ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα 2016 – 2017

1η Εργαστηριακή Άσκηση:
«Μελέτη των πολυεπίπεδων Perceptrons και εφαρμογή σε προβλήματα ταξινόμησης εικόνας»

Χρήστος Θεοδωρόπουλος, 03112904

1. Εισαγωγή

Τα πολυεπίπεδα δίκτυα εμπρόσθιας τροφοδότησης (Multilayer Perceptrons) αποτελούν μια θεμελιώδη κατηγορία νευρωνικών δικτύων. Ένα τέτοιο δίκτυο αποτελείται από ένα σύνολο αισθητήρων που αποτελούν το επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα στα οποία γίνονται οι διάφοροι υπολογισμοί και ένα επίπεδο εξόδου. Το πρότυπο που παρουσιάζεται στο επίπεδο εισόδου διαδίδεται κατά την εμπρόσθια διεύθυνση, επίπεδο προς επίπεδο. Ένας διαδεδομένος αλγόριθμος που εφαρμόζεται για την εκπαίδευση πολυεπίπεδων δικτύων εμπρόσθιας τροφοδότησης είναι ο αλγόριθμος οπίσθιας διάδοσης (Back Propagation).

Τα χαρακτηριστικά του πολυεπίπεδου perceptron είναι τα εξής:

- Κάθε νευρώνας του δικτύου περιλαμβάνει μία μη γραμμική (διαφορίσιμη) συνάρτηση ενεργοποίησης.
- Το δίκτυο αποτελείται από ένα ή περισσότερα κρυμμένα επίπεδα νευρώνων, τα οποία δεν αποτελούν είσοδο ή έξοδο και του παρέχουν τη δυνατότητα μάθησης.
- Το δίκτυο έχει μεγάλο βαθμό συνεκτικότητας, που καθορίζεται από τις συνάψεις του.

Ο συνδυασμός των παραπάνω χαρακτηριστικών, μαζί με την ικανότητα της μάθησης μέσω την εκπαίδευσης προσδίδει στο δίκτυο μεγάλη υπολογιστική ισχύ. Στη σχεδίαση ενός νευρωνικού δικτύου υπεισέρχονται πολλά θέματα αρχιτεκτονικής στα οποία δεν υπάρχει συγκεκριμένη λύση. Αντίθετα ο χρήστης πρέπει να τα αντιμετωπίσει στηριζόμενος στην προσωπική του εμπειρία.

Ο σκοπός της άσκησης αυτής είναι η κατασκευή και η μελέτη ενός Multilayer Perceptron (MLP), που θα εκπαιδευτεί με τον κλασικό αλγόριθμο Back Propagation για την επίλυση ενός προβλήματος ταξινόμησης τμημάτων εικόνων, με βάση κάποια χαρακτηριστικά χαμηλού επιπέδου.

2. Μελέτη ενός Νευρωνικού Δικτύου για την ταξινόμηση τμημάτων εικόνων

Μια συνηθισμένη διαδικασία για την επίτευξη της ταξινόμησης τμημάτων εικόνων με βάση το περιεχόμενο είναι η ακόλουθη:

- Αρχικά ένας αλγόριθμος τμηματοποίησης επεξεργάζεται την εικόνα και τη χωρίζει σε περιοχές με παράμοια χαρακτηριστικά χαμηλού επιπέδου.
- Τα τμήματα αυτά ταξινομούνται χειροκίνητα σύμφωνα με το περιεχόμενό τους σε κάποιες κατηγορίες.
- Με αυτό το τρόπο δημιουργείται ένα σύνολο εκπαίδευσης (χαρακτηριστικά και επιθυμητές αποκρίσεις) το οποίο χρησιμοποιείται από ένα νευρωνικό δίκτυο για την ταξινόμηση των τμημάτων εικόνας.

Οι πίνακες που θα χρησιμοποιηθούν για την εκπαίδευση του δικτύου έχουν τα εξής περιεχόμενα:

- Κάθε στήλη του πίνακα TrainData αντιστοιχεί στο τμήμα μίας εικόνας ενώ κάθε γραμμή περιέχει τα χαρακτηριστικά χαμηλού επιπέδου που αντιστοιχούν σε Descriptors) του τμήματος αυτού.
- Κάθε στήλη του πίνακα TrainDataTargets αντιστοιχεί στο τμήμα μίας εικόνας ενώ κάθε γραμμή αντιστοιχεί σε μία από τις κατηγορίες «Κτήριο», «Χορτάρι», «Δέντρο», «Άλογο», «Γάτα», «Ουρανός», «Βουνό», «Αεροπλάνο», «Νερό», «Πρόσωπο», «Αυτοκίνητο», «Ποδήλατο», «Λουλούδι» κ.ά. Για την υλοποίηση της άσκησης έχουν επιλεγεί 5 μόνο κατηγορίες. Στην περίπτωση που το τμήμα μιας εικόνας ανήκει σε μία από τις κατηγορίες τότε η αντίστοιχη γραμμή θα έχει την τιμή 1 ενώ οι υπόλοιπες γραμμές θα πάρουν την τιμή 0.

Τα δεδομένα των πινάκων TrainData, TrainDataTargets που θα χρησιμοποιηθούν για την εκπαίδευση του νευρωνικού δικτύου ενώ τα δεδομένα των πινάκων TestData, TestDataTargets θα χρησιμοποιηθούν ως μέτρο των επιδόσεων του νευρωνικού δικτύου.

2.1 Προεπεξεργασία των δεδομένων

Βήμα 1

Ο κώδικας που υλοποιεί το Βήμα 1 βρίσκεται στο αρχείο ex1.m μαζί με όλα τα απαραίτητα σχόλια για την κατανόησή του.

Βήμα 2

Ο κώδικας που υλοποιεί το Βήμα 2 βρίσκεται στο αρχείο ex1.m μαζί με όλα τα απαραίτητα σχόλια για την κατανόησή του.

Βήμα 3

Ο κώδικας που υλοποιεί το Βήμα 3 βρίσκεται στο αρχείο ex1.m μαζί με όλα τα απαραίτητα σχόλια για την κατανόησή του.

Βήμα 4

Σε αυτό το βήμα πειραματίστηκα με το πλήθος των νευρώνων. Στην αρχή χρησιμοποίησα ένα κρυφό layer με 5 νευρώνές και στη συνέχεια αύξανα τους νευρώνες επαναληπτικά μέχρι να φτάσω τους 30, με βήμα 5. Μετά πρόσθεσα και δεύτερο κρυφό layer και έτρεξα προσομοιώσεις μεταβάλλοντας τον αριθμό των νευρώνων ανά layer.

Βήμα 5

Εκπαίδευσα το δίκτυο μου και με τις 4 συναρτήσεις εκπαίδευσης αλλάζοντας κάθε φορά το πλήθος των νευρώνων ανά κρυφό layer. Για κάθε εκπαίδευση υπολόγισα τα accuracy, precision και recall και τα πέρασα σε φύλλα excel (1 φύλλο για κάθε μία από τις 4 συναρτήσεις εκπαίδευσης) έτσι ώστε να μπορέσω τελικά να αποφασίσω ποια συνάρτηση θα χρησιμοποιήσω και πόσους νευρώνες θα βάλω σε κάθε κρυφό layer.

Ο κώδικας που υλοποιεί την παραπάνω διαδικασία βρίσκεται στο αρχείο ex1_checking.m.

Προέκυψαν τελικά 4 φύλλα excel. Για να μπορέσω να δω ποια τοπολογία νευρώνων μου έδωσε το καλύτερο αποτέλεσμα υπολόγισα τα F1 Scores (2*Precision*Recall/(Precision+Recall) ανά κατηγορία. Στη συνέχεια υπολόγισα το μέσο όρο των F1 scores ανά τοπολογία νευρώνων.

Έτσι λοιπόν προέκυψαν τα ακόλουθα αποτελέσματα:

Traingd		**Trainda**		**Traingdx**		**Trainlm**	
Max Accuracy	Max Average F1 Score	Max Accuracy	Max Average F1 Score	Max Accuracy	Max Average F1 Score	Max Accuracy	Max Average F1 Score
0,71028	0,675154793	0,91588	0,901604789	0,906542	0,899595869	0,925234	0,937854973
[30 30]	[30]	[5 10]	[5 10]	[20]	[20]	[10 20]	[10 20]

Σημειώσεις:

- Στην τελευταία γραμμή γράφεται ο αριθμός των νευρώνων ανά κρυφό layer.
- Συμβολισμός [i j], όπου i ο αριθμός των νευρώνων που βρίσκεται στο πρώτο κρυφό layer και j ο αριθμός των νευρώνων στο δεύτερο κρυφό layer.

Φαίνεται λοιπόν ότι παίρνω το βέλτιστο αποτέλεσμα όταν εκπαιδεύω το νευρωνικό δίκτυο ([10 20]) με τη συνάρτηση *trainlm*. Για αυτό το λόγο επιλέγω να έχω στο πρώτο κρυφό layer 10 νευρώνες, στο δεύτερο κρυφό layer 20 νευρώνες και η εκπαίδευση του δικτύου να γίνεται με τη συνάρτηση *trainlm*. Επίσης η συνάρτηση εκπαιδεύει γρήγορα το δίκτυο και αυτό είναι σημαντικό ιδιαίτερα σε μεγάλες εφαρμογές.

