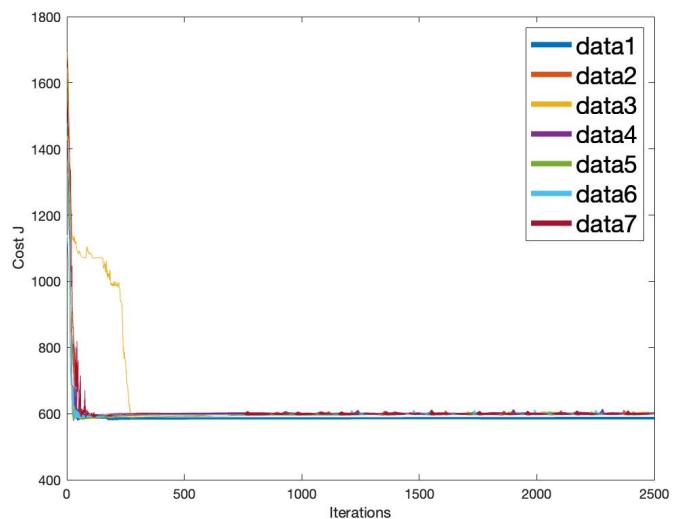


(i) $N = 500$

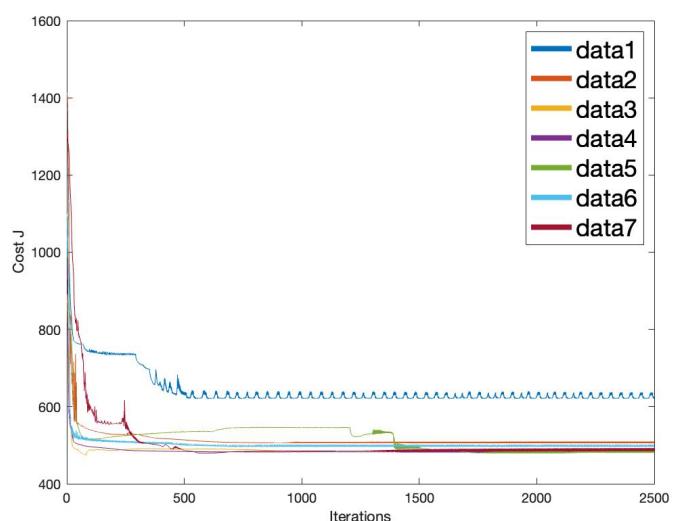
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



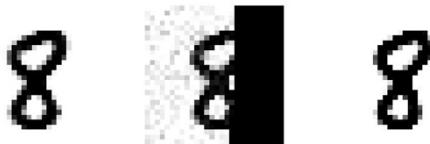
(ii) $N = 400$



(iii) $N = 350$

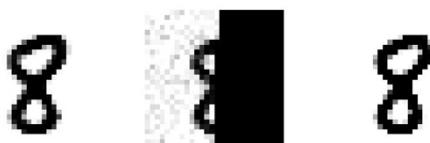
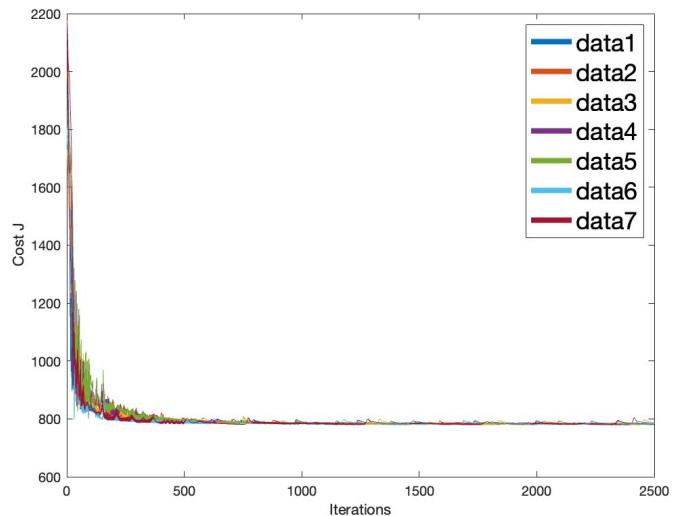


2^ο «χειρόγραφο» οκτώ:

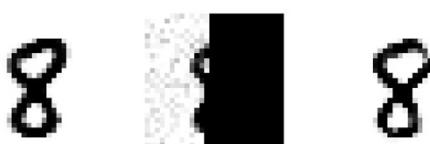
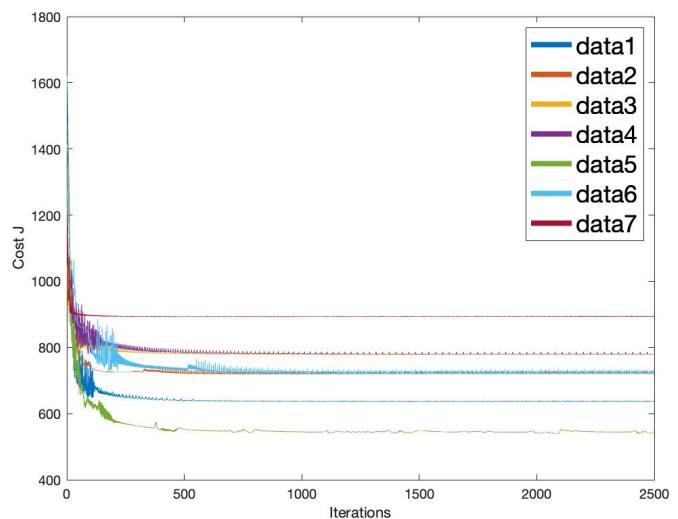


(i) $N = 500$

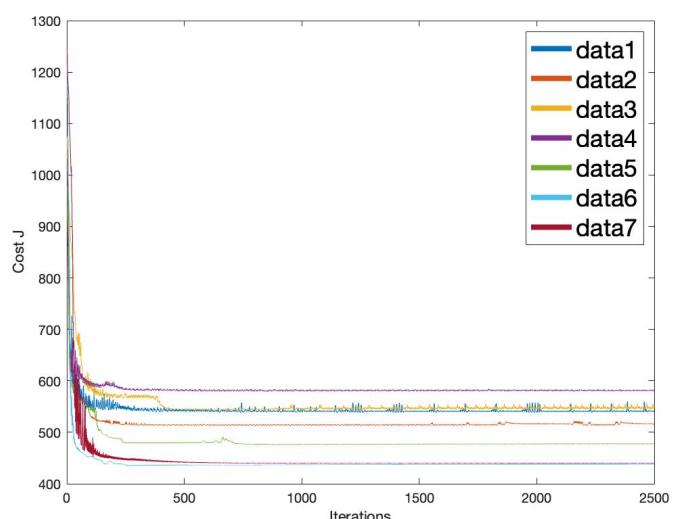
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



(ii) $N = 400$



(iii) $N = 350$

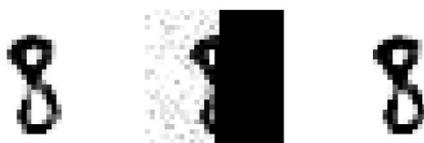
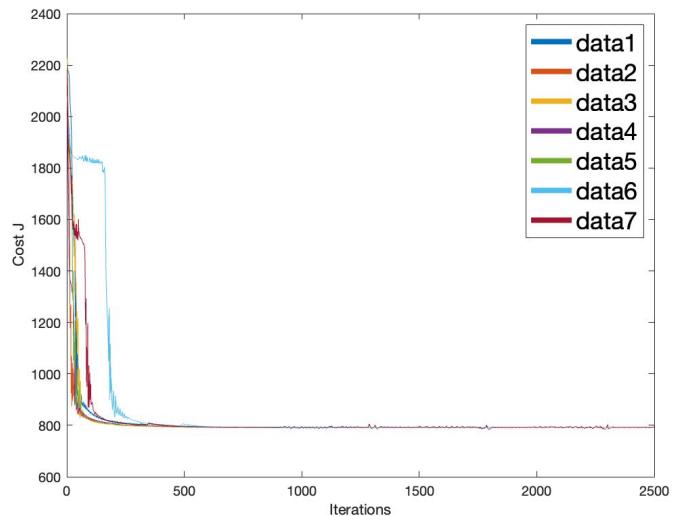


3^ο «χειρόγραφο» οκτώ:

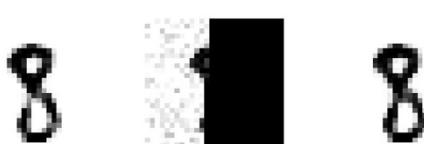
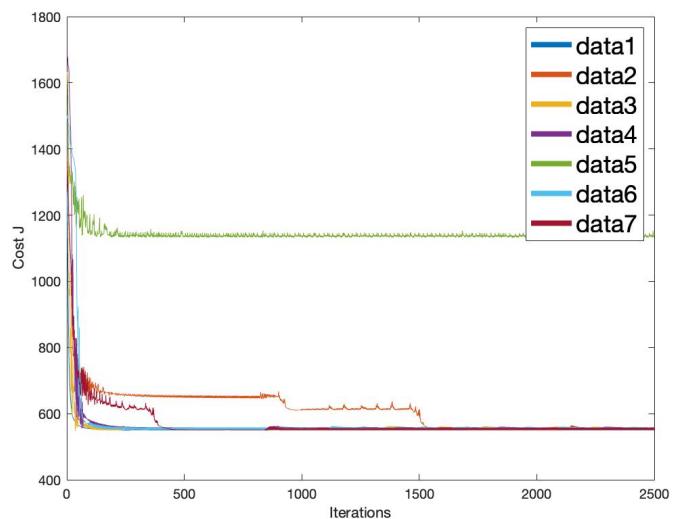


(i) $N = 500$

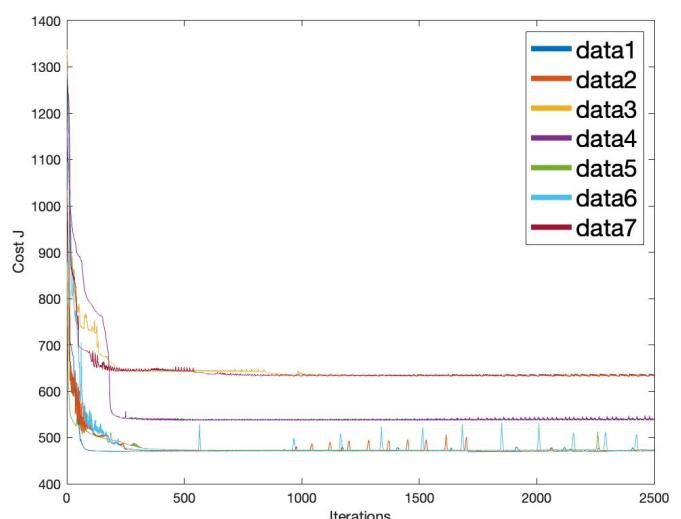
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



(ii) $N = 400$



(iii) $N = 350$

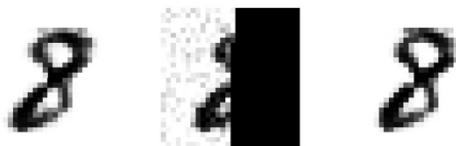
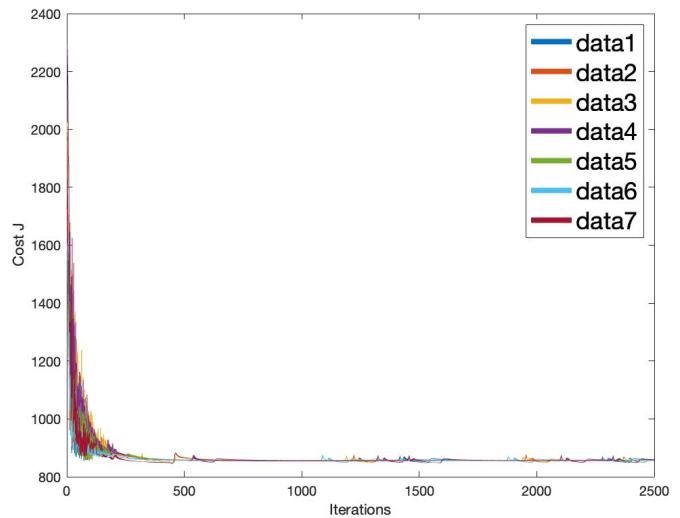


4^ο «χειρόγραφο» οκτώ:

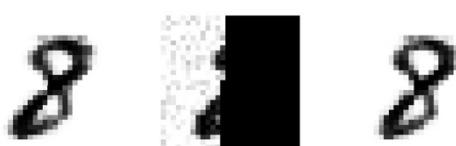
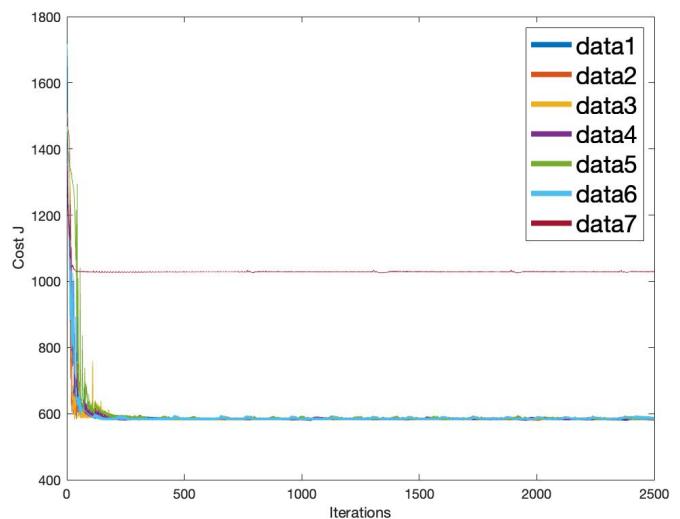


(i) $N = 500$

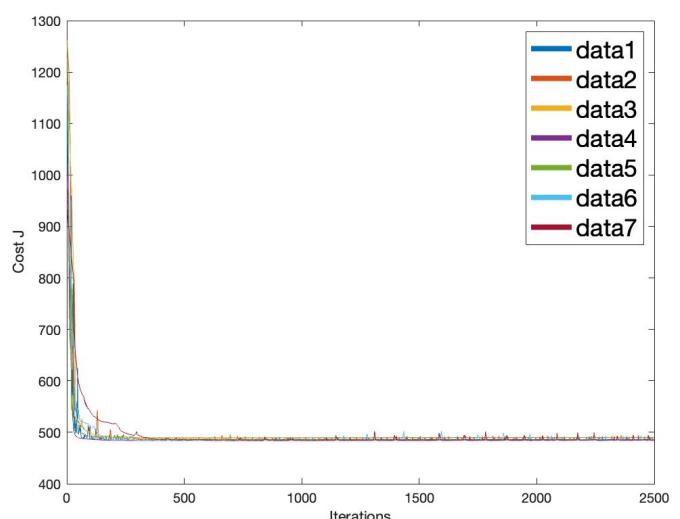
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



(ii) $N = 400$



(iii) $N = 350$

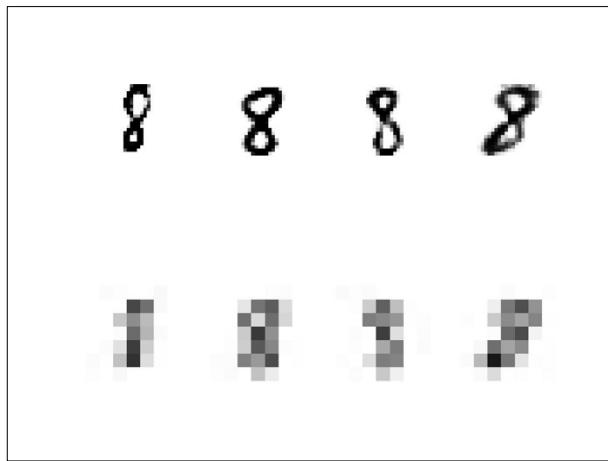


Πρόβλημα 2.3

Σε αυτό το Πρόβλημα έχουμε στη διάθεση μας δύο πίνακες X_i διάστασης 784×4 και X_n διάστασης 49×4 .

- Ο X_i περιέχει τέσσερις εικόνες (μία σε κάθε στήλη του) από τον αριθμό οκτώ και ο X_n έχει προέλθει από ελάττωση της ανάλυσης (resolution) των ιδανικών X_i και πρόσθεση θορύβου.

