Νευρωνικά Δίκτυα για τη Λύση Συνήθων Διαφορικών Εξισώσεων

Οκτώβριος 23

Πανεπιστήμιο Κρήτης – Τμήμα Εφαρμοσμένων Μαθηματικών
Βασιλική Ρίζου

Περιεχόμενα

Κεφάλαιο 1. Μηχανική Μάθηση 3
1.1 Εισαγωγή στην Μηχανική Μάθηση 3
Κεφάλαιο 2. Η ιδέα των Τεχνητών Νευρωνικών Δικτύων με ένα παράδειγμα 6
2.1 Μηχανική Μάθηση και Νευρωνικά Δίκτυα 6
2.2 Εισαγωγή στην έννοια του τεχνητού νευρωνικού δικτύου 6
2.3 Βιολογικά Νευρωνικά Δίκτυα 7
2.4 Η δομή και η λειτουργία του Νευρώνα 8
2.5 Τεχνητό Νευρωνικό Δίκτυο- Άντληση πετρελαίου 12
2.6 Τεχνητό Νευρωνικό Δίκτυο- Γενικός κανόνας 16
2.7 Παράδειγμα Τεχνητού Νευρωνικού Δικτύου Πέντε Φάσεων 17
2.8 Μέθοδος Απότομης Καθόδου (Steepest Descent Method) 18
2.9 Μέθοδος Στοχαστικής Κλίσης (Stochastic Gradient) 19
2.10 Αλγόριθμος Οπισθοδιάδοσης (Backtracking Algorithm) 21
2.11 Διαδικασίες Μάθησης των ΤΝΔ 24
2.12 Εφαρμογές των Νευρωνικών Δικτύων 26
Κεφάλαιο 3. Τεχνητά Νευρωνικά Δίκτυα για την λύση Διαφορικών Εξισώσεων 28
3.1 Εισαγωγή 28
3.2 Περιγραφή της μεθόδου 28
3.2.1 Υπολογισμός κλίσης 29
3.3 Απεικόνιση της Μεθόδου 30
3.3.1 Λύση μιας ΣΔΕ και ενός συστήματος ΣΔΕ 31
3.3.2 Λύση μιας ΜΔΕ 32
3.4 Παραδείγματα 33
3.4.1 ΣΔΕ και Σύστημα ΣΔΕ 33
Βιβλιογραφία 40
Παράρτημα 41

Κεφάλαιο 1. Μηχανική Μάθηση

1.1 Εισαγωγή στην Μηχανική Μάθηση

Από τη στιγμή που ανακαλύφθηκαν οι ηλεκτρονικοί υπολογιστές, οι άνθρωποι αναρωτιούνται πότε θα μπορούν να αποκτήσουν τη δυνατότητα να εκμεταλλεύονται λειτουργίες και να αποκτούν ικανότητες που θα είναι χρήσιμες στην καθημερινή ζωή τους. Ενδεικτικά, φαντάζονται ρομπότ που μπορούν να εκτελούν εργασίες νοικοκυριού και αυτοκίνητα που μπορούν να οδηγούν αυτόνομα. Αυτές οι καινοτομίες αντιπροσωπεύουν μόνο μερικά από τα μηχανήματα που πολλοί έχουν φανταστεί. Εμπνευσμένοι από το επιστημονικό αυτό όραμα, ένας νέος επιστημονικός τομέας ξεκίνησε να αναπτύσσεται: η Μηχανική Μάθηση. Οι επιστήμονες συνειδητοποίησαν ότι αν ήταν δυνατόν να προγραμματιστούν τα μηχανήματα να μαθαίνουν από μόνα τους και μέσω της εμπειρίας, τότε η τεχνολογική εξέλιξη και η καθημερινότητα των ανθρώπων θα είχαν διαφορετική πορεία. Έτσι, οι ερευνητές είχαν την ιδέα να δημιουργήσουν υπολογιστές που θα μπορούσαν να μαθαίνουν από ιατρικά αρχεία ποιες θεραπείες είναι αποτελεσματικές για νέες ασθένειες, ή ακόμα και σπίτια που θα μπορούσαν να βελτιστοποιούν την ενεργειακή τους απόδοση βάσει των αναγκών των ενοίκων τους. Μέσω επιτυχημένης εκπαίδευσης των υπολογιστών, αναδύονται νέες εφαρμογές που προσφέρουν λύσεις σε ποικίλα ζητήματα που αντιμετωπίζει ο άνθρωπος στη σύγχρονη εποχή. Επιπλέον, η λεπτομερής κατανόηση των αλγορίθμων επεξεργασίας πληροφοριών για τη Μηχανική Μάθηση ίσως οδηγήσει σε βαθύτερη κατανόηση των ανθρώπινων ικανοτήτων μάθησης σε όλες τις φάσεις της ζωής.

Καθώς η Μηχανική Μάθηση εξελίσσεται ως ένας επιστημονικός κλάδος, πολλοί αξιοσημείωτοι επιστήμονες έχουν διατυπώσει διάφορες προσεγγίσεις ανάλογα με την οπτική τους, αφού αφιερώθηκαν χρόνια στην ανάπτυξη και μελέτη αυτού του πεδίου. Συνολικά, μπορούμε να πούμε πως η Μηχανική Μάθηση είναι ένας τομέας της Επιστήμης των Υπολογιστών που αναπτύχθηκε μέσα από την αναζήτηση και την μελέτη της αναγνώρισης προτύπων και της υπολογιστικής μάθησης στο πλαίσιο της Τεχνητής Νοημοσύνης. Βασίζεται σε μεθόδους και τεχνικές που δίνουν τη δυνατότητα στους υπολογιστές να "μαθαίνουν". Ο Arthur Samuel, το 1959, ορίζει τη Μηχανική Μάθηση ως "το πεδίο μελέτης που επιτρέπει στους υπολογιστές να αποκτούν γνώση χωρίς να χρειάζεται να τους προγραμματίσουμε αυστηρά". Στην πραγματικότητα, η Μηχανική Μάθηση αναφέρεται σε μια συλλογή μεθόδων που ανιχνεύουν αυτόματα μοτίβα σε δεδομένα και στη συνέχεια χρησιμοποιούν αυτά τα μοτίβα για να προβλέψουν μελλοντικά δεδομένα. Απλούστερα, εξετάζει τη μελέτη και δημιουργία αλγορίθμων που μπορούν να μαθαίνουν από μια συλλογή δεδομένων και να προβλέπουν πληροφορίες. Επιπλέον, η Μηχανική Μάθηση συνδέεται στενά με την Υπολογιστική Στατιστική, μια επιστημονική περιοχή που χρησιμοποιεί επίσης υπολογιστές για προβλέψεις. Επιπλέον, συνεργάζεται στενά με τη Μαθηματική Βελτιστοποίηση, που παρέχει μεθόδους, θεωρία και πεδία εφαρμογής. Κατά κύριο λόγο, η Μηχανική Μάθηση γίνεται απαραίτητη λόγω της ανικανότητας της παραδοσιακής προγραμματισμένης προσέγγισης να αντιμετωπίσει σύνθετες υπολογιστικές εργασίες και να σχεδιάσει αποτελεσματικούς αλγορίθμους. Παραδείγματα τέτοιων εργασιών περιλαμβάνουν μηχανές αναζήτησης, φίλτρα για την ανίχνευση ανεπιθύμητων μηνυμάτων και αναγνώριση μοτίβων.

Στον πλούσιο χώρο των επιστημόνων που εξετάζουν την πεδίο της Επιστήμης των Υπολογιστών, ειδικά όσον αφορά τη Μηχανική Μάθηση, ο Tom.M *Mitchell[5]* **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** πρότεινε μια πιο ακριβή αναπαράσταση της Μηχανικής Μάθησης. Σύμφωνα με αυτήν, η Μηχανική Μάθηση είναι η διαδικασία μέσω της οποίας ένας υπολογιστής αποκτά γνώση από την εμπειρία E σε σχέση με μια συγκεκριμένη κατηγορία εργασιών T και την αξιολόγηση της απόδοσής του με βάση το μέτρο επίδοσης P. Σε ουσία, η απόδοση του σε εργασίες της κατηγορίας T, όπως αξιολογείται με το μέτρο P, βελτιώνεται με τη συνεχή απόκτηση εμπειρίας E. Μέσω αυτού του ορισμού, ο Mitchell δημιουργεί τη βάση για μια ακριβή αντίληψη της Μηχανικής Μάθησης, αποφεύγοντας την χρήση πολύπλοκων επιστημονικών όρων. Αυτός ο ορισμός ακολουθεί την προσέγγιση του Alan Turing στο έργο του "Υπολογιστικές Μηχανές και Νοημοσύνη", όπου η ερώτηση για τη δυνατότητα των μηχανών να σκεφτούν αντικαθίσταται από το ερώτημα για τη δυνατότητά τους να εκτελούν εκείνες τις δραστηριότητες που οι άνθρωποι, ως δημιουργικά όντα, εκτελούν.

Συνοψίζοντας τα παραπάνω, κατανοούμε ότι η αποτελεσματικότητα ενός υπολογιστή, δηλαδή η ικανότητά του να εκτελεί καλύτερα εργασίες, ή η ικανότητά του να αποκτά γνώση, εξαρτάται από την εμπειρία που αποκτά μέσω αλγορίθμων κατά τη διαδικασία της μάθησης. Με βάση αυτό, η Μηχανική Μάθηση κατατάσσεται ευρέως σε τρεις κατηγορίες: Επιβλεπόμενη Μάθηση (Supervised Learning), όπου υπάρχει εμπειρία που επιβλέπεται από συγκεκριμένα δεδομένα, Μη-Επιβλεπόμενη Μάθηση (Unsupervised Learning), όπου η εμπειρία δεν είναι δομημένη με τον ίδιο τρόπο, και Ενισχυτική Μάθηση (Reinforcement Learning), όπου η εμπειρία προέρχεται από αλληλεπιδράσεις με το περιβάλλον, ανάλογα με τον τύπο της εμπειρίας που αξιοποιείται κατά τη διαδικασία της μάθησης.

Η Επιβλεπόμενη Μάθηση, αποσκοπεί στην απόκτηση γνώσης με βάση μια συγκεκριμένη αντιστοίχιση μεταξύ εισόδων (*inputs*) x και εξόδων (*outputs*) y. Αυτή η αντιστοίχιση δημιουργείται μέσω μιας γνωστής σειράς ζευγών εισόδων και εξόδων, που αναπαριστούνται από το σύνολο D = {(x_i, y_i)}, όπου i = 1, ..., N. Αυτό το σύνολο D, που αποτελείται από τα εκπαιδευτικά παραδείγματα, λέγεται σύνολο εκπαίδευσης (*training set*), και το N αναπαριστά τον αριθμό των παραδειγμάτων που ο 'δάσκαλος' χρησιμοποιεί κατά την διαδικασία της εκπαίδευσης του υπολογιστή. Στην ουσία, στην Επιβλεπόμενη Μάθηση, ο υπολογιστής δέχεται τα παραδείγματα εισόδων x και τα αντίστοιχα επιθυμητά αποτελέσματα y από τον 'δάσκαλο', με σκοπό να αναπτύξει έναν γενικό κανόνα ή αλγόριθμο, που του επιτρέπει να προβλέπει τα επιθυμητά αποτελέσματα y για νέες εισόδους. Αυτή η διαδικασία είναι παρόμοια με τον τρόπο που ένας μαθητής μαθαίνει από έναν δάσκαλο στην τάξη. Σε πιο απλή μορφή, κάθε παράδειγμα εκπαίδευσης xi αποτελεί ένα διάνυσμα με χαρακτηριστικά, όπως το ύψος και το βάρος ενός ανθρώπου. Αυτά τα χαρακτηριστικά μπορούν να είναι απλοί αριθμοί ή και πιο σύνθετες δομές όπως εικόνες, προτάσεις, χρονοσειρές ή γραφήματα. Επίσης, τα επιθυμητά αποτελέσματα yi μπορούν να είναι κατηγορηματικές μεταβλητές, δηλαδή κατηγορίες όπως 'άντρας' ή 'γυναίκα', ή πραγματικές τιμές που αναπαριστούν βαθμωτά μεγέθη όπως το εισόδημα ενός νοικοκυριού. Τα προβλήματα ταξινόμησης, όπου ο στόχος είναι να ταξινομήσουμε τις εισόδους σε διάφορες κατηγορίες, και τα προβλήματα οπισθοδρόμησης, όπου αναζητούμε μια συσχέτιση μεταξύ των εισόδων και πραγματικών τιμών εξόδου, αποτελούν βασικές περιπτώσεις σε αυτήν την κατηγορία προβλημάτων.

Στη Μη-Επιβλεπόμενη Μάθηση, το σενάριο είναι διαφορετικό. Εδώ, ο υπολογιστής λαμβάνει μόνο τις εισόδους D = {x_i}, όπου i = 1, ..., N, χωρίς να του δίνονται αντίστοιχα επιθυμητά αποτελέσματα. Ο στόχος είναι να ανακαλυφθούν ενδιαφέροντα μοτίβα ή δομές στο σύνολο δεδομένων D. Αυτή η διαδικασία ονομάζεται 'Ανακάλυψη Γνώσης'. Σε αντίθεση με την Επιβλεπόμενη Μάθηση, όπου ο 'δάσκαλος' παρέχει τα ζευγάρια εισόδων-εξόδων και ο υπολογιστής εκμεταλλεύεται αυτά για να εξάγει τον αλγόριθμο που συνδέει τις γνωστές εισόδους με τις αντίστοιχες εξόδους, στη Μη-Επιβλεπόμενη Μάθηση, ο υπολογιστής έχει μόνο τα διανύσματα εισόδων και επιδιώκει να εντοπίσει ενσωματωμένα μοτίβα ή δομές στα δεδομένα εισόδου που έχει στη διάθεσή του. Συνεπώς, στη Μη-Επιβλεπόμενη Μάθηση, η πρόκληση είναι να ανακαλύψεις αυτά τα μοτίβα χωρίς να έχεις προκαθορισμένη γνώση για το τι αναζητάς. Αυτό καθιστά τη Μη-Επιβλεπόμενη Μάθηση ένα πιο δύσκολο πρόβλημα, καθώς δεν υπάρχει συγκεκριμένο μέτρο αξιολόγησης που να είναι προφανές για την αξιολόγηση των αποτελεσμάτων. Απουσιάζει ένα κριτήριο για να μετρήσεις την απόδοση των πιθανών λύσεων που προκύπτουν από τη διαδικασία ανίχνευσης μοτίβων.

Εκτός από τις δύο κατηγορίες της Μηχανικής Μάθησης που παρουσιάστηκαν προηγουμένως, υπάρχει επίσης η Ενισχυτική Μάθηση, μια προσέγγιση λιγότερο διαδεδομένη. Η Ενισχυτική Μάθηση αναδύεται ως μια συλλογή τεχνικών, με τις οποίες ένα σύστημα μάθησης προσπαθεί να αποκτήσει γνώση μέσα από την αλληλεπίδρασή του με το περιβάλλον του. Στην ουσία, η Ενισχυτική Μάθηση αντλεί έμπνευση από τα πρότυπα της μάθησης με ανταμοιβές και κυρώσεις, όπως αυτά παρατηρούνται στη διαδικασία εκπαίδευσης των ζώων, όπως σκύλοι και γάτες. Η εξέλιξη της κίνησης των ρομπότ, η βελτιστοποίηση των εργασιών σε βιομηχανικά περιβάλλοντα, αλλά και η μάθηση παιχνιδιών όπως το σκάκι από υπολογιστικά συστήματα, αποτελούν σαφή παραδείγματα της εφαρμογής και της χρησιμοποίησης της Ενισχυτικής Μάθησης. Σε αυτό το πλαίσιο, το σύστημα δεν καθοδηγείται από εξωτερικό επιτηρητή για το ποια ενέργεια πρέπει να πραγματοποιήσει. Αντίθετα,

έχει την ελευθερία να εξερευνήσει από μόνο του τις ενέργειες που θα του επιφέρουν το μέγιστο κέρδος καθώς δεν υπάρχει μέτρο για την αξιολόγηση των αποτελεσμάτων.

Με την χρήση των παραπάνω μεθόδων έχουμε φτάσει ένα βήμα πιο κοντά στο ερώτημα του Alan Turing, εάν οι μηχανές θα μπορούν να κάνουν αυτό που οι άνθρωποι ως νοήμον όντα μπορούν. Υπολογιστές νικάνε μερικούς από τους καλύτερους παίκτες σκάκι στον κόσμο, αυτοκίνητα μαθαίνουν να "οδηγούν" με χαρακτηριστικό παράδειγμα το σύστημα ALVINN του Pomerleau (1989), αλλά και άλλα τεχνολογικά επιτεύγματα, τα οποία παρότι δεν έχουν φτάσει όλα στο τελικό στάδιο εφαρμογής τους, δείχνουν την τεράστια εξέλιξη, καθώς και τις δυνατότητες που η Μηχανική Μάθηση μπορεί να προσφέρει στην ανθρωπότητα. Συνοψίζοντας, λοιπόν, καταλήγουμε στο συμπέρασμα ότι η Μηχανική Μάθηση έχει αποκτήσει περίοπτη θέση στην επιστημονική κοινότητα, γεγονός που διαφαίνεται και από την αύξηση του αριθμού των επιστημόνων που ασχολούνται με τον τομέα αυτόν.

Κεφάλαιο 2. Η ιδέα των Τεχνητών Νευρωνικών Δικτύων με ένα παράδειγμα

2.1 Μηχανική Μάθηση και Νευρωνικά Δίκτυα

Η Μηχανική Μάθηση αναπτύσσει τεχνικές αντιμετώπισης προβλημάτων που βασίζονται στην ανάλυση των δεδομένων (*data-driven*), επικεντρώνοντας κυρίως στην ικανότητα των αναπτυσσόμενων συστημάτων να προσαρμόζονται σύμφωνα με τα παρεχόμενα δεδομένα, αντί να εξαρτώνται αποκλειστικά από την εμπειρία των μηχανικών. Ο συνδυασμός των ευρύτατων ικανοτήτων που προσφέρει το πεδίο της Μηχανικής Μάθησης και η αδυναμία των αυστηρά αλγοριθμικών μεθόδων να αντιμετωπίσουν προβλήματα υψηλότερης πολυπλοκότητας, όπως η ταξινόμηση, μαζί με την διαρκή προσπάθεια των επιστημόνων να δημιουργήσουν μηχανές που να μπορούν να εκπαιδεύονται, οδήγησαν στη διαμόρφωση των τεχνικών που εφαρμόζονται στο πλαίσιο της Μηχανικής Μάθησης.

Ένα από τα πιο σημαντικά εργαλεία της Μηχανικής Μάθησης είναι τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ), τα οποία προσπαθούν να αντιμετωπίσουν τα προβλήματα με τον τρόπο που λειτουργεί το ανθρώπινο μυαλό. Αυτά τα δίκτυα έχουν ευρεία εφαρμογή σε ποικίλους επιστημονικούς τομείς, όπως η Ιατρική, η Βιολογία, η Γεωλογία και η Φυσική, για την επίλυση προβλημάτων όπως η ταξινόμηση και η πρόβλεψη. Τα ΤΝΔ βασίζονται στη δομή των φυσικών νευρωνικών δικτύων, επιτρέποντάς τους να εκτελούν πολύπλοκους υπολογισμούς. Στόχος των ΤΝΔ είναι να αναπαράγουν τους υπολογισμούς που διεξάγει το ανθρώπινο μυαλό, με τη δυνατότητα να μεταδίδουν πληροφορίες για τα ερεθίσματα που λαμβάνουν. Αυτά τα δίκτυα εκπαιδεύονται για να αντιμετωπίζουν εκχωρούμενα προβλήματα ή να εκτελούν αυτόνομα διεργασίες, όπως την αναγνώριση εικόνων. Προηγουμένως, ωστόσο, απαιτείται σωστή εκπαίδευσή τους. Όπως και η Μηχανική Μάθηση, που χωρίζεται σε Επιβλεπόμενη Μάθηση και Μη-Επιβλεπόμενη Μάθηση, τα ΤΝΔ εκπαιδεύονται είτε με ή χωρίς επίβλεψη. Παρόμοια με τη Μηχανική Μάθηση, τα Νευρωνικά Δίκτυα δεν είναι αποτελεσματικά σε προβλήματα που απαιτούν υψηλή αριθμητική ακρίβεια. Υπάρχουν ποικίλα μοντέλα Νευρωνικών Δικτύων με διαφορετικές αρχιτεκτονικές και τρόπους λειτουργίας και εκπαίδευσης. Η εξέλιξη του πεδίου των ΤΝΔ έχει περάσει μέσα από διάφορες φάσεις, με επιτυχημένα επιτεύγματα αλλά και αποτυχίες. Κατά την διάρκεια των τελευταίων 30 ετών, αυτός ο επιστημονικός τομέας έχει ενσωματωθεί όλο και περισσότερο σε διάφορους τομείς της καθημερινής ζωής, όπως η οικονομία, η ασφάλεια και η εκπαίδευση.

