ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ

ΑΕΙ ΠΕΙΡΑΙΑ ΤΤ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Νευρωνικά δίκτυα και εφαρμογή τους στην αναγνώριση χαρακτήρων

Γεώργιος Μαλανδράκης

Α.Μ.: 43856

Εισηγητής: Νικόλαος Ζάχαρης

ΑΠΡΙΛΙΟΣ 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Νευρωνικά δίκτυα και εφαρμογή τους στην αναγνώριση χαρακτήρων

Μαλανδράκης Γεώργιος

43856

Εισηγητής:

Δρ. Νικόλαος Ζάχαρης

Εξεταστική επιτροπή:

Ημερομηνία εξέτασης:

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Μαλανδράκης Γεώργιος, του Βασιλείου, με αριθμό μητρώου 43856 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω: «Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο. Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης. Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω ιδιαιτέρως τους γονείς μου, μεταξύ άλλων γιατί με ενεθάρρυναν να ασχοληθώ με προηγμένους τομείς της πληροφορικής. Θα ήθελα επίσης να ευχαριστήσω τον επιβλέποντα καθηγητή, Νικόλαο Ζάχαρη, για τη βοήθεια και τις συμβουλές του κατά την εκπόνηση της παρούσης πτυχιακής.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή καταπιάνεται με τα τεχνητά νευρωνικά δίκτυα. Αφότου παρουσιαστούν οι θεμελιώδεις τους έννοιες, οι αρχιτεκτονικές τους, καθώς και οι πιο γνωστές μέθοδοι εκπαίδευσής τους, υλοποιείται σε matlab εφαρμογή δημιουργίας και εκπαίδευσης νευρωνικών δικτύων για ανάγνωση κειμένων από εικόνες. Κατόπιν, γίνονται συγκρίσεις σε επιδόσεις νευρωνικών δικτύων ποικίλλων προδιαγραφών στην αναγνώριση των χαρακτήρων και εκφράζονται συμπεράσματα.

Λέξεις κλειδιά: Τεχνητά νευρωνικά δίκτυα, αναγνώριση χαρακτήρων, matlab, Μέθοδοι εκπαίδευσης νευρωνικών δικτύων, ανάγνωση κειμένου

ABSTRACT

This thesis is about artificial neural networks. After presenting their fundamental principles, their architectures, and the most widely used training methods suitable for them, we will present a matlab-based application for creating and training neural networks for the purpose of recognizing text from images. Experimentation and comparisons of the performance of neural networks of various specifications when it comes to character recognition will be shown thereafter.

Keywords: Artificial neural networks, character recognition, matlab, neural network training methods, text recognition

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΤΥΠΩΝ

1. ΕΙΣΑΓΩΓΗ

1.1. Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Τα τελευταία χρόνια η πληροφορική καλείται να επιλύσει προβλήματα που δεν λύνονται με τη χρήση συμβατικών αλγορίθμων. Στα πλαίσια της επίλυσής τους χρησιμοποιούνται κατά κόρον και τα τεχνητά νευρωνικά δίκτυα, με τα οποία καταπιάνεται η παρούσα πτυχιακή για την επίλυση ενός εκ των χαρακτηριστικότερων προβλημάτων στα οποία χρησιμοποιούνται -την αναγνώριση χαρακτήρων.

Η πτυχιακή ξεκινάει με ένα θεωρητικό κομμάτι στο οποίο περιγράφονται οι θεμελιώδεις έννοιες των νευρωνικών δικτύων, οι κύριες αρχιτεκτονικές τους, καθώς και οι κυριότερες μέθοδοι εκπαίδευσής τους. Έπειτα, αφότου γίνει μια περιγραφή των δυνατοτήτων που μας παρέχει το matlab για τα νευρωνικά δίκτυα και την αναγνώριση χαρακτήρων, θα υλοποιηθεί και θα παρουσιαστεί μια εφαρμογή που θα επιτρέπει στο χρήστη να καθορίσει τις τεχνικές παραμέτρους του νευρωνικού δικτύου προς δημιουργία, και να το εκπαιδεύσει για αναγνώριση χαρακτήρων από εικόνες. Μεταξύ άλλων, θα παρέχονται δυνατότητες παρακολούθησης των ενδιάμεσων σταδίων επεξεργασίας της εικόνας μέχρι να αναγνωριστούν οι χαρακτήρες από τα νευρωνικά δίκτυα, καθώς και η δυνατότητα τροποποίησης και αποθήκευσης του αναγνωρισθέντος κειμένου.

Τέλος, έχουμε ένα πειραματικό κομμάτι στο οποίο γίνονται συγκρίσεις και εκφράζονται συμπεράσματα για νευρωνικά δίκτυα ποικίλλων προδιαγραφών (π.χ. μέθοδοι εκπαίδευσης) ώστε να αποφανθούμε ως προς το ποιο είναι καταλληλότερο για εφαρμογή τέτοιου είδους.

1.2. Επισκόπηση της πτυχιακής εργασίας

- Κεφάλαιο 2: Νευρωνικά δίκτυα

Γενικά περί νευρωνικών δικτύων, χρήσεών τους, ιστορίας, και πλεονεκτημάτων τους.

- Κεφάλαιο 3: Θεμελιώδεις έννοιες

Γίνεται η επισκόπηση των θεμελιωδών εννοιών, όπως οι τεχνητοί νευρώνες και η

λειτουργία τους, το πρότυπο του perceptron, η μέθοδος οπισθοδιάδοσης του σφάλματος, και οι αρχιτεκτονικές των δικτύων.

- Κεφάλαιο 4: Εκπαίδευση τεχνητών νευρώνων και νευρωνικών δικτύων

Συνοπτική λεκτική και μαθηματική περιγραφή των βασικών μεθόδων εκπαίδευσης και νευρωνικών δικτύων, καθώς και ορισμένων πιο εξειδικευμένων που βασίστηκαν σε αυτές.

- Κεφάλαιο 5: Αναγνώριση χαρακτήρων με νευρωνικά δίκτυα μέσω matlab

Παρουσίαση των δυνατοτήτων του matlab, οδηγίες και παράδειγμα χρήσης της αφ'ημών υλοποιηθείσας εφαρμογής στο matlab, και συνοπτική περιγραφή του τμημάτων του κώδικα και των λειτουργιών τους.

- Κεφάλαιο 6: Μετρήσεις και σύγκριση αποτελεσμάτων για νευρωνικά δίκτυα ποικίλων προδιαγραφών

Μετρήσεις για δίκτυα εμπρόσθιας τροφοδότησης και πιθανοκρατικά, με μεταβολές του αριθμού των νευρώνων και των μεθόδων εκπαίδευσης. Λαμβάνονται υπόψιν παράγοντες όπως ο χρόνος εκπαίδευσης, το ποσοστό των σφαλμάτων, και το ελάχιστο σφάλμα ανά set μετρήσεων.

- Κεφάλαιο 7: Επίλογος-συμπεράσματα

Συμπεράσματα για τα νευρωνικά δίκτυα, τις επιδόσεις τους στην εφαρμογή, και τις μετρήσεις που ελήφθησαν.

- Βιβλιογραφία - πηγές

Η χρησιμοποιηθείσα βιβλιογραφία και οι ιστοσελίδες απ’όπου αντλήθηκαν πληροφορίες, αλλά και οι εικόνες της πτυχιακής.

- Παράρτημα

Ο κώδικας της εφαρμογής με σχόλια.

2. ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

2.1. Γενικά

Ως νευρωνικό δίκτυο μπορούμε να ορίσουμε κάθε δίκτυο του οποίου η λειτουργία στηρίζεται σε διασυνδεδεμένους μεταξύ τους υπολογιστικούς κόμβους, των οποίων η συνδεσμολογία τροποποιείται όταν το συνολικό δίκτυο δέχεται κάποιο νέο ερέθισμα - είτε αυτό έχει την έννοια των ανθρωπινων αισθήσεων και εμπειριών, είτε αφορά σε σειρές ή πίνακες δεδομένων σε ηλεκτρονική μορφή.

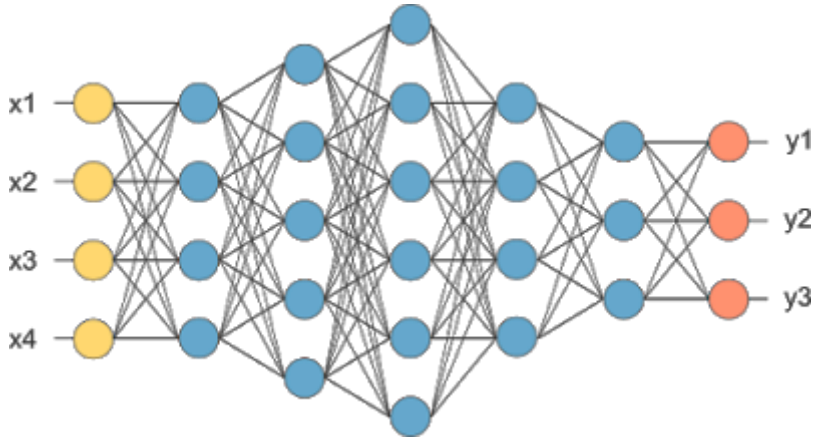
Μπορούμε να διακρίνουμε τα νευρωνικά δίκτυα σε δυο μεγάλες κατηγορίες. Η μία εξ'αυτών είναι τα επονομαζόμενα βιολογικά ή **φυσικά νευρωνικά δίκτυα** (biological ή natural neural networks), χαρακτηριστικό παράδειγμα των οποίων αποτελεί ο ανθρώπινος εγκέφαλος. Ένα φυσικό νευρωνικό δίκτυο στηρίζει τη λειτουργία του σε **βιολογικούς νευρώνες** (biological neurons) και τις μεταξύ τους διασυνδέσεις. Οι βιολογικοί νευρώνες, που θα εξεταστούν αναλυτικότερα σε άλλη ενότητα, είναι γνωστοί και ως εγκεφαλικά κύτταρα και είναι εξειδικευμένα κύτταρα του νευρικού συστήματος χαρακτηριζόμενα εκ της ιδιότητάς τους να δέχονται ηλεκτροχημικά σήματα από άλλα με τα οποία διασυνδέονται και, ανάλογα με τη φύση τους, να τα αναμεταδίδουν.

Ένα φυσικό νευρωνικό δίκτυο αποτελεί προϊόν εκατομμυρίων χρόνων εξέλιξης και, ανάλογα με το ζωικό είδος για το οποίο μιλάμε, κατά τη γέννηση του οργανισμού είναι ήδη με κάποιο τρόπο **προεκπαιδευμένο** (pretrained) και δομημένο από τη φυσική επιλογή. Ένας υγιής ανθρώπινος εγκέφαλος, φερ'ειπείν, γεννιέται προεκπαιδευμένος να μπορεί να αξιοποιήσει τα μέρη του ανθρώπινου σώματος (χέρια, πόδια, δάχτυλα, μυες προσώπου κ.α.) καθώς και να αποκτήσει ικανότητες απαραίτητες στον οργανισμό στον οποίο ανήκει (π.χ. ομιλία). Θεωρούντες το νευρωνικό δίκτυο ως ένα είδος "hardware" και τη νόηση ως μια μορφή "software", μπορούμε να θεωρήσουμε ότι ένα βιολογικό νευρωνικό δίκτυο έχει ικανότητες που αντιστοιχούν στα προεγκατεστημένα προγράμματα των λειτουργικών συστημάτων - που ο χρήστης δεν χρειάζεται να εγκαταστήσει ο ίδιος. Έτσι, ένα βιολογικό νευρωνικό δίκτυο έχει από τη φύση του λειτουργικότητα.

Αυτό σε αντιδιαστολή με την δεύτερη μεγάλη κατηγορία νευρωνικών δικτύων,

τα **τεχνητά νευρωνικά δίκτυα**. Ένα τεχνητό νευρωνικό δίκτυο δεν έχει τίποτα "προεγκατεστημένο" -δεν είναι, δηλαδή, προεκπαιδευμένο και, μέχρι να ρυθμιστεί και να εκπαιδευτεί από τον χρήστη, δεν έχει καμία πρακτική λειτουργία από μόνο του. Όπως και τα φυσικά, έτσι και τα τεχνητά νευρωνικά δίκτυα στηρίζουν τη λειτουργία τους σε νευρώνες. Ένα **τεχνητός νευρώνας** είναι ένα αλγοριθμικό κατασκεύασμα που στηρίζει τις βασικές αρχές του στις θεμελιώδεις λειτουργίες του βιολογικού. Αποτελεί τη βάση των τεχνητών νευρωνικών δικτύων.

Όπως και οι βιολογικοί, έτσι και οι τεχνητοί νευρώνες δέχονται σήματα από άλλους τέτοιους και ανάλογα με διάφορους παράγοντες και μεταβλητές που θα εξεταστούν σε άλλες ενότητες (ονομαστικά αναφέρουμε το συναπτικό βάρος και το κατώφλι [threshold] εισόδου), μεταδίδουν και οι ίδιοι κάποιο σήμα στους επόμενους με τους οποίους συνδέονται. Μια σχηματική αναπαράσταση νευρωνικού δικτύου εμφανίζεται στη παρακάτω εικόνα:



Εικόνα 2.1.1- Σχηματική αναπαράσταση τεχνητού νευρωνικού δικτύου

Οι νευρώνες του τεχνητού νευρωνικού δικτύου -απεικονιζόμενοι παραπάνω ως κόμβοι- χωρίζονται σε τρεις κατηγορίες, που αντιστοιχούν σε τρία επίπεδα του δικτύου. Η πρώτη είναι οι **νευρώνες εισόδου**, που αποτελούν την είσοδο του νευρωνικού δικτύου και δέχονται δεδομένα από τη χρήστη τα οποία μεταβιβάζουν στους **κρυμμένους** ή **υπολογιστικούς νευρώνες**. Ενώ οι νευρώνες εισόδου δεν εκτελούν υπολογισμούς και απλώς μεταβιβάζουν τα δεδομένα, οι υπολογιστικοί εκτελούν υπολογισμούς και τροποποιούν τις μεταξύ τους διασυνδέσεις για να δίνουν την επιθυμητή έξοδο. Η έξοδος δίνεται από τους **νευρώνες εξόδου**, που δέχονται

τα αποτελέσματα ως σήματα από τους υπολογιστικούς.

Τα επίπεδα ενός νευρωνικού δικτύου λοιπόν είναι τρία: εισόδου, κρυφό, και εξόδου. Το κρυφό επίπεδο (hidden layer) ενός νευρωνικού δικτύου για εμάς λειτουργεί ως ένα blackbox στο οποίο δίνουμε εισόδους και παίρνουμε εξόδους, χωρίς να μπορούμε να ξέρουμε πώς ακριβώς κατέληξε το σύστημα από την είσοδο στην έξοδο, και συνήθως χωρίς να επεμβαίνουμε άμεσα στη λειτουργία των επιμέρους νευρώνων του. Ένα νευρωνικό δίκτυο μπορεί να έχει πολλαπλά hidden-layers και αυτό μπορεί να επηρεάσει άμεσα την αποτελεσματικότητά του.

Μπορούμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο δίνοντάς του εισόδους και εξόδους και αφήνοντάς το να διασυνδέσει τους νευρώνες του κατάλληλα ώστε να καταλήγει, σε κάθε περίπτωση, από την είσοδο στην έξοδο. Ο χρήστης-δημιουργός του νευρωνικού δικτύου μπορεί να καθορίσει τις παραμέτρους που το διαμορφώνουν, όπως φερ'ειπείν ο αριθμός των νευρώνων ανά επίπεδο και ο αλγόριθμος εκπαίδευσής τους και εντοπισμού λαθών και να το αφήσει να εκπαιδευτεί από μόνο του με βάση αυτές.

Μπορούμε να θεωρήσουμε ένα τέτοιο δίκτυο ως, κατά κάποιο τρόπο, νοήμον. Τα τεχνητά νευρωνικά δίκτυα συμπεριφέρονται και αποκρίνονται σαν να έχουν κατανοήσει τη λογική πίσω από ένα πρόβλημα αντί να ακολουθούν τυποποιημένους αλγορίθμους, και ένα τέτοιο σύστημα δυνητικά μπορεί να μιμηθεί την ανθρώπινη νοημοσύνη. Προς το παρόν όμως αυτό παραμένει επιστημονική φαντασία.

Τα τεχνητά νευρωνικά δίκτυα έχουν ως βάση τους τα φυσικά αλλά δεν τα μιμούνται. Ένα τυπικό ΤΝΔ επιτελεί συγκεκριμένες μόνο διεργασίες και αποτελεί πιότερο ένα εργαλείο επεξεργασίας δεδομένων απ'ό,τι έναν εγκέφαλο με πραγματική σκέψη. Όμως η δυναμικότητα που του δίνουν οι αρχές των βιολογικών νευρώνων και των μεταξύ τους αναδιασυνδέσεων το καθιστούν ένα πολλά υποσχόμενο εργαλείο στην επιστήμη των υπολογιστών, γιατί μπορεί να επιλύσει προβλήματα για τα οποία δεν υπάρχει αλγόριθμος που μπορεί να περιγραφεί λεκτικά και προγραμματιστικά ώστε να υλοποιηθεί σε συμβατικό πρόγραμμα.

Θα προβούμε σε μια σύντομη ανασκόπηση και περιγραφή των πλεονεκτημάτων των τεχνητών νευρωνικών δικτύων έναντι των συμβατικών υπολογιστών. Στο εφεξής

με τον ευρύτερο όρο "νευρωνικά δίκτυα" θα αναφερόμαστε στα τεχνητά, ενώ όπου μιλάμε για τα φυσικά θα διευκρινίζεται.

2.2. Τα πλεονεκτήματα των νευρωνικών δικτύων έναντι των συμβατικών υπολογιστών

Όπως ανεφέρθη και προηγουμένως, η αξιοποίηση των βασικών αρχών της λειτουργίας ενός φυσικού νευρωνικού δικτύου μπορεί να προσδώσει μεγάλη δυναμικότητα σε ένα υπολογιστικό σύστημα. Η δυναμικότητα αυτή δεν οφείλεται σε αύξηση της καθ'αυτής υπολογιστικής ισχύος ή της ταχύτητας του εκάστοτε συστήματος, παρά στο ότι οι θεμελιώδεις αρχές της λειτουργίας των νευρώνων επιτρέπουν την ανάδυση της ιδιότητας των φυσικών νευρωνικών δικτύων που ονομάζουμε **πλαστικότητα**.

Ως πλαστικότητα ορίζουμε την ικανότητα ενός νευρωνικού δικτύου να αλλάζει την εσωτερική του δομή ως απόκριση σε ερεθίσματα -όπως νέες πληροφορίες-, ή σε προκύψασες ανάγκες -όπως η ανάγκη αναπροσαρμογής των διασυνδέσεων μεταξύ των νευρώνων για τροποποίηση της απόκρισης στα τεχνητά νευρωνικά δίκτυα ή, στα φυσικά, την ανάθεση κάποιας διεργασίας σε άλλο τμήμα του εγκεφάλου όταν το αρχικό έχει πάθει ζημιά (π.χ. εγκεφαλικό).

Η εκπαίδευση ενός νευρωνικού δικτύου στηρίζεται εξ'ολοκλήρου στη πλαστικότητα, γιατί κάθε αντιστοίχηση κατάστασης εισόδου σε μια επιθυμητή έξοδο περιλαμβάνει αλλαγές στις συνδέσεις αναμεταξύ των νευρώνων. Ενώ λοιπόν οι συμβατικοί υπολογιστές δεν μπορούν παρά να ακολουθήσουν τυποποιημένους αλγορίθμους, ένα νευρωνικό δίκτυο χάρη σ'αυτή του την ιδιότητα μπορεί να εντοπίσει τη λογική πίσω από ένα πρόβλημα, και να αναδιαταχθεί έτσι που να μπορεί να το λύσει.

Ένα βασικό πλεονέκτημα των νευρωνικών δικτύων λοιπόν, που απορρέει άμεσα από τη πλαστικότητα, είναι η δυνατότητα **αυτοπρογραμματισμού**. Ακολουθώντας κάποιους βασικούς αλγορίθμους λειτουργίας νευρώνων, ένα νευρωνικό δίκτυο μπορεί να εντοπίσει τη λογική πίσω από την επίλυση ενός προβλήματος χωρίς να χρειαστεί να το προγραμματίσουμε οι ίδιοι. Το μόνο που χρειάζεται είναι να του

δώσουμε παραδείγματα εισόδων και εξόδων.

Εκτός από το πλεονέκτημα της πλαστικότητας και του απορρέοντος αυτοπρογραμματισμού, ένα νευρωνικό δίκτυο χαρακτηρίζεται από το ότι, σε αντίθεση με τους υπολογιστές, η επεξεργασία δεδομένων και η μνήμη αποτελούν δυο άρρηκτα συνδεδεμένες και παράλληλες λειτουργίες. Στους συμβατικούς υπολογιστές ο χειρισμός της μνήμης είναι ανεξάρτητος και ξέχωρος από την επεξεργασία των δεδομένων. Στα νευρωνικά δίκτυα το ένα εξαρτάται άμεσα απ'το άλλο, και αυτό επιτρέπει την ταυτόχρονη επιτέλεσή τους -έχουμε πρακτικά μια **ενοποίηση της μνήμης και της επεξεργασίας και ταυτόχρονη διαχείρισή τους**.

Τέλος, τα νευρωνικά δίκτυα χαρακτηρίζονται από **απλότητα**, **από τη σκοπιά του διαχειριστή**, σε σχέση με τους υπολογιστές και τον παραδοσιακό προγραμματισμό. Ενώ στους συμβατικούς υπολογιστές η επίλυση ενός προβλήματος προϋποθέτει σύνταξη κώδικα, με ό,τι αυτή συνεπάγεται (δυσκολία εντοπισμού λαθών, ενίοτε απαίτηση συνολικής τροποποίησης σε περίπτωση που δεν λειτουργεί), το νευρωνικό δίκτυο χρειάζεται απλώς κάποιες βασικές παραμέτρους και δεδομένα για να αυτοπρογραμματιστεί.

Προτού εξετάσουμε λεπτομερέστερα τον βιολογικό και τον τεχνητό νευρώνα και τις βασικές έννοιες των τεχνητών νευρωνικών δικτύων θα προβούμε σε μια σύντομη ιστορική ανασκόπηση των προαναφερθέντων.

2.3. Σύντομη ιστορική αναδρομή στα νευρωνικά δίκτυα

Εντοπίζουμε τις απαρχές των νευρωνικών δικτύων στη δουλειά των **Walter Pitts** (που στην ηλικία των δώδεκα ετών διάβασε ολομόναχος το Principia Mathematica και απευθύνθηκε στον Bertrand Rusell για τα προβλήματα που εντόπιζε σε ορισμένες διατυπώσεις του) και **Warren McCulloch**, οιτινες ανέπτυξαν το 1943 ένα υπολογιστικό μοντέλο για τα νευρωνικά δίκτυα στο οποίο μπορούσε, θεωρητικά, να επιλυθεί κάθε πρόβλημα. Τέσσερα χρόνια αργότερα έδειξαν ότι θα μπορούσε να λύσει, φερ'ειπείν, το πρόβλημα της χωρικής αναγνώρισης προτύπων (μοτίβων).

Το 1949 ο ψυχολόγος **Donald Hebb** διατύπωσε έναν κανόνα που έπαιξε

καθοριστικό ρόλο στην ανάπτυξη των νευρωνικών δικτύων. Σύμφωνα με τον **κανόνα του Hebb**, ένας προσυναπτικός νευρώνας που πυροδοτεί συχνά έναν μετασυναπτικό (όροι που θα εξεταστούν σε επόμενη ενότητα), η μεταξύ τους σύνδεση ενισχύεται. Ο κανόνας αυτός συνοψίζεται ως "**fire together, wire together**" -οι νευρώνες, δηλαδή, που πυροδοτούνται μαζί, συνδέονται και μεταξύ τους.

Γύρω στο 1951 δημιουργήθηκε από τον **Marvin Minsky** ο πρώτος νευρωνικός υπολογιστής, με την ονομασία **Snark**, ο οποίος όμως δεν είχε καμία πρακτική χρησιμότητα. Εφτά χρόνια αργότερα οι **Rosenblatt** και **Wightman** ανέπτυξαν τον **perceptron**, έναν αλγόριθμο για αναγνώριση προτύπων που υλοποιήθηκε στη μηχανή **Mark I Perceptron**, ήτις ηδύνατο να αναγνωρίσει ψηφία σε εικόνες 20x20 pixels.

Το 1960 υλοποιήθηκε ο **ADALINE** (ADAptive LInear Neuron), ένα νευρωνικό δίκτυο ενός επιπέδου, από τον **Bernand Widrow** και τους συνεργάτες του, που οδήγησε στην ανάπτυξη σύνθετων πολυεπίπεδων νευρωνικών δικτύων από τους **Ivakhnenko** και **Lapa** ήδη το 1965. Στη πορεία όμως αποδείχθηκε πως δεν γινόταν όλα τα προβλήματα να λυθούν με τους υπάρχοντες αλγόριθμους -η λογική πύλη XOR, φερ'ειπείν, ήταν πρακτικά μη-υλοποιήσιμη με τις τότε μεθόδους-, και η έρευνα πάνω στα νευρωνικά δίκτυα σχεδόν πάγωσε ώσπου εξελίχθηκαν αρκετά οι συμβατικοί υπολογιστές.

Το ενδιαφέρον για τα νευρωνικά δίκτυα αναζωπυρώθηκε με την ανάπτυξη του **αλγορίθμου ανάστροφης αναδιάδοσης (Backpropagation)** από τον **Werbos**, που έλυνε το φαινομενικά ανεπίλυτο πρόβλημα της υλοποίησης μιας λογικής πύλης XOR σε νευρωνικό δίκτυο, το 1975. Το επόμενο κιόλας έτος αναπτύχθηκε από τους **Stephen Grossberg** και **Gail Carpenter** ένα μαθηματικό μοντέλο για την εκπαίδευση νευρωνικών δικτύων χωρίς να χάνεται η προηγούμενη γνώση τους, που έμεινε γνωστό ως **Adaptive Resonance Theory**, και χρησιμοποιήθηκε εκτενώς στη μετέπειτα έρευνα.

Από τη δεκαετία του '80, με την ανάπτυξη των **δικτύων Hopfield,** που στηρίζονταν στη μνήμη τους, και των **Neocognitrons**, οι εξελίξεις στον τομέα των νευρωνικών δικτύων είναι ραγδαίες. Σήμερα τα νευρωνικά δίκτυα χρησιμοποιούνται εκτενώς για

επίλυση προβλημάτων αναγνώρισης μοτίβων.

2.4. Εφαρμογές των νευρωνικών δικτύων

Όπως καθίσταται εμφανές εκ των άνωθεν, τα νευρωνικά δίκτυα έχουν ως πεδίο εφαρμογής κάθε πρόβλημα για το οποίο δεν υπάρχει σαφής μεθοδολογία επίλυσης και κατ'επέκτασιν αλγόριθμοι. Τα προβλήματα αυτά είναι βασικά και πρωτίστως προβλήματα αναγνώρισης προτύπων, και αναφέρουμε συνοπτικά τα εξής:

2.4.1. Αναγνώριση προσώπων

Μια από τις χαρακτηριστικότερες εφαρμογές των νευρωνικών δικτύων είναι εκείνη της αναγνώρισης προσώπων. Οι συμβατικοί υπολογιστές ακόμα και με ανεπτυγμένους αλγορίθμους ανάλυσης του προσώπου (π.χ. με υπολογισμούς αναλογιών όπως η απόσταση των ματιών σε σχέση με το μήκος του στόματος), έχουν, σε σχέση με τα νευρωνικά δίκτυα, περιορισμένες δυνατότητες αναγνώρισής τους. Τα νευρωνικά δίκτυα επιτελούν την εν λόγω λειτουργία με πολύ μεγαλύτερη επιτυχία και μπορούν να αναγνωρίσουν ένα πρόσωπο ακόμα και από φωτογραφίες προφίλ (έστω και αν η ακρίβειά τους εν προκειμένω είναι επίσης περιορισμένη). Είναι χαρακτηριστικό ότι το σύστημα DeepFace, το πιο επιτυχημένο μέχρι στιγμής σύστημα αναγνώρισης προσώπων, με 97% ακρίβεια στην αναγνώριση, στηρίχθηκε σε νευρωνικό δίκτυο.

