Εργαστηριακή Άσκηση Υπολογιστικής Νοημοσύνης
Μέρος Α

Ονοματεπώνυμο: Νάνος Νικόλαος

ΑΜ:1054386

Έτος:4ο

Για την υλοποίηση της άσκηση επέλεξα την γλώσσα προγραμματισμού Python και χρησιμοποίησα το περιβάλλον tensforflow για τη δημιουργία-εκπαίδευση-έλεγχο του νευρωνικού δικτύου που ζητήται . Επιπλέον χρησιμοποίησα και βιβλιοθήκες όπως numpy , pandas , sci-kit learn, matlib και άλλες. Τέλος για την επίτευξη της άσκησης αναπτύχθηκαν τα script : load_movielens_data.py,model_module.py, run_training_evaluation

.py.

Το url του git repository στο οποίο έχω τα παραπάνω scripts και επαρκή σχολιασμό για αυτά είναι : https://github.com/nnanos/Neural_net_code.git

Α1. Προεπεξεργασία και Προετοιμασία δεδομένων

Τα ερωτήματα του Α1 εξετάζονται με συναρτήσεις ενεργοποίησης relu και sigmoid σε νευρωνικό με 1 κρυφό επίπεδο το οποίο έχει 20 νευρώνες με early stopping

α)Με μερικά τρεξίματα του script run_training_evaluation.py διαπίστωσα από τα losses ότι με την τεχνική του κεντραρίσματος παίρνω λίγο καλύτερα αποτελέσματα. Το νέο διάστημα τιμών είναι [-3.724137931034483, 3.508045977011494].

β)

- Με centering

Με συμπλήρωση μηδενικών έχουμε:

Relu:

```
cross validated score (RMSE): 0.2590590421328874

holdout score (RMSE): 0.25698257459456897
```

sigmoid:

```
cross validated score (RMSE): 0.2556957021813585

holdout score (RMSE): 0.2548253219531501
```

Με συμπλήρωση τυχαίων τιμών έχουμε:

Relu:

```
cross validated score (RMSE): 1.0027998759220533

holdout score (RMSE): 1.0036809524704455
```

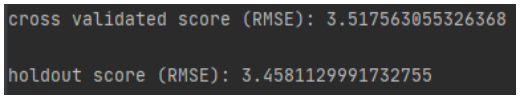
sigmoid:

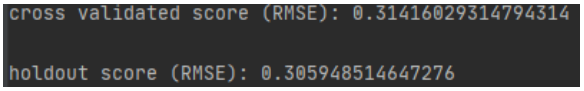
```
cross validated score (RMSE): 0.16199199805602887

holdout score (RMSE): 0.16187221028423165
```

Με συμπλήρωση του μέσου όρου του διανύσματος αξιολόγησης έχουμε:

Relu:

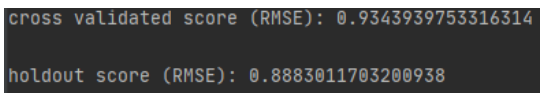


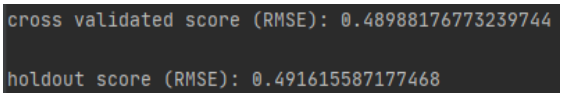


- Δίχως centering

Με συμπλήρωση μηδενικών έχουμε:

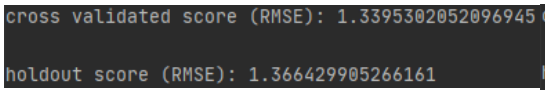
Relu: sigmoid:

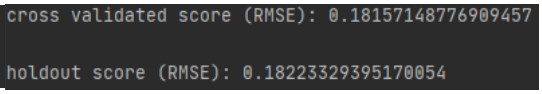




Με συμπλήρωση τυχαίων τιμών έχουμε:

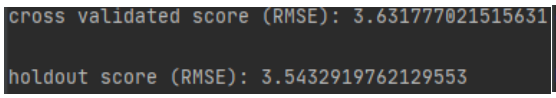
Relu: sigmoid:

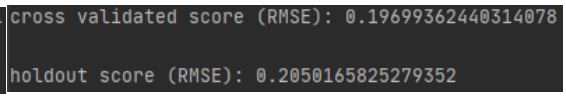




Με συμπλήρωση του μέσου όρου του διανύσματος αξιολόγησης έχουμε:

Relu: sigmoid:





Για την πρόβλεψη των αξιολογήσεων νέων χρηστών θα ήταν καλύτερο να συμπληρώσουμε τα missing values με τον μέσο όρο του αντίστοιχου διανύσματος αξιολόγησης. Με αυτόν τον τρόπο σίγουρα η εκπαίδευση του δικτύου θα δυσκολευτεί περισσότερο (από το να βάζαμε 0) αλλά πιστεύω οι προβλέψεις μας θα είναι πιο αντικειμενικές.

γ)Εάν χρησιμοποιήσω την Relu δεν χρειάζεται κάποιο rescaling στα δεδομένα μου(διότι είναι γραμμική) ενώ όταν χρησιμοποιώ την σιγμοειδή κανονικοποιώ τα δεδομένα μου στο διάστημα [0,1] κάνοντας χρήση του MinMaxScaler και έπειτα εάν θέλω να πάρω πίσω τις πραγματικές προβλέψεις χρησιμοποιώ την συνάρτηση inverse_tranform του scaler που δημιούργησα.

δ)Η διαδικασία του 5 fold cross-validation γίνεται στο for loop του script run_training_evaluation.py . Αρχικά χωρίζω τους one_hot_encoded_users και το

προεπεξεργαζμένο user-movie matrix σε ένα main και holdout set έτσι ώστε να γίνει το cross-validation στο main set και το τελικό testing στο holdout set.Επιπροσθέτως σε κάθε fold κρατάω και ένα validation set έτσι ώστε να δουλέψει το early stopping.

Α2. Επιλογή αρχιτεκτονικής

α)Η μετρική RMSE δίνει μεγαλύτερο βάρος σε μεγάλα errors δηλαδή εάν πάρουμε ένα μεγάλο error από έναν outlier τότε αυτό θα υψωθεί στο τετράγωνο με αποτέλεσμα να πάρουμε μεγαλύτερο loss από ότι θα πέρναμε εάν χρησιμοποιούσαμε MAE καθώς σε αυτό error είναι απλά η απόλυτη τιμή της διαφοράς της επιθυμητής από την πραγματική τιμή. Επομένως η χρησιμοποίηση MAE για την αξιολόγηση του μοντέλου μας θα δώσει μεγαλύτερη διαίσθηση για το μέσο σφάλμα που κάνει στην πρόβλεψη αξιολογήσεων των άγνωστων χρηστών. Ενώ εάν το RMSE γίνει μεγάλο τότε δεν έχουμε διαίσθηση για το αν το μέσο σφάλμα είναι μεγάλο ή αν αυτό προκλήθηκε από κάποιους outliers.Άρα θεωρώ ότι θα πρέπει να χρησιμοποιούνται και οι δύο μετρικές. Ωστόσο στο πρόβλημά μας τα σφάλματα είναι μικρότερα του ένα καθώς χρησιμοποιώ σιγμοειδή (τιμές [0,1]) οπότε το RMSE τις περισσότερες φορές θα είναι μικρότερο του MAE.

β)Στην είσοδο του νευρωνικού δικτύου ο κάθε χρήστης θα αναπαρίσταται ως ένα one_hot διάνυσμα το οποίο θα έχει διάσταση 943 (όσοι και οι χρήστες) όπου ο i-οστος χρήστης θα είναι αυτός του οποίου το διάνυσμα στην i-οστη θέση έχει την τιμή 1 και όλες τις άλλες 0.Θεωρώ πως η φάση της εκπαίδευσης το νευρώνικό θα αποδώσει καλύτερα αποτελέσματα εάν χρησιμοποιήσουμε one_hot διανύσματα καθώς θα είναι πιο εύκολο(θα χρειαστούν λιγότερες εποχές) για αυτό να ταυτοποιήσει ποιος χρήστης είναι έναντι της δυαδικής κωδικοποίησης .

γ)Στην έξοδο θα χρειαστούμε 1682 νευρώνες όσες δηλαδή και οι αξιολογήσεις του κάθε χρήστη.