2.2 Εισαγωγή στην έννοια του τεχνητού νευρωνικού δικτύου

Νευρωνικό δίκτυο ονομάζεται ένα κύκλωμα διασυνδέσεων μονάδων επεξεργασίας που ονομάζουμε νευρώνες**.** Στους υπολογιστές είναι ένα υπολογιστικό μοντέλο που χρησιμοποιείται για την επίλυση κάποιων υπολογιστικών προβλημάτων.

Η δομή του τεχνητού νευρωνικού δικτύου μιμείται κατά το δυνατό εκείνη του βιολογικού νευρωνικού δικτύου, ώστε να εμφανίζει παρόμοιες ιδιότητες. Κατ' αναλογία επομένως με ένα δίκτυο νευρώνων του εγκεφάλου, ένα τεχνητό δίκτυο από ένα σύνολο τεχνητών νευρώνων που αλληλεπιδρούν, συνδεόμενοι μεταξύ τους με τις λεγόμενες συνάψεις (synapses). Ο βαθμός αλληλεπίδρασης είναι διαφορετικός για κάθε ζεύγος νευρώνων και καθορίζεται από τα λεγόμενα βάρη (weights).

Πιο αναλυτικά, το νευρωνικό δίκτυο αλληλεπιδρά με το περιβάλλον και εκπαιδεύεται από αυτό, έτσι τα βάρη μεταβάλλονται συνεχώς με αποτέλεσμα να ενδυναμώνει ή να αποδυναμώνει την ισχύ κάθε δεσμού. Η εμπειρική γνώση που αποκτά το νευρωνικό δίκτυο από το περιβάλλον

κωδικοποιείται στα βάρη, τα οποία αποτελούν το χαρακτηριστικό που δίνει στο δίκτυο την ικανότητα προσαρμογής και εξέλιξης στο περιβάλλον.

Υπάρχουν δύο τρόποι για να εκπαιδεύσουμε ένα δίκτυο. Στο πρώτο τρόπο η εκπαίδευση γίνεται με εποπτεία, δηλαδή το δίκτυο τροφοδοτείται με ένα σύνολο καταστάσεων στις οποίες μπορεί να περιέλθει το δίκτυο μας, μαζί με τα αποτελέσματα που θέλουμε να δίνει το δίκτυο. Για να μάθει το δίκτυο τις καταστάσεις αυτές, χρησιμοποιούμε έναν αλγόριθμο εκπαίδευσης. Κατά τον δεύτερο τρόπο, η εκπαίδευση γίνεται χωρίς εποπτεία, δηλαδή το δίκτυο καλείται να αναγνωρίσει ομοιότητες και μοτίβα σε δεδομένα που έχουμε τροφοδοτήσει.

Το κύριο πλεονέκτημα των νευρωνικών δικτύων είναι ότι μπορούν να αποθηκεύσουν γνώση και εμπειρία από το περιβάλλον, την οποία μπορούν στη συνέχεια να ανακαλέσουν.

2.3 Βιολογικά Νευρωνικά Δίκτυα

Ένα κεντρικό στοιχείο στην πρόοδο των Τεχνητών Νευρωνικών Δικτύων αποτέλεσε η ενσωμάτωση της γνώσης που αντλήθηκε από την δομή και τη λειτουργία των Βιολογικών Νευρωνικών Δικτύων, και ιδίως του ανθρώπινου εγκεφάλου. Ακόμη και μικρά παιδιά καταφέρνουν να εκτελούν καθημερινές εργασίες που αποτελούν πρόκληση για τους υπολογιστές, παρά την τεράστια υπεροχή τους σε υπολογιστική ισχύ. Η γνώση της δομής και της λειτουργίας των Βιολογικών Νευρωνικών Δικτύων υπήρξε πηγή έμπνευσης και βασικό στοιχείο στην εξέλιξη των Τεχνητών Νευρωνικών Δικτύων. Είναι ζωτικής σημασίας να αντιληφθούμε ορισμένες κύριες πτυχές της δομής και της λειτουργίας των Βιολογικών Νευρωνικών Δικτύων πριν αναφερθούμε στο αντίστοιχο πεδίο των Τεχνητών Νευρωνικών Δικτύων.

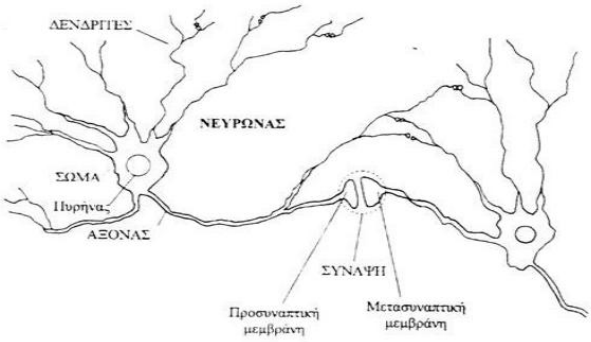
Το ανθρώπινο νευρικό σύστημα βασίζεται στον νευρώνα ως βασική δομική μονάδα. Ο νευρώνας, ένας εξειδικευμένος τύπος κυττάρου, αποτελεί τον πυρήνα των συστημάτων επεξεργασίας πληροφοριών του ανθρώπινου κεντρικού νευρικού συστήματος. Στην ουσία, αποτελεί το κλειδί για τη λειτουργία του εγκεφάλου, τόσο στον άνθρωπο όσο και στα ζώα. Υπολογίζεται ότι ένας ενήλικος εγκέφαλος περιέχει περίπου 100 δισεκατομμύρια νευρώνες, με κάθε νευρώνα να συνδέεται με περίπου 10.000 άλλους μέσω συνάψεων. Οι συνάψεις, που λειτουργούν ως σύνδεσμοι μεταξύ των νευρώνων, δεν έχουν σταθερό αριθμό αλλά διαφέρουν από νευρώνα σε νευρώνα.

Σε σύνολο, οι νευρώνες μαζί με τις συνάψεις τους δημιουργούν τα νευρωνικά δίκτυα και όλα αυτά συνθέτουν το Κεντρικό Νευρικό Σύστημα. Εντυπωσιακό είναι ότι, ενώ τα κύτταρα των νευρώνων δεν αναπαράγονται, οι συνάψεις συνεχώς δημιουργούνται και καταστρέφονται. Ο εγκέφαλος, με τους νευρώνες ως τα βασικά του στοιχεία, μεταφέρει πληροφορίες στα διάφορα συστήματα που ο ανθρώπινος οργανισμός διαθέτει, όπως για παράδειγμα το πεπτικό σύστημα.

Η δομή του νευρώνα παρουσιάζει ομοιότητες με τα υπόλοιπα κύτταρα του ανθρώπινου οργανισμού. Το σύνολο του νευρώνα αποτελείται από πολλά στοιχεία: το σώμα του, τον πυρήνα του, τους δενδρίτες, τον άξονα και τις συνάψεις. Για να γίνω πιο συγκεκριμένος, το σώμα αποτελεί το κυρίως μέρος του νευρώνα, μέσα στο οποίο βρίσκεται ο πυρήνας. Ο πυρήνας περιλαμβάνει όλες τις γενετικές πληροφορίες του οργανισμού.

Οι δενδρίτες λειτουργούν ως το σημείο εισόδου για τις πληροφορίες, δεχόμενοι εισερχόμενα σήματα από τους γειτονικούς νευρώνες. Τα δεδομένα ρέουν στον κάθε νευρώνα μέσω των δενδριτών, παίρνοντας τη μορφή ηλεκτρικών παλμών. Αφού εισέλθει σε έναν νευρώνα, η πληροφορία απομακρύνεται από αυτόν μέσω ηλεκτρικών σημάτων μέσω του αξονικού του τμήματος.

Για να ολοκληρωθεί η βασική δομή του νευρώνα, οι συνάψεις αναφέρονται στα λεπτά κενά όπου οι διακλαδώσεις του αξονικού τμήματος συναντούν τους δενδρίτες των γειτονικών νευρώνων. Η ταχύτητα με την οποία προωθείται η ηλεκτρική δραστηριότητα από τον αξονικό προς τους δενδρίτες εξαρτάται από παράγοντες όπως το πλάτος της σύναψης και η πυκνότητα των ηλεκτροχημικών υλικών. Το ποσό του ηλεκτρικού παλμού που τελικά μεταφέρεται στους δενδρίτες του νευρώνα, γνωστό ως συναπτικό βάρος, αναφέρεται στην ένταση του σήματος.



Σχήμα 2.1: Βιολογικός Νευρώνας [9]

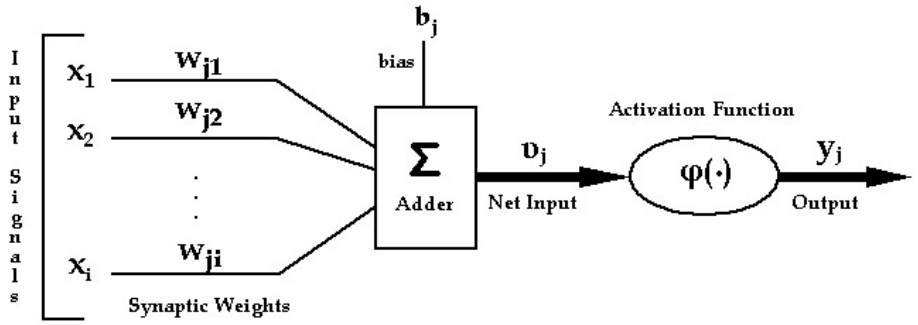
Οι νευρώνες εμφανίζουν δύο δυνατές καταστάσεις. Κατά τη λειτουργία, ένας νευρώνας μπορεί να βρίσκεται σε κατάσταση ενεργού ή μη-ενεργού. Όταν λέμε ότι ένας νευρώνας είναι ενεργός, εννοούμε ότι δημιουργεί μια ηλεκτρική αγωγή με την οποία μεταφέρονται οι πληροφορίες του νευρώνα σε άλλους κοντινούς νευρώνες. Αντίστοιχα, όταν ένας νευρώνας παραμένει ανενεργός, δεν παράγει καμία ηλεκτρική διακύμανση. Αυτή η ευαισθησία του νευρώνα να ανταποκρίνεται σε διάφορα εξωτερικά ερεθίσματα, όπως ηλεκτρομαγνητικά πεδία, χημικά σήματα, θερμότητα κ.λπ., είναι αυτή που δημιουργεί τους ηλεκτρικούς παλμούς που μεταφέρουν τις πληροφορίες. Κάθε νευρώνας συγκεντρώνει το συνολικό ηλεκτρικό φορτίο που λαμβάνει από όλες τις συνδέσεις του, μέσω των δενδριτών του. Όσο πιο ισχυρή είναι η σύνδεση σε μια σύναψη μεταξύ δύο νευρώνων, τόσο περισσότερο επηρεάζει το συνολικό φορτίο που μεταφέρεται στο σύνολο του φορτίου εισόδου. Ένας νευρώνας θεωρείται ενεργός, δηλαδή δημιουργεί ηλεκτρικούς παλμούς, όταν το συνολικό φορτίο εισόδου υπερβαίνει ένα καθορισμένο κατώφλι. Σε αντίθετη περίπτωση, ο νευρώνας παραμένει αδρανής. Συνεπώς, μπορούμε να αναφέρουμε ότι ο νευρώνας λειτουργεί ως δυαδικό στοιχείο.

2.4 Η δομή και η λειτουργία του Νευρώνα

Το Τεχνητό Νευρωνικό Δίκτυο – ΤΝΔ (*Artificial Neural Network-ANN*) ονομάζεται δίκτυο καθώς αποτελείται από τεχνητούς νευρώνες που συνδέονται μεταξύ τους και αποτελούν τη δομική μονάδα του. Κάθε τεχνητός νευρώνας δέχεται ένα σύνολο αριθμητικών εισόδων (από άλλους νευρώνες είτε από κάποια άλλη είσοδος), στη συνέχεια εκτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μια έξοδο. Η έξοδος από αυτόν τον κόμβο μπορεί είτε να αποτελέσει μέρος της συνολικής εξόδου του ΤΝΔ είτε να διοχετευτεί σε άλλους κόμβους. Έτσι θεωρούμε πως υπάρχουν τριών ειδών νευρώνων:

1. Οι *νευρώνες εισόδου*, των οποίων η εργασία είναι να διοχετεύσουν στους υπολογιστικούς νευρώνες την είσοδο του προβλήματος (πχ. αρχικά δεδομένα).
2. Οι *νευρώνες εξόδου*, χρησιμοποιούνται για να παρουσιάσουν στο περιβάλλον την απάντηση του ΤΝΔ σε κάποιο πρόβλημα, όπως για παράδειγμα τη ν εκτίμηση της κατηγορίας ενός προβλήματος κατηγοριοποίησης.
3. Οι *υπολογιστικοί νευρώνες* ή *κρυμμένοι νευρώνες*, οι οποίοι πολλαπλασιάζουν κάθε είσοδο που δέχονται από νευρώνες εισόδου ή από άλλους νευρώνες επεξεργασίας με μια τιμή συσχετισμένη με αυτούς που ονομάζεται βάρος. Το συνολικό αποτέλεσμα εισάγεται σε μια συνάρτηση που θα ονομάζουμε συνάρτηση ενεργοποίησης.

Παρακάτω βλέπουμε το βασικό μοντέλο του νευρώνα που χρησιμοποιείται για τις υλοποιήσεις τεχνητών νευρωνικών δικτύων.



Σχήμα 2.2: Απεικόνιση του βασικού μοντέλου του Νευρώνα. Δηλαδή τα δεδομένα εισόδου, τα βάρη, η συνάρτηση ενεργοποίησης και τα δεδομένα εξόδου.

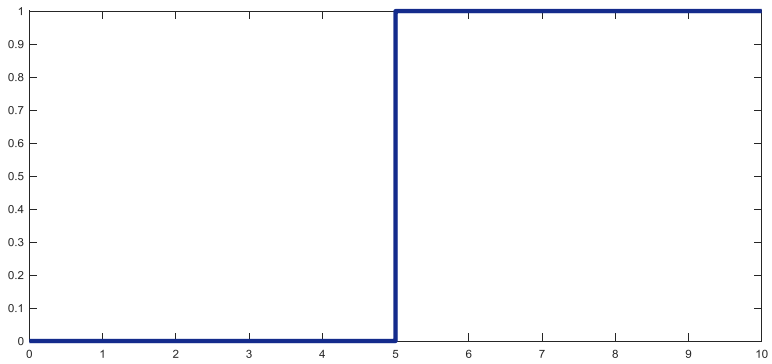
Στο νευρώνα, η πληροφορία ρέει πάντα προς μια κατεύθυνση, από αριστερά προς τα δεξιά και διακρίνουμε τις εξής φάσεις της λειτουργία του:

Στη πρώτη φάση, κάθε είσοδος πολλαπλασιάζεται με το βάρος που της αντιστοιχεί. Στη δεύτερη φάση, οι σταθμισμένες πλέον είσοδοι και ένας εξωτερικά εφαρμοσμένος παράγοντας, η *μεροληψία* ή *πόλωση* ή *κατώφλι* (*bias, threshold*), αθροίζονται και δίνουν το *τοπικό πεδίο* (*net input, activation potential*). Στην Τρίτη φάση, εφαρμόζεται η *συνάρτηση ενεργοποίησης* ή *συνάρτηση μεταφοράς* (*activation function or squashing function*) στο τοπικό πεδίο και το αποτέλεσμα δίνει την έξοδο του νευρώνα.

Στα πρώτα μοντέλα νευρώνα , η *συνάρτηση ενεργοποίησης* ήταν μια βηματική συνάρτηση (*step function*).

$$\varphi(x) = \begin{cases} 0, & x \leq \theta \\ 1, & \chi > \theta \end{cases} . \quad (2.1)$$

Παρατηρούμε πως αν το ενδιάμεσο αποτέλεσμα ήταν μικρότερο μιας τιμής μεροληψίας, η έξοδος του νευρώνα θα έτεινε στο 0 (αδρανής νευρώνας), αλλιώς θα έτεινε στο 1 (ενεργοποιημένος νευρώνας). Το παραπάνω μοντέλο αναφέρεται ως μοντέλο McCulloch-Pitts.



Σχήμα 2.3: Γραφική παράσταση της βηματικής συνάρτησης (2.1) για θ=5.

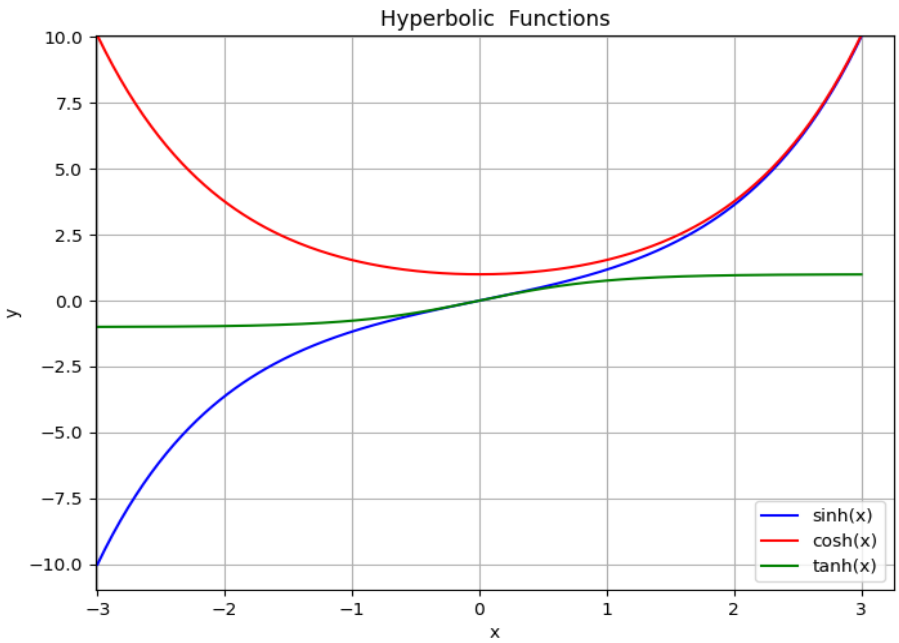
Στα περισσότερα μοντέλα πλέον, η συνάρτηση ενεργοποίησης είναι μια σιγμοειδής συνάρτηση που είναι μια πραγματική συνάρτηση, συνεχής και φραγμένη, της οποίας η παράγωγος είναι θετική.

