Μηχανική Μάθηση

2ο Σετ Ασκήσεων

ΤΜΗΜΑ: Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

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

Πρόβλημα 1

Στο συγκεκριμένο πρόβλημα μας δίνεται ένα generative model { G(z,θo) , g(z) } , όπου για είσοδο z, η οποία είναι υλοποίηση τυχαίας μεταβλητής με πυκνότητα πιθανότητας g(z), ο Generator G(z) είναι σε θέση να μας παράγει υλοποιήσεις από « ¨οκτάρια¨.
Ο κλασσικός τρόπος εκπαίδευσης αυτών των Generative Μοντέλων είναι με Adversarial τεχνικές, λύνοντας κατάλληλα ένα minimax πρόβλημα.
Στην περίπτωση μας ο Generator (fully connected network) είναι ήδη εκπαιδευμένος και οι παράμετροι του μας δίνονται στο data21.mat αρχείο(πίνακες A1,A2 και διανύσματα B1,B2).
Επομένως, δίνοντας ως είσοδο 10 ανεξάρτητες υλοποιήσεις μιας N(0,1) τυχαίας μεταβλητής μπορούμε να υπολογίσουμε την έξοδο του Generator ακολουθώντας τις παρακάτω εξισώσεις :

$W_1 = A_1 * Z + B_1$
$Z_1 = f_1(W_1)$ ($f_1(x) = \max\{0,x\}$) (ReLU)
$W_2 = A_2 * Z_1 + B_2$
$X = f_2(W_2)$ ($f_2(x) = \frac{1}{1+e^{x}}$) (Sigmoid)

Η έξοδος του νευρονικού δικτύου είναι διάνυσμα μήκους 784 , όπου αποτελεί τα pixel μιας 28×28 ανάλυσης εικόνας ενός χειρόγραφου ¨8¨.

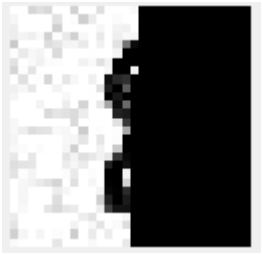
Για διαφορετική υλοποίηση του Gaussian τυχαίου διανύσματος Z , ο Generator παράγει μια διαφορετική υλοποίηση του τυχαίου διανύσματος που αντιπροσωπεύει τα pixel ενός χειρόγραφου ¨8¨ .

Παρακάτω παρατίθενται 100 διαφορετικές εικόνες από χειρόγραφα ¨οκτάρια¨ που παράγει ο Generator για 100 διαφορετικές υλοποιήσεις του τυχαίου διανύσματος Z.

Πρόβλημα 2

Σε αυτό το πρόβλημα βλέπουμε πως μπορούμε να χρησιμοποιήσουμε τα GANS σε προβλήματα αποκατάστασης εικόνων.

Στο συγκεκριμένο πρόβλημα βλέπουμε πως έχοντας μια εικόνας της οποία της λείπουν ορισμένα pixels και της έχει προστεθεί Gaussian θόρυβος πχ :



μπορούμε να ανακτήσουμε την εικόνας από την οποία προήλθε, δλδ :

Οι παραπάνω εικόνες είναι πίνακες διάστασης 28×28 (= 784 pixels) , αλλά μπορούμε να στοιχίσουμε τις τιμές των pixel κάθε στήλης σε ένα διάνυσμα μήκους 784.

Επομένως αν Y το διάνυσμα της εικόνας που έχουμε (με θόρυβο και λιγότερα pixel) και X το διάνυσμα της ιδανικής εικόνας, τότε τα δύο αυτά διανύσματα συνδέονται από την σχέση :

Y = T×X + W , όπου T ένας πίνακας που αποτελεί τον μετασχηματισμό που έχει υποστεί η ιδανική εικόνα και W ο Γκαουσιανός θόρυβος.

Όπως θα δείξουμε ο μετασχηματισμός εδώ (που αφαιρούνται κάποια pixel) είναι γραμμικός.

Όπως είπαμε, στο διάνυσμα X τα στοιχεία από την θέση 1 έως 28 αποτελούν τα pixel της πρώτης στήλης του πίνακα της εικόνας , τα στοιχεία από την θέση 29 έως 56 αποτελούν τα pixel της δεύτερης στήλης του πίνακα της εικόνας κ.ο.κ.
Επομένως, αν φανταστούμε ότι ο μετασχηματισμός T είναι ένας πίνακας της μορφής [I 0] όπου I ένας Μοναδιαίος Πίνακας διάστασης N×N, που N ο αριθμός των pixel που θα έχει η μετασχηματισμένη εικόνα, και 0 ένας μηδενικός πίνακας διάστασης N×(784-N), τότε ο μετασχηματισμός T είναι ένας πίνακας διάστασης N×784.

Είναι εύκολο να δούμε πως πράγματι, αν κάνουμε τον πολλαπλασιασμό T×X , το Y διάνυσμα(εικόνα) στο οποίο θα καταλήξουμε θα είναι μήκους N, και τα στοιχεία του θα είναι τα πρώτα N στοιχεία του διανύσματος (εικόνας) X.

Σε αυτό το πρόβλημα ουσιαστικά έχοντας την εικόνα Y θέλουμε να εκτιμήσουμε από ποια εικόνα X έχει προέλθει. Ψάχνουμε δηλαδή μια εκτίμηση $\hat{X}$ της ιδανικής εικόνας.

Όπως δείξαμε με πολύ λεπτομέρεια στο μάθημα 10 , αντί να κάνουμε μια εκτίμηση $\hat{X}$,θα ακολουθήσουμε μια μέθοδο βασισμένη στην Θεωρία Εκτίμησης Παραμέτρων, η οποία κάνει εκτίμηση της εισόδου $\hat{Z}$ του generator, καθώς θα έχουμε εμπιστοσύνη στο μοντέλο μας ότι για μια καλή εκτίμηση του Z , τότε το G($\hat{Z}$) θα είναι μια καλή εκτίμηση του X.
Και δείξαμε πως ο βέλτιστος τρόπος εκτίμησης του Z όταν ο μετασχηματισμός T είναι γνωστός, όπως στην περίπτωσή μας, είναι ελαχιστοποιώντας το κόστος

$$\text{J(z)} = \{\, Nlog(||Y - TX||^2) + ||z||^2 \,\} \text{ , όπου X = G(z)}$$

Δηλαδή η βέλτιστη εκτίμησή μας είναι :

$$\hat{z} = argmin_{\,z}\, J(z)$$

Την παραπάνω ελαχιστοποίηση της γνωστής συνάρτησης J(z) μπορούμε να την υλοποιήσουμε με τον αναδρομικό αλγόριθμο Gradient descent :

$$z_t = z_{t-1} - \mu \times \nabla_z J(z)$$

Επομένως αυτό που χρειάζεται να κάνουμε είναι αν βρούμε την κλίση $\nabla_z J(z)$.

