Κωνσταντίνος Ανδρέου 4316
Ιωάννης Μάνος 4416
Μιλτιάδης Παπαθεοδωρόπουλος 4463

Άσκηση 1 MLP:

Το πρότζεκτ υλοποιήθηκε σε γλώσσα προγραμματισμού Java και αποτελείται από 8 διαφορετικές κλάσεις. Μία κλάση Coordinate που αναπαριστά μία συντεταγμένη, μία κλάση ClassificationDataset που δημιουργεί τα training και test Sets, μία κλάση Neuron που αναπαριστά έναν νευρώνα, μια κλάση Layer για την αναπαράσταση ενός επιπέδου του δικτύου, μία κλάση ActivationFunction που αναπαριστά την συνάρτηση ενεργοποίησης, τις κλάσεις MyFrame και MyPanel για την γραφική αποτύπωση των παραδειγμάτων και την βασική κλάση MLP που έχει μέσα την main() και υλοποιεί την βασική λειτουργία του MLP.

Η κλάση Coordinate.java αποτελείται από έναν constructor οπού με βάση τις παραμέτρους x1 και x2 κατατάσσει την συντεταγμένη στην κατηγορία της. Επίσης, περιέχει την μέθοδο encodeCategory οπού κωδικοποιεί μια κατηγορία σε ένα διάνυσμα(για την C1:[1, 0, 0] για την C2:[0, 1, 0] και για την C3:[0, 0 , 1]) και την decodeCategory όπου αποκωδικοποιεί το διάνυσμα στην αντίστοιχη κατηγορία με βάση ποιος όρος του διανύσματος έχει την μεγαλύτερη τιμή(πιο κοντά στο 1).

Η κλάση ClassificationDataset.java χρησιμοποιείται προκειμένου να δημιουργηθούν οι 8000 τυχαίες συντεταγμένες και γράφει μισές από αυτές σε ένα αρχείο LearningSet και τις άλλες μισές σε ένα αρχειο TestSet. Επισης, η κλάση αυτή περιέχει μια main() οπού αν εκτελεστεί δημιουργεί καινούρια αρχεία μάθησης και ελέγχου.

Η κλάση Neuron.java αναπαριστά έναν νευρώνα και αποτελείται από πεδία όπως η τιμή(έξοδος) του, τα βάρη, η πόλωση του, το δέλτα(σφάλμα), και περιέχει και έναν πίνακα που κρατά τις μερικές παράγωγους του σφάλματος προς τα βάρη. Με την κλήση του constructor για την δημιουργία ενός νευρώνα τα βάρη και η πόλωση του νευρώνα λαμβάνουν τυχαίες τιμές στο διάστημα (0,1).

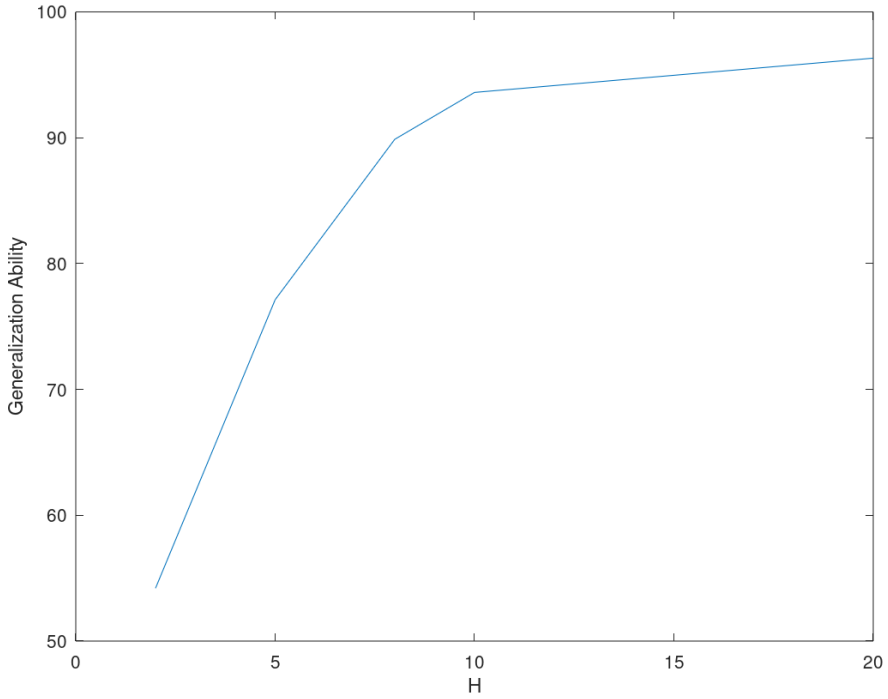
Η κλάση Layer.java αναπαριστά ένα επίπεδο του νευρωνικού δικτύου και περιέχει ένα πίνακα Neurons με τους νευρώνες του επιπέδου.