Το πεδίο ορισμού της θεωρητικά είναι όλο το σύνολο των πραγματικών αριθμών, αλλά μπορεί να περιοριστεί θέτοντας όρια στις τιμές των βαρών. Το σύνολο τιμών είναι συνήθως το διάστημα [0,1] ή [-1,1]. Ένα γνωστό παράδειγμα σιγμοειδούς συνάρτησης που χρησιμοποιείται ως συνάρτηση ενεργοποίησης είναι η λογιστική συνάρτηση (*logistic function*), που δίνεται από τον τύπο

$$\varphi(\chi) = \frac{1}{1+e^{-ax}} \quad (2.2)$$

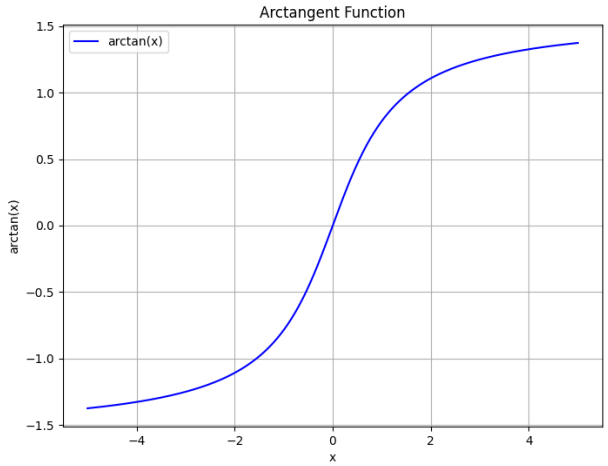
όπου α η παράμετρος κλίσης. Παρατηρούμε ότι, όσο το α τείνει στο άπειρο, η λογιστική συνάρτηση τείνει προς τη βηματική εξίσωση και έχουμε το μοντέλο McCulloch-Pitts.

Κάποια ακόμη παραδείγματα σιγμοειδών συναρτήσεων που χρησιμοποιούνται ως συναρτήσεις ενεργοποίησης είναι η υπερβολική συνάρτηση (hyperbolic function)



Σχήμα 2.4: Γραφικές παραστάσεις των συναρτήσεων υπερβολικού ημιτόνου sinh, υπερβολικού συνημίτονου cosh και υπερβολικής εφαπτόμενης tanh.

και η συνάρτηση τόξου εφαπτομένης (arctangent function).



Σχήμα 2.5: : Γραφική παράσταση των συνάρτησης του τόξου εφαπτομένης.

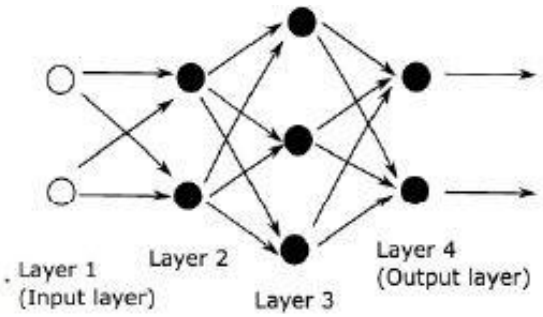
Με την εισαγωγή της συνάρτησης ενεργοποίησης, ο νευρώνας γίνεται μη γραμμικός και κατ επέκταση και το αντίστοιχο τεχνητό νευρωνικό δίκτυο θα είναι μη γραμμικό. Πολλές φορές χρειάζεται κάποιοι από τους νευρώνες του μη-γραμμικού δικτύου να είναι γραμμικοί, μια επιπλέον συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η γραμμική, όπως η $\varphi(x) = x$ ή κάποια άλλη συνάρτηση.

Σύμφωνα με τα παραπάνω, η λειτουργία του νευρώνα περιγράφεται ως εξής,

$$\varphi\left(\sum_j w_{ij} a_j + b_i\right) \quad (2.3)$$

όπου ο πίνακας W περιέχει τα βάρη, το διάνυσμα b περιέχει τις μεροληψίες και το διάνυσμα α συλλέγει τις τιμές που παράχθηκαν στη προηγούμενη φάση. Το πλήθος των στηλών του πίνακα W ταιριάζει με τον αριθμό των νευρώνων που παρήγαγαν το διάνυσμα α στη προηγούμενη φάση. Ο αριθμός των γραμμών του πίνακα W ταιριάζει με το πλήθος των νευρώνων στη τρέχουσα φάση. Ο αριθμός των συνιστωσών του διανύσματος b ταιριάζει με τον αριθμό των νευρώνων της τρέχουσας φάσης.

Σύμφωνα με τα παραπάνω θα δώσουμε ένα απλό παράδειγμα της λειτουργίας ενός τεχνητού νευρωνικού δικτύου που αποτελείται από τέσσερεις φάσεις.



Σχήμα 2.6: Νευρωνικό δίκτυο τεσσάρων φάσεων.

Για το νευρωνικό δίκτυο του σχήματος 3, αρχική φάση (*input layer*) αναπαριστάται από δύο κύκλους, διότι τα δεδομένα εισόδου έχουν δύο παραμέτρους. Τα βέλη από τη πρώτη στη δεύτερη φάση υποδηλώνουν ότι οι δύο παράμετροι των δεδομένων εισόδου είναι διαθέσιμες για δύο νευρώνες της δεύτερης φάσης. Τα δεδομένα εισόδου είναι της μορφής διανύσματος στον δισδιάστατο χώρο, δηλαδή $x \epsilon \mathbb{R}^2$. Επιπλέον, τα βάρη και οι μεροληψίες της δεύτερης φάσης μπορούν να συγκεντρωθούν και να αναπαρασταθούν από έναν πίνακα $W^{[2]} \in \mathbb{R}^{2\times 2}$ και ένα διάνυσμα $b^{[2]} \in \mathbb{R}^2$. Το αποτέλεσμα της δεύτερης φάσης θα έχει τη μορφή

$$\varphi\left(W^{[2]}x + b^{[2]}\right) \epsilon \mathbb{R}^2. \quad (2.4)$$

Στη συνέχεια παρατηρούμε ότι η τρίτη φάση θα έχει τρείς νευρώνες, ο καθένας θα λαμβάνει διανύσματα που ανήκουν στον $\mathbb{R}^2$, σαν αποτέλεσμα τα βάρη και οι μεροληψίες της τρέχουσας φάσης να μπορούν να αποθηκευτούν σε πίνακα $W^{[3]} \in \mathbb{R}^{3\times 2}$ και σε ένα διάνυσμα $b^{[3]} \in \mathbb{R}^3$αντίστοιχα. Τελικά το αποτέλεσμα της τρίτης φάσης θα έχει τη μορφή

$$\varphi\left(W^{[3]}\varphi\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) \epsilon \mathbb{R}^3. \quad (2.5)$$

Η τέταρτη φάση θα έχει δύο νευρώνες, ο καθένας θα λαμβάνει διανύσματα που ανήκουν στον $\mathbb{R}^3$, σαν αποτέλεσμα τα βάρη και οι μεροληψίες της τρέχουσας φάσης να μπορούν να αποθηκευτούν

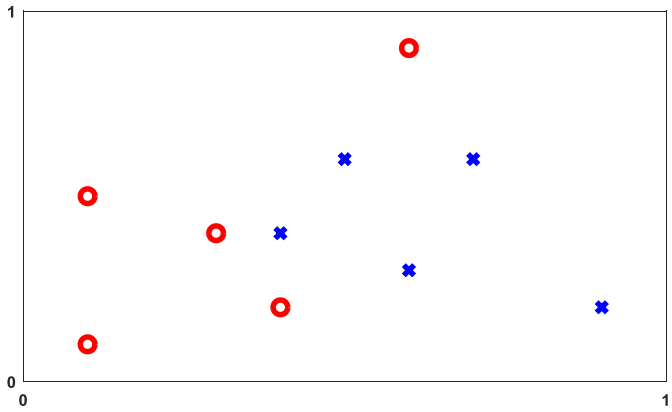
σε πίνακα $W^{[4]} \in \mathbb{R}^{2\times 3}$ και σε ένα διάνυσμα $b^{[4]} \in \mathbb{R}^2$ αντίστοιχα. Τελικά το αποτέλεσμα της τέταρτης φάσης θα έχει τη μορφή

$$F(x) = \varphi\left(W^{[4]}\varphi\left(W^{[3]}\varphi\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) + b^{[4]}\right)\epsilon\mathbb{R}^2. \quad (2.6)$$

2.5 Τεχνητό Νευρωνικό Δίκτυο- Άντληση πετρελαίου

Έχοντας δει τη δομή και τη λειτουργία ενός ΤΝΔ θα εξετάσουμε την εφαρμογή του σε ένα πρόβλημα επιλογής σημείου γεώτρησης για την άντληση πετρελαίου.

Θεωρούμε ότι έχουμε ένα σύνολο σημείων να οποία αναπαριστούν κάποια δεδομένα. Πιο συγκεκριμένα, τα σημεία που τα συμβολίζουμε με κύκλο τα κατατάσσουμε στη κατηγορία Α και μας δείχνουν ότι έχουμε επιτυχημένη άντλησης πετρελαίου στη περιοχή που έχουμε σημαδέψει. Τα υπόλοιπα σημεία τα συμβολίζουμε με σταυρό, ανήκουν στη κατηγορία Β, και μας δηλώνουν ότι η άντληση πετρελαίου στη περιοχή που σημαδέψαμε ήταν ανεπιτυχής. Στόχος μας είναι να ιχνηλατήσουμε μια περιοχή, δηλαδή να παίρνουμε σημεία που λαμβάνουν συντεταγμένες από τον $\mathbb{R}^2$ χώρο και να κατατάσσει τα σημεία είτε στη κατηγορία Α είτε στη Β. Στο σχήμα που ακολουθεί φαίνονται πως έχουν κατηγοριοποιηθεί τα σημεία ανάλογα με την επιτυχή ή μη επιτυχή άντληση πετρελαίου.

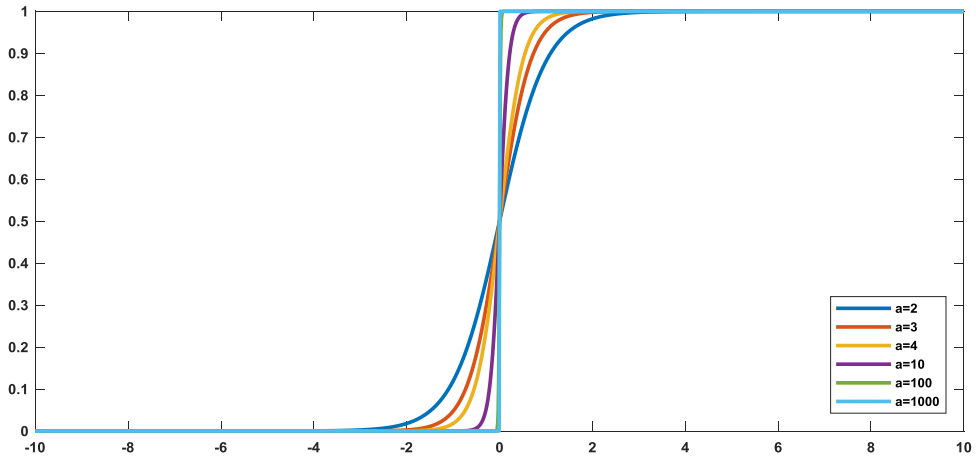


Σχήμα 2.7: Σημεία στο $\mathbb{R}^2$. Οι κύκλοι ανήκουν στην κατηγορία Α. Οι σταυροί ανήκουν στη κατηγορία Β.

Το ερώτημα που θα κληθούμε να απαντήσουμε είναι, αν χρησιμοποιήσουμε τα δεδομένα που έχουμε λάβει, μπορούμε να κατηγοριοποιήσουμε με επιτυχία νέα σημεία άντλησης στη περιοχή μας. Η προσέγγιση του τεχνητού νευρωνικού δικτύου χρησιμοποιεί την επαναλαμβανόμενη χρήση μιας απλής και μη γραμμικής συνάρτησης.

Στη περίπτωση μας, θα βασίσουμε το δίκτυο στη σιγμοειδή συνάρτηση

$$\varphi(x) = \frac{1}{1+e^{-ax}}\ , \mu\varepsilon -10 \leq x \leq 10. \quad (2.7)$$



Σχήμα 2.8: Γραφική αναπαράσταση της σιγμοειδής συνάρτησης για τις διάφορες τιμές του α.

Μπορούμε να θεωρήσουμε την φ(χ) σαν την πιο ομαλή μορφή της βηματικής συνάρτησης, η οποία μιμείται την συμπεριφορά του εγκεφαλικού νευρώνα. Συγκεκριμένα, κατά την ενεργοποίηση δίνει αποτέλεσμα ίσο με τη μονάδα εάν τα δεδομένα εισόδου είναι αρκετά με μεγάλα, και παραμένει ανενεργή, δηλαδή δίνει αποτέλεσμα ίσο με μηδέν, σε κάθε άλλη περίπτωση. Αν εφαρμόσουμε την πρώτη παράγωγο στη σιγμοειδή συνάρτηση, η μορφή που παίρνει είναι

$$\varphi'(\chi) = \varphi(x)\big(1 - \varphi(x)\big). \quad (2.8)$$

Αν θέλουμε να γράψουμε τη σιγμοειδή συνάρτηση σε διανυσματική μορφή τότε, για $z \in \mathbb{R}^m, \varphi: \mathbb{R}^m \to \mathbb{R}^m$ ορίζεται ως $(\varphi(z))_i = \varphi(z_i)$.

Σύμφωνα με τα παραπάνω μπορούμε να ορίσουμε τις φάσεις των νευρώνων. Σε κάθε φάση, κάθε νευρώνας παράγει σαν αποτέλεσμα έναν πραγματικό αριθμό, ο οποίος περνάει σε κάθε νευρώνα της επόμενης φάσης. Στην επόμενη φάση, κάθε νευρώνας παίρνει τη μορφή του από τον συνδυασμό των βαρών, προσθέτοντας τις μεροληψίες και εφαρμόζοντας τη σιγμοειδή συνάρτηση. Εάν μαθηματικοποιήσουμε τα παραπάνω, οι πραγματικοί αριθμοί που παράγονται από τους νευρώνες σε κάθε φάση, απεικονίζονται μέσω ενός διανύσματος α, έπειτα το διάνυσμα των αποτελεσμάτων της επόμενης φάσης έχει τη μορφή

$$\varphi(Wa + b) \quad (2.9)$$

όπου ο πίνακας W περιέχει τα βάρη, το διάνυσμα b περιέχει τις μεροληψίες και το διάνυσμα α συλλέγει τις τιμές που παράχθηκαν στη προηγούμενη φάση. Για να δώσουμε έμφαση στη λειτουργία του ι-οστού νευρώνα της σχέσης (2.9), θα διαλέξουμε την ι-συνιστώσα ως εξής:

$$\varphi\big(\textstyle\sum_j w_{ij} a_j + b_i\big) \quad (2.10)$$

όπου το άθροισμα υπολογίζεται για όλες τις τιμές του διανύσματος α.

Εάν τώρα για το πρόβλημα άντλησης πετρελαίου εφαρμόσουμε το δίκτυο τεσσάρων φάσεων που αναλύσαμε στη προηγούμενη παράγραφο και φτάσουμε στη σχέση (2.6), δηλαδή την

$$F(x) = \varphi\big(W^{[4]}\varphi\big(W^{[3]}\varphi\big(W^{[2]}x + b^{[2]}\big) + b^{[3]}\big) + b^{[4]}\big)\epsilon\mathbb{R}^2.$$

Παρατηρούμε ότι η σχέση (2.6) ορίζει μια συνάρτηση $F: \mathbb{R}^2 \to \mathbb{R}^2$ ως προς τις 23 παραμέτρους των δεδομένων εισόδου στους πίνακες των βαρών και στα διανύσματα των μεροληψιών. Θέλουμε να βελτιστοποιήσουμε τις παραμέτρους έτσι ώστε να δημιουργήσουμε μια ροή βάσει των δεδομένων που παίρνουμε από το σχήμα (2.5). Απαιτούμε η $F(x)$ να πλησιάζει το $[1,0]^T$ για τα δεδομένα που ανήκουν στη κατηγορία Α και να πλησιάζει $[0,1]^T$ για τα δεδομένα που ανήκουν στη κατηγορία Β. Στη συνέχεια αν πάρουμε ένα σημείο $x \epsilon \mathbb{R}^2$, θα ήταν σκόπιμο να ταξινομηθεί ανάλογα με τη μεγαλύτερη συνιστώσα της $F(x)$. Πιο συγκεκριμένα, στη κατηγορία Α θα ανήκουν εάν $F_1(x) > F_2(x)$ και στη κατηγορία Β αν $F_1(x) < F_2(x)$. Η απαίτηση για την F μπορεί να προσδιοριστεί μέσω μιας συνάρτησης κόστους (cost function). Αν δηλώσουμε τα δέκα σημεία του σχήματος (2.5), γράφοντας $\{x^{\{i\}}\}_{i=1}^{10}$ και $y(x^{\{i\}})$ για το αποτέλεσμα ως εξής,

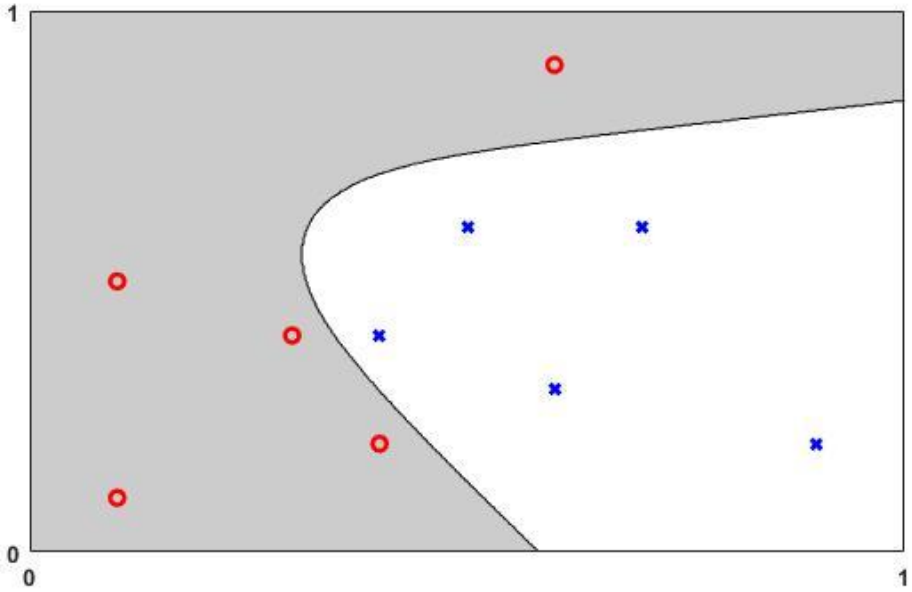
$$y\left(x^{\{i\}}\right) = \begin{cases} \binom{1}{0} \text{ εάν } x^{\{i\}} \text{ανήκει στη κατηγορία Α} \\ \binom{0}{1} \text{ εάν } x^{\{i\}} \text{ανήκει στη κατηγορία Β.} \end{cases} \quad (2.11)$$

Τελικά, η συνάρτηση κόστους παίρνει τη μορφή

$$Cost\left(W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]}\right) = \frac{1}{10}\sum_{i=1}^{10}\frac{1}{2}\left\|y\left(x^{\{i\}}\right) - F\left(x^{\{i\}}\right)\right\|_2^2. \quad (2.12)$$

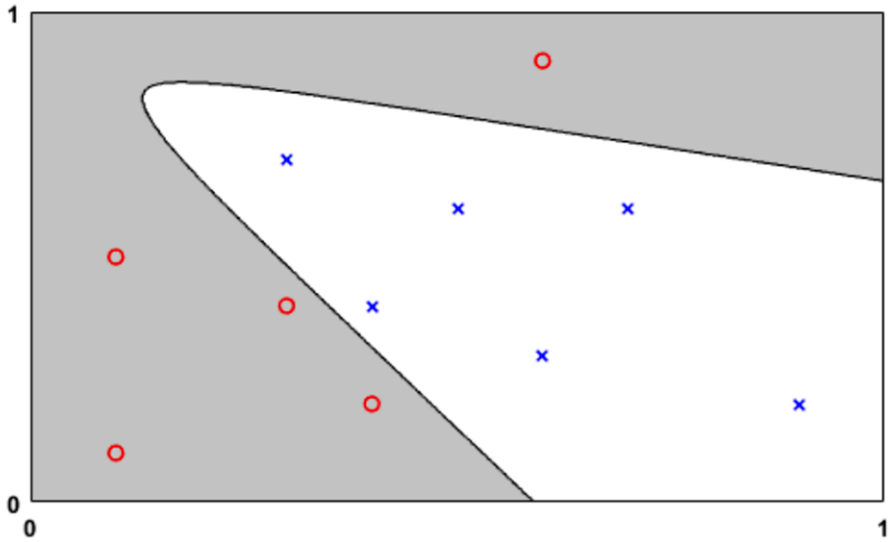
όπου ο όρος $\frac{1}{2}$ προστίθεται για την απλοποίηση των παραγώγων της συνάρτησης κόστους. Η συνάρτηση κόστους είναι μια συνάρτηση που αποτελείται από βάρη και μεροληψίες και τα σημεία δεδομένων είναι σταθερά. Ο τύπος της συνάρτησης (2.6), όπου η απόκλιση δίνεται από το μέσο όρο της Ευκλείδειας τετραγωνικής νόρμας στα σημεία των δεδομένων, αναφέρεται συχνά ως συνάρτηση τετραγωνικού κόστους (quadratic cost function).

