Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Ηλεκτρονικών υπολογιστών

Νευρωνικά Δίκτυα & Ευφυή Υπολογιστικά Συστήματα

Εργαστηριακή Άσκηση 1

Μελέτη των πολυεπίπεδων Perceptrons και εφαρμογή σε προβλήματα ταξινόμησης εικόνας

Αθανασίου Νικόλαος

03112074

9ο Εξάμηνο

Στην άσκηση αυτή καλούμαστε να δημιουργήσαμε ένα feedforward error νευρωνικό δίκτυο που χρησιμοποιεί αλγόριθμο back propagation.Για την καθολική κατανόηση του κώδικα παρατίθεται μια επεξήγηση της βασικής συνάρτησης του matlab που χρησιμοποιήσαμε για να κατασκευάσουμε το δίκτυο αυτό.Οι διαστάσεις που παρατίθονται παρακάτω αντιστοιχούν στα δεδομένα μετά από το κομμάτι του preprocessing που θα εξηγηθεί αναλυτικά στη συνέχεια:

newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF) takes optional inputs,

- **P:** Διάνυσμα εισόδου.Στην περίπτωση μας 20*535 που κάθε στήλη του αντιπροσωπεύει τα (MPEG-7)χαρακτηριστικά του τμήματος μιας εικόνας ή απλούστερα κάθε του στήλη περιέχει τα χαρακτηριστικά ενός τμήματος μιας εικόνας.Επομένως μετά την προεπεξεργασία έχουμε 535 τμήματα εικόνων
- **T:** Διάνυσμα στόχος εκπαίδευσης.Στην περίπτωση μας 5*535 στο οποίο η μεγαλύτερη τιμή σε κάθε διάσταση καθορίζει την κατηγορία που ανήκει το τμήμα αυτό.
- **S:** Αριθμός επιπέδων κρυφών νευρώνων.Για παράδειγμα αν S=[2] έχουμε 1 επίπεδο κρυφών νευρώνων με 2 νευρώνων.Αντίστοιχα για 2 στρώματα [#FirstLayer #SecondLayer] κτλ.
- **TFi:** Συνάρτηση μεταφοράς i-οστού επειπέδου.Αν δεν καθοριστεί είναι η tansig για όλα τα κρυμμένα επίπεδα και η purelin για το στρώμα εξόδου.
- **BTF:** Συνάρτηση εκπαίδευσης BackpropagationAν δεν οριστεί χρησιμοποιείται η **trainlm**.
- **BLF:** - Backpropagation συνάρτησης εκμάθησης,προκαθορισμένη=learngdm.
- **PF:** - Performance function, Χρησιμοποιείται καθ'όλη την άσκηση η προκαθορισμένη 'mse'.
- **IPF:** Προκαθορισμένες {fixunknowns,remconstantrows,mapminmax}.
- **OPF:** Προκαθορισμένες {'remconstantrows','mapminmax'}.
- **DDF:** Προκαθορισμένη= 'dividerand';

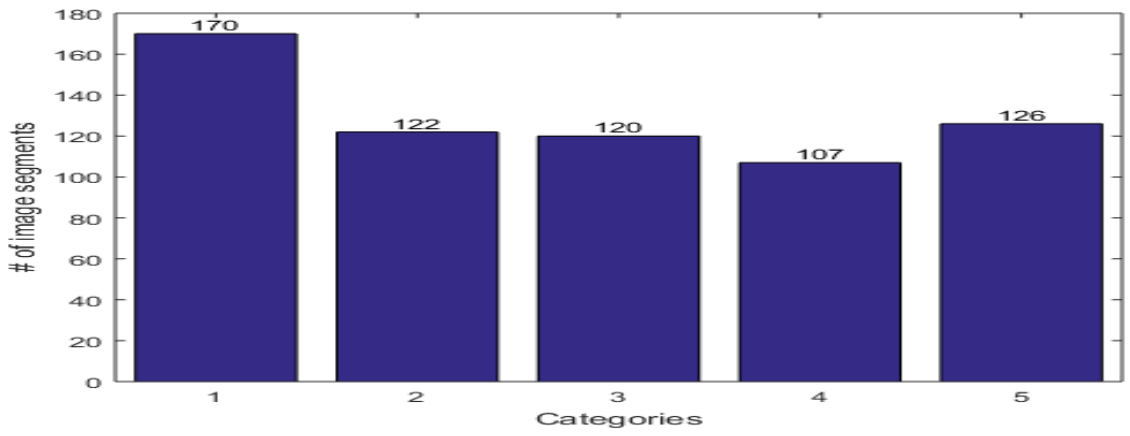
Και μας επιστρέφει ένα feedforward backpropagation δίκτυο με τις παραμέτρους που του ορίσαμε.Εδώ να σημειώθει ότι οι 4 τελευταιές παράμετροι που δεν εξηγήθηκαν δεν μελετήθηκαν ούτε μας απασχόλησαν κατά την άσκηση και χρησιμοποιήθηκαν οι προκαθορισμένες τιμές τους.

Αντίστοιχα για τον έλεγχο των αποτελεσμάτων του δικτύου χρησιμοποιούνται οι αντίστοιχοι πίνακες TestData TestDataTargets με παραπλήσσιες διαστάσεις οι οποίες δίνονται ως παράμετρο στην συνάρτηση sim μαζί με το δίκτυο που δημιουργούμε όπως περιγράφουμε παραπάνω για να ελέγξουμε τη συμπεριφορά του δικτύου μας.Στη συνέχεια με βάση τα παραπάνω σχόλια επεξηγούνται και αναλύονται συνοπτικά το πως υλοποιήθηκε κάθε βήμα στην παρούσα άσκηση με βάση τον κώδικα.

Προεπεξεργασία των Δεδομένων

Βήμα 1

Αρχικά φορτώνουμε τα δεδομένα μας και χρησιμοποιώντας την εντολή sum κατά την δεύτερη διάσταση δηλαδή κατά γραμμές βρίσκουμε πόσα τμήματα εικόνας ανήκουν σε κάθε κατηγόρια.



Στη συνέχεια επιλέγουμε να κρατήσουμε το ελάχιστο από το παραπάνω αποτέλεσμα ώστε να κρατήσουμε τον ίδιο αριθμό από κάθε κατηγορία και έπειτα κρατάμε μέσω ενός πίνακα δεικτών τόσα(ελάχιστο από τις μπάρες) στοιχεία και στη συνέχεια εργαζόμαστε όπως υποδεικνύεται στην παρουσίαση της άσκηση αναμιγνύουμε αντίστοιχα τους απαιτούμενους πίνακες

Βήμα 2

Στο παρόν βήμα με τη χρήση των συναρτήσεων *removeconstantrows() processpca()* αφαιρούμε τις αμετάβλητες γραμμές και μειώνουμε τις διαστάσεις των δεδομένων.Ότι αλλαγή εφαρμοζέται στα TrainData εφαρμόζεται και στα TestData με τη βήθεια της *mapstd()* με παράμετρο 'apply'.Αντίστοιχα εργαζόμαστε και για τα TrainDataTargets ,TestDatTargets.

Μελέτη της Αρχιτεκτονικής του Νευρωνικού Δικτύου

Βήμα 3

Δημιουργούμε το δίκτυο όπως περιγράφηκε και θέτουμε τις αντίστοιχες παραμέτρους με πρόσβαση στην παράμετρο του struct του δικτυου
`divideParam.trainRatio=0.8,testRatio=0.0,valRatio=0.2`

Βήμα 4

Θέτουμε την 3η παράμετρο της newff παραμετρικά από 5 μέχρι 30 νευρώνες για κάθε επίπεδο επαναληπτικά.Πειραματιζόμαστε με ένα και δύο επίπεδα.

Βήμα 5

Σχηματίζουμε ένα cell array με τα ονόματα των συναρτήσεων εκπαίδευσης και τις δίνουμε παραμετρικά στην newff και πάλι επαναληπτικά πειραματιζόμενοι για τους συνδυασμούς νευρώνων-επιπέδων του Βήματος 4.

Βήμα 6

a. Παρόμοια με το βήμα 5 δίνουμε παραμετρικά στο δίκτυο μας τις διαφορετικές συναρτήσεις εξόδου επαναληπτικά μέσω ενός cell array.

b. Και πάλι εργαζόμαστε ομοίως όμως σε αυτή την περίπτωση πρέπει να χρησιμοποιήσουμε ως συνάρτηση εκπαίδευση την **traingd** γιατί ορίζει ένα όρο ανανέωσης βάρους η learning function μας .Επίσης πρέπει να αυξσουμε τον ρυθμό εκμάθησης ως εξής `net.trainParam.lr` ώστε να έχουμε θετικά αποτελέσματα

c. Απλώς θέτουμε `divideParam.valRatio=0 .`

d. Εργαζόμαστε όμοια με το c.

e. Θέτουμε την παράμετρο `net.trainParam.lr` επαναληπτικά σύμφωνα με την εκφώνηση έχοντας τη μία φορά συνάρτηση εκπαίδευσης την traigd και μία την traingdx

Βήμα 7

Όπως θα φανεί και από τα παρακάτω το δίκτυο με μεγάλη ικανότητα γενίκευσης και καλά αποτελέσματα είναι αυτό με συνάρτησης εκπαίδευσης την trainlm συνάρτηση ενεργοποίησης εξόδου tansig ή purelin και δύο επίπεδα νευρώνων με 15-20 νευρώνες το καθένα .

Οι traingd traingdx είναι αλγόριθμοι εκπαίδευσης απότομης καθόδου(gradient descent) ο triangda ειναι και αυτός gradient descent με προσθήκη όρου ορμής ενώ σε αντίθεση με τους παραπάνω ο trainlm χρησιμοποιοεί τη μέθοδο εντάσσεται στους quasi Newton αλγόριθμους και ειδικότερα στον Levenberg-Marquardt εντοπίζει το ελάχιστο της συνάρτησης μέσω αθροίσματος τετραγώνων με επαναληπτικό τρόπο έτσι ώστε σε κάθε βήμα να ελαττώνει τη συνάρτηση επίδοσης.Αυτό το χαρακτηριστικό του τον κάνει να είναι ο καλυτερος όπως φαίνεται για λογικά μεγέθη δεδομένων.Λόγω όμως του ότι χρειάζετια αρκετή μνήμη και έχει μεγάλο υπολογιστικό κόστος (υπολογισμός παραγώγου προσέγγιση Hessian) τον κάνει ιδιαίτερα αργό για μεγάλο όγκο δεδομένων και απαιτητικό σε μνήμη.