δ)Οι τιμές των νευρώνων του κρυφού επιπέδου θα είναι όσο η τιμή του βάρους που συνδέει το χρήστη(αφού αυτή η τιμή είναι 1 και οι άλλες 0).Επομένως θα επιλέξω τη σιγμοειδή συνάρτηση η οποία ορίζεται σε όλους τους πραγματικούς διότι δεν ξέρω κάθε φορά τι τιμή θα έχει το αντίστοιχο βάρος.

ε)Επέλεξα σιγμοειδή συνάρτηση ενεργοποίησης και χρησιμοποίησα τον minmaxscaler για να κάνω το μετασχηματισμό των δεδομένων του δοθέντος dataset.Εάν θελήσω να δώ τη πραγματική πρόβλεψη του νευρωνικού τότε εφαρμόζω τον αντίστροφο μετασχηματισμό στην έξοδο.

στ)

#κόμβων=20:

Για το training χρησιμοποιήθηκαν οι μετρικές mse κ’ mae. Στα παρακάτω γραφήματα απεικονίζονται τα cross-validated mse κ’ mae:

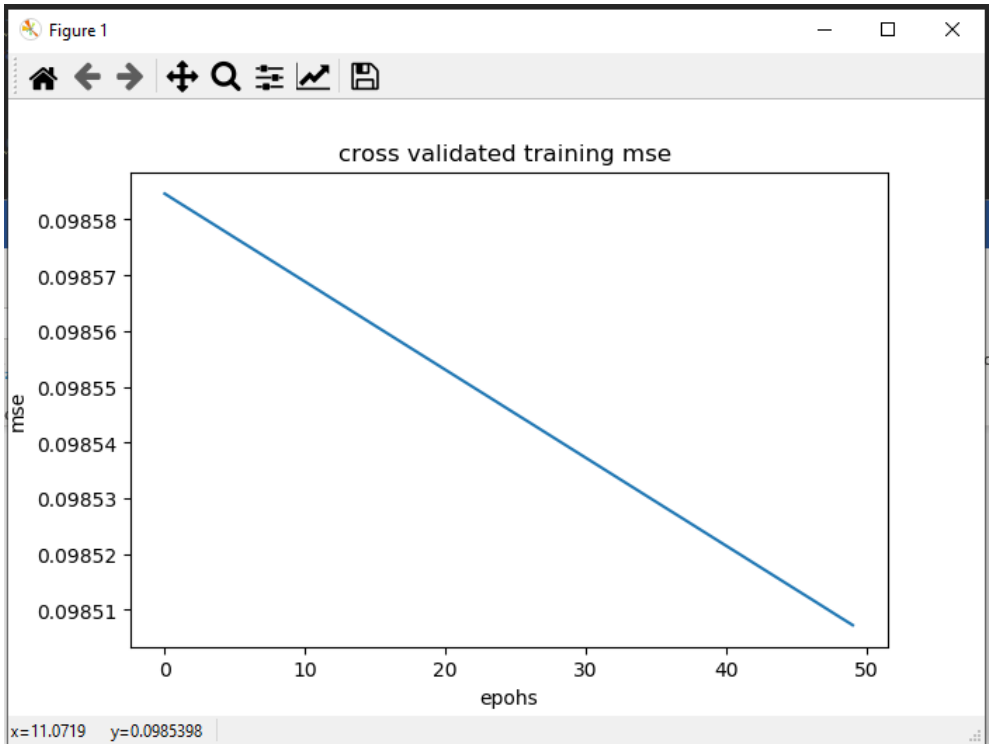


Figure 1
cross validated training mse
0.09858
0.09857
0.09856
0.09855
0.09854
0.09853
0.09852
0.09851
mse
0
10
20
30
40
50
epohs
x=11.0719 y=0.0985398

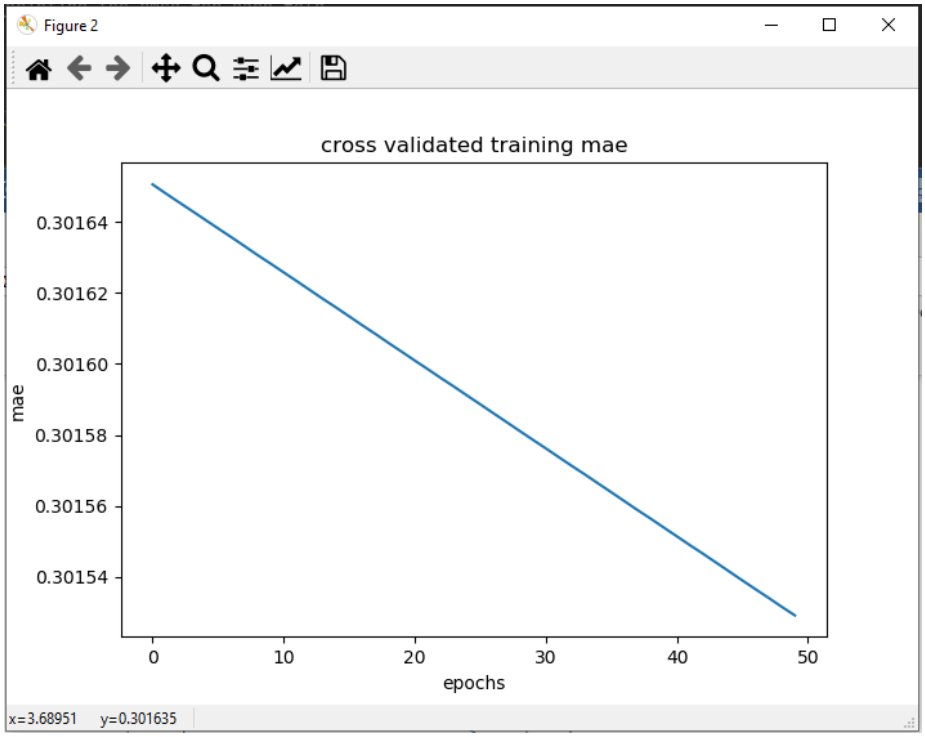


Figure 2
cross validated training mae
0.30164
0.30162
0.30160
0.30158
0.30156
0.30154
mae
0
10
20
30
40
50
epochs
x=3.68951 y=0.301635

#κόμβων=10:

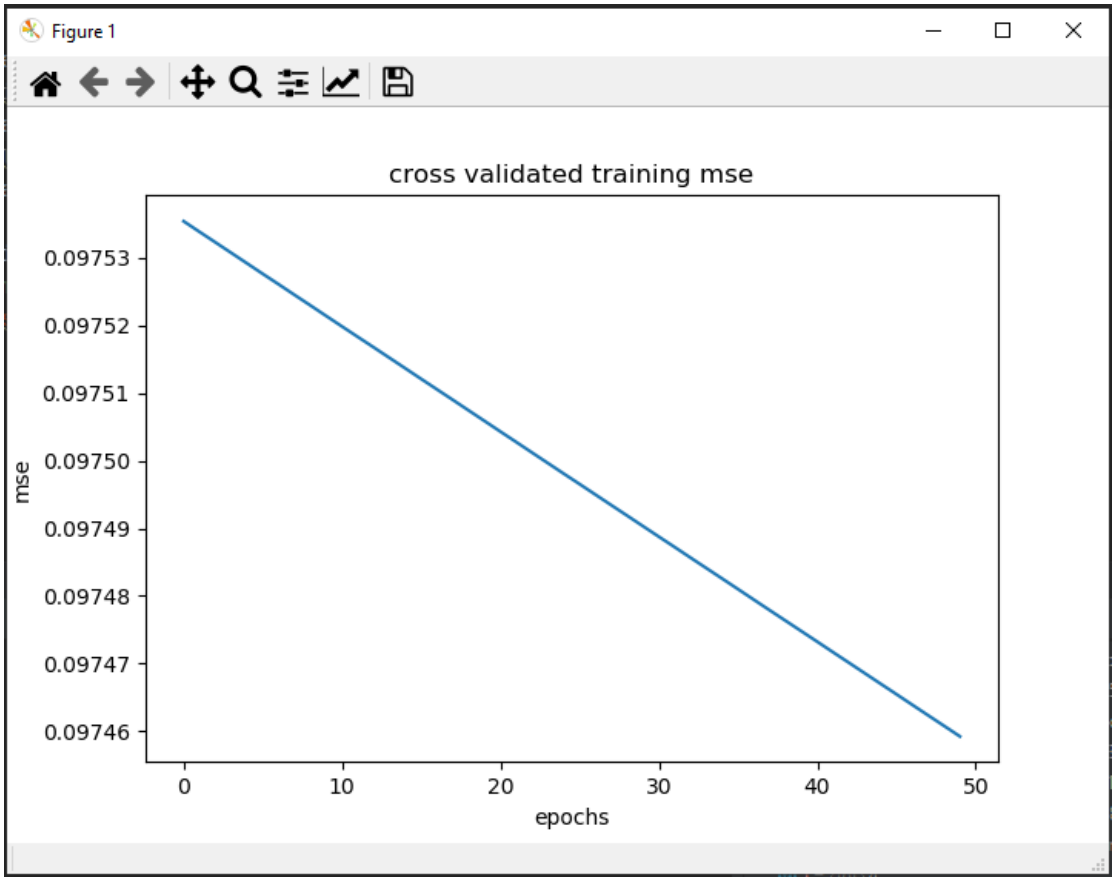


Figure 1
cross validated training mse
0.09753
0.09752
0.09751
0.09750
0.09749
0.09748
0.09747
0.09746
mse
0
10
20
30
40
50
epochs