Αξίζει να σημειώθει ότι κάθε φορά που τρέχει το πρόγραμμα τα βάρη που επιλέγονται για κάθε νευρώνα αλλάζουν. Αυτό έχει σαν αποτέλεσμα τα accuracy, precision και recall να αλλάζουν κάθε φορά ελάχιστα. Μία άλλη στρατηγική που θα μπορούσε να ακολουθήσει κανείς είναι να κάνει δυναμικά την επιλογή του πλήθους νευρώνων ανά layer κάθε φορά που τρέχει το πρόγραμμα, επιλέγοντας την τοπολογία που δίνει τη βέλτιστη απόδοση. Αυτό βέβαια θα κοστίζει σε χρόνο, ιδιαίτερα σε εφαρμογές με πολύ μεγάλο όγκο δεδομένων και πολύ μεγάλο πλήθος νευρώνων.

Σημείωση:

Στην συνέχεια της άσκησης δοκίμασα πάλι την traingd για να απαντήσω σε ένα ερώτημα, μεταβάλλοντας το learning rate. Στην περίπτωση που το learning rate ισούται με 0.2 έχω πολύ καλή επίδοση στο δίκτυο. Για αυτό και στη συνεχεία αναφέρω ότι τελικά επιλέγω την traingd.

Βήμα 6

a. Δοκίμασα και τις τέσσερις συναρτήσεις ενεργοποίησης. Η hardlim και η logsig τα πήγαν άσχημα ενώ η tansig και η purelin τα πήγαν καλά. Τρέχοντας αρκετές φορές το πρόγραμμα παρατήρησα ότι η *tagsig* τα πήγε λίγο καλύτερα οπότε επιλέγω αυτή ως συνάρτηση ενεργοποίησης.

Ενδεικτικές έξοδοι:

→ Hardlim:

```
Loading and Visualizing Data ...
accuracy: 0.093458
precision: 0.093458
precision: NaN
precision: NaN
precision: NaN
precision: NaN
recall: 1.000000
recall: 0.000000
recall: 0.000000
recall: 0.000000
recall: 0.000000
>>
```

→ Tansig:

```
Loading and Visualizing Data ...
accuracy: 0.859813
precision: 0.750000
precision: 0.900000
precision: 0.625000
precision: 0.956522
precision: 1.000000
recall: 0.900000
recall: 0.750000
recall: 0.882353
recall: 0.880000
recall: 0.903226
>>
```

→ Logsig:

```
Loading and Visualizing Data ...
accuracy: 0.299065
precision: 0.131579
precision: 0.709677
precision: NaN
precision: NaN
precision: NaN
recall: 1.000000
recall: 0.916667
recall: 0.000000
recall: 0.000000
recall: 0.000000
>>
```

→ Purelin:

```
Loading and Visualizing Data .
accuracy: 0.878505
precision: 0.900000
precision: 0.904762
precision: 0.640000
precision: 0.958333
precision: 1.000000
recall: 0.900000
recall: 0.791667
recall: 0.941176
recall: 0.920000
recall: 0.870968
>>
```

b. Όταν χρησιμοποιήθηκε ο αλγόριθμος απότομης καθόδου με προσθήκη του όρου ορμής (*learngdm*), οι επιδόσεις του νευρωνικού δικτύου ήταν καλύτερες.

Ενδεικτικές έξοδοι:

→ learngd:

```
Loading and Visualizing Data ...
accuracy: 0.878505
precision: 0.692308
precision: 0.913043
precision: 0.739130
precision: 0.950000
precision: 1.000000
recall: 0.900000
recall: 0.875000
recall: 1.000000
recall: 0.760000
recall: 0.903226
>>
```

```
Loading and Visualizing Data ...
accuracy: 0.859813
precision: 0.818182
precision: 0.857143
precision: 0.640000
precision: 0.956522
precision: 1.000000
recall: 0.900000
recall: 0.750000
recall: 0.941176
recall: 0.880000
recall: 0.870968
>>
```

→ learngdm:

```
Loading and Visualizing Data ..
accuracy: 0.915888
precision: 0.900000
precision: 0.875000
precision: 0.789474
precision: 0.958333
precision: 1.000000
recall: 0.900000
recall: 0.875000
recall: 0.882353
recall: 0.920000
recall: 0.967742
>>
```

```
Loading and Visualizing Data ...
accuracy: 0.925234
precision: 1.000000
precision: 0.875000
precision: 0.800000
precision: 0.958333
precision: 1.000000
recall: 0.900000
recall: 0.875000
recall: 0.941176
recall: 0.920000
recall: 0.967742
>>
```

c. Οι επιδόσεις του νευρωνικού δικτύου ήταν περίπου οι ίδιες είτε χρησιμοποιούσα validation set είτε όχι. Βέβαια όταν δε χρησιμοποιούσα validation set η συνάρτηση έτρεχε περισσότερες φορές.

Ενδεικτικές έξοδοι:

→ Με validation set:

```
Loading and Visualizing Data ...
accuracy: 0.841121
precision: 0.750000
precision: 0.869565
precision: 0.615385
precision: 0.950000
precision: 1.000000
recall: 0.900000
recall: 0.833333
recall: 0.941176
recall: 0.760000
recall: 0.838710
>>
```

```
Loading and Visualizing Data ...
accuracy: 0.887850
precision: 0.750000
precision: 0.875000
precision: 0.789474
precision: 0.913043
precision: 1.000000
recall: 0.900000
recall: 0.875000
recall: 0.882353
recall: 0.840000
recall: 0.935484
>>
```

→ Χωρίς validation set:

```
Loading and Visualizing Data ...
accuracy: 0.878505
precision: 0.714286
precision: 0.833333
precision: 0.800000
precision: 1.000000
precision: 0.967742
recall: 1.000000
recall: 0.833333
recall: 0.941176
recall: 0.720000
recall: 0.967742
>>
```

```
Loading and Visualizing Data ...
accuracy: 0.869159
precision: 0.900000
precision: 0.869565
precision: 0.640000
precision: 0.954545
precision: 1.000000
recall: 0.900000
recall: 0.833333
recall: 0.941176
recall: 0.840000
recall: 0.870968
>>
```

d. Το δίκτυο έχει τη δυνατότητα γενίκευσης. Η επίδοσή του εξακολουθεί να είναι υψηλή και χωρίς τη χρήση validation set.

e. Έτρεξα το πρόγραμμα αλλάζοντας κάθε φορά το learning rate για κάθε μία από τις δύο συναρτήσεις. Η συνάρτηση traingd τα πήγε πολύ καλά με learning rate με σχεδόν όλα τα διαφορετικά learning rates. Ιδιαίτερα καλά τα πήγε με learning rate 0.2, 0.4 και 0.15. Αντιθέτως η συνάρτηση traingdx παρουσίαζε πολύ διαφορετική συμπεριφορά αλλάζοντας το learning rate. Είχε καλή απόδοση με learning rate 0.1 (default τιμή) και 0.05, ενώ για τις υπόλοιπες τιμές τα πήγε πολύ άσχημα. Σε αυτό έπαιξε ρόλο και το γεγονός ότι η συνάρτηση

έτρεξε λίγες φορές επειδή τα επιτρεπόμενα validation failures (6) έγιναν πολύ σύντομα.

Αναλυτικά τα αποτελέσματα των προσομοιώσεων υπάρχουν στα αντίστοιχα φύλλα excel.

Βήμα 7

Η τελική επιλογή των παραμέτρων του νευρωνικού δικτύου είναι πολύ δύσκολη καθώς δεν υπάρχει σωστό και λάθος. Αυτό που σε ενδιαφέρει είναι να επιλέξεις παραμέτρους έτσι ώστε το νευρωνικό σου δίκτυο να έχει πολύ καλό performance στη συγκεκριμένη εφαρμογή που θες να το χρησιμοποιήσεις.

Συνοψίζοντας η συνάρτηση trainlm εκπαιδεύει το δίκτυο γρήγορα αλλά δεν παρουσιάζει μεγάλη σταθερότητα καθώς η απόδοση του δικτύου στη συγκεκριμένη εφαρμογή μεταβάλλεται (μερικές φορές σημαντικά) κάθε φορά που τρέχει το πρόγραμμα και αλλάζουν τα βάρη. Επίσης η trainlm είναι γρήγορη αλλά χρειάζεται πολλή μνήμη. Αντιθέτως η συνάρτηση traingd είναι πιο αργή αλλά παρουσιάζει μεγαλύτερη σταθερότητα στην απόδοση κάθε φορά που τρέχει το πρόγραμμα και τα βάρη μεταβάλλονται.

Επειδή στη συγκεκριμένη εφαρμογή ο όγκος των δεδομένων είναι μικρός και δε μου κοστίζει πολύ χρονικά ο περισσότερος χρόνος που χρειάζεται η traingd για να εκπαιδεύσει το δίκτυο μου θα επιλέξω αυτή για να εκπαιδεύσω το δίκτυο μου. Σε αντίθετη περίπτωση θα επέλεγα την trainlm.

Τελικές επιλογές:

Αριθμός νευρώνων ανά κρυφό layer	[10 20]
Συνάρτηση εκπαίδευσης	traingd
Συνάρτηση ενεργοποίησης για το στρώμα εξόδου	tansig
Αλγόριθμος Μάθησης	learngdm
Validation Set	20% των δεδομένων
Ρυθμός Μάθησης (Learning Rate)	0.2

Σημείωση:

Η διαδικασία της επιλογής όλων των παραμέτρων είναι χρονοβόρα. Αν είχα στη διάθεση μου περισσότερο χρόνο θα έτρεχα περισσότερες προσομοιώσεις.