Αρχικά ορίζουμε ως $\Phi(\text{X}) = log(||Y - TX||^2)$ και παρατηρούμε ότι
$\nabla_z J(z) = MU_o + 2Z$, όπου U_o η παράγωγος της συνάρτησης $\Phi(\text{X})$ ως προς την είσοδο (Z) του νευρονικού δικτύου.
Έχουμε δείξει στην προηγούμενη εργασία τρόπο υπολογισμού της παραγώγου μιας συνάρτησης Φ(X), όπου X η έξοδος ενός νευρονικού δικτύου, ως προς την είσοδο του νευρονικού δικτύου.

Στην συγκεκριμένη περίπτωση για να υπολογίσουμε το U_o :
$U_2 = \nabla_X\, \Phi(\text{X})$
$V_2 = U_2 \cdot f_2'(W_2)$ (πολλαπλασιασμός στοιχείο προς στοιχείο)
$U_1 = A_2^T \times V_2$
$V_1 = U_1 \cdot f_1'(W_1)$ (πολλαπλασιασμός στοιχείο προς στοιχείο)
$U_0 = A_1^T \times V_1$
Και $U_0 = \nabla_Z\, \Phi(\text{X})$

Επομένως για να υπολογίσουμε το U_0 αρκεί να υπολογίσουμε της συναρτήσεις $f_1'(x)$, $f_2'(x)$ και την κλίση $\nabla_X \Phi(\mathrm{X})$.

- $f_1'(x)$ είναι η παράγωγος της ReLU επομένως,

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

- $f_2'(x)$ είναι η παράγωγος της Sigmoid ($f_2(x) = \frac{1}{1+e^{x}}$) επομένως,

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

- $\nabla_X \Phi(\mathrm{X}) = \nabla_X \left\{ log\left(\left|\left| Y - TX \right|\right|^2 \right) \right\} = \frac{1}{||Y-TX||^2} \times \nabla_X \{||Y - TX||^2\}$
$$= \frac{1}{||Y-TX||^2} \times 2 \times (-T^{'}) \times (Y - TX)$$

,όπου Τ΄ ο ανάστροφος πίνακας του Τ.

Όπως ήταν αναμενόμενο η κλίση $\nabla_X \Phi(\mathrm{X})$ θα είναι ένα διάνυσμα μήκους 784, αφού η $\Phi(\mathrm{X})$ είναι μια βαθμωτή συνάρτηση 784 μεταβλητών.

Για να βοηθήσουμε στην σύγκλιση του αλγορίθμου κάνουμε και την κανονικοποίηση κατά Adams,η οποία όπως έχει φανεί στην πράξη βοηθάει τον αλγόριθμο να συγκλίνει γρηγορότερα, αφού αυτό που κάνει ουσιαστικά είναι ότι πάει σε κάθε αναδρομή και διαιρεί τα στοιχεία της κλίσης $\nabla_z J(z)$ με την ισχύ τους με αποτέλεσμα να τα κανονικοποιεί και ο αλγόριθμος να συγκλίνει με τον ίδιο ρυθμό για όλα τα στοιχεία του Z.

Στον Gradient descent :

$$z_t = z_{t-1} - \mu \times \nabla_z J(z)$$

χρησιμοποιήθηκε ως step-size : μ = 10^{-2}.

Ενώ στον Adams smoothing factor (1-lamda) = 0.99

Επομένως, αφού είμαστε πλέον σε θέση να υπολογίζομε την κλίση $\nabla_z J(z)$, μπορούμε να τρέξουμε τον αναδρομικό αλγόριθμο που δείξαμε παραπάνω ο οποίος είναι σε θέση να βρίσκει $\hat{Z}$ για τα οποία η συνάρτηση κόστους $J(z)$ εμφανίζει κάποιο τοπικό ελάχιστο, και έχουμε εμπιστοσύνη στο μοντέλο μας όπως είπαμε ότι για μια καλή εκτίμηση του Z , τότε το G($\hat{Z}$) θα είναι μια καλή εκτίμηση του X.

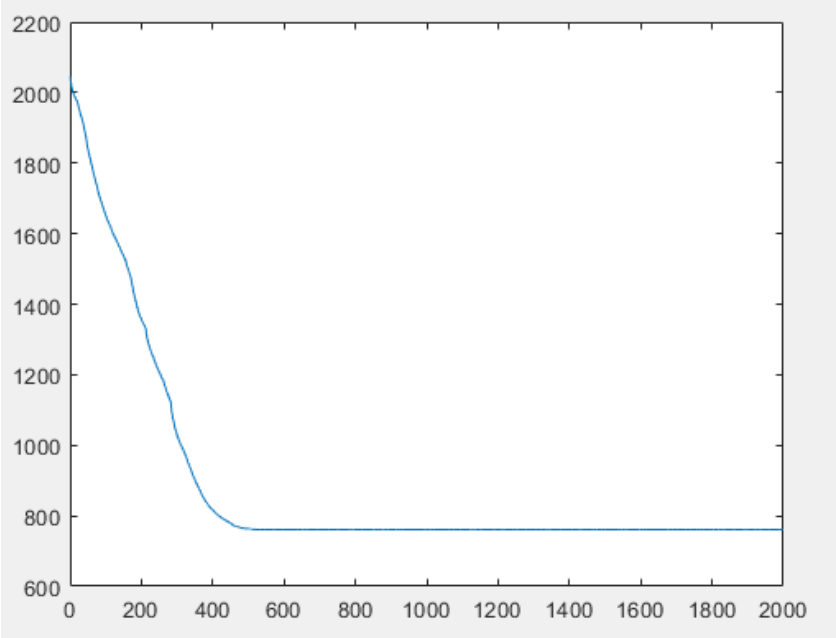
Ακολουθώντας πιστά όσα αναφέρθηκαν παραπάνω πήγαμε σε εικόνες 8αριών (εικόνες X) ανάλυσης 28×28, τους αφαιρέσαμε N σε αριθμό pixels(μετασχηματισμός T) και τους προσθέσαμε Gaussian θόρυβο(μη γνωστό). (εικόνες Y) Χρησιμοποιώντας αυτές τις μετασχηματισμένες εικόνες πήγαμε και εκτιμήσαμε για ποια είσοδο $\hat{Z}$, η έξοδος του Generator $\hat{X}$ αν την μετασχηματίσουμε με το T και της προσθέσουμε Gaussian θόρυβο θα είναι πολύ κοντά στην εικόνα Y (με την μέθοδο που δείξαμε στο μάθημα 10, όπου συνυπολογίζουμε την πληροφορία ότι z~N(0,1)). Έτσι, θα μπορούμε να έχουμε μια εκτίμηση της ιδανικής(αρχικής) εικόνας X, αφού $\hat{X} = G(\hat{Z})$.

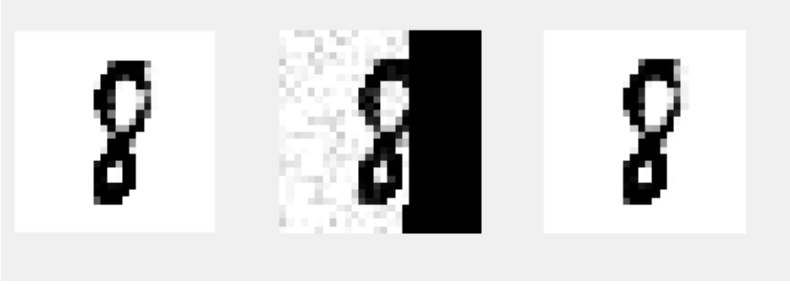
Οι αναδρομικοί αυτοί αλγόριθμοι εξαρτούνται αρκετά και από την αρχικοποίηση του z, επομένως χρειάζεται να τον τρέχουμε 20 ξεχωριστές φορές με διαφορετικές αρχικοποιήσεις για το z και να κρατήσουμε το τελικό z στο οποίο συνέκλινε και βρήκε το μικρότερο τοπικό ελάχιστο της συνάρτησης κόστους $J(z)$.