Τα αποτελέσματα φαίνονται στη παρακάτω εικόνα:



Πιο συγκεκριμένα θεωρούμε ότι κάθε στήλη Y του X_n που μπορούμε να χρησιμοποιήσουμε (δηλαδή τα N σημεία) έχει προέλθει από έναν μετασχηματισμό μιας στήλης X του X_i της μορφής:

$$Y = TX + W$$

Όπου W είναι ο προστιθέμενος θόρυβος και T είναι η μήτρα μετασχηματισμού που έχει διαστάσεις 49×784 και θέλουμε κάθε γραμμή να δειγματοληπτεί 16 σημεία του διανύσματος X που αντιστοιχούν σε ένα 4×4 κουτάκι της εικόνας και να παίρνει το μέσο όρο τους διαιρώντας το άθροισμά τους με 16 και έχει τη μορφή:

1^η γραμμή:

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (5 - 28) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (33 - 56) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (61 - 84) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & 0 & \dots & 0 \end{bmatrix}$$

2^η γραμμή:

$$\begin{bmatrix} (1 - 4) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (9 - 32) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (37 - 60) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (65 - 88) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & 0 & \dots & 0 \end{bmatrix}$$

8^η γραμμή:

$$\begin{bmatrix} (1 - 112) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (117 - 140) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (145 - 168) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (173 - 196) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & 0 & \dots & 0 \end{bmatrix}$$

⋮

15^η γραμμή:

$$\begin{bmatrix} (1 - 228) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (233 - 256) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (261 - 284) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & (289 - 312) & \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & 0 & \dots & 0 \end{bmatrix}$$

⋮

Ο Κώδικας MatLab για τη μήτρα μετασχηματισμού T:

```
%Transformation Matrix
mo = 1/16;
T = zeros(N,784);
t1 = 0;
t2 = 0;
for i = 0:48
    if mod(i,7) == 0 %correct μολις φτασει 8η γραμμη(τρεχω απο το 0 αρα για
                    % i = 7 ειμαι στην 8η επαναληψη αρα 8η γραμμη του T
        t1 = 0;
        t2 = 4 * int16(i/7) * 28; %correct
    end
    for j = 0*28 : 28 : 3*28
        T( i+1, (j+1) + t2 + t1 : (j+4) + t2 + t1 ) = mo;
    end
    t1 = t1+4;
end
```

Κάθε $7^*K + 1$ γραμμές της μήτρας μετασχηματισμού γίνεται μια αλλαγή στις θέσεις των τιμών που βάζουμε στη μήτρα. Πιο συγκεκριμένα κάθε 8 γραμμές τα πρώτα K^*4^*28 στοιχεία είναι 0 (όπου K συμβολίζει σε ποια στήλη των μπλοκ είμαστε $K = 0,1,2,3,4,5,6$). Αυτό συμβαίνει διότι τα μπλοκ των 4×4 χωράνε 7 φορές οριζόντια και 7 φορές κατακόρυφα σε μια εικόνα 28×28 και την K φορά που επεξεργαζόμαστε τον πίνακα X μεταφέρομαστε στην διπλανή στήλη των μπλοκ του πίνακα 28×28 . Αυτή η μεταφορά όταν ο πίνακας είναι διάνυσμα 784×1 μεταφράζεται ότι επεξεργαζόμαστε από το $K^*4^*28^o$ στοιχείο του διανύσματος και μετά.

Πιο κατανοητό γίνεται με το παρακάτω σχήμα:

1 ^η γραμμή	8 ^η γραμμή	15 ^η γραμμή	22 ^η γραμμή	29 ^η γραμμή	36 ^η γραμμή	43 ^η γραμμή
2 ^η γραμμή						
3 ^η γραμμή						
4 ^η γραμμή						
5 ^η γραμμή						
6 ^η γραμμή						
7 ^η γραμμή						

Όπου όλος ο πίνακας είναι 28×28 και κάθε κουτάκι είναι το μπλοκ 7×7

Το κάθε κουτάκι δείχνει ποια γραμμή της μήτρας T το επηρεάζει.

Για να λύσουμε το πρόβλημα της ανακατασκευής ψάχνουμε εκείνο το \hat{Z} για το οποίο εάν υπολογίσουμε το $\hat{X} = \hat{G}(Z)$ το \hat{X} θα προσεγγίζει το ζητούμενο X.

Θα χρησιμοποιήσουμε όπως και στο προηγούμενο ερώτημα τον αλγόριθμο Gradient Descent όπως εξηγήσαμε παραπάνω. Σε αυτό το πρόβλημα η μόνη μας διαφορά είναι ο πίνακας μετασχηματισμού T που θα εφαρμοστεί στο \hat{X} .