Παρατήρηση:

Όλος ο κώδικας υλοποίησης βρίσκεται στο αρχείο *ex1.m.*

3. Ερωτήματα

Ερώτημα 1

Η προεπεξεργασία των δεδομένων είναι ιδιαίτερα σημαντική για την ομαλή και επιτυχημένη εκπαίδευση του νευρωνικού δικτύου. Σε πρώτη φάση είναι σημαντικό να υπάρχει περίπου ο ίδιος αριθμός δεδομένων ανά κατηγορία στο training set έτσι ώστε το νευρωνικό δίκτυο να εκπαίδευτει ισάξια στην αναγνώριση της κάθε κατηγορίας. Αυτό επιτυγχάνεται στο πρώτο βήμα της προεπεξεργασίας. Στη συνέχεια οι σειρές με σταθερές τιμές πρέπει να αφαιρεθούν καθώς δεν υπάρχει ιδιαίτερα χρήσιμη πληροφορία σε αυτές. Τέλος με χρήση της processpca εφαρμόζουμε Principal Component Analysis (PCA), έτσι ώστε να προκύψει ένας πίνακας με περίπου 20 γραμμές χωρίς να χάσουμε χρήσιμη πληροφορία. Με αυτό τον τρόπο καταφέρνουμε να αυξήσουμε την ταχύτητα με την οποία εκπαιδεύεται το δίκτυό μας αλλά και να περιορίσουμε τη μνήμη που χρειάζεται η συνάρτηση εκπαίδευσης (η trainlm χρειάζεται πολλή μνήμη).

Χωρίς την προεπεξεργασία των δεδομένων η εκπαίδευση του δικτύου καθίσταται δυνατή αλλά η επίδοσή του είναι σαφώς μειωμένη. Έτρεξα το πρόγραμμα χωρίς προεπεξεργασία δεδομένων και τα accuracy, precision, recall ήταν μειωμένα. Σε κάμιες περιπτώσεις μάλιστα ήταν πολύ μειωμένα καθώς δεν υπήρχε σταθερότητα στο performance.

Ερώτημα 2

Το ερώτημα αυτό απαντήθηκε σε όλη την ανάλυση που έχει προηγηθεί. Αναλυτικά οι επιδόσεις των διαφόρων νευρωνικών δικτύο βρίσκονται στα excel φύλλα που έχουν παραχθεί.

Ερώτημα 3

Πολλά στοιχεία που απαντούν αυτό το ερώτημα έχουν δωθεί στο σημείο της ανάλυσης όπου έγινε η επιλογή της συνάρτησης ενεργοποίησης του στρώματος εξόδου. Οι tansig και η purelin έδιναν πολύ καλά αποτελέσματα, με την tagsig να δίνει λίγο καλύτερα. Για αυτό το λόγο επιλέχθηκε η tansig. Οι άλλες δύο συναρτήσεις (hardlim, logsig) δεν έδιναν καλά αποτελέσματα.

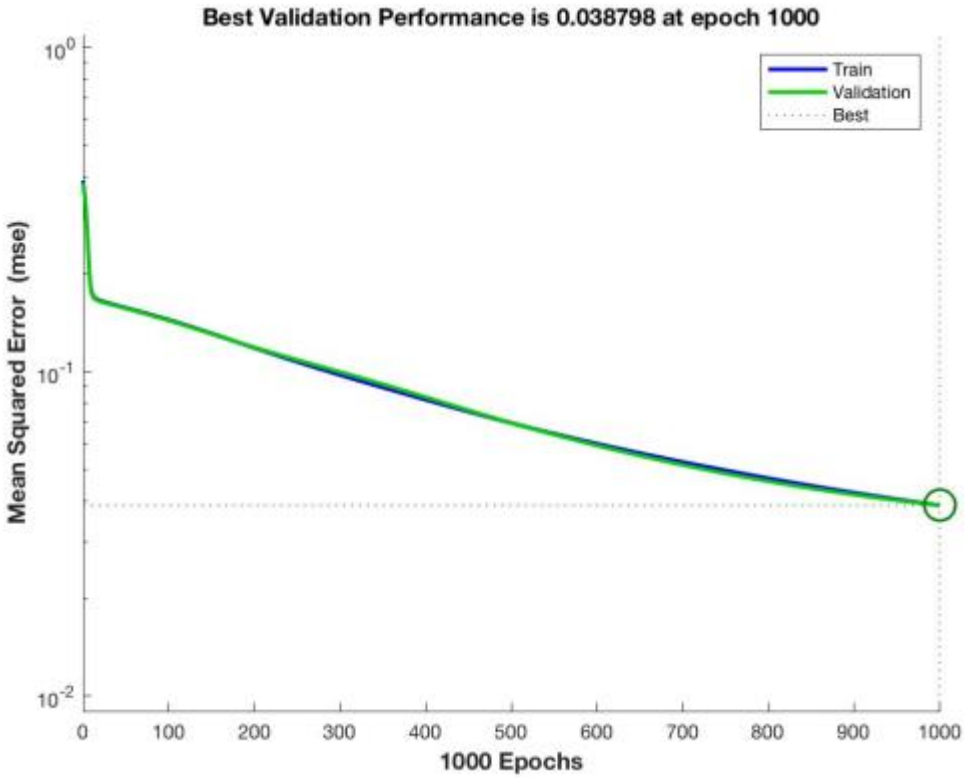
Ερώτημα 4

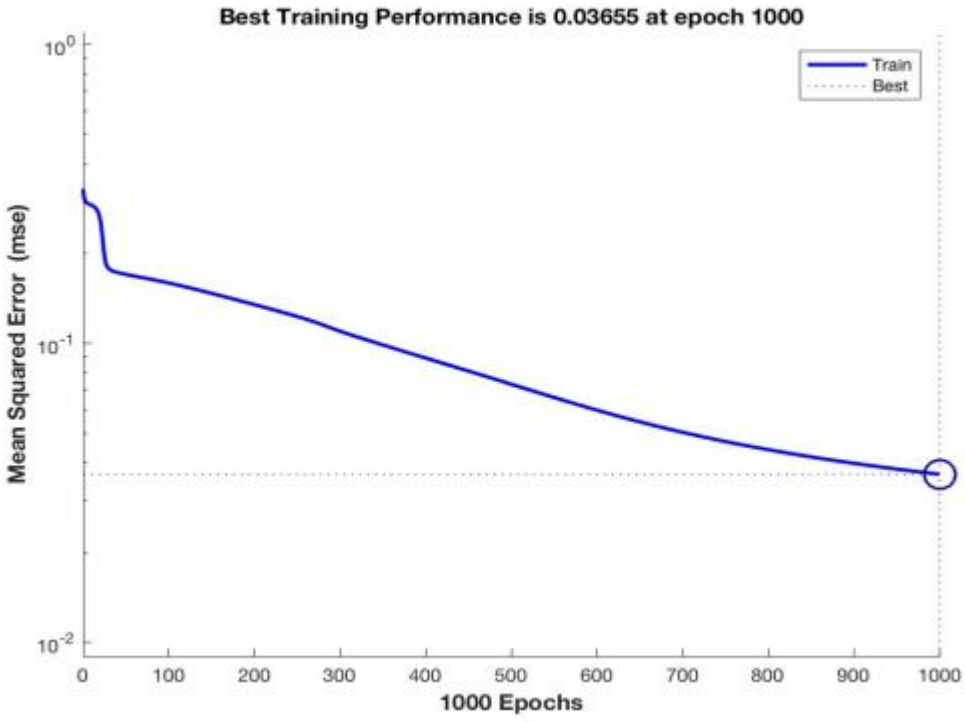
Η χρήση και των 2 αλγορίθμων έδινε καλές επιδόσεις στο δίκτυο αλλά ο αλγόριθμος learngdm (μάθησης απότομης καθόδου με προσθήκη του όρου ορμής) έδινε ελαφρός καλύτερα αποτελέσματα, για αυτό και επιλέχθηκε.

Ερώτημα 5

Η μέθοδος Early Stopping χωρίζει τα δεδομένα σε ένα σύνολο δεδομένων εκπαίδευσης, ένα σύνολο δεδομένων επαλήθευσης και ένα σύνολο δεδομένων test. Στην περίπτωση που τα αποτελέσματα για τα δεδομένα επαλήθευσης δεν είναι ικανοποιητικά τότε τερματίζεται η εκπαίδευση του νευρωνικού δικτύου. Η συνάρτηση traingd που χρησιμοποιήθηκε τρέχει by default 1000 εποχές. Ακόμα και όταν

χρησιμοποιηθεί validation set η συνάρτηση τρέχει 1000 φορές καθώς ο αριθμός διαδοχικών εποχών που μπορώ να έχω χειρότερα αποτελέσματα για τα δεδομένα επαλήθευσης σε σχέση με προηγούμενες εποχές δεν φτάνει ποτέ στο μέγιστο επιτρεπτό αριθμό (6) ώστε να τερματιστεί ο αλγόριθμος νωρίτερα.