Η κλάση ActivationFunction.java περιέχει ένα πεδίο type που περνιέται ως παράμετρος στον constructor και με βάση την τιμή που θα έχει αυτό το πεδίο επιλέγεται η αντίστοιχη συνάρτηση ενεργοποίησης. Για type=0 έχουμε σιγμοειδή για type=1 υπερβολική εφαπτομένη ενώ για type=2 relu.

Η κλάση MLP.java έχει στην αρχή αρχικοποιημένη τη μεταβλητή d ίση με 2 καθώς θέλουμε να υπάρχουν δυο νευρώνες στο επίπεδο εισόδου όπου ο ένας θα λαμβάνει την x1 συντεταγμένη και ο άλλος την x2 και επίσης υπάρχει και η μεταβλητή K=3 που δηλώνει τους τρεις νευρώνες στο επίπεδο εξόδου λόγω των τριών διαφορετικών κατηγοριών. Επίσης, στην αρχή είναι αρχικοποιημένες μεταβλητές που καθορίζουν την αρχιτεκτονική του δικτύου όπως το πλήθος των νευρώνων στα επίπεδα, την συνάρτηση ενεργοποίησης στα κρυμμένα επίπεδα μέσω της μεταβλητής FUNCTION,

την ouputFunction για σιγμοειδή συνάρτηση στο επίπεδο εξόδου και το μέγεθος του batch(B) για την ενημέρωση των βαρών. Στον constructor του MLP αρχικοποιούνται ζητώντας input από τον χρήστη ο ρυθμός μάθησης, το κατώφλι τερματισμού και αρχικοποιούνται οι νευρώνες σε κάθε επίπεδο με αρχικά τυχαία βάρη. Στην κλάση MLP είναι υλοποιημένη η μέθοδος forward_pass , η οποία υπολογίζει τις εξόδους των νευρώνων σε κάθε επίπεδο και επιστρέφει το διάνυσμα εξόδου y (διάστασης K) του MLP δοθέντος του διανύσματος εισόδου x (διάστασης d). Η μέθοδος backpropagation παίρνει ως παράμετρο ένα παράδειγμα και την αναμενόμενη έξοδο του(t) και υπολογίζει τις μερικές παράγωγους του σφάλματος ως προς τα βάρη και ενημερώνει τους πίνακες errorGradients κάθε νευρώνα προσθέτοντας τις νέες τιμές. Επίσης, υπολογίζει και επιστρέφει το τετραγωνικό σφάλμα για αυτό το παράδειγμα. Η μέθοδος updateWeights εκτελείται για κάθε batch και υπολογίζει τα νέα βάρη κάθε νευρώνα χρησιμοποιώντας τον αλγόριθμο gradient descent , ενώ στο τέλος της εκτελείται η μέθοδος resetErrorGradients για να μηδενιστούν οι μερικές παράγωγοι του σφάλματος προς τα βάρη/πόλωση. Ο βασικός αλγόριθμος που υλοποιείται στην main έχει ως εξής: Σε κάθε εποχή, για κάθε παράδειγμα καλείται η συνάρτηση backpropagation και για το τετραγωνικό σφάλμα του παραδείγματος που υπολογίζει το προσθέτει σε μια μεταβλητή errorSum. Ανάλογα με την τιμή του B(batch) ενημερώνονται τα βάρη ανά B παραδείγματα. Μετά το τέλος μίας εποχής υπολογίζεται και τυπώνεται το σφάλμα μάθησης το οποίο ισούται με errorSum/πληθος παραδειγμάτων μάθησης, δηλαδη 4000. Μετά την εποχή 700 και για κάθε επόμενη εποχή καλείται η μέθοδος checkTermination , η οποία υπολογίζει την διαφορά μεταξύ του σφάλματος της προηγούμενης εποχής και της τωρινής εποχής και αν αυτή η διαφορά είναι μικρότερη από το κατώφλι τερματισμού που είχε θέσει ο χρήστης η μάθηση τελειώνει και αποθηκεύονται μέσω της saveWeights() τα βάρη που βρέθηκαν στο αρχείο Weights.txt. Έτσι, αφού τελειώσει η μάθηση εκτελείται η μέθοδος computeGeneralizationAbility , η οποία περνά μέσα από το δίκτυο κάθε παράδειγμα του test set και υπολογίζει την γενικευτική ικανότητα του δικτύου και ενημερώνει τον πίνακα printAtPanel , ο οποίος χρησιμοποιείται για την τύπωση των παραδειγμάτων. Τέλος, καλείτε ο MyFrame constructor και τυπώνονται τα παραδείγματα του συνόλου ελέγχου με τρία διαφορετικά χρώματα ανάλογα την κατηγορία στην οποία βρίσκονται και με + αν ταξινομήθηκαν σωστά και – αν όχι.