Παρακάτω παρατίθενται τα αποτελέσματα για διαφορετικό αριθμό N.

Για N = 500 :

Η ελαχιστοποίηση του κόστους (σύγκλιση αλγορίθμου)





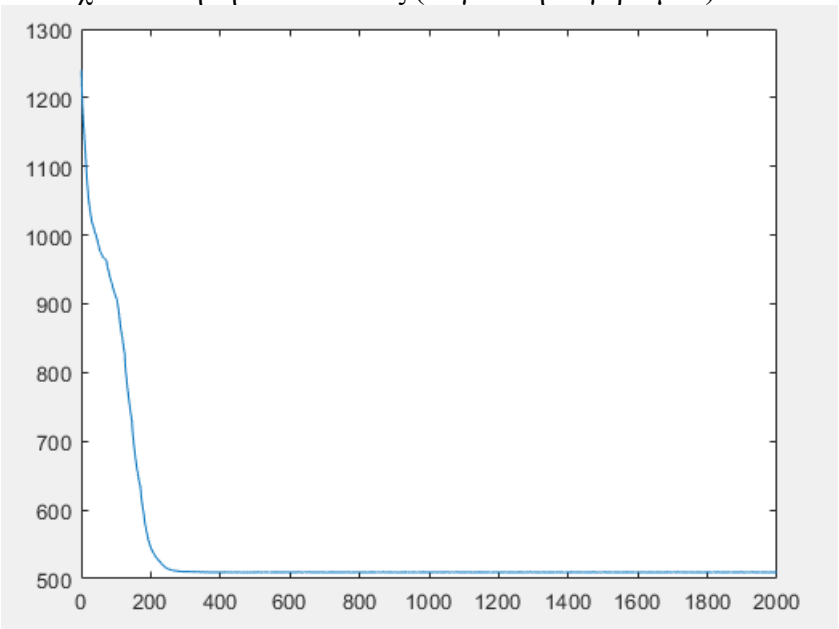
$1^{η}$ από αριστερά εικόνα είναι η ιδανική αυτή που θα θέλαμε να βρει η μέθοδος μας ιδανικά, και από την οποία προέρχεται η $2^{η}$ εικόνα.

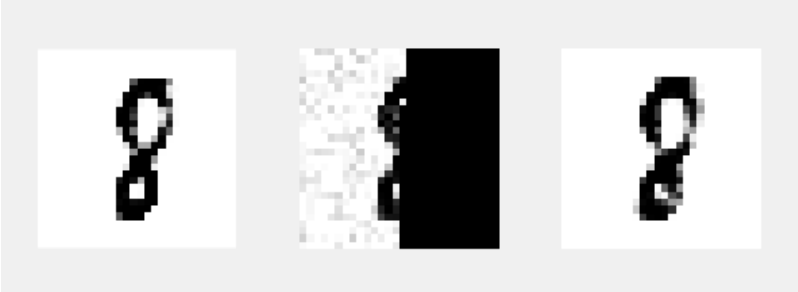
$2^{η}$ από αριστερά εικόνα είναι αυτή που έχουμε στην πραγματικότητα.

$3^{η}$ από αριστερά εικόνα αυτή που βρίσκει τελικά η μέθοδος.

Για N = 400 :

Η ελαχιστοποίηση του κόστους (σύγκλιση αλγορίθμου)

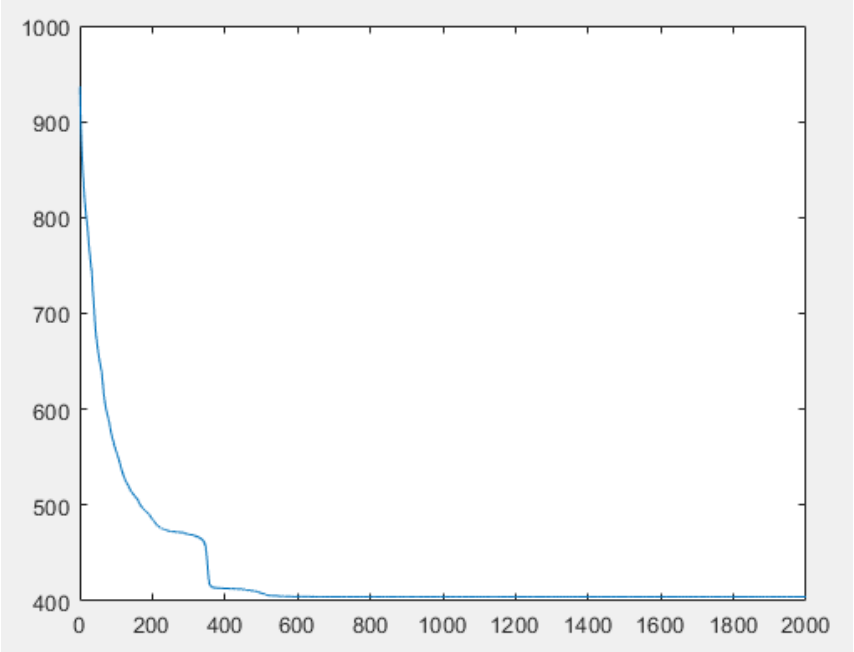




(η στοίχιση των εικόνων είναι η ίδια με πριν)

Για N = 350 :

Η ελαχιστοποίηση του κόστους (σύγκλιση αλγορίθμου)