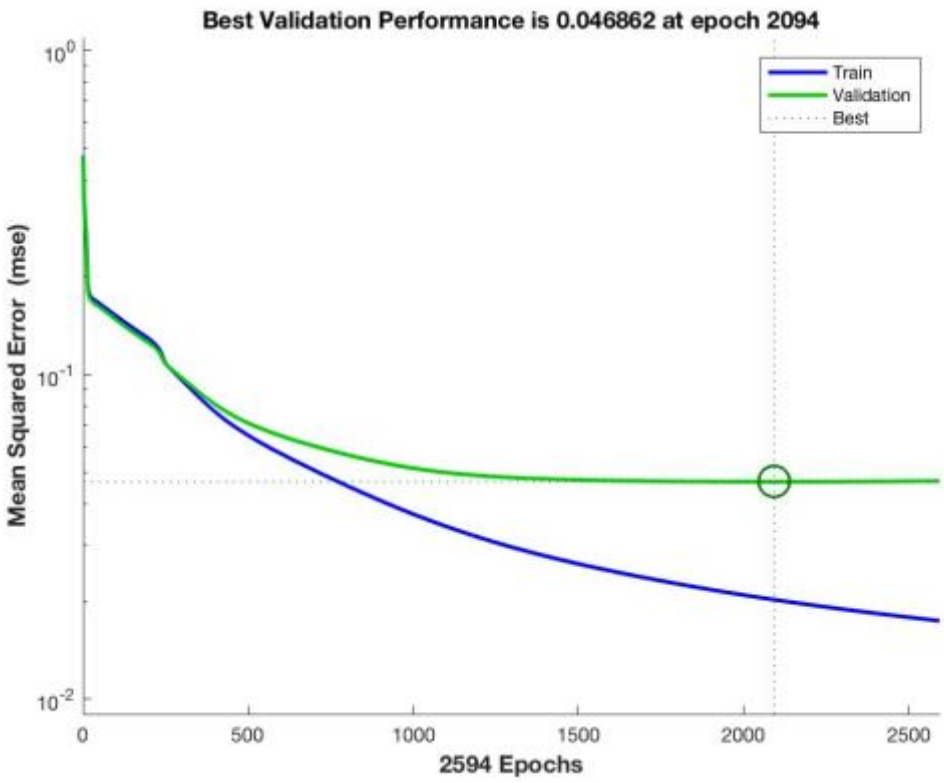


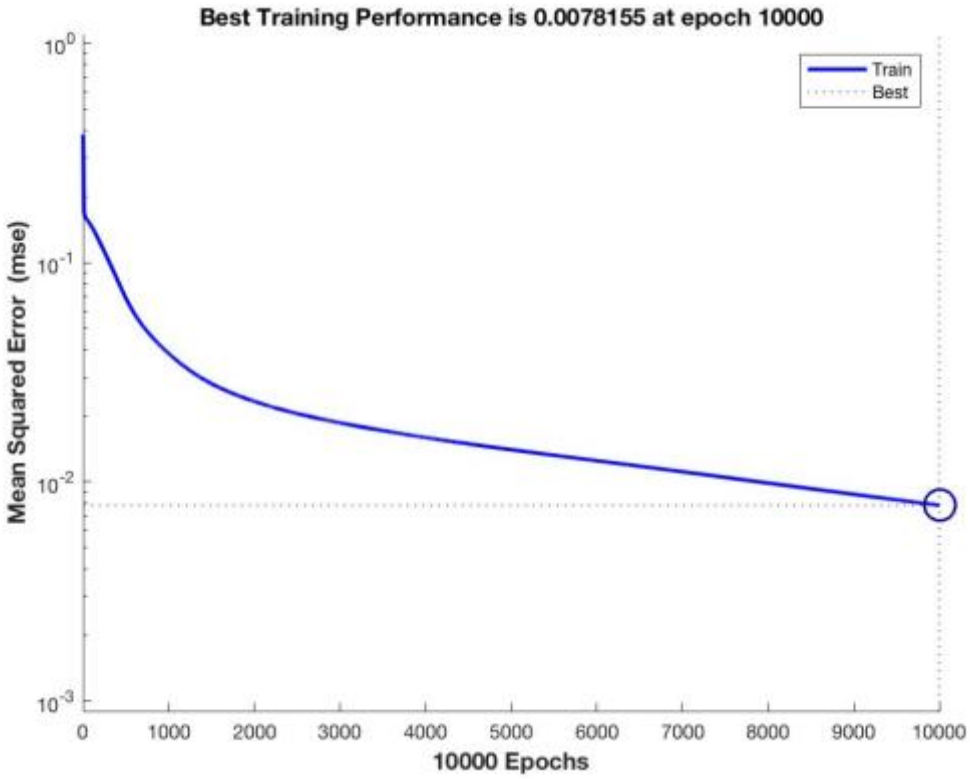


Για να μπορέσω να δω πιο αναλυτικά τι γίνεται με τη συνάρτηση traingd αύξησα τον αριθμό των εποχών που τρέχει η συνάρτηση και τον αριθμό διαδοχικών εποχών που μπορώ να έχω χειρότερα αποτελέσματα για τα δεδομένα επαλήθευσης σε σχέση με προηγούμενες εποχές (validation failures).

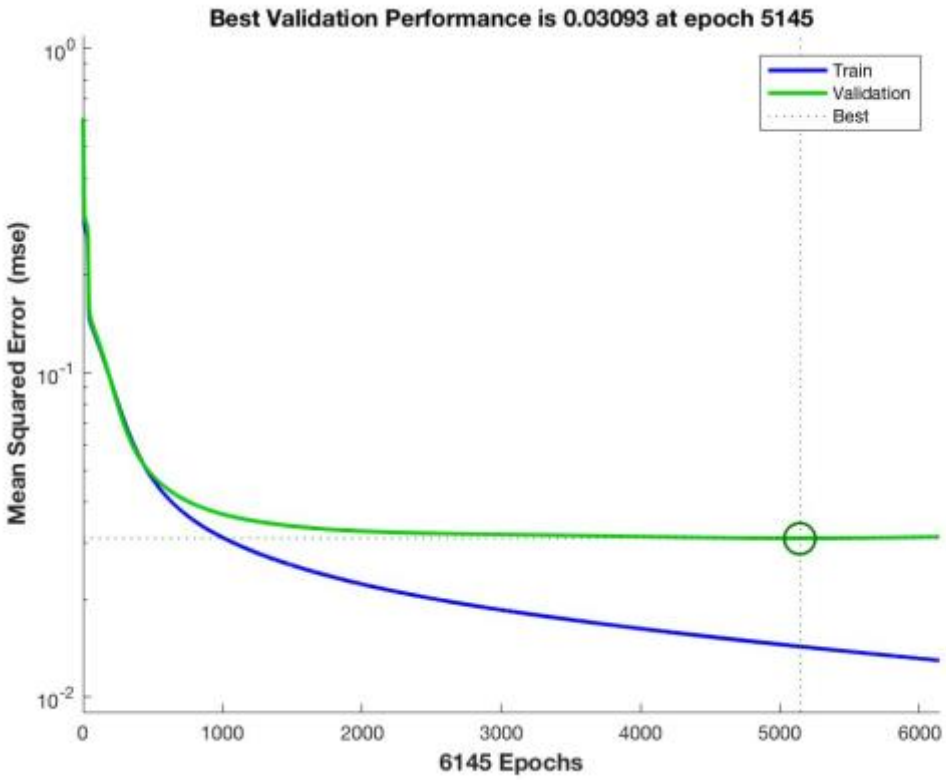
Ενδεικτικές περιπτώσεις:

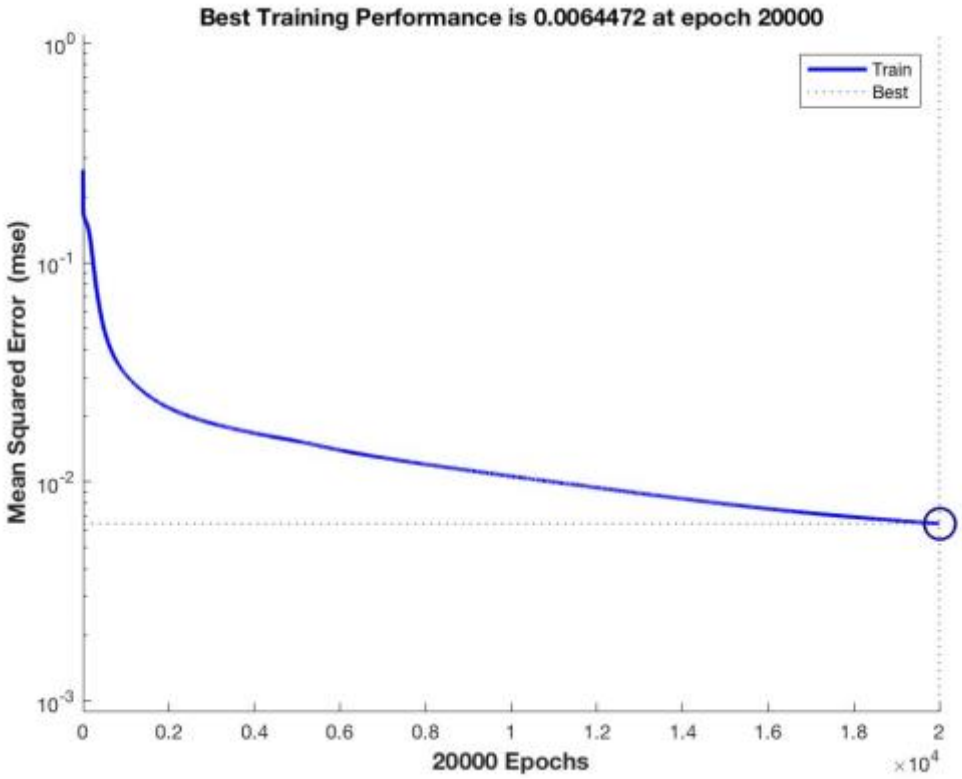
→ εποχές = 10000, validation failures = 500