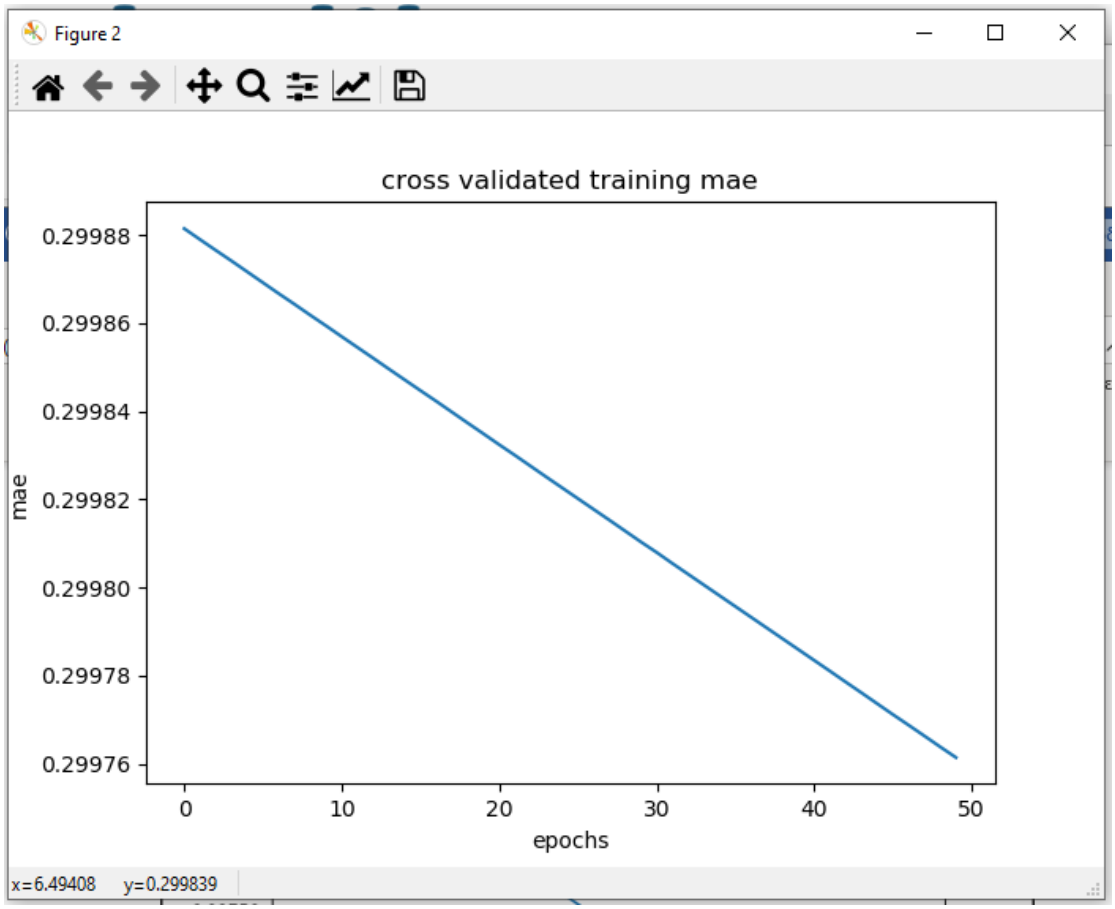


Figure 2
cross validated training mae
0.29988
0.29986
0.29984
0.29982
0.29980
0.29978
0.29976
mae
0
10
20
30
40
50
epochs
x=6.49408 y=0.299839

#κόμβων=60:

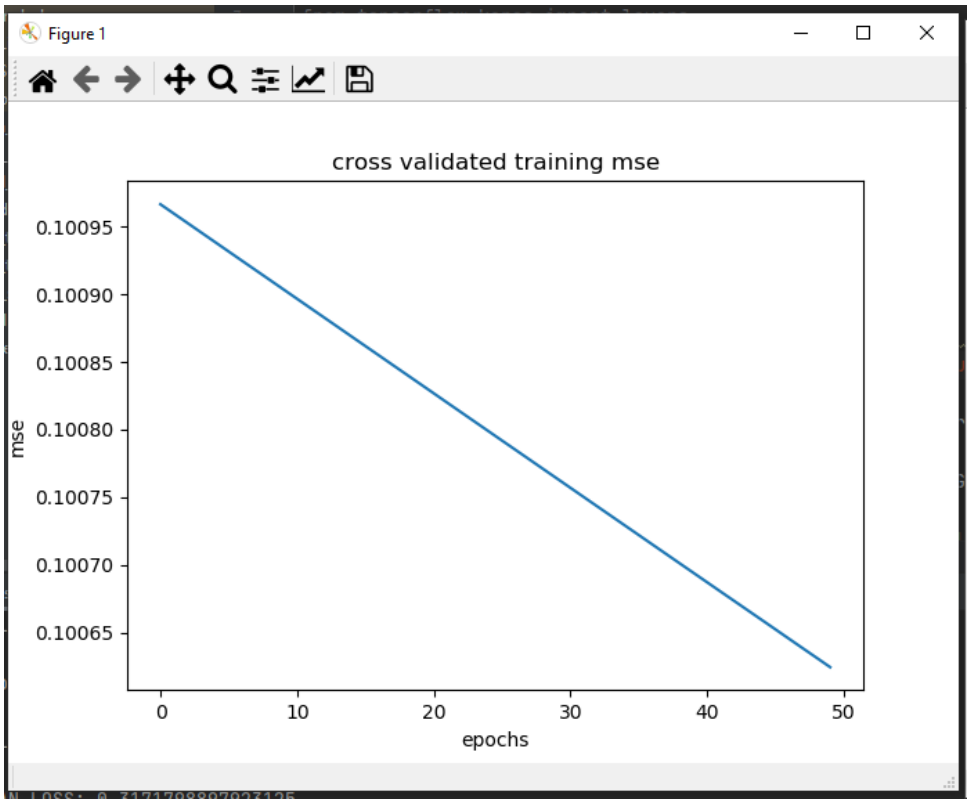


Figure 1
cross validated training mse
mse
0.10095
0.10090
0.10085
0.10080
0.10075
0.10070
0.10065
0
10
20
30
40
50
epochs