2.4.2. Αναγνώριση χαρακτήρων

Μια άλλη εφαρμογή των νευρωνικών δικτύων, με την οποία θα καταπιαστούμε και στη παρούσα πτυχιακή εργασία, είναι η αναγνώριση χαρακτήρων μέσω αυτών - χειρόγραφων ή μη. Αν και οι συμβατικοί υπολογιστές μπορούν να αναγνωρίσουν χαρακτήρες, πρέπει να πληρούνται προϋποθέσεις όπως το να είναι καθαρογραμμένοι, και για έναν αλγόριθμο συμβατικού υπολογιστή η ακρίβεια αναγνώρισης σπανίως υπερβαίνει το 95% -ένα ποσοστό που ένα νευρωνικό δίκτυο μπορεί να υπερβεί εως και τέσσερις μονάδες. Το πρόβλημα της άνευ σφαλμάτων αναγνώρισης των ενωμένων χαρακτήρων ή της πλάγιας/καλλιγραφικής γραφής, επίσης, αναμένεται να επιλυθεί από τα νευρωνικά δίκτυα αντί των συμβατικών υπολογιστών.

2.4.3. Ιατρικές διαγνώσεις

Τα νευρωνικά δίκτυα μπορούν να εκπαιδευτούν ώστε να διαγιγνώσκουν, με βάση συμπτώματα και ενδείξεις, μεγάλη πληθώρα ασθενειών, συχνά με μεγαλύτερη επιτυχία από τον άνθρωπο. Είναι χαρακτηριστικό πως το 2016 αναπτύχθηκε νευρωνικό δίκτυο αναγνώρισης του καρκίνου του πνεύμονα με εμφανώς μεγαλύτερα ποσοστά ορθών διαγνώσεων σε σχέση με τους γιατρούς, των οποίων οι δυνατότητες περιορίζονται από ανθρώπινους παράγοντες όπως η κούραση, η υπερεμπιστοσύνη στην εμπειρία τους, ή η αδυναμία συγκράτησης και διαχείρισης μεγάλου όγκου δεδομένων -περιορισμοί που δεν υφίστανται στα νευρωνικά δίκτυα. Ενδεικτικά αναφέρουμε πως τα προαναφερθέντα διαγιγνώσκουν την οξεία νεφρίτιδα με ποσοστό επιτυχίας 99% και τις καρδιοπάθειες με 95%.

2.4.4. Λήψη αποφάσεων σε σύνθετα συστήματα

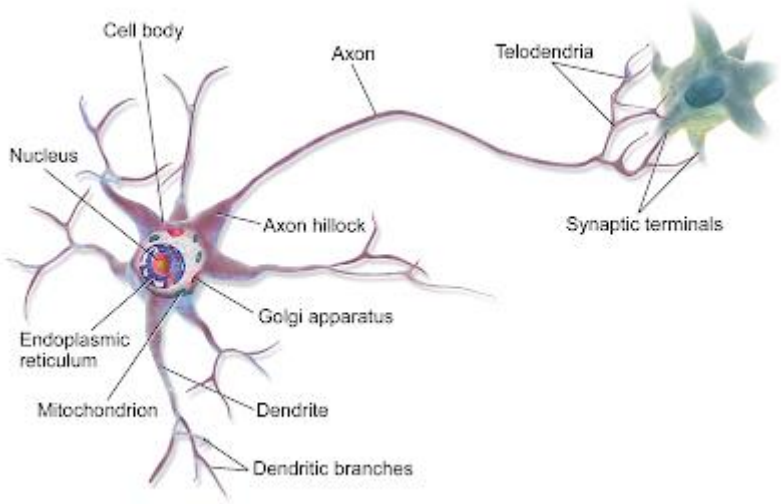
Τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν επίσης για τη λήψη αποφάσεων σε σύνθετα συστήματα, όπως ο έλεγχος οχημάτων ή το σκάκι. Είναι χαρακτηριστικό ότι το 2015 το νευρωνικό δίκτυο Giraffe που εκπαιδεύτηκε 72 ώρες στο σκάκι, μπόρεσε να αντιμετωπίσει masters του σκακιού -με χιλιάδες ώρες εμπειρίας- ως ίσο προς ίσους.

3. ΘΕΜΕΛΙΩΔΕΙΣ ΕΝΝΟΙΕΣ

3.1. Βιολογικοί και τεχνητοί νευρώνες

3.1.1 Ο βιολογικός νευρώνας

Ως ανεφέρθη, τα νευρωνικά δίκτυα στηρίζονται στις αρχές λειτουργίας των βιολογικών νευρώνων -ήτοι, των εγκεφαλικών μας κυττάρων. Ένας βιολογικός νευρώνας απεικονίζεται στη παρακάτω εικόνα:



Εικόνα 3.1.1 - Ο βιολογικός νευρώνας

Ο βιολογικός νευρώνας είναι ένα κύτταρο που ειδικεύεται στη λήψη και αναμετάδοση ηλεκτροχημικών σημάτων, και πέραν του κυτταρικού σώματος χαρακτηρίζεται και δομές όπως ο **άξονας**, κατά μήκος του οποίου μεταδίδονται τα εκάστοτε σήματα προς τα επόμενα κύτταρα, και οι **δενδρίτες**, που λαμβάνουν τα σήματα από άλλους νευρώνες (τους επονομαζόμενους προσυναπτικούς), αν βρίσκονται συνδεδεμένοι στο κυτταρικό σώμα, ή τα μεταδίδουν σε άλλους (τους μετασυναπτικούς), αν βρίσκονται στο τέρμα του άξονα.

Τα σημεία διασύνδεσης των νευρώνων ονομάζονται **συνάψεις**. Ένας ανθρώπινος νευρώνας μπορεί να έχει και περισσότερους από 1000 δενδρίτες, που επιτρέπουν τη σύνδεση με ακόμα μεγαλύτερο αριθμό άλλων νευρώνων. Ένας τυπικός νευρώνας συνδέεται με περίπου 7000 άλλους νευρώνες, και το γεγονός ότι ο ανθρώπινος

εγκέφαλος έχει συνολικά περίπου 100 δισεκατομμύρια τέτοιους συνεπάγεται έναν αστρονομικό αριθμό συνολικών συνάψεων, ο αριθμός των οποίων επηρεάζει άμεσα τις υπολογιστικές δυνατότητες ενός νευρωνικού δικτύου.

Μια σύναψη μπορεί να είναι είτε **διεγερτική** είτε **ανασταλτική**. Στη πρώτη περίπτωση, ένα σήμα από έναν άλλο νευρώνα μπορεί να αυξήσει την ηλεκτρική δραστηριότητα του λαμβάνοντος κυττάρου, ενώ στη δεύτερη να τη μειώσει. Αν τα σήματα που λαμβάνονται από τους άλλους νευρώνες συνολικά υπερβαίνουν κάποιο **κατώφλι** ενέργειας, ο νευρώνας μεταδίδει μέσω του άξονά του ένα σήμα στους μετασυναπτικούς του νευρώνες.

Η **πλαστικότητα** των φυσικών νευρώνων είναι άρρηκτα συνδεδεμένη με τις έννοιες της **βραχυπρόθεσμης** και **μακροπρόθεσμης ισχυροποίησης**. Εν ολίγοις, όταν δυο συνδεδεμένοι νευρώνες στέλνουν σήματα ο ένας στον άλλον συχνά, η μεταξύ τους σύνδεση ισχυροποιείται - είτε βραχυπρόθεσμα είτε μακροπρόθεσμα, ανάλογα με τη συχνότητα της διαδικασίας- ενώ αν δεν αποστέλλουν σήματα συχνά η μεταξύ τους διασύνδεση εκφυλίζεται.

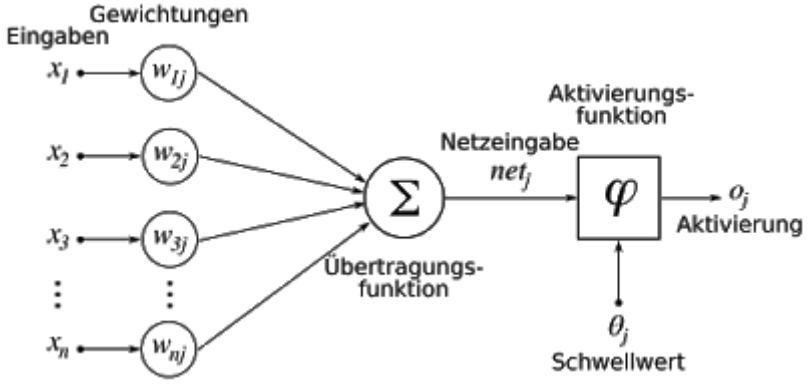
Η ισχυροποίηση έγκειται στην αύξηση του **συναπτικού βάρους** που έχει ο ένας νευρώνας για τον άλλον. Όσο μεγαλύτερο το συναπτικό βάρος ενός νευρώνα για έναν άλλον, τόσο μεγαλύτερη η επιρροή του σ'αυτόν και κατ'επέκταση στη τάση της μεμβράνης του.

Τα σήματα που μεταδίδονται κατά μήκος των αξόνων έχουν ταχύτητα έως 119m/sec. Το μέγεθός τους συνεπάγεται πως είναι μια αρκετά μεγάλη ταχύτητα για τη λειτουργία τους, και επιτρέπει στον εγκέφαλο επεξεργασία ερεθισμάτων σε πολύ μικρά κλάσματα του δευτερολέπτου. Οι βασικές αρχές και έννοιες των βιολογικών νευρώνων απαντώνται και στους τεχνητούς.

3.1.2 Ο τεχνητός νευρώνας

Ο τεχνητός νευρώνας είναι μια αλγοριθμική ή μαθηματική συνάρτηση που αποτελεί το δομικό στοιχείο των τεχνητών νευρωνικών δικτύων. Όπως και ο βιολογικός νευρώνας, λαμβάνει σήματα από άλλους νευρώνες του συστήματος και ανάλογα με

το αν ξεπερνούν το αντίστοιχο κατώφλι -που υπολογίζεται από μια συνάρτηση που λαμβάνει υπόψιν τα σήματα των νευρώνων και το βάρος τους ή είναι προκαθορισμένο-, μεταδίδει κάποιο σήμα σε άλλους. Στη παρακάτω εικόνα απεικονίζεται το βασικό μοντέλο του τεχνητού νευρώνα:



Εικόνα 3.1.2 - Ο τεχνητός νευρώνας

όπου x1 έως xn και w1j έως wnj οι **είσοδοι** και τα **βάρη** τους αντίστοιχα, το άθροισμα Σ ("συνάρτηση μεταφοράς" κατά την εικόνα) που είναι το **άθροισμα των γινομένων** τους, φ η **συνάρτηση μεταφοράς** (ή, κατά την εικόνα, "συνάρτηση ενεργοποίησης"), και θj η **τιμή του κατωφλίου** (γνωστή και ως ουδός πυροδότησης).

Οι είσοδοι δεν είναι παρά τα σήματα από άλλους νευρώνες. Τα βάρη είναι αντίστοιχα των βιολογικών συναπτικών βαρών, όπου η επιρροή κάθε σήματος εξαρτάται από το πόσο καλή είναι η σύνδεση μεταξύ του νευρώνα-αποστολέα και του νευρώνα-παραλήπτη του σήματος. Τα βάρη αυτά δεν είναι σταθερά και μεταβάλλονται από το σύστημα ούτως ώστε να δίνει το επιθυμητό αποτέλεσμα, όπως μπορεί να μεταβάλλεται και η διάταξη των συνδέσεων. Η συνάρτηση μεταφοράς καταπιάνεται με τον υπολογισμό της πρέπουσας απόκρισης ανάλογα με τις εισόδους.

Η έξοδος του τεχνητού νευρώνα εν προκειμένω δίδεται από τον παρακάτω τύπο:

$$o_j = \varphi(\sum_{a=0}^{n} w_{aj} x_a - \vartheta j)$$

Τύπος 3.1.1 - Η έξοδος του τεχνητού νευρώνα

Εναλλακτικά μπορούμε να απαλείψουμε τον ουδό πυροδότησης -να μην τον λάβουμε υπόψιν ή να τον θεωρήσουμε προϋπόθεση και κομμάτι της συνάρτησης μεταφοράς- και η έξοδος να ισούται με

$$o_j = \varphi(\sum_{a=0}^{n} w_{aj} x_a)$$

Τύπος 3.1.2 - Η έξοδος του τεχνητού νευρώνα χωρίς ουδό πυροδότησης

Μια συνάρτηση μεταφοράς φ μπορεί να είναι οποιουδήποτε είδους συνάρτηση, ανάλογα με την τοπολογία και τους σκοπούς του δικτύου ή και τη θέση του νευρώνα σ'αυτό (π.χ. άλλη συνάρτηση μεταφοράς για έναν νευρώνα στο κρυμμένο επίπεδο 1, άλλη για κάποιον στο 2, άλλη για των εισόδων, κ.ο.κ).

Μια συνηθης συνάρτηση μεταφοράς είναι η επονομαζόμενη βηματική. Στη βηματική συνάρτηση μεταφοράς η έξοδος ισούται με 1 αν τηρείται κάποια συνθήκη και με 0 σε όλες τις άλλες περιπτώσεις. Στη περίπτωση του ουδού ενεργοποίησης, φερ'ειπείν, η βηματική συνάρτηση περιγράφεται ως εξής:

$$y = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$$

Τύπος 3.1.3 - Η βηματική συνάρτηση

αν θεωρήσουμε ότι για το u είναι το άθροισμα των γινομένων των εισόδων και των βαρών, δηλαδή

$$u = \sum_{a=0}^{n} w_{aj} x_a$$

Τύπος 3.1.4 - Το άθροισμα γινομένων εισόδων και βαρών

Στους τεχνητούς νευρώνες στηρίζονται οι **perceptrons** (αντίληπτρα), που αποτελούν το πιο κλασσικό παράδειγμα γραμμικού ταξινομητή και τα απλούστερα

νευρωνικά δίκτυα που υλοποιήθηκαν.

3.2. Ο perceptron

Ο perceptron ("αντίληπτρο" ή "αισθητήρας" στην Ελληνική βιβλιογραφία) είναι ένας εμπροσθετοτροφοδοτούμενος (feed-forward) γραμμικός ταξινομητής (linear classifier) βασισμένος στις αρχές του τεχνητού νευρώνα και δυνάμενος να λύσει προβλήματα γραμμικής ταξινόμησης. Αν και οι δυνατότητές του είναι, δηλαδή, περιορισμένες, καθώς δεν υπάρχει γραμμική ταξινόμηση σε όλα τα προβλήματα, παραμένει ένα καλό απλοϊκό μοντέλο νευρωνικού δικτύου που χρησιμοποιείται ως σημείο αναφοράς.

Η απόκριση ενός perceptron μιας εισόδου του περιγράφεται ως:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{else} \end{cases}$$

Τύπος 3.2.1 - Η απόκριση του perceptron

όπου w το συναπτικό βάρος της εισόδου, x η είσοδος, και b το bias -μια τιμή που καθορίζεται από τους σκοπούς της εφαρμογής και μπορεί να μείνει μηδενική. Για περισσότερες εισόδους προσθέτουμε το b στο άθροισμα των γινομένων των εισόδων και των βαρών.

Για την εκπαίδευση ενός perceptron ακολουθούνται τρεις βασικοί κανόνες:

1. Αν η έξοδος για μια είσοδο X είναι 1 και έπρεπε να είναι 1, το συναπτικό της βάρος μένει ως έχει

2. Αν η έξοδος είναι 0 ενώ έπρεπε να είναι 1, το συναπτικό βάρος αυξάνεται.

3. Ειδάλλως (αν είναι 1 ενώ έπρεπε να είναι 0), μειώνεται.

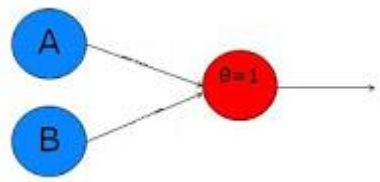
Ο μαθηματικός φορμαλισμός των παραπάνω, όπως εμφανίζεται στη γερμανόφωνη wikipedia, έχει ως

$$w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta w_{ij} ,$$
$$\Delta w_{ij} = \alpha \cdot (t_j - o_j) \cdot x_i .$$

Τύπος 3.2.2 - Μεταβολή συναπτικού βάρους σε perceptron

όπου j ο νευρώνας εξόδου, i ο νευρώνας ή το σύστημα εισόδου, Wij το συναπτικό βάρος του i για τον j, Tj η επιθυμητή έξοδος, Oj η δοθείσα έξοδος, Xi η είσοδος, και α ο ρυθμός εκμάθησης. Όσο μεγαλύτερος ο ρυθμός εκμάθησης ή η ασυμφωνία επιθυμητής και δοθείσας εξόδου, τόσο μεγαλύτερη η μεταβολή του συναπτικού βάρους.

Μια λογική πύλη AND δυο εισόδων με έναν perceptron έχει ως εξής:



Εικόνα 3.2.1 - Λογική πύλη AND με perceptron

Μπορούμε να θεωρήσουμε το κατώφλι θ ως ένα bias ίσο με την αρνητική του τιμή, δηλαδή b = -θ = -1. Με βάση τη περιγραφή απόκρισης,η έξοδος θα ισούται με ένα μόνο όταν το άθροισμα των γινομένων εισόδων και συναπτικών βαρών ξεπερνάει την τιμή b.

Εν προκειμένω, αν το συναπτικό βάρος του A ήταν 0.4 και του B ήταν 0.5 και ενεργοποιούνταν και τα δυο, το άθροισμα των γινομένων για τιμή ίση με τη μονάδα θα ισούτο με 0.9, και ως εκ τούτου τα συναπτικά βάρη θα έπρεπε να τροποποιηθούν ώστε συνολικά να βγαίνει πάνω από ένα. Η μηδενική επιθυμητή έξοδος για τη περίπτωση που μόνο μια είσοδος από τις δυο ενεργοποιείτο θα απέκλειε το σενάριο του να πάρει, φερ'ειπείν, το B συναπτικό βάρος 1.5 και να ξεπερνάει την τιμή του θ από μόνο του.

Ο αλγόριθμος εκπαίδευσης του perceptron αποτελείται από τρία απλά βήματα:

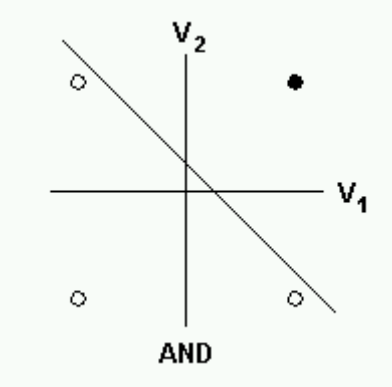
1. Απόδοση τυχαίου -μηδενικού η μη- συναπτικού βάρους στις εισόδους.

2. Υπολογισμός εξόδου και σύγκρισή της με την επιθυμητή

3. Τροποποίηση συναπτικών βαρών με βάση την εξίσωση.

Τα βήματα (2) και (3) επαναλαμβάνονται μέχρι να έχουμε κανονική έξοδο.

Ως γραμμικό σύστημα, η πύλη AND μπορεί να απεικονιστεί ως εξής:



Εικόνα 3.2.2 - Λογική πύλη AND ως γραμμικό σύστημα

Στη παραπάνω εικόνα η γραμμή διαχωρίζει τις επιθυμητές από τις ανεπιθύμητες εξόδους με απλό και σαφέστατο τρόπο. Δεν είναι, ωστόσο, όλα τα προβλήματα γραμμικά: μια λογική πύλη XOR, για παράδειγμα, δεν μπορεί να απεικονιστεί και να αντιμετωπιστεί ως γραμμικό σύστημα, και η υλοποίησή της είναι αδύνατη σε έναν απλό perceptron.

Παρά τις περιορισμένες του δυνατότητες όμως, ο perceptron παραμένει ένα καλό μοντέλο απλοϊκού νευρωνικού δικτύου.

3.3. Η μέθοδος οπισθοδιάδοσης σφάλματος

Η μέθοδος οπισθοδιάδοσης σφάλματος (error backpropagation) αποτελεί τη βασικότερη μέθοδο εκπαίδευσης νευρωνικών δικτύων, και οι κυριότεροι αλγόριθμοι εκπαίδευσής τους στηρίζονται στις αρχές της.

Όπως είδαμε προηγουμένως, τα πολύ απλά νευρωνικά δίκτυα τύπου perceptron έχουν μάλλον περιορισμένες δυνατότητες. Στα πιο προηγμένα, που χρησιμοποιούνται και για προβλήματα μη-γραμμικής ταξινόμησης, υπάρχουν

περισσότερα του ενός επίπεδα και η έξοδος δεν είναι το άθροισμα των εισόδων που υπερβαίνει κάποιο κατώφλι, αλλά σύνθετες μαθηματικές συναρτήσεις. Για την εκπαίδευση τέτοιων δικτύων χρησιμοποιείται η μέθοδος οπισθοδιάδοσης του σφάλματος, στην οποία, όπως υποδηλώνει το όνομα, το σφάλμα της εξόδου διαδίδεται στο δίκτυο για να προσαρμόσει τα συναπτικά του βάρη.

Ο αλγόριθμος της μεθόδου μπορεί να περιγραφεί με τρία βήματα:

1. Εισάγονται οι είσοδοι στο δίκτυο και υπολογίζεται η έξοδος

2. Η έξοδος του δικτύου συγκρίνεται με την επιθυμητή, και η διαφορά τους θεωρείται ως το *σφάλμα* του δικτύου

3. Το σφάλμα αναμεταδίδεται στο επίπεδο εισόδου, και μεταβάλλονται τα βάρη των νευρώνων ανάλογα με τη συνεισφορά τους στο σφάλμα.

Θεωρούμε ότι η έξοδος *o* ενός νευρώνα *j* ισούται με φ(netj), όπου *net* η είσοδος και *φ* το αντίστοιχο της συνάρτησης μεταφοράς του νευρώνα. Το σφάλμα υπολογίζεται ως

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - o_i)^2.$$

Τύπος 3.3.1. - Το σφάλμα ενός νευρώνα

για δεδομένα από i έως n, όπου t η επιθυμητή (target) έξοδος και o η δοθείσα.

Για τον υπολογισμό της πρέπουσας μεταβολής των συναπτικών βαρών είναι απαραίτητος ο υπολογισμός της μερικής παραγώγου του σφάλματος ως προς τα εκάστοτε συναπτικά βάρη.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j}\frac{\partial \mathrm{net}_j}{\partial w_{ij}}$$

Τύπος 3.3.2. - Καθορισμός παραγώγου σφάλματος προς συναπτικό βάρος

όπου *net* η είσοδος κάποιου νευρώνα. Η μεταβολή των βαρών ισούται με

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \delta_j o_i$$

Τύπος 3.3.4. - Μεταβολή συναπτικού βάρους

όπου *η* ο ρυθμός εκμάθησης και

$$\delta_j = \begin{cases} \varphi'(\mathrm{net}_j)(o_j - t_j) & \text{falls } j \text{ Ausgabeneuron ist,} \\ \varphi'(\mathrm{net}_j) \sum_k \delta_k w_{jk} & \text{falls } j \text{ verdecktes Neuron ist.} \end{cases}$$

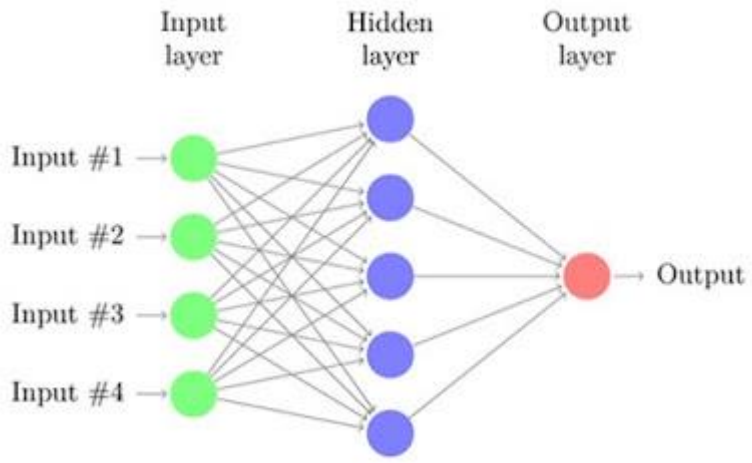
Τύπος 3.3.5. - Περιγραφή παραμέτρου δj

δηλαδή, άλλο *δj* για τους νευρώνες εξόδου (Ausgabeneurone) και άλλο *δj* για τους κρυμμένους (verdeckte) νευρώνες. Το *κ* του τύπου είναι ο αριθμός (index) του μετασυναπτικού νευρώνα του *j*, το *φ* μια παραγωγίσιμη συνάρτηση μεταφοράς, *oj* η έξοδος του νευρώνα, και *tj* η επιθυμητή του έξοδος .

3.4. Αρχιτεκτονικές των νευρωνικών δικτύων

3.4.1. Δίκτυα εμπρόσθιας τροφοδότησης

Τα δίκτυα εμπρόσθιας τροφοδότησης (feedforward) αποτελούν την απλούστερη κατηγορία νευρωνικών δικτύων, και την ευκολότερη όσον αφορά την εκπαίδευση. Στην εν λόγω αρχιτεκτονική οι είσοδοι μεταβιβάζονται κατευθείαν -χωρίς καθυστέρηση, όπως σε άλλες αρχιτεκτονικές- στους υπολογιστικούς νευρώνες, και αφότου γίνουν οι όποιοι υπολογισμοί, το αποτέλεσμα μεταδίδεται στους νευρώνες εξόδου. Ένα τέτοιο δίκτυο μπορεί να απεικονιστεί ως εξής:



Εικόνα 3.4.1 - Η δομή ενός νευρωνικού δικτύου εμπρόσθιας τροφοδότησης

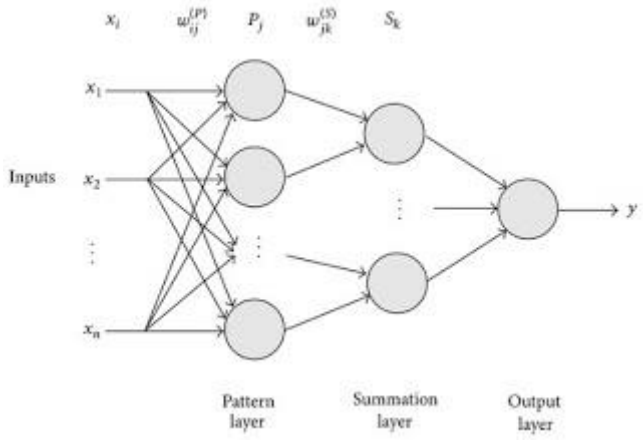
Ένα δίκτυο εμπρόσθιας τροφοδότησης μπορεί να έχει περισσότερα από ένα υπολογιστικά/κρυφά επίπεδα.

Τα συγκεκριμένα δίκτυα με τη κατάλληλη μέθοδο εκπαίδευσης και τον πρέποντα αριθμό νευρώνων μπορούν να γίνουν πολύ αποτελεσματικά, αν και ενίοτε προτιμούνται αρχιτεκτονικές ανατροφοδότησης.

3.4.2 Πιθανοκρατικά νευρωνικά δίκτυα

Ένα πιθανοκρατικό νευρωνικό δίκτυο (probabilistic neural network ή PNN) είναι μια μορφή δικτύου εμπρόσθιας τροφοδότησης, αποτελούμενη όμως από τέσσερα επίπεδα αντί τριών. Συγκεκριμένα, ανάμεσα στους υπολογιστικούς/κρυφούς νευρώνες και το επίπεδο εξόδου μεσολαβεί ένα **επίπεδο άθροισης** (summation layer) στο οποίο υπολογίζεται αθροιστικά η πιθανότητα μια είσοδος να αντιστοιχεί σε μια έξοδο. Το επίπεδο άθροισης έχει έναν αριθμό νευρώνων τουλάχιστον ίσο με τις κατηγορίες των πιθανών εξόδων (φερ'ειπείν, στην αναγνώριση χαρακτήρων του αλφαβήτου ίσο με τον αριθμό των γραμμάτων), και το επίπεδο εξόδου απλώς συγκρίνει τις εξόδους του επιπέδου άθροισης και αποκρίνεται ανάλογα.