Αν διαλέξουμε τα βάρη και τις μεροληψίες έτσι ώστε να ελαχιστοποιείται η συνάρτηση κόστους τότε αναφερόμαστε στην εκπαίδευση δικτύου. Στο σχήμα (2.7) που ακολουθεί θα υλοποιήσουμε την περίπτωση όπου το $F_1(x) > F_2(x)$.



Σχήμα 2.9: Σχηματική αναπαράσταση των αποτελεσμάτων ενός τεχνητού νευρωνικού δικτύου των δεδομένων του σχήματος 2.7.

Παρατηρούμε ότι κάθε σημείο που βρίσκεται στη γκρι χρώματος περιοχή υπάγεται στη κατηγορία Α και οποιοδήποτε άλλο σημείο ανήκει στη περιοχή Β. Θα επαναλάβουμε τη παραπάνω διαδικασία με ένα πρόσθετο δεδομένο εισόδου-εκμάθησης, ώστε να δούμε την ανταπόκριση του δικτύου. Επιλέγουμε να προσθέσουμε το σημείο με συντεταγμένες (0.3, 0.7) και να ανήκει στην κατηγορία Β, προκύπτει το σχήμα που ακολουθεί.



Σχήμα 2.10: Επανάληψη της διαδικασίας του σχήματος 2.9 με ένα πρόσθετο σημείο.

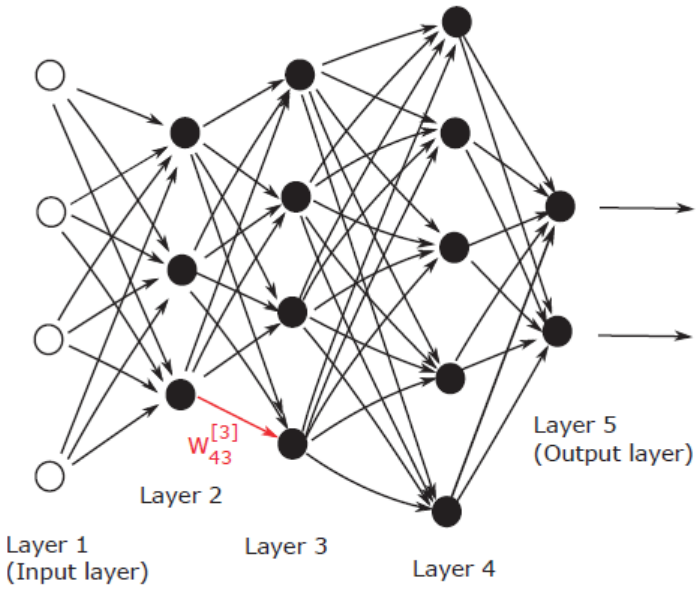
2.6 Τεχνητό Νευρωνικό Δίκτυο- Γενικός κανόνας

Σε προηγούμενη παράγραφο είδαμε το παράδειγμα ενός τεχνητού νευρωνικού δικτύου τεσσάρων φάσεων όπου και γίνεται χρήση της σιγμοειδής συνάρτησης σε κάθε φάση. Γενικά, κάθε νευρώνας λαμβάνει την ίδια τιμή εισόδου, δηλαδή έναν πραγματικό αριθμό για κάθε νευρώνα της προηγούμενης φάσης και παράγει μια νέα πραγματική τιμή, η οποία περνάει σε κάθε νευρώνα της επόμενης φάσης. Από την διαδικασία που αναφέραμε θα εξαιρέσουμε δύο φάσεις. Κατά τη διάρκεια της αρχικής φάσης, δεν υπάρχει η έννοια της προηγούμενης φάσης και κάθε νευρώνας λαμβάνει ένα διάνυσμα εισόδου. Κατά την τελική φάση, δεν υπάρχει η έννοια της επόμενης φάσης και οι νευρώνες παράγουν το συνολικό αποτέλεσμα. Οι φάσεις μεταξύ των δύο που αναφέραμε ονομάζονται κρυμμένες φάσεις (hidden layers).

Υποθέτουμε ότι το δίκτυο μας έχει L φάσεις, με τις φάσεις με αριθμό 1 και L να αποτελούν την αρχική και τελική φάση αντίστοιχα. Επίσης, υποθέτουμε ότι η φάση l, με $l = 1, 2, 3, \ldots\ldots, L$, να περιέχει n_l νευρώνες. Έτσι ο νευρώνας n_1 έχει τη διάσταση των δεδομένων εισόδου στο δίκτυο. Γενικά, το δίκτυο χαρτογραφείται από το $\mathbb{R}^{n_1}$ μέχρι $\mathbb{R}^{n_l}$. Χρησιμοποιούμε τον πίνακα $W^{[l]} \in \mathbb{R}^{n_l \times n_{l-1}}$ για να δημιουργήσουμε τον πίνακα με τα βάρη στη φάση l. Συγκεκριμένα, το $w_{jk}^{[l]}$ είναι το βάρος όπου ο νευρώνας j στη φάση l εφαρμόζεται στη τιμή εξόδου του νευρώνα k στη φάση $l-1$. Όμοια, $b^{[l]} \in \mathbb{R}^{n_1}$ είναι το διάνυσμα των μεροληψιών στη φάση l, έτσι ο νευρώνας j στη φάση l χρησιμοποιεί τη μεροληψία $b_j^{[l]}$.

2.7 Παράδειγμα Τεχνητού Νευρωνικού Δικτύου Πέντε Φάσεων

Σε συνέχεια των προηγούμενων, σε αυτή τη παράγραφο, θα δούμε ένα τεχνητό νευρωνικό δίκτυο πέντε φάσεων. Έχουμε, $L = 5$ φάσεις και $n_1 = 4,\ n_2 = 3,\ n_3 = 4,\ n_4 = 5,\ n_5 = 2$. Έτσι προκύπτουν τα εξής, $W^{[2]} \in \mathbb{R}^{3\times4}$, $W^{[3]} \in \mathbb{R}^{4\times3}$, $W^{[4]} \in \mathbb{R}^{5\times4}$, $W^{[5]} \in \mathbb{R}^{2\times5}$, $b^{[2]} \in \mathbb{R}^3$, $b^{[3]} \in \mathbb{R}^4$, $b^{[4]} \in \mathbb{R}^5$ και $b^{[5]} \in \mathbb{R}^2$. Στο σχήμα που ακολουθεί μπορούμε να δούμε γραφικά τα δεδομένα που περιγράψαμε.



Σχήμα 2.11: Δίκτυο πέντε φάσεων. Η ακμή που αντιστοιχεί στο βάρος $w_{43}^{[3]}$ είναι με κόκκινο χρώμα. Το αποτέλεσμα από το νευρώνα 3 στη φάση 2 έχει βάρος από τον συντελεστή $w_{43}^{[3]}$ όταν τροφοδοτείται στο νευρώνα 4 της φάσης 3.

Αν δώσουμε σαν είσοδο στο δίκτυο $x \in \mathbb{R}^{n_1}$, τότε μπορούμε να συνοψίσουμε τη δράση του νευρωνικού δικτύου, δηλώνοντας ως $a_j^{[l]}$ την έξοδο ή την δράση από έναν νευρώνα j στη φάση l. Έτσι προκύπτουν οι παρακάτω σχέσεις:

$$a^{[1]} = x \in \mathbb{R}^{n_1} \quad (2.13)$$

$$a^{[l]} = \varphi\left(W^{[l]}a^{[l-1]} + b^{[l]}\right)\mathbb{R}^{n_l} \ \ \gamma\iota\alpha\ l = 2,3,\ldots,L \quad (2.14)$$

Οι σχέσεις (2.13) και (2.14) αποτελούν μέρη ενός αλγορίθμου τροφοδοσίας δικτύου προκειμένου να παραχθεί μια έξοδος $a^{[L]} \in \mathbb{R}^{n_L}$.

Υποθέτουμε ότι έχουμε N μονάδες δεδομένων ή σημεία εκπαίδευσης στο $\mathbb{R}^{n_1}$, $\{x^{\{i\}}\}_{i=1}^{N}$ για τα οποία δίνονται τα αντίστοιχα δεδομένα εξόδου $\{y(x^{\{i\}})\}_{i=1}^{N}$ στον $\mathbb{R}^{n_L}$. Αν γενικεύσουμε την (2.12), η τετραγωνική συνάρτηση κόστους που θέλουμε να ελαχιστοποιήσουμε έχει τη μορφή

$$Cost = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|y\left(x^{\{i\}}\right) - a^{[L]}\left(x^{\{i\}}\right)\right\|_2^2, \quad (2.15)$$

όπου δεν έχουμε υποδείξει ρητά ότι η συνάρτηση κόστους είναι συνάρτηση όλων των βαρών και των μεροληψιών.

2.8 Μέθοδος Απότομης Καθόδου (Steepest Descent Method)

Θα εισάγουμε τη μέθοδο απότομης καθόδου ώστε να μας βοηθήσει στην βελτιστοποίηση του προβλήματος. Στη μέθοδο απότομης καθόδου χρησιμοποιούμε τον επαναληπτικό τύπο $x_{k+1} = x_k + a_k d_k$ λαμβάνοντας ως μήκος βήματος το βήμα που προκαλεί μέγιστη μείωση της συνάρτησης σε κάθε επανάληψη.

Η σύγκλιση της μεθόδου προϋποθέτει σε κάθε βήμα την κατάλληλη κατεύθυνση που θα μας μετατοπίσει στο επόμενο βήμα και η διαδικασία θα επαναλαμβάνεται μέχρι να εντοπιστεί το σημείο x^* .

Θα πρέπει να προσδιορίσουμε το κατάλληλο βήμα έτσι ώστε να μην αποκλίνει η μέθοδος μας. Θέτουμε μια συνάρτηση φ ίση με

$$\varphi(\alpha) = f(x_k + a d_k)$$

για πραγματικό αριθμό α, όπου λαμβάνουμε ως κατεύθυνση την gradient $-\nabla f$ για την τιμή του σημείου χ στη κ-οστή επανάληψη, δηλαδή για

$$d_k = -\nabla f(x_k).$$

Τότε για την μέθοδο απότομης καθόδου επιλέγουμε το βήμα α^* για το οποίο η συνάρτηση φ γίνεται ελάχιστη, δηλαδή

$$\alpha^*: f(\alpha^*) = \min_a f(x_k - a\nabla f(x_k)).$$

Για να προσδιορίσουμε το μήκος του βήματος, ορίζουμε μια συνάρτηση $g(a_i) = f(x_i + a_i S_i)$ και βρίσκουμε τα κρίσιμα σημεία της, δηλαδή

$$\nabla g(a_i) = 0$$

Και έχουμε

$$\nabla g(a_i) = 0 \Longrightarrow a_i = \alpha^*$$

Δηλαδή το κρίσιμο σημείο α^* της συνάρτησης g προσδιορίζει το μήκος βήματος που ελαχιστοποιεί τη συνάρτηση φ.

Στη συνέχεια μέσω του επαναληπτικού σχήματος

$$x_{i+1} = x_i + a_i \nabla f(x_i) \text{ ή } x_{i+1} = x_i + a_i S_i.$$

Βρίσκουμε την τιμή χ στην επόμενη φάση, δηλαδή την τιμή του x_i για $i = i + 1$. Αν η κλίση της συνάρτησης για τη νέα τιμή του σημείου προσέγγισης είναι ίση με το μηδέν, τότε το σημείο στην i-οστή επανάληψη είναι ίσο με το α^*. Άρα θα έχουμε τον τερματισμό του αλγορίθμου, διαφορετικά η διαδικασία θα επαναλαμβάνεται για το καινούριο $i = i + 1$.

Έτσι καταλήγουμε στο παρακάτω αλγόριθμο:

<u>Αλγόριθμος Απότομης Καθόδου</u>

1. Για δοσμένο αρχικό σημείο x_0 θέτουμε το επαναληπτικό βήμα i=1.
2. Βρίσκουμε την κατεύθυνση, έστω S_i, που θα ακολουθήσουμε ορίζοντας την ίση με το αντίθετο της κλίσης (gradient), δηλαδή $S_i = -\nabla f_i$.
3. Βρίσκουμε το μήκος του βήματος λ_i που ελαχιστοποιεί την συνάρτηση $f(x_i + \lambda_i S_i)$ προσδιορίζοντας τα κρίσιμα σημεία.
4. Θέτουμε $x_{i+1} = x_i + \lambda_i S_i$.
5. Εξετάζουμε αν η gradient της συνάρτησης f με μεταβλητή x_{i+1} είναι ίση με μηδέν. Αν όχι, τότε επιστρέφουμε στο Βήμα 2 και επαναλαμβάνουμε την ίδια διαδικασία μέχρι να επαληθευτεί η ισότητα της συνθήκης.
6. Αν ναι, τότε θέτουμε την τιμή αυτή ίση με το ζητούμενο x^* .
7. Τέλος αλγορίθμου.

2.9 Μέθοδος Στοχαστικής Κλίσης (Stochastic Gradient)

Όπως αναλύσαμε στις προηγούμενες παραγράφους, για να ελαχιστοποιήσουμε τη συνάρτηση κόστους εκπαιδεύσαμε το δίκτυο ώστε να διαλέγει τους κατάλληλους πίνακες βαρών και διανύσματα μεροληψιών. Θα συμβολίζουμε ως p το διάνυσμα που θα αποθηκεύει το σύνολο των βαρών και μεροληψιών και θα γράφουμε $\mathrm{p} \in \mathbb{R}^{\mathrm{s}}$. Ακόμη θα γράφουμε τη συνάρτηση κόστους (2.15) ως $\mathrm{Cost}(\mathrm{p})$ με $\mathrm{Cost}: \mathbb{R}^{\mathrm{s}} \to \mathbb{R}$.

Η Μέθοδος Απότομης Καθόδου, είναι μια κλασσική μέθοδος βελτιστοποίησης που στοχεύει στον επαναληπτικό υπολογισμό μιας ακολουθίας διανυσμάτων που ανήκουν στον $\mathbb{R}^{\mathrm{s}}$ που συγκλίνουν σε ένα διάνυσμα που ελαχιστοποιεί τη συνάρτηση κόστους. Υποθέτουμε ότι το διάνυσμα που πληρεί τις παραπάνω προϋποθέσεις είναι το p. Διαλέγουμε βήμα $\Delta\mathrm{p}$ έτσι ώστε το επόμενο διάνυσμα $\mathrm{p} + \Delta\mathrm{p}$ να αντιπροσωπεύει τη βελτίωση. Εάν το $\Delta\mathrm{p}$ είναι αρκετά μικρό, τότε αγνοούμε όρους τάξης $\|\Delta\mathrm{p}\|^2$, με τη εφαρμογή σειρών Taylor έχουμε

$$\mathrm{Cost}(\mathrm{p} + \Delta\mathrm{p}) \approx \mathrm{Cost}(\mathrm{p}) + \sum_{\mathrm{r}=1}^{\mathrm{s}} \frac{\partial \mathrm{Cost}(\mathrm{p})}{\partial \mathrm{p_r}} \Delta\mathrm{p_r}. \quad (2.16)$$

Όπου ο όρος $\frac{\partial \mathrm{Cost}(\mathrm{p})}{\partial \mathrm{p_r}}$ αναφέρεται στη μερική παράγωγο της συνάρτησης κόστους σε σχέση με την r-παράμετρο. Επίσης, θα υποδηλώνουμε το την κλίση ως $\nabla\mathrm{Cost}(\mathrm{p}) \in \mathbb{R}^{\mathrm{s}}$ και θα έχουμε

$$(\nabla\mathrm{Cost}(\mathrm{p}))_{\mathrm{r}} = \frac{\partial \mathrm{Cost}(\mathrm{p})}{\partial \mathrm{p_r}}.$$

Τελικά, η σχέση (2.16) θα γίνει

$$\mathrm{Cost}(\mathrm{p} + \Delta\mathrm{p}) \approx \mathrm{Cost}(\mathrm{p}) + \nabla\mathrm{Cost}(\mathrm{p})^{\mathrm{T}}\Delta\mathrm{p}. \quad (2.17)$$

Στόχος μας είναι να διαλέξουμε κατάλληλα $\Delta\mathrm{p}$ έτσι ώστε ο όρος $\nabla\mathrm{Cost}(\mathrm{p})^{\mathrm{T}}$ να γίνεται αρνητικός. Μέσω της ανισότητας Cauchy-Schwarz για $\mathrm{f}, \mathrm{g} \in \mathbb{R}^{\mathrm{s}}$ έχουμε $\left|\mathrm{f}^{\mathrm{T}}\mathrm{g}\right| \leq \|\mathrm{f}\|_2\|\mathrm{g}\|_2$. Έτσι αρνητικός γίνεται ο όρος $\mathrm{f}^{\mathrm{T}}\mathrm{g}$ όταν $\mathrm{f} = -\mathrm{g}$.

Άρα για την σχέση (2.17) πρέπει να διαλέξουμε το $\Delta\mathrm{p}$ να κινείται στη κατεύθυνση $-\nabla\mathrm{Cost}(\mathrm{p})$. Έχουμε

$$\mathrm{p} \to \mathrm{p} - \mathrm{r}\nabla\mathrm{Cost}(\mathrm{p}) \quad (2.18)$$

όπου το r είναι μια αρκετά μικρή ποσότητα και την ονομάζουμε *ρυθμό μάθησης* (*learning rate*). Είμαστε πλέον σε θέση να εφαρμόσουμε τη μέθοδο απότομης καθόδου, σε αρχικά διανύσματα και η επανάληψη συνεχίζεται μέχρι να ικανοποιηθεί ένα κριτήριο διακοπής ή να επιτευχτεί ο μέγιστος αριθμός επαναλήψεων.

Η συνάρτηση κόστους (2.15) είναι ένα άθροισμα μεμονωμένων όρων πάνω στα δεδομένα εκπαίδευσης, που σημαίνει ότι η μερική παράγωγος $\nabla\mathrm{Cost}(\mathrm{p})$ είναι ένα άθροισμα μεμονωμένων μερικών παραγώγων πάνω στα δεδομένα.