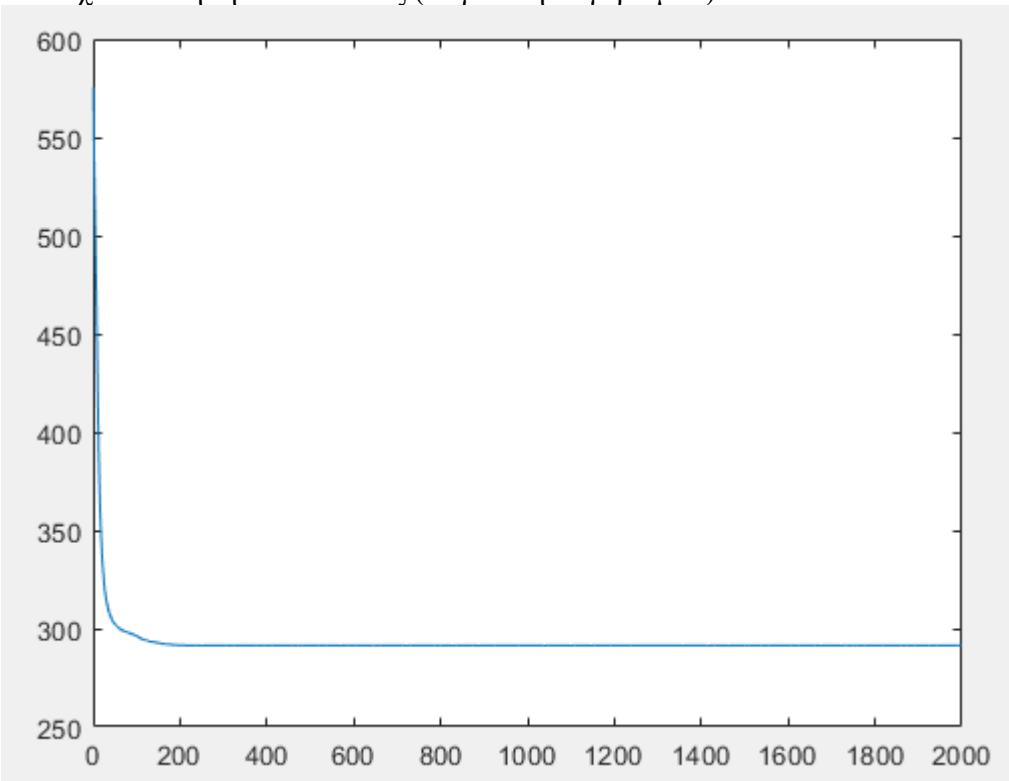


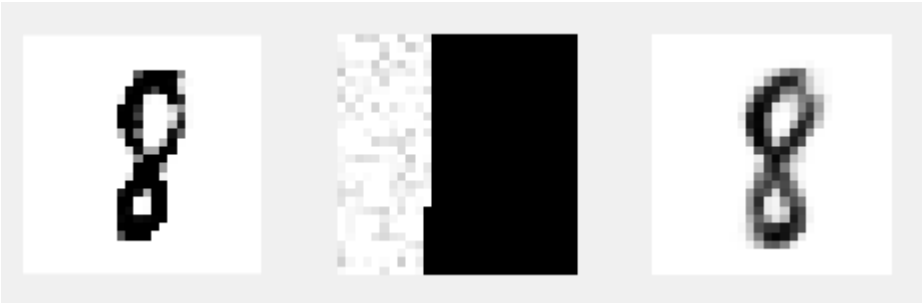
(η στοίχιση των εικόνων είναι η ίδια με πριν)

Φοβερά εντυπωσιακό το πόση λίγη πληροφορία γνωρίζουμε για το ιδανικό 8αρι και παρόλαυτα καταφέρνουμε να βρούμε είσοδο για τον generator η οποία μας δίνει ως έξοδο ένα 8αρι με μεγάλο βαθμό ομοιότητας με τον ιδανικό.

Για N = 300 :

Η ελαχιστοποίηση του κόστους (σύγκλιση αλγορίθμου)

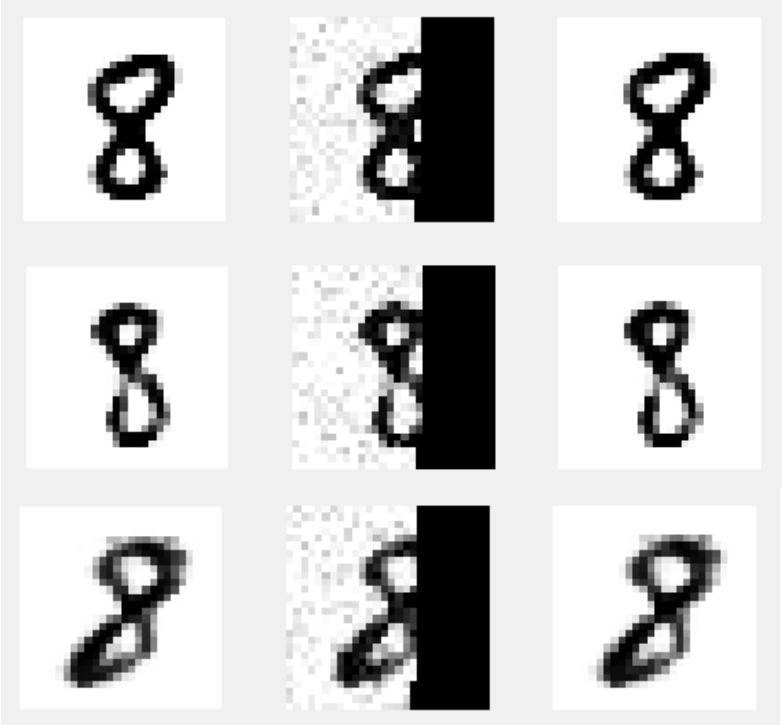




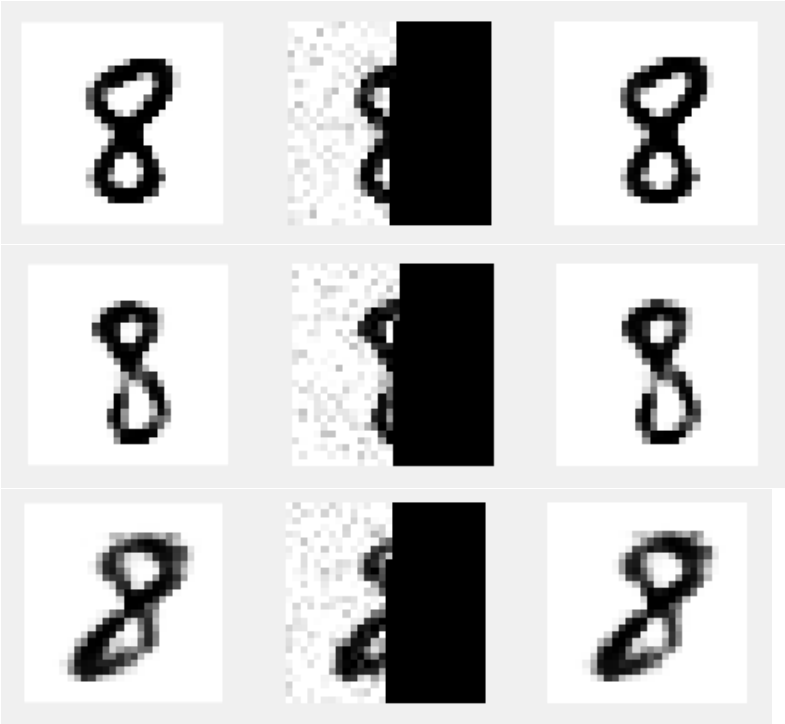
(η στοίχιση των εικόνων είναι η ίδια με πριν)
Στο συγκεκριμένο 8άρι η πληροφορία που έχουμε εδώ για $N \leq 300$ είναι πρακτικά μηδενική για αυτό και τα αποτελέσματα πλέον είναι άσχημα.

Παρακάτω παραθέτω τα οπτικά αποτελέσματα και για τα **υπόλοιπα τρία** 8άρια που μας δόθηκαν.