Μια απεικόνιση πιθανοκρατικού δικτύου φαίνεται στη παρακάτω εικόνα:



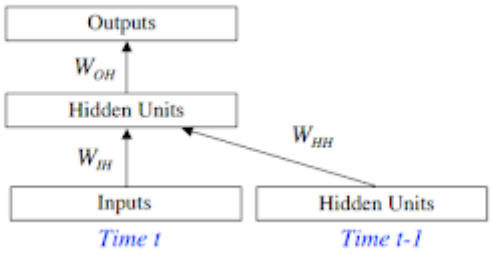
Εικόνα 3.4.2 - Η δομή ενός πιθανοκρατικού νευρωνικού δικτύου

3.4.3 Περιοδικά νευρωνικά δίκτυα

Τα περιοδικά νευρωνικά δίκτυα (recurrent neural networks) είναι μια κατηγορία δικτύων άτινα, εν αντιθέσει με τα εμπρόσθιας τροφοδότησης, επιτρέπουν **ανατροφοδότηση** (feedback). Επιτρέπουν δηλαδή μετάδοση πληροφορίας σε δυο κατευθύνσεις - όχι μόνο από τους υπολογιστικούς νευρώνες προς την έξοδο, αλλά και από την έξοδο προς τους υπολογιστικούς νευρώνες ή από το ίδιο το υπολογιστικό επίπεδο στον εαυτό του ή σε προηγούμενα τέτοια (αν είναι περισσότερα από ένα). Είναι, δηλαδή, δίκτυα που ανατροφοδοτούν τις εξόδους διαφόρων επιπέδων στην είσοδο -με τις συνδέσεις μεταξύ των επιμέρους επιπέδων του να δημιουργούν έναν κύκλο επιπέδων, με ανατροφοδότηση από το ένα στο άλλο κάθε **περίοδο** υπολογισμού.

Η συγκεκριμένη κατηγορία δικτύων χαρακτηρίζεται από δυναμική και μεταβαλλόμενη συμπεριφορά, καθώς και από τη δυνατότητα να χρησιμοποιεί τη μνήμη της για να επεξεργαστεί εισόδους. Τα δίκτυα αυτά μπορούν είτε να έχουν ανατροφοδότηση σε επίπεδο νευρώνων, είτε σε επίπεδο επιπέδων.

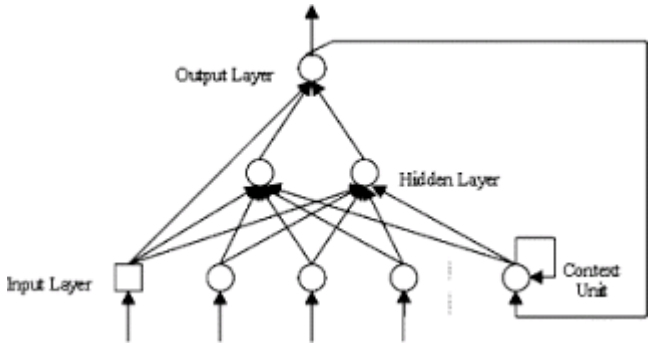
Η απλούστερη μορφή αυτής της κατηγορίας δικτύων, γνωστή ως περιοδικό **δίκτυο Elman**, έχει ως εξής:



Εικόνα 3.4.3 - Η δομή ενός περιοδικού δικτύου Elman

όπου Whh (hidden to hidden) η έξοδος του κρυφού επιπέδου τη προηγούμενη χρονική στιγμή t-1, που χρησιμοποιείται ως δεύτερη είσοδος για το κρυφό επίπεδο.

Μια παραλλαγή τους είναι τα **δίκτυα Jordan**, στα οποία ως δεύτερη είσοδος στο κρυφό επίπεδο μπαίνει η προηγούμενη έξοδος του συνολικού δικτύου (το outputs αντί του hidden units της παραπάνω εικόνας) αντί της εξόδου του κρυφού επιπέδου τη προηγούμενη στιγμή:



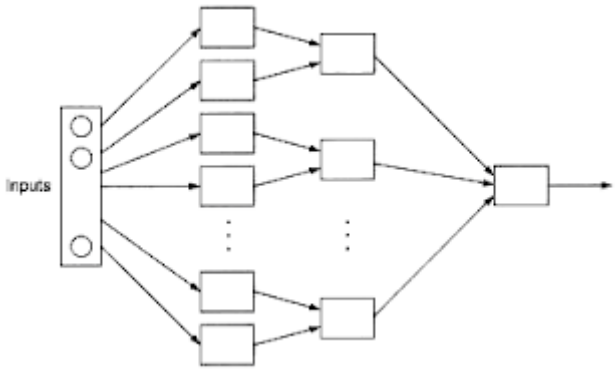
Εικόνα 3.4.4 - Η δομή ενός περιοδικού δικτύου Jordan

Άλλες υποκατηγορίες περιοδικών νευρωνικών δικτύων, ονομαστικά, είναι τα **αναδρομικά** (recursive) νευρωνικά δίκτυα, στα οποία η επεξεργασία των δεδομένων γίνεται ιεραρχικά σε πολλαπλά επίπεδα, και τα **δίκτυα Hopfield** (βλ. ιστορική αναδρομή), που στηρίζονται στη μνήμη αντί της αναγνώρισης μοτίβων.

3.4.4. Δομοστοιχειωτά νευρωνικά δίκτυα

Τα δομοστοιχειωτά νευρωνικά δίκτυα (modular neural networks) αποτελούνται από ανεξάρτητα επιμέρους δίκτυα, έκαστο εκ των οποίων αναλαμβάνει τη διεκπεραίωση κάποιας υποεργασίας, και από ένα δίκτυο μεσολάβησης (intermediary) που χρησιμοποιεί τις εξόδους των άλλων υποδικτύων για να καταλήξει σε ένα

αποτέλεσμα ή στη λήψη μιας απόφασης. Μια απεικόνιση τέτοιου δικτύου φαίνεται παρακάτω:



Εικόνα 3.4.5 - Η δομή ενός δομοστοιχειωτού νευρωνικού δικτύου

Το τελευταίο δίκτυο πριν την έξοδο το θεωρούμε ως το δίκτυο μεσολάβησης, και μπορούμε να θεωρήσουμε ότι είναι το καθ'αυτό νευρωνικό δίκτυο που απλώς δέχεται εισόδους επεξεργασμένες από άλλα νευρωνικά δίκτυα.

3.4.5 Δίκτυα συνάρτησης ακτινικής βάσης

Τα δίκτυα συνάρτησης ακτινικής βάσης (radial basies function networks) είναι δίκτυα που χρησιμοποιούν συναρτήσεις ακτινικής βάσης ως συναρτήσεις μεταφοράς στους νευρώνες. Τυπικά έχουν εμπροσθοτοτροφοδοτούμενη αρχιτεκτονική, αποτελούμενη από τρια επίπεδα: εισόδου, νευρώνων μη-γραμμικών συναρτήσεων ακτινικής βάσης, και εξόδου.

Χρησιμοποιούνται κυρίως σε μαθηματικές εφαρμογές, όπως προβλέψεις σε χαοτικά συστήματα και σε χρονολογικές σειρές.

´

4. ΕΚΠΑΙΔΕΥΣΗ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΩΝ ΚΑΙ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

4.1. Εκπαίδευση εκ διορθώσεως σφαλμάτων

Η πιο απλή, ίσως, μέθοδος εκπαίδευσης ενός νευρώνα είναι η επονομαζόμενη εκπαίδευση εκ **διορθώσεως σφαλμάτων (error-correction learning)**. Θεωρούντες έναν νευρώνα *p* που διεγείρεται μια δεδομένη χρονική στιγμή *n* από ένα σήμα *x(n)*, που μπορεί να είναι έξοδος κάποιου προηγούμενου νευρώνα του δικτύου ή και απλή είσοδος ελεγχόμενη από τον χειριστή του δικτύου, μπορούμε να εκλάβουμε την έξοδο του νευρώνα p ως Yp(n).

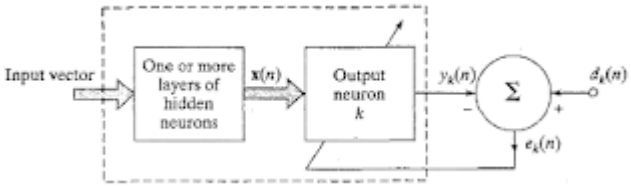
Η δοθείσα έξοδος Yp(n) συγκρίνεται με την επιθυμητή έξοδο (disired ή target output), έστω Tp(n), και η μεταξύ τους διαφορά ή σφάλμα Dp(n) εξάγεται από τη σχέση

$$Dp(n) = Tp(n) - Yp(n)$$

Τύπος 4.1.1 - Σφάλμα εξόδου νευρώνα

Όσο μεγαλύτερη η ασυμφωνία μεταξύ επιθυμητής και δοθείσας τιμής, τόσο μεγαλύτερο το σφάλμα της εξόδου. Σκοπός της εκπαίδευσης ενός νευρώνα -όπως και ενός νευρωνικού δικτύου εν γένει- είναι η εξομάλυνση της διαφοράς μεταξύ δοθέντων και επιθυμητών εξόδων δια της τροποποιήσεως των συναπτικών συνδέσεων ή βαρών.

Η πραγματοποίηση του συγκεκριμένου σκοπού επιτυγχάνεται εύκολα μέσω ενός feedback, στο οποίο το σφάλμα Dp(n) δίδεται στον νευρώνα ως είσοδος. Παρακάτω απεικονίζεται ένας νευρώνας με feedback για το σφάλμα κατά τη χρονική στιγμή n:



Εικόνα 4.1.1 - Τεχνητός νευρώνας με feedback

όπου Yk(n) η έξοδος του νευρώνα, Dk(n) η επιθυμητή έξοδος, και ek(n) το σφάλμα.

Στην εκπαίδευση μεμονωμένων νευρώνων αποβλέπουμε πιότερο στη μεταβολή των βαρών των προσυναπτικών νευρώνων απ'ό,τι στην αλλαγή των καθ'αυτών συνδέσεων, και η επιθυμητή τροποποίηση του συναπτικού βάρους Δwpj(n) δίδεται από τη σχέση

$$\Delta w_{pj}(n) = \eta D_p(n) X_j(n)$$

Τύπος 4.1.2 - Μεταβολή συναπτικού βάρους

όπου το η είναι μια σταθερά που αντιστοιχεί στον **ρυθμό εκμάθησης (rate of learning)** του δικτύου. Η παραπάνω σχέση είναι γνωστή ως **κανόνας Widrow-Hoff (Widrow-Hoff rule)** και είναι από τις πλέον θεμελιώδεις στα νευρωνικά δίκτυα. Για την εφαρμογή του θεωρούμε αυτοδικαίως ότι το σήμα του σφάλματος για έναν νευρώνα μπορεί να μετρηθεί άμεσα

Το συναπτικό βάρος κατά τη στιγμή n+1 ισούται με το άθροισμα του προηγούμενου και της μεταβολής, δηλαδή

$$W_{pj}(n+1) = W_{pj}(n) + \Delta W_{pj}(n)$$

Τύπος 4.1.3 - Νέο συναπτικό βάρος

Να σημειωθεί ότι μια τιμή που λαμβάνεται υπόψιν είναι το mean squared error, που ισούται με

(Dp(n)^2)/2.

4.2. Hebbian learning

Στην ιστορική αναδρομή αναφερθήκαμε στον **κανόνα του Hebb**, σύμφωνα με τον οποίο όσο πιο συχνά αποστέλλει ένας νευρώνας σήματα σε έναν άλλον τόσο ισχυρότερη γίνεται η μεταξύ τους διασύνδεση. Ο κανόνας αυτός, που συνοψίζεται ως "**fire together, wire toghether**", αποτελεί μια από τις βασικότερες αρχές τόσο των φυσικών όσο και των τεχνητών νευρωνικών δικτύων, και τον πιο απλό αλγόριθμο τροποποίησης των συναπτικών βαρών: όσο συχνότερη η διέγερση από το προσυναπτικό κύτταρο, τόσο ισχυρότερη η διασύνδεσή του με το μετασυναπτικό την επόμενη στιγμή.

Μπορούμε να τον περιγράψουμε με δυο βασικές αρχές:

1. Αν δυο συνδεδεμένοι μεταξύ τους νευρώνες ενεργοποιούνται ταυτόχρονα λόγω σήματος του ενός στον άλλον, η μεταξύ τους διασύνδεση ισχυροποιείται.

2. Αν δεν ενεργοποιούνται ταυτόχρονα, και δεν αποστέλλουν σήματα ο ένας στον άλλον, η μεταξύ τους σύναψη εκφυλίζεται.

Θεωρούμε τις συνάψεις που ακολουθούν αυτόν τον κανόνα ως Hebbian, και σε τεχνητά νευρωνικά δίκτυα μπορούν να υλοποιηθούν διατάξεις στηριζόμενες σε άλλους κανόνες.

Ο υπολογισμός του συναπτικού βάρους στη εκπαίδευση κατά Hebb έχει ως εξής:

$$w_{ij} = \frac{1}{p} \sum_{k=1}^{p} x_i^k x_j^k,$$

Τύπος 4.2.1 - Συναπτικό βάρος στην εκπαίδευση κατά Hebb

όπου x οι είσοδοι των νευρώνων i και j για k δεδομένα, από συνολικό αριθμό δεδομένων p.

Για εκπαίδευση νευρωνικού δικτύου με p παραδείγματα, δηλαδή, ο συναπτικό βάρος της διασύνδεσης οποιονδήποτε νευρώνων i και j, θα ισούτο με το άθροισμα των γινομένων των εισόδων διαιρεμένο δια p. Όπως καθίσταται πασιφανές, δυο νευρώνες που ενεργοποιούνται ταυτόχρονα (που έχουν, δηλαδή, μη μηδενική είσοδο

x), έχουν και αυξημένο συναπτικό βάρος σε σχέση με δυο νευρώνες που ενεργοποιούνται ασύγχρονα.

Για την τροποποίηση των συναπτικών βαρών σε περίπτωση που τα αποτελέσματα δεν είναι επιθυμητά, εφαρμόζεται ο **κανόνας του Hebb**, με βάση τον οποίο

$$\Delta W_j = \eta x_i y_i$$

Τύπος 4.2.2 - Κανόνας του Hebb

όπου η ο ρυθμός εκμάθησης, Xi η είσοδος του νευρώνα, και Yi η απόκρισή του.

Η μεταβολή, δηλαδή, του συναπτικού βάρος μεταξύ δυο νευρώνων k και j κατά τη χρονική στιγμή n ισούται με το γινόμενο της εξόδου του k, της εισόδου του j, και του ρυθμού εκμάθησης η.

4.3. Ανταγωνιστική εκμάθηση

Η ανταγωνιστική εκμάθηση (competitive learning) είναι μια μορφή μη-επιβλεπόμενης (unsupervised) εκπαίδευσης νευρωνικών δικτύων στην οποία οι νευρώνες "ανταγωνίζονται" για να μπορέσουν να ενεργοποιηθούν. Σε ένα σύστημα που στηρίζεται στην ανταγωνιστική εκμάθηση, από ένα σύνολο νευρώνων που δέχονται ταυτόχρονα ένα ερέθισμα μόνο ένας μπορεί να χρησιμοποιηθεί ως νευρώνας εξόδου.

Η ανταγωνιστική εκμάθηση χαρακτηρίζεται από τρία θεμελιώδη στοιχεία:

1. Ένα σύνολο νευρώνων των οποίων οι προδιαγραφές είναι ίδιες, με μόνη διαφορά στα συναπτικά τους βάρη που αρχικοποιούνται τυχαία

2. Ένα όριο στη δύναμη και την επίδραση που μπορεί να έχει κάθε νευρώνας

3. Έναν μηχανισμό δια του οποίου οι νευρώνες μπορούν να ανταγωνιστούν για την ενεργοποίησή τους ως απόκριση σε ένα σύνολο δεδομένων, ώστε τελικά μόνο ένας νευρώνας από το σύνολο να ενεργοποιείται.

Ο νευρώνας που επικρατεί στον "ανταγωνισμό" αυτόν είναι εκείνος με τη μεγαλύτερη απόκριση για τα δοθέντα δεδομένα. Θεωρούντες ότι η έξοδος του συνόλου των νευρώνων ισούται με 1, θεωρούμε και ότι η απόκριση Vk ενός νευρώνα για

δεδομένα x πρέπει να είναι μεγαλύτερη από την απόκριση Vj των άλλων νευρώνων, δηλαδή

$$x = \begin{cases} 1 & V_k > V_j, j \neq k \\ 0 & otherwise \end{cases}$$

Τύπος 4.3.1 - Η απόκριση του νευρώνα στην ανταγωνιστική εκμάθηση

για κάθε νευρώνα j.

Το νέο συναπτικό βάρος Wkj[n+1] ισούται με το άθροισμα του υπάρχοντος Wkj[n] και της μεταβολής ΔWkj (βλ. τύπος 4.1.3), όπου η μεταβολή συνάγεται από τη σχέση

$$\Delta W_{kj} = \begin{cases} \eta(X_j - W_{kj}) & if \ \ k \ \ wins \\ 0 & otherwise \end{cases}$$

Τύπος 4.3.2 - Η μεταβολή συναπτικού βάρους στην ανταγωνιστική εκμάθηση

όπου η ο ρυθμός εκμάθησης και Xj το σήμα από τον νευρώνα X. Άρα συνολικά το νέο βάρος ισούται με

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + u_i \, \eta \, (\mathbf{x} - \mathbf{w}_i^{old}) = \mathbf{w}_i^{old} + u_i \, \triangle \mathbf{w}_i = \begin{cases} (1-\eta)\mathbf{w}_k^{old} + \eta \mathbf{x} & u_i = 1 \\ \mathbf{w}_i^{old} & u_i = 0 \end{cases}$$

Τύπος 4.3.3 - Αναλυτική περιγραφή μεταβολής συναπτικού βάρους στην ανταγωνιστική εκμάθηση

ήτοι, μένει ως έχει αν το Ui (αυτό που στον τύπο 4.3.1. ονομάζουμε X) ισούται με μηδέν, ειδάλλως ισούται με το άθροισμα του προηγούμενου συναπτικού βάρους WKold, του γινομένου του WKold με τον ρυθμό εκμάθησης, και του γινομενου του ρυθμού εκμάθησης με την είσοδο X -η οποία είναι το σήμα από κάποιον νευρώνα j.

Για τα βάρη κάθε μεμονωμένου νευρώνα ισχύει

$$\sum_{j=1}^{n} w_{ij} = 1$$

Τύπος 4.3.4 - Τα βάρη μεμονωμένων νευρώνων σε ανταγωνιστικά δίκτυα

δηλαδή το σύνολο των συναπτικών βαρών στις διασυνδέσεις με j νευρώνες για κάθε

νευρώνα i, ισούται με τη μονάδα.

Αυτό που συμβαίνει στις περιπτώσεις που έχουμε ανταγωνιστική εκμάθηση μπορεί να περιγραφεί με τρία βήματα:

1. Δημιουργία ενός νευρωνικού δικτύου με τυχαία βάρη στις συνάψεις μεταξύ των νευρώνων.

2. Για είσοδο X, "νικητής" του ανταγωνισμού θεωρείται ο νευρώνας k με την υψηλότερη απόκριση.

3. Οι συνδέσεις προς τον j αναπροσαρμόζονται ούτως ώστε εκείνοι που του έδωσαν ισχυρότερο σήμα να έχουν ακόμα μεγαλύτερο βάρος, ενώ εκείνοι των λιγότερο ισχυρών σημάτων μειώνονται.

4.4. Ειδικοί αλγόριθμοι εκπαίδευσης τεχνητών νευρωνικών δικτύων

4.4.1. Η μέθοδος gradient-descent

Η μέθοδος gradient-descent, γνωστή και ως steepest descent (χωρίς να είναι ίδια με την ομώνυμη μέθοδο προσέγγισης ολοκληρωμάτων), βασίζεται στην οπισθοδιάδοση σφάλματος και αποτελεί την πιο απλή μέθοδο υπολογισμού των συναπτικών βαρών w, και βασίζεται σε μια λογική που θυμίζει τις μεθόδους διαταραχών -ήτοι, προσεγγιστικός υπολογισμός της λύσης και μεταβολή ("διαταραχή") των παραμέτρων αν το αποτέλεσμα δεν συμφωνεί με αυτό που πραγματικά χρειάζεται.

Παραδείγματος χάριν και πολύ απλουστευμένα, αν πρέπει να συνάγουμε μια σχέση του τύπου Y=aX+b για μη-γραμμικά δεδομένα χρησιμοποιώντας τη μέθοδο gradient-descent θα πρέπει να αποδώσουμε αρχικά τυχαίες τιμές στο a και στο b, και να ελέγξουμε την απόκλιση των τιμών της σχέσης Y με τις πραγματικές. Κατόπιν, θα πρέπει να μεταβάλλουμε τις τιμές a και b ωσπου η απόκλιση να γίνει όσο το δυνατόν μικρότερη. Η μεταβολή εν προκειμένω υπολογίζεται με τη χρήση της μερικής παραγώγου της απόκλισης ως προς το a και το b αντίστοιχα.

Γενικά, για τη συγκεκριμένη μέθοδο ισχύει

$$x_{i+1} = x_i - \alpha g_i$$

Τύπος 4.4.1 - Μεταβολή συναπτικού βάρους με gradient descent

όπου a ο ρυθμός εκμάθησης (learning rate) και gi η μερική παράγωγος του σφάλματος ως προς τη παρούσα τιμή.

Στη περίπτωση των συναπτικών βαρών, για να υπολογιστεί η μεταβολή τους είναι απαραίτητο να εξάγουμε έναν πίνακα g ίσο με

$$g = \frac{\partial E(x,w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \quad \cdots \quad \frac{\partial E}{\partial w_N} \right]^T$$

Τύπος 4.4.2 - Περιγραφή του πίνακα g

όπου x η είσοδος, w το παρόν συναπτικό βάρος για έναν νευρώνα σε νευρωνικό δίκτυο με νευρώνες από 1 ως N, E το sum square error των σφαλμάτων, ίσο με

$$E(x,w) = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2$$

Τύπος 4.4.3 - Περιγραφή του sum square error

όπου M ο αριθμός των εξόδων, P ο αριθμός των patterns εκπαίδευσης, και σφάλμα e ίσο με

$$e_{p,m} = d_{p,m} - o_{p,m}$$

Τύπος 4.4.4 - Περιγραφή του σφάλματος

όπου d η επιθυμητή έξοδος και o η πραγματική.

4.4.2. Η νευτώνεια μέθοδος

Στην αρκετά πιο σύνθετη νευτώνεια μέθοδο, η μεταβολή του συναπτικού βάρους ισούται με

$$W_{i+1} = x_i + H_i^{-1} g_i$$

Τύπος 4.4.5 - Μεταβολή συναπτικού βάρους στη νευτώνεια μέθοδο

όπου Η ο Hessian πίνακας, αποτελούμενος από τις μερικές παραγώγους δεύτερης τάξης για το σφάλμα Ε ως προς τις καταστάσεις W1 ως WN επί W1 ως WN, δηλαδή

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix}$$

Τύπος 4.4.6 - Πίνακας Hessian στη νευτώνεια μέθοδο

και g ο πίνακας καταστάσεων από -g1 ως -gN, που εξάγεται από τον Hessian πίνακα πολλαπλασιασμένο με τις μεταβολές των βαρών από W1 ως WN, ήτοι

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \cdots \\ -g_N \end{bmatrix} = \begin{bmatrix} -\frac{\partial E}{\partial w_1} \\ -\frac{\partial E}{\partial w_2} \\ \cdots \\ -\frac{\partial E}{\partial w_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \cdots \\ \Delta w_N \end{bmatrix}$$

Τύπος 4.4.7 - Πίνακας g στη νευτώνεια μέθοδο

Πρόκειται περί μιας αρκετά σύνθετης μεθόδου και είναι πρακτικά ανέφικτο να υπολογιστούν μέσω αυτής τα συναπτικά βάρη σύνθετου νευρωνικού δικτύου σε εύλογο χρονικό διάστημα χωρίς τη χρήση μεγάλης υπολογιστικής ισχύος. Είναι, ωστόσο, αρκετά πιο αποτελεσματική από την gradient-descent, και η **μέθοδος quasi-Newton** που θα υλοποιήσουμε, μεταξύ άλλων, στην εφαρμογή αποτελεί μια παραλλαγή της με μικρότερη απαίτηση σε υπολογιστική ισχύ (χρησιμοποιείται ένας προσεγγιστικός υπολογισμός του Hessian πίνακα)

4.4.3. Η μέθοδος Levenberg-Marquardt

Στη μέθοδο Levenberg-Marquardt χρησιμοποιείται ένας πίνακας Hessian της

μορφής

$$H = J^T J$$

Τύπος 4.4.8 - Πίνακας Hessian με Levenberg-Marquardt

, όπου J ένας Jacobian πίνακας αποτελούμενος από τις μερικές παραγώγους των σφαλμάτων ως προς τους νευρώνες και τις καταστάσεις τους.

$$J_{i,j} f(w) = \frac{de_i}{dw_j}$$

Τύπος 4.4.9 - Πίνακας Jacobian με Levenberg-Marquardt

όπου i ένας αριθμός από 1 έως τον συνολικό αριθμό των δεδομένων και j από 1 εως τον αριθμό των νευρώνων.

Η μεταβολή του συναπτικού βάρους ισούται με

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e$$

Τύπος 4.4.10 - Μεταβολή συναπτικού βάρους με Levenberg-Marquardt

όπου μ ένας θετικός αριθμός (combination coefficient) και I ένας identity matrix, όπου παρουσιάζει μια διαγώνιο από άσσους στο κέντρο ενώ οι άλλες τιμές είναι μηδέν

Αποτελεί μια μέθοδο που θεωρείται αρκετά γρήγορη παρά τη μαθηματική της πολυπλοκότητα.

Μια παραλλαγή της είναι η **Levenberg-Markquardt με Bayesian κανονικοποίηση**, η οποία έχει μεγαλύτερη απαίτηση σε υπολογιστική ισχύ.

4.4.4. Μέθοδος ευπροσάρμοστης οπισθοδιάδοσης

Η μέθοδος ευπροσάρμοστης οπισθοδιάδοσης (resilient backpropagation) θεωρείται ως μια από τις πιο γρήγορες μεθόδους εκπαίδευσης νευρωνικών δικτύων.

Η μαθηματική περιγραφή της μεταβολής των συναπτικών βαρών έχει ως εξής:

$$\Delta_{ij}(t) = \begin{cases} \eta^{+} \cdot \Delta_{ij}(t-1) & , \text{ if } s_{ij} > 0 \\ \eta^{-} \cdot \Delta_{ij}(t-1) & , \text{ if } s_{ij} < 0 \\ \Delta_{ij}(t-1) & , \text{ otherwise} \end{cases}$$

Τύπος 4.4.11 - Μεταβολή συναπτικού βάρους σε resilient backpropagation

με το Sij να έχει ως εξής:

$$s_{ij} = \frac{\partial E}{\partial w_{ij}}(t-1) \cdot \frac{\partial E}{\partial w_{ij}}(t) \quad .$$

Τύπος 4.4.12 - Περιγραφή παραμέτρου Sij

Για κάθε επανάληψη, δηλαδή, ελέγχεται το κατά πόσον οι μερικές παράγωγοι των σφαλμάτων της συγκεκριμένης επανάληψης και της προηγούμενης είναι μεγαλύτεροι ή μικρότεροι του μηδενός και μεταβάλλεται το συναπτικό βάρος ανάλογα. Τυπικές τιμές για το η- και η+ είναι 0.5 και 1.2 αντίστοιχα.

Το αν η μεταβολή είναι θετική ή αρνητική θα εξαρτηθεί από τη σχέση της μερικής παραγώγου του σφάλματος και του συναπτικού βάρους της προηγούμενης επανάληψης με το μηδέν.