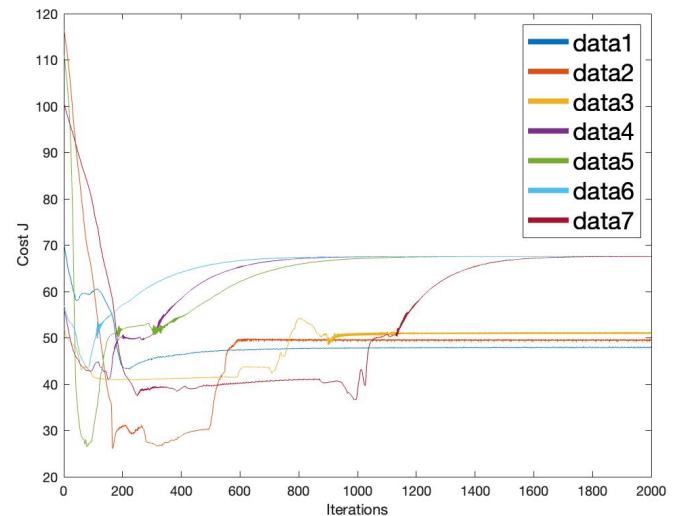
Για την εφαρμογή του αλγορίθμου όπως και στο προηγούμενο ερώτημα για κάθε διαφορετικό 8 που μας δίνεται «τρέχουμε» 7 φορές για διαφορετικές αρχικές τιμές των διανυσμάτων εισόδου Z. Από αυτά τα Z διαλέγουμε εκείνο που συγκλίνει καλύτερα ο αλγόριθμος για να έχουμε καλύτερα αποτελέσματα.

Παρακάτω φαίνονται τα αποτελέσματα του αλγορίθμου:

1^o «χειρόγραφο» οκτώ:



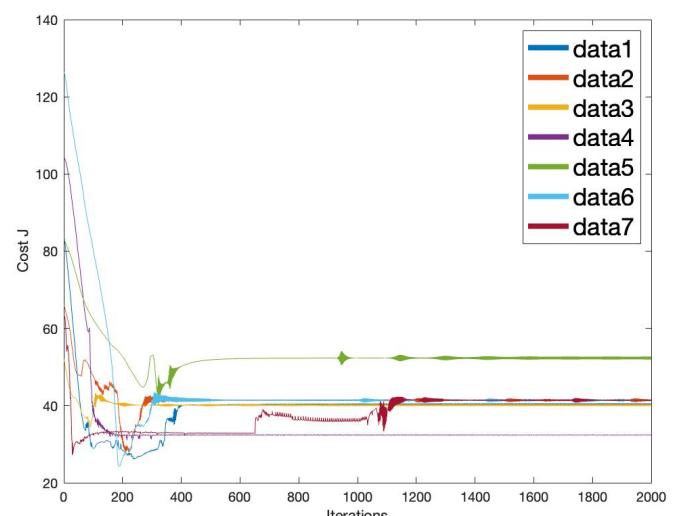
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



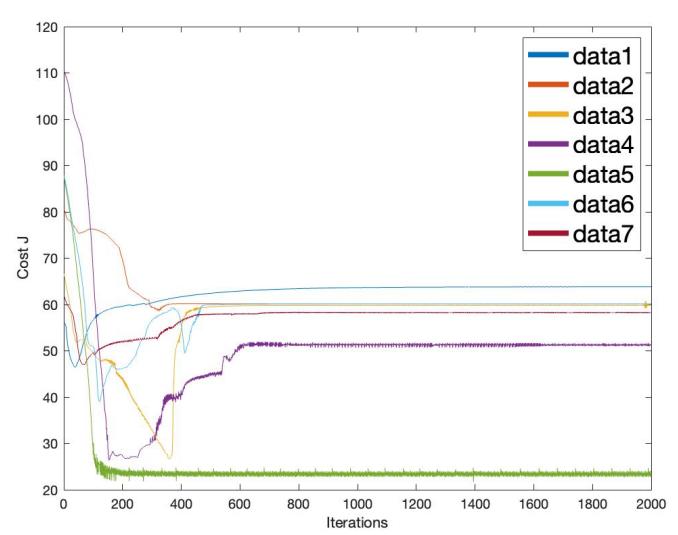
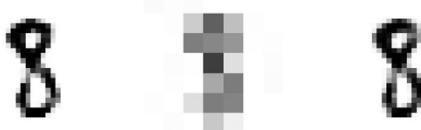
2^o «χειρόγραφο» οκτώ:



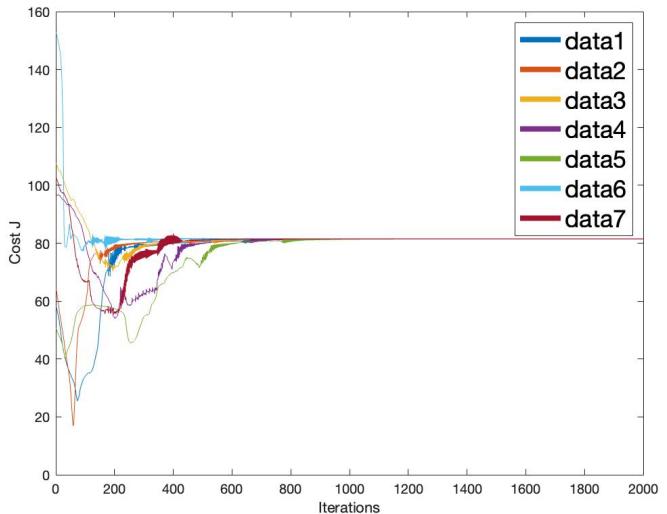
Κόστος $J(Z)$ για τα 7 διαφορετικά Z :



3^o «χειρόγραφο» οκτώ:



4^ο «χειρόγραφο» οκτώ:



Κώδικες MatLab

Πρόβλημα 1:

```
clc;
clear;
close all;

data21 = load ('data21.mat');
A1 = data21.A_1; A2 = data21.A_2;
B1 = data21.B_1; B2 = data21.B_2;

for i=1:100
    Z = randn(10,1); %Random Input
    W1 = A1*Z + B1;
    Z1 = max(W1,0); %ReLU
    W2 = A2*Z1 + B2;
    X = 1./(1+exp(W2)); %Sigmoid

    subplot(10,10,i)
    imshow(reshape(X,28,28))
end
```

Πρόβλημα 2:

```
clc;
clear;
close all;

%-----Choosing which eight i'd like to see -----
%-----(eight = 1 --> means the 1st eight etc.)-----
% Change also the line 32
eight = 1;
N = 500;

%Loading Data
data21 = load ('data21.mat');
A1 = data21.A_1; A2 = data21.A_2;
B1 = data21.B_1; B2 = data21.B_2;
data22 = load ('data22.mat');
Xi = data22.X_i;
Xn = data22.X_n;

%Initial Conditions
gradJ = [];
iterations = 2000;
learning_rate = 0.05;
T = eye(N,784);
Xn_new = T*Xn;
Xn_new(N:784,:) = 0;

% Z1 = load("Z1.mat");
% Z2 = load("Z2.mat");
% Z3 = load("Z3.mat");
% Z4 = load("Z4.mat");

% Zinitial = Z1.Zall(:,i); %Input from file Z.Zall(:,i) /{i =
1,2,3,4,5,6,7}/
% where i choose the best initial Z for the
algorithm

Zinitial = randn(10,1);

lamb = 1;          %ADAM constant
P = 0;            %ADAM Power
c = 10^(-6);      %ADAM small number
Z = Zinitial;    %Input

%Gradient Descent
for i=1:iterations

    W1 = A1*Z + B1;
    Z1 = max(W1,0); %ReLU
    W2 = A2*Z1 + B2;
    X = 1./(1+exp(W2)); %Sigmoid
```

```

%Steps for GRAD_Z(PHI(X)) = U_0
f1_grad_W1 = DerivativeReLU(W1);
f2_grad_W2 = -exp(W2)./((1 + exp(W2)).^2);

U_2 = DerivativePhi(X , T, Xn, N, eight);
V_2 = U_2 .* f2_grad_W2;
U_1 = A2' * V_2;
V_1 = U_1 .* f1_grad_W1;
U_0 = A1' * V_1;
gradJ = N*U_0 + 2*Z;

%ADAM Power for Normalization
P = (1-lamb)*P + lamb * gradJ.^2;
lamb = 0.001;

%Algorithm
Z_next = Z - learning_rate * gradJ./sqrt(c + P);
Z = Z_next;

end

figure(1)
% ---Xi---
subplot(1,3,1)
imshow(reshape(Xi(:,eight),28,28));
% ---Xn---
subplot(1,3,2)
imshow(reshape(Xn_new(:,eight),28,28));
% ---New---
subplot(1,3,3)
imshow(reshape(X,28,28));

function y = DerivativeReLU(W)
W(W(:,:,1)<=0) = 0;
W(W(:,:,1)>0) = 1;
y = W;
end

function y = DerivativePhi(X , T, Xn, N, eight)
y = (2 / (norm( T*X - Xn(1:N,eight) )^2 ) ) * T' * (T*X - Xn(1:N,eight));
end