Συγκεκριμένα, έστω

$$\mathrm{C}_{\mathrm{x}^{[\mathrm{i}]}} = \frac{1}{2}\left\|\mathrm{y}\left(\mathrm{x}^{[\mathrm{i}]}\right) - \mathrm{a}^{[\mathrm{L}]}\left(\mathrm{x}^{[\mathrm{i}]}\right)\right\|_2^2, \quad (2.19)$$

τότε από την (2.15), έχουμε

$$\nabla\mathrm{Cost}(\mathrm{p}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \nabla\mathrm{C}_{\mathrm{x}^{[\mathrm{i}]}}, \quad (2.20)$$

όπου N είναι ο αριθμός των *σημείων εκπαίδευσης* (*training points*). Ωστόσο, όταν ασχολούμαστε με μεγάλο αριθμό παραμέτρων και σημείων εκπαίδευσης, ο υπολογισμός του διανύσματος κλίσης (2.20) σε κάθε επανάληψη της μεθόδου απότομης καθόδου μπορεί να είναι υπολογιστικά δαπανηρός. Επομένως, μια λιγότερο δαπανηρή, για το σύστημα, εναλλακτική είναι η αντικατάσταση του μέσου όρου των επιμέρους κλίσεων σε όλα τα σημεία προπόνησης με την κλίση σε ένα μόνο, τυχαία επιλεγμένο, σημείο εκπαίδευσης. Αυτό οδηγεί στη μέθοδο της στοχαστικής κλίσης (stochastic gradient method), η οποία μπορεί να συνοψιστεί ως εξής:

1. Επιλέγουμε ένα τυχαίο ακέραιο i από το {1, 2, 3,…, N}.
2. Ενημερώνουμε το

$$p \rightarrow p - \mathrm{r}\nabla C_{x^{[i]}}(\mathrm{p}) \quad (2.21)$$

όπου είναι ο ρυθμός εκμάθησης. Σε κάθε βήμα, η *μέθοδος στοχαστικής κλίσης* (*stochastic gradient method*) χρησιμοποιεί ένα τυχαία επιλεγμένο σημείο εκπαίδευσης για να αναπαραστήσει ολόκληρο το σετ εκπαίδευσης. Καθώς η επανάληψη προχωρά, η μέθοδος συναντά περισσότερα σημεία εκπαίδευσης, με στόχο τη σημαντική μείωση στο κόστος ανά επανάληψη. Είναι σημαντικό να σημειωθεί ότι ακόμη και για πολύ μικρές τιμές του n , η ενημέρωση της (2.21) δεν είναι εγγυημένη ότι θα μειώσει τη συνάρτηση συνολικού κόστους, καθώς έχουμε αντικαταστήσει τη μέση τιμή με ένα μόνο δείγμα.

Η μέθοδος στοχαστικής κλίσης που εισάγεται στο (2.21) είναι μόνο μία από τις πολλές πιθανές παραλλαγές. Σε αυτήν την εκδοχή, ο δείκτης i της (2.21) επιλέγεται με τη δειγματοληψία της επανάθεσης, που σημαίνει ότι αφού χρησιμοποιηθεί ένα σημείο εκπαίδευσης, επιστρέφει στο σύνολο και έχει ίσες πιθανότητες να επιλεγεί ξανά. Μια άλλη επιλογή είναι να κάνετε δειγματοληψία χωρίς επανάθεση, επιλέγοντας κάθε ένα από τα N σημεία εκπαίδευσης με τυχαία σειρά. Αυτό ονομάζεται ολοκλήρωση μιας εποχής, η οποία περιλαμβάνει:

1. Το ανακάτεμα των ακεραίων {1, 2, 3,…,N} σε νέα σειρά $\{k_1, k_2, k_3, \dots, k_N\}$.
2. Για i=1 έως N, ενημερώνουμε την

$$p \rightarrow p - r\nabla C_{x^{[i]}}(\mathrm{p}). \quad (2.21)$$

Ένας συμβιβασμός μεταξύ αυτών των δύο προσεγγίσεων είναι η χρήση ενός μικρού μέσου όρου δείγματος κάνοντας βήματα χρησιμοποιώντας ένα υποσύνολο μεγέθους m, όπου το m<<N. Το υποσύνολο δημιουργείται:

1. Επιλέγοντας m ακεραίους $k_1, k_2, k_3, \dots, k_m$ ομοιόμορφα τυχαίους από το {1, 2, 3,…,N}
2. Ενημερώνοντας το

$$\mathrm{p} \rightarrow \mathrm{p} - \mathrm{r}\frac{1}{\mathrm{m}}\nabla \mathrm{C}_{\mathrm{x}[\mathrm{i}]}(\mathrm{p}). \quad (2.22)$$

Υπάρχει επίσης μια εναλλακτική λύση χωρίς επανάθεση όπου το σύνολο των δεδομένων εκπαίδευσης χωρίζεται τυχαία σε K διακριτές υποσύνολα, υποθέτοντας N = Km για κάποιο K, και τα χρησιμοποιεί όλα περιοδικά.

Η επιλογή του μεγέθους του υποσυνόλου και η μορφή τυχαιοποίησης καθορίζονται συχνά από τις απαιτήσεις των αρχιτεκτονικών υπολογιστών υψηλής απόδοσης κατά την εφαρμογή της μεθόδου στοχαστικής κλίσης σε μεγάλη κλίμακα. Είναι επίσης δυνατό να διαφοροποιηθούν δυναμικά αυτές οι επιλογές, μαζί με το ποσοστό εκμάθησης, καθώς προχωρά η εκπαίδευση για να επιταχυνθεί η σύγκλιση. Ένας επιτυχημένος αλγόριθμος για τον καθορισμό του ρυθμού εκμάθησης είναι ένα ουσιαστικό συστατικό για πρακτικούς υπολογισμούς.

2.10 Αλγόριθμος Οπισθοδιάδοσης (Backtracking Algorithm)

Θα χρησιμοποιήσουμε τη μέθοδο στοχαστικής κλίσης (stochastic gradient method) για να εκπαιδεύσουμε ένα τεχνητό νευρωνικό δίκτυο υπολογίζοντας τις μερικές παραγώγους της συνάρτησης κόστους (cost function), σε σχέση με τους πίνακες βάρους και τα διανύσματα μεροληψίας. Με αυτόν τον τρόπο, εγκαταλείπουμε το γενικό διάνυσμα p των παραμέτρων που χρησιμοποιήσαμε σε προηγούμενη ενότητα. Η μέθοδος στοχαστικής κλίσης εκμεταλλεύεται τη δομή της συνάρτησης κόστους (cost function), η οποία είναι ένας γραμμικός συνδυασμός μεμονωμένων όρων που διατρέχουν τα δεδομένα εκπαίδευσης (training points). Ως εκ τούτου, επικεντρωνόμαστε στον υπολογισμό αυτών των μεμονωμένων μερικών παραγώγων.

Έτσι, για ένα συγκεκριμένο σημείο δεδομένων εκπαίδευσης (training points), θεωρούμε τη συνάρτηση κόστους (cost function) $C_{x^{[i]}}$ στην εξίσωση (2.19) ως συνάρτηση των βαρών και των μεροληψιών στο νευρωνικό δίκτυο. Ως αποτέλεσμα, μπορούμε να απλοποιήσουμε την εξίσωση θέτοντας την ανεξάρτητη από το $x^{[i]}$ και να την γράψουμε ως εξής:

$$C = \frac{1}{2}\left\|y - a^{[L]}\right\|_2^2 \quad (2.23)$$

όπου $a^{[L]}$ είναι η έξοδος του νευρικού δικτύου και η εξάρτηση του C από βάρη και μεροληψίες προκύπτει μόνο μέσω του $a^{[L]}$.

Για να λάβουμε χρήσιμες εξισώσεις για μερικές παραγώγους, εισάγουμε δύο επιπλέον σύνολα μεταβλητών. Πρώτον, ορίζουμε το

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} \in \mathbb{R}^{n_l} \quad (2.24)$$

$\text{όπου } l = 2, 3, \dots, L$. Ως τη *σταθμισμένη είσοδο* (*weighted input*) για τον νευρώνα j στη φάση l. Επιπλέον, $W^{[l]}$ είναι ο πίνακας βάρους και $b^{[l]}$είναι το διάνυσμα μεροληψίας για τη φάση l. Δεύτερον, ορίζουμε το $\delta^{[l]}$ ως το σφάλμα (error) στον j-οστό νευρώνα στη φάση l, που δίνεται από το

$$\delta_j^{[l]} = \frac{\partial C}{\partial z_j^{[l]}} \quad (2.25)$$

$\text{όπου } 1 \leq j \leq n_l \text{ και } 2 \leq l \leq L$. Αυτή η έκφραση είναι μια ενδιάμεση ποσότητα που μετρά την ευαισθησία (sensitivity) της συνάρτησης κόστους στη σταθμισμένη είσοδο για τον νευρώνα j στη φάση l.

Ορίζουμε επίσης το εσωτερικό γινόμενο ή γινόμενο Hadamard δύο διανυσμάτων x και y στο $\mathbb{R}^n$, που συμβολίζεται με $x \circ y \in \mathbb{R}^n$, το οποίο σχηματίζεται πολλαπλασιάζοντας τις αντίστοιχες συνιστώσες των x και y κατά ζεύγη.

Χρησιμοποιώντας αυτούς τους ορισμούς, μπορούμε να εξαγάγουμε χρήσιμα αποτελέσματα εφαρμόζοντας τον κανόνα της αλυσίδας (chain rule). Ωστόσο, θα πρέπει να σημειώσουμε ότι ο όρος «σφάλμα» στο $\delta_j^{[l]}$ είναι κάπως διφορούμενος, ιδιαίτερα στα κρυφές φάσεις (hidden layers), καθώς δεν είναι ξεκάθαρο πόσο υπεύθυνος είναι κάθε νευρώνας για αποκλίσεις κατά την έξοδο. Επιπλέον, το $\delta_j^{[l]}$ δεν ποσοτικοποιεί τις αποκλίσεις απευθείας στη φάση της εξόδου L. Αντίθετα, μετρά την ευαισθησία της συνάρτησης κόστους στη σταθμισμένη είσοδο για τον νευρώνα j στη φάση l.

Λήμμα 2.1. Έχουμε

$\delta^{[l]} = \sigma'\left(z^{[l]}\right) \circ (a^L - y)$, (2.26)

$\delta^{[l]} = \sigma'\left(z^{[l]}\right) \circ (W^{[l+1]})^T \delta^{[l+1]}$ (2.27)

$\text{όπου } 2 \leq l \leq L - 1.$

$\frac{\partial C}{\partial b_j^{[l]}} = \delta_j^{[l]}$ (2.28)

$\text{όπου } 2 \leq l \leq L,$

$\frac{\partial C}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} a_k^{[l-1]}$ (2.29)

$\text{όπου } 2 \leq l \leq L.$

Απόδειξη: Αρχικά θέλουμε να αποδείξουμε την (2.26). Στη σχέση

$$a^{[l]} = \sigma\left(z^{[l]}\right) \text{ όπου } l = 2, 3, \dots, L$$

Για $l = L$ έχουμε ότι $a^{[L]} = \sigma\left(z^{[L]}\right)$ και

$$\frac{\partial a_j^{[L]}}{\partial z_j^{[l]}} = \sigma'\left(z^{[L]}\right).$$

Από την (2.23) προκύπτει

$$\frac{\partial C}{\partial a_j^{[L]}} = \frac{\partial}{\partial a_j^{[L]}} \frac{1}{2} \sum_{k=1}^{n_L} (y_k - a_k^{[L]})^2 = -\left(y_j - a_j^{[L]}\right).$$

Από τον κανόνα αλυσίδας έχουμε,

$$\delta_j^{[L]} = \frac{\partial C}{\partial z_j^{[L]}} = \frac{\partial C}{\partial a_j^{[L]}} \frac{\partial a_j^{[L]}}{\partial z_j^{[l]}} = \left(a_j^{[L]} - y_j\right) \sigma'\left(z^{[L]}\right) = \sigma'\left(z^{[l]}\right) \circ (a^L - y).$$

Για να αποδείξουμε τη σχέση (2.27), χρησιμοποιούμε τον κανόνα αλυσίδας για να μετατρέψουμε το $z_j^{[L]}$ σε $\{z_\kappa^{[l+1]}\}_{k=1}^{n_l}$. Εφαρμόζοντας τον κανόνα αλυσίδας και τη σχέση (2.25) έχουμε,

$$\delta_j^{[l]} = \frac{\partial C}{\partial z_j^{[L]}} = \sum_{k=1}^{n_l+1} \frac{\partial C}{\partial z_k^{[l+1]}} \frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}} = \sum_{k=1}^{n_l+1} \delta_k^{[l+1]} \frac{\partial z_k^{[l+1]}}{\partial z_j^{[L]}}.$$

(2.30)

Από την (1.24) γνωρίζουμε ότι το $z_k^{[l+1]}$ συνδέεται με το $z_k^{[l]}$ μέσω της παρακάτω σχέσης

$$z_k^{[l+1]} = \sum_{s=1}^{n_l} w_{ks}^{[l+1]} \sigma\left(z_s^{[l]}\right) + b_k^{[l+1]}.$$

Ως εκ τούτου,

$$\frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}} = w_{kj}^{[l+1]} \sigma'\left(z_j^{[l]}\right).$$

Άρα η (2.30) γίνεται,

$$\delta_j^{[l]} = \sum_{k=1}^{n_l+1} \delta_k^{[l+1]} w_{kj}^{[l+1]} \sigma'\left(z_j^{[l]}\right),$$

Ή

$$\delta_j^{[l]} = \sigma'\left(z_j^{[l]}\right)\left(\left(W^{[l+1]}\right)^T \delta^{[l+1]}\right)_j.$$

Για να δείξουμε τη σχέση (2.28) θα βασιστούμε στη σχέση που έχει προκύψει από τα παραπάνω

$$z_j^{[l]} = \left(W^{[l]} \sigma(z^{[l-1]})\right)_j + b_j^{[l]}.$$

Εφόσον το $z^{[l-1]}$ είναι ανεξάρτητο από το $b_j^{[l]}$, έχουμε ότι:

$$\frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} = 1.$$

Από τον κανόνα της αλυσίδας έχουμε,

$$\frac{\partial C}{\partial b_j^{[l]}} = \frac{\partial C}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} = \frac{\partial C}{\partial z_j^{[l]}} = \delta_j^{[l]},$$

που προκύπτει από τον ορισμό (2.25).

Τέλος για να αποδείξουμε τη σχέση (2.29) θα ξεκινήσουμε από τη σχέση (2.24),

$$z_j^{[l]} = \sum_{k=1}^{n_l-1} w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]},$$

Το οποίο δίνει,

$$\frac{\partial z_j^{[l]}}{\partial w_{j\kappa}^{[l]}} = a_k^{[l-1]},$$

(2.31)

ανεξάρτητο από το j και

$$\frac{\partial z_s^{[l]}}{\partial w_{j\kappa}^{[l]}} = 0 \ \ \gamma\iota\alpha \ s \neq j.$$

(2.32)

Με άλλα λόγια, οι εξισώσεις (2.31) και (2.32) προέρχονται θεωρώντας ότι ο j-οστός νευρώνας στη φάση l χρησιμοποιεί αποκλειστικά τα βάρη από την j-οστή σειρά του πίνακα $W^{[l]}$ και εφαρμόζει αυτά τα βάρη γραμμικά. Στη συνέχεια εφαρμόζεται ο κανόνας της αλυσίδας για να προκύψουν οι εξισώσεις (2.31) και (2.32).

$$\frac{\partial C}{\partial w_{j\kappa}^{[l]}} = \sum_{s=1}^{n_l} \frac{\partial C}{\partial z_s^{[l]}} \frac{\partial z_s^{[l]}}{\partial w_{j\kappa}^{[l]}} = \frac{\partial C}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial w_{j\kappa}^{[l]}} = \frac{\partial C}{\partial z_j^{[l]}} a_k^{[l-1]} = \delta_j^{[l]} a_k^{[l-1]}. \qquad \blacksquare$$

Συμπεράσματα: Από το Λήμμα 2.1 που αξίζει να σημειωθεί, όπως ο αλγόριθμος back propagation που προκύπτει, ο οποίος χρησιμοποιείται για τον υπολογισμό των κλίσεων σε ένα νευρωνικό δίκτυο κατά τη διάρκεια της εκπαίδευσης. Το Λήμμα 2.1 περιγράφει τα διαφορετικά στάδια που εμπλέκονται στην *back propagation*, συμπεριλαμβανομένου της *forward pass,* το οποίο υπολογίζει τις ενεργοποιήσεις σε κάθε φάσης του δικτύου, και της *backward pass*, που υπολογίζει τις κλίσεις της συνάρτησης κόστους σε σχέση με τα βάρη και τις μεροληψίες σε κάθε φάση.

2.11 Διαδικασίες Μάθησης των ΤΝΔ

Βασικός στόχος των τεχνητών νευρωνικών δικτύων είναι να αντιμετωπίζουν συγκεκριμένα προβλήματα που τους ανατίθενται ή να εκτελούν αυτόνομα ορισμένες διεργασίες, όπως για παράδειγμα την αναγνώριση εικόνων. Για να αποκτήσουν την ικανότητα, πρέπει να εκπαιδευτούν σωστά. Η διαδικασία της μάθησης αποτελεί το κύριο χαρακτηριστικό των νευρωνικών δικτύων. Τα τεχνητά νευρωνικά δίκτυα εκτελούν δύο βασικές λειτουργίες, τη μάθηση και την ανάκλαση. Με τον όρο "ανάκλαση" εννοείται η διαδικασία υπολογισμού της εξόδου βάσει ενός δεδομένου διανύσματος εισόδου και των συναπτικών βαρών. Από την άλλη πλευρά, η "μάθηση" αφορά τη διαδικασία προσαρμογής των βαρών, έτσι ώστε για μια δεδομένη είσοδο να παράγεται η επιθυμητή έξοδος. Μέσα από τη μάθηση, στοχεύεται στη συνεχή βελτίωση της απόδοσης των νευρωνικών δικτύων.

Συνεπώς, ένα νευρωνικό δίκτυο κατακτά τη μάθηση με τη χρήση δειγμάτων εκπαίδευσης και ενός αλγορίθμου εκπαίδευσης. Υπάρχουν πολλοί διαφορετικοί αλγόριθμοι μάθησης, όπως οι αλγόριθμοι οπισθοδρόμησης, ομαδοποίησης και δέντρων αποφάσεων, με τους οποίους ένα δίκτυο μαθαίνει σταδιακά. Οι αλγόριθμοι αυτοί εκτελούν επαναληπτικές διαδικασίες με σκοπό τη μείωση του σφάλματος μεταξύ της επιθυμητής και της πραγματικής εξόδου που παράγει το δίκτυο. Για να επιτευχθεί αυτό, τα νευρωνικά δίκτυα εκπαιδεύονται με διάφορες μεθόδους, προκειμένου να προσαρμόσουν τις τιμές των βαρών και των κατωφλιών του δικτύου.

Έτσι, τα ΤΝΔ ανάλογα με τους διαφορετικούς τρόπους αλλαγής των παραμέτρων που χρησιμοποιούνται, χωρίζονται σε 2 κατηγορίες μάθησης:

1) Επιβλεπόμενη Μάθηση (*Supervised Learning*)
2) Μη-Επιβλεπόμενη Μάθηση (*Unsupervised Learning*)

Επιγραμματικά, η διαδικασία της μάθησης που ακολουθεί ένα ΤΝΔ, αποτελείται από τα ακόλουθα βήματα:

1) Αρχικά το ΤΝΔ, διεγείρεται από το περιβάλλον του.
2) Στην συνέχεια, το ΤΝΔ ως αποτέλεσμα αυτής της διέγερσης, υλοποιεί μεταβολές στις παραμέτρους του.
3) Τέλος, το ΤΝΔ, αλληλεπιδρά με το νέο περιβάλλον του με καινούριο τρόπο, λόγω των μεταβολών που επήλθαν, μέσω των αλλαγών των παραμέτρων του.