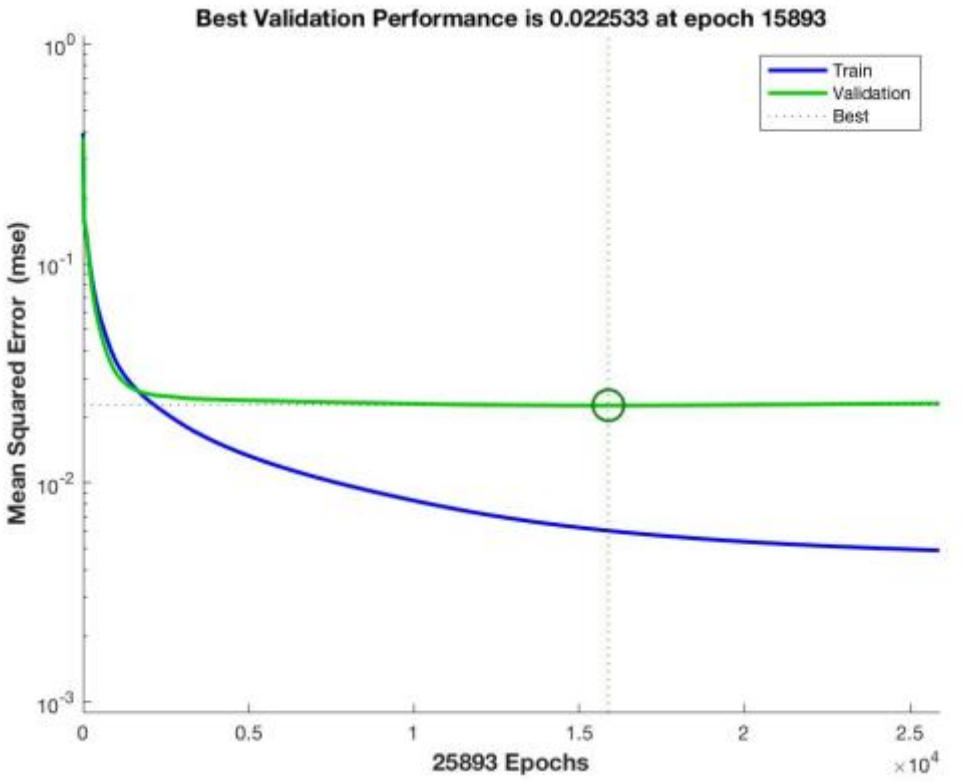


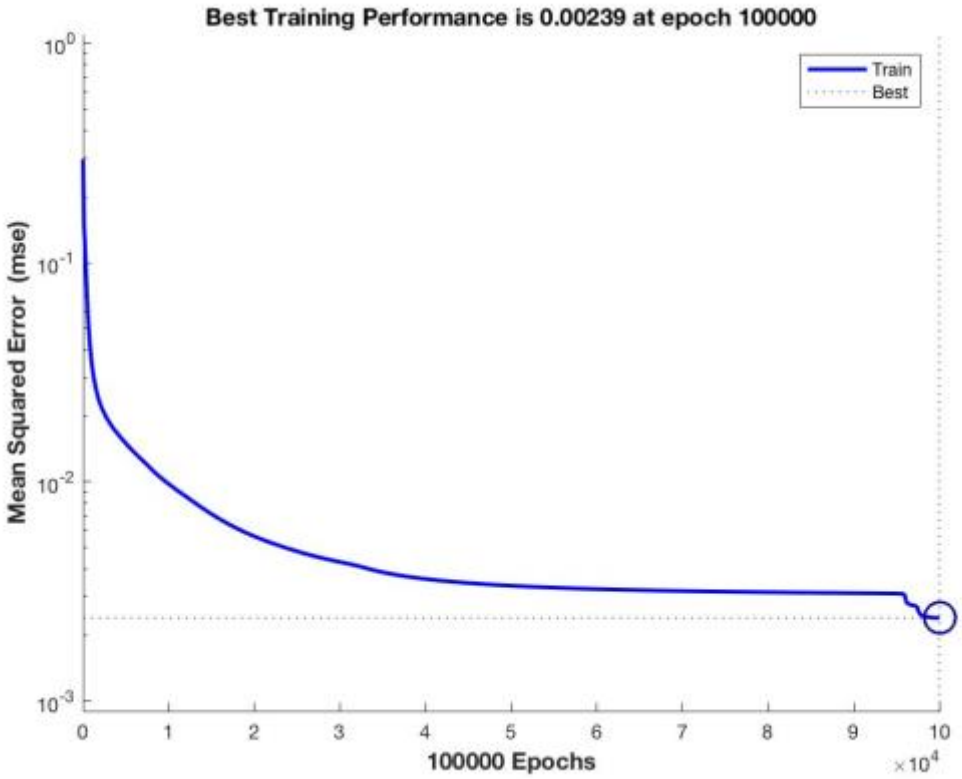
→ εποχές = 20000, validation failures = 1000





→ εποχές = 100000, validation failures = 10000





Φαίνεται λοιπόν ότι από ένα σημείο και μετά η απόδοση στο validation set δεν μπορεί να βελτιωθεί, καθώς σταθεροποείται. Η απόδοση στο training set συνεχίζει να βελτιώνεται όσο περνάνε οι εποχές και κάποια στιγμή ασυμπτωτικά το σφάλμα θα φτάσει στο μηδέν (πολύ μικρή τιμή).

Σε περίπτωση όμως που γινόταν χρήση μίας από τις άλλες τρεις συναρτήσεις θα υπήρχε διαφορά καθώς με τη μέθοδο του Early Stopping η συνάρτηση θα έτρεχε λιγότερες φορές από τη by default τιμή των εποχών (1000) καθώς ο αριθμός διαδοχικών εποχών που μπορώ να έχω χειρότερα αποτελέσματα για τα δεδομένα επαλήθευσης σε σχέση με προηγούμενες εποχές φτάνει στο μέγιστο επιτρεπτό αριθμό (6) ώστε να τερματιστεί ο αλγόριθμος νωρίτερα. Χαρακτηριστικό παράδειγμα είναι η συνάρτηση trainlm που τερματιζόταν μετά από 15-30 εποχές.

Ερώτημα 6

Η μείωση του αριθμού των εποχών στη συνάρτηση traingd είχε σαν αποτέλεσμα τη μείωση των επιδόσεων του νευρωνικού δικτύου καθώς δεν εκπαιδευόταν αρκετά. Όταν οι εποχές που τρέχει η συνάρτηση είναι πάρα πολλές (100000) τότε το σφάλμα σταθεροποιείται σε μία πολύ μικρή τιμή, ασυμπτωτικά προσεγγίζει το μηδέν.

Ερώτημα 7

→ traingd

Όταν ο ρυθμός μάθησης είναι πολύ μικρός (0.05, 0.1) η επίδοση του νευρωνικού δικτύου είναι μειωμένη. Με την αύξηση του ρυθμού μάθησης βελτιώνεται η επίδοση.

→ traingdx

Όταν ο ρυθμός μάθησης είναι πολύ μικρός (0.05, 0.1) η επίδοση του νευρωνικού δικτύου είναι καλή. Η τιμή 0.1 (default) είναι η βέλτιστη για τη συγκεκριμένη συνάρτηση εκπαίδευσης. Με την αύξηση του ρυθμού μάθησης στην ουσία το δίκτυο καταρρέει καθώς η επίδοσή του πέφτει πάρα πολύ. Γίνονται πολύ γρήγορα validation errors με αποτέλεσμα η συνάρτηση να σταματάει πολύ νωρίς και το δίκτυο να εκπαιδεύεται πολύ λίγο.

Ερώτημα 8

Γενικά η απόδοση του ταξινομητή είναι πολύ καλή. Στις κατηγορίες 1, 2 και 3 παρατηρείται ελαφρώς χαμηλότερο precision και στην κατηγορία 2 ελαφρώς χαμηλότερο recall. Βέβαια αυτό αλλάζει κάθε φορά που τρέχει το πρόγραμμα. Υπάρχουν και φορές που η απόδοση είναι πολύ καλή σε όλες τις κατηγορίες. Ένας τρόπος να βελτιώσω ακόμα περισσότερο την επίδοση του δικτύου θα ήταν να προσθέσω και άλλα δεδομένα, που ανήκουν στις κατηγορίες 1, 2 και 3, στο training set. Επίσης θα μπορούσα να δοκιμάσω να προσθέσω επιπλέον features.

4. Η μέθοδος της αποσύνθεσης βαρών

Στο ερώτημα αυτό χρησιμοποιώ τη μεθοδολογία κλαδέματος (pruning) της αποσύνθεσης βαρών (weight decay) για να εκπαιδεύσω το δίκτυο. Ο κώδικας με τα απαραίτητα σχόλια για την κατανοησή του υπάρχει στο φάκελο Code, στο αρχείο *ex1_pruning.m*. Η υλοποίηση έγινε με βάση τις οδηγίες της εκφώνησης της άσκησης.