5. ΑΝΑΓΝΩΡΙΣΗ ΧΑΡΑΚΤΗΡΩΝ ΜΕ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΜΕΣΩ MATLAB

5.1. Γενικά - Αναγνώριση χαρακτήρων από εικόνες στο matlab

Τώρα που έχουμε αναλύσει κάποιες θεμελιώδεις έννοιες των νευρωνικών δικτύων, θα προβούμε στην παρουσίαση μιας εφαρμογής που επιτρέπει την υλοποίηση νευρωνικών δικτύων ποικιλλων προδιαγραφών -ήτοι, με διαφορετική αρχιτεκτονική, αριθμό νευρώνων, αριθμό επιπέδων, μέθοδο εκπαίδευσης- με σκοπό την αναγνώριση **χαρακτήρων** (βλ. κεφ. 1.3.2) από εικόνες, χειρόγραφων ή μη. Η εφαρμογή αυτή θα μας επιτρέψει να συγκρίνουμε την αποτελεσματικότητα των διαφόρων κατηγοριών δικτύων και να εξάγουμε τα συμπεράσματά μας.

Η εφαρμογή περιλαμβάνει μια διεπαφή (interface) δια της οποίας καθορίζονται οι παράμετροι του νευρωνικού δικτύου, τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευσή του, καθώς και οι χαρακτήρες προς αναγνώριση. Χρησιμοποιεί δυο ξεχωριστές μεθόδους αναγνώρισης χαρακτήρων, εκ των οποίων η μία στηρίζεται σε μια ανάμειξη του συμβατικού OCR με νευρωνικά δίκτυα και χρησιμοποιείται εύκολα σε πρακτικές εφαρμογές. Σε πιο σύγχρονες ή μελλοντικές εκδοχές του matlab η δεύτερη αυτή εκδοχή θα είναι σημαντικά πρακτικότερη από τη σημερινή, που ενίοτε αδυνατεί να εντοπίσει τους χαρακτήρες σωστά.

Η εφαρμογή -τόσο η κύρια όσο και η πειραματική- υλοποιήθηκε στο **matlab R2015a**, που παρέχει μια **εργαλειοθήκη νευρωνικών δικτύων** (Neural Network Toolbox) και, αν και υπάρχουν πιο εξειδικευμένα λογισμικά ανάπτυξης δικτύων τέτοιου είδους, η εργαλειοθήκη αυτή χρησιμοποιείται ευρύτατα για την απλότητά της ακόμα και από ειδικευμένους επιστήμονες για πειραματικούς σκοπούς.

Στο matlab υπάρχουν δυο μέθοδοι αναγνώρισης χαρακτήρων. Ο πρώτος εξ'αυτών στηρίζεται στην εντολή OCR εφαρμοζόμενη σε εικόνες, η οποία αποτελεί μια έτοιμη συνάρτηση ανάγνωσης χαρακτήρων με ποσοστό επιτυχίας έως 95%, για καθαρογραμμένα κείμενα. Η OCR, που στα πιο πολλά κείμενα που δοκιμάσαμε έχει σχετικά περιορισμένη επιτυχία, χρησιμοποιεί συμβατικούς αλγορίθμους και όχι νευρωνικό δίκτυο όπως η εφαρμογή μας. Επιστρέφει έναν πίνακα με το κείμενο της

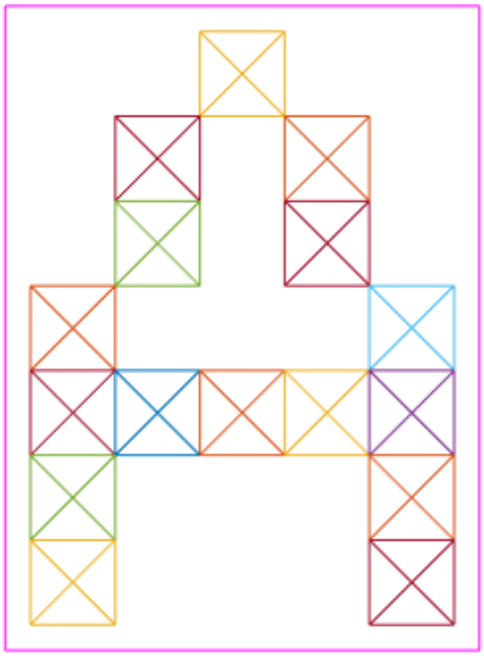
εικόνας, καθώς και υποπίνακες με πληροφορίες όπως οι συντεταγμένες των μεμονωμένων χαρακτήρων, καθώς και των λέξεων που αποτελούν το κείμενο.

Αυτός ο τρόπος, που αναγνωρίζει και τα κενά, θα χρησιμοποιηθεί στην εναλλακτική μέθοδο αναγνώρισης, και συγκεκριμένα θα χρησιμοποιηθούν κατά κόρον οι συντεταγμένες των χαρακτήρων όπως εντοπίζονται από την εντολή αυτή. Όπως θα δούμε όμως, και αυτό ακόμα έχει περιορισμένη επιτυχία, γιατί ορισμένοι χαρακτήρες μπορεί να φαίνονται σαν ένας -εξού και το ότι δεν χρησιμοποιήσαμε τη μέθοδο αυτή για την αποκοπή των χαρακτήρων από την εικόνα και την εκπαίδευση των νευρωνικών δικτύων μ'αυτούς.

Η δεύτερη μέθοδος, που αφορά πειραματισμούς σε νευρωνικά δίκτυα, στηρίζεται στην εντολή

```
[X,T] = prprob;
```

για τη δημιουργία δυο πινάκων που θα χρησιμοποιηθούν ως είσοδοι σε ένα νευρωνικό δίκτυο. Ο πρώτος, πίνακας X, είναι ένας πίνακας διαστάσεων 35x26 -ο αριθμός των χαρακτήρων του λατινικού αλφαβητου, δηλαδή, επί 35. Ο αριθμός 35 προκύπτει από την απεικόνιση των χαρακτήρων μέσω ενός πίνακα διαστάσεων 5x7, στον οποίον κάθε τιμή αφορά στη θέση ενός τετραγώνου σε μια εικόνα 35 στοιχείων, στα οποία μπορούν να απεικονιστούν οι χαρακτήρες του αλφαβητου. Το γράμμα Α, για παράδειγμα, απεικονίζεται με την εντολή plotchar(X(1:35)); ως:



Εικόνα 5.1.1. - Χαρακτήρας σε πίνακα matlab 5x7 με plotchar

Κάθε τετράγωνο αντιστοιχεί στον άσσο κάποιας θέσης στον πίνακα Χ.

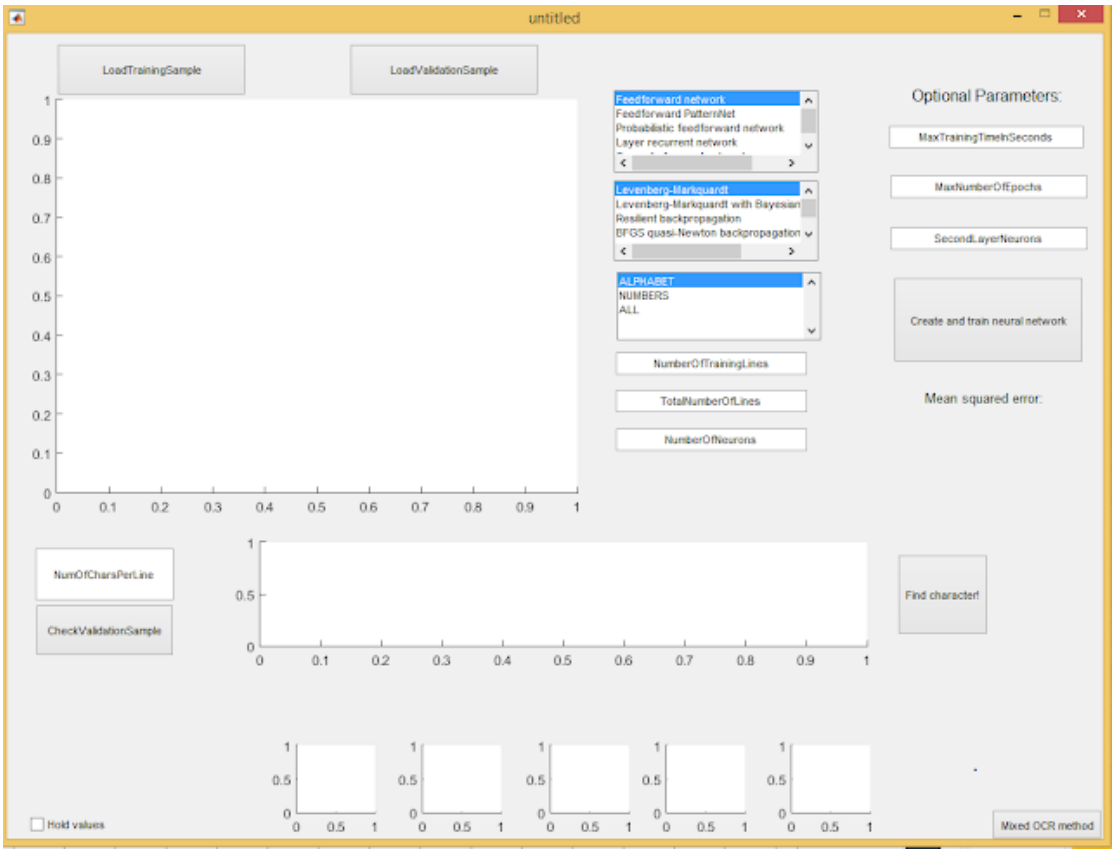
Ο πίνακας Τ συνάγεται από την εντολή repmat(eye((26)),[1]);, όστις δημιουργεί με τη συνάρτηση eye έναν πίνακα 26 γραμμών και στηλών με έναν άσσο και 25 μηδενικά ο καθένας, με τον άσσο κάθε φορά σε διαφορετική θέση της γραμμής/στήλης. Η repmat καθορίζει το πόσες φορές εμφανίζεται ο πίνακας αυτός, ανάλογα με τον αριθμό των αλφαβητων προς αναγνώριση. Αν επρόκειτο να εκπαιδεύσουμε ένα Ν.Δ. με 8 αλφάβητα, η εντολή θα ήταν repmat(eye((26)),[1 (8)]), όπου 26 ο αριθμός των χαρακτήρων και 8 ο αριθμός των αλφαβητων.

Η μέθοδος που θα χρησιμοποιήσουμε χωρίζει τις εικόνες με τα κείμενα σε υποεικόνες, έκαστη εκ των οποίων απεικονίζει ένα μεμονωμένο γράμμα (με την προϋπόθεση, πάντα, να είναι καθαρογραμμένο το κείμενο και χωρίς να ενώνονται τα γράμματα μεταξύ τους -περίπτωση που ούτε τα πιο προηγμένα συστήματα μπορούν προς το παρόν να αντιμετωπίσουν με σεβαστά ποσοστά επιτυχίας), και κατόπιν μετατρέπονται σε πίνακες 5x7 ή 35x1 που μπορούν να απεικονιστούν όπως

παραπάνω.

5.2. Το interface - Οδηγίες χρήσης

Η διεπαφή (interface) της εφαρμογής έχει ως εξής:



Εικόνα 5.1.2. - Το interface της εφαρμογής

Συνολικά περιέχει εφτά άξονες εμφάνισης εικόνων, οι πέντε εκ των οποίων χρησιμοποιούνται για την εμφάνιση μεμονωμένων γραμμάτων στα διάφορα στάδια μετατροπής τους σε εικόνα αναγνωρίσιμη από το λογισμικό, ενώ οι άλλες δυο για την εικόνα εκπαίδευσης (training sample) και την εικόνα δείγματος προς έλεγχο (validation sample).

Το πρώτο βήμα που πρέπει να ακολουθήσουμε για τη δημιουργία και εκπαίδευση του νευρωνικού δικτύου είναι η επιλογή εικόνας εκπαίδευσης, μέσω του κουμπιού "LoadTrainingSample" πάνω αριστερά. Η εικόνα που θα επιλέξουμε θα εμφανιστεί στον άξονα εικόνων που βρίσκεται ακριβώς από κάτω, και είναι η εικόνα που θα αξιοποιηθεί από το νευρωνικό δίκτυο για να αυτοεκπαιδευτεί.

Στη συνέχεια πρέπει να καθορίσουμε, μέσω των listboxes που βρίσκονται δεξιά του άξονα όπου εμφανίζεται το training sample, το είδος του νευρωνικού δικτύου, τη μέθοδο εκπαίδευσης, και το αν η εικόνα περιλαμβάνει αλφάβητα (λατινικοί χαρακτήρες), αριθμούς (0-9), ή τον συνδυασμό τους (με το αλφάβητο να προηγείται).

ΠΡΟΣΟΧΗ: Λόγω του ότι ούτε ο υπολογιστής, αλλά ούτε και το προς δημιουργία νευρωνικό δίκτυο μπορεί να ξέρει τι απεικονίζεται στο training sample, πρέπει να είμαστε ιδιαίτερα προσεκτικοί με την τήρηση των προδιαγραφών αυτών στις εικόνες - το αλφάβητο να είναι λατινικό με τη συμβατική σειρά από A έως Z, ενώ οι αριθμοί από 0 εως 9, και σε τυχόν συνδυασμό τους οι αριθμοί να έρχονται μετά το αλφάβητο. Να σημειωθεί επίσης ότι οι εικόνες εκπαίδευσης και ελέγχου που χρησιμοποιήθηκαν στα πλαίσια της εργασίας, ήταν bmp 24bit και δεν είναι ξεκάθαρο το κατά πόσον δουλεύει η εφαρμογή με εικόνες άλλων προδιαγραφών. Συνιστάται η μετατροπή τους στη προαναφερθείσα μορφή. Πρέπει επίσης να είναι καθαρογραμμένο και να υπάρχουν κενά ανάμεσα στα γράμματα.

Το προκαθορισμένο είδος δικτύου ειναι ένα απλό εμπρόσθιας τροφοδότησης, και η προκαθορισμένη μέθοδος εκπαίδευσης η Levenberg-Markquardt, λόγω του ότι είναι τα πιο συνηθισμένα.. Αξίζει να σημειωθεί ότι για το σκοπό της παρούσης εφαρμογής ως καλύτερο είδος δικτύου θεωρείται το Feedforward PatternNet -ένα δίκτυο εμπρόσθιας τροφοδότησης κατάλληλο για την αντιστοίχηση patterns σε πίνακα T (βλ. ενότητα 4.1.) στο matlab-, ωστόσο χρησιμοποιούμε ως προκαθορισμένο το απλής εμπρόσθιας τροφοδότησης λόγω του ότι χρησιμοποιείται περισσότερο. Οι μετρήσεις του επόμενου κεφαλαίου θα αναδείξουν ενδεχομένως και άλλες αποτελεσματικές τοπολογίες και μεθόδους εκπαίδευσης.

Παρέχεται η δυνατότητα καθορισμού των γραμμών της εικόνας που θα χρησιμοποιηθούν για την εκπαίδευση, καθώς και ο συνολικός της εικόνας. Φερ'ειπείν, αν μια εικόνα έχει δέκα αλφάβητα και θέλουμε να χρησιμοποιήσουμε μόνο τα οχτώ για την εκπαίδευση, οι τιμές του NumberOfTrainingLines και TotalNumberOfLines πρέπει να είναι οχτώ και δέκα αντίστοιχα.

ΣΗΜΑΝΤΙΚΟ: η εφαρμογή θεωρεί αυτοδικαίως ότι η εικόνα απαρτίζεται από γραμμές χαρακτήρων A-Z ή 0-9 για τους αριθμούς και ότι κάθε δείγμα έχει δική του γραμμή. Υπάρχει ο συνδυασμός και των δυο, και τα γράμματα στις εικόνες πρέπει

να είναι A-Z, 0-9, και τα σημεία στίξης ? ! . με αυτή τη σειρά. Διαφορετικές σειρές καθιστούν το πρόγραμμα πρακτικά άχρηστο, μιας και η αντιστοιχία των αναγνωρισθέντων χαρακτήρων θεωρεί πως η σειρά είναι αυτή.

Οι προαναφερθείσες παράμετροι είναι απολύτως απαραίτητες για τη δημιουργία του δικτύου, ενώ στο πάνω δεξιά μέρος του interface υπάρχουν και ορισμένες προαιρετικές -όπως ο μέγιστος χρόνος εκπαίδευσης σε δευτερόλεπτα (προκαθορισμένη τιμή το άπειρο), και ο μέγιστος αριθμός εποχών (iterations) εκπαίδευσης (προκαθορισμένη τιμή 100.000). Υπάρχει επίσης η δυνατότητα προσθήκης και ενός δεύτερου hidden layer στο νευρωνικό δίκτυο, και αν ο αριθμός των νευρώνων του δεν καθοριστεί λαμβάνεται υπόψιν μόνο το NumberOfNeurons. Να σημειωθεί ότι υπάρχουν δίκτυα (π.χ. πιθανοκρατικό) ο αριθμός των νευρώνων των οποίων καθορίζεται από το set εκπαίδευσης. Αν στη λίστα επιλέξουμε "probabilistic feedforward network" τα πεδία καθορισμού αριθμού νευρώνων θα απενεργοποιηθούν.

Όταν ολοκληρωθεί η δημιουργία και εκπαίδευση του δικτύου, οι τιμές των κουτιών αντικαθίστανται από τις προκαθορισμένες (δηλ. τις περιγραφές τους), εκτός αν είναι επιλεγμένο το "Hold values" κάτω αριστερά, ο σκοπός του οποίου είναι να διευκολύνει τον χρήστη εφόσον θέλει να πειραματιστεί με νευρωνικά δίκτυα αλλάζοντας κάθε φορά μια μόνο παράμετρο, ούτως ώστε να μη χρειαστεί να συμπληρώσει όλα τα πεδία εκ νέου.

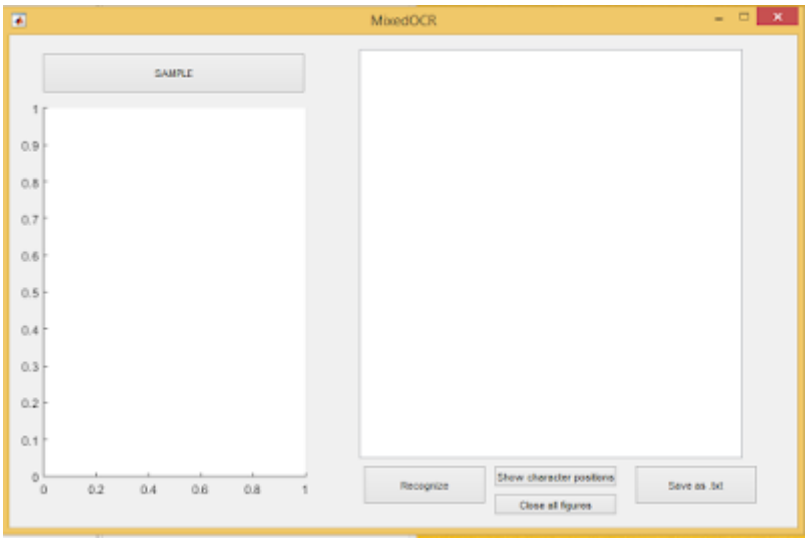
Μετά τη δημιουργία και εκπαίδευση του νευρωνικού δικτύου με το σχετικό κουμπί, επιλέγουμε ValidationSample το οποίο εμφανίζεται ακριβώς από κάτω. Το κουμπί "CheckValidationSample", το πάτημα του οποίου έπεται της συμπλήρωσης του NumOfCharsPerLine που περιγράφει μας επιτρέπει να δούμε τι αναγιγνώσκει το δίκτυο όταν του δίνουμε ως είσοδο το ValidationSample.

ΠΡΟΣΟΧΗ: Η αναγνώριση του ValidationSample γίνεται προβληματική για περισσότερες από μια γραμμές, εφόσον αυτές εμπεριέχουν αριθμό χαρακτήρων διάφορο του NumOfCharsPerLine, ενώ ακόμα και τότε δεν θα εμφανίζονταν στο κάτω μέρος το σύνολο του κειμένου που αναγνώστηκε από το δίκτυο. Για πρακτική αναγνώριση κειμένου λοιπόν χρησιμοποιείται η Mixed OCR Method (βλ. και ενότητα

4.4.) που θα περιγραφεί παρακάτω.

Το κουμπί "Find character!" μας δείχνει. εφόσον επιλέξουμε έναν χαρακτήρα του ValidationSample και κάνουμε δεξί κλικ και "crop", τα διαδοχικά στάδια επεξεργασίας ώσπου να φτάσει στη μορφή πίνακα 5x7. Οι εικόνες των διαδοχικών σταδίων εμφανίζονται στους πέντε άξονες ακριβώς κάτω από τη θέση του κειμένου που αναγνωρίστηκε.

Το πάτημα του "Mixed OCR method" οδηγεί στην ανάδυση ενός νέου παραθύρου, μέσω του οποίου το νευρωνικό δίκτυο χρησιμοποιείται για πρακτική εφαρμογή ανάγνωσης κειμένου. Η μέθοδος που χρησιμοποιείται σε αυτό περιγράφεται στην ενότητα 4.4.. Ο χρήστης στο αναδυθέν παράθυρο πρέπει πρώτα να επιλέξει την εικόνα προς ανάγνωση πατώντας το κουμπί "sample" στο πάνω αριστερό μέρος του νέου παραθύρου.



Εικόνα 5.2.1. - Το παράθυρο MixedOCR

Η εικόνα θα εμφανιστεί τότε στον άξονα που βρίσκεται ακριβώς από κάτω, οπότε και μπορεί να γίνει ανάγνωση του κειμένου με το κουμπί "recognize". Το κείμενο θα εμφανιστεί στο textbox του δεξιού μέρους του παραθύρου, οπού ο χρήστης δύναται να διορθώσει τυχόν λάθη χειροκίνητα και να το αποθηκεύσει στο σύστημα με το κουμπί "save as .txt". Λόγω της εγγενούς φύσης του αλγορίθμου του MixedOCR, μπορούμε να ισχυριστούμε πως τα λάθη ενός καλά εκπαιδευμένου δικτύου με τη χρήση του οφείλονται είτε σε κακογραμμένο/μη-καθαρό κείμενο (θυμηθείτε ότι ακόμα και τα ισχυρότερα νευρωνικά δίκτυα προς το παρόν δυσκολεύονται στην ανάγνωσή

τους), είτε σε κάποιο λάθος στην αναγνώριση των θέσεων των χαρακτήρων από το OCR.

Για τη δεύτερη περίπτωση, ο χρήστης δύναται να δει τις θέσεις των χαρακτήρων όπως εντοπίζονται από το σύστημα προτού αναγνωριστούν από το δίκτυο, με το πάτημα του κουμπιού "show character positions" που θα οδηγήσει στην ανάδυση παραθύρων ισάριθμων με τις λέξεις της εικόνας, όπου οι χαρακτήρες θα περικλείονται από τετράγωνα. Ο χρήστης μπορεί να τα κλείσει όλα μαζί με το κουμπί "close all figures".

ΠΡΟΣΟΧΗ: Λόγω της εγγενούς φύσης του matlab και του ότι πρέπει να φορτωθούν ορισμένες βιβλιοθήκες για την εν λόγω λειτουργία, ο χρήστης να μην αναμένει απόκριση των δυο αυτών πλήκτρων -και ιδιαίτερα του "Close all figures"- σε κλάσματα του δευτερολέπτου. Απεναντίας, μπορεί να χρειαστεί αρκετά δευτερόλεπτα η ολοκλήρωση της εντολής. Να σημειωθεί επίσης ότι δεν προβλέπεται το σενάριο του να πατηθεί κάποιο από τα υπό του textbox ευρισκόμενα κουμπιά χωρίς να έχει φορτωθεί εικόνα μέσω του "Sample".

Παρέχεται επίσης η δυνατότητα αποθήκευσης του κειμένου με το κουμπί "Save as .txt", και ο χρήστης πρέπει απλώς να δώσει ένα όνομα στο αρχείο όταν του ζητηθεί.

5.3. Περιγραφή του κώδικα της εφαρμογής

Η εφαρμογή αποτελείται από έξι κομμάτια κώδικα που παρατίθενται αναλυτικα στο παράρτημα. Το πρώτο εξ'αυτών ονομάζεται interface.m και αποτελεί τον κώδικα της διεπαφής της εφαρμογής και αρχίζει να τρέχει με το που ανοίγει. Τα άλλα, που καλούνται μέσω πατημάτων κουμπιών της διεπαφής, είναι τα, function1.m, functionZ.m, CreateNetwork.m, MixedOCR.m , και το ToAlphabet.m.

Η πρώτη από αυτές, function1.m, βασισμένη σε εντολές από την επίσημη ιστοσελίδα του matlab και ιδιαίτερα στους αλγορίθμους των παραπομπών [13] εως [15] του κεφαλαίου 6.2, εντοπίζει τα γράμματα στις εικόνες αντιμετωπίζοντάς τα ως αντικείμενα και αφότου τα αποκόψει από αυτήν, τα μετατρέπει σε πίνακες των 35 θέσεων (5x7, βλ. κεφάλαιο 4.1) και τα επιστρέφει ως πίνακα 35xN, όπου N ο αριθμός των χαρακτήρων.

Ο κώδικας CreateNetwork.m καταπιάνεται με την δημιουργία, τη ρύθμιση παραμέτρων, και την εκπαίδευση των νευρωνικών δικτύων. Δέχεται τις παραμέτρους τους και τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση ως ορίσματα, και επιστρέφει το εκπαιδευμένο δίκτυο στην untitled.m για περαιτέρω χρήση.

Η MixedOCR αφορά το παράθυρο MixedOCR.fig που καταπιάνεται με τη μεικτή μέθοδο αναγνώρισης κειμένου, με ταυτόχρονη χρήση συμβατικού OCR και νευρωνικού δικτύου –με το πρώτο να αξιοποιείται για τον εντοπισμό των χαρακτήρων.

Η συνάρτηση ToAlphabet.m καταπιάνεται με την αντιστοίχηση της εξόδου του νευρωνικού δικτύου σε χαρακτήρες, ανάλογα με το αν είναι γράμματα, αριθμοί, ή και τα δυο.

Οι εν λόγω κώδικες επισυνάπτονται στο παράρτημα της παρούσης πτυχιακής, αλλά παρατίθενται και στα αρχεία της.

5.4. Δυο εναλλακτικές μέθοδοι εντοπισμού των χαρακτήρων

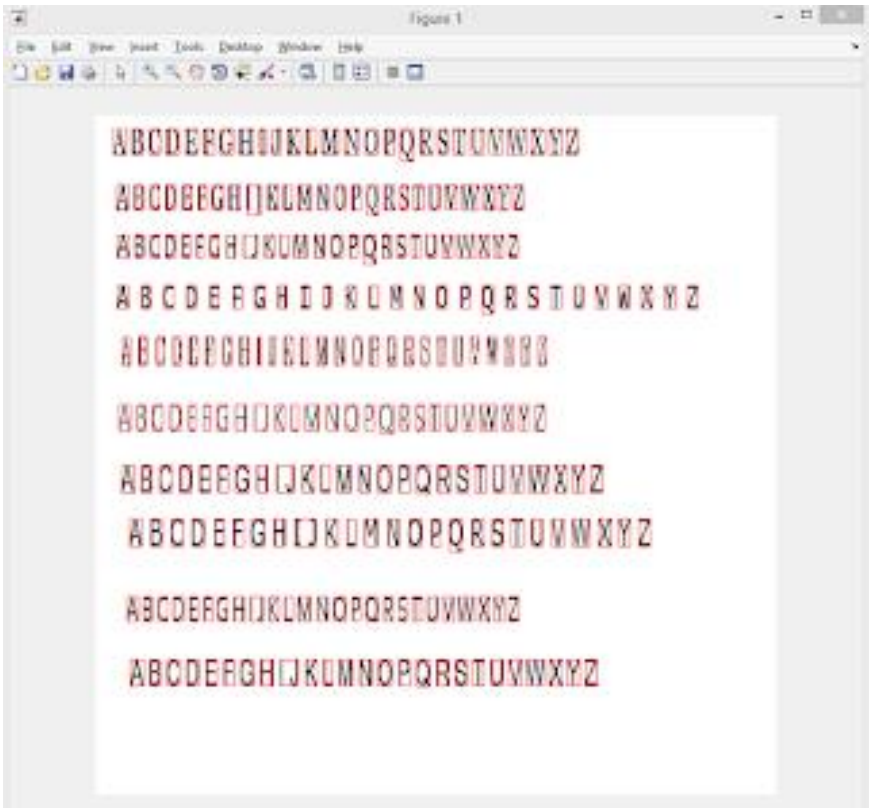
Το καθ'αυτό interface αξιοποιεί τη συνάρτηση function1.m, η οποία περνάει την εικόνα από διάφορα στάδια επεξεργασίας και, αφότου τη τροποποιήσει έτσι ώστε τα γράμματα να φαίνονται ως συμπαγή αντικείμενα, προβαίνει στον εντοπισμό των χαρακτήρων τους με χρήση της συνάρτησης regionprops. Η ταξινόμηση από τα αριστερά προς τα δεξιά προϋποθέτει τον καθορισμό των χαρακτήρων ανά γραμμή στην εικόνα, κάτι που καθιστά τη συνάρτηση παντελώς ακατάλληλη ως εφαρμογή αναγνωσης κειμένου αντί μεμονωμένων χαρακτήρων, ενώ δεν προβλέπει αναγνώριση του κενού. Παρόλα αυτά για τους σκοπούς της παρούσης εργασίας ωστόσο, που ήταν η απλή αναγνώριση χαρακτήρων, μας κάλυψε, οπότε χρησιμοποιήσαμε αυτή στο κυρίως gui.