Παρακάτω, περιγράφονται οι κατηγορίες μάθησης και ο τρόπος λειτουργίας τους:

1) Η μέθοδος της εποπτευόμενης μάθησης αποτελεί την πλέον κοινή πρακτική κατά την εκπαίδευση των νευρωνικών δικτύων. Κατά την εποπτευόμενη μάθηση, ένα πρόσωπο ή σύστημα - που ονομάζεται "εκπαιδευτής" - παρέχει στο δίκτυο τις εισόδους και τους επιθυμητούς στόχους που θέλει το δίκτυο να μάθει. Οι είσοδοι και οι αντίστοιχοι στόχοι αποτελούν τα παραδείγματα που χρησιμοποιούνται για την εκπαίδευση του νευρωνικού δικτύου. Αρχικά, τα βάρη των συνάψεων, τα οποία συμβολίζονται ως W έχουν τυχαίες τιμές για τις διάφορες εισόδους. Κατά τη διάρκεια της εκπαίδευσης, τα βάρη αλλάζουν ώστε να μειωθεί το σφάλμα (η απόκλιση) μεταξύ του επιθυμητού στόχου και της πραγματικής εξόδου του δικτύου. Κατά την προσπάθεια αύξησης της αποδοτικότητας του δικτύου, απαιτείται η διαμόρφωση πολλών διαφορετικών συνδυασμών βαρών, γεγονός που απαιτεί χρονοβόρο υπολογιστικό χρόνο. Γενικά, κατά την εκπαίδευση ενός δικτύου με εποπτευόμενη μάθηση, ένα πρόσωπο ή σύστημα επηρεάζει το περιβάλλον του δικτύου (βήμα 1), κινητοποιώντας το να αλλάξει τη δομή του (βήμα 2), με σκοπό να φτάσει σε μέγιστη απόδοση όπου το δίκτυο είναι σε θέση να φτάσει (βήμα 3).
2) Στο πλαίσιο της μη-επιβλεπόμενης μάθησης, αλλάζουν τα δεδομένα καθώς απομακρύνεται ο ρόλος του "εκπαιδευτή", όπως αυτός έχει καθιερωθεί στην επιβλεπόμενη μάθηση. Σε αντίθεση με τη διαδικασία της επιβλεπόμενης μάθησης, εδώ γίνεται παροχή πληροφορίας στο δίκτυο χωρίς να παρέχονται συγκεκριμένοι στόχοι εξόδων. Αυτό οδηγεί σε μια διαφορετική δυναμική κατά τη διάρκεια της εκπαίδευσης, όπου δεν υπάρχει περιορισμός ή σύγκριση του σφάλματος. Στην ουσία, το δίκτυο δεν εξαρτάται από εξωτερικές παραμέτρους για να προσαρμόσει τα βάρη του. Αντ' αυτού, ακολουθεί μια συγκεκριμένη διαδικασία που οδηγεί στην εκπαίδευσή του. Στην μη-επιβλεπόμενη μάθηση, το δίκτυο αναζητά τάσεις και μοτίβα στα εισαγόμενα δεδομένα μέσω ενός εσωτερικού ελέγχου. Ο στόχος είναι οι εξόδοι του δικτύου να αντικατοπτρίζουν τα ίδια χαρακτηριστικά που παρουσιάζουν οι εισόδοι. Με αυτό τον τρόπο, το δίκτυο ελέγχει και διορθώνει τα σφάλματα στα δεδομένα χρησιμοποιώντας έναν μηχανισμό ανάδρασης (feedback). Το δίκτυο αυτορυθμίζεται, προσαρμόζοντας από μόνο του τα βάρη του. Τελικά, η μη-επιβλεπόμενη μάθηση αν και λιγότερο συνηθισμένη, παραμένει εξαιρετικά χρήσιμη για ειδικές περιπτώσεις.

2.12 Εφαρμογές των Νευρωνικών Δικτύων

Τα τελευταία χρόνια, οι τεχνητές νευρωνικές δικτύωσεις έχουν κερδίσει αυξανόμενο ενδιαφέρον από τους ερευνητές λόγω της δομής και των λειτουργιών τους. Αυτές οι τεχνολογίες βρίσκονται στο επίκεντρο της ανάπτυξης δικτύων, ενισχύονται από τις υψηλές υπολογιστικές ταχύτητες και είναι ικανές να αντιμετωπίσουν πολύπλοκες μη γραμμικές λειτουργίες. Αυτές επίσης μπορούν να αναγνωρίσουν δυσδιάκριτες σχέσεις ανάμεσα σε μεταβλητές ποσότητες, που είναι δύσκολο να περιγραφούν με τρόπους παραδοσιακής μοντελοποίησης. Αυτές οι τεχνολογίες έχουν εφαρμογές σε διάφορους τομείς, παρέχοντας ευκολία στην υλοποίηση, υψηλή αξιοπιστία λειτουργίας και άμεση απόκριση όταν ενσωματώνονται σε υλικούς πόρους. Πολλές από αυτές τις εφαρμογές είναι ήδη διαθέσιμες στην αγορά και χρησιμοποιούνται ευρέως. Κάποιες από τις δυνατότητες των τεχνητών νευρωνικών δικτύων περιλαμβάνουν την αναγνώριση προτύπων, την προσέγγιση συναρτήσεων, την πρόβλεψη, τη βελτιστοποίηση και τον αυτοματοποιημένο έλεγχο. Σε ακόμα περισσότερες πρακτικές εφαρμογές, όπως η Οικονομία, η Επιστήμη των Υπολογιστών, η Επεξεργασία Σημάτων και η Ιατρική, οι τεχνητές νευρωνικές δικτύωσεις βρίσκουν ευρεία χρήση.

Η Οικονομία αποτελεί έναν τομέα όπου τα τεχνητά νευρωνικά δίκτυα αποκτούν ιδιαίτερη σημασία, παρέχοντας λύσεις σε σημαντικά θέματα. Ένα παράδειγμα είναι η χρήση τους στις τραπεζικές διεργασίες για την εκτίμηση κινδύνου στη χορήγηση στεγαστικών δανείων. Στην περίπτωση αυτή, ο στόχος είναι να αξιολογηθεί η δυνατότητα ενός δανειολήπτη να αποπληρώσει το δάνειο που ζητά. Για να εξαλειφθεί οποιοδήποτε ρίσκο λάθους, οι τράπεζες δημιουργούν προσαρμοσμένα διαγράμματα εισοδήματος-φερεγγυότητας για κάθε δανειολήπτη. Αυτά τα διαγράμματα απεικονίζουν δύο αντίθετες περιοχές: μία όπου ο δανειολήπτης έχει υψηλό εισόδημα και φερεγγυότητα, καθιστώντας την χορήγηση δανείου προφανή, και μία άλλη όπου οι συνθήκες ειναι αντίστροφες. Οι τεχνητές νευρωνικές δίκτυα εισέρχονται εδώ για να παράσχουν λύσεις. Ένα παράδειγμα είναι το πρόγραμμα νευρωνικού δικτύου με το όνομα "Νέστωρ" (Nestor)[11], το οποίο εκπαιδεύεται χρησιμοποιώντας χιλιάδες αιτήσεις δανείων, οι μισές από τις οποίες εγκρίνονται και οι άλλες μισές απορρίπτονται. Το σύστημα συγκρίνει τα νέα δεδομένα μιας αίτησης με τη βάση δεδομένων και αναζητά στοιχεία που αφορούν την κινδυνολογία της χορήγησης. Μέσω αυτής της διαδικασίας, καταλήγει σε μια απόφαση. Φυσικά, το πρόγραμμα "Νέστωρ" δεν είναι πάντα απόλυτα αποτελεσματικό, αλλά συνολικά θεωρείται επιτυχημένο για τις αποφάσεις που λαμβάνει.

Τα νευρωνικά δίκτυα έχουν σημειώσει εντυπωσιακή επιτυχία στον τομέα της επεξεργασίας δεδομένων. Ανήκουν στην κατηγορία των τεχνολογιών που εξειδικεύονται στην αναγνώριση προτύπων σε εικόνες, κείμενα και άλλες μορφές πληροφορίας. Ένα παράδειγμα αυτής της λειτουργίας είναι το πρόγραμμα Omnipage[12], που δημιουργήθηκε από την εταιρεία Scansoft το 1994. Αυτό το πρόγραμμα εκτελείται σε απλούς υπολογιστές και ασχολείται με την ανάγνωση κειμένων. Σε αυτήν τη διαδικασία, ένας σαρωτής διαβάζει το κείμενο και το πρόγραμμα το μετατρέπει σε πρότυπο κώδικα ASCII για περαιτέρω επεξεργασία. Το Omnipage παρουσιάζει ενθαρρυντικά αποτελέσματα, ακόμα και σε περιπτώσεις παραμορφωμένων χαρακτήρων. Μια άλλη πρωτοποριακή εφαρμογή των νευρωνικών δικτύων είναι το πρόγραμμα NETtalk, που αναπτύχθηκε από την εταιρεία Nestor. Αυτό το πρόγραμμα μπορεί να αναγνωρίσει και να μετατρέψει αυτόματα ιαπωνική γραφή σε αγγλικά. Αρχικά, το πρόγραμμα μπορούσε να αναγνωρίσει 2500 λέξεις με ακρίβεια 92%. Καθώς εξελίσσονταν, έγινε επιτυχημένο στο να αναγνωρίζει και να αποδίδει την ομιλία σε ποσοστό 95%. Μία σύγχρονη εφαρμογή των νευρωνικών δικτύων στην επιστήμη των υπολογιστών είναι το Google Neural Machine Translation (GNMT), που παρουσιάστηκε το 2016. Αυτό το πρόγραμμα χρησιμοποιεί νευρωνικά δίκτυα για να βελτιώσει την ακρίβεια του Google Translate στη μετάφραση προτάσεων και κειμένων, διευκολύνοντας τη μετατροπή του κειμένου σε φωνή και αντίστροφα.

Ένα πολύ χρήσιμο παράδειγμα της εφαρμογής των νευρωνικών δικτύων στην καθημερινή ζωή είναι η δυνατότητά τους να απαλείφουν τις παρεμβολές (ήχους) από τις τηλεφωνικές γραμμές. Για την επίτευξη αυτής της λειτουργίας, χρησιμοποιείται ένα φίλτρο που ταυτόχρονα περιορίζει τα σφάλματα. Το πρώτο από αυτά τα φίλτρα είναι γνωστό ως "Adaline" και δημιουργήθηκε από τον ερευνητή B. Widrow [10]. Η λειτουργία αυτή βασίζεται στο μοντέλο του απλού αισθητήρα

(Perceptron). Αυτό το σύστημα επιτρέπει τον καθαρισμό της τηλεφωνικής γραμμής από ανεπιθύμητους ήχους, παράλληλα με την ελαχιστοποίηση των σφαλμάτων που μπορεί να προκύψουν.

Η Ιατρική αποτελεί έναν εξαιρετικά σημαντικό τομέα όπου τα νευρωνικά δίκτυα παρουσιάζουν αξιοσημείωτες λύσεις. Αυτά τα δίκτυα έχουν αποδειχθεί πολύ αποτελεσματικά στη διενέργεια δύσκολων χειρουργικών επεμβάσεων και στην ανάγνωση ακτινογραφιών. Επιπλέον, αποτελούν αξιόλογο εργαλείο για τους καρδιολόγους, καθώς μπορούν να ανιχνεύσουν τα επίπεδα ενζύμων της καρδιάς και να αναγνωρίσουν διάφορα καρδιακά συμπτώματα. Η αξιόπιστη διάγνωση παίζει ένα ζωτικό ρόλο στην αντιμετώπιση των προβλημάτων υγείας και συχνά σώζει ζωές. Ένα εντυπωσιακό παράδειγμα της λειτουργίας των νευρωνικών δικτύων είναι η χρήση τους για την ανάλυση της ομιλίας προκειμένου να δημιουργηθούν ακουστικές βαρηκοΐες. Επιπλέον, η δυνατότητα πρόβλεψης των αντιδράσεων των οργανισμών σε διάφορες φαρμακευτικές αγωγές αποτελεί εξαιρετική εφαρμογή των νευρωνικών δικτύων.

Με όλες αυτές τις ενδείξεις και τα πραγματικά οφέλη που προσφέρουν τα τεχνητά νευρωνικά δίκτυα, είναι προφανές ότι η επιστημονική αυτή περιοχή θα επηρεάσει σημαντικά την καθημερινή ζωή των ανθρώπων.

Κεφάλαιο 3. Τεχνητά Νευρωνικά Δίκτυα για την λύση Διαφορικών Εξισώσεων

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε τη χρήση των τεχνητών νευρωνικών δικτύων (ΤΝΔ) για την επίλυση διαφορικών εξισώσεων. Οι διαφορικές εξισώσεις χρησιμοποιούνται για τη μοντελοποίηση πολλών φυσικών φαινομένων και η επίλυσή τους με ακρίβεια και αποτελεσματικότητα είναι σημαντική για ένα ευρύ φάσμα εφαρμογών, από τη μηχανική έως τα οικονομικά.

Οι κλασικές αριθμητικές μέθοδοι για την επίλυση διαφορικών εξισώσεων (ΔΕ) μπορεί να είναι υπολογιστικά δαπανηρές, ειδικά για πολύπλοκα μη-γραμμικά συστήματα. Τα ΤΝΔ, από την άλλη πλευρά, έχουν τη δυνατότητα να προσφέρουν πιο γρήγορες και ακριβείς λύσεις, καθιστώντας τα μια εναλλακτική λύση στις κλασσικές μεθόδους. Θα παρουσιάσουμε επίσης διάφορες αρχιτεκτονικές ΤΝΔ και συζητήσουμε πώς μπορούν να χρησιμοποιηθούν για την επίλυση διαφορετικών τύπων διαφορικών εξισώσεων.

3.2 Περιγραφή της μεθόδου

Αρχικά χρησιμοποιούμε τον ακόλουθο γενικό ορισμό μιας διαφορικής εξίσωσης, που υπόκειται σε συγκεκριμένες συνοριακές συνθήκες (*boundary conditions: BC's*) όπως οι συνθήκες Dirichlet ή Neumann:

$$G\left(\vec{x}, \Psi(\vec{x}), \nabla\Psi(\vec{x}), \nabla^2\Psi(\vec{x})\right) = 0, \vec{x} \in D \quad (3.1)$$

όπου $\vec{\mathrm{x}} = (\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}) \in \mathbb{R}^\mathrm{n}$, $\mathrm{D} \subset \mathbb{R}^\mathrm{n}$ είναι το πεδίο ορισμού και $\Psi(\vec{\mathrm{x}})$ είναι η λύση που θα υπολογιστεί.

Για να λάβουμε μια λύση, χρησιμοποιούμε τη μέθοδο συνεγγισμού (*collocation method*), η οποία περιλαμβάνει τη διακριτοποίηση του πεδίου ορισμού D και των ορίων S του σε σύνολα σημείων $\widehat{\mathrm{D}}$ και $\widehat{\mathrm{S}}$, αντίστοιχα. Αυτό μετατρέπει το πρόβλημα σε ένα σύστημα εξισώσεων:

$$G\left(\vec{x}_i, \Psi(\vec{x}_i), \nabla\Psi(\vec{x}_i), \nabla^2\Psi(\vec{x}_i)\right) = 0, \forall\vec{x}_i \in \widehat{D} \quad (3.2)$$

όπου υπόκεινται στους περιορισμούς που επιβάλλουν οι συνοριακές συνθήκες. Εάν μια δοκιμαστική λύση συμβολίζεται με $\Psi_t(\vec{x}, \vec{p})$ ρυθμιζόμενες παραμέτρους $\vec{p}$, το πρόβλημα μπορεί περαιτέρω να μετατραπεί σε:

$$\min_{\vec{p}} \sum_{\vec{x}_i \in \widehat{D}} \left(G\left(\vec{x}_i, \Psi_t(\vec{x}_i, \vec{p}), \nabla\Psi_t(\vec{x}_i, \vec{p}), \nabla^2\Psi_t(\vec{x}_i, \vec{p})\right)\right)^2 \quad (3.3)$$

υπόκεινται στους περιορισμούς που επιβάλλουν οι συνοριακές συνθήκες.

Στην προτεινόμενη προσέγγιση, η δοκιμαστική λύση Ψ_t περιλαμβάνει τη χρήση ενός feedforward νευρωνικού δικτύου, όπου οι παράμετροι αντιστοιχούν στα βάρη και τις μεροληψίες της αρχιτεκτονικής του δικτύου. Επιλέγουμε μια συγκεκριμένη μορφή για τη δοκιμαστική

συνάρτηση$\Psi_t(\vec{x})$ που ικανοποιεί τις συνοριακές συνθήκες. Η δοκιμαστική συνάρτηση εκφράζεται ως άθροισμα δύο όρων:

$$\Psi_t(\vec{x}) = A(\vec{x}) + F\big(\vec{x}, N(\vec{x}, \vec{p})\big) \quad (3.4)$$

Ο όρος $N(\vec{x}, \vec{p})$ είναι ένα feedforward νευρωνικό δίκτυο μίας εξόδου με παραμέτρους $\vec{p}$ και n μονάδες εισόδου που τροφοδοτούνται με το διάνυσμα εισόδου $\vec{x}$. Ο όρος $A(\vec{x})$ ικανοποίει τις συνοριακές συνθήκες επειδή περιέχει αμετάβλητες παραμέτρους. Ο όρος F χρησιμοποιεί ένα νευρωνικό δίκτυο του οποίου τα βάρη και οι μεροληψίες προσαρμόζονται για να αντιμετωπίσουν το πρόβλημα ελαχιστοποίησης. Το αρχικό πρόβλημα βελτιστοποίησης με περιορισμούς έχει αναχθεί σε ένα πρόβλημα χωρίς περιορισμούς λόγω της επιλογής της δοκιμαστικής λύσης η οποία ικανοποιεί εκ κατασκευής τις συνοριακές συνθήκες.

3.2.1 Υπολογισμός κλίσης

Σε αυτήν την ενότητα, θα περιγράφουμε τον υπολογισμό των κλίσεων της δοκιμαστικής λύσης σε σχέση με τις ρυθμιζόμενες παραμέτρους (δηλαδή, τα βάρη και τις μεροληψίες του νευρωνικού δικτύου). Αυτές οι διαβαθμίσεις απαιτούνται για τη βελτιστοποίηση της δοκιμαστικής λύσης χρησιμοποιώντας μεθόδους βελτιστοποίησης που βασίζονται στη κλίση.

Η αποτελεσματική ελαχιστοποίηση της (3.3) ανάγεται στη διαδικασία εκπαίδευσης του νευρωνικού δικτύου, όπου το σφάλμα που αντιστοιχεί σε κάθε διάνυσμα εισόδου $\vec{x}_i$ είναι η τιμή $G(\vec{x}_i)$ η οποία πρέπει να γίνει μηδενική. Ο υπολογισμός αυτής της τιμής σφάλματος περιλαμβάνει όχι μόνο την έξοδο του δικτύου αλλά και τα παράγωγα της εξόδου σε σχέση με οποιαδήποτε από τις εισόδους του. Επομένως, στον υπολογισμό της κλίσης του σφάλματος σε σχέση με τα βάρη του δικτύου, πρέπει να υπολογίσουμε όχι μόνο τη κλίση του δικτύου αλλά και τις κλίσεις των παραγώγων του δικτύου σε σχέση με τις εισόδους του.

Θεωρούμε ένα πολυστρωματικό Perceptron με n εισόδους, ένα κρυμμένο στρώμα με H σιγμοειδείς συναρτήσεις και μια γραμμική έξοδο. Για δεδομένο διάνυσμα εισόδου $\vec{x} = (x_1, \dots, x_n)$ τότε η έξοδος του δικτύου είναι $N = \sum_{i=1}^{H} b_i \sigma(z_i)$ όπου $z_i = \sum_{i=1}^{n} w_{ij} x_j + u_i$. Το w_{ij} συμβολίζει το βάρος της εισόδου j στο κρυμμένο στρώμα i και το u_i συμβολίζει το βάρος από το κρυμμένο στρώμα i στην έξοδο. Το b_i συμβολίζει τη μεροληψία του κρυμμένου στρώματος i, και $\sigma(z)$ είναι η σιγμοειδής συνάρτηση ενεργοποίησης. Για να υπολογίσουμε τις κλίσεις, ορίζουμε πρώτα την υπολειπόμενη συνάρτηση ως τη διαφορά μεταξύ της διαφορικής εξίσωσης και της δοκιμαστικής λύσης, που υπολογίζεται στα σημεία συνεγγισμού (*collocation points*). Συγκεκριμένα, ορίζουμε την υπολειπόμενη συνάρτηση για το i-οστό σημείο συνεγκατάστασης ως:

$$\frac{\partial^k N}{\partial x_j^k} = \sum_{i=1}^{H} b_i w_{ij}^k \sigma_i^{(k)} \quad (3.5)$$

όπου υποδηλώνει το i-οστό σημείο συνεγγισμού και υποδηλώνει την αντίστοιχη τιμή της ανεξάρτητης μεταβλητής (δηλαδή χρόνο ή χώρο). Ακόμη είναι η δοκιμαστική λύση που υπολογίζεται στο i-οστό σημείο συντοπισμού και ο διαφορικός τελεστής που εφαρμόζεται στη δοκιμαστική. Τέλος, η λύση και είναι η δεξιά πλευρά της διαφορικής εξίσωσης που υπολογίζεται στο i-οστό σημείο συντοπισμού.