Έτρεξα το πρόγραμμα με διάφορες τιμές του λ και του d. Ο όρος λ (regularization) χρησιμοποιείται για να αντιμετωπιστεί το φαινόμενο του overfitting. Η συγκεκριμένη εφαρμογή δεν παρουσιάζει overfitting.

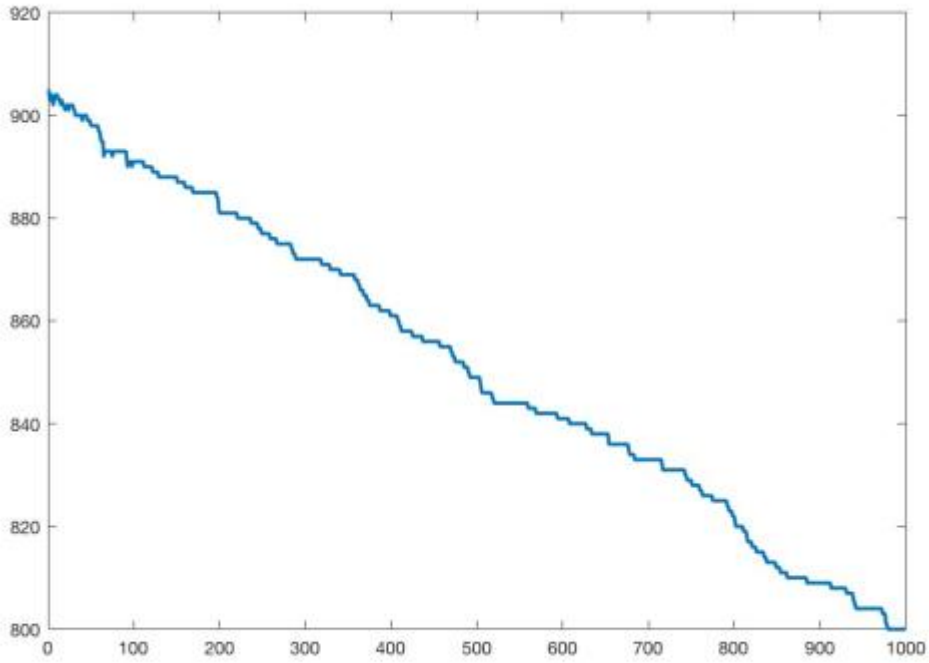
Γενικές παρατηρήσεις:

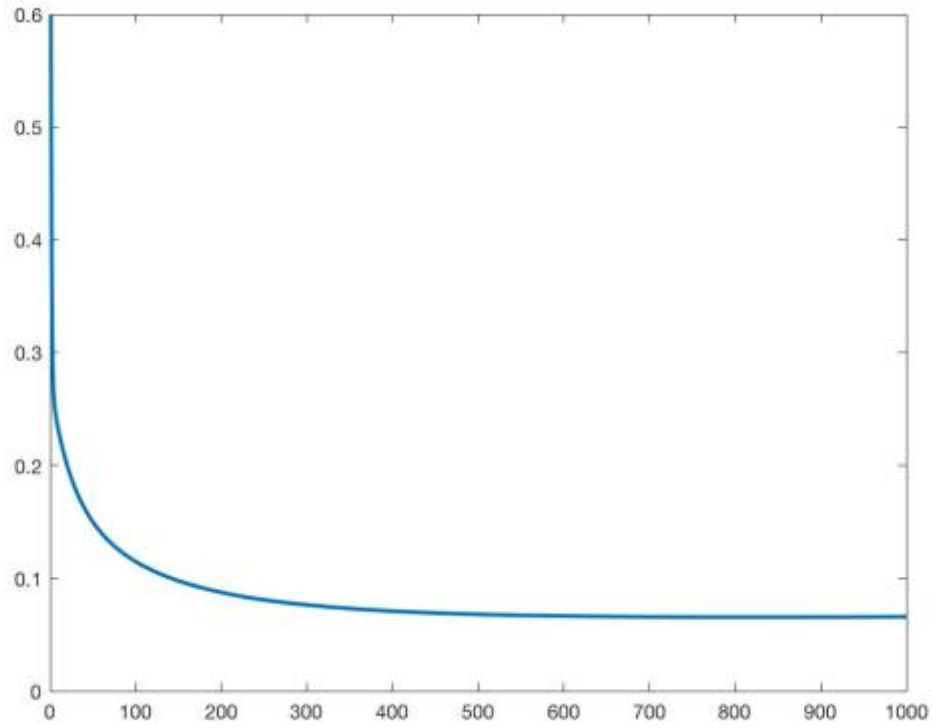
- Το πλήθος των βαρών που είναι μη μηδενικά μειώνεται όσο αυξάνονται οι επαναλήψεις.
- Η επίδοση του δικτύου βελτιώνεται όσο τρέχουν οι επαναλήψεις. Το σφάλμα μετά από μεγάλο αριθμό επαναλήψεων φτάνει σε μία ελάχιστη τιμή (της τάξεως του 0.06). Αυτό φαίνεται στη γραφική παράσταση του performance που γίνεται ασυμπτωτική μετά από πολλές επαναλήψεις.
- Οι επίδοση για μικρές τιμές του λ και d είναι πολύ υψηλές. Ενδεικτικά για λ = 0.001, d = 0.005:
    - accuracy: 0.915888
    - precision: 1.000000

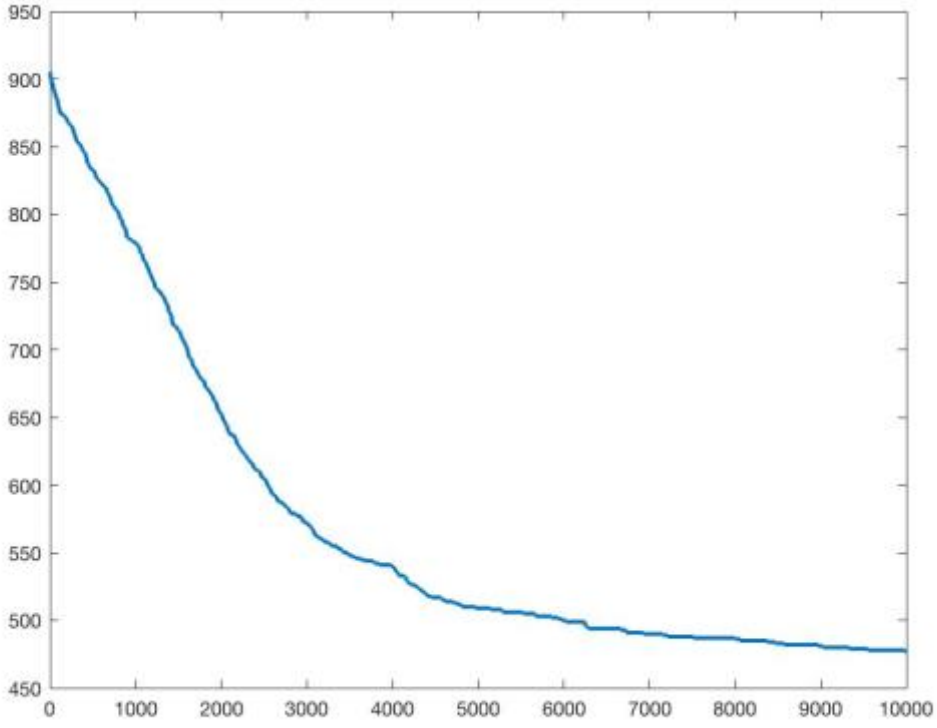
    - precision: 0.904762
    - precision: 0.714286
    - precision: 0.961538
    - precision: 1.000000
    - recall: 0.900000
    - recall: 0.791667
    - recall: 0.882353
    - recall: 1.000000
    - recall: 0.967742
- Όταν αυξάνονται οι τιμές του λ και d περισσότερα βάρη γίνονται μηδενικά και οι επιδόσεις του νευρωνικού δικτύου χειροτερεύουν.

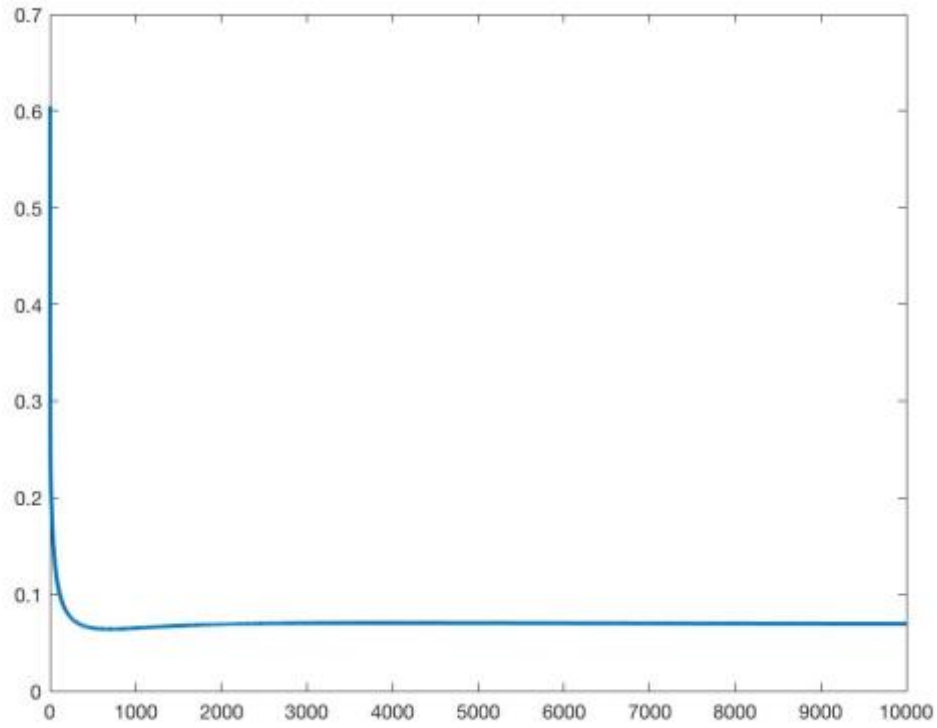
Γραφικές παραστάσεις για διάφορες τιμές του λ και του d:

→ λ = 0.001, d = 0.001




0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600
700
800
900
1000

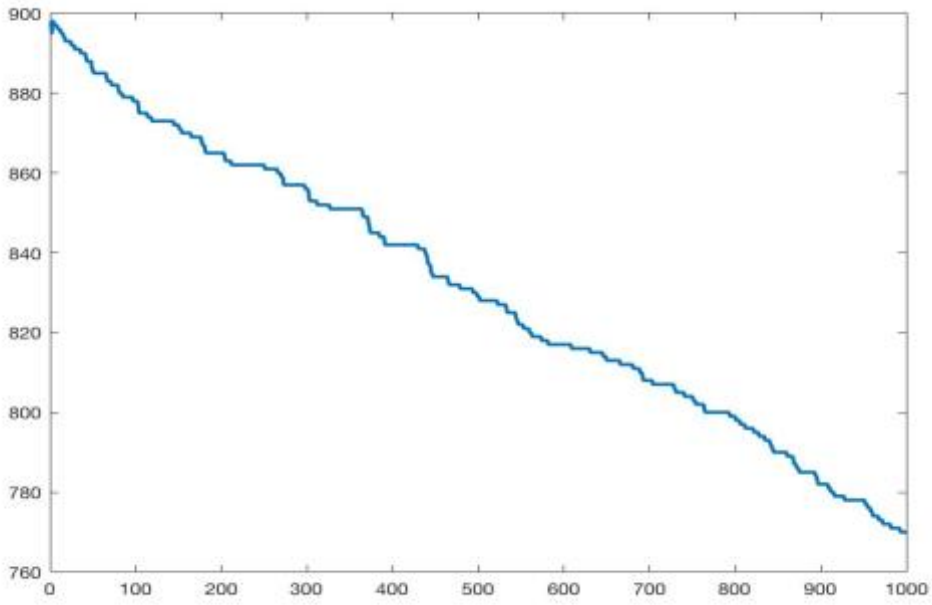

950
900
850
800
750
700
650
600
550
500
450
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

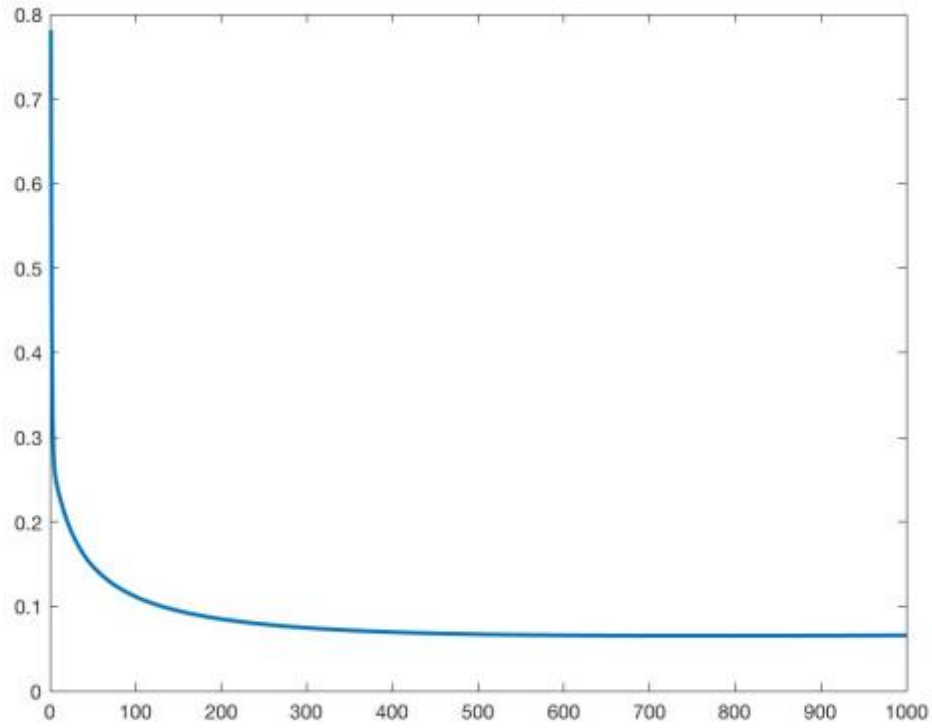


Σημείωση:

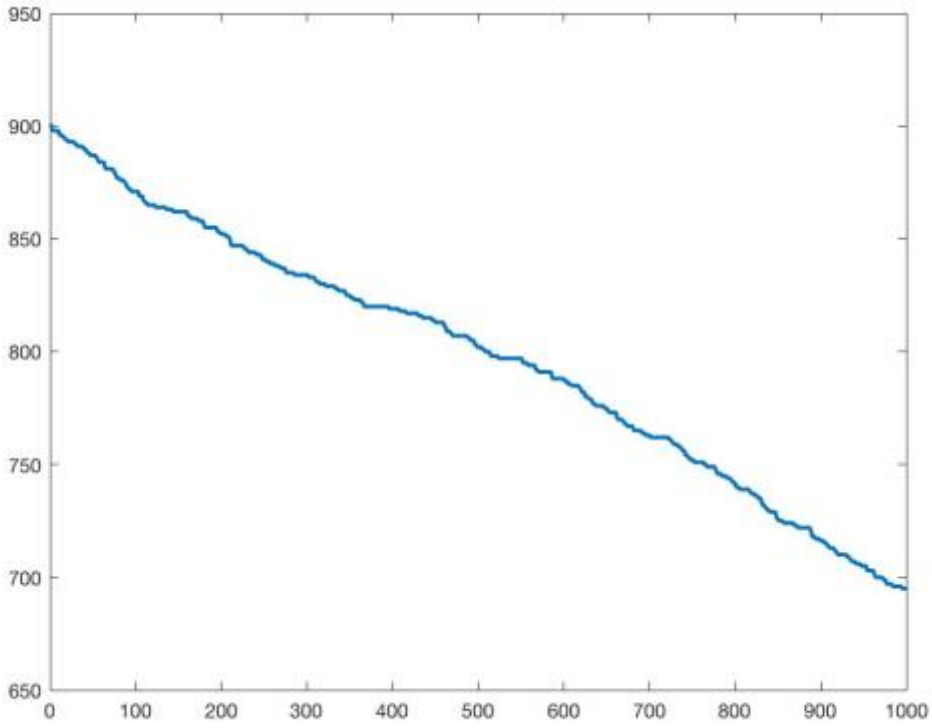
Οι δύο τελευταίες γραφικές προέκυψαν μετά από 10.000 επαναλήψεις.