Η μέθοδος που αρχικά θέλαμε να χρησιμοποιήσουμε -και η οποία είναι βέβαιο ότι θα μπορεί να χρησιμοποιηθεί σε μελλοντικές εκδοχές του matlab- αξιοποιούσε τη δυνατότητα του προκαθορισμένου OCR του προγράμματος να εντοπίζει (θεωρητικά) σωστά τη θέση των λέξεων και των γραμμάτων.

Η αναγνώριση των χαρακτήρων μέσω αυτής γίνεται με χρήση συμβατικών

μεθόδων αντί νευρωνικών δικτύων, που έχει ως αποτέλεσμα να αναγνωρίζονται ορθά σε ποσοστό το πολύ 95%, αν οι χαρακτήρες είναι τυποποιημένοι, ενώ είναι ακατάλληλη για αναγνώριση χειρόγραφου κειμένου. Παρόλα αυτά η δυνατότητά του να εντοπίζει ορθά τις θέσεις των λέξεων και των χαρακτήρων ως συντεταγμένες ήταν, θεωρητικά, ικανοποιητική.

Θεωρητικά. Πρακτικά, όπως φαίνεται στις παρακάτω εικόνες, στη παρούσα εκδοχή του έχει μια μάλλον πεπερασμένη δυνατότητα ορθού εντοπισμού της θέσης των χαρακτήρων:



Εικόνα 5.4.1. - Θέσεις χαρακτήρων με συμβατικό OCR

Όπως φαίνεται, στην εικόνα που χρησιμοποιήθηκε για την εκπαίδευση των δικτύων, ορισμένα γράμματα (π.χ. το I και το J) αναγνωρίζονται σαν ένα ενώ είναι δυο ξεχωριστά. Έστω και αν κάτι τέτοιο δεν συμβαίνει πάντοτε, μέθοδος δεν χρησιμοποιείται κατά την εκπαίδευση γιατί αν γίνουν τότε του νευρωνικού δικτύου αυτό θα καταστεί πρακτικά άχρηστο. Θέλοντας όμως να θέσουμε τις βάσεις για μια πρακτική εφαρμογή ανάγνωσης κειμένου μέσω νευρωνικού δικτύου (π.χ. scanάρισμα χειρόγραφου κειμένου στον υπολογιστή και μετατροπή του σε word), θα

δώσουμε μια σύντομη περιγραφή του πώς δουλεύει η εναλλακτική μέθοδος ενώ θα παραθέσουμε και τους σχετικούς κώδικες.

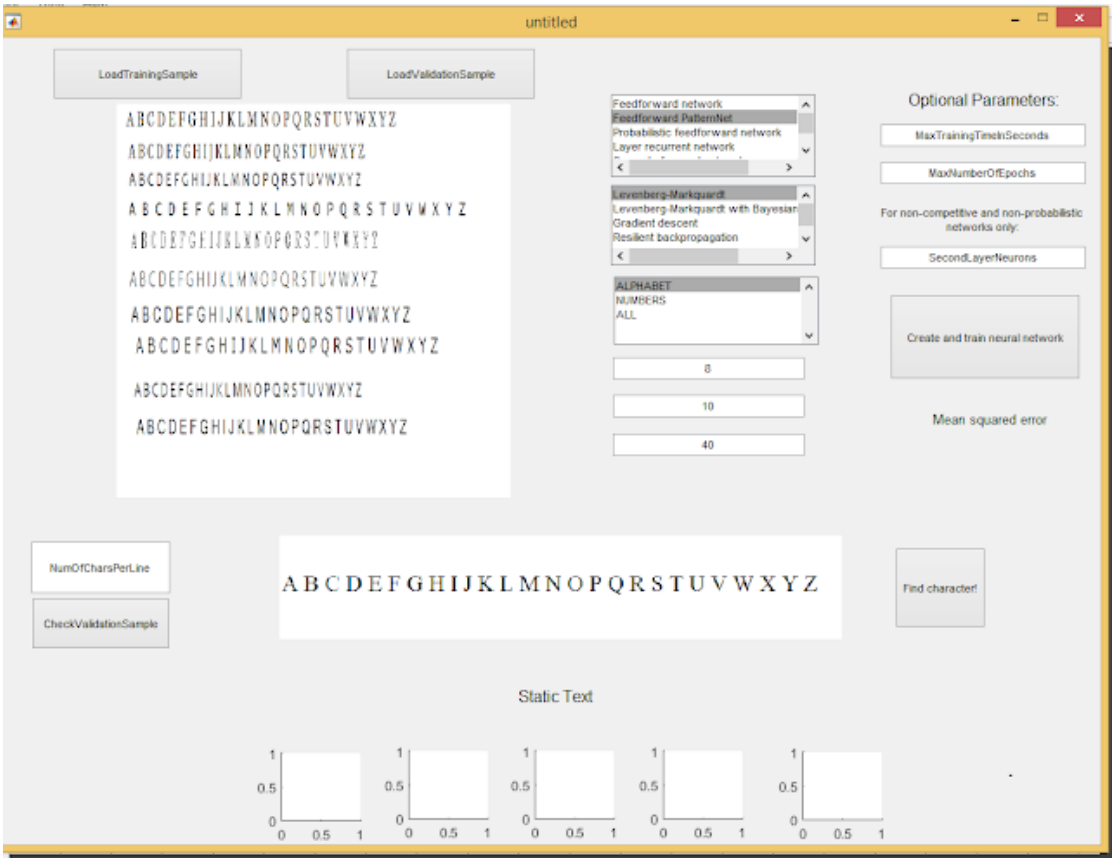
Το σύστημα με αυτή είναι ίδιο με αυτό που παρουσιάσαμε, με μια μονάχα διαφορά: δεν γίνεται η χρήση της regionprops για τον εντοπισμό των χαρακτήρων, ούτε υπάρχει ειδικός αλγόριθμος για τη ταξινόμηση από τα αριστερά. Αντί αυτών γίνεται η χρήση της OCR -όχι για την αναγνώριση χαρακτήρων, παρά για τον εντοπισμό της θέσης τους και τη χρήση των αντίστοιχων συντεταγμένων για crop. Η υπόλοιπη διαδικασία μένει ως έχει.

Η αναγνώριση γίνεται πρώτα σε επίπεδο λέξεων (για να μπουν κενά ανάμεσά τους αργότερα, χωρίς να ληφθούν υπόψιν τα περιττά που εισάγονται από το OCR στο τέλος τους [βλ. σχόλια κώδικα], αλλά και για τη περίπτωση που γίνουν σφάλματα στην αναγνώριση του αριθμού των χαρακτήρων ώστε αυτά να περιοριστούν μόνο στη συγκεκριμένη λέξη) και μετά σε επίπεδο χαρακτήρων σε κάθε λέξη.

Οι χαρακτήρες μετατρέπονται σε πίνακα 5x7 με την ίδια μεθοδολογία της μεθόδου στην οποία καταφύγαμε, και έπειτα σε έναν πίνακα out, όστις καταχωρείται σε πίνακα 1xN, όπου N ο αριθμός των λέξεων της εικόνας. Η θέση είναι αντίστοιχη της σειράς της λέξης (π.χ. (1,3) για τη τρίτη στη σειρά). Αυτός ο πίνακας έπειτα εισάγεται σε νευρωνικό δίκτυο το οποίο αναλαμβάνει την αναγνώριση, και επιστρέφει πίνακα πινάκων αποτελούμενων από χαρακτήρες. Το interface "χτίζει" λέξεις από αυτούς και βάζει τα κενά ανάμεσά τους.

5.5. Παράδειγμα χρήσης της εφαρμογής

Με τη χρήση των κουμπιών LoadTrainingSample και LoadValidationSample της εφαρμογής, επιλέγουμε τις εικόνες εκπαίδευσης και δοκιμής αντίστοιχα, ενώ συμπληρώνουμε και τις προδιαγραφές του προς εκπαίδευση νευρωνικού δικτύου.



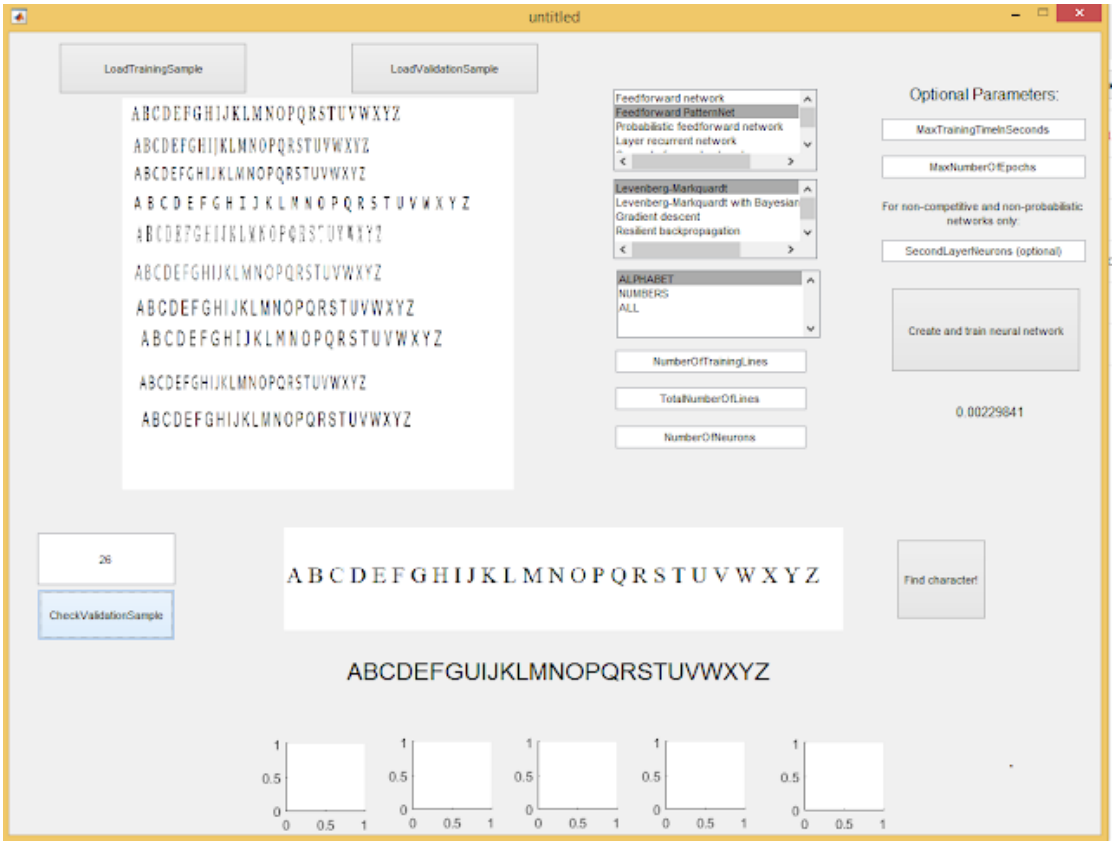
Εικόνα 5.5.1. - Παράδειγμα χρήσης της εφαρμογής

Με το πάτημα του πλήκτρου Create and train neural network δημιουργείται ένα εμπροσθοτοτροφοδοτούμενο pattern-net, που αποτελεί μια κατηγορία νευρωνικών δικτύων ιδανική για εκπαίδευση σε περιβάλλον matlab εφόσον χρησιμοποιούνται αντιστοιχίες σε πίνακες. Περιέχει σαράντα νευρώνες, ενώ χρησιμοποιεί οχτώ από τα δέκα αλφάβητα του TrainingSample για εκπαίδευση και τα άλλα δυο για υπολογισμό της απόδοσης του δικτύου. Η μέθοδος εκπαίδευσης Levenberg-Markquardt θεωρείται ως η πιο αποτελεσματική, και τη χρησιμοποιούμε ως προκαθορισμένη.

Εικόνα 5.5.2. - Εκπαιδευμένο νευρωνικό δίκτυο στο matlab

Το δημιουργηθέν νευρωνικό δίκτυο με μόλις έντεκα κύκλους εκπαίδευσης (iterations) πέτυχε ένα πολύ μικρό ποσοστό σφαλμάτων για τα δοθέντα δεδομένα -μόλις 1.76e-09-, ενώ η εκπαίδευσή του τερματίστηκε όταν έφτασε το minimum gradient -πολύ χονδρικά, όταν οι μεταβολές στα συναπτικά βάρη του δικτύου για κάθε επανεκπαίδευση ήταν αμελητέες.

Προβαίνουμε στη δοκιμή του validation sample συμπληρώνοντας πάνω από το σχετικό κουμπί, τον αριθμό των χαρακτήρων ανά γραμμή σ'αυτό. Τονίζεται εκ νέου ότι, λόγω της αδυναμίας του matlab να ταξινομεί τις εικόνες από αριστερά προς τα δεξιά και δεδομένων των προβλημάτων που δημιουργεί η χρήση των συντεταγμένων σ'αυτόν το τομέα, είμαστε υποχρεωμένοι να εκπαιδεύουμε και να δοκιμάζουμε νευρωνικά δίκτυα με σταθερό αριθμό χαρακτήρων ανά γραμμή. Στο κεφάλαιο "εναλλακτική μέθοδος" παρακάμπτουμε το πρόβλημα αυτό -και σε μελλοντικές εκδοχές του matlab είναι σίγουρο ότι ο κώδικας θα μπορέσει να αποτελέσει τη βάση για μια πρακτική εφαρμογή αναγνώρισης κειμένου.



Εικόνα 5.5.3 - Αναγνώριση χαρακτήρων στο interface

Όπως παρατηρούμε, στο συγκεκριμένο αλφάβητο υπάρχει μόλις ένα σφάλμα στο γράμμα "H" που αναγνωρίζεται ως "U". Σε άλλα αλφάβητα θα μπορούσε κάλλιστα να μην υπάρξει κανένα λάθος, όπως υπάρχουν και δείγματα με περισσότερα πιθανά σφάλματα. Αυτό εξαρτάται καθαρά από την ομοιότητά τους με εκείνα που χρησιμοποιήθηκαν για την εκπαίδευση, καθώς και από στατιστικούς παράγοντες.

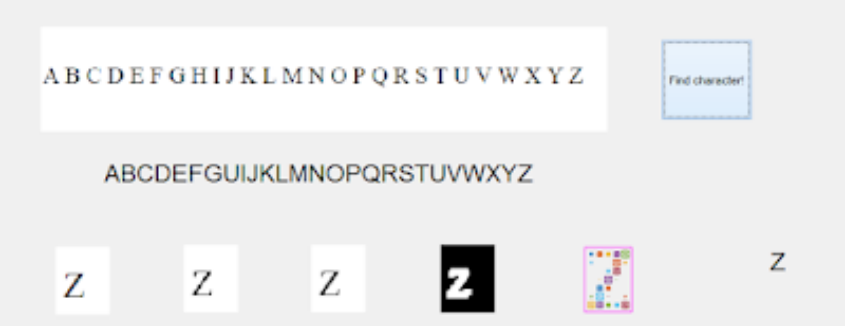
Ας προβούμε στην αναγνώριση ενός μεμονωμένου γράμματος επιλέγοντάς του και κάνοντας δεξί κλικ και "crop":

Εικόνα 5.5.4 - Επιλογή χαρακτήρα προς αποκοπή

Με το πάτημα του κουμπιού find character, εμφανίζονται στα παρακάτω κουτιά τα διαδοχικά στάδια επεξεργασίας που ακολουθούνται για να καταστεί η εικόνα

πρόσφορη προς επεξεργασία από το νευρωνικό δίκτυο.

ΣΗΜΑΝΤΙΚΟ: Η εφαρμογή δεν προβλέπει το σενάριο επιλογής περισσότερων χαρακτήρων του ενός. Οι χρήστες θα πρέπει να αποφεύγουν την επιλογή πολλαπλών χαρακτήρων.



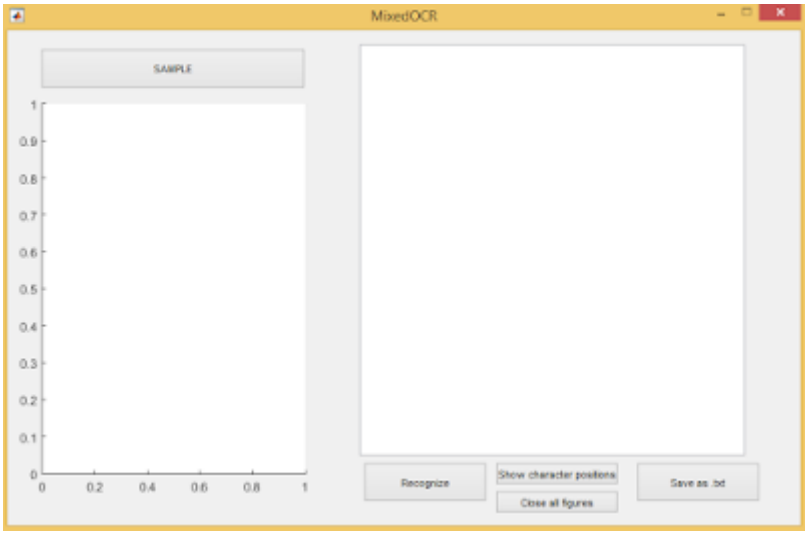
Εικόνα 5.5.5 - Επεξεργασία και αναγνώριση αποκεκομμένου χαρακτήρα

Αρχικά εμφανίζεται η εικόνα που κάναμε "crop", ενώ στα άλλα δυο κουτια δεξιά η ασπρόμαυρη και η δυαδική της -εν προκειμένω δεν υπάρχει καμία εμφανής διαφορά γιατί και η πρωταρχική εικόνα ήταν ούτως ή άλλως ασπρόμαυρη. Έπειτα εμφανίζεται η ίδια εικόνα, με αντεστραμμένο το άσπρο και το μαύρο και τα σημεία στα οποία υπήρχαν μαύρα pixels στην αρχική εικόνα "παραφουσκωμένα" ώστε να μην υπάρχουν κενά ανάμεσά στα pixels και να εντοπίζεται το γράμμα από το matlab ως ενιαία οντότητα, και όχι ως, φερ'ειπείν, τρεις γραμμές.

Η εικόνα αυτή χρησιμεύει για τον προσεγγιστικό εντοπισμό των συντεταγμένων του γράμματος στην εικόνα, και αυτές χρησιμοποιούνται για την απομόνωση του χαρακτήρα από το λευκό υπόβαθρο, με παράλληλη χρήση ενός αλγορίθμου βασισμένου στον αντίστοιχο του συνδέσμου [16] της σχετικής λίστας.

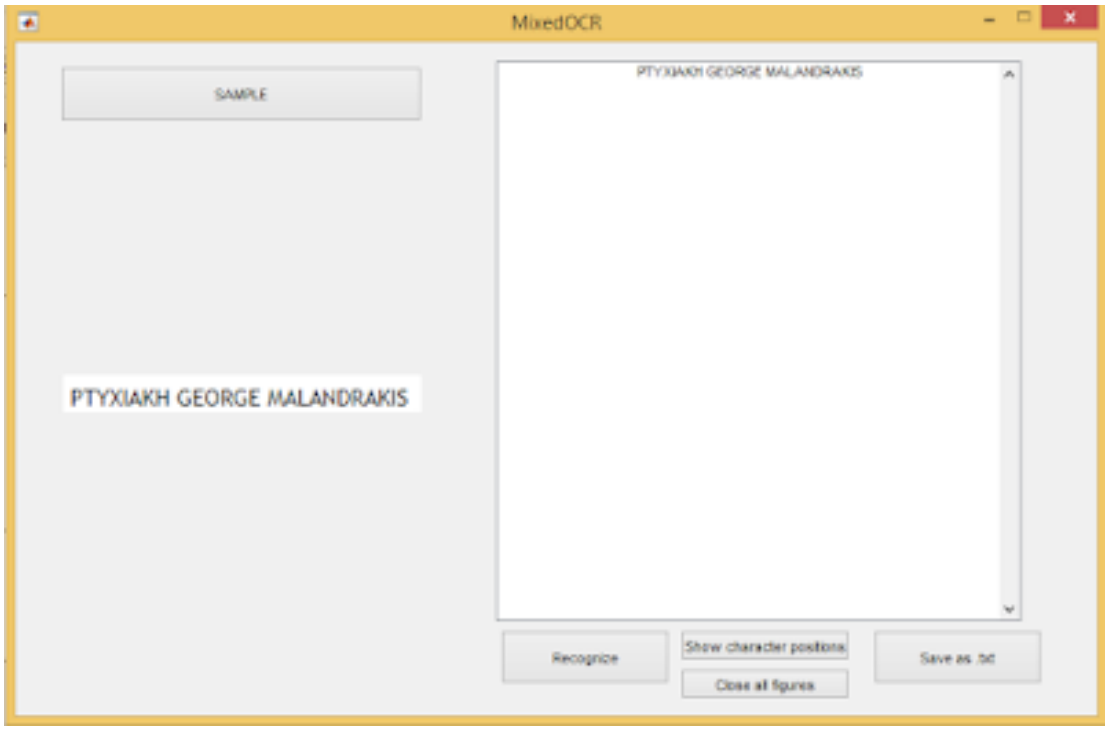
Αφότου απομονωθεί από το λευκό υπόβαθρο, το γράμμα μετατρέπεται σε πίνακα 5x7 του οποίου τα δεδομένα, αν απεικονιστούν, δίνουν την εικόνα που εμπεριέχεται στο ροζ πλαίσιο που φαίνεται παραπάνω. Αυτός ο πίνακας εισάγεται στο νευρωνικό δίκτυο και γίνεται -ορθά- αντιστοίχηση στο γράμμα Z, που εμφανίζεται παραδίπλα.

Αν πατήσουμε το MixedOCR, εμφανίζεται το κάτωθι παράθυρο:



Εικόνα 5.5.6 - Το παράθυρο MixedOCR σε χρήση

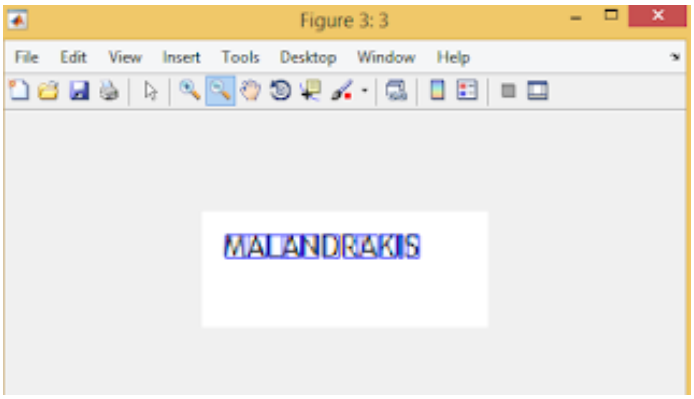
Επιλέγουμε ως sample μια εικόνα εμπεριέχουσα τη γραμμή "PTYXIAKH GEORGE MALANDRAKIS". Παρατηρούμε πως, με το συγκεκριμένο δίκτυο που εκπαιδεύσαμε, το κείμενο αναγνωρίζεται ως έχει



Εικόνα 5.5.7 - Ανάγνωση κειμένου εικόνας στο MixedOCR

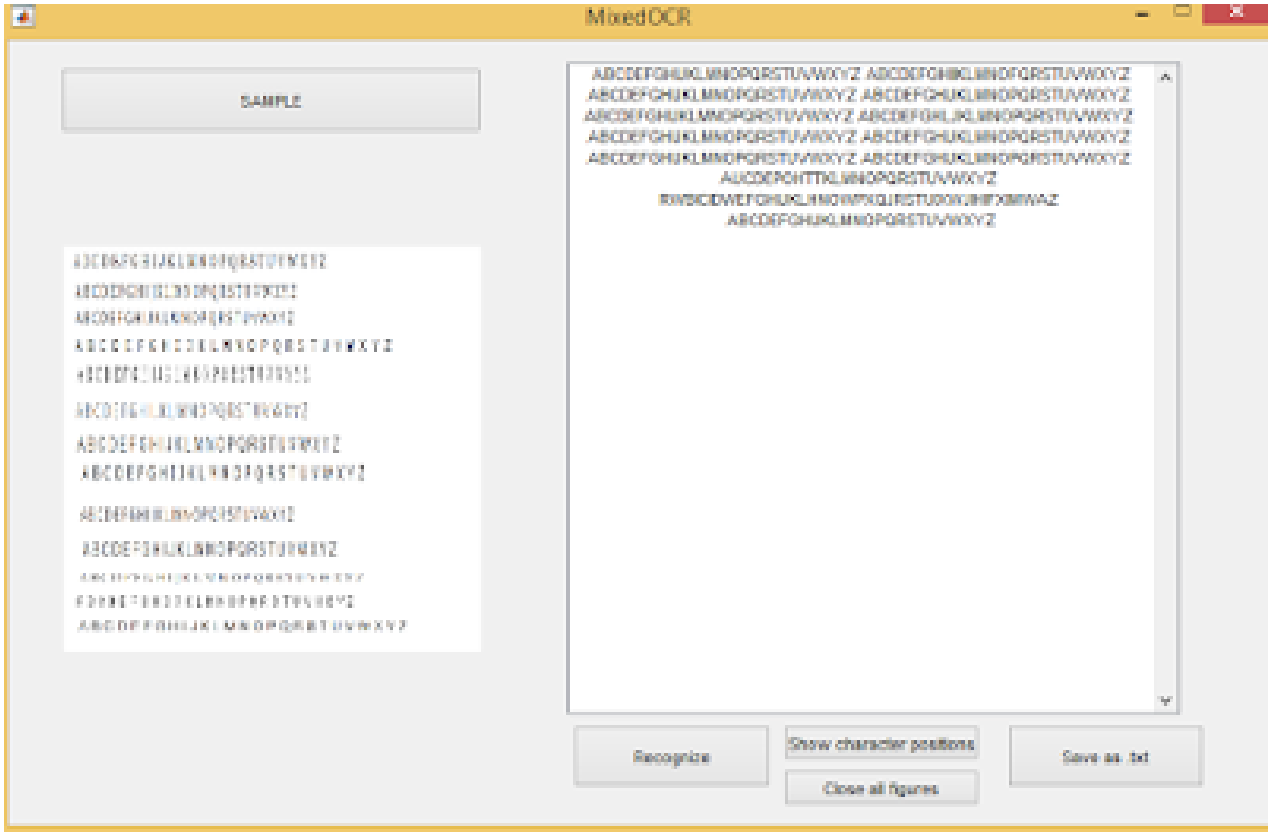
Το κουμπί "show character positions" μας εμφανίζει τις λέξεις και τα γράμματα όπως εντοπίζονται από τη συμβατική OCR μέθοδο, πράγμα που μας επιτρέπει να

εξακριβώσουμε αν τυχόν σφάλματα οφείλονται σε αδυναμία του δικτύου (π.χ. κακή εκπαίδευση) ή σε σφάλμα του συμβατικού αλγορίθμου που χρησιμοποιείται για τον εντοπισμό της θέσης των χαρακτήρων.