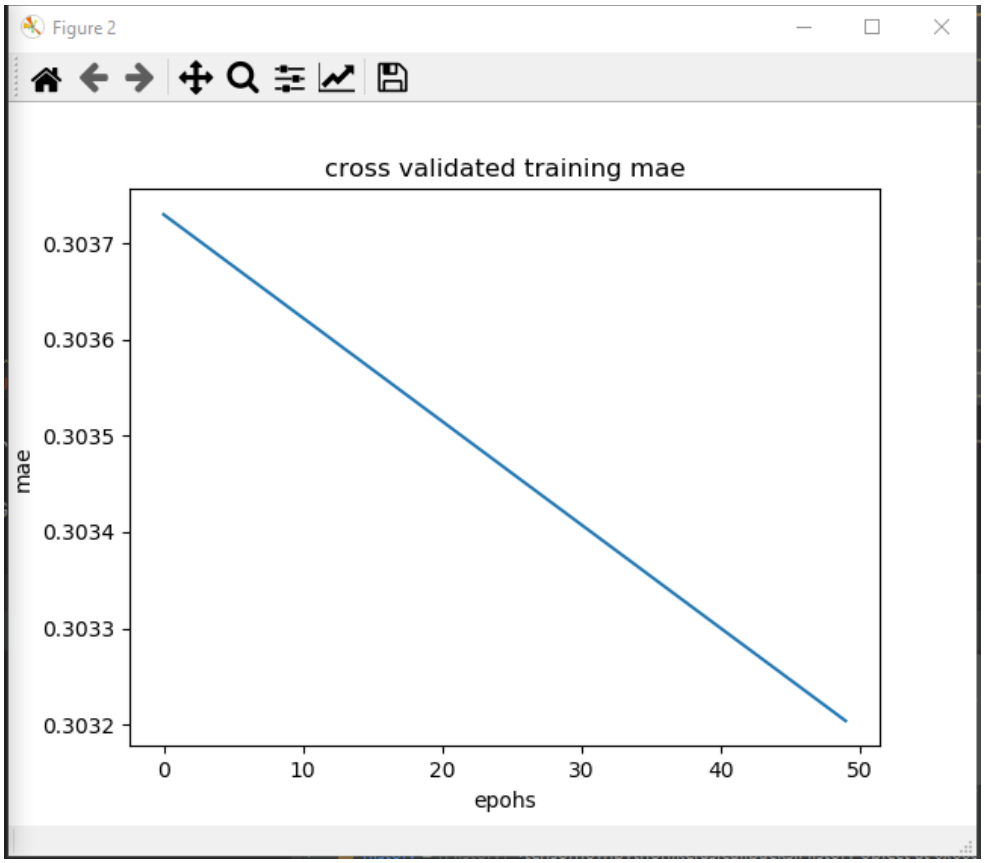


Figure 2
cross validated training mae
mae
0.3037
0.3036
0.3035
0.3034
0.3033
0.3032
0
10
20
30
40
50
epohs

Στον παρακάτω πίνακα παρατίθενται τα holdout scores για τις μετρικές RMSE κ΄MAE για κάθε περίπτωση πλήθους κόμβων.

Αριθμός νευρώνων στο κρυφό επίπεδο	RMSE	MAE
H=10	0.31334916443628175	0.3005985476012111
H=20	0.3092899714846855	0.29609422240748323
H=60	0.30507377934679164	0.2897517246988881

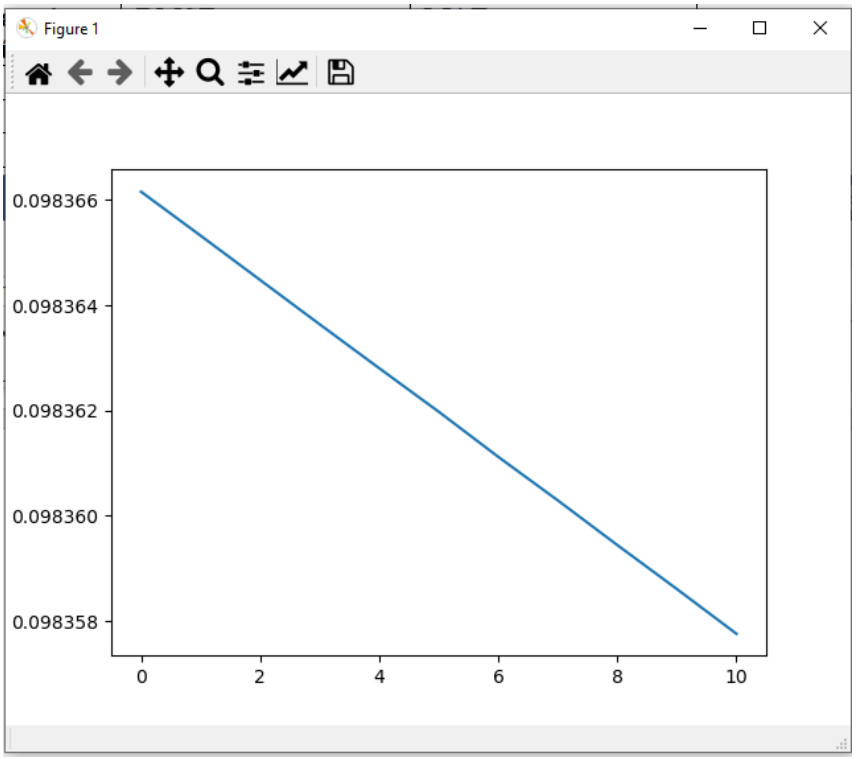
Παρατηρούμε ότι αυξάνοντας το πλήθος των κρυφών κόμβων αυξάνεται και το σφάλμα των μετρικών μας(mse,mae) για τη φάση της εκπαίδευσης ενώ για την τελική αξιολόγηση για τα άγνωστα δεδομένα αυξάνοντας το πλήθος των κρυφών κόμβων μειώνονται οι μετρικές μας(rmse,mae).

ζ)Ως κριτήριο τερματισμού επέλεξα το ακόλουθο: οποιαδήποτε αλλαγή για το val_loss μικρότερη ή ίση του 1e-3 δεν θα λαμβάνεται υπόψιν . Έβαλα patience=10 epochs.

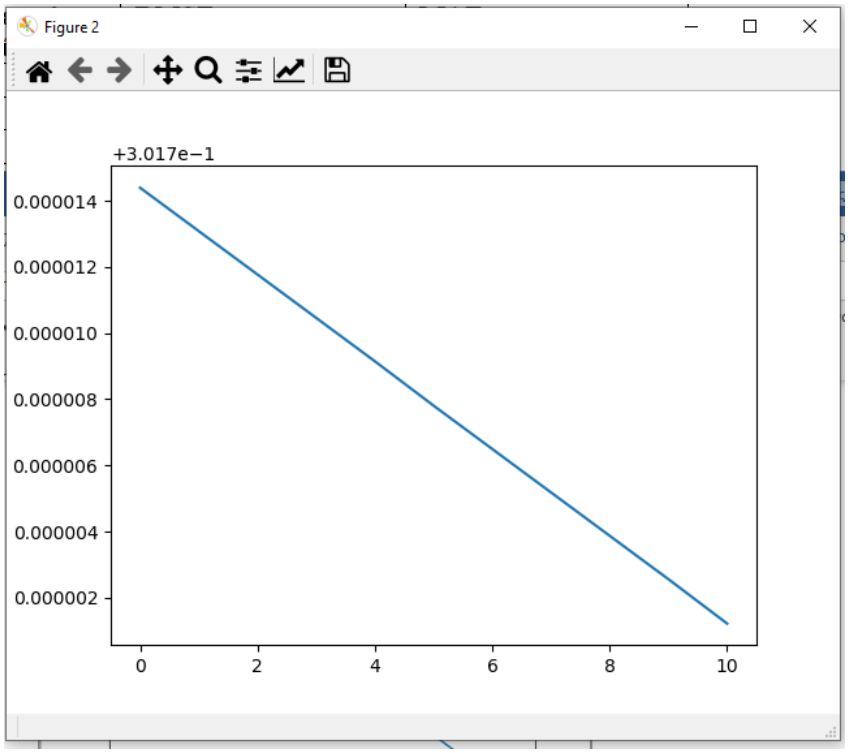
Α3. Μεταβολές στο ρυθμό εκπαίδευσης και σταθεράς ορμής

Βάση των προηγούμενων ερωτημάτων επιλέγω την αρχιτεκτονική του δικτύου με τους 10 κρυφούς κόμβους διότι δεν αποκλίνει πολύ από τις υπόλοιπες και έχει την μικρότερη πολυπλοκότητα.