Στη συνέχεια, θα διαφοροποιήσουμε την υπολειπόμενη συνάρτηση σε σχέση με κάθε ρυθμιζόμενη παράμετρο στη δοκιμαστική λύση. Αυτό αποδίδει τη διαβάθμιση της υπολειπόμενης συνάρτησης σε σχέση με κάθε παράμετρο.

$$\frac{\partial^{\lambda_1}}{\partial x_1^{\lambda_1}}\frac{\partial^{\lambda_2}}{\partial x_2^{2}}\dots\frac{\partial^{\lambda_n}}{\partial x_n^{\lambda_n}}N = \sum_{i=1}^{n} b_i P_i \sigma_i^{(\Lambda)}. \quad (3.6)$$

όπου,

$$P_i = \prod_{k=1}^{n} w_{ik}^{\lambda_k} \quad (3.7)$$

και $\Lambda = \sum_{i=1}^{n} \lambda_{i.}$

Οι διαβαθμίσεις της δοκιμαστικής λύσης (3.6) σε σχέση με τα βάρη w_{ij} και τις μεροληψίες b_i του νευρωνικού δικτύου $N_g(x)$ μπορούν να υπολογιστούν αποτελεσματικά χρησιμοποιώντας τον backpropagation αλγόριθμο, ο οποίος είναι ένας καλά καθιερωμένος αλγόριθμος για τον υπολογισμό των κλίσεων στα νευρωνικά δίκτυα. Η feedforward περιλαμβάνει τον υπολογισμό των διαβαθμίσεων της εξόδου του νευρωνικού δικτύου (δηλαδή της δοκιμαστικής λύσης) σε σχέση με τις εισόδους του (δηλαδή τα βάρη και τις μεροληψίες) χρησιμοποιώντας τον κανόνα της αλυσίδας.

Επομένως, η κλίση του N_g σε σχέση με τις παραμέτρους του αρχικού δικτύου μπορεί εύκολα να υπολογιστεί ως:

$$\frac{\partial N_g}{\partial b_i} = P_i \sigma_i^{(\Lambda)} \quad (3.8)$$

$$\frac{\partial N_g}{\partial b_i} = b_i P_i \sigma_i^{(\Lambda+1)} \quad (3.9)$$

$$\frac{\partial N_g}{\partial w_{ij}} = x_j b_i P_i \sigma_i^{(\Lambda+1)} + b_i \lambda_j w_{ij}^{\lambda_j - 1}\left(\prod_{k\neq 1, k\neq j} w_{ik}^{\lambda_k}\right)\sigma_i^{(\Lambda)}. \quad (3.10)$$

Μόλις υπολογιστούν οι κλίσεις της υπολειπόμενης συνάρτησης σε σχέση με τις ρυθμιζόμενες παραμέτρους, μπορούν να χρησιμοποιηθούν για την ενημέρωση των τιμών αυτών των παραμέτρων κατά τη βελτιστοποίηση χρησιμοποιώντας μεθόδους που βασίζονται σε κλίση, όπως η μέθοδος απότομης καθόδου (*conjugate gradient method*). Ο στόχος της βελτιστοποίησης είναι η ελαχιστοποίηση της υπολειπόμενης συνάρτησης στο πεδίο ορισμού, συνυπολογίζοντας τις συνοριακές συνθήκες της διαφορικής εξίσωσης.

3.3 Απεικόνιση της Μεθόδου

3.3.1 Λύση μιας ΣΔΕ και ενός συστήματος ΣΔΕ

Για να δείξουμε την προτεινόμενη μέθοδο, ξεκινάμε εξετάζοντας μια συνήθη διαφορική εξίσωση πρώτης τάξης που δίνεται ως:

$$\frac{dy(x)}{dx} = f(x, y) \quad (3.11)$$

με $x \in [0,1]$ και αρχική συνθήκη $y(0) = A$.
Κατασκευάζουμε μια δοκιμαστική λύση για το ΣΔΕ ως εξής:

$$y_t(x) = A + xN(x, \vec{p}) \quad (3.12)$$

Όπου $N(x, \vec{p})$ είναι η έξοδος ενός νευρωνικού δικτύου τροφοδοσίας με μία μονάδα εισόδου για το x και βάρη $\vec{p}$. Σημειώστε ότι η $y_t(x)$ ικανοποιεί αρχική συνθήκη από την δομή του προβλήματος. Η ποσότητα σφάλματος που πρέπει να είναι ελαχιστοποιείται δίνεται από:

$$E[\vec{p}] = \sum_i \{\frac{dy_t(x_i)}{dx} - f(x_i, y_t(x_i))\}^2 \quad (3.13)$$

όπου τα x_i είναι σημεία στο διάστημα [0, 1]. Από

$$\frac{dy_t(x_i)}{dx} = N(x, \vec{p}) + x\frac{dN(x, \vec{p})}{dx}$$

είναι εύκολο να υπολογιστεί το κλίση του σφάλματος σε σχέση με τις παραμέτρους $\vec{p}$ που χρησιμοποιούν εξισώσεις (3.5)–(3.10).

Μπορούμε να υπολογίσουμε τη κλίση του σφάλματος σε σχέση με τις παραμέτρους χρησιμοποιώντας τον κανόνα αλυσίδας και τον αλγόριθμο backpropagation. Η ίδια διαδικασία μπορεί να εφαρμοστεί σε ΣΔΕ δεύτερης τάξης με αρχικές ή οριακές συνθήκες. Για συστήματα ΣΔΕ πρώτης τάξης, κατασκευάζουμε ένα νευρωνικό δίκτυο για κάθε δοκιμαστική λύση και ελαχιστοποιούμε τη συνάρτηση σφάλματος χρησιμοποιώντας την ίδια προσέγγιση.

ΣΔΕ δεύτερης τάξης:

$$\frac{d^2y(x)}{dx^2} = f\left(x, y, \frac{dy(x)}{dx}\right). \quad (3.14)$$

Με αρχικές συνθήκες $y(0) = A$ και $\frac{dy(0)}{dx} = A'$, η λύση είναι

$$y_t(x) = A + A'x + x^2N(x, \vec{p}). \quad (3.15)$$

Για δύο συνοριακά σημεία Dirichlet: $y(0) = A$ και $y(1) = B$, η λύση γράφεται ως εξής:

$$y_t(x) = A(1-x) + Bx + x(1-x)N(x,\vec{p}). \quad (3.16)$$

Στις δύο παραπάνω περιπτώσεις ΣΔΕ δεύτερης τάξης, η συνάρτηση σφάλματος που πρέπει να ελαχιστοποιηθεί δίνεται από την ακόλουθη εξίσωση:

$$E[\vec{p}] = \sum_i \{\frac{d^2 y_t(x_i)}{dx^2} - f(x_i, y_t(x_i), \frac{dy_t(x_i)}{dx})\}^2. \quad (3.17)$$

3.3.2 Λύση μιας ΜΔΕ

Σε αυτή την ενότητα, συζητάμε τη λύση μεμονωμένων ΜΔΕ, εστιάζοντας σε δυο διαστάσεων προβλήματα. Η προσέγγιση μπορεί να επεκταθεί σε περισσότερες διαστάσεις. Ξεκινάμε με την εξίσωση Poisson:

$$\frac{d^2 y(x,y)}{dx^2} + \frac{d^2 y(x,y)}{dy^2} = f(x,y) \quad (3.18)$$

όπου $x \in [0,1]$, $y \in [0,1]$ και τις οριακές συνθήκες Dirichlet: $y(0,y) = f_0(y), y(1,y) = f_1(y)$, $y(x,0) = g_0(x), y(x,1) = g_1(x)$.

Η δοκιμαστική λύση για αυτό το πρόβλημα δίνεται από:

$$y_t(x,y) = A(x,y) + x(1-x)y(1-y)N(x,y,\vec{p}) \quad (3.19)$$

Εδώ, το $A(x,y)$ επιλέγεται έτσι ώστε να ικανοποιεί την οριακή συνθήκη:

$$A(x,y) = (1-x)f_0(y) + xf_1(y) + (1-y)\{g_0(x) - [(1-x)g_0(0) + xg_0(1)]\} + y\{g_1(x) - 1-xg10+xg11. \quad (3.20)$$

Για μικτές οριακές συνθήκες της μορφής: $y(0,y) = f_0(y), y(1,y) = f_1(y)$, $y(x,0) = g_0(x)$ και $(\partial y(x,1)/\partial y) = g_1(x)$ (δηλαδή, ο Dirichlet σε μέρος του ορίου και ο Neumann αλλού), η δοκιμαστική λύση γράφεται ως:

$$y_t(x,y) = B(x,y) + x(1-x)y(1-y)[N(x,y,\vec{p}) - N(x,1,\vec{p}) - \frac{dN(x,1,\vec{p})}{dx}] \quad (3.21)$$

Και εδώ το $B(x,y)$, επιλέγεται έτσι ώστε να ικανοποιεί τις οριακές συνθήκες:

$$B(x,y) = (1-x)f_0(y) + xf_1(y) + g_0(x) - [(1-x)g_0(0) + xg_0(1)] + y\{g_1(x) - 1-xg10+xg11.$$ (3.22)

Ο δεύτερος όρος της δοκιμαστικής λύσης δεν επηρεάζει τις οριακές συνθήκες καθώς εξαφανίζεται στο τμήμα του ορίου όπου εφαρμόζονται οι Dirichlet οριακές συνθήκες και η βαθμιδωτή συνιστώσα του κάθετη προς το όριο εξαφανίζεται στο τμήμα του ορίου όπου εφαρμόζονται οι Neumann οριακές συνθήκες.

Για όλα τα παραπάνω προβλήματα ΜΔΕ, το σφάλμα που πρέπει να ελαχιστοποιηθεί δίνεται από:

$$E[\vec{p}] = \sum_i \{\frac{d^2y(x_i,y_i)}{dx^2} + \frac{d^2y(x_i,y_i)}{dy^2}) - f(x_i,y_i)\}^2$$ (3.23)

όπου (x_i, y_i) τα σημεία είναι στο $[0,1] \times [0,1]$.

3.4 Παραδείγματα

Σε αυτή την παράγραφο, θα παρουσιάσουμε αποτελέσματα από την επίλυσης διαφόρων προβλημάτων μοντέλου χρησιμοποιώντας ένα πολλών φάσεων perceptron[1] με μία κρυφή φάση που έχει δέκα κρυφές μονάδες και μία γραμμική μονάδα εξόδου. Η συνάρτηση ενεργοποίησης σιγμοειδούς χρησιμοποιείται για κάθε κρυφή μονάδα, η οποία ορίζεται ως :

$$\sigma(x) = \frac{1}{(1+e^{-x})}$$

Η ακριβής αναλυτική λύση $y_a(\vec{x})$είναι γνωστή εκ των προτέρων για κάθε πρόβλημα δοκιμής, επιτρέποντάς μας να αξιολογήσουμε την ακρίβεια των λύσεων που λαμβάνονται με τον υπολογισμό της απόκλισης $\triangle\, y(\vec{x}) = y_t(\vec{x}) - y_a(\vec{x})$.

3.4.1 ΣΔΕ και Σύστημα ΣΔΕ

Πρόβλημα 1:

$$\frac{dy}{dx} + \left(x + \frac{1+3x^2}{1+x+x^3}\right)y = x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3}$$ (3.24)

[1] Το perceptron είναι ένα μοντέλο νευρωνικού δικτύου με μία είσοδο και μία έξοδο, που αναπτύχθηκε από τον Frank Rosenblatt το 1957. Το perceptron είναι ικανό να μάθει από μια σειρά από δεδομένα εισόδου-εξόδου και να εξάγει συμπεράσματα για νέες εισόδους. Το perceptron αναγνωρίζει τις εισόδους του βασιζόμενο στο βάρος που δίνεται σε κάθε είσοδο και στην αντίστοιχη έξοδο που παράγεται. Το perceptron είναι ένα από τα πιο απλά μοντέλα νευρωνικών δικτύων και χρησιμοποιείται συχνά ως βάση για πιο πολύπλοκες δομές.

με $y(0) = 1$και $x \in [0,1]$. Όπου για την επίλυση της (3.24) θα δημιουργήσουμε τον αντίστοιχο πολλαπλασιαστή Euler:

$$\mu(x) = e^{\int\left(x+\frac{1+3x^2}{1+x+x^3}\right)dx}. \quad (3.25)$$

Έτσι πολλαπλασιάζοντας την (3.24) με $\mu(x)$, παίρνουμε

$$\frac{d}{dx}\big(\mu(x)y(x)\big) = e^{\int\left(x+\frac{1+3x^2}{1+x+x^3}\right)dx}\left(x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3}\right) \quad (3.26)$$

και ολοκληρώνοντας ως προς x αμφότερα τα μέλη της (2.26), έχουμε:

$$y(x) = e^{-\int\left(x+\frac{1+3x^2}{1+x+x^3}\right)dx}\left(C + \int x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3}e^{\int\left(x+\frac{1+3x^2}{1+x+x^3}\right)dx}dx\right), \quad (3.27)$$

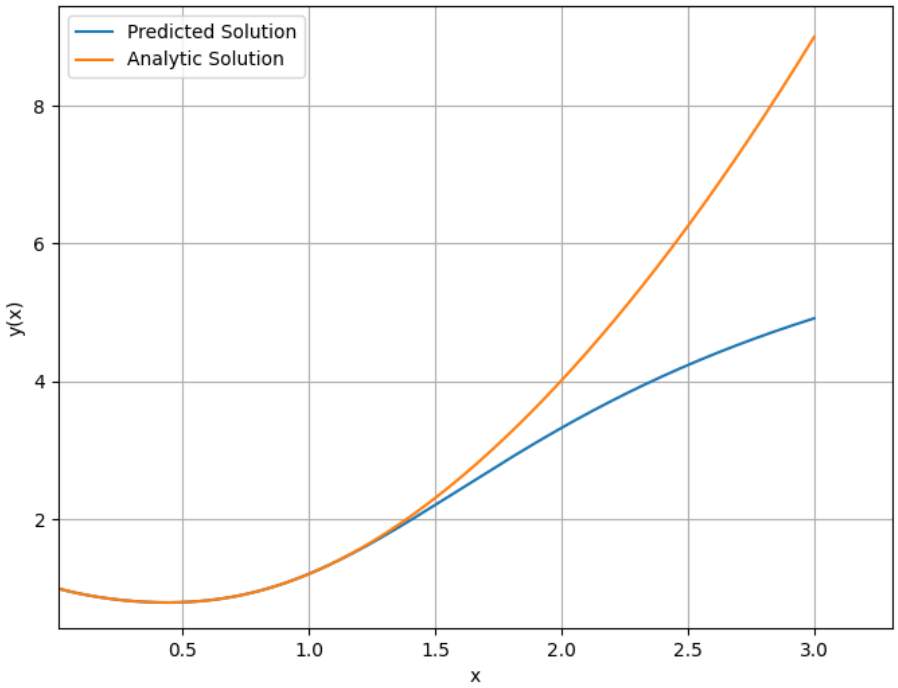
όπου C είναι μια αυθαίρετη πραγματική σταθερά. Αν εφαρμόσουμε και την αρχική συνθήκη, η μοναδική λύση που προκύπτει είναι:

$$y_a(x) = \frac{e^{-\frac{x^2}{2}}}{1+x+x^3} + x^2. \quad (3.28)$$

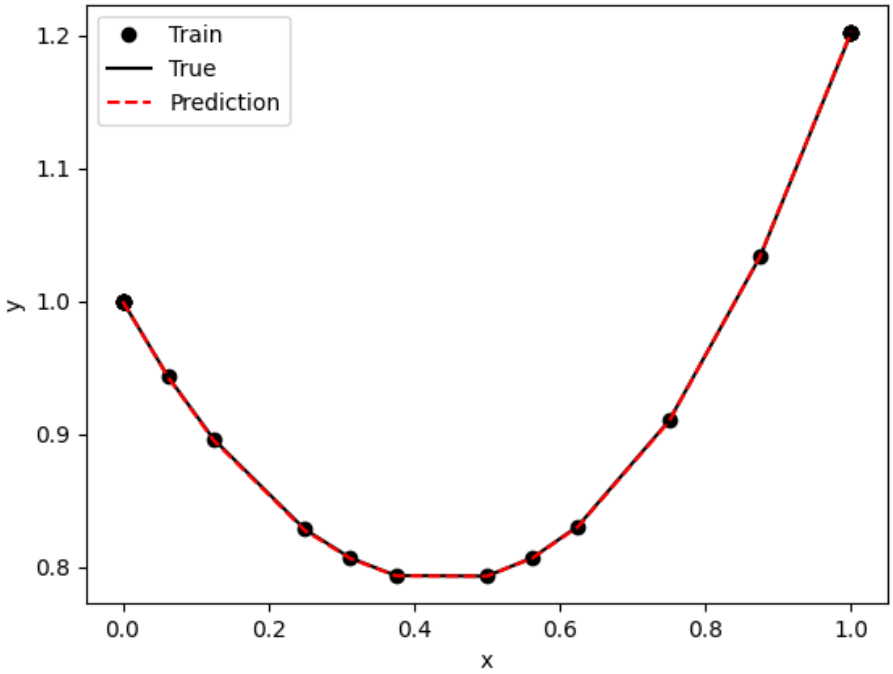
Σύμφωνα με την (3.12), η δοκιμαστική νευρωνική μορφή της λύσης είναι:

$$y_t(x) = 1 + xN(x, \vec{p}).$$

Η εκπαίδευση του δικτύου πραγματοποιήθηκε σε ένα πλέγμα δώδεκα σημείων ίσης απόστασης στο διάστημα [0, 1]. Για την υλοποίηση των γραφικών παραστάσεων χρησιμοποιήσαμε σε γλώσσα προγραμματισμού Python την βιβλιοθήκη DeepXDe. Δημιουργήσαμε ένα νευρωνικό δίκτυο μιας εισόδου, τριών ενδιάμεσων στρωμάτων και μιας εξόδου καθώς και συνάρτηση εφαπτόμενης ως συνάρτηση ενεργοποίησης. Στο σχήμα που ακολουθεί βλέπουμε τη γραφική αναπαράσταση της (3.28), και έπειτα την γραφική αναπαράσταση της ακριβής λύσης σε σχέση με την προβλεπόμενη καθώς και σημεία εκπαιδευσης.



Σχήμα 3.1: Ακριβής και προβλεπόμενη λύση της ΣΔΕ του προβλήματος 1.



Σχήμα 3.2: Ακριβής και προβλεπόμενη λύση καθώς και σημεία εκπαίδευσης της ΣΔΕ του προβλήματος 1.

Είναι σαφές πως η λύση είναι υψηλής ακρίβειας, αν και η εκπαίδευση πραγματοποιήθηκε χρησιμοποιώντας μικρό αριθμό σημείων.

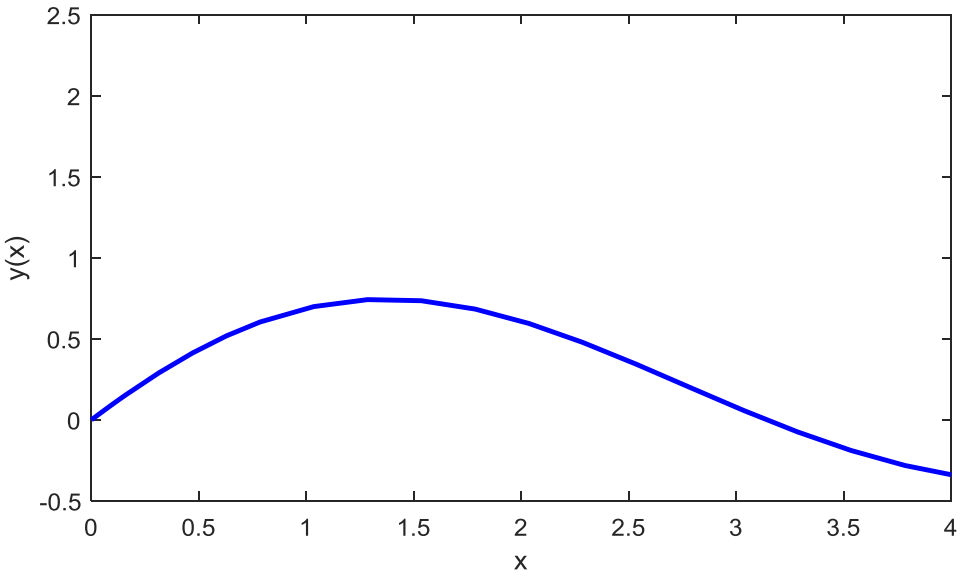
Πρόβλημα 2:

$$\frac{dy}{dx} + \frac{1}{5}y = e^{-(x/5)}\cos(x). \quad (3.29)$$

με $y(0) = 0$και $x \in [0,2]$. Η αναλυτική λύση που προκύπτει είναι

$$y_a(x) = e^{-(x/5)}\sin(\mathrm{x}),$$

Όπως φαίνεται στο σχήμα που ακολουθεί.