Στη συνέχεια επεξηγούνται μερικές απότις βασικές συναρτήσεις που θα χρησιμοποιηθούν:

Learngdm(gradient descent με όρο ορμής) : υπολογίζει το διαφορά βάρους dW για κα'ποιον νευρώνα ως εξής :

$$dW = (momentum\ const.) * dW_{prev} + (1 - momentum\ const.) * (learning\ ratio) * (grad.)$$

dW = mc*dWprev + (1-mc)**gW

mc σταθερά ορμής

lr ρυθμός εκμάθησης

learngd υπολογίζει τη διαφοράς βάρους ως εξής:

$$dW = (learning\ ratio) * (grad.)$$

traingdx έχει και όρο ορμής αλλά και προσαρμοστικής μάθησης

$$dX = (momentum\ const.) * dX_{prev} + (momentum\ const.) * (learning\ ratio) * (grad.)$$

Με X να είναι τα βάρη ή bias και grad είναι ως γνωστόν η παράγωγος της επίδοσης(διαφορά σφάλματος από επιθυμητό) ως προς τα βάρη όπως και σε όλες τις υπόλοιπες προαναφερθείσες συναρτήσεις.

Χρησιμοποιεί την εξής προσέγγιση για τον Hessian πίνακα μέσω της Jacobian:

$$\mathbf{H} = \mathbf{J}^T\mathbf{J}$$

Και υπολογίζει την παράγωγο ώς:

$$\mathbf{Grad/} = \mathbf{J}^T\mathbf{e}$$

Αυτό γιατί ο jacobian υπολογίζεται πολύ απλούστερα από τον Hessian.

Ο Levenberg-Marquardt (trainlm) ενημερώνει τα βάρη ως εξής :

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e}$$

Για μ μεγάλο θυμίζει μέθοδο απότομης καθόδου όμως αυτό που αποσκοπεί η μέθοδος είναι μικραίνει συνεχώς ώστε σταδιακά να προσεγγίσει τη μέθοδο Newton η οποί συγκλίνει γρήγορα κια με επιτυχία.

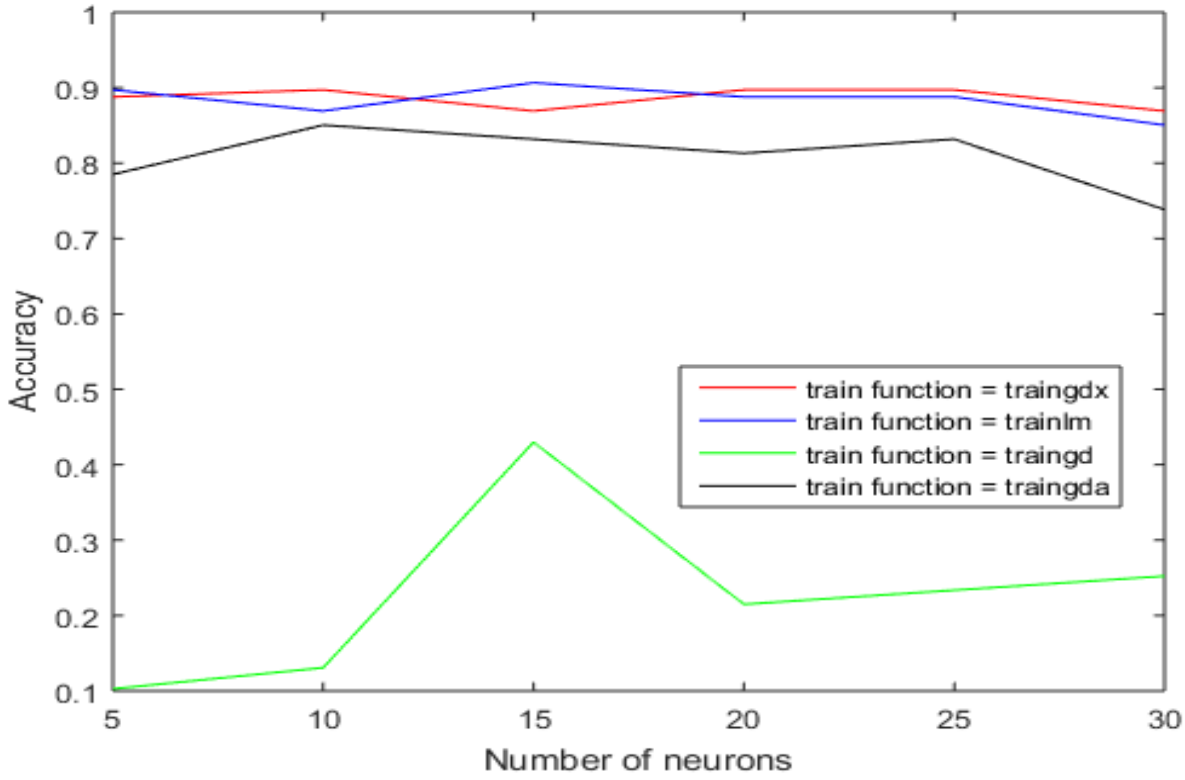
Ερωτήματα

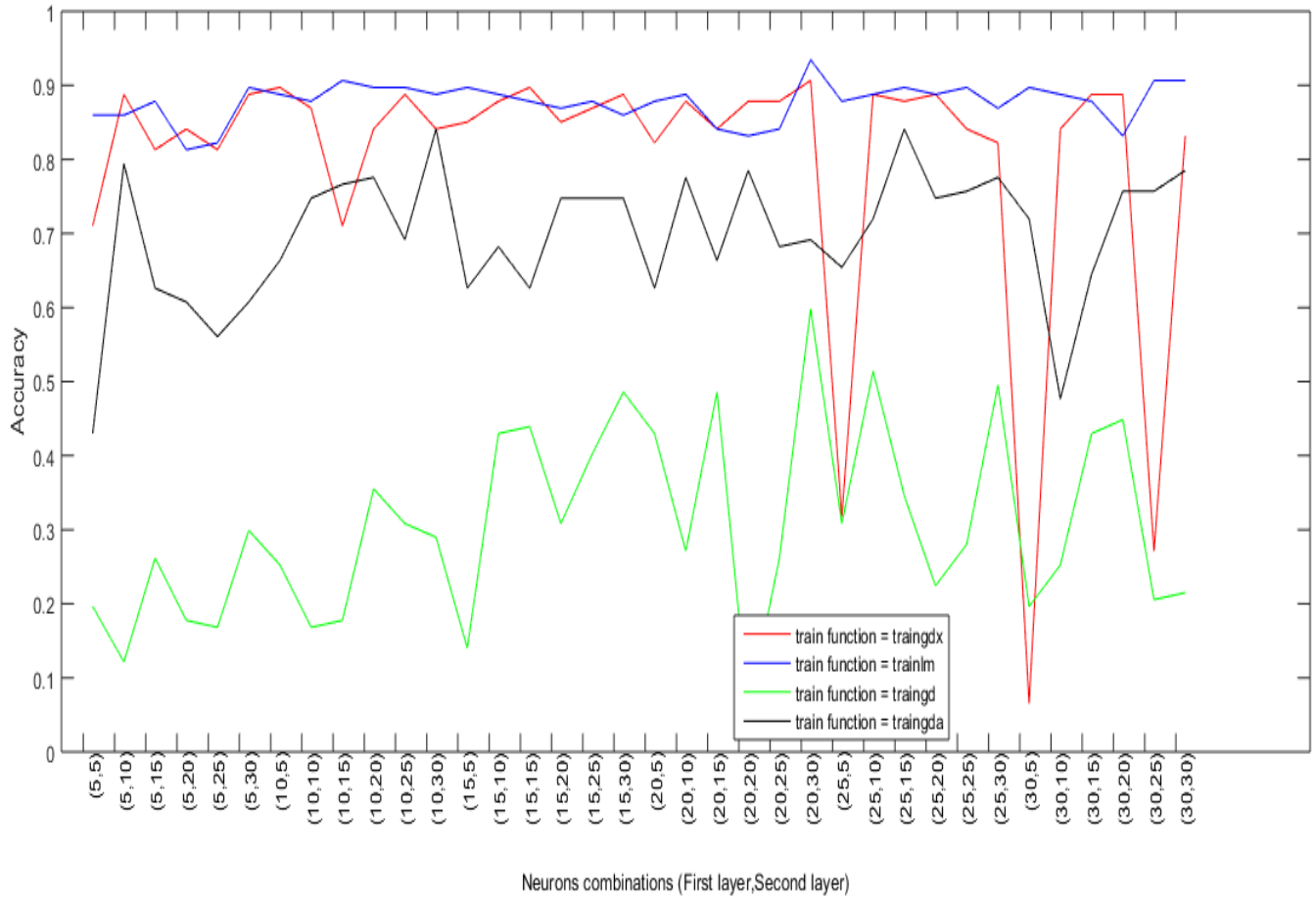
Ερώτημα 1

Αρχικά βρίσκουμε πόσο κομμάτια κάθε εικόνας περιέχει κάθε κατηγορία και στη συνέχεια κρατάμε το ελάχιστο αυτών ώστε να μη μεροληπτήσουμε απέναντι σε κάποια κατηγορία.Έπειτα αντιμεταθέτουμε τυχαία τα train στόχους δεδομένα ώστε να εξασφαλίσουμε τυχαιότητα στα δεδομένα και να μην εκπαιδεύουμε μονόπλευρα το μοντέλο μας.Στη συνέχεια φροντίζουμε για τα data και test αντίστοιχα δεδομένα μας με τη βοήθεια της mapstd ώστε το ελάχιστο και το μέγιστο κάθε γραμμής να mapαριστούν σε μέση τιμή 0 και τυπική απόκλιση 1 δηλαδή να ακολουθούν κανονική κατανομή.Επεξεργαζόμαστε τα test δεδομένα μας όπως τα train με τη βοήθεια της mapstd με παράμετρο apply και τα settings που επιστράφηκαν από την mapstd στα traindata.Αυτό το κάνουμε έτσι ώστε να δημιουργήσουμε ομοιομορφία στα δεδομένα μας και την κατανομή τους.Στη συνέχεια με χρήση της removevonstantrows() αφαιρούμε τις σταθερές γραμμές από τα δεδομένα μας καθώς δεν έχουν κάποια διασπορά καθώς για τη σωστή εκπαίδευση του δικτύου μας πρέπει τα δεδομένα μας να έχουν απόκλιση από τη μέση τιμή ώστε να περιέχουν χρήσιμη πληροφορία αποφεύγοντας έτσι την υπερκπαίδευση του δικτύου και την προσθήκη πολυπλοκότητας χωρίς ανάλογη βελτίωση των αποτελεσμάτων(overfit).Τέλος, χρησιμοποιώντας την proccespca() ελαττώνουμε τη διάσταση του προβλήματος μας(δηλαδή των δεδομένων μας TrainData) και με την βοήθεια και πάλι της mapstd εφαρμόζουμε τις παραπάνω αλλαγές και στα TestData μας για να εξασφαλίσουμε την ομοιοκατανομή train και test.Η τελευταία αλλαγή είναι αυτή που μειώνει κατά πολύ την πολυπλοκότητα και μειώνει τον κίνδυνο υοερεκπαίδευσης των δεδομένων,