Αποτελέσματα: α)



Στην παραπάνω γραφική παράσταση βλέπουμε πως μεταβάλλεται η γενικευτική ικανότητα του MLP για διάφορες τιμές νευρώνων στα κρυμμένα επίπεδα. Όσο αυξάνεται ο αριθμός των νευρώνων(άξονας χ) σε κάθε κρυμμένο επίπεδο τόσο αυξάνεται και η γενικευτική ικανότητα.(Χρησιμοποιήθηκε ίδιος αριθμός νευρώνων σε κάθε επίπεδο, B=1, ρυθμός μάθησης=0.1 και σιγμοειδής/λογιστική συνάρτηση ενεργ.)
Νευρώνες = [2, 5, 8, 10, 20]
Γενικευτική ικανότητα = [54.2, 77.125, 89.875, 93.6, 96.325]

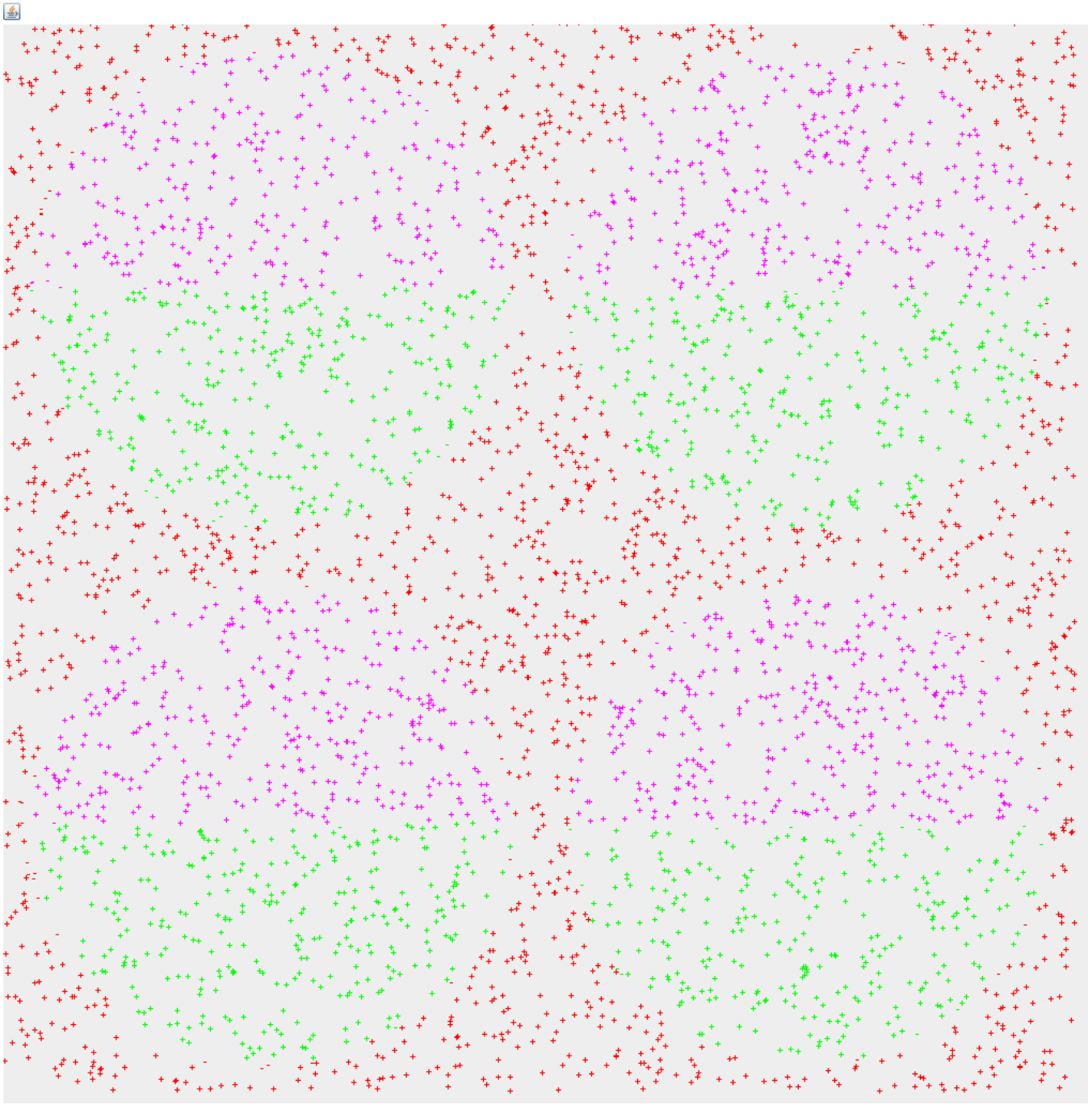
Β) Για σιγμοειδή/λογιστική συνάρτηση ενεργοποίησης στους κρυμμένους νευρώνες η γενικευτική ικανότητα του δικτύου ήταν 89.4% , για υπερβολική εφαπτομένη(tanh) 95.5%, ενώ για relu 93.55% . Βλέπουμε ότι την καλύτερη γενικευτική ικανότητα παρουσίασε η tanh έχοντας μικρή διαφορά με την relu. Χρησιμοποιήθηκαν 8 κρυμμένοι νευρώνες σε κάθε επίπεδο, B=1 και ρυθμός μάθησης 0.01(με ρυθμό μάθησης 0.1 σύγκλινε νωρίς σε ένα ελάχιστο με μικρή γενικευτική ικανότητα).