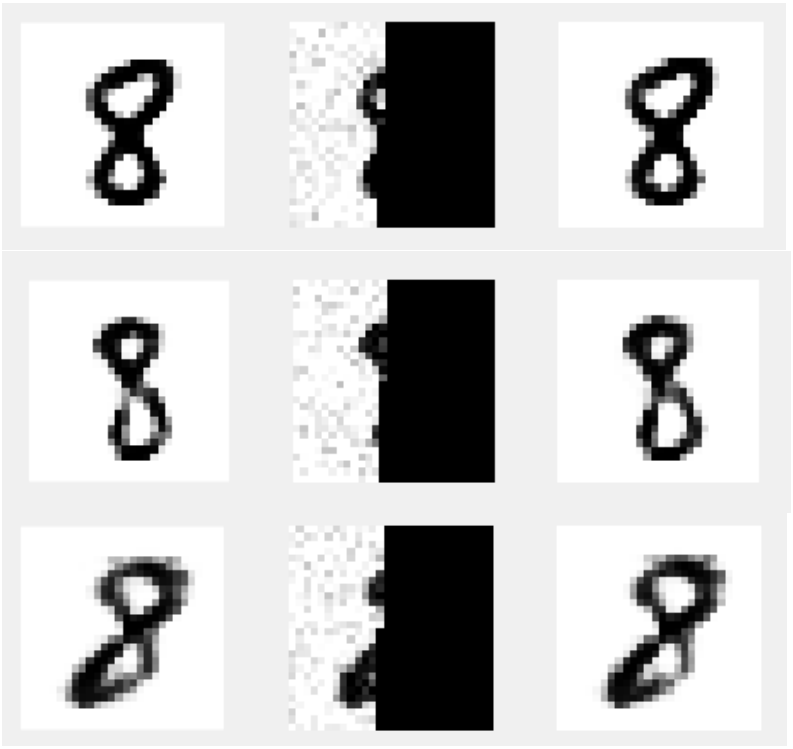
N = 500 :



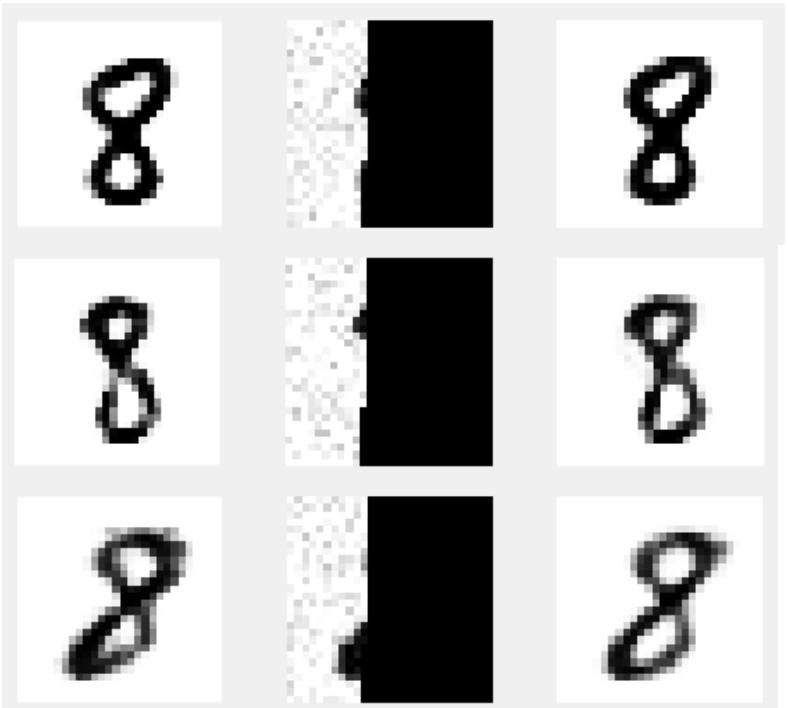
N = 400 :



<u>N = 350</u> :

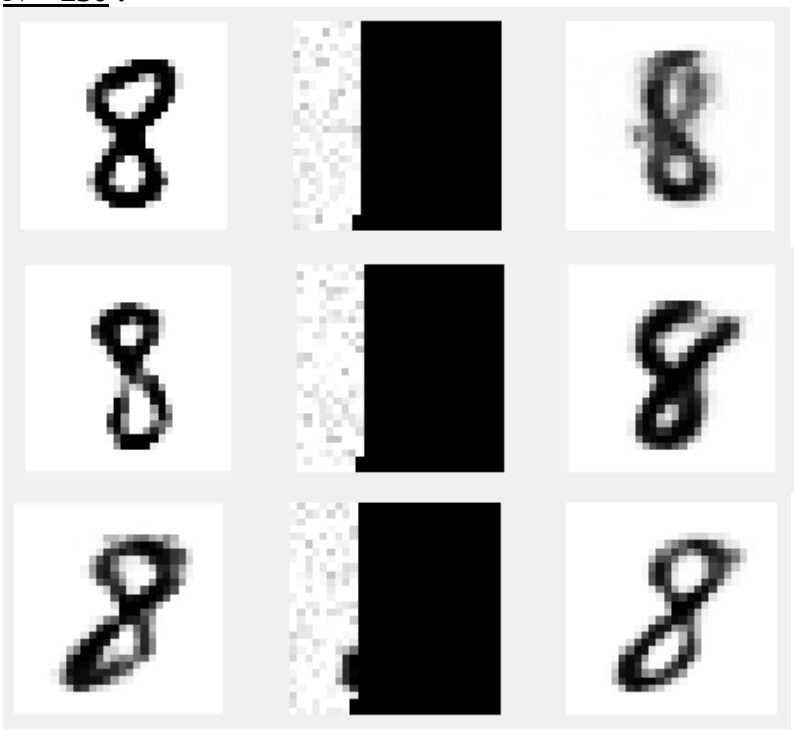


<u>N = 300</u> :



(Σε αυτή την περίπτωση έχουμε στην διάθεση μας 300 pixel και παρόλαυτα καταφέρνουμε να βρούμε μια σχετικά καλή εκτίμηση της αρχικής εικόνας.
Αφού τα αποτελέσματα για τα συγκεκριμένα δαρια φαίνονται ¨κάπως ικανοποιητικά¨ ας δοκιμάσουμε και για N = 250.

<u>N = 250</u> :



(εδώ πλέον η πληροφορία που έχουμε για τα δάρια είναι πρακτικά μηδενική οπότε είναι αναμενόμενο και τα αποτελέσματα να είναι πολύ άσχημα)
Μικρή εξαίρεση αποτελεί το τελευταίο δάρι για το οποίο φαίνεται πως κάποια πληροφορία του σχήματος του (κάτω αριστερά) ¨γλιτώνει¨ από τον μετασχηματισμό T με αποτέλεσμα να μας είναι διαθέσιμη και για αυτό να έχουμε ένα κάπως πιο αποδεκτό αποτέλεσμα, σε σχέση με τα προηγούμενα δάρια των οποίων όλη η πληροφορία του σχήματος τους έχει χαθεί.