η=0.001 m =0.2
MSE:

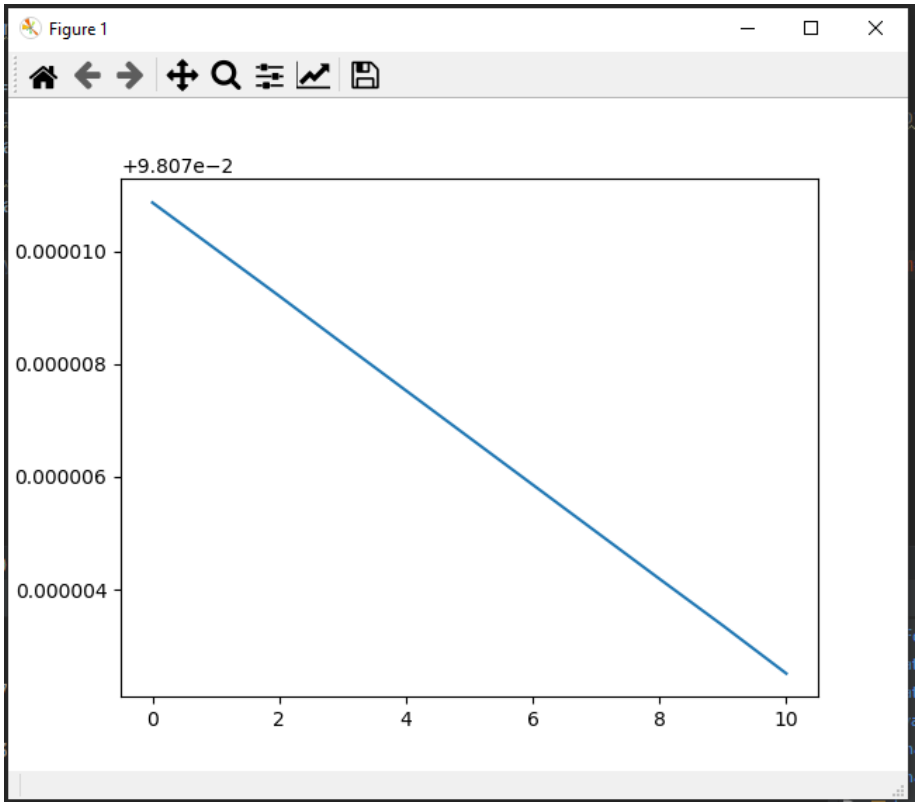


MAE:

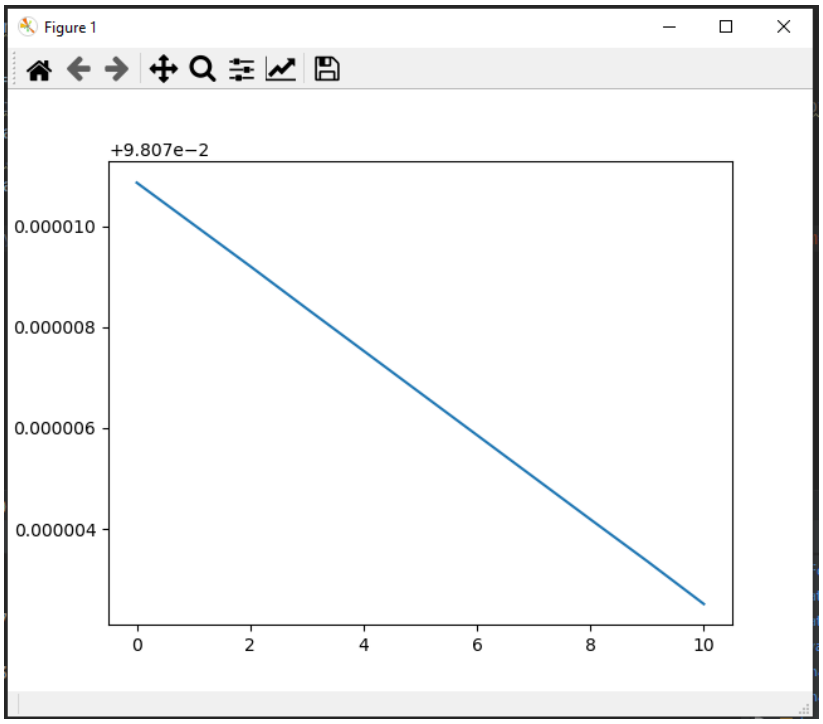


η=0.001 **m =0.6**

MSE:

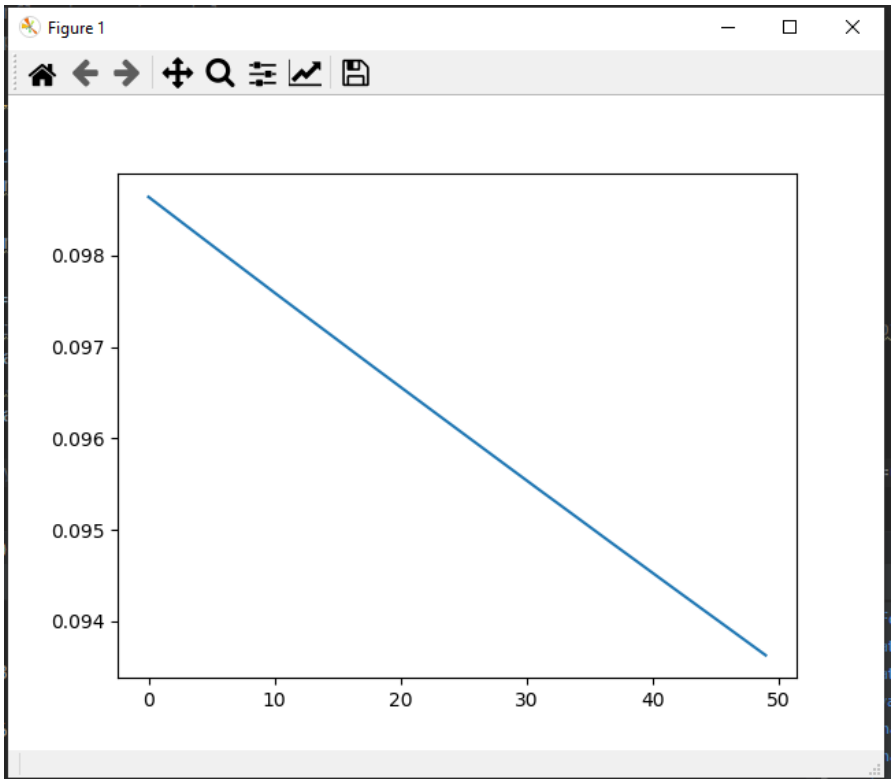


MAE:

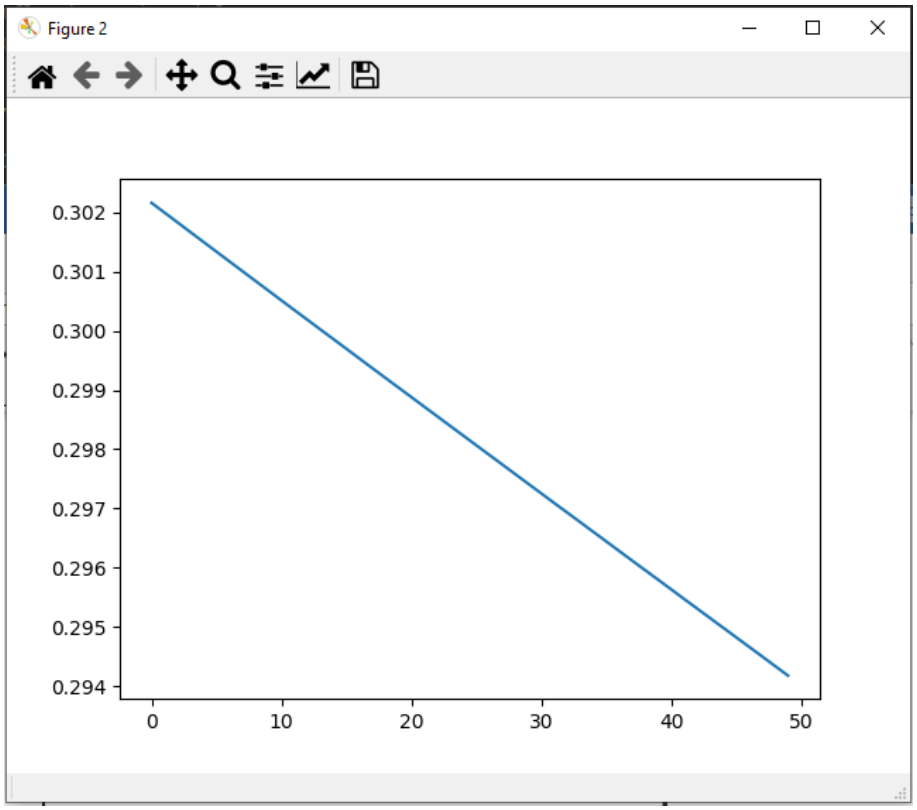


η=0.05 **m =0.6**

MSE:

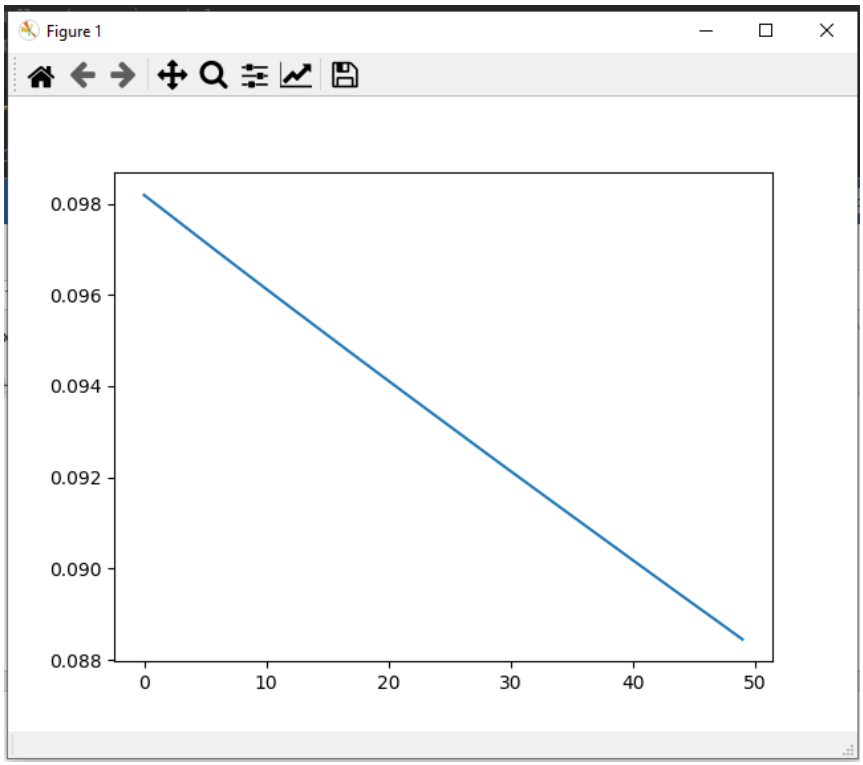


MAE:

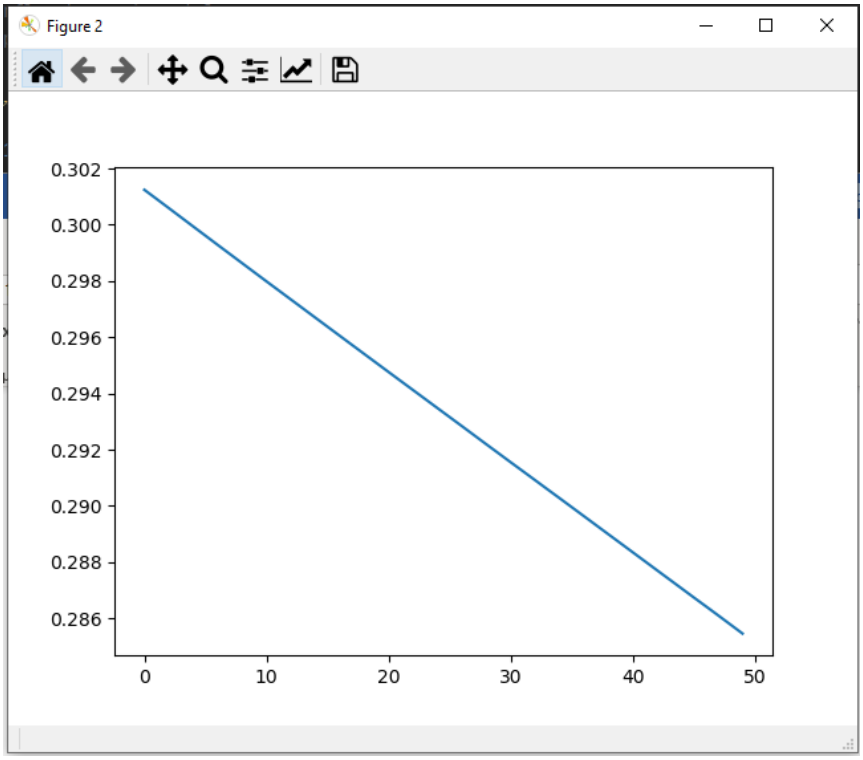


η=0.1 **m =0.6**

MSE:



MAE:



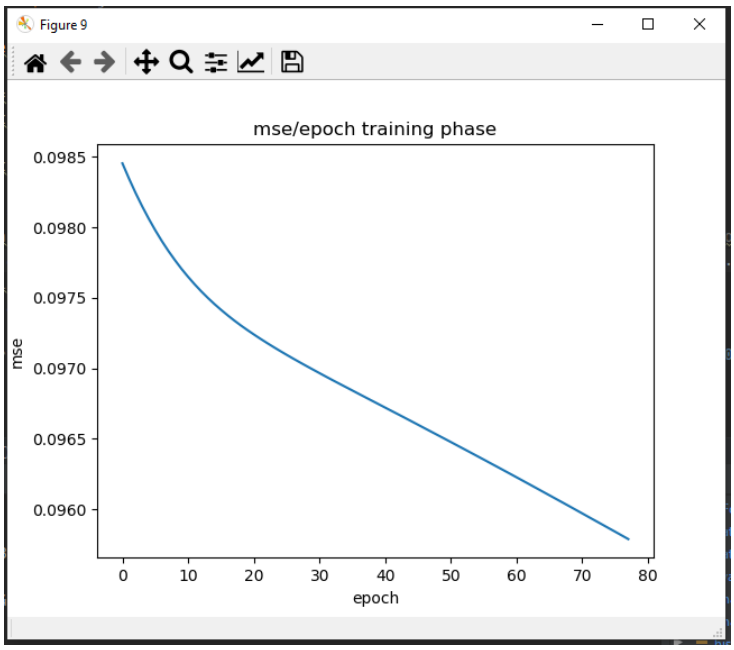
Παρατηρούμε ότι όταν μεγαλώσαμε το learning rate το νευρωνικό χρειάστηκε περισσότερες εποχές για να πραγματοποιήσει την εκπαίδευση(Το early stopping δεν πυροδοτήθηκε). Σε κάθε βήμα ανανέωσης βαρών προσθέτουμε και ένα ποσοστό των προηγούμενων ανανεώσεων βαρών. Το τι ποσοστό προσθέτουμε είναι το momentum γι'αυτό θα πρέπει να παίρνει τιμή από το 0 έως το 1.

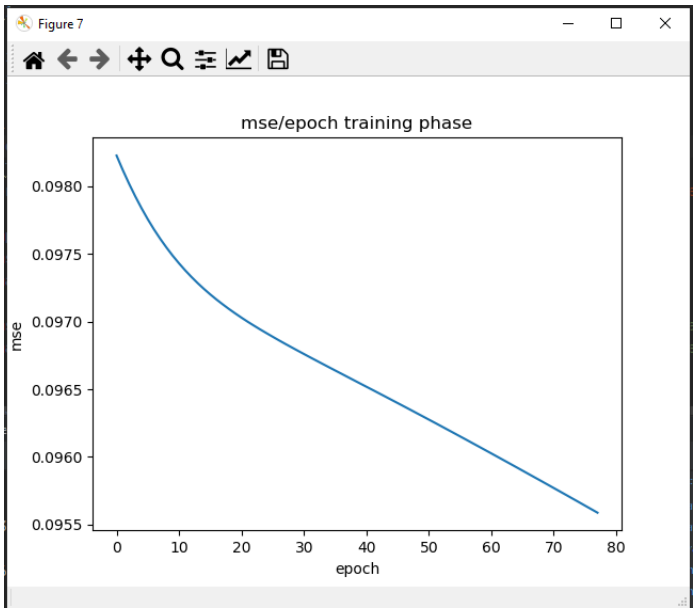
η	m	RMSE	MAE
0.001	0.2	0.3155811166678007	0.3042885488911825
0.001	0.6	0.29644547857359804	0.2862978199841341
0.05	0.6	0.30360146150264605	0.29360244226310794
0.1	0.6	0.29212768068186473	0.28104433191531913