Ερώτημα 2

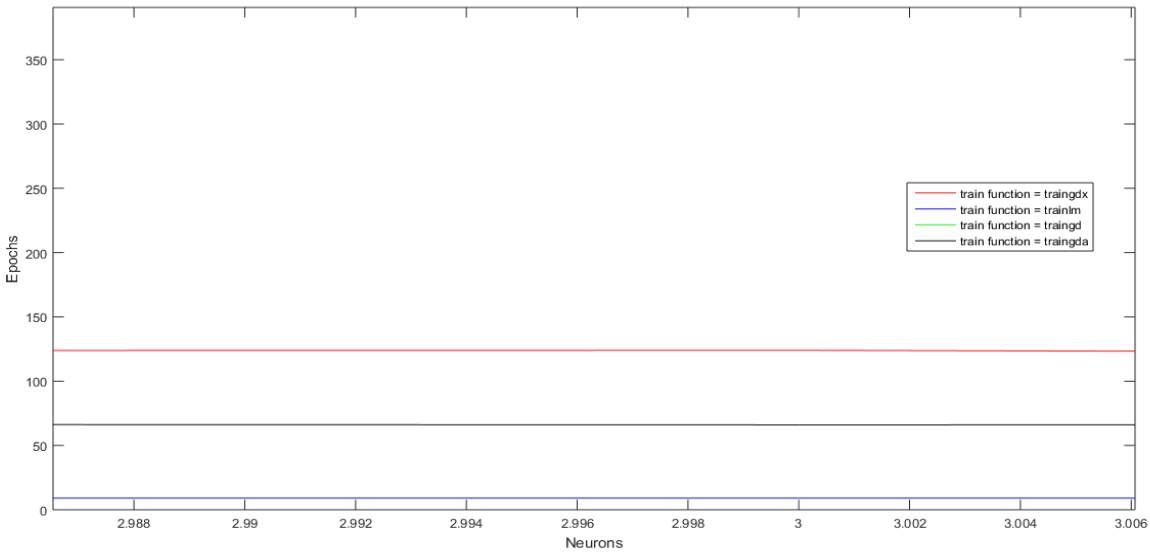
Παρατηρούμε αρχικά στο βήμα 4 ότι τα 2 επίπεδα νευρώνων όπως φαίνεται και από το παρακάτω σχήμα δίνουν σαφώς καλύτερα αποτελέσματα από ότι ένα κρυφό επίπεδο νευρώνων αλλά ιδιαίτερα για τις συναρτήσεις που αδυνατούν να συγκλίνουν όπως η traingd η οποία για να συγκλίνει θα πρέπει να αυξήσουμε τις εποχές ή το ρυθμό εκμάθησης της -learning rate- .Δεν σταματάει με τη μέθοδο Early Stopping και εγκλωβίζεται σε τοπικά ελάχιστα μεταπειδώντας από τη μία πλευρά του ελάχιστου στην άλλη αφού δεν έχει όρο ορμής ώστε να υπερβεί αυτές τις ταλαντώσεις.Η συνάρτηση traingdx έρχεται δεύτερη καθώς πετυχαίνει πάρα πολύ καλά αποτελέσματα ακόμη και με ένα επίεπδο νευρώνων αλλά τα καλύτερα αποτελέσματα ανήκουν στην trainlm.Η μεγάλη επιτυχία της traingdx οφείλεται στη χρήση όρου ορμής που βοηθάει στη γρήγορη σύγκλιση αλλά και στον όρο προσαρμοστικής μάθησης που διαθέτει καθώς αλλάζει τη ταχύτητα καθόδου προς την αρνητικότερη παράγωγο και συγκρατεί τις απότομες αλλαγές των βαρών αποφεύγοντας έτσι ταλαντώσεις γύρω από τοπικά ελάχιστα.Τρίτη έρχεται η μέθοδος traingda από την οποία λείπει ο όρος ορμής αλλά έχει όρο μάθησης κάνοντας την καλύτερη από την traingd αλλά χειρότερη από την traingdx αφού κολλάει σε τοπικά ελάχιστα.Τα καλύτερα αποτελέσματα έδωσε η trainlm η οποία όπως προαναφέρθηκε χρησιμοποιεί μαι προσέγγιση Hessian πίνακα.Όσο το μ είναι μεγάλο είναι αλγόριθμος απότομης καθόδου ενώ όσο μικραίνει προσεγγίζει όλο και πιο πολύ τη μέθοδο Newton.Αυτό ο συνδυασμός οδηγεί το μ να μικραίνει αργά ελέγχοντας τη συνάρτησης επίδοσης του δικτύου και αυξάνεται μόνο αν βελτιώνονται σημαντικά τα αποτελέσματα πχ κοντα ολικο ελάχιστο.Το μόνο κακό αυτής της μεθόδου είναι το μεγάλο memory και υπολογιστικό overhead που έχει επιβραδύνοντας τη διαδικασία για πολλά δεδομένα πράγμα που την κάνει πολλές φορές δύσχρηστη παρά την επιτυχία της.

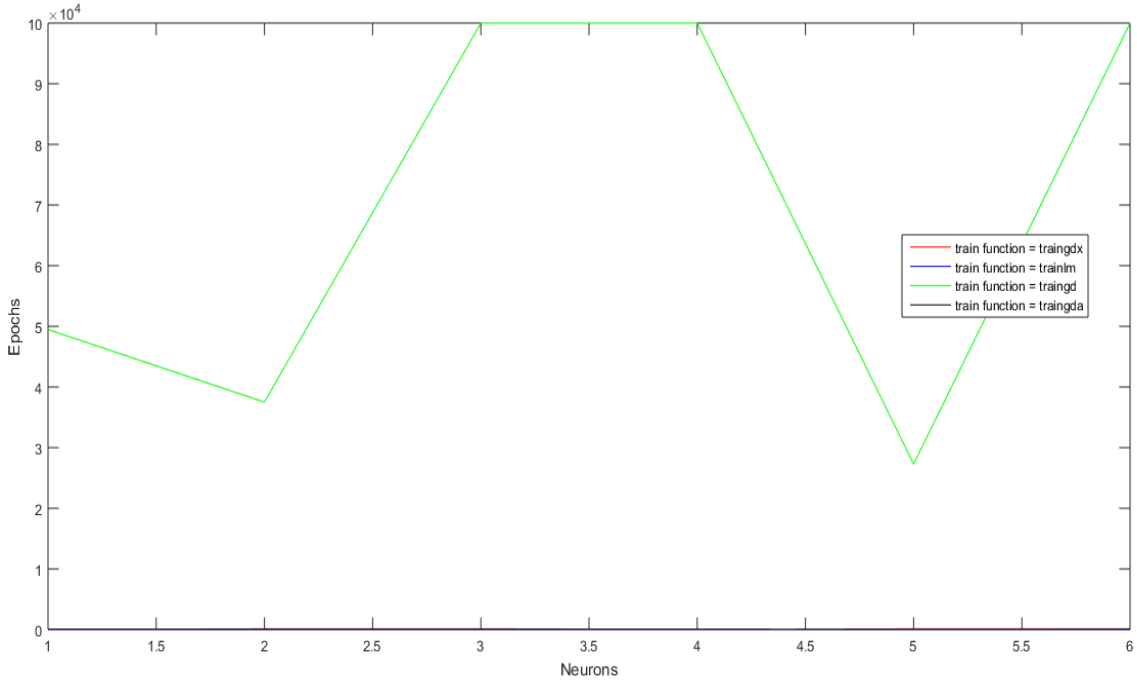




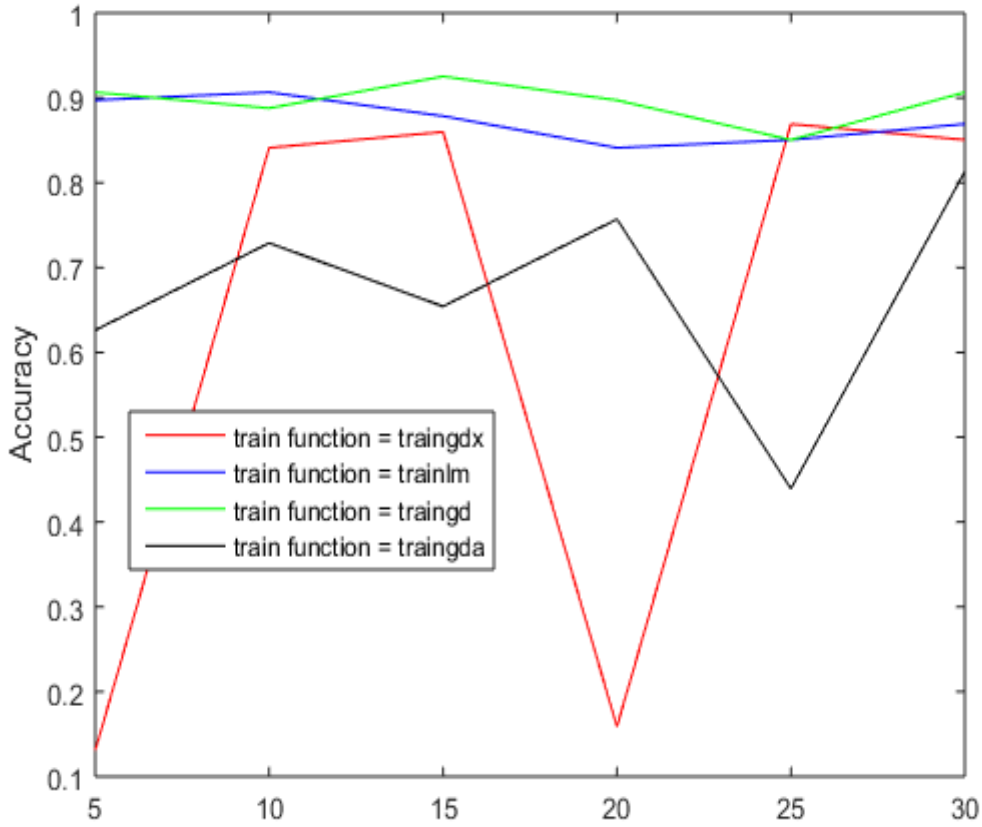
Εδώ να σημειώθεί ότι τα καλύτερα αποτελέσματα μας τα έδωσε για 1 επιπεδο η trainlm για 15 νευρώνες και για 2 και πάλι η ίδια για 25,5 νευρώνες αν και αυτό το αποτέλεσμα ποικίλλει λόγω της τυχαίοτητας κάθε φορά που τρέχους την randperm για να ανακατέψουμε τα δεδομένα μας.Εδώ να σημειωθεί ότι η αύξηση των νευρώνων σε κάθε επίπεδο δε συνάγεται την αύξηση της επίδοσης του δικτύου και υπάρχει όπως φαίνεται και στο σχήμα υπάρχει