```

Πρόβλημα 3:

```
clc;
clear;
close all;

%-----Choosing which eight i'd like to see -----
%-----(eight = 1 --> means the 1st eight etc.)-----
% Change also the line 47
eight = 2;
N = 49;

%Loading Data
data21 = load ('data21.mat');
A1 = data21.A_1; A2 = data21.A_2;
B1 = data21.B_1; B2 = data21.B_2;
data23 = load ('data23.mat');
Xi = data23.X_i;
Xn = data23.X_n;

%Initial Conditions
gradJ = [];
iterations = 2000;
learning_rate = 0.005;
%Transformation Matrix
mo = 1/16;
T = zeros(N,784);
t1 = 0;
t2 = 0;
for i = 0:48
    if mod(i,7) == 0 %correct μολις φτασει 8η γραμμη(τρεχω απο το 0 αρα για
                    % i = 7 ειμαι στην 8η επαναληψη αρα 8η γραμμη του T
        t1 = 0;
        t2 = 4 * int16(i/7) * 28; %correct
    end
    for j = 0*28 : 28 : 3*28
        T( i+1, (j+1) + t2 + t1 : (j+4) + t2 + t1 ) = mo;
    end
    t1 = t1+4;
end

% Z1 = load("Z1.mat");
% Z2 = load("Z2.mat");
% Z3 = load("Z3.mat");
% Z4 = load("Z4.mat");

%Zintilial = Z2.Zall(:,3); %Input from file Z.Zall(:,i) /{i = 1,2,3,4,5,6,7}/
                           % where i choose the best initial Z for the
algorithm
Zintilial = randn(10,1);

lamb = 1;      %ADAM constant
P = 0;         %ADAM Power
c = 10^(-6);  %ADAM small number
Z = Zintilial;%Random Input
```

```

%Gradient Descent
for i=1:iterations

    W1 = A1*Z + B1;
    Z1 = max(W1,0); %ReLU
    W2 = A2*Z1 + B2;
    X = 1./(1+exp(W2)); %Sigmoid

    %Steps for GRAD_Z(PHI(X)) = U_0
    f1_grad_W1 = DerivativeReLU(W1);
    f2_grad_W2 = -exp(W2)./(1 + exp(W2)).^2;

    U_2 = DerivativePhi(X , T, Xn, N, eight);
    V_2 = U_2 .* f2_grad_W2;
    U_1 = A2' * V_2;
    V_1 = U_1 .* f1_grad_W1;
    U_0 = A1' * V_1;
    gradJ = N*U_0 + 2*Z;

    %ADAM Power for Normalization
    P = (1-lamb)*P + lamb * gradJ.^2;
    lamb = 0.001;

    %Algorithm
    Z_next = Z - learning_rate * gradJ./sqrt(c + P);
    Z = Z_next;

end

figure(1)
% ---Xi---
subplot(1,3,1)
imshow(reshape(Xi(:,eight),28,28));
% ---Xn---
subplot(1,3,2)
imshow(kron(reshape(Xn(:,eight),7,7),ones(4,4)));
% ---New---
subplot(1,3,3)
imshow(reshape(X,28,28));

function y = DerivativeReLU(W)
    W(W(:,:)<=0) = 0;
    W(W(:,:)>0) = 1;
    y = W;
end

function y = DerivativePhi(X , T, Xn, N, eight)
    y = (2 / (norm( T*X - Xn(1:N,eight) )^2 ) ) * T' * (T*X - Xn(1:N,eight));
end

```