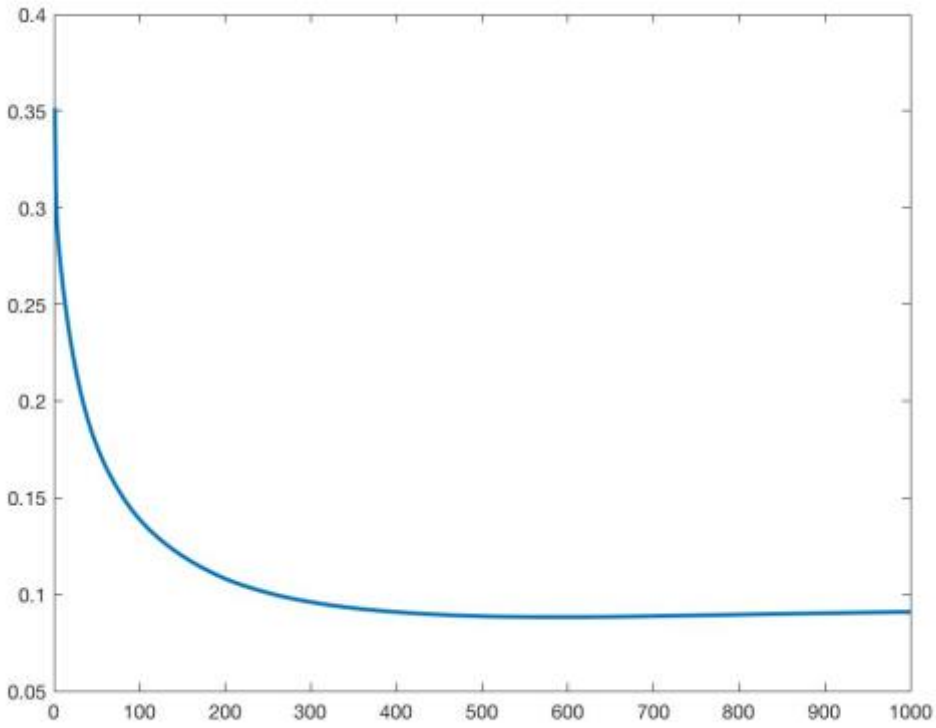
→ λ = 0.001, d = 0.005



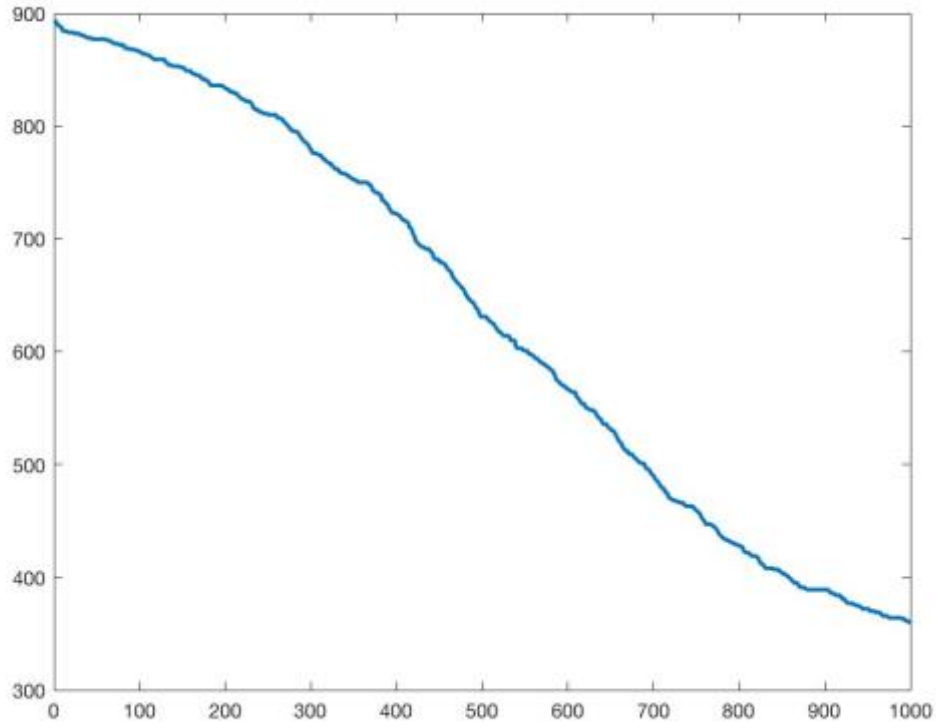


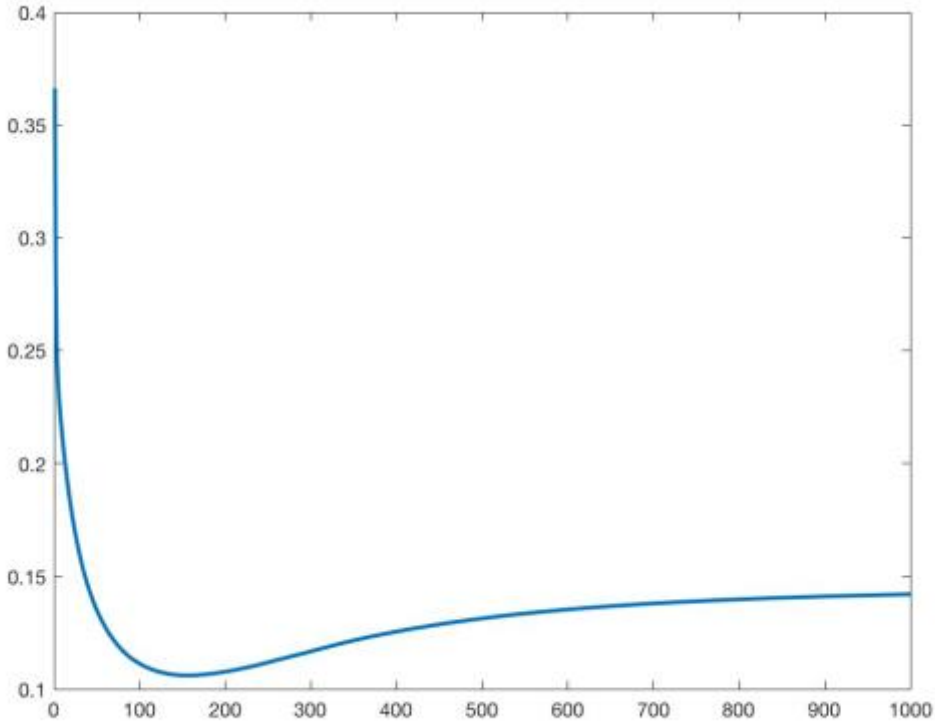
→ $\lambda = 0.002$, $d = 0.005$





→ $\lambda = 0.005$, $d = 0.008$

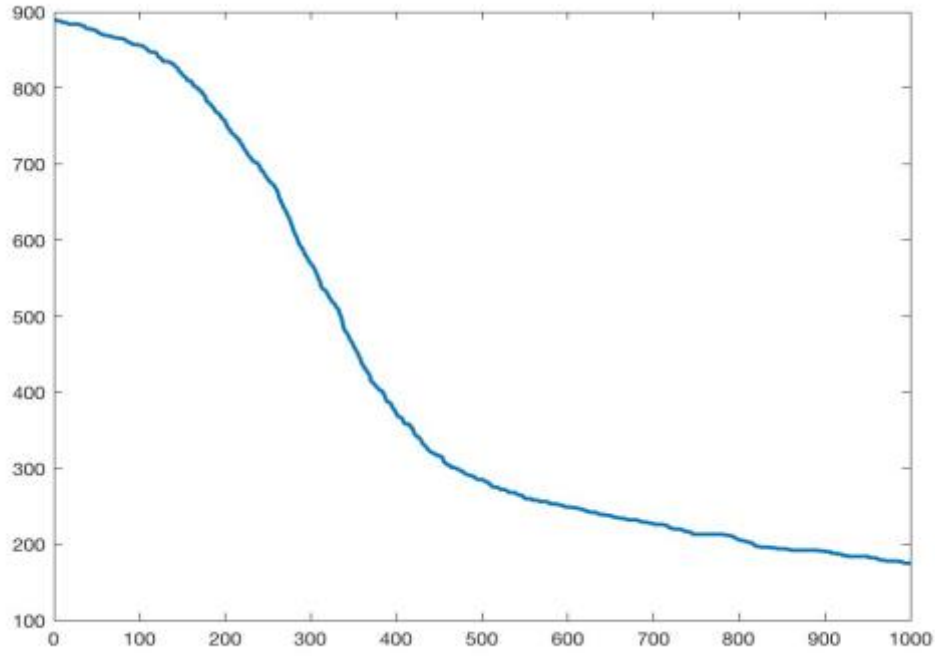


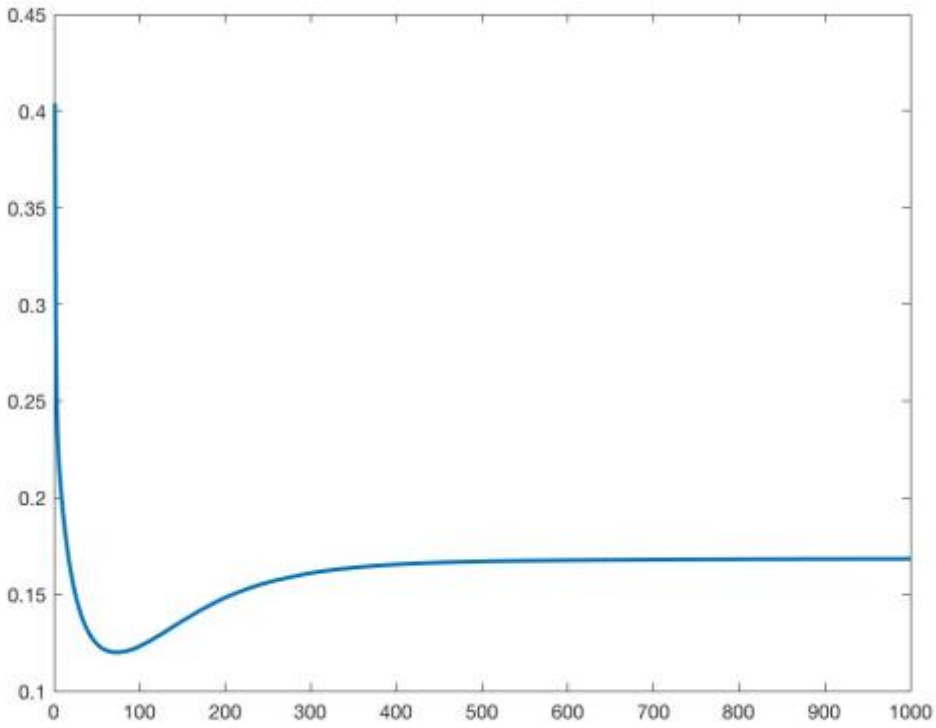


Σημείωση:

Στην περίπτωση αυτή τελικά πολλά βάρη γίνονται 0 και η επίδοση του δικτύου πέφτει μέχρι ένα σημείο και μετά αρχίζει να ανεβαίνει.

→ $\lambda = 0.01$, $d = 0.01$





Σημείωση:

Πλέον είναι προφανές ότι η αύξηση των τιμών του λ και του d έχουν αρνητικό αντίκτυπο στις επιδόσεις του δικτύου.