ένα κατώφλι το οποίο όταν ξεπερνάμε η επίδοση συμπεριφέρεται αντιστρόφως ανάλογα του αριθμού των νευρώνων πράγμα λογικό αφού αυξάνοντας την πολυπλοκότητα και τις παραμέτρους του δικτύου υπάρχει κίνδυνος υπερεκπαίδευσης αφού η πολυπλοκότητα του δικτύου ξεπερνάει αυτή του προβλήματος κάνοντας το δίκτυο αργό και «ανεπιτυχές».Παρακάτω παρατίθενται διαγράμματα των εποχών για σύγκλιση κάθε συνάρτησης.



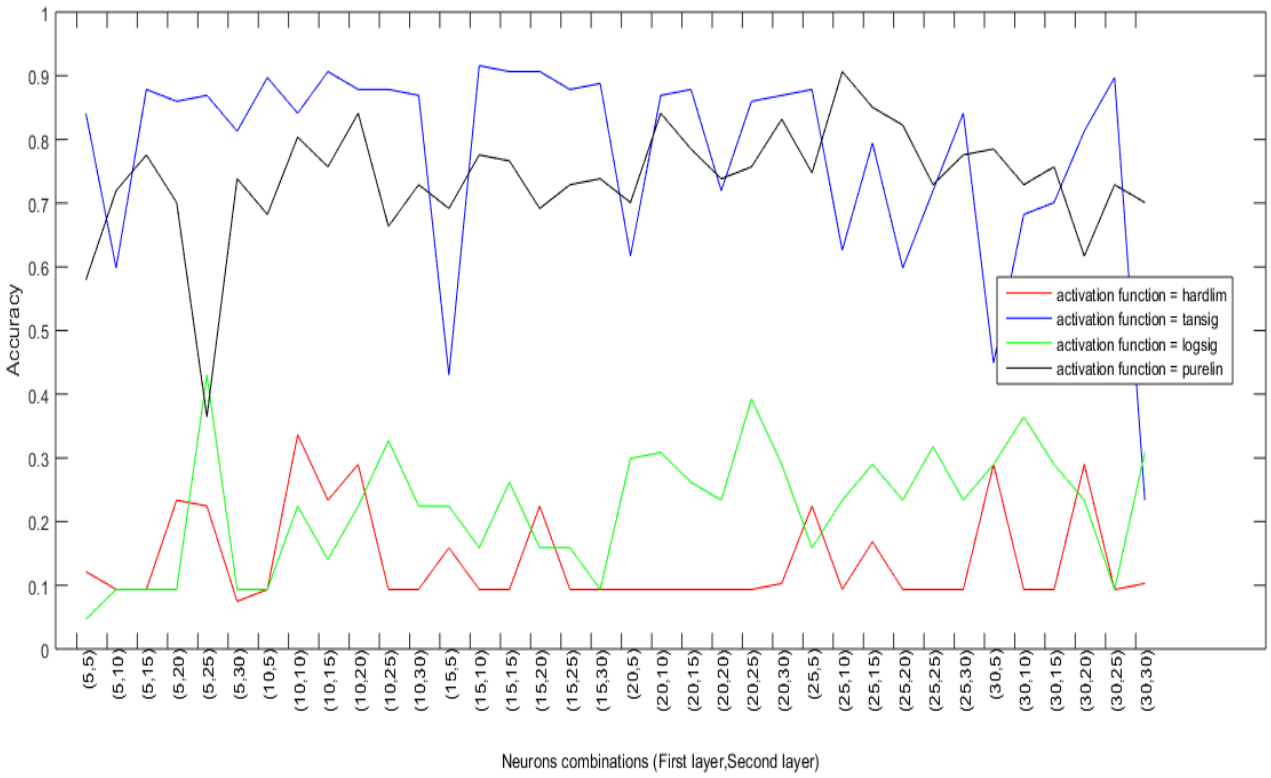


Παρατηρούμε την εξαρετικά αργή σύγκλιση της traingd όπως επισημάνθηκε σε σημείο να της παίρνει πάντα πάνω από 50000 εποχές μέχρι και 100000 και οι άλλες συναρτήσεις να χρειαστεί μεγένθυνση για να φανούν στο διάγραμμα αφού της σταματάει η μέθοδος Early Stopping.Παρ'όλα αυτά όπως φαίνεται παρακάτω βελτιώνεται σημαντικά η απόδοση της όταν αυξήσουμε αρκετά 100000 των αριθμό εποχών.



Ερώτημα 3

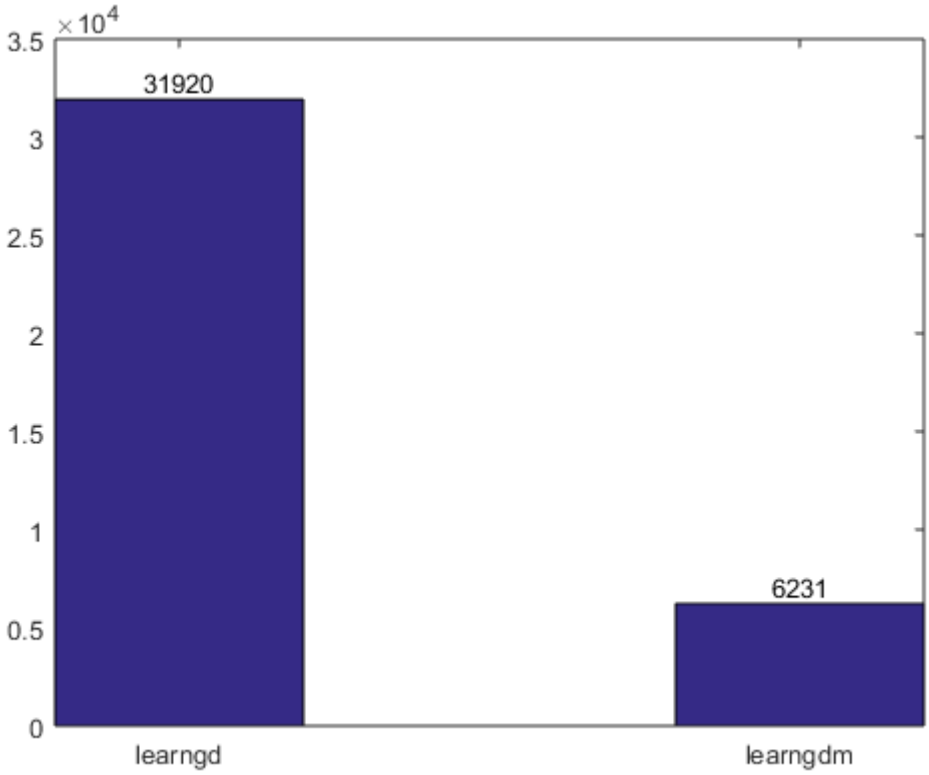
Επιλέγοντας την trainlm ως συνάρτηση παρατίθεται συγκριτικό διάγραμμα διαφόρων συναρτήσεων ενεργοποίησης για το στρώμα εξόδου για 2 επίπεδα κρυφών νευρώνων που μας έδωσαν και τα καλύτερα αποτελέσματα:



- Purelin(ταυτοτική) $f(x) = x$ είναι μονότονη και αυτή και η παράγωγος της δίνει τιμές στο (-∞,+∞).Είναι γραμμική γεγονός που όπως προκύπτει και από τα αποτλέσματα η προσθήκη κι άλλων κρυμμένων επιπέδων δε βελτιώνει ιδιαίτερα το αποτέλεσμα καθώς το δίκτυο μπορεί να μοντελοποιιηθεί μονοεπίπεδα.Το γεγονός της παραγωγισιμότητας της και αυτή και της παραγώγου της έχουν το θετικό του ότι για gradient descent μεθόδους σταδιακά βελτιώνονται και παρατηρούμε.Τέλος,η ικανότητα της να έχει «άπειρο» εύρος την κάνει πολύ αποδοτική στην εκπαίδευση καθώς η αναπαράσταση των προτύπων μεταβάλλει θετικά τα βάρη.Σε αυτή την περίπτωση είναι αναγκαίο η χρήση μικρού ρυθμού εκμάθησης.Γι αυτούς τους λόγους δίνει τα καλύτερα αποτελέσματα αλλά δεν έχουν μεγάλη διαφορά τα αποτελέσματα με τη χρήση 2 επιπέδων.Άρα γενικά προτιμάται για 1 επίπεδο νευρώνων(αν και 2 δεν την χειροτερεύουν) με οποιοαδήποτε από τις συναρτήσεις εκπαίδευσης λόγω της παραγωγισιμότητας της αλλά για αυτές που χρησιμοποιούν ρυθμό εκμάθησης δικό τους πρέπει να αρχικοποιηθεί χειροκίνητα σε μικρές τιμές.
- Hardlim είναι η βηματική συνάρτηση η οποία είναι μονότονη αλλά όχι και η παράγωγος της δίνει τιμές στο {0,1}.Το ότι δεν είναι παραγωγίσιμη στο 0 την κάνει να έχει κακή απόδοση σε μεθόδους απότομη καθόδου αφού αυτές κινούνται με βάση την αρνητικότητα της παραγώγου.Η χρήση 2 επιπέδων βελτιώνει την απόδοση της αν και αυτό δεν είναι αρκετό γιατί το περιορισμένο εύρος τιμών της δυσχεραίνει την εκπαίδευση της ιδιαίτερα σε μεθόδους που δεν χρησιμοποιούν αλγόριθμου απότομης καθόδου όπως η trainlm παραπάνω.Τέλος δεν προσεγγίζει την ταυτοτική κοντά στο 0 πράγμα που την κάνει ευαίσθητη σε τυχαίες αρχικοποιήσεις των βαρών χειροτερεύοντας πολύ την επίδοση της.Άρα γενικά δεν έχει καλή απόδοση αλλά στην καλύτερη περίπτωση θα την χρησιμοποιούσαμε με την μέθοδο trainlm η οποία δεν χρησιμοποιεί παράγωγο αλλά προσέγγιση Hessian για 2 ή περισσότερα επίπεδα κρυφών νευρώνων.
- Tansig $f(x) = \frac{2}{1+e^{-2x}} - 1$ είναι μονότονη αλλά όχι και η παράγωγος της δίνει τιμές στο (-1,1).Το κύριο συστατικό της επιτυχίας της είναι η παραγωγισιμότητα της και η συνέχεια της παραγώγου της που την κάνει επιτυχή σε όλες τις συναρτήσεις εκπαίδευσης απότομης καθόδου αλλά και στην trainlm.Έχει πεπαρασμένο εύρος τιμών άρα δεν είναι ευαίσθητη ως προς τον βαθμό εκμάθησης και έτσι θα έδινε καλά αποτελέσματα και στις περιπτώσεις των συναρτήσεων εκπαίδευσης με δικούς τους ρυθμούς εκμάθησης.Τέλος η μη γραμμικότητα της χρήζει ανάγκης πολλαπλών κρυφών επιπέδων για βελτίωση απόδοσης.Τέλος, προσσεγίζει την ταυτοτική στο 0 άρα είναι αναίσθητη σε αρχικοποιήσεις των βαρών.'Αρα ιδανικά μπορεί να αποδόσει για όλες τις συναρτήσεις εκπαίδευσης όμως για πολλαπλά επίπεδα νευρώνων.
- Logsig $f(x) = \frac{1}{1+e^{-x}}$ είναι μονότονη αλλά όχι η παράγωγος της δίνει τιμές στο (0,1).Είναι παραγωγίσιμη και συνεχής η παράγωγος της,και το εύρος της περιορισμένο κάνοντας την κατάλληλη για όλες τις μεθόδους εκπαίδευσης.Το γεγονός όμως της ευαισθησίας της σε αρχικοποιήσεις των βαρών λόγω της μη προσέγγισης της ταυτοτικής στο 0 την κάνει να είναι καλύτερη από την βηματική αλλά χειρότερη από τις υπόλοιπες.Ιδανικά ,λοιπόν, εφόσον είναι μη γραμμική οποιαδήποτε μέθοδος με καλές αρχικοποιήσεις των βαρών και πολλαπλά επίπεδα νευρώνων θα μπορούσε να μας δώσει καλά αποτελέσματα αλλά θα ήταν υπολογιστικά αρκετά πιο πολύπλοκη από τις υπόλοιπες.

Ερώτημα 4

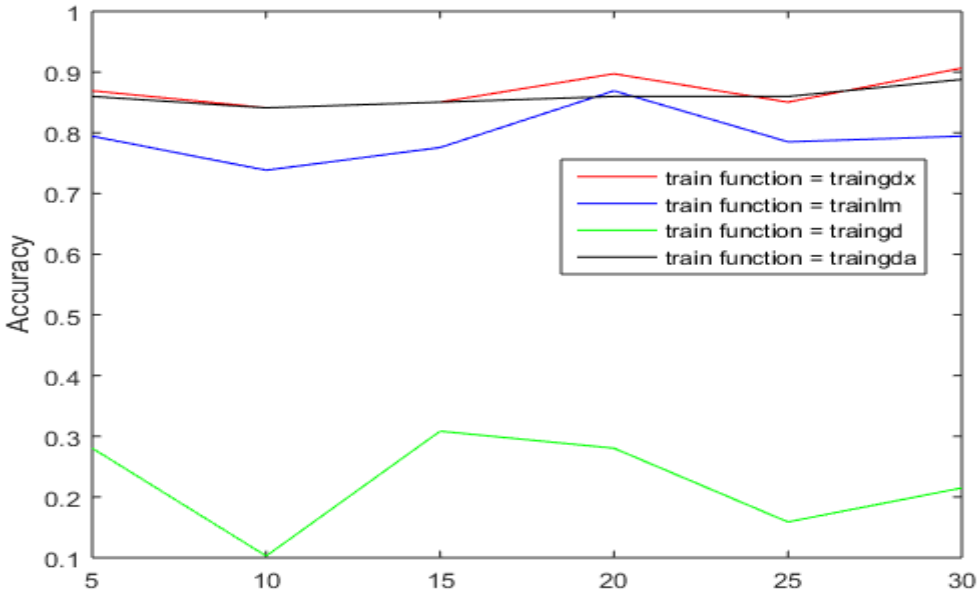
Είναι προφανές από τα παρακάτω αποτελέσματα ότι όπως αναμέναμε λόγω του όρου ορμής αλλά και του δικού του δείκτη εκμάθησης που διαθέτει η learngdm συγκλινει κάτι παραπάνω από φορές γρηγορότερα από την learngd η οποία λόγω απουσίας όρου ορμής αργεί να συγκλίνει και ταλαντώνεται σε τοπικά ελάχιστα.Παρατηρούμε όμως παραπλήσσιες επιδόσεις και στις δύο με τη learngdm να είναι καλύτερη αφού και συγκλίνει γρηγορότερα και ο δικός της δείκτης μάθησης της δίνει προβάδισμα σε αντίθεση με τη learngd η οποία συγκλίνει αργά αλλά και απαιτεί πειραματισμούς για την εύρεση του καταλληλότερου δείκτη εκμάθησης .Παρατίθενται τα αποτελέσματα εποχών και επίδοσης αντίστοιχα.



Οι επιδόσεις τους ήταν γύρω στο 86-88% για την learngd και 87-90% για διάφορες φορές που έτρεξα τον κώδικα.

Ερώτημα 5

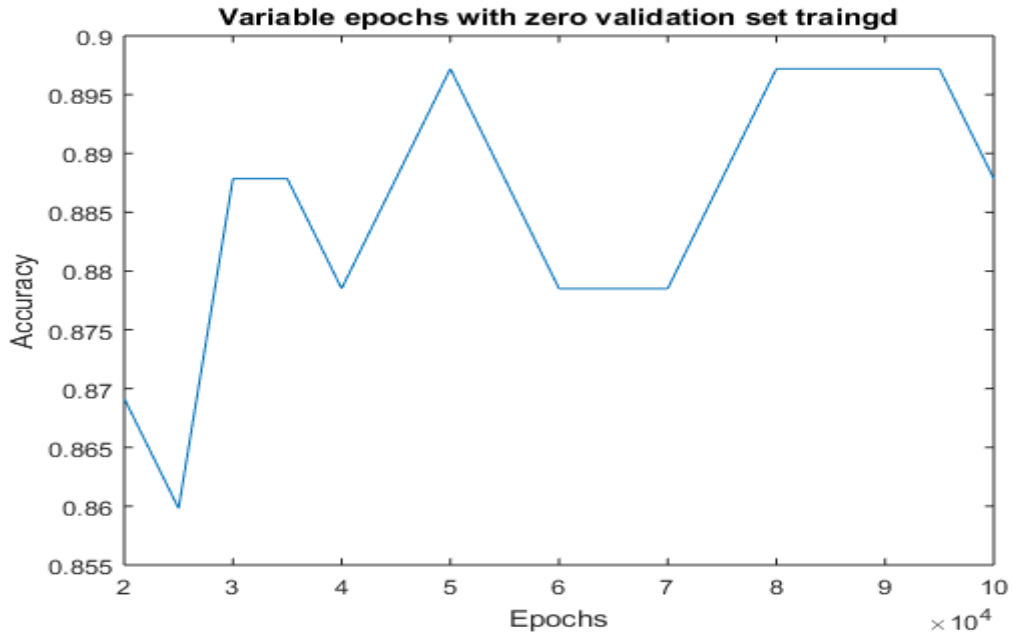
Παρακάτω παρατίθεται το αποτέλεσμα που λάβαμε χωρίς τη μέθοδο Early Stopping δηλαδή χωρίς validation set.