Πρόβλημα 3

Το πρόβλημα αυτό μοιάζει πολύ με το 2ο πρόβλημα. Η διαφορά είναι πως εδώ έχουμε ιδανικές εικόνες (X) 8αριών διάστασης 28×28, τις μετασχηματίζουμε κατάλληλα, μειώνοντας την ανάλυση τους, παίρνουμε εικόνες (Y) διάστασης 7×7 και τους προσθέτουμε Γκαουσιανό θόρυβο. Έχοντας τις εικόνες Y εμείς ψάχνουμε να βρούμε από ποιες εικόνες X προέρχονται(**ισοδύναμα** μπορούμε να πούμε ότι θέλουμε να αυξήσουμε την ανάλυση τους). Πρόκειται δλδ για το ίδιο πρόβλημα με πριν, απλώς εδώ διαφέρει ο μετασχηματισμός T, που προκαλεί την μείωση της ανάλυσης της εικόνας, τον οποίο πρέπει να βρούμε.

Για να φανταστούμε τον μετασχηματισμό μπορούμε να φανταστούμε την εικόνα X διάστασης 28×28 να χωρίζεται σε ένα πλέγμα, όπου κάθε τετράγωνο του πλέγματος είναι διάστασης 4×4 και τα αντίστοιχα 16 pixel να αντικαθίστανται από ένα pixel που έχει τιμή τον μέσο όρο των τιμών των 16 pixel.

Η εικόνα X την εκφράζουμε όπως και πριν ως ένα διάνυσμα μήκους 784 (28×28) και την τελική μετασχηματισμένη εικόνα Y ως διάνυσμα μήκους 49 (7×7).

$Y = T \times X + W$,
άρα ο μετασχηματισμός T θα πρέπει να είναι ένας πίνακας διάστασης 49×784.

Για να βρω την μορφή του πίνακα T, αρχικά προσπάθησα να βρω την μορφή του σε ένα πρόβλημα με εικόνες μικρότερων διαστάσεων, και στη συνέχεια να το γενικεύσω για τις εικόνες των διαστάσεων που έχουμε στο πρόβλημα 3.

Επομένως έστω μια εικόνα ανάλυσης 4×4 :

$$\left[\begin{array}{cc|cc} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \hline \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{array}\right]$$

(Τις υπο-περιοχές του πίνακα τις σχεδίασα απλώς για να φαίνεται το πλέγμα το οποίο θα μας ρίξει την ανάλυση από 4×4 σε 2×2).

Τον οποίο τον γράφουμε ως ένα διάνυσμα X = $\begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \\ \alpha_{41} \\ \vdots \\ \alpha_{14} \\ \alpha_{24} \\ \alpha_{34} \\ \alpha_{44} \end{bmatrix}$ (μήκους 16)

Είναι εύκολο εδώ να δούμε πως ο πίνακας T που πρέπει να πολλαπλασιάσουμε το διάνυσμά X ώστε να προκύψει το διάνυσμα Y το οποίο θα είναι μήκους 4 και θα αντιστοιχεί σε εικόνα ανάλυσης 2×2 είναι :

$$T = \left[\begin{array}{cccccccc|cccccccc} \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} \end{array}\right]$$

Ο πίνακας T μπορούμε να διακρίνουμε ότι αποτελεί έναν **<u>σύνθετο</u> διαγώνιο πίνακα** διάστασης 2×2 (όσο και η διάσταση της εικόνας Y), όπου μάλιστα τα στοιχεία(πίνακες) τις διαγωνίου του **είναι ίσα**.

Έστω Tdiag ο πίνακας που αποτελεί ένα από τα στοιχεία της διαγωνίου του σύνθετου πίνακα.
Τότε ο Tdiag είναι διάστασης ($\frac{\mu\acute{\eta}κος\ του\ Y}{2} \times \frac{\mu\acute{\eta}κος\ του\ X}{2} = 2 \times 8$)
*(το 2 στον παρονομαστή προκύπτει από την διάσταση της εικόνας Y (εδώ 2×2)
Και τα μη μηδενικά στοιχεία του Tdiag είναι ίσα με $\frac{1}{4}$ (όπου το 4 προκύπτει από το πλήθος των στοιχείων που περιέχονται στα τετράγωνα του πλέγματος)

Τώρα, αυτή την λογική εύρεσης του μετασχηματισμού T μπορούμε να την μεταφέρουμε και στο πρόβλημα 3.

Όπου X διάνυσμα μήκους 784 και Y διάνυσμα μήκους 49

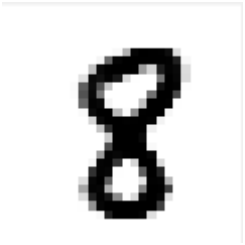
Εδώ ο T θα είναι ένας <u>σύνθετος διαγώνιος πίνακας</u> διάστασης (7×7) , όπου τα στοιχεία (πίνακες) τις διαγωνίου του θα είναι <u>ίσα</u> μεταξύ τους.
Αρκεί λοιπόν να βρούμε αυτά τα στοιχεία(πίνακες) της διαγωνίου του σύνθετου πίνακα T για να βρούμε ολόκληρο τον πίνακα.

Αν ονομάσουμε όπως και πριν Tdiag τον πίνακα που αποτελεί στοιχείο της διαγωνίου του σύνθετου πίνακα, τότε ο Tdiag θα είναι πίνακας διάστασης
$\frac{\mu\acute{\eta}κος\ του\ Y}{7} \times \frac{\mu\acute{\eta}κος\ του\ X}{7} = 7 \times 112$

Και τα μη μηδενικά στοιχεία του θα είναι ίσα με $\frac{1}{16}$.
Και όσο αφορά σε ποιες θέσεις του πίνακα τα στοιχεία θα είναι μη μηδενικά αρκεί να ακολουθήσουμε για το συγκεκριμένο παράδειγμα τον ¨αλγόριθμο που μας δίνετε¨, στις υποδείξεις του προβλήματος 3, για τις πρώτες 2 γραμμές του πίνακα T, απλά εμείς να τον εφαρμόσουμε για τον πίνακα Tdiag και να τον προεκτείνουμε για όλες, και τις 7 δηλαδή, γραμμές του.

Έτσι βρίσκοντας τον Tdiag τον αντικαθιστούμε στην διαγώνιο του πίνακα T και τελικά ο πίνακας T καταλήγει, όπως αναμενόταν, διάστασης 49×784.

Έτσι αντί για εικόνες X διάστασης 28×28 πχ :



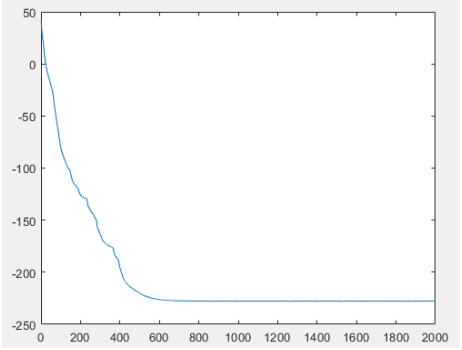
Έχουμε εικόνες Y διάστασης 7×7 πχ :

Και θέλουμε ουσιαστικά όπως και στο πρόβλημα 2 να βρούμε μια καλή εκτίμηση της εικόνας από την οποία έχει προέλθει η εικόνα χαμηλής ανάλυσης.