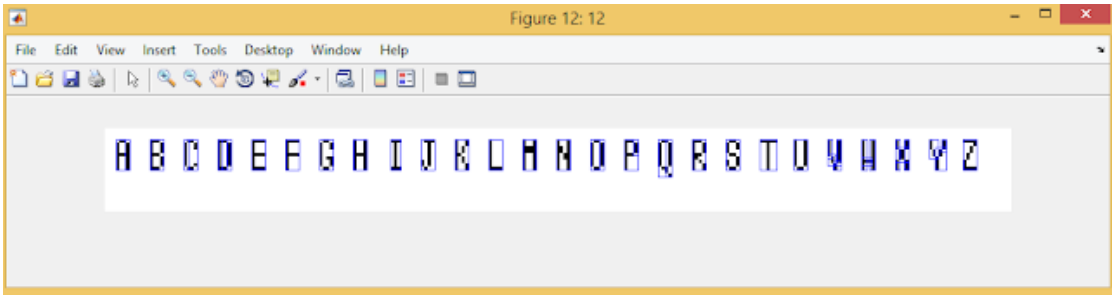
Εικόνα 5.5.8 - Εμφάνιση θέσεων χαρακτήρων

Έτσι, στη μικρότερης επιτυχίας αναγνώριση του κειμένου μιας εικόνας 13 αλφαβήτων (παρατηρήστε τα σφάλματα στο προτελευταίο αλφάβητο), μπορούμε να δούμε πού ακριβώς έχει γίνει το λάθος στην αναγνώριση των θέσεων.



Εικόνα 5.5.9 - Αναγνώριση κειμένου στο MixedOCR (με πρόβλημα)

Πατώντας το και εξετάζοντας τις εμφανιζόμενες εικόνες, βλέπουμε ότι το προτελευταίο αλφάβητο στο οποίο έχουν γίνει τα σφάλματα, έχει ως εξής:



Εικόνα 5.5.10 - Εντοπισμός πηγής προβλήματος

Λόγω της φύσης της γραμματοσειράς, πολλά γράμματα (ιδιαίτερα το από V εως και Y), αναγνωρίζονται από το OCR ως αποτελούμενα από πολλά μικρότερα. Ο χρήστης δύναται να κλείσει τις αναδυθείσες φιγούρες με το κουμπί "close all figures", το οποίο όμως, λόγω της φύσης του matlab και των συναρτήσεων που πρέπει να κληθούν, αποκρίνεται ενίοτε με σημαντική καθυστέρηση. Τέλος, το κουμπί "Save as .txt" επιτρέπει στο χρήστη να αποθηκεύσει το κείμενο που αναγνωρίστηκε.

6. ΜΕΤΡΗΣΕΙΣ ΚΑΙ ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΓΙΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΠΟΙΚΙΛΩΝ ΠΡΟΔΙΑΓΡΑΦΩΝ

6.1. Γενικά - Μεθοδολογία - Προβλέψεις

Η διεπαφή μας επιτρέπει να πειραματιστούμε με έναν πολύ μεγάλο αριθμό νευρωνικών δικτύων που διαφέρουν σε ποικίλες παραμέτρους των προδιαγραφών τους. Είναι πρακτικά αδύνατον να πειραματιστούμε με όλες τους δυνατούς συνδυασμούς προδιαγραφών, καθώς αυτός είναι αστρονομικός, οπότε θα αρκεστούμε σε πειραματισμούς με ορισμένες μόνο κατηγορίες δικτύων.

Συγκεκριμένα, θα αποπειραθούμε να μετρήσουμε τις επιδόσεις νευρωνικών δικτύων εμπρόσθιας τροφοδοτήσεως -που είναι και τα πιο συνήθη-, για 10 έως 100 νευρώνες με αύξηση κατά δέκα κάθε φορά, για όλες τις μεθόδους εκπαίδευσης. Αν μια μέθοδος εκπαίδευσης έχει μεγάλη απαίτηση σε υπολογιστική ισχύ (π.χ. η Νευτώνεια), οι μετρήσεις θα γίνουν μέχρι τον αριθμό των νευρώνων για τον οποίο μπορεί να ολοκληρωθεί η εκπαίδευση σε εύλογο χρονικό διάστημα (π.χ. 80 νευρώνες).

Θα πάρουμε αντίστοιχες μετρήσεις και σε ένα πιθανοκρατικό, το οποίο όμως χρησιμοποιεί δική του μέθοδο εκπαίδευσης. Η δοκιμή των μεθόδων εκπαίδευσης σε όλα τα διαθέσιμα είδη δικτύων θα υπερέβαινε κατά πολύ τους σκοπούς της παρούσης πτυχιακής, οπότε θα αρκεστούμε σε αυτές.

Μας είναι απαραίτητος ένας τύπος καθορισμού της απόδοσης του δικτύου -χωρίς όμως να υπάρχει κάποιος έτοιμος. Ο τύπος θα πρέπει σίγουρα να λαμβάνει υπόψιν τις παραμέτρους του μέσου χρόνου εκπαίδευσης και του μέσου MSE, ενώ το αποτέλεσμα θα ήταν προτιμότερο να έχει ως σημείο αναφοράς κάποια δύναμη του δέκα ώστε να μπορεί να εκφραστεί ως ποσοστό τοις εκατό προς κάποια μέγιστη δυνατή τιμή. Είναι επίσης απαραίτητο ο τύπος να μην μπορεί να οδηγήσει σε απειρισμούς (π.χ. διαίρεση δια του μηδενός).

Για να επιτευχθεί ο δεύτερος σκοπός -σημείο αναφοράς κάποια δύναμη του δέκα- θα χρησιμοποιηθεί κλάσμα με άσσο στον αριθμητή, ενώ για τον τρίτο θα

εξασφαλίσουμε ότι ο παρονομαστής θα έχει ελάχιστη τιμή τη μονάδα. Ο παρονομαστής θα βασίζεται στις τιμές του MSE και του μέσου χρόνου εκπαίδευσης. Λόγω του ότι οι εν λόγω παράμετροι μπορεί να είναι μηδενικές, θα προστεθεί σε αυτές η μονάδα για να αποφεύγονται οι απειρισμοί (έτσι η ελάχιστες τιμές τους θα είναι 1 αντί για 0), ενώ θα πολλαπλασιάζονται με συντελεστές ούτως ώστε το άθροισμά τους στην ελάχιστη τιμή να είναι η μονάδα.

Για τη παρούσα εργασία θα χρησιμοποιηθεί ο εξής τύπος:

$$\frac{1}{(AverageMSE + 1) * 0.7 + (AverageTrainingTime + 1) * 0.3}$$

Τύπος 6.1.1 - Επίδοση νευρωνικού δικτύου

Οι συντελεστές 0.7 και 0.3 είναι ομολογουμένως αυθαίρετοι, και κάποιος που δίνει μεγαλύτερη σημασία στο χρόνο εκπαίδευσης θα μπορούσε να τους τροποποιήσει ανάλογα. Οι προστιθέμενοι άσσοι διασφαλίζουν, ως ανεφέρθη, ότι οι ελάχιστες τιμές δεν θα οδηγούν σε απειρισμούς και ότι θα υπάρχει μια μέγιστη δυνατή τιμή.

Με τη χρήση του εν λόγω τύπου ένα δίκτυο με μηδενικό χρόνο εκπαίδευσης και μηδενικό σφάλμα θα είχε απόδοση ακριβώς ίση με τη μονάδα. Από εκεί και έπειτα οι καλύτερες επιδόσεις στα δίκτυα προς εξέταση θα είναι εκείνες που τη πλησιάζουν περισσότερο

Ο σκοπός των μετρήσεων είναι αφενός η καθ'αυτή δοκιμή των νευρωνικών δικτύων, και αφετέρου η επαλήθευση ή η διάψευση των παρακάτω προβλέψεων:

1. Όταν ο αριθμός των νευρώνων είναι δυσανάλογα μεγαλύτερος του αριθμού παραδειγμάτων προς εκπαίδευση, οι επιδόσεις του δικτύου θα είναι μειωμένες.

Αυτό γιατί, για να εκπαιδευτεί σωστά ένας μεγάλος αριθμός νευρώνων, πιθανότατα θα χρειάζεται και ένας αντίστοιχα μεγάλος δεδομένων. Εκπαίδευση πολλών νευρώνων με λίγα παραδείγματα αφήνει μεγάλα περιθώρια για τυχαιότητα στα αποτελέσματα, γιατί οι νευρώνες δεν λαμβάνουν επαρκές "feedback" και το αποτέλεσμα θα είναι ένα νευρωνικό δίκτυο ικανό να αναγνωρίσει σωστά μόνο τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση ή όσα μοιάζουν πολύ σ'αυτά. Η πρόβλεψη αυτή θα εξεταστεί δια της μεταβολής του αριθμού των νευρώνων με

σταθερό αριθμό παραδειγμάτων.

2. Όταν ο αριθμός των νευρώνων θα είναι δυσανάλογα μικρότερος του αριθμού παραδειγμάτων, οι επιδόσεις του δικτύου θα είναι επίσης μειωμένες.

Σε μια τέτοια περίπτωση οι νευρώνες θα εξωθούνταν στο να αναγνωρίζουν μόνο πολύ γενικά patterns των χαρακτήρων, που θα είχε ως αποτέλεσμα π.χ. το "Q" και το "O" να θεωρούνται ως το ίδιο γράμμα λόγω του ότι αποτελούνται από έναν κύκλο, ή αντίστοιχα για το "V" και το "U" ή το "I" με το "J". Ο συνδυασμός των πρώτων δυο προβλέψεων πρακτικά σημαίνει πως, αν βάλουμε στον άξονα Χ ενός γραφήματος τον αριθμό των νευρώνων και στο Υ το ποσοστό σφαλμάτων, περιμένουμε με ένωση των αντίστοιχων σημείων να προκύψει περίπου ένα "V" -με τα σφάλματα αυξημένα στους λίγους και τους πολλούς νευρώνες, και μειωμένα σε ενδιάμεσο αριθμό.

3. Η αποτελεσματικότητα των μεθόδων εκπαίδευσης θα είναι ευθέως ανάλογη της πολυπλοκότητάς τους.

Με βάση αυτό, περιμένουμε μια κατάταξη των αλγορίθμων με βάση την αποτελεσματικότητα από τον λιγότερο στον περισσότερο αποτελεσματικο ως εξής:

α. Resilient backpropagation

β. Gradient-descent

γ. Quasi-newton (το matlab δεν χρησιμοποιεί αμιγή νευτώνεια λόγω απαιτήσεων σε υπολογιστική ισχύ)

δ. Levenberg-Markquardt

ε. Levenberg-Markquardt με bayesian κανονικοποίηση

Θεωρητικά, όλα τα νευρωνικά δίκτυα μπορούν να εκπαιδευθούν εξίσου αποτελεσματικά με όλες τις μεθόδους εκπαίδευσης -και το μόνο που αλλάζει είναι ο απαιτούμενος χρόνος.

4. Τα πιθανοκρατικά δίκτυα εμπρόσθιας τροφοδότησης θα είναι αποτελεσματικότερα από τα απλά εμπρόσθιας τροφοδότησης στην αναγνώριση χαρακτήρων.

Λόγω του ότι η αναγνώριση των χαρακτήρων γίνεται με χρήση ενός πίνακα 5x7,

ένα νευρωνικό δίκτυο που βασίζεται σε γενικές πιθανότητες αντί να στηρίζεται στις τιμές του πίνακα σε κάθε θέση του, θα έχει καλύτερα αποτελέσματα από ένα συμβατικό εμπρόσθιας τροφοδότησης. Εν προκειμένω, με τον όρο "αποτελέσματα" εννοούμε το ποσοστό των σφαλμάτων. Να σημειωθεί ότι στο matlab τα πιθανοκρατικά δίκτυα χρησιμοποιούν δική τους μέθοδο εκπαίδευσης.

6.2. Διευκρινίσεις

1. Καθώς τα νευρωνικά δίκτυα στην αρχική τους μορφή έχουν αυθαίρετες, τυχαίες τιμές στα συναπτικά βάρη, ενυπάρχει ένας βαθμός τυχαιότητας στα αποτελέσματα της εκπαίδευσης και στις μετέπειτα τιμές των συναπτικών βαρών των νευρώνων. Ως εκ τούτου, η επανάληψη των μετρήσεων θα βγάλει συναφή μεν, διαφορετικά δε, αποτελέσματα, και αν κάποιος τις πάρει εκ νέου η απόδοση των δικτύων μετρουμενη σε mean squared error θα διαφέρει από την αρχική. Οι μετρήσεις που παίρνουμε είναι ενδεικτικές και προσεγγιστικές, και εξυπηρετούν το σκοπό της εξαγωγής συμπερασμάτων, και όχι της καταχώρησης συγκεκριμένων τιμών σε πίνακες.

2.Το neural network toolbox του matlab με τις αρχικές του ρυθμίσεις χρησιμοποιεί ένα τμήμα των δεδομένων εκπαίδευσης για μέτρηση της απόδοσης. Δηλαδή, από κάθε σειρά δεδομένων ένα ποσοστό (default: 70%) θα χρησιμοποιηθεί για εκπαίδευση, και το υπόλοιπο για τη μέτρηση της απόδοσης, που μετριέται σε Mean Squared Error και όσο μικρότερη είναι η τιμή της, τόσο πιο αποτελεσματικό το δίκτυο. Στην εφαρμογή, ωστόσο, η απόδοση υπολογίζεται με την εντολή perform και τη χρήση ενός τμήματος του data sample που δεν αποτελούσε τμήμα των δεδομένων εκπαίδευσης (βλ. μεταβλητή Ptest στον κώδικα). Αυτό ούτως ώστε να υπολογίζει την απόδοση για ολόκληρες σειρές χαρακτήρων (η χρήση ποσοστών σχεδόν αποκλείει αυτό το σενάριο). Αν χρησιμοποιούνται όλα τα δεδομένα στο τμήμα εκπαίδευσης, ο υπολογισμός γίνεται με τη συμβατική μέθοδο.

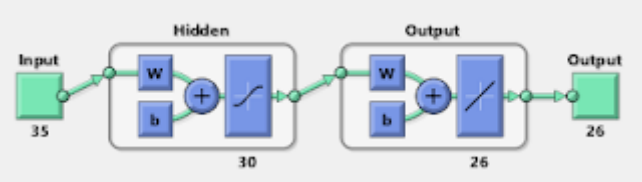
3. Στις μετρήσεις θα χρησιμοποιήσουμε, όπου δεν ορίζεται αλλιώς, οχτώ αλφάβητα για εκπαίδευση, δυο για μέτρηση της απόδοσης, και ένα επιπλέον (ValidationSample) για αναγνώριση και εμφάνιση των χαρακτήρων στο interface της εφαρμογής. Η εικόνα του set εκπαίδευσης είναι η EIK1-10.bmp των αρχείων, και του ValidationSample η alph-1.bmp.

4. Ο διαχωρισμός των χαρακτήρων του ValidationSample έγινε με τη χρήση της function1, όπως και η εκπαίδευση του νευρωνικού δικτύου, και όχι της εναλλακτικής μεθόδου functionZ. Πράγμα πολύ σημαντικό, γιατί η εναλλακτική μέθοδος στηρίζεται στο OCR του matlab για τον εντοπισμό των χαρακτήρων και αποδεικνύεται συχνά προβληματική (π.χ. εντοπίζει δυο γράμματα εκεί που υπάρχει ένα).

6.3. Μετρήσεις δικτύων εμπρόσθιας τροφοδότησης - Γραφικές παραστάσεις

Τα δίκτυα εμπρόσθιας τροφοδότησης (βλ. κεφάλαιο 2.5.1) αποτελούν τα πιο απλά δίκτυα, από σκοπιά αρχιτεκτονικής, και αποτελούν τη προκαθορισμένη επιλογή του interface της εφαρμογής. Υλοποιούνται με την εντολή **feedforwardnet(neur)**, όπου neur ο αριθμός των νευρώνων.

Ένα δίκτυο εμπρόσθιας τροφοδότησης τριάντα νευρώνων, για την αναγνώριση χαρακτήρων του λατινικού αλφαβήτου στη παρούσα εφαρμογή, απεικονίζεται στο matlab ως εξής:



Εικόνα 6.3.1 - Νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης στο matlab

Αν ο εξεταστής ενδιαφέρεται για τις τιμές των πινάκων και των μεταβλητών κατά τη προετοιμασία των εικόνων, και τη δημιουργία και εκπαίδευση του δικτύου, μπορεί να κάνει αντιγραφή και επικόλληση στο matlab τις εντολές του experimental.m (βλ. αρχεία κώδικα), με χρήση της

```
net = feedforwardnet(neur);
```

όπου neur ο επιθυμητός αριθμός των νευρώνων.

Για τις παρακάτω μετρήσεις, στις οποίες έγινε εκπαίδευση και δοκιμή σε λατινικά αλφάβητα, και συγκεκριμένα με TrainingSet την εικόνα EIK1-10.bmp των αρχείων και ValidationSet την alpha-1.bmp, χρησιμοποιήθηκαν οχτώ (από τα δέκα διαθέσιμα)

αλφάβητα για την εκπαίδευση του δικτύου, δυο για τον υπολογισμό του mean-squared error, και ένα επιπλέον για την εμφάνιση των ευρεθέντων χαρακτήρων. Σε περιπτώσεις που ισχύει οτιδήποτε άλλο αυτό αναγράφεται.

6.3.1. Μετρήσεις για μέθοδο Levenberg-Markquardt

Οι μετρήσεις έχουν ως εξής:

Πίνακας 6.3.1 - Μετρήσεις για feedforward network με εκπαίδευση Levenberg-Markquardt

NEURONS	MEAN-SQUARED ERROR	TIME	MISTAKES	ITERATIONS
10	0.0267805	2	7/26	11
20	0.0160986	5	3/26	13
30	0.0089415	11	1/26	15
40	0.0094116	16	0/26	15
50	0.0089700	25	1/26	13
60	0.0141975	23	1/26	9
70	0.0132253	43	1/26	12
80	0.0125012	61	1/26	12
90	0.0304614	76	2/26	11
100	0.0262820	58	4/26	8

Το mean-squared error υπολογίστηκε με χρήση της εντολής perform, που είναι πρακτικά ισοδύναμη και βγάζει ίδια αποτελέσματα με τα της αντίστοιχης mse. Ο αριθμός των λαθών (mistakes) είναι τα λάθη που εντοπίστηκαν στην αναγνώριση του ValidationSet με τη function1.

Για να ελέγξουμε την αξιοπιστία των μετρήσεων και το πόσο διαφέρουν μεταξύ τους αν ληφθούν εκ νέου, τις πήραμε και δεύτερη φορά, και έχουν ως εξής:

Πίνακας 6.3.2 - Επαλήθευση μετρήσεων

NEURONS	MEAN-SQUARED ERROR	TIME	MISTAKES	ITERATIONS
10	0.0248373	2	9/26	14
20	0.0170551	3	5/26	11
30	0.0081761	11	0/26	16
40	0.0145589	12	0/26	11
50	0.0101512	17	2/26	10
60	0.0127238	30	2/26	12
70	0.0122952	38	2/26	11
80	0.0146040	43	5/26	19
90	0.0152556	69	2/26	11
100	0.0273287	50	3/26	8

Οι διαφορές είναι μεν υπαρκτές, αλλά πρακτικά αμελητέες, μιας και όπως βλέπουμε η απόκλιση στην απόδοση είναι πολύ μικρή, μόλις 0.014, -κάτι ενθαρρυντικό για την αξιοπιστία των μετρήσεων.

πρώτο set:

Μέσο σφάλμα: 0.016686955

Ελάχιστο σφάλμα: 0.0089415

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 0

Μέσος χρόνος εκπαίδευσης: 32

Μέση απόδοση: 0.0942357778

δεύτερο:

Μέσο σφάλμα: 0.015698594

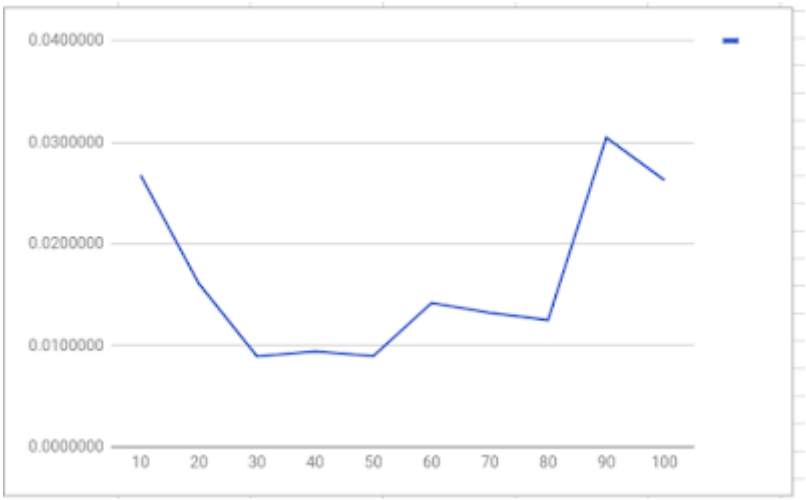
Ελάχιστο σφάλμα: 0.0081761

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 0

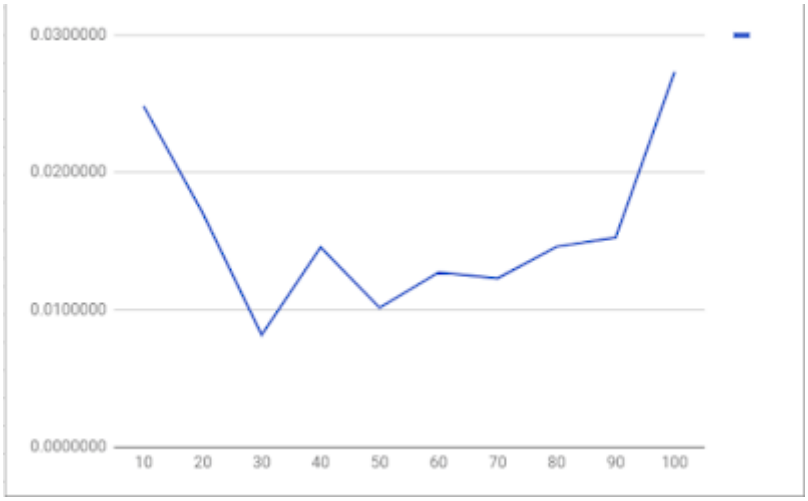
Μέσος χρόνος εκπαίδευσης: 27.5

Μέση απόδοση: 0.107979828

Τα διαγράμματα συσχέτισης αριθμού νευρώνων και σφαλμάτων για το πρώτο και το δεύτερο set εκπαίδευσης αντίστοιχα, έχουν ως εξής:



Διάγραμμα 6.3.1 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για Levenberg-Markquardt



Διάγραμμα 6.3.2 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για Levenberg-Markquardt (set επαλήθευσης)

Παρατηρούμε μια επιβεβαίωση των πρώτων δυο υποθέσεών μας -ότι τα σφάλματα θα είναι αυξημένα στους πολύ λίγους και τους πολλούς νευρώνες αντίστοιχα, έστω και αν στο δεύτερο διάγραμμα παρατηρείται ένα "spike" στους σαράντα νευρώνες. Λόγω της αντιεπιστημονικότητας στη τροποίηση ή επανάληψη των μετρήσεων μέχρι να συμφωνήσουν με τις προβλέψεις, αφήνουμε το "spike" ως έχει και το αποδίδουμε στη τυχαιότητα των αρχικών τιμών των νευρωνικών δικτύων (βλ. διευκρινίσεις).

6.3.2. Μετρήσεις για μέθοδο εκπαίδευσης Resilient-backpropagation

Οι μετρήσεις έχουν ως εξής:

Πίνακας 6.3.3 - Μετρήσεις για feedforward network με εκπαίδευση Resilient-backpropagation

10	0.0284991	0	9/26	69
20	0.0216685	0	6/26	106
30	0.0157434	0	2/26	156
40	0.0149452	0	0/26	261
50	0.0209774	0	4/26	151
60	0.0340136	0	3/26	305
70	0.0220652	0	5/26	288
80	0.0327020	0	6/26	353
90	0.0360044	1	8/26	486
100	0.0377362	1	6/26	632

Μέσο σφάλμα: 0.0264355

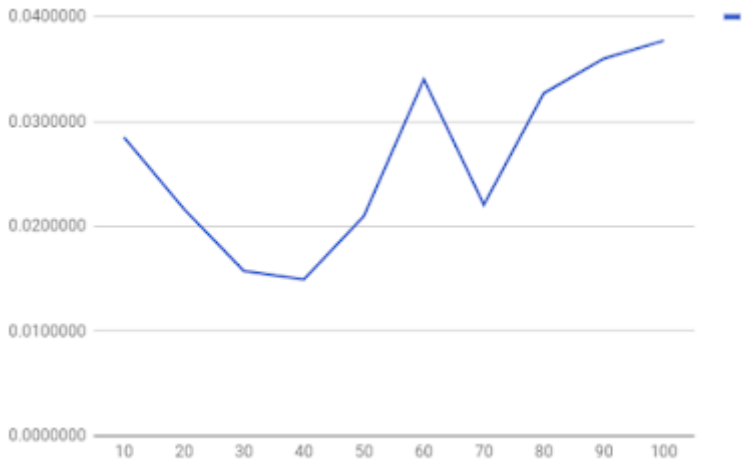
Ελάχιστο σφάλμα: 0.0149452

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 0

Μέσος χρόνος εκπαίδευσης: 0.2

Μέση απόδοση: 0.927209553

Και το διάγραμμα συσχέτισης:



Διάγραμμα 6.3.3 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για resilient-backrpropagation

Τα spikes είναι υπαρκτά και τα αφήνουμε ως έχουν, ενώ η πρόβλεψή μας περί αυξημένων σφαλμάτων στους λίγους και τους πολλούς νευρώνες επιβεβαιώνεται εκ νέου. Η πρόβλεψη που διαψεύστηκε εντελώς όμως, είναι η αφορούσα στην ευθεία συσχέτιση της πολυπλοκότητας των μεθόδων εκπαίδευσης και την απόδοση. Αν και θα περιμέναμε χαμηλή απόδοση (π.χ. μεγάλο χρόνο εκπαίδευσης) για μια μαθηματικά απλή μέθοδο, όπως η gradient-descent ή η παρούσα, βλέπουμε ότι η τελευταία έχει μια μέση απόδοση ίση με το 92.7% της μέγιστης και ιδανικής (μηδενικός χρόνος εκπαίδευσης και μηδέν σφάλματα), χωρίς το ελάχιστο σφάλμα να είναι ιδιαίτερα υψηλό.