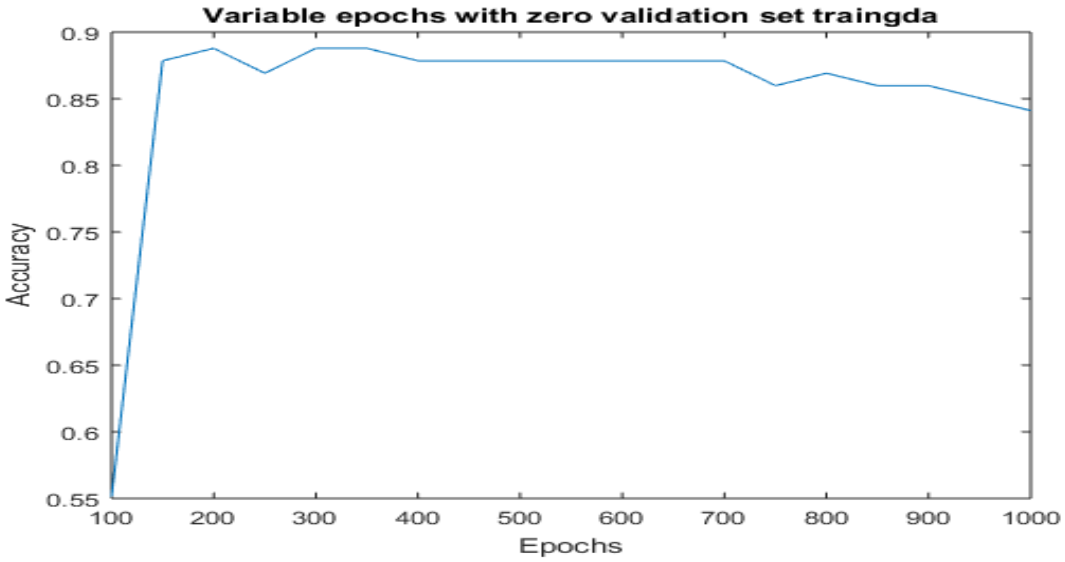


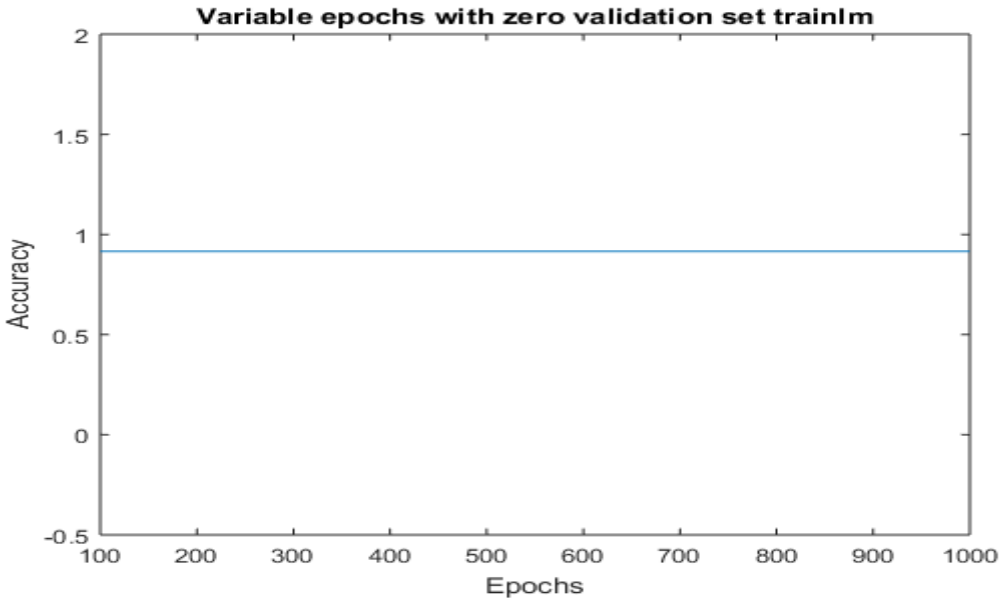
Η μέθοδος Early stopping αποφεύγει το φαινόμενο του overfitting αφού μειώνει ουσιαστικά τη διάσταση του προβλήματος μας.Όπως παρατηρούμε από το παραπάνω διάγραμμα εφού οι τρείς μέθοδοι trainngdx,trainlm,traingda χρειάζονται λίγες εποχές να συγκλίνουν δεν επηρεάζονται ιδιαίτερα από τη μέθοδο σε αντίθεση με την traingd η οποία χρεάζεται τεράστιο αριθμό εποχών για να συγκλίνει επομένως τα αποτελέσματα της είναι κάτω από μια τυχαία ταξινόμηση πολλές φορές η οποία έχει ~20% επιτυχία αφού έχουμε 5 κατηγορίες.Άρα η μέθοδος του Early Stopping αν ορίσουμε τις εποχές να είναι μεγάλο νούμερο αυτό που επιτυχάνει είναι μέσω

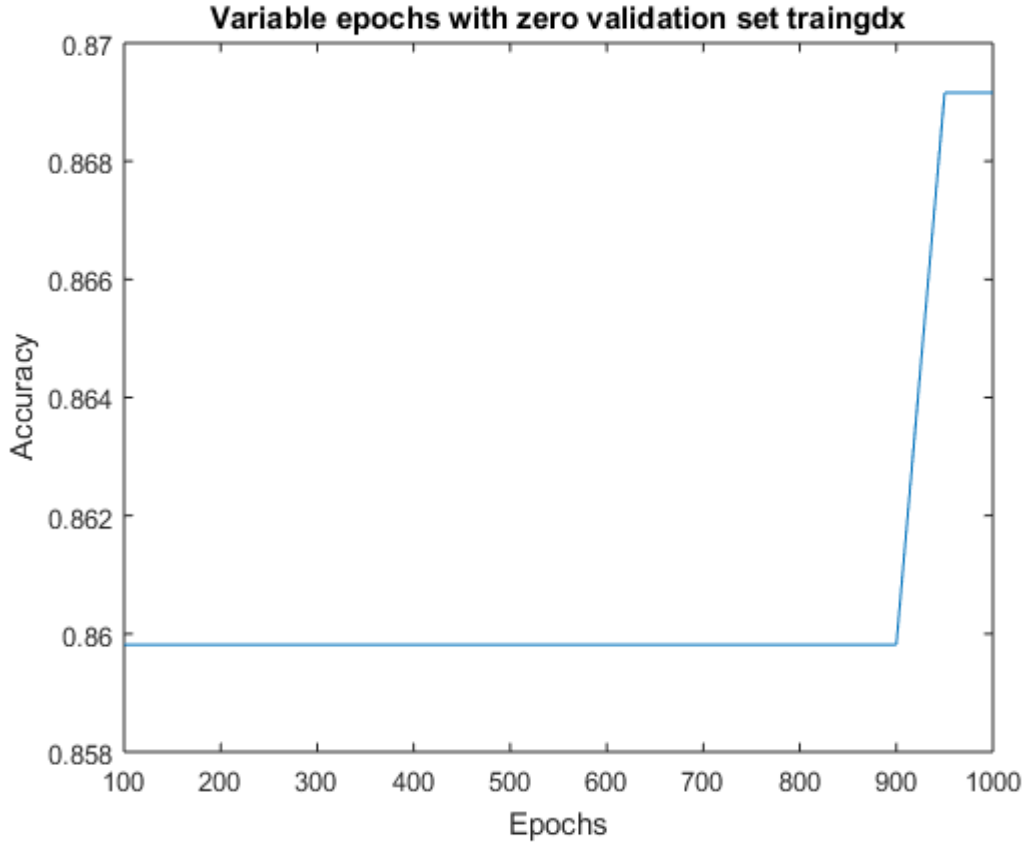
εύρεσης του λάθους να σταματάει την εκπαίδευση πριν το πέρας όλων των εποχών και να μειώνει έτσι την πιθανότητα υπερεκπαίδευσης.Επομένως αν μια συναάρτηση εκπαίδευσης αργεί πολύ να συγκλίνει και επηρεάζεται από τον αριθμό των εποχών είναι απαραίτητη η αποφυγή η υπεκπαίδευση και η χρήση του Early Stopping ώστε να εξασφαλίζει επιτυχία του μοντέλου και ασφάλεια των αποτελεσμάτων όσο νωρίτερα γίνεται.Από την άλλη,αν γνωρίζουμε τις εποχές που χρειάζεται για να συγκλίνει η συνάρτηση μας δεν είναι απαραίτητη η χρήση της αφού προσθέτει υπολογιστικό κόστος.Στη συνέχεια παρατίθονται τα αποτελέσματα κάθε συνάρτησης συναρτήσει του ορίου εποχών που τις έχουμε θέσει χωρίς validation set.