Α4. Ομαλοποίηση

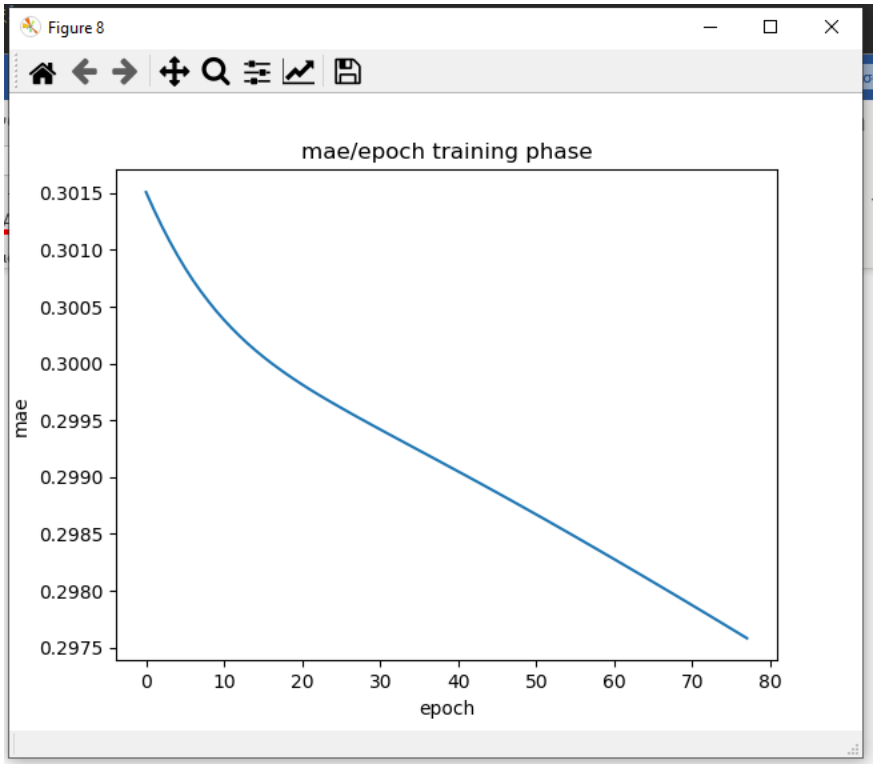
Το regularization νομίζω είναι τεχνική η οποία εφαρμόζεται σε κάθε κόμβο όπου θέλουμε να κάνουμε minimize το y-Wx με την ένοια της ευκλείδιας νόρμας με κάποια συνθήκη(όπου y έξοδος του νευρώνα,W ο kernel και x οι είσοδοι) . Η συνθήκη αυτή μπορεί να είναι η L1 νόρμα (L1 regularization) να είναι ίση με κάποια καθορισμένη τιμή είτε η L2(L2 regularization) .Η χρήση του L1 regularization ευνοεί τις αραιές αναπαραστάσεις των errors (δηλαδή δεν έχουμε θέμα εάν το διάνυσμα των error είναι αραιό και με μεγάλες τιμές) ενώ η L2 όχι γιατί χρησιμοποιεί την L2 νόρμα η οποία δίνει βαρύτητα στα μεγάλα errors.

Για r =0.1 :

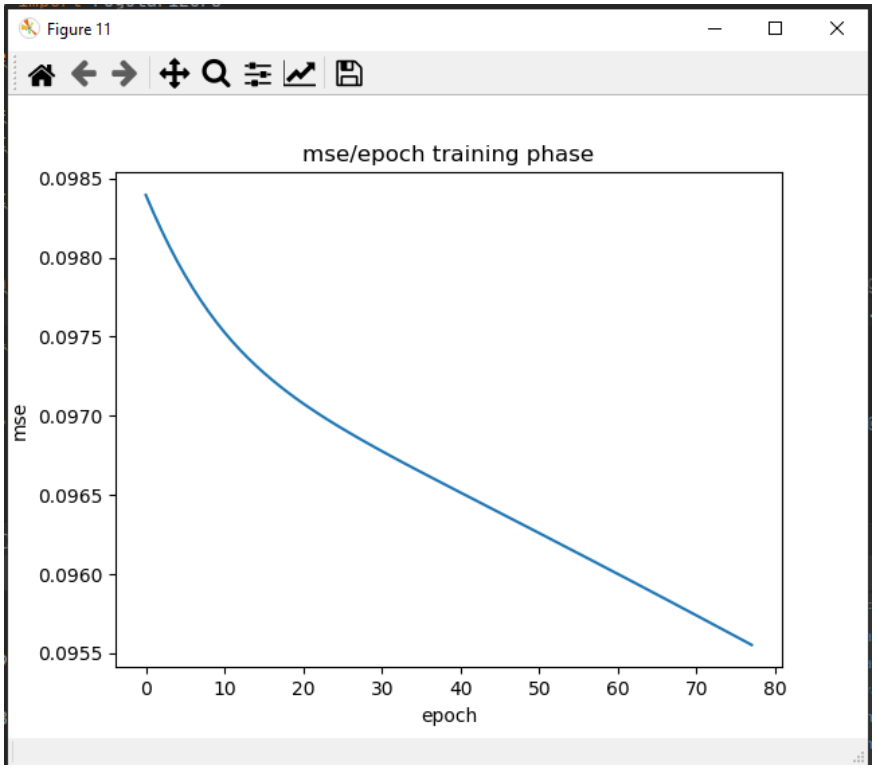


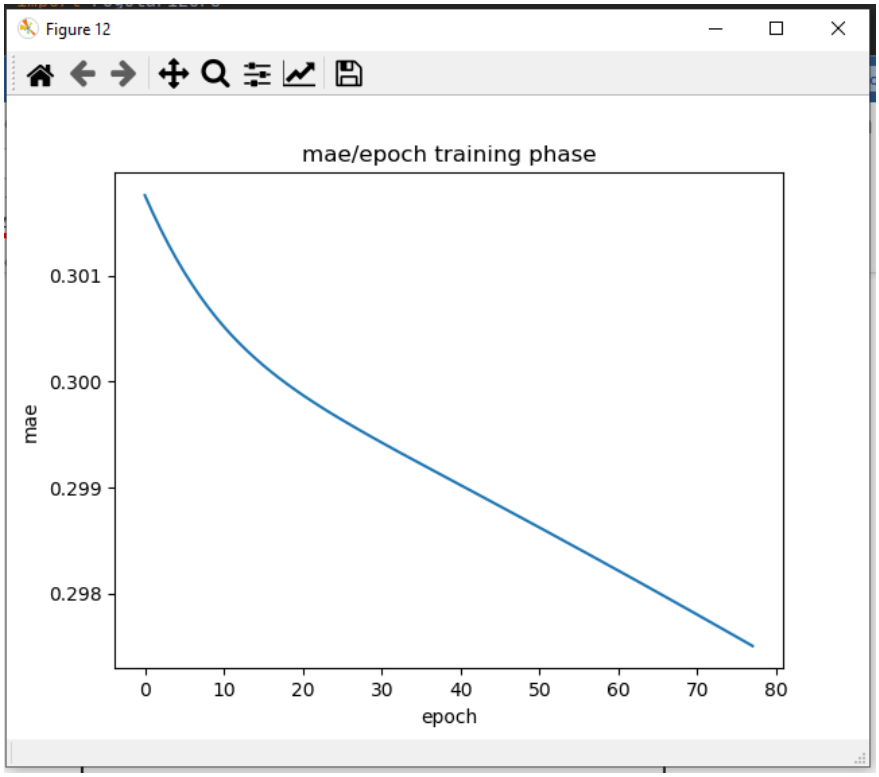


Για r=0.5:



Για r=0.9:





Συντελεστής Φθοράς	RMSE	MAE
r=0.1	0.2989051961695541	0.28796342843075773
r=0.5	0.30980527537695296	0.29875902040792823
r=0.9	0.3149429472653341	0.3024754124373025

Παρατηρούμε ότι όσο αυξάνεται ο συντελεστής φθοράς τόσο χειρότερο performance πέρνουμε. Επομένως στο παρόν πρόβλημα δεν πετυχαίνουμε καλύτερη γενικευτική ικανότητα με το να μεγαλώνουμε το συντελεστή φθοράς.

Α5. Βαθύ Νευρωνικό Δίκτυο.