6.3.3. Μετρήσεις για μέθοδο εκπαίδευσης gradient-descent

Οι μετρήσεις έχουν ως εξής:

Πίνακας 6.3.4. - Μετρήσεις για feedforward network με εκπαίδευση gradient descent

NEURONS	MEAN-SQUARED ERROR	TIME	MISTAKES	ITERATIONS
10	0.0291366	211	11/26	100.000
20	0.0251502	214	9/26	100.000
30	0.0194972	255	3/26	100.000
40	0.0199229	280	5/26	100.000
50	0.0221766	301	4/26	100.000
60	0.0208130	302	2/26	100.000
70	0.0234711	327	2/26	100.000
80	0.0447529	354	5/26	100.000
90	0.0288683	349	2/26	100.000
100	0.0354881	340	4/26	100.000

Μέσο σφάλμα: 0.02692769

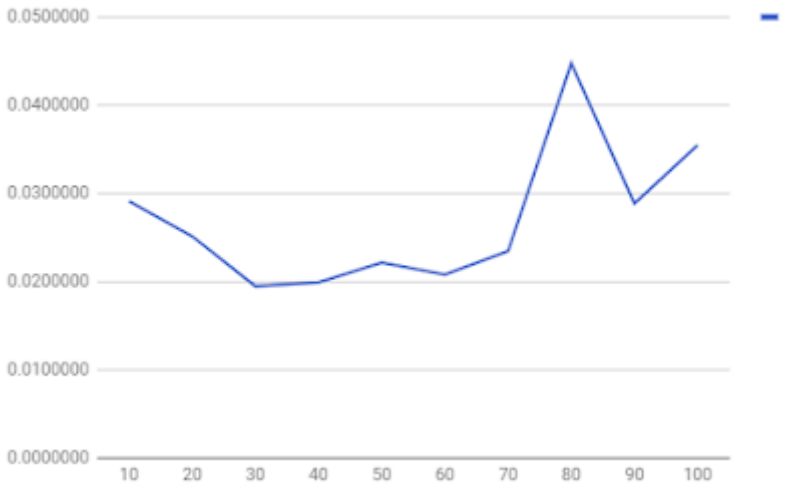
Ελάχιστο σφάλμα: 0.0194972

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 2

Μέσος χρόνος εκπαίδευσης: 294.3

Μέση απόδοση: 0.0111970987

Και η γραφική παράσταση:



Διάγραμμα 6.3.4 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για gradient-descent

Έστω και αν υπάρχει κάποια απόκλιση στους μεγάλους αριθμούς νευρώνων, εδώ

φαίνεται να επιβεβαιώνονται και οι δυο πρώτες υποθέσεις μας, καθώς και η αφορούσα στη περιορισμένη αποτελεσματικότητα των απλών αλγορίθμων εκπαίδευσης (η οποία διαψεύστηκε από τη resilient-backpropagation).

6.3.4. Μετρήσεις για μέθοδο εκπαίδευσης Levenberg-Markquardt με Bayesian κανονικοποίηση

Οι μετρήσεις έχουν ως εξής:

Πίνακας 6.3.5. - Μετρήσεις για feedforward network με εκπαίδευση levenberg-Markquardt με Bayesian Regularization:

NEURONS	MEAN-SQUARED ERROR	TIME	MISTAKES	ITERATIONS
10	0.02341610	113	11/26	627
20	0.11104700	293	7/26	525
30	0.14474100	200	2/26	171
40	0.01880220	197	2/26	95
50	0.02652530	1,675	2/26	481
60	0.00748089	12,388	2/26	2344

Στους εξήντα νευρώνες αφήσαμε το δίκτυο να εκπαιδευτεί συνολικά πάνω από τρεις ώρες προτού το αναγκάσουμε να σταματήσει (τα άλλα σταματούσαν από μόνα τους με την ολοκλήρωση της εκπαίδευσης) γιατί δεν μειωνόταν (εμφανώς) περαιτέρω το mean squared error για περισσότερες από χίλιες επαναλήψεις (iterations), και επομένως δεν είχε νόημα να συνεχιστεί η εκπαίδευση.

Μέσο σφάλμα: 0.055335415

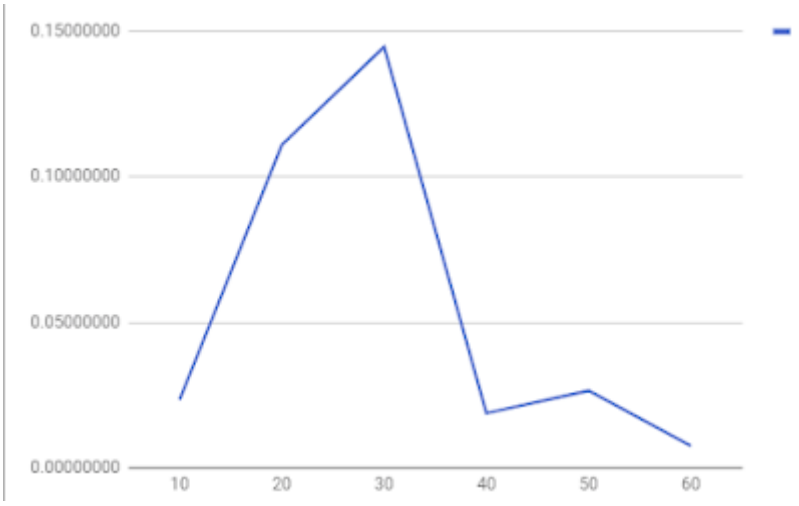
Ελάχιστο σφάλμα: 0.00748089

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 2

Μέσος χρόνος εκπαίδευσης: 2477.666667 (με επιφύλαξη)

Μέση απόδοση: 0.00134347435 (με επιφύλαξη)

Η γραφική παράσταση:



Διάγραμμα 6.3.5 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων Bayesian Regularization

Το δίκτυο με αυτή τη μέθοδο φαίνεται να έχει αποδοτικότητα 0.007293801678, για την οποία διατηρούμε επιφύλαξη αφενός γιατί δεν πήραμε αρκετές μετρήσεις, και αφετέρου γιατί η τελευταία διακόπηκε χειροκίνητα. Η πραγματική αποδοτικότητα κατά πάσα πιθανότατα είναι μικρότερη.

6.3.5. Μετρήσεις για μέθοδο εκπαίδευσης Quasi-Newton

Οι μετρήσεις έχουν ως εξής:

Πίνακας 6.3.6. - Μετρήσεις για feedforward network με εκπαίδευση Quasi-Newton

NEV	SF	TIME	LATHI	ITER
10	0.0280376	7	17/26	76
20	0.0229485	73	6/26	82
30	0.0196878	365	6/26	115
40	0.0152390	2,300	1/26	312
50	0.0127911	1,921	2/26	136
60	0.0180110	3,052	3/26	123
70	0.0134477	6,408	2/26	166

Σταματήσαμε στους εβδομήντα νευρώνες λόγω του μεγάλου απαιτούμενου χρόνου εκπαίδευσης. Θυμηθείτε ότι η εν λόγω μέθοδος είναι ιδιαίτερα απαιτητική σε υπολογιστική ισχύ.

Μέσο σφάλμα: 0.01859467143

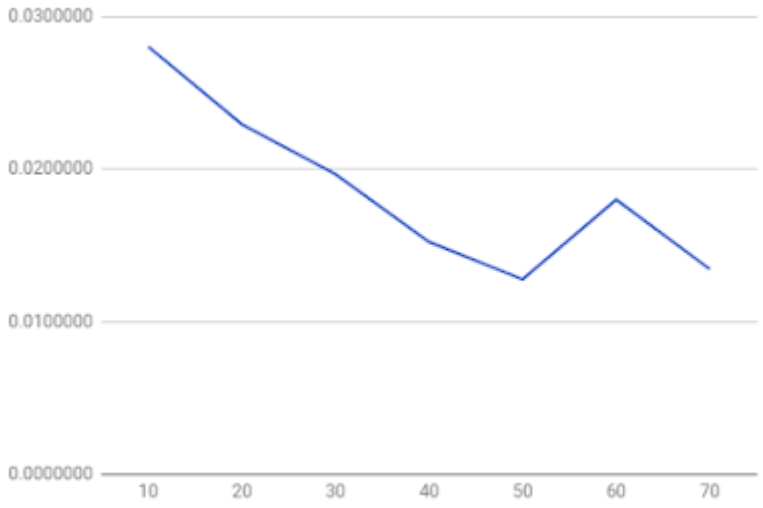
Ελάχιστο σφάλμα: 0.0127911

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 1

Μέσος χρόνος εκπαίδευσης: 2018 (με επιφύλαξη)

Μέση απόδοση: 0.00164904112 (με επιφύλαξη)

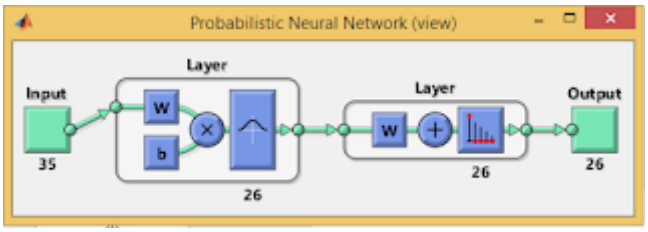
Η γραφική παράσταση έχει ως εξής:



Διάγραμμα 6.3.6 - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για Quasi-Newton

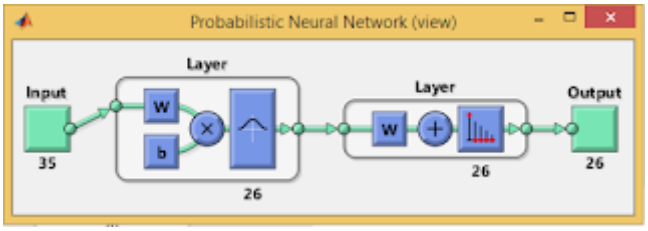
6.4. Μετρήσεις πιθανοκρατικού δικτύου - Γραφικές παραστάσεις

Τα πιθανοκρατικά (probabilistic) δίκτυα, που περιγράφτηκαν στο θεωρητικό μέρος, μοιάζουν με τα εμπροσθοτοτροφοδοτούμενα με τη διαφορά ότι χρησιμοποιούν και ένα επιπλέον επίπεδο για τη συνάθροιση των αποτελεσμάτων και την εξαγωγή της μαθηματικής πιθανότητας. Σε αντίθεση με τα άλλα δίκτυα που διατίθενται στο matlab, τα πιθανοκρατικά έχουν δική τους μέθοδο εκπαίδευσης και ο αριθμός των νευρώνων τους καθορίζεται από τον αριθμό των στοιχείων ενός training set, ενώ δεν μας επιστρέφονται πληροφορίες όπως ο χρόνος εκπαίδευσης. Ένα τέτοιο δίκτυο φαίνεται στη παρακάτω εικόνα:



Εικόνα 6.4.1. - Πιθανοκρατικό νευρωνικό δίκτυο στο matlab

Το συγκεκριμένο έχει μόνο 26 νευρώνες στο πρώτο επίπεδο (μόνο ένα training set) και 26 στο δεύτερο (αντιστοίχηση σε 26 πιθανές εξόδους). Αν είχαμε 5 training sets για Ελληνικό αλφάβητο οι αντίστοιχοι αριθμοί θα ήταν 120 και 24. Οι μετρήσεις έχουν ως εξής:



Πίνακας 6.4.1. - Μετρήσεις πιθανοκρατικού δικτύου

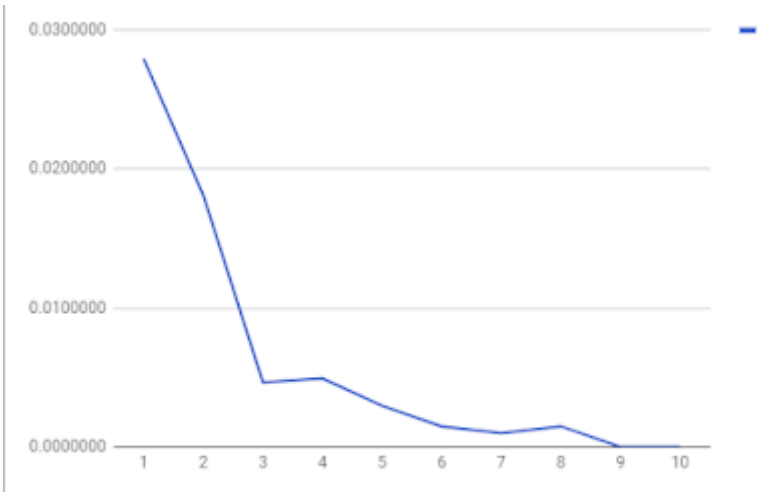
Μέσο σφάλμα: 0.0062546923

Ελάχιστο σφάλμα: 0

Ελάχιστος αριθμός σφαλμάτων στο αλφάβητο: 0

Μέσος χρόνος εκπαίδευσης: 0.1

Μέση απόδοση: 0.966764302



Διάγραμμα 6.4.1. - Διάγραμμα σφαλμάτων-αριθμού νευρώνων για πιθανοκρατικό δίκτυο

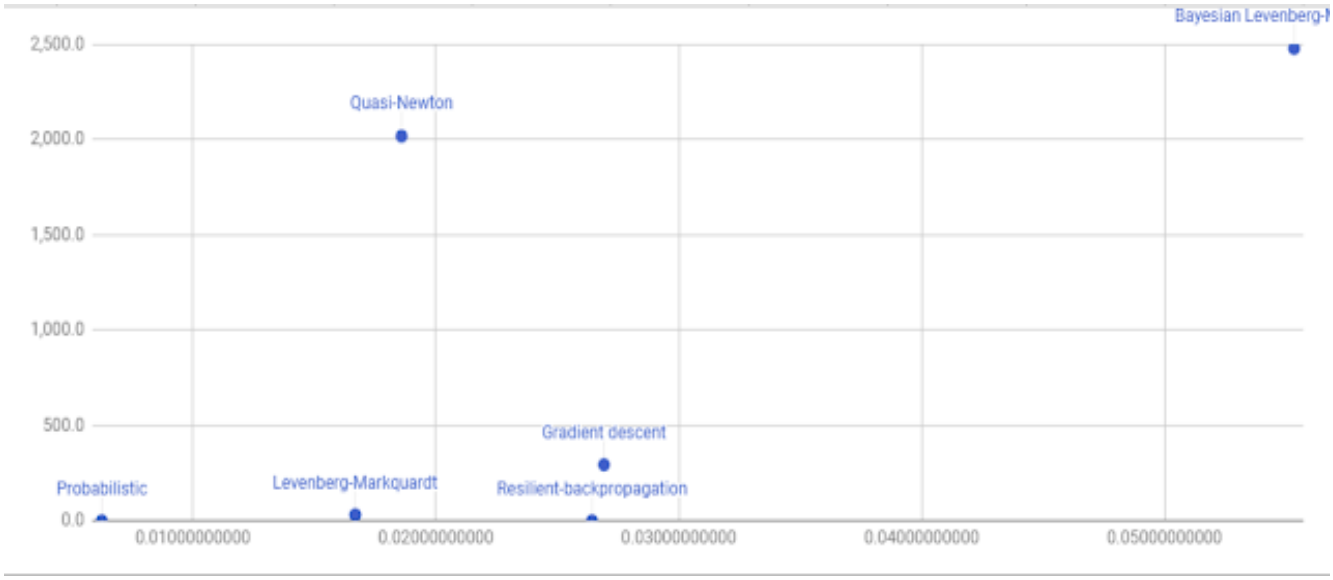
Παρατηρείται όντως μια θεαματικότατη μείωση των σφαλμάτων όσο αυξάνεται ο αριθμός των training sets, που τελικά καταλήγει στο μηδέν. Έτσι, επιβεβαιώνεται η

πρόβλεψή μας ότι τα πιθανοκρατικά δίκτυα θα ήταν αποτελεσματικότερα για εφαρμογή αναγνώρισης χαρακτήρων.

6.5. Γενικές παρατηρήσεις και καταλληλότητα δικτύων

Οι καλύτερες επιδόσεις (ήτοι, αυτές που πλησιάζουν περισσότερο την ιδανική, που ισούται με τη μονάδα), είναι εκείνες του εμπροσθοτοτροφοδοτούμενου με μέθοδο εκπαίδευσης Levenberg-Markquardt, resilient backpropagation καθώς και του πιθανοκρατικού δικτύου. Οι επιδόσεις τους ισούνται με 0.0942357778, 0.927209553 και 0.966764302 αντίστοιχα, και αντιστοιχούν έτσι στο 94%, 92% και 96% της ιδανικής επίδοσης.

Η γραφική παράσταση των επιδόσεων συναρτήσει του μέσου χρόνου εκπαίδευσης και μέσου σφάλματος στους άξονες Y και X αντίστοιχα, έχει ως εξής:



Εικόνα 6.5.1. - Σύγκριση επιδόσεων για μεθόδους εκπαίδευσης

Στην εν λόγω γραφική παράσταση, όσο περισσότερο απέχει από τις αρχές των αξόνων μια κουκκίδα τόσο λιγότερο κατάλληλη είναι η τοπολογία ή μέθοδος εκπαίδευσης του δικτύου. Ο λόγος για τον οποίο η resilient back propagation απέχει περισσότερο από το ιδανικό σε σχέση με τη Levenberg-Markquardt, είναι ο χαμηλότερος αριθμός σφαλμάτων της δεύτερης.

Όπως προβλέψαμε εξαρχής, η καταλληλότερη τοπολογία θα ήταν εκείνη των πιθανοκρατικών δικτύων αντί των απλής εμπρόσθιας τροφοδότησης. ενώ η

εξαιρετική αποδοτικότητα της μεθόδου resilient backpropagation διέψευσε πλήρως τη τρίτη πρόβλεψή μας περί αναλογίας μαθηματικής πολυπλοκότητας και αποτελεσματικότητας των αλγορίθμων εκπαίδευσης. Η πιο σύνθετη μέθοδος - Levenberg-Markquardt με bayesian κανονικοποίηση- είναι μακράν η πιο ακατάλληλη, ενώ η απλούστερη resilient backpropagation πλησιάζει την ιδανική επίδοση στα εμπροσθοτοτροφοδοτούμενα δίκτυα.

Παρατηρήθηκαν επίσης, στις περισσότερες περιπτώσεις τουλάχιστον, καμπύλες όμοιες με U ή V έστω και με "spikes", στις γραφικές παραστάσεις αναλογίας αριθμού νευρώνων και MSE, και επομένως η πρόβλεψή μας περί αναλογιών σφαλμάτων και αριθμού νευρώνων ήταν αρκετά κοντά στη πραγματικότητα.

Αν επρόκειτο να πάρουμε μετρήσεις για άλλες κατηγορίες δικτύων (π.χ. cascade forward) θα έπρεπε να δοκιμάσουμε εκ νέου όλες τις μεθόδους για να βγάλουμε ορθά συμπεράσματα, και κάτι τέτοιο θα υπερέβαινε κατά πολύ τους σκοπούς της παρούσης πτυχιακής. Όμως δεν έχουμε λόγο να πιστεύουμε ότι θα υπήρχε ουσιώδης διαφορά στις αντιστοιχίες και την αποδοτικότητα των μεθόδων εκπαίδευσης σε περίπτωση που τις χρησιμοποιούσαμε σε άλλες τοπολογίες δικτύων.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα νευρωνικά δίκτυα αποτελούν, δυνητικά, πανίσχυρα υπολογιστικά εργαλεία, άτινα δύνανται να επιλύσουν προβλήματα ανυπέρβλητης δυσκολίας για τους συμβατικούς αλγορίθμους. Η αναγνώριση χειρόγραφων χαρακτήρων αποτελεί μια μόνο από τις εφαρμογές αυτού του είδους, με την εκπαίδευση ενός νευρωνικού δικτύου για την υλοποίηση τέτοιας εφαρμογής μπορεί να γίνει και σε κλάσματα του δευτερολέπτου.

Το πάντρεμα συμβατικών μεθόδων με νευρωνικά δίκτυα, όπως στην εφαρμογή μας, φαίνεται να έχει μεγαλύτερη επιτυχία στην επίλυση ορισμένων προβλημάτων από τη χρήση συμβατικών αλγορίθμων μόνο, ενώ θεωρείται βέβαιο ότι στο μέλλον ακόμα και τα προβλήματα που προς το παρόν δυσκολεύουν και τα νευρωνικά δίκτυα (π.χ. αναγνώριση μη-καθαρογραμμένων ή καλλιγραφικών χαρακτήρων) θα επιλυθούν με κάποιο συνδυασμό τέτοιου δικτύου με προηγμένους αλγορίθμους.

Υπάρχει μεγάλη πληθώρα αρχιτεκτονικών και μεθόδων εκπαίδευσης των τεχνητών νευρωνικών δικτύων, με μεγάλη ποικιλία στη πολυπλοκότητα και την ευκολία εκπαίδευσης. Ενώ μια εκ των προβλέψεών μας ήταν πως η αποτελεσματικότητα θα ήταν ευθέως ανάλογη της πολυπλοκότητας, στη πράξη είδαμε πως για τους σκοπούς της εφαρμογής αυτό δεν ισχύει.

Από το πειραματικό μέρος κατέστη πασιφανές πως οι καταλληλότερες, από συνδυαστική σκοπιά χρόνου εκπαίδευσης και αριθμού σφαλμάτων, μέθοδοι και αρχιτεκτονικές νευρωνικών δικτύων για την αναγνώριση χαρακτήρων ήταν οι Resilient backpropagation και Levenberg Markquardt για τα εμπροσθοτοτροφοδοτούμενα δίκτυα, αλλά κυρίως η πιθανοκρατική αρχιτεκτονική με τη δική της μέθοδο εκπαίδευσης.

Θα ίσχυε κάτι άλλο για πιο σύνθετες εφαρμογές; Πιθανόν. Είναι ωστόσο πασιφανές πως τα νευρωνικά δίκτυα είναι καταλληλότερα από τους συμβατικούς αλγορίθμους για την επίλυση σύνθετων προβλημάτων, καθώς και ότι δεν μπορούν να γινουν ακριβείς προβλέψεις για την αποτελεσματικότητα ενός αλγορίθμου από τη μαθηματική του πολυπλοκότητα και μόνο.

8. ΠΗΓΕΣ - ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Simon Haykin, 2008, Neural networks - A comprehensive foundation

[2] Ι. Βλαχαβάς, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κοκκορας, Η. Σακελλαρίου, 2006, Τεχνητή Νοημοσύνη

[3] Fausett Laurene, 1993, Fundamentals of Neural Networks: Architectures, Algorithms And Applications

[4] B. Krose, P. van der Smagt, 1996, An introduction to Neural Networks

[5] The industrial electronics handbook-Intelligent systems - B.M. Wilamowski, J.D. Irwin

8.2. ΙΣΤΟΣΕΛΙΔΕΣ

[1] https://en.wikipedia.org/wiki/Artificial_neuron

[2] https://de.wikipedia.org/wiki/K%C3%BCnstliches_Neuron

[3] https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz

[4] https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

[5] https://sv.wikipedia.org/wiki/Hebbs_teori

[6] https://en.wikipedia.org/wiki/Hebbian_theory

[7] https://el.wikipedia.org/wiki/Perceptron

[8] https://de.wikipedia.org/wiki/Perzeptron

[9] https://de.wikipedia.org/wiki/Backpropagation

[10] https://en.wikipedia.org/wiki/Backpropagation

[11] http://kelifos.physics.auth.gr/COURSES/neural/K5.pdf

[12] https://appliedgo.net/perceptron/

[13] http://basic-eng.blogspot.gr

[14] https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/8676/versions/1/previews/edu_imgcrop.m/index.html?access_key=

[15] https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/8676/versions/1/previews/edu_imgpreprocess.m/index.html?access_key=

[16] http://chamarisilva.blogspot.gr/2015/04/remove-extra-white-space-and-crop-image.html

[18] http://www.icsd.aegean.gr/lecturers/kavallieratou/NN&EP_files/ci_8.pdf

[19] http://fourier.eng.hmc.edu/e161/lectures/nn/node13.html

[20] https://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0ahUKEwjj0KXUv_7WAhWECBoKHYPLAdoQFgg8MAM&url=https%3A%2F%2Fwww.researchgate.net%2Fpublication%2F250310836_Artificial_neural_networks_in_medical_diagnosis&usg=AOvVaw0pXkKAYE1rgLVadj7Vqazf

[21] https://www.technologyreview.com/s/541276/deep-learning-machine-teaches-itself-chess-in-72-hours-plays-at-international-master/

[22] http://toritris.weebly.com/perceptron-2-logical-operations.html

[23] http://www.cs.bham.ac.uk/~pxt/NC/l5_JB.pdf

[24] http://www.oocities.org/hjayathilake/an5.html

[25] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.451.1103&rep=rep1&type

=pdf

[26] https://www.researchgate.net/publication/250310836_Artificial_neural_networks_in_medical_diagnosis

[27] https://www.extremetech.com/extreme/233746-ai-beats-doctors-at-visual-diagnosis-observes-many-times-more-lung-cancer-signals

[28] http://aass.oru.se/~lilien/ml/seminars/2007_03_12c-Markus_Ingvarsson-RPROP.pdf

9. ΠΑΡΑΡΤΗΜΑ

Παρατίθενται οι κώδικες όπως στα αρχεία, με αφαιρεμένα τα περιττά σχόλια (π.χ., αυτά που μπαίνουν αυτόματα από το matlab). Οι πλήρεις και αξιοποιήσιμες μορφές τους υπάρχουν στα αρχεία.