Ερώτημα 6



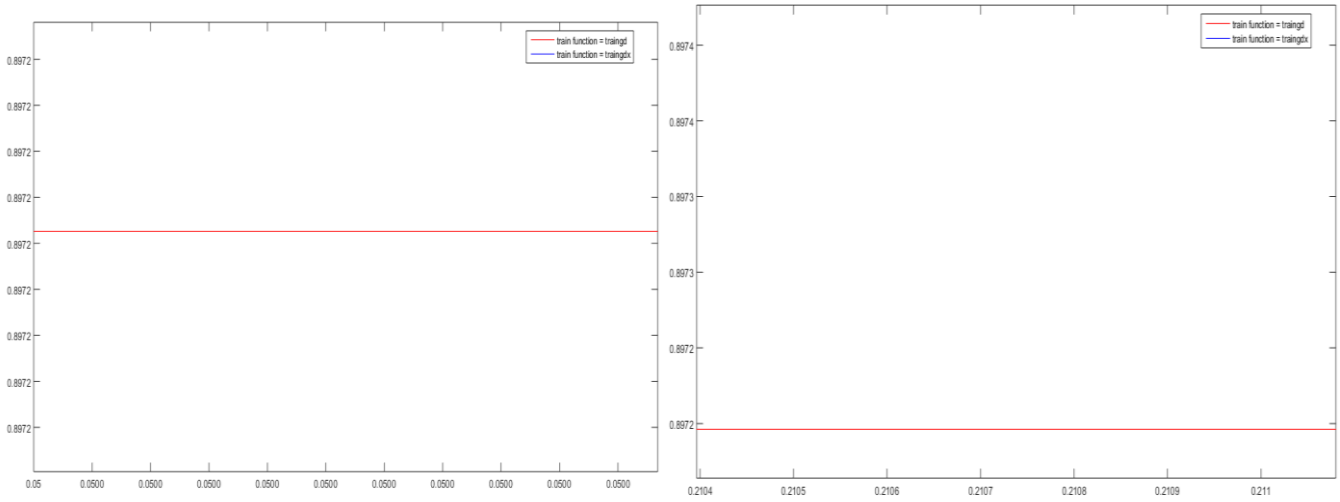


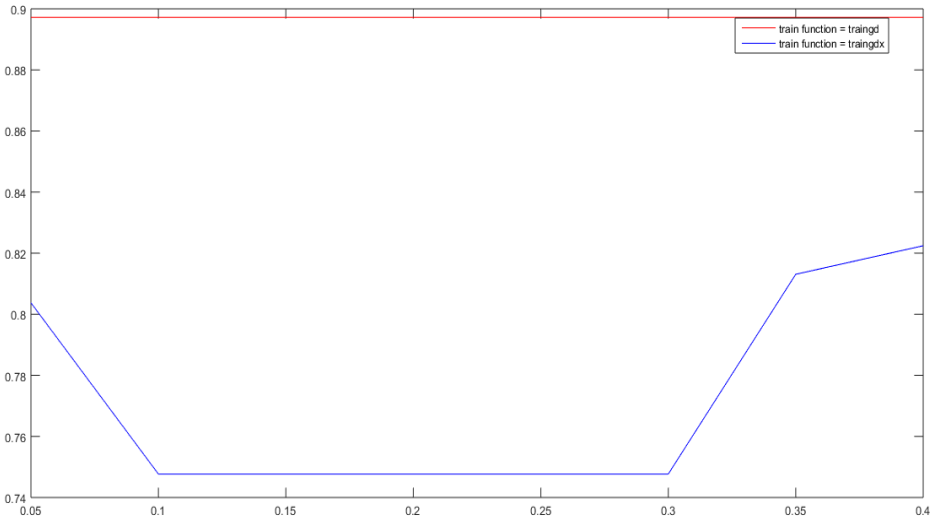


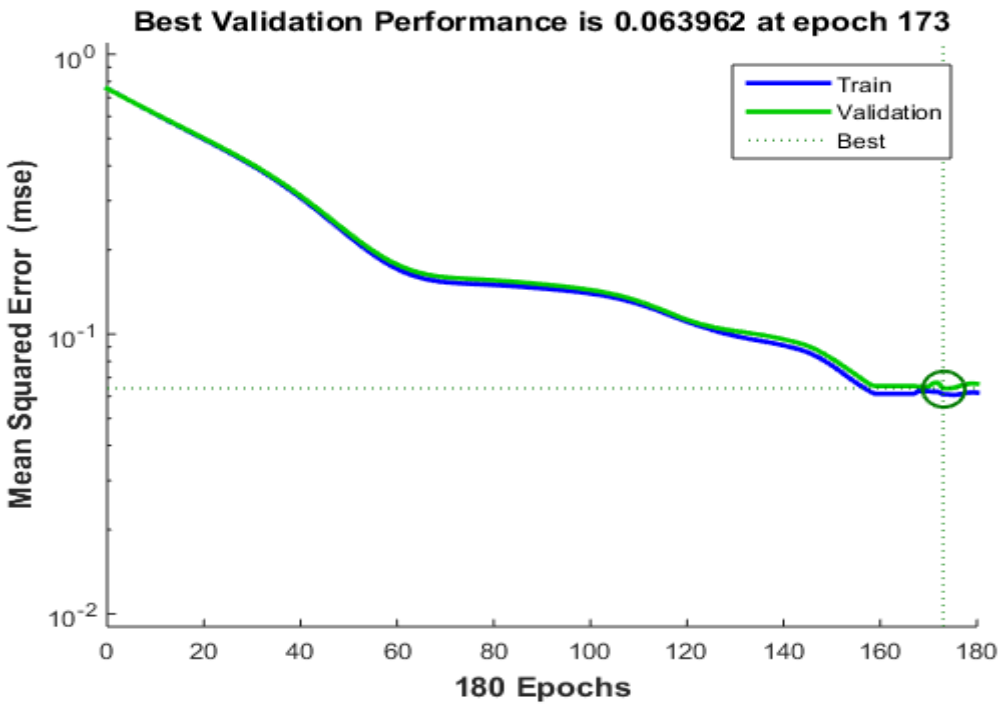


Παρατηρούμε όπως αναμέναμε αυξομειώσεις στην απόδοση της traingd η οποία κάποιες φορές προλαβαίνει να συγκλινει κάποιες όχι με όχι τόσο καλά αποτελέσματα όσο τις υπόλοιπες παρότι τα όρια που τις τέθηκαν είναι 100 φορές παραπάνω από τις υπόλοιπες συναρτήσεις(2000 μέχρι 100000 ενώ οι υπόλοιπες 100 μέχρι 1000).Για την traingda παρατηρούμε ότι συγκλίνει με καλά αποτελέσματα πό 200 εποχές και πάνω η traingdx από την αρχή με καλύτερα αποτελέσματα για πάνω από 900 εποχές.Τέλος, η trainlm μας δίνει σταθερά καλό αποτέλεσμα συγκλίνοντας σχεδόν αμέσως κάθε όριο εποχών.

Ερώτημα 7


train function = traingd
train function = traingdx


train function = traingd
train function = traingdx


Best Validation Performance is 0.063962 at epoch 173
Train
Validation
Best
Mean Squared Error (mse)
180 Epochs

Παρατηρούμε ότι η traingd δεν βελτιώνει την απόδοση αντίθετα ελαττώνεται όπως φαίνεται στα πάνω 2 μεγενθυμένα διαγράμματα αυτό οφείλεται στην αδυναμία της να συγκλίνει και στο ότι αναγκάζεται αν σταματήσει εξαιτίας validation check αφού ενώ φτάνει κοντά στο ελάχιστο συμπεριφέρεται ταλαντωτικά γύρω του.

Όσον αφορά την traingdx παρατηρούμε για μικρόρυθμό μάθησης ανεπιτυχή αποτελέσματα για το δίκτυο μας αφού το δίκτυο τερματίζει πρόωρα λόγω validation αφού αλλάζουν ελάχιστα τα βάρη και έτσι η παράγωγος του σφάλματος(επίδοση δικτύου/performance) και η παράγωγος μένει σχεδόν σταθερή αφήνοντας τον όρο ορμής να χειροτερεύει αρκετά την επίδοση αυξάνοντας τα βάρη.Αυξάνοντας το ρυθμό μάθησης παρατηρούμε ότι η αύξηση των βαρών οδηγεί το δίκτυο στο να χάσει το επιδιωκόμενο ελάχιστο και να τερματίζεται και πάλι λόγω validation check όπως φαίνεται και στο τελευταίο διάγραμμα.

Ερώτημα 8

Όσον αφορά τις επιμέρους κατηγορίες επιλέξαμε να αξιολογήσουμε το δίκτυο με βάση την παρακάτω μετρική με τη βοήθεια της https://en.wikipedia.org/wiki/Precision_and_recall :

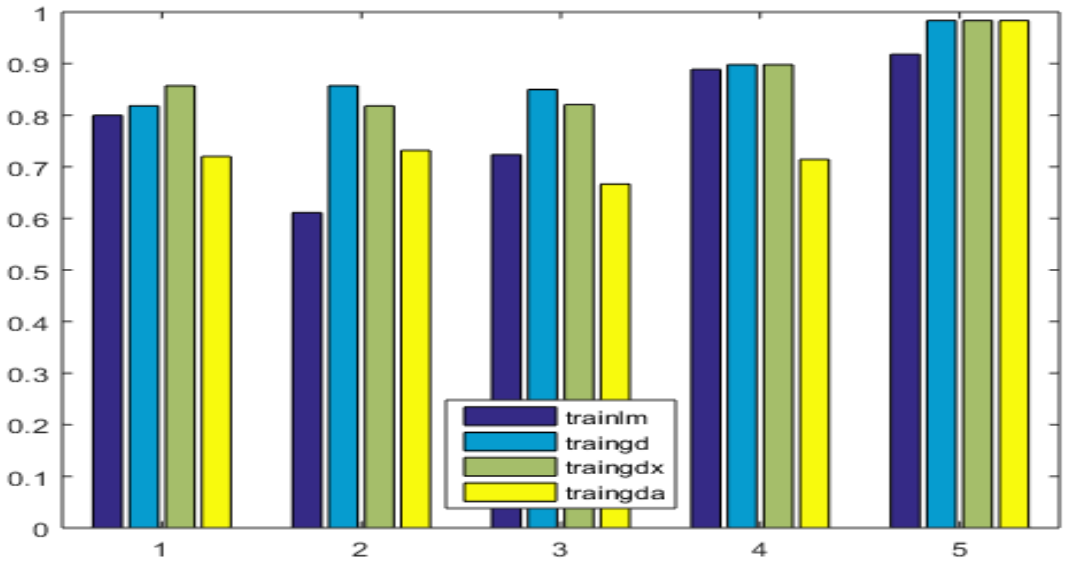
$$\frac{2 * precision * recall}{precision + recal}$$

$$= 2\frac{\frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία}{Ταξινομημένα\ τμήματα\ στην\ κατηγορία} * \frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία}{Σύνολο\ τμημάτων\ στην\ κατηγορία}}{\frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία}{Ταξινομημένα\ τμήματα\ στην\ κατηγορία} + \frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία}{Σύνολο\ τμημάτων\ στην\ κατηγορία}}$$

$$= 2\frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία^2}{Σωστ.ταξιν.κατ.* Σύν.τμημ.κατηγορία + Σωσ.ταξιν.κατ.* Ταξινομημένα\ τμήματα\ \ κατ.} =$$

$$2\frac{Σωστά\ ταξινομημένα\ στην\ κατηγορία}{Σύνολο\ τμημάτων\ στην\ κατηγορία + Ταξινομημένα\ τμήματα\ στην\ κατηγορία}$$

Που αποτελεί τον αρμονικό μέσο όρο ουσιαστικά των precision και recall και μας δίνει σαφή εικόνα όσων αφορά την επιτυχία κάθε συνάρτησης εκπαίδευσης σε κάθε κατηγορία για ένα συνδυασμό νευρώνων που απέδοσαν τα δέοντα όλες τους.Θα μπορούσαμε ενναλακτικά να χρησιμοποιήσουμε και τη μέθοδο του confusion matrix.



Παρατηρούμε ότι ορισμένες κατηγορίες και ιδιαίτερα οι 4,5 παρουσιάζουν σαφώς καλύτερα αποτελέσματα από τις υπόλοιπες πράγμα που οφείλεται στο ότι είναι ευκολότερα διαχωρίσιμες από τις υπόλοιπες κατηγορίες για όλες τις συναρτήσεις.Γι'αυτό το λόγο θα μπορούσαμε να πετύχουμε καλύτερα αποτελέσματα με μηχανές διανυσμάτων υποστήριξης(SVM) ή ακόμα ίσως και KNN (k-nearest neighbour)μέθοδος η οποία δεν στηρίζεται στις ιδιομορφίες των δεδομένων αλλά είναι πιο γενική.