Αρχικά ορίζω των αριθμό των κόμβων των κρυφών επιπέδων να μειώνεται κατ'αυτόν τον τρόπο:

```
#defining the first input layer (visible layer) and the first hidden layer with the above line of code
model.add(tf.keras.layers.Dense(30, activation=str, activity_regularizer=regularizers.l1(0.1),input_dim = 943))
model.add(tf.keras.layers.Dense(20, activation=str))
model.add(tf.keras.layers.Dense(10, activation=str))
model.add(tf.keras.layers.Dense(1682, activation=str))
```

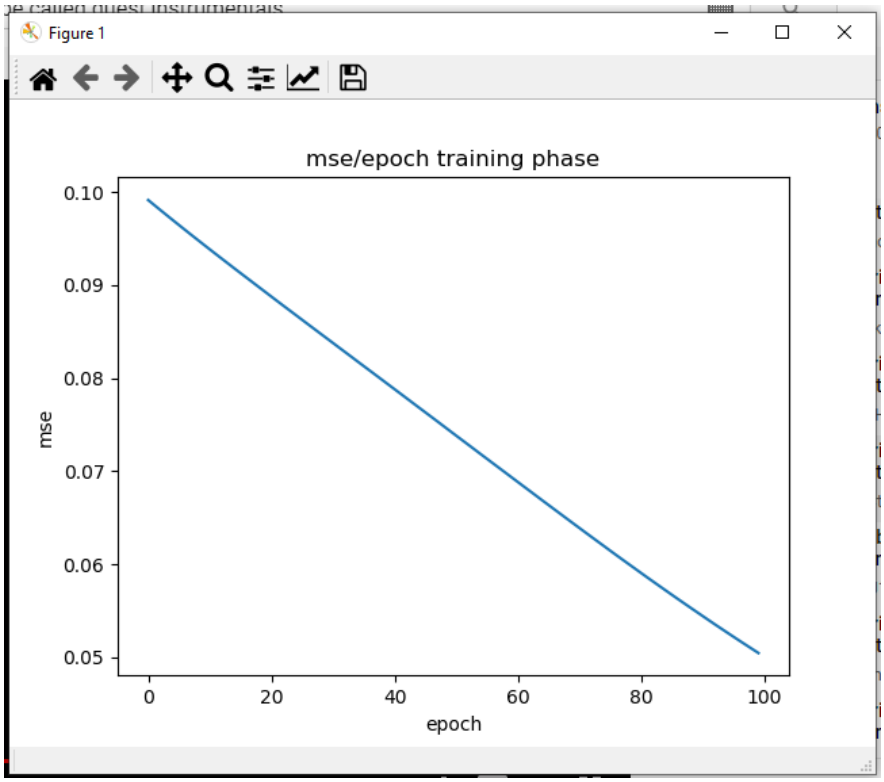
Η παραπάνω δομή μας δίνει performance:

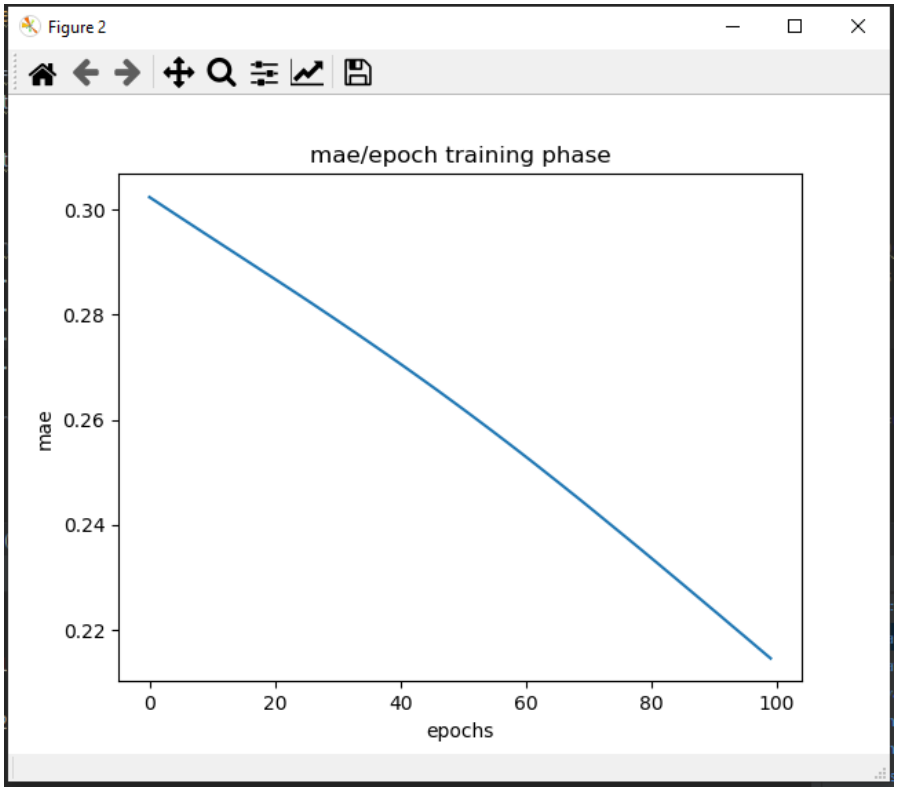
```
cross validated score (RMSE): 0.2256536332337415

holdout score (RMSE): 0.21479265423213015

holdout score (MAE): 0.19809331830730237
```

Ενώ για το training phase έχουμε :





Παρατηρούμε ότι το performance όσον αφορά training αλλά και testing έχει καλυτερέψει. Είναι εμφανές ότι στο training φτάνουμε σε αρκετά χαμηλότερο mse/mae σε σχέση με το να είχαμε 1 κρυφό επίπεδο στις ίδιες εποχές. Παρατήρησα επίσης (με πείραμα) ότι εάν ορίζαμε την αντίστροφη αρχιτεκτονική (αύξων αριθμό κρυφών κόμβων ανά επίπεδο) παίρνουμε παρόμοια αποτελέσματα συνολικά.

Έπειτα ορίζω των αριθμό των κόμβων των κρυφών επιπέδων να είναι σταθερός κατ'αυτόν τον τρόπο:

```
#defining the first input layer (visible layer) and the first hidden layer with the above line of code
model.add(tf.keras.layers.Dense(10, activation=str, activity_regularizer=regularizers.l1(0.1),input_dim=943))
model.add(tf.keras.layers.Dense(10, activation=str))
model.add(tf.keras.layers.Dense(10, activation=str))
model.add(tf.keras.layers.Dense(1682, activation=str))
```

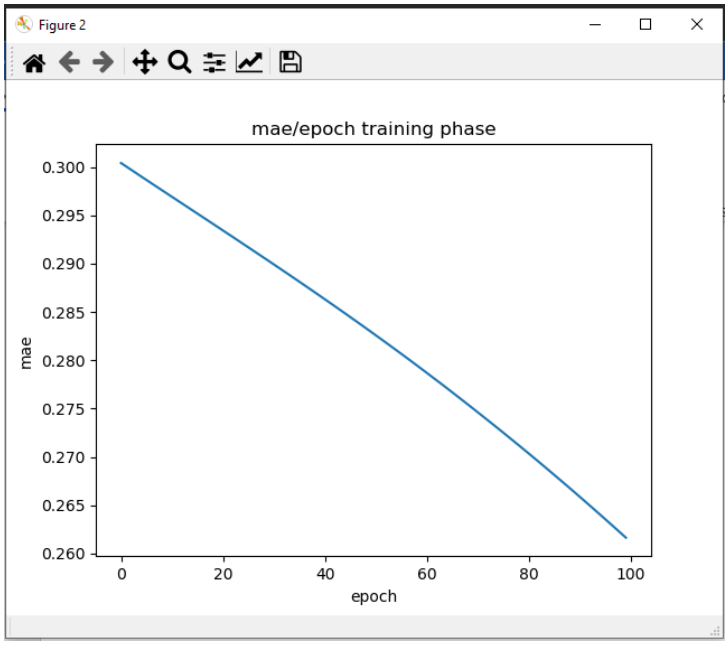
Αυτή η δομή μας δίνει performance:

```
cross validated score (RMSE): 0.2740010859194914

holdout score (RMSE): 0.2742362720428585

holdout score (MAE): 0.25837193857864815
```

Ενώ για το training phase έχουμε :



Τέλος παρατηρούμε ότι τα αποτελέσματα που παίρνουμε από αυτή την αρχιτεκτονική τόσο στο training όσο και στο testing phase είναι χειρότερα από την αύξουσα ή τη φθίνουσα αρχιτεκτονική που περιγράφηκαν παραπάνω.