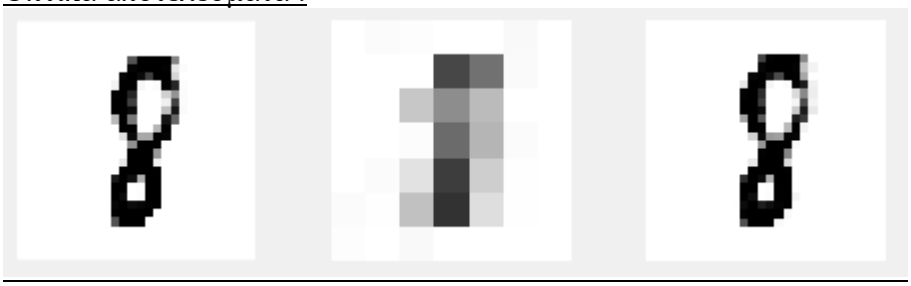
Όπως και στο πρόβλημα 2 εμείς έχουμε εικόνες Y(που έχουν προέλθει από εικόνες X, τις οποίες δεν έχουμε) και εμείς αντί να κάνουμε εκτίμηση της εικόνας X, κάνουμε εκτίμση της εισόδου Z ενός generator η οποία θα μας δώσει μια έξοδο $\hat{X}$ η οποία θέλουμε να είναι όσο πιο κοντά γίνεται στην εικόνα X.

Το πρόβλημα αυτό μαθηματικά είναι ακριβώς το ίδιο με το πρόβλημα 2, επομένως εφαρμόζουμε ακριβώς την ίδια μέθοδο που αναφέραμε και εκεί.

Παρακάτω παραθέτω τα αποτελέσματα :
(σύγκλιση αλγορίθμου για πρώτη εικόνα)

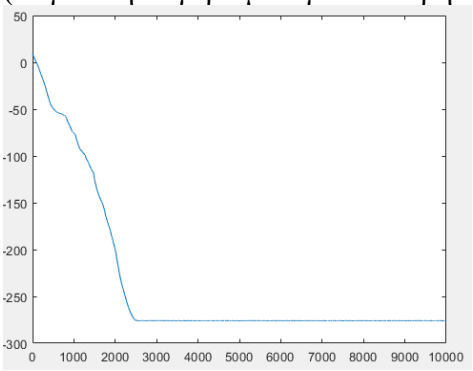


Οπτικά αποτελέσματα :

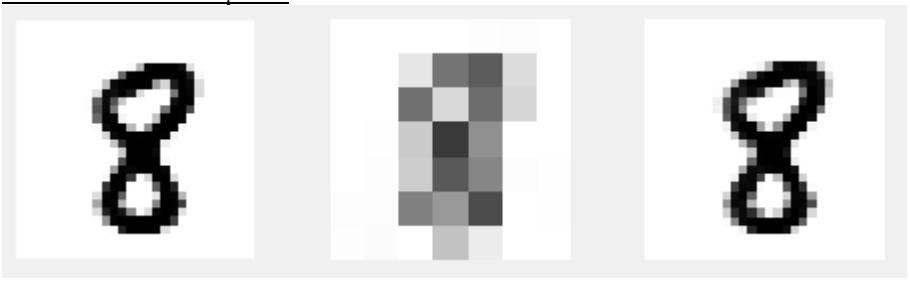


Η μεσαία εικόνα είναι η εικόνα την οποία εμείς έχομε και έχει προέλθει από την εικόνα στα αριστερά της. Η δεξιά εικόνα είναι η εικόνα μου καταφέραμε και βρήκαμε.

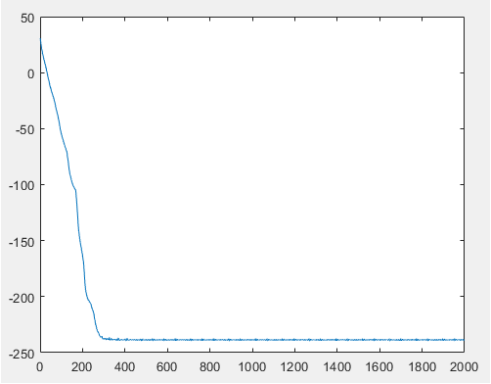
(σύγκλιση αλγορίθμου για δεύτερη εικόνα)



Οπτικά αποτελέσματα :

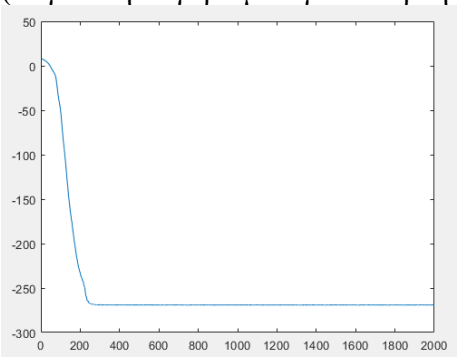


(σύγκλιση αλγορίθμου για τρίτη εικόνα)



Οπτικά αποτελέσματα :

(σύγκλιση αλγορίθμου για τέταρτη εικόνα)



Οπτικά αποτελέσματα :

Πολύ εντυπωσιακό πως από 49 μόνο pixel όπου τα 8άρια είναι πρακτικά μη αναγνωρίσιμα από το ανθρώπινο μάτι, καταφέρνει η μέθοδος και βρίσκει είσοδο για τον generator η οποία αποδίδει έξοδο απίστευτα κοντά στο πραγματικό 8άρι από το οποίο έχει προέλθει το ¨πιξελιασμένο¨ 8άρι.

(Κάτι επιπλέον σχετικά με την πρώτη άσκηση)

Κύριε Μουστακίδη διεκπεραιώνοντας την παρών Σετ ασκήσεων συνειδητοποίησα πως στο πρώτο σετ δεν σας είχα συμπεριλάβει στην αναφορά τις παραγώγους που είχα υπολογίσει για τις διαφορετικές μεθόδους (Cross Entropy, Exponential, Hinge) εκπαίδευσης των νευρονικών δικτύων.
Επομένως σας τις συμπεριλαμβάνω εδώ (ακόμη και αν δεν τις συνυπολογίσετε) :

Για την μέθοδο Cross Entropy $\nabla_Y \{\Phi(Y) + \Psi(Y)\} = \nabla_Y \{-\log(1-Y) \text{ - } log(Y)\} = = \frac{1}{1-Y} - \frac{1}{Y}$

Για την μέθοδο Exponential $\nabla_Y \Phi(Y) + \Psi(Y) = \nabla_Y \{e^{0.5Y} + e^{-0.5Y}\} = = 0.5e^{0.5Y} - 0.5e^{-0.5Y}$

Για την μέθοδο Hinge $\nabla_Y \Phi(Y) + \Psi(Y) = \nabla_Y \{\max(1+Y, 0) + max(1-Y, 0)\}$

Επομένως $\nabla_Y \{\Phi(Y) + \Psi(Y)\} = \begin{cases} 1 & x > 1 \\ 0 & -1 \le x \le 1 \\ -1 & x < -1 \end{cases}$

Για την υλοποίηση του παρώντος σετ ασκήσεων χρησιμοποιήθηκε ο παρακάτω κώδικας σε matlab.