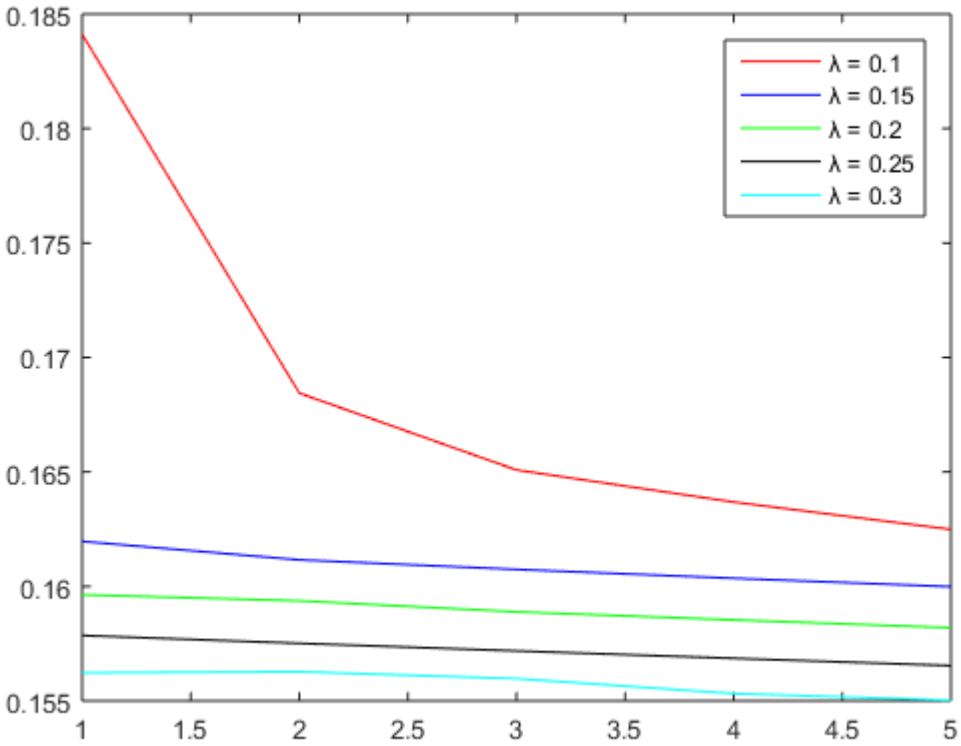
Η μέθοδος της αποσύνθεσης των βαρών

Στο τελευταίο ερώτημα καλούμαστε να εντάξουμε στον αλγόριθμο μας τη διαδικασία αποσύνθεσης βαρών και κλαδέματος την οποία υλοποιούμε με το ανανεώνουμε τα βάρη με βάση την παρακάτω εξίσωση

$$w_{ij}(k) - w_{ij}(k-1) = -\mu \frac{\partial E}{\partial w_{ij}(k-1)} - \lambda w_{ij}(k-1) \quad \textbf{(1)}$$

Γνωρίζουμε ότι ισχύει η σχέση $\lambda \sim \frac{1}{ct*(\#iterations)*learning\_rate}$

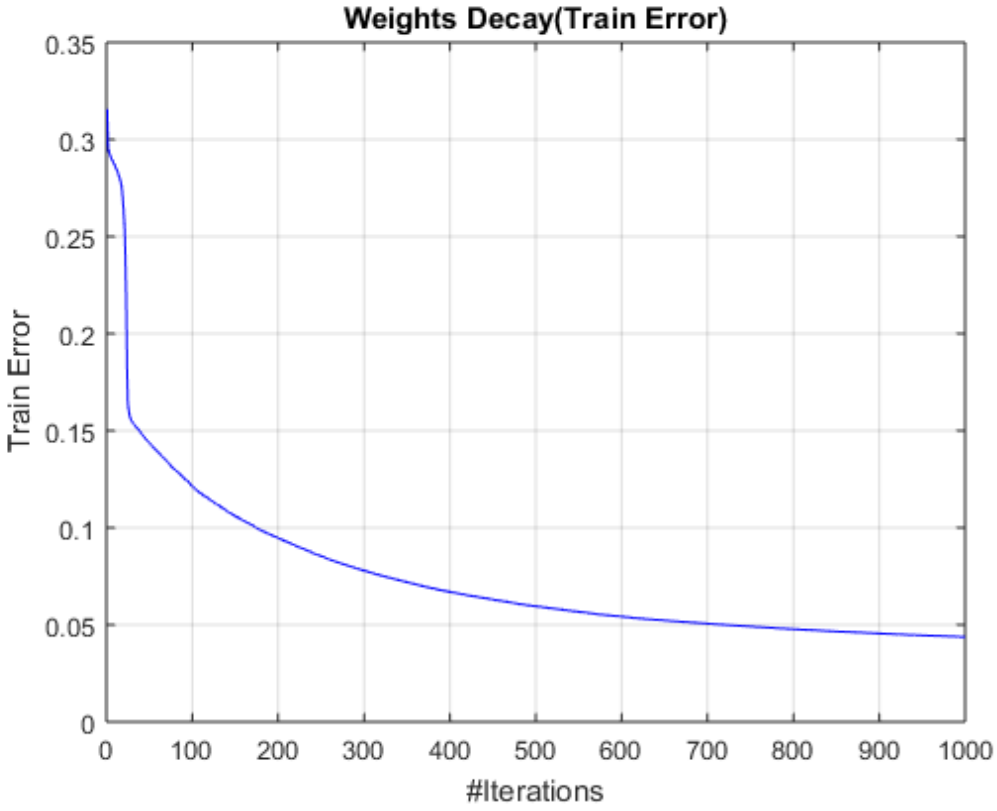
Σύμφωνα με την τελευταία σχέση πειραματίστηκα με την παράμετρο λ για την εύρεση αυτού που μας δίνει το μικρότερο λάθος για το προκαθορισμένο learning rate 5 επαναλήψεις είχαμε τα εξής:

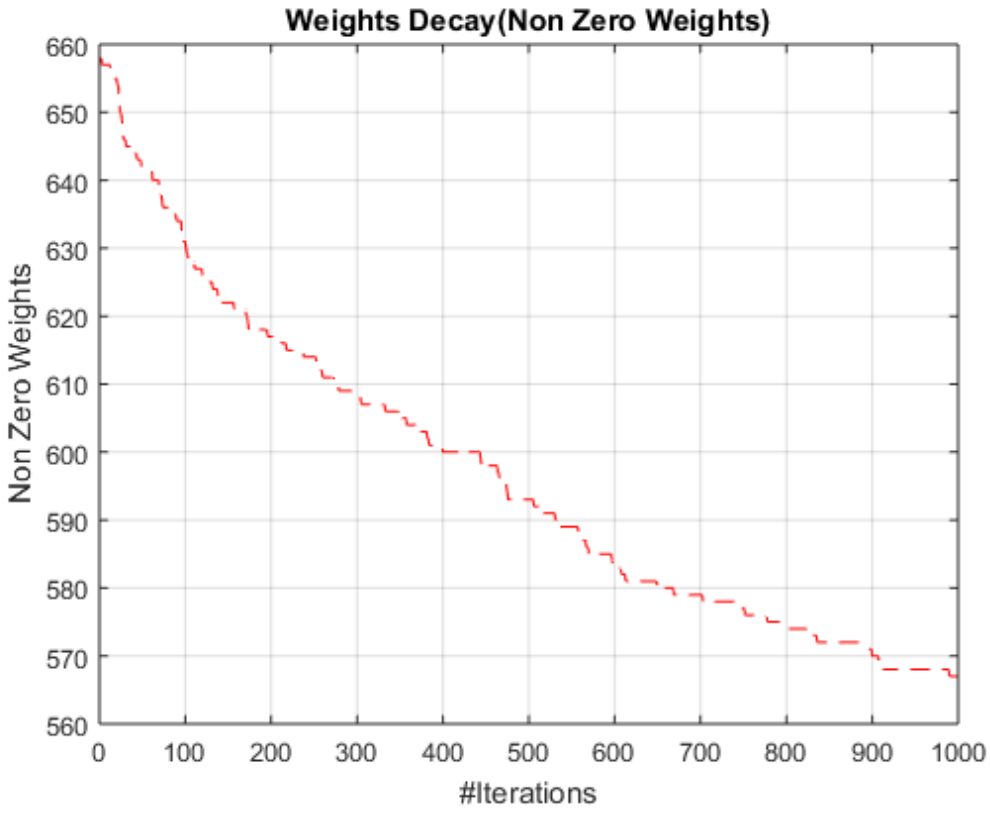


Τα παραπάνω επιβεβαιώνουν τη σχέση μεταξύ λ και ρυθμού εκμάθησης και επαναλήψεωνΣτη συνέχεια δημιουργήσαμε το δίκτυο ,αρχικοποιήσαμε τα βάρη,τις εποχές και τις κατάλληλες παραμέτρους εκτός της επανάληψης και έπειτα εργαστήκαμε σύμφωνα με την **(1)** ως εξής :

```
%λήψη παλιών βαρών
old_w = getwb(net);
%εκπαίδευση του δικτύου για μια εποχή όπως έχει οριστεί %εκτός for
[net,tr] = train(net,TrainData,TrainDataTargets);
% μείωση των βαρών κατά λ
weights=getwb(net);
new_w_changed = getwb(net) - l*old_w;
%αποθήκευση λάθους
performance = [performance tr.perf(2)];
%εύρεση και μηδενισμός βαρών κάτω από d
idxs = find(abs(new_w_changed) < d);
weights(idxs) = 0;
%αποθήκευση μη μηδενικών βαρών
n_zero_weights=[non_zero_weights (length(new_w_changed)-length(idxs))];
%ανανέωση βαρών του δικτύου
net = setwb(net,weights);
```

Στη συνέχεια παρατίθενται τα ζητούμενα αποτελέσματα για το δίκτυο για κατώφλι 0.1 και λ 0.001:





Είναι λογικό στην αρχή να έχουμε περισσότερα μηδενικά βάρη από ότι στη συνέχεια αφού σιγά σιγά το δίκτυο καταλήγει στο ζητούμενο αποτέλεσμα μειώνοντας σταδιακά το σφάλμα και συγκλίνοντας σε όλο και καλύτερο αποτελέσμα.

Παρατηρούμε ότι έχουμε εξαιρετικά αποτελέσματα με απόδοση να φτάνει και το 95% και παραπάνω.Το δίκτυο συγκλίνει πολύ γρήγορα προς καλά αποτελέσματα αφού πέφτει σχεδόν εκθετικά το σφάλμα εκπαίδευσης.Παρατηρούμε όμως ότι για πάνω από 600 επαναλήψεις η βελτίωση επιβραδύνεται.Η μέθοδος του Pruning έδωσε τα καλύτερα αποτελέσματα από όλους τους άλλους συνδυασμούς σχεδόν και συγκλίνει ταχύτατα Μάλιστα, αν πειραματιστούμε περισσότερο μπορούμε να βρούμε καλύτερες παραμέτρους κατώφλι και λ –αλλά και τον αριθμό των επαναλήψεων-για τις οποίες θα κλαδεύουμε από το δίκτυο όλο και περισσότερους άχρηστους κόμβους οδηγούμενοι σε ασφαλή σύγκλιση με πάρα πολύ θετικά αποτελέσματα.