9.1. Το interface.m

```
function varargout = interface(varargin)
% INTERFACE MATLAB code for interface.fig


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interface_OpeningFcn, ...
                   'gui_OutputFcn',  @interface_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end


% --- Executes just before interface is made visible.
function interface_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
end

% --- Outputs from this function are returned to the command line.
function varargout = interface_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
global datab dataa; %oi arxikes times tou listbox twn methodwn.
datab = get(handles.TrainingMethodList,'String'); %xrhsimopoieitai sto function
listbox3
dataa = 'default method'; % epishs function listbox3, gia ta probabilistic
end

% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
filename = uigetfile({'*.*','All Files'});
global TrainingSample
TrainingSample = imread(filename);
axes(handles.axes1);
imshow(TrainingSample); %emfanizei thn eikona sto axes1
end

% --- Executes on button press in CreateAndTrain.
function CreateAndTrain_Callback(hObject, eventdata, handles)
```

```
global NumOfTrainingLines NumOfChars NumOfNeurons NumOfLayers TotalLines
global net c tr MaxEpochs;
global TrainingSample T  P per
global  EPILOGH EPILOGHMETHODOU EPILOGHDIKTUOU Method1  MaxTime
ChosenStrelNum
NumOfTrainingLines = str2num(get(handles.edit4,'String'));
TotalLines = str2num(get(handles.edit5,'String'));
NumOfNeurons = str2num(get(handles.NumOfNeurons,'String'));
NumOfLayers = str2num(get(handles.NumOfNeurons2,'String'));
MaxEpochs = str2num(get(handles.MaxEpochs,'String'));
MaxTime = str2num(get(handles.MaxTrainingTimeInSeconds1,'String'));
if NumOfLayers==[]
   NumOfLayers=0;
end
%prohgeitai to pathma tou "OK" (vl. function pushbutton7)
tf=strcmp(EPILOGH,'NUMBERS');
tp= strcmp(EPILOGH,'ALPHABET');
tc= strcmp(EPILOGH,'ALL');
NUMBER1=TotalLines - NumOfTrainingLines;
%To TargetD xrhsimopoieitai apokleistika gia ton elegxo ths apodoshs tou N.D.
if (tf)
   NumOfChars=10; TargetD=repmat(eye((10)),[1 (NUMBER1)]); %gia arithmous
elseif (tc) %arxika tp
   NumOfChars=40;  TargetD=repmat(eye((40)),[1 (NUMBER1)]); %gia arithmous,
grammata, kai 4 shmeia stikshs
else
   NumOfChars=26; TargetD=repmat(eye((26)),[1 (NUMBER1)]); %gia latiniko
alfavhto h mh epilogh
end

%Sthrizetai ston prokathorismeno algorithmo anagnwshs (vl. kefalaio 5.1.), opou
kathe
%xarakthras antistoixizetai se mia grammh apoteloumenh apo enan asso se
```

```
%kapoia thesh kai mhdenika stis apodeloipes.
if (strcmp(EPILOGHMETHODOU,'BFGS quasi-Newton backpropagation'))
        Method1='trainbfg';
elseif (strcmp(EPILOGHMETHODOU,'Levenberg-Markquardt with Bayesian
normalization'))
        Method1='trainbr';
elseif(strcmp(EPILOGHMETHODOU,'Gradient descent'))
        Method1='traingd';
elseif(strcmp(EPILOGHMETHODOU,'Resilient backpropagation'))
        Method1='trainrp';
else
        Method1='trainlm';
end
ExpectedNum=NumOfChars*TotalLines; %o sunolikos arithmos xarakthrwn sthn
eikona
[num, out, ImageEdges11, Ifill11, BinaryImage11, GrayImage11, ChosenStrelNum] =
function1(TrainingSample, NumOfChars, ExpectedNum, 1, 3); %GIA FUNCTION1
G=(NumOfChars*NumOfTrainingLines);
P = out(:,1:G);
if (0 == get(handles.HoldValues, 'Value'))
set(handles.edit4,'String','NumberOfTrainingLines');
set(handles.edit5,'String','TotalNumberOfLines');
set(handles.NumOfNeurons,'String','NumberOfNeurons');
set(handles.NumOfNeurons2,'String','SecondLayerNeurons (optional)');
set(handles.MaxEpochs,'String','MaxNumberOfEpochs');
set(handles.MaxTrainingTimeInSeconds1,'String','MaxTrainingTimeInSeconds');
end
T=repmat(eye((NumOfChars)),[1 (NumOfTrainingLines)]); %antistoixhsh gia ta
traininglines
Ptest = out(:,G+1:num);

sunthiki=0; %an ginei 1 xeirokinhta ekpaidevei kalutera ta diktua gia
if (sunthiki==1) %anagnwrish kefalaiwn latinikwn
```

```
[X,T1] = prprob; %sumvatikh methodos dhmiourgias kefalaiwn xarakthrwn gia
nevrwnika diktua
numNoise = 25;
Xn = min(max(repmat(X,1,numNoise)+randn(35,26*numNoise)*0.2,0),1);
Tn = repmat(T1,1,numNoise);
P=cat(2,P,Xn);
T=cat(2,T,Tn);
end
[perr, case1, net, tr] = CreateNetwork(P,T, NumOfNeurons, NumOfLayers, Method1,
EPILOGHDIKTUOU, NumOfChars, MaxEpochs, MaxTime);
if (case1==1 || case1==2) && NUMBER1>0  %gia kathgories diktuwn (1) kai (2)
    per=perform(net, TargetD, net(Ptest)); %performance gia to Ptest gia to pinaka
TargetD
elseif NUMBER1==0 %an oles oi grammes tou TrainingSample xrhsimopoiountai gia
ekpaideush
    per=perr;  %performance gia thn anagnwrish twn xarakthrwn tous (mikroterh
aksiopistia)
end
%ginotan kai me mse anti gia perform, alla ta apotelesmata einai idia
set(handles.ShowMSE, 'String', per);
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');

    str=get(edit2,'String');
if isempty(str2num(edit2))
    set(edit2,'string','0');
    warndlg('Input must be numerical');
```

```
    end
end
end
% --- Executes on button press in FindChar.
function FindChar_Callback(hObject, eventdata, handles)
%euresh tou cropped character apo to ValidationSample
global y NumOfChars lett net ChosenStrelNum;
%y einai to cropped letter, kai orizetai sto function LoadValidationSample1
[num, lett, ImageEdges, Ifill, BinaryImage, GrayImage, StrelNum] = function1(y, 1, 1,
0, 3);
%ekei pou tha htan o pinakas "out", einai twra enas memonwmenos xarakthras
"lett".
%Katopin dinetai to apotelesma ths anagnwshs tou "lett" apo to "net":
[a,b]=max(sim(net,lett));  %pou ekxwreitai to "b"
axes(handles.LettAxes);
plotchar(lett);
axes(handles.IfillAxes);
imshow(Ifill);
axes(handles.GrayImageAxes);
imshow(GrayImage);
axes(handles.BinaryImageAxes);
imshow(BinaryImage);
[str1,c] = ToAlphabet(b, NumOfChars); %eisagwgh apotelesmatos "b" me to an einai
alfavhto h arithmoi
set(handles.DetectedLetter, 'String', c); %emfanish tou antistoixou xarakthra dipla
ap'to koumpi
end

% --- Executes on button press in CheckValidationSampleButton.
function CheckValidationSampleButton_Callback(hObject, eventdata, handles)
global ValidationSample T lett2 net NumOfChars ChosenStrelNum;
var1 = str2num(get(handles.NumOfCharsPerLine1,'String'));
[num, lett2, ImageEdges, Ifill, BinaryImage, GrayImage, StrelNum] =
```

```
function1(ValidationSample, var1, var1, 0, ChosenStrelNum);
[a,b]=max(sim(net,lett2)); %sugkrish gia to ValidationSample
[str1,c] = ToAlphabet(b, NumOfChars); %metatroph se keimeno
set(handles.text6, 'String', c);
end

function NumOfCharsPerLine1_Callback(hObject, eventdata, handles)
end
% --- Executes during object creation, after setting all properties.
function NumOfCharsPerLine1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function edit4_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function edit5_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on selection change in ModeListbox.
function ModeListbox_Callback(hObject, eventdata, handles)
global String EPILOGH
contents = cellstr(get(hObject,'String'));
EPILOGH = contents{get(hObject,'Value')};
end

% --- Executes during object creation, after setting all properties.
function ModeListbox_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function NumOfNeurons_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function NumOfNeurons_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

function NumOfNeurons2_Callback(hObject, eventdata, handles)
end
```

```
% --- Executes during object creation, after setting all properties.
function NumOfNeurons2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in LoadValidationSample1.
function LoadValidationSample1_Callback(hObject, eventdata, handles)
filename = uigetfile({'*.*','All Files'});
global ValidationSample TrainingSample y
ValidationSample = imread(filename);
axes(handles.axes4);
imshow(ValidationSample);
while(1)
axes(handles.axes4); %gia na ginei crop sto axes4
y = imcrop(ValidationSample);
axes(handles.CroppedPic);
imshow(y); %emfanizei to y sto CroppedPic
axes(handles.axes4); %gia peraiterw crops
end
end

% --- Executes on selection change in TrainingMethodList.
function TrainingMethodList_Callback(hObject, eventdata, handles)
global EPILOGHMETHODOU EPILOGHDIKTUOU
contents = cellstr(get(hObject,'String'));
EPILOGHMETHODOU = strtrim(contents{get(hObject,'Value')}); %xwris strtrim exei
thema me tis epipleon grammes
end
% --- Executes during object creation, after setting all properties.
```

```
function TrainingMethodList_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end


% --- Executes on selection change in TypeOfNetworkList.
function TypeOfNetworkList_Callback(hObject, eventdata, handles)
% Hints: contents = cellstr(get(hObject,'String')) returns TypeOfNetworkList contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from TypeOfNetworkList

global EPILOGHDIKTUOU
global datab dataa
contents = cellstr(get(hObject,'String'));
EPILOGHDIKTUOU = strtrim(contents{get(hObject,'Value')});

%kapoia diktua xrhsimopoioun sugkekrimenes methodous ekpaideushs
%p.x. ta probabilistic exoun dikh tous. Oi parakatw grammes tropopoioun ta
%listboxes se pragmatiko xrono:

if strcmp(EPILOGHDIKTUOU,'Probabilistic feedforward network') %arxika sto
listbox3
    set(handles.TrainingMethodList,'Value',1);  %xwris mia arxikopoihsh me integer
value emfanizei
    %"Warning: single-selection listbox control requires that Value be an
    %integer within String range". Aksioshmeiwto oti to emfanizei afotou
    %exei ekpaideutei opoiodhpote diktuo, ektos an ekpaideutei me aplh
    %levenberg-markquardt methodo.

    set(handles.TrainingMethodList,'String','Default method'); %allagh tou listbox
```

```
    set(handles.NumOfNeurons,'Visible','off') %mh-tropopoihsimes parametroi
    set(handles.NumOfNeurons2,'Visible','off') %sta pithanokratika diktua
    set(handles.MaxTrainingTimeInSeconds1,'Visible','off')
    set(handles.MaxEpochs,'Visible','off')

else
    set(handles.TrainingMethodList,'String',datab);
    set(handles.NumOfNeurons,'Visible','on')
    set(handles.NumOfNeurons2,'Visible','on')
    set(handles.MaxTrainingTimeInSeconds1,'Visible','on')
    set(handles.MaxEpochs,'Visible','on')
end



end


% --- Executes during object creation, after setting all properties.
function TypeOfNetworkList_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end


function MaxTrainingTimeInSeconds1_Callback(hObject, eventdata, handles)

end
% --- Executes during object creation, after setting all properties.
function MaxTrainingTimeInSeconds1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
end

function MaxEpochs_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function MaxEpochs_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in AlternativeMethod.
function AlternativeMethod_Callback(hObject, eventdata, handles)
%openfig('MixedOCR.fig');
File1= fullfile('MixedOCR.m');
run(File1);
end

% --- Executes on button press in HoldValues.
function HoldValues_Callback(hObject, eventdata, handles)
end
```

9.2. H MixedOCR.m

```
function varargout = MixedOCR(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @MixedOCR_OpeningFcn, ...
                'gui_OutputFcn',  @MixedOCR_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before MixedOCR is made visible.
function MixedOCR_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

end
% --- Outputs from this function are returned to the command line.
function varargout = MixedOCR_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
end

% --- Executes on button press in SAMPLE.
function SAMPLE_Callback(hObject, eventdata, handles)
```

```
filename = uigetfile({'*.*','All Files'});
global ValidationSampleMethod2 %An htan sketo ValidationSample tha
dhmiourgouse thema sto axes4
ValidationSampleMethod2 = imread(filename); %logw tou oti einai global
axes(handles.axes22);
imshow(imresize(ValidationSampleMethod2,10));
end

function edit1_Callback(hObject, eventdata, handles)
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get

(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in SAVE.
function SAVE_Callback(hObject, eventdata, handles)
global c outstring string1
FileName=input('Insert file name:','s');
fid=fopen(FileName,'w');
fprintf(fid,'%s\n',char(string1)); %an to arxeio vrisketai se thesh sthn opoia

xreiazetai adeia admin
%gia tropopoihseis, DEN doulevei para mono an ginei run as administrator
fclose(fid);
end

% --- Executes on button press in RecButton.
```

```
function RecButton_Callback(hObject, eventdata, handles)

global ValidationSampleMethod2 lett2 b net NumOfChars c outstring string1 stre1
   lett2 = functionZ(ValidationSampleMethod2); %h enallaktikh methodos
    sizelett2=size(lett2);
    sizelett2=sizelett2(:,2);
    string1='';

     for i=1:sizelett2 %gia kathe leksh ksexwrista
     lett22 = lett2{1,i};
     [a,b]=max(sim(net,lett22)); %anagnwrish twn xarakthrwn
     [str1,c] = ToAlphabet(b, NumOfChars); %antistoixish
     if (~(isequal(char(c),' ')) && ~isempty(char(c)))
     string1=[string1 ' ' char(c)]; %prosthikh leksewn mia pros mia
     end
     end

     string1=cellstr(string1);
     pos=get(handles.edit1,'Position');
     ht = uicontrol('Style','Text','Position',pos);
     string1 = regexprep(string1,' +',' '); %sugxwneush kenwn me regular

expression
     outstring = textwrap(ht,string1,64) %megisto diathesimo ana grammh gia auto

to megethos
     set(handles.edit1, 'Max', 64); %xwris auto eksafanizetai-authaireth timh
     set(handles.edit1,'String',outstring);

     %logw kapoiou bug tou matlab, an mia leksh den diavastei apo to OCR
     %(sunhthws kapoia me mono enan xarakthra) kai to 'c' meinei keno, sto
     %edit1 emfanizetai h ekastote grammh mono mexri to shmeio pou
     %emfanizetai diplo h triplo keno, kai meta paei kateutheian sthn
```

```
    %epomenh. To apotelesma einai mises protaseis. Logw kapoias
    %kwdikopoihshs (pithanotata), den diorthwnetai oute me regular
    %expression, oute me th sunthiki if sth grammh 141 (h opoia menei ws exei

prolhptika).
    %To provlhma prolamvanetai me th prwth 'if' tou ToAlphabet
end


% --- Executes on button press in ShowPos.
function ShowPos_Callback(hObject, eventdata, handles)

global ValidationSampleMethod2
ImageSize=size(ValidationSampleMethod2);
if (ImageSize(1,1)<200) %oi mikres eikones diavazontai pio duskola apo to matlab

ocr
ValidationSampleMethod2=imresize(ValidationSampleMethod2,5); %an ginoun polu

megales mporei na uparksei kai tote provlhma
end
res = ocr(ValidationSampleMethod2);
numofwords = size(res.Words);
numofwords = numofwords(1,1);

for n=1:numofwords
G1 = res.WordBoundingBoxes(n,:);
P=imcrop(ValidationSampleMethod2, [G1(:,1), G1(:,2), (G1(:,3)), (G1(:,4))] );
P=im2double(P);

[sizex, sizey, sizez]= size(P); %gia na xrhsimopoieitai akrivws idia methodos me

th functionZ
```

```
blank = ones(sizex+40,sizey+40,sizez);
    for i=1:sizex
        i1=i+10;
        for j=1:sizey
            j1=j+10;
            for d=1:sizez
                blank(i1,j1,d)=P(i,j,d);
            end
        end
    end
    name1=int2str(n);
 figure('Name',name1); imshow(blank)%me onoma ton arithmo n gia evresh tous apo

th closefigs
res2=ocr(blank);
str1=res2.Text;
str1=strtrim(str1);
numofchars = size(str1);
numofchars = (numofchars(1,2));
G2 = res2.CharacterBoundingBoxes(:,:);

for p = 1:length(G2)
rectangle('position',G2(p,:),'edgecolor','b');
end
end
end

% --- Executes on button press in CloseFigs.
function CloseFigs_Callback(hObject, eventdata, handles)
global ValidationSampleMethod2

res = ocr(ValidationSampleMethod2);
numofwords = size(res.Words);
```

```
for n=1:numofwords(1,1)
%name1=int2str(n);
delete(findall(0,'Name',int2str(n))); %an htan (0,'Type',figure), tha ekleinan

kai ta parathura
end
bdclose('all');
end
```

9.3. H function1.m

```
function [y,z,ImageEdges,Ifill, BinaryImage, GrayImage, StrelNum] = function1(I,
Numofchars1, ExpectedNum, TrainOrVal, StrelNum)

%vasismeno se entoles pou didaxthikame sta plaisia ths sxolhs, kathws kai
%me th vohtheia algorithmwn twn sundesmwn [13] ws [16]

GrayImage = rgb2gray(I); %metatroph se gray
BinaryImage = im2bw(GrayImage,graythresh(GrayImage));
NumOfTries=1;
OK1=0;
ImageEdges = edge(uint8(BinaryImage)); %metatroph se binary me vash to
threshold, kai entopismos edges kai dhmiourgia eikona me vash auta
if(StrelNum==0)
StrelNum=3;
End

if (TrainOrVal==0) %an einai validation sample h cropped image
OK1=1;
str1 = strel('square',StrelNum);  %dhmiourgia strel element me platos StrelNum
pixels
Ifill= imfill((imdilate(ImageEdges, str1)),'holes'); %diastolh twn gwniwn (edges), kai
gemizei ta kena stous xarakthres ths dilated eikonas
```

```
[Var1, num] = bwlabel(Ifill); %metatrepei to Ifill se double
end

while (OK1==0 && NumOfTries<10)
str1 = strel('square',StrelNum);  %dhmiourgia strel element me platos StrelNum
pixels
%kanonika htan 3, alla auto dhmiourgouse provlhma me ta erwthmatika ths
%eikonas "ALL", logw tou oti anagnwrizontan san mia teleia kai mia kampulh,
%enw mse strel 5-6 enwnontai.

Ifill= imfill((imdilate(ImageEdges, str1)),'holes'); %diastolh twn gwniwn (edges), kai
gemizei ta kena stous xarakthres ths dilated eikonas
[Var1, num] = bwlabel(Ifill); %metatrepei to Ifill se double

if (num==ExpectedNum)
OK1=1;
else  %an logw strel vgainei allo noumero apo to anamenomeno
StrelNum=StrelNum+1;
NumOfTries=NumOfTries+1;%epanalhpsh ths diadikasias
end
end

if(OK1==1)
regprops = regionprops(Var1); %theseis xarakthrwn sthn eikona

regbox = [regprops.BoundingBox]; %aksiopoihsh tou pediou "Bounding box"
regbox = reshape(regbox,[4 (num)]);

Cent = [regprops.Centroid]; %Aksiopoihsh tou pediou "Centroid"
Cent = reshape(Cent,[2 (num)]);
Cent = Cent';
Cent(:,3) = 1:num;
```

```
% Extra lines compare to example2 to extract all the components into an
% cell array
Cent2 = sortrows(Cent,2);
%num2=10;

for cnt = 1:(fix(num/Numofchars1)) %sort apo aristera pros ta deksia
    Cent2((cnt-1)*Numofchars1+1:cnt*Numofchars1,:) = sortrows(Cent2((cnt-
1)*Numofchars1+1:cnt*Numofchars1,:),3);
end

%Cent3 = Cent2(:,1:2);
ind = Cent2(:,3);   %arxika 4

for cnt = 1:num
    img{cnt} = imcrop(BinaryImage,regbox(:,ind(cnt)));  %(:,ind(cnt)));
end

for cnt = 1:num
bw = img{cnt};

% Find the boundary of the image
[y2temp, x2temp] = size(bw);
x1=1;
y1=1;
x2=x2temp;
y2=y2temp;

% Finding left side blank spaces
cntB=1;
while (sum(bw(:,cntB))==y2temp)
    x1=x1+1;
    cntB=cntB+1;
end
```

```
% Finding right side blank spaces
cntB=1;
while (sum(bw(cntB,:))==x2temp)
    y1=y1+1;
    cntB=cntB+1;
end

% Finding upper side blank spaces
cntB=x2temp;
while (sum(bw(:,cntB))==y2temp)
    x2=x2-1;
    cntB=cntB-1;
end

% Finding lower side blank spaces
cntB=y2temp;
while (sum(bw(cntB,:))==x2temp)
    y2=y2-1;
    cntB=cntB-1;
end
% Crop the image to the edge
bw2=imcrop(bw,[x1,y1,(x2-x1),(y2-y1)]);

bw_7050=imresize(bw2,[70,50]);   %KANONIKA BW2
for cntt=1:7
    for cnt2=1:5
        Atemp=sum(bw_7050((cntt*10-9:cntt*10),(cnt2*10-9:cnt2*10)));
        lett((cntt-1)*5+cnt2)=sum(Atemp);
    end
end
lett=((100-lett)/100);
lett=lett';
```

```
out(:,cnt)=lett;
end

else
    disp('Recognition error: The image cannot be used as a training sample');
end

y = num;
z = out;
ImageEdges;
Ifill;
StrelNum;
```

9.4. H funtionZ.m

```
function z = functionZ(I)

ImageSize=size(I);
if (ImageSize(1,1)<200) %oi mikres eikones diavazontai pio duskola apo to matlab
ocr
I=imresize(I,5); %an ginoun polu megales mporei na uparksei kai tote provlhma
end
res = ocr(I);
numofwords = size(res.Words);
numofwords = numofwords(1,1);

out=[];
out1={};
cc=load('KENO-OCR.mat');
cd=load('KENO-OCR2.mat');
cc=cc.char1;
cd=cd.char1;
OK1=0;
```

```
for n=1:numofwords
     G1 = res.WordBoundingBoxes(n,:);

   %if (n==1)
    %   P=imcrop(I, [G1(:,1)-4, G1(:,2)-4, (G1(:,3))+8, (G1(:,4)+8)] );
   %else
    %   P=imcrop(I, [G1(:,1)-12, G1(:,2)-12, (G1(:,3))+24, (G1(:,4)+24)] );
   %end
   P=imcrop(I, [G1(:,1), G1(:,2), (G1(:,3)), (G1(:,4))] );
   P=im2double(P);
   [sizex, sizey, sizez]= size(P);
   blank = ones(sizex+40,sizey+40,sizez); %keno upovathro megaluterou
megethous
   for i=1:sizex %gia na mporoun na entopistoun apo to OCR
      i1=i+10;
      for j=1:sizey
         j1=j+10;
         for d=1:sizez
            blank(i1,j1,d)=P(i,j,d); %copy-paste twn pixels sto keno upovathro
         end
      end
   end

   res2=ocr(blank);
   str1=res2.Text;
   str1=strtrim(str1); %apaloifh perittwn kenwn
   numofchars = 0;
   k=0;
   numofchars = size(str1);
   numofchars = (numofchars(1,2));
   %ARXIKA 1,1
   out=[];
```

```
    for k=1:numofchars
        G2 = res2.CharacterBoundingBoxes(k,:);
        char1 = imcrop(blank,[G2(:,1), G2(:,2), (G2(:,3)), (G2(:,4))] );

        %oi katwthi if aposkopoun sthn agnohsh twn kenwn
        if((isequal(size(cc), size(char1)) || (isvector(cc) && isvector(char1) && numel(cc)
== numel(char1)))==1)
            if(cc~=char1)
                OK1=1;
            end
         elseif ((isequal(size(cc), size(char1)) || (isvector(cc) && isvector(char1) &&
numel(cc) == numel(char1)))==0)
                OK1=1;
        elseif ((isequal(size(cd), size(char1)) || (isvector(cd) && isvector(char1) &&
numel(cd) == numel(char1)))==1)
            if(cd~=char1)
                OK1=1;
            end
         elseif ((isequal(size(cd), size(char1)) || (isvector(cd) && isvector(char1) &&
numel(cd) == numel(char1)))==0)
                OK1=1;
        else
                OK1=0;
        end


        if(OK1)

        char1 = rgb2gray(char1);
        bw = im2bw(char1,graythresh(char1));

        [y2temp x2temp] = size(bw);
x1=1;
```

```
y1=1;
x2=x2temp;
y2=y2temp;
if(bw ~= 1)


%oi parakatw grammes vasizontai se algorithmous twn sundesmwn [13]-[16]


% Finding left side blank spaces
cntB=1;
while (sum(bw(:,cntB))==y2temp)
    x1=x1+1;
    cntB=cntB+1;
end

% Finding right side blank spaces
cntB=1;
while (sum(bw(cntB,:))==x2temp)
    y1=y1+1;
    cntB=cntB+1;
end

% Finding upper side blank spaces
cntB=x2temp;  %arxika x2temp
while (sum(bw(:,cntB))==y2temp)
    x2=x2-1;
    cntB=cntB-1;
end

% Finding lower side blank spaces
cntB=y2temp; %arxika y2temp
while (sum(bw(cntB,:))==x2temp)
```

```
    y2=y2-1;
    cntB=cntB-1;
end
end
% Crop the image to the edge
gray1=imcrop(bw,[x1,y1,(x2-x1),(y2-y1)]);

        bw_7050=imresize(gray1,[70,50]);   %KANONIKA BW2
        for cntt=1:7
            for cnt2=1:5
             Atemp=sum(bw_7050((cntt*10-9:cntt*10),(cnt2*10-9:cnt2*10)));
             lett((cntt-1)*5+cnt2)=sum(Atemp);
            end
        end

        lett=((100-lett)/100);
        lett=lett';
        out(:,k)=lett;
        %figure; plotchar(out(:,k));
       end
out1{n}=out;

    end
end

   z=out1;
```

9.5. Η ToAlphabet.m

```
function  [str1,c] = ToAlphabet(b, NumOfChars)

c = length(b);
```

```
if(c==0)
c=' '; %an auto to kommati diagrafei tha uparxei provlhma sto edit text tou
mixedOCR
end %vl. sxetika sxolia sto MixedOCR
str1={};
if (NumOfChars==26)
    Start1=1;
    End1=length(b);
    allowed=1;
elseif(NumOfChars==10)
    c='';
    for i=1:length(b)
    J=(int2str(b(i)-1));
    c(i) = (J(1,1)); %amesh antistoixish gia ta noumera, alla an to 0 einai meta to 9 to
thewrei ws 10
   % if i==10
    %    c(i)='0'; %thewroume autodikaiws oti se ola ta training sets to 0 einai meta to
9 (vl. odhgies)
    %end
    end
    allowed=0;
else
    Start1=1;
    End1=length(b);
    allowed=1;
end

if allowed==1;
    for i=Start1:End1
        if b(i)==1
            c(i)='A';
       elseif b(i)==2
            c(i)='B';
```

```
elseif b(i)==3
    c(i)='C';
elseif b(i)==4
    c(i)='D';
 elseif b(i)==5
    c(i)='E';
 elseif b(i)==6
  c(i)='F';
 elseif b(i)==7
 c(i)='G';
 elseif b(i)==8
 c(i)='H';
 elseif b(i)==9
 c(i)='I';
 elseif b(i)==10
 c(i)='J';
 elseif b(i)==11
 c(i)='K';
 elseif b(i)==12
 c(i)='L';
 elseif b(i)==13
 c(i)='M';
 elseif b(i)==14
 c(i)='N';
 elseif b(i)==15
 c(i)='O';
 elseif b(i)==16
 c(i)='P';
 elseif b(i)==17
 c(i)='Q';
 elseif b(i)==18
 c(i)='R';
 elseif b(i)==19
```

```
c(i)='S';
elseif b(i)==20
c(i)='T';
elseif b(i)==21
c(i)='U';
elseif b(i)==22
c(i)='V';
elseif b(i)==23
c(i)='W';
elseif b(i)==24
c(i)='X';
 elseif b(i)==25
c(i)='Y';
elseif b(i)==26
c(i)='Z';
elseif b(i)==27
c(i)='0';
elseif b(i)==28
c(i)='1';
elseif b(i)==29
c(i)='2';
elseif b(i)==30
c(i)='3';
elseif b(i)==31
c(i)='4';
elseif b(i)==32
c(i)='5';
elseif b(i)==33
c(i)='6';
elseif b(i)==34
c(i)='7';
elseif b(i)==35
c(i)='8';
```

```
        elseif b(i)==36
        c(i)='9';
        elseif b(i)==37
        c(i)='?';
        elseif b(i)==38
        c(i)='!';
        elseif b(i)==39
        c(i)='.';
        elseif b(i)==40
        c(i)=',';
         end
    end
end

    c = char(c);   %arxika string
    T=length(b)/NumOfChars; %o arithmos twn character sets
   for i=1:T
    j=((i-1)*NumOfChars)+1;
    jj=j+(NumOfChars-1);
    str1{i}=c(j:jj); %leksh gia kathe alfavhto
   end
c = strtrim(c); %afairesh kenwn

c;
str1 ;

end
```

9.6. Η CreateNetwork.m

```
function [perr, case1, net, tr] =
CreateNetwork(P,T,neurons,SecondLayerNeurons,Method,EPILOGHDIKTUOU,Nu
```

```
mOfChars, MaxEpochs, MaxTime)

%gia th periptwsh mh-apodekths timhs
if (~isnumeric(MaxEpochs) || (isequal(MaxEpochs,0)) || ~isscalar(MaxEpochs))
    MaxEpochs=100000;
end

%antistoixa
if ((~isnumeric(MaxTime)) || (isequal(MaxTime,0)) || ~isscalar(MaxTime))
    MaxTime=Inf;
end

%ena h duo hidden-layers
if SecondLayerNeurons==0
    neur=[neurons];
else
    neur=[neurons SecondLayerNeurons]; %gia diktua duo epipedwn
end

if (strcmp(EPILOGHDIKTUOU,'Feedforward network'))
    net = feedforwardnet(neur); case1=1;
elseif (strcmp(EPILOGHDIKTUOU,'Feedforward PatternNet'))
    net = patternnet(neur); case1=1;
elseif (strcmp(EPILOGHDIKTUOU,'Layer recurrent network'))
    net= layrecnet(1:2,neur); case1=1;
elseif(strcmp(EPILOGHDIKTUOU,'Probabilistic feedforward network'))
    net=newpnn(P,T); case1=2; %den lamvanontai upopsin oi methodoi ekpaideushs
    view(net)
elseif(strcmp(EPILOGHDIKTUOU,'Cascade forward network'))
    net = cascadeforwardnet(neur); case1=1;
else
    net = feedforwardnet(neur); case1=1;
end
```

```
if case1==1 %ta diktua me case=1 xrhsimopoioun diaforetikes parametrous apo ta
alla
net.trainFcn = Method ;
net.trainParam.epochs = MaxEpochs;
net.trainParam.time=MaxTime;
[net1, tr]=train(net,P,T); %ekpaideush gia eisodous P kai targets T
performance=perform(net1, T, net1(P));  %apodosh tou net1 gia target T kai thn
ekdodo tou gia eisodo P
%den lamvanetai upopsin, giati sth sunarthsh tou GUI ginetai allos
%upologismos
end

if case1==2;
    net1 = net; %ekpaidevetai kata th dhmiourgia
    tr=net;

    %[net2, tr]=train(net,P,T)
    performance=perform(net1, T, net1(P));

end


net=net1;  %epistrofh ekpaideumenou diktuou
case1=case1;
perr=performance;
tr;
```