Πρόβλημα 1 :

```
close all;
clear ;
clc;
load('data21.mat');

X2D_100 = zeros(280,280);
for j = 0:9
    for i = 0:9

    Z0 = randn(10,1); % Gaussian , mean 0 , covariance 1
    W1 = A_1 * Z0 + B_1;
    Z1 = max(W1,0);        % ReLU
    W2 = A_2 * Z1 + B_2;
    X = 1./(1 + exp(W2));  % Sigmoid

    X2D = reshape(X,28,28);
    X2D_100(28*j+1:28*j+28,28*i+1:28*i+28) = X2D;
    end
end
imshow(X2D_100);
```

Πρόβλημα 2 :

```
close all;
clear ;
clc;
load('data21.mat');
load('data22.mat');

N = 250; % number of pixels that we keep
T = [eye(N) zeros(N,784-N)]; % trasformation

n = 4; % number of X_n's collumn(image) that is used

X_ideal = X_i(:,n);
X_ideal_2D = reshape(X_ideal,28,28);

X_given = X_n(1:N,n);
X_given = [X_given;zeros(784-N,1)];
X_given_2D = reshape(X_given,28,28);


% Gradient Descent's Parameters
iter = 2*1e3; % number of algorithm's iterations
m = 1e-2; % step-size
L = 0.1; % smoothing factor(on Adam)
C = 1e-20; % a really small number to avoid devision with 0(on adam)


Best_Cost = Inf(1,iter);

for j = 1:10

    Cost = zeros(1,iter);

    Z0 = randn(10,1); % Gaussian input , mean 0 , covariance 1

    for i = 1:iter
        Cost1 = 0;
        Cost2 = 0;
        norm2 = 0;

        W1 = A_1 * Z0 + B_1;
        Z1 = max(W1,0);        % ReLU
        W2 = A_2 * Z1 + B_2;
        X = 1./(1 + exp(W2)); % Sigmoid given

        X_gan_tranformed = T*X;

        % calculation of main cost

        for k = 1:N
            norm2 = norm2 + (X_given(k) - X_gan_tranformed(k))^2;
            Cost1 = N*log(norm2);
        end

        for d = 1:10
            Cost2 = Cost2 + Z0(d)^2;
        end
        Cost(i) = Cost1 + Cost2;
```

```
        U2 = -2*T'*(X_n(1:N,n) - T*X)/norm2; % gradient of log((norm(Xn-
T*X))^2) with respect of X
        V2 = U2 .* diff_sigmoid(W2);
        U1 = A_2' * V2;
        V1 = U1 .* diff_Relu(W1);
        U0 = A_1' * V1;     %    gradient of log((norm(Xn-T*X))^2) with
respect of Z

        diff_Cost = N*U0 + 2*Z0;


        if (i==1 && j==1)
            P = diff_Cost .^ 2;  % initiation of Adams normalization
        else
            P = (1-L) * P_old + L * (diff_Cost).^2;
        end

        P_old = P;

        Z0 = Z0 - m * diff_Cost ./ sqrt(C + P);

    end

    if Best_Cost(iter) > Cost(iter)
        Z0_Best = Z0;
        Best_Cost = Cost;
    end

end


W1 = A_1 * Z0_Best + B_1;
Z1 = max(W1,0);        % ReLU
W2 = A_2 * Z1 + B_2;
X = 1./(1 + exp(W2)); % Sigmoid given

X_gan_finds = X;
X_gan_finds_2D = reshape(X_gan_finds,28,28);
figure;
plot(Best_Cost);

figure;
subplot(1,3,1);
imshow(X_ideal_2D)
subplot(1,3,2);
imshow(X_given_2D);
subplot(1,3,3);
imshow(X_gan_finds_2D);
```

Πρόβλημα 3 :

```
close all;
clear ;
clc;
load('data21.mat');
load('data23.mat');

N = 49; %proccesed image's number of pixel (7x7resolution)
% Modeling of the Trasformation Array (T)
T = zeros(49,784);
Tdiag = zeros(7,112);
count = 0;

for i = 1:7
    for j = 0:3
        for k = 1+count:4+count
            Tdiag(i,j*28+k) = 1/16;
        end
    end
    count = k;
end

for i = 0:6

    T(i*7+1 :(i+1)*7 , i*112+1 :(i+1)*112) = Tdiag;

end

n = 4; % number of X_n's collumn(image) that is used
X_ideal = X_i(:,n);
X_ideal_2D = reshape(X_ideal,28,28);

X_given = X_n(:,n);
X_given_2D = reshape(X_given,7,7);
X_given_2D = kron(X_given_2D,ones(4,4));


% Gradient Descent's Parameters
m = 1e-2; % step-size
L = 0.1; % smoothing factor
C = 1e-20; % a really small number to avoid devision with 0
iter = 2*1e3;


Best_Cost = Inf(1,iter);

for j = 1:10

    Cost = zeros(1,iter);

    Z0 = randn(10,1); % Gaussian input , mean 0 , covariance 1

    for i = 1:iter
        Cost1 = 0;
        Cost2 = 0;
        norm2 = 0;

        W1 = A_1 * Z0 + B_1;
```

```
        Z1 = max(W1,0);        % ReLU
        W2 = A_2 * Z1 + B_2;
        X = 1./(1 + exp(W2)); % Sigmoid given

        X_gan_tranformed = T*X;

        % calculation of main cost

        for k = 1:N
            norm2 = norm2 + (X_given(k) - X_gan_tranformed(k))^2; % norm2 =
(second_norm)^2
            Cost1 = N*log(norm2);
        end

        for d = 1:10
            Cost2 = Cost2 + Z0(d)^2;
        end
        Cost(i) = Cost1 + Cost2;


        U2 = -2*T'*(X_n(1:N,n) - T*X)/norm2; % gradient of log((norm(Xn-
T*X))^2) with respect of X
        V2 = U2 .* diff_sigmoid(W2);
        U1 = A_2' * V2;
        V1 = U1 .* diff_Relu(W1);
        U0 = A_1' * V1;            % Gradient of log((norm(Xn-T*X))^2) with
respect of Z

        diff_Cost = N*U0 + 2*Z0;   % Gradient of J(z)


        if (i==1 && j==1)
            P = diff_Cost .^ 2;   % initiation of Adams normalization
        else
            P = (1-L) * P_old + L * (diff_Cost).^2;
        end

        P_old = P;

        Z0 = Z0 - m * diff_Cost ./ sqrt(C + P);

    end

    if Best_Cost(iter) > Cost(iter)
        Z0_Best = Z0;
        Best_Cost = Cost;
    end

end


W1 = A_1 * Z0_Best + B_1;
Z1 = max(W1,0);        % ReLU
W2 = A_2 * Z1 + B_2;
X = 1./(1 + exp(W2)); % Sigmoid given

X_gan_finds = X;
X_gan_finds_2D = reshape(X_gan_finds,28,28);
figure;
```

```
plot(Best_Cost);

figure;
subplot(1,3,1);
imshow(X_ideal_2D)
subplot(1,3,2);
imshow(X_given_2D);
subplot(1,3,3);
imshow(X_gan_finds_2D);
```