Γ)

	B=40	B=400
Λογιστική:	85%	73.075%
TANH :	85.05%	78.575%
RELU :	79%	74%

Βλέπουμε ότι με την αύξηση του minibatch size(B) υπήρχε μια μείωση της γενικευτικής ικανότητας του δικτύου και για τις τρεις συναρτήσεις ενεργοποίησης χρησιμοποιώντας 5 νευρώνες σε κάθε κρυμμένο και 0.01 ρυθμό μάθησης.

Παρακάτω βλέπουμε ενδεικτικά για το δίκτυο με την καλύτερη γενικευτική ικανότητα(97.4%) που βρήκαμε χρησιμοποιώντας 10 κρυμμένους νευρώνες σε κάθε επίπεδο, με B=40 ,tanh συνάρτηση ενεργοποίησης και ρυθμό μάθησης 0.01 , τα παραδείγματα για κάθε κατηγορία καθώς και τις επιτυχημένες κατατάξεις του δικτύου με + και τις αποτυχημένες με - , οι οποίες βρίσκονται κοντά στα όρια μεταξύ των διαφορετικών περιοχών



Εντολές μεταγλώττισης:

javac Coordinate.java MyFrame.java Neuron.java ActivationFunction.java Layer.java MyPanel.java ClassificationDataset.java MLP.java
Εκτέλεση:
java ClassificationDataset (αν θέλουμε καινούρια dataset)
java MLP