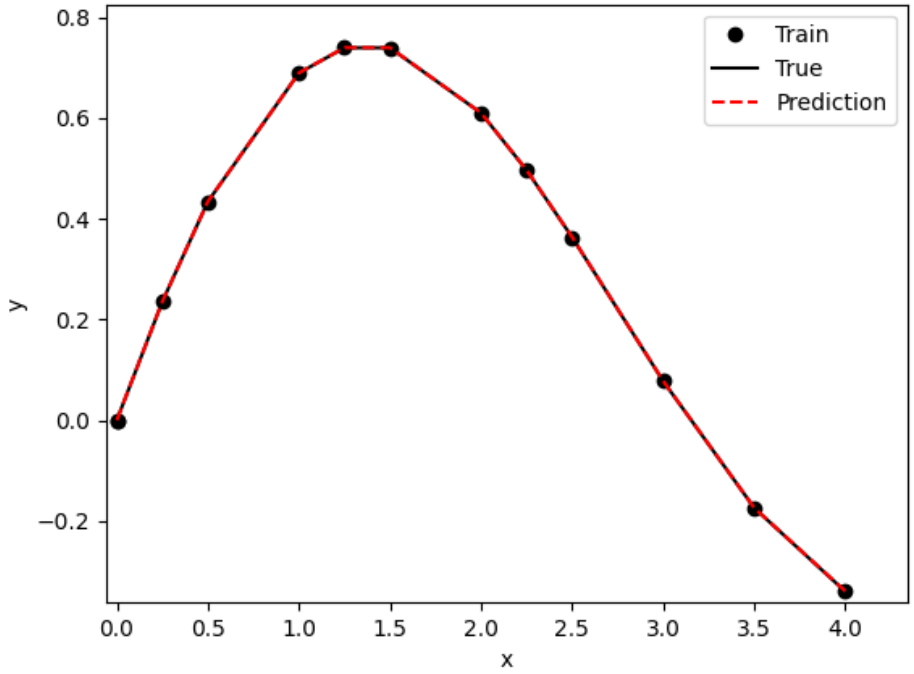


Σχήμα 3.3: Ακριβής λύση της ΣΔΕ του προβλήματος 2.

Σύμφωνα με την (3.12), η δοκιμαστική νευρωνική μορφή της λύσης είναι:

$$y_t(x) = xN(x, \vec{p}).$$

Η εκπαίδευση του δικτύου πραγματοποιήθηκε σε ένα πλέγμα δώδεκα σημείων ίσης απόστασης στο διάστημα [0, 2] και προκύπτει η παρακάτω γραφική παράσταση.



Σχήμα 3.4: Ακριβής και προβλεπόμενη λύση καθώς και σημεία εκπαίδευσης της ΣΔΕ του προβλήματος 2.

Πρόβλημα 3:

Έχουμε ένα σύστημα με δύο πρώτης τάξης ΣΔΕ

$$\frac{dy_1}{dx} = \cos(x) + {y_1}^2 + y_2 - (1 + x^2 + sin^2(x)) \quad (3.30)$$

$$\frac{\mathrm{dy_2}}{\mathrm{dx}} = 2\mathrm{x} - (1 + \mathrm{x}^2)\sin(\mathrm{x}) + \mathrm{y_1 y_2} \quad (3.31)$$

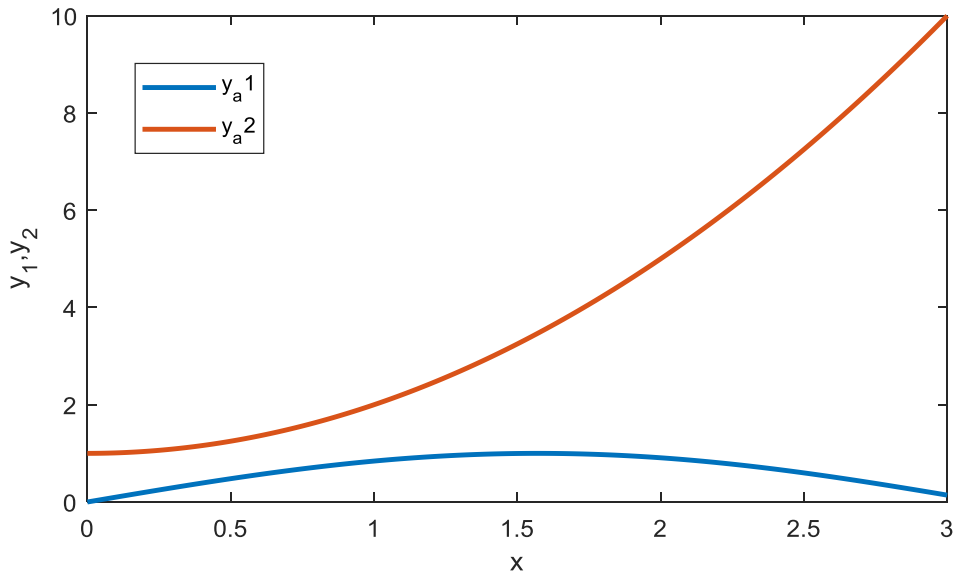
Με $y_1(0) = 0$ $y_2(0) = 1$και $x \in [0,3]$. Η αναλυτικές λύσεις που προκύπτουν είναι:

$$y_{a1}(x) = \sin(x)$$

Και

$$y_{a2}(x) = 1 + \mathrm{x}^2.$$

Όπως φαίνονται στο σχήμα που ακολουθεί.



Σχήμα 3.5: Αναλυτικές λύσεις του συστήματος ΣΔΕ του προβλήματος 3.

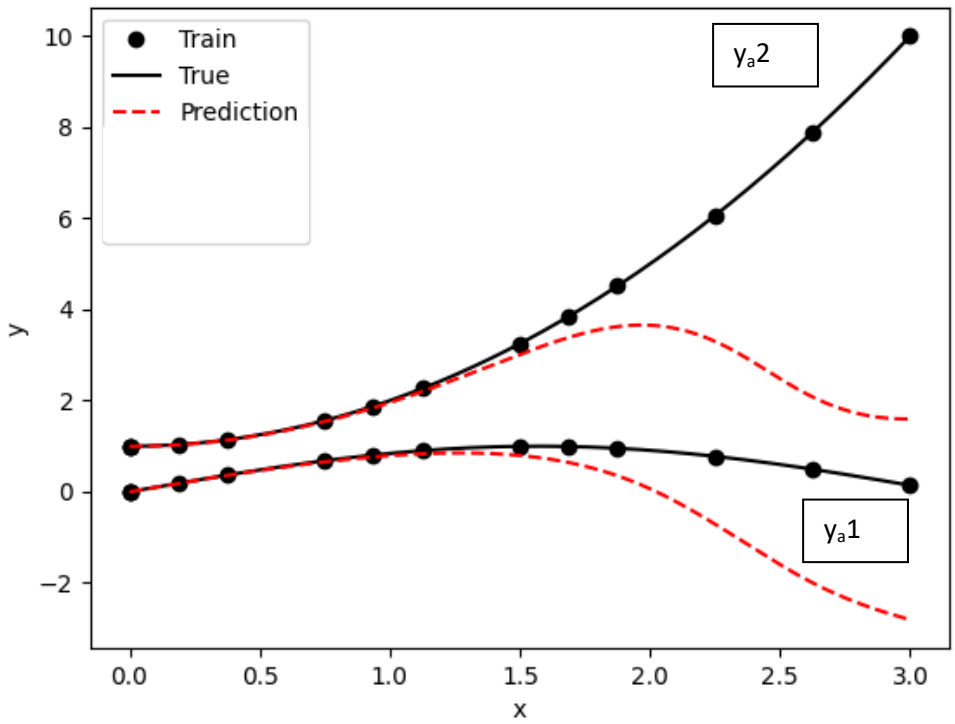
Έτσι, η νευρωνική μορφή των δοκιμαστικών της λύσεων είναι:

$$y_{t1}(x) = xN_1(x, \overrightarrow{p_1})$$

και

$$y_{t2}(x) = 1 + xN_2(x, \overrightarrow{p_2})$$

όπου τα νευρωνικά δίκτυα N_1 και N_2 έχουν την ίδια αρχιτεκτονική με τις προηγούμενες περιπτώσεις. Η εκπαίδευση του δικτύου πραγματοποιήθηκε σε ένα πλέγμα δώδεκα σημείων ίσης απόστασης στο διάστημα [0, 3] και προκύπτει η παρακάτω γραφική παράσταση.



Σχήμα 3.6: Ακριβής και προβλεπόμενη λύση καθώς και σημεία εκπαίδευσης του συστήματος ΣΔΕ του προβλήματος 3.

Βιβλιογραφία

[1]. *Catherine F. Higham, Desmond J. Higham, Deep Learning: An Introduction for Applied Mathematicians, SIAM Review (61)(2019)*

[2]. *Robert M. Freund, The Steepest Descent Algorithm for Unconstrained Optimization and a Bisection Line-search Method, Massachusetts Institute of Technology (2004)*

[3]. *Isaac Elias Lagaris, Aristidis Likas, Member IEEE and Dimitrios I. Fotiadis, Artificial Neural Networks for Solving Ordinary and Partial Differential Equations, IEEE Transactions on Neutral Networks, vol 9, No5 (1998)*

[4]. *Lu Lu, Xuhui Meng, Zhiping Mao, and George Em. Karniadakis, DeepXDE: A Deep Learning Library for Solving Differential Equations (2021)*

[5]. *Tom M.Mitchell, Machine Learning, McGraw-Hill Series in Computer Science (1997)*

[6]. *Λάουρα Θεοδόση-Κοκκίνου, Τεχνητά Νευρωνικά Δίκτυα και Εφαρμογές στα Συστήματα Αυτόματου Ελέγχου, Διπλωματική Εργασία, Πανεπιστήμιο Πατρών (2013)*

[7]. *Πάνος Αργυράκης, Νευρωνικά Δίκτυα και Εφαρμογές, Ελληνικό Ανοιχτό Πανεπιστήμιο (2001)*

[8]. *Arthur L. Samuel, Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development pp. 210–229 (1959) .*

[9]. *Κουσανάκης Εμμανουήλ, Επιτάχυνση Προσομοίωσης Δικτυού Νευρώνων με χρήση Αναδιατασσόμενης Λογικής, Διπλωματική Εργασία, Πολυτεχνείο Κρήτης (2015).*

[10]. Bernard Widrow and Marcian. E. Hoff, Adaptive Switching Circuits, 1960 WESCON Convention, Record Part 4, pp. 96-104; Human Neurobiology, 4,229(1985).

[11]. Marilyn McCord Nelson and W.T. Illingworth, A practical guide to Neural Nets, AddisonWesley (Reading, Mass), 1991.

[12].Το πρόγραμμα "Omnipage" περιγράφεται λεπτομερώς στην ηλεκτρονική διεύθυνση της εταιρίας Caere, η οποία είναι η εξής http://www.caere.com .

Παράρτημα

Στο δεύτερο κεφάλαιο παρουσιάσαμε ένα παράδειγμα άντλησης πετρελαίου, πιο αναλυτικά επιλέξαμε αρχικά πέντε σημεία όπου είχαμε επιτυχή άντληση πετρελαίου και ακόμη πέντε που έχουμε ανεπιτυχή άντληση πετρελαίου και έτσι προκύπτει το σχήμα 2.4. Έπειτα ορίσαμε τα βάρη και τις μεροληψίες για κάθε στάδιο του Τεχνητού Νευρωνικού Δικτύου που θα κατασκευάσαμε καθώς και τις Forward and Back propagate συναρτήσεις με το κατάλληλο βαθμό μάθησης και παραμέτρους. Η διαδικασία επαναλήφθηκε προσθέτοντας ένα ακόμη σημείο ανεπιτυχής άντλησης. Ο παρακάτω κώδικας είναι γραμμένος σε MATLAB.

```
%Data%
x1=[0.1,0.3,0.1,0.6,0.4,0.3,0.6,0.5,0.9,0.4,0.7];
x2=[0.1,0.4,0.5,0.9,0.2,0.7,0.3,0.6,0.2,0.4,0.6];
y=[ones(1,5) zeros(1,6); zeros(1,5) ones(1,6)];
figure(1)
a1=subplot(1,1,1);
plot(x1(1:5),x2(1:5),'ro','MarkerSize',8,'LineWidth',2)
hold on
plot(x1(6:10),x2(6:10),'bx','MarkerSize',8,'LineWidth',2)
a1.XTick = [0 1];
a1.YTick = [0 1];
a1.FontWeight = 'Bold';
a1.FontSize = 8;
xlim([0,1])
ylim([0,1])

%Weights and Biases%
rng(5000);
W2=0.5*randn(2,2); W3=0.5*randn(3,2); W4=0.5*randn(2,3);
b2=0.5*randn(2,1); b3=0.5*randn(3,1); b4=0.5*randn(2,1);

%Forward and Back propagate%
eta=0.05; %learnig rate%
Niter=1e6; %iteration%
savecost =zeros(Niter,1);  %value of cost function at each iteration

for counter=1:Niter
   k=randi(11);
   x=[x1(k);x2(k)];

   %forward pass%
   a2=activate(x,W2,b2);
   a3=activate(a2,W3,b3);
```

```
    a4=activate(a3,W4,b4);

    %backward pass%
    delta4=a4.*(1-a4).*(a4-y(:,k));
    delta3=a3.*(1-a3).*(W4'*delta4);
    delta2=a2.*(1-a2).*(W3'*delta3);

    %Gradient step%
    W2=W2-eta*delta2*x';
    W3=W3-eta*delta3*a2';
    W4=W4-eta*delta4*a3';
    b2=b2-eta*delta2;
    b3=b3-eta*delta3;
    b4=b4-eta*delta4;

    %Monitor Progress%
    newcost=cost(W2,W3,W4,b2,b3,b4);
    savecost(counter)=newcost;
end

figure(2)
semilogy([1:1e4:Niter],savecost(1:1e4:Niter),'b-','LineWidth',2)
xlabel('Iteration Number')
ylabel('Value of cost function')
set(gca,'FontWeight','Bold','FontSize',10)
N = 500;
Dx = 1/N;
Dy = 1/N;
xvals = [0:Dx:1];
yvals = [0:Dy:1];
for k1 = 1:N+1
    xk = xvals(k1);
    for k2 = 1:N+1
        yk = yvals(k2);
        xy = [xk;yk];
        a2 = activate(xy,W2,b2);
        a3 = activate(a2,W3,b3);
        a4 = activate(a3,W4,b4);
        Aval(k2,k1) = a4(1);
        Bval(k2,k1) = a4(2);
    end
```

```
end
[X,Y] = meshgrid(xvals,yvals);
figure(3)
clf
a2 = subplot(1,1,1);
Mval = Aval>Bval;
contourf(X,Y,Mval,[0.5 0.5])
hold on
colormap([1 1 1; 0.8 0.8 0.8])
plot(x1(1:5),x2(1:5),'ro','MarkerSize',8,'LineWidth',2)
plot(x1(6:11),x2(6:11),'bx','MarkerSize',8,'LineWidth',2)
a2.XTick = [0 1];
a2.YTick = [0 1];
a2.FontWeight = 'Bold';
a2.FontSize = 16;
xlim([0,1])
ylim([0,1])

  function costval=cost(W2,W3,W4,b2,b3,b4)
    costvec=zeros(10,1);
    for i=1:10
      x=[x1(i);x2(i)];
      a2=activate(x,W2,b2);
      a3=activate(a2,W3,b3);
      a4=activate(a3,W4,b4);
      costvec(i)=norm(y(:,i)-a4,2);
    end
    costval=norm(costvec,2)^2;
  end

end
```

----Σχήμα 2.3------

```
% Define the ODE
dydx = @(x,y)x.^3 + 2*x + x^2*(1 + 3*x^2)/(1 + x + x^3)-(x + (1 + 3*x^2)/(1 + x + x^3))*y ;
exact_soln = @(x)exp(-(1/2)*x^2)/(1 + x + x^3)+x^2 ;
% Set initial conditions
x0 = 0;
y0 = 1;

% Define the range of x values to solve for
xspan = [0 1];
n = 10;
x = linspace(xspan(1),xspan(2),n);
% Solve the ODE using the built-in ODE solver ode45
%[x,y] = ode45(dydx, xspan, y0);

%solve ode using the neutral network

net = feedforwardnet(10);
net.layers{1}.transferFcn = 'tansig';
net.trainFcn = 'trainlm';
net.trainParam.showWindow = false;
net = train(net,x,f(x,y0));
y = net(x);

% Compute the error
error = abs(y - exact_soln(x));
% Plot the solution
%plot(x,y,'b-', 'LineWidth',2)
%xlabel('x')
%ylabel('y(x)')
%plot the error
plot(x,error,'o',linspace(xspan(1),xspan(2),100),exact(linspace(xspan(1),xspan(2),100)) - exact(0),'--')
%legend('Error','Exact Solution')
xlabel('x')
ylabel('y')
```

----Σχήμα 2.4-----

```
% Define the ODE
dydx = @(x,y) exp(-x/5)*cos(x) - (1/5)*y;
```

```
% Set initial conditions
x0 = 0;
y0 = 0;
% Define the range of x values to solve for
xspan = [0 10];
% Solve the ODE using the built-in ODE solver ode45
[x,y] = ode45(dydx, xspan, y0);
% Plot the solution
plot(x,y,'b-', 'LineWidth',2)
xlabel('x')
ylabel('y(x)')
%title('Solution of ODE ')
```

----Σχήμα 2.5-----

```
% Define the ODE system and its exact solution
f1 = @(x, y1, y2) cos(x)+y1^2+y2-(1+x^2+(sin(x))^2);
f2 = @(x, y1, y2) 2*x -(1*x^2)*sin(x)+y1*y2;
y1_exact = @(x) sin(x);
y2_exact = @(x) 1+x.^2;
% Define the inputs for plotting the exact solution
x_exact = linspace(0, 3, 1000)';
y1_exact_vals = y1_exact(x_exact);
y2_exact_vals = y2_exact(x_exact);

% Plot the exact solutions
plot(x_exact, y1_exact_vals,'LineWidth',2);
hold on
plot(x_exact, y2_exact_vals,'LineWidth',2);
xlabel('x');
ylabel('y_1,y_2');
```

Στο τρίτο κεφάλαιο παρουσιάσαμε παραδείγματα επίλυσης ΣΔΕ με τη χρήση Νευρωνικών Δικτύων. Ο παρακάτω κώδικας είναι γραμμένος σε Python και χρησιμοποιήσαμε την βιβλιοθήκη DeepXDe.

--Σχήμα 3.1 και 3.2

```
# Importing Libraries
import deepxde as dde
import numpy as np
import matplotlib.pyplot as plt
# Defining the ODE
def ode(x, y):
  return dde.grad.jacobian(y, x) + (x + (1 + 3 * x**2)/(1 + x + x**3)) * y - x**3 - 2*x - x**2 * ((1 + 3 * x**2) / (1
+ x + x**3))
# Defining the analytical solution
def analytic_solution(x):
  return (np.exp(-x**2/2) / (1 + x + x**3))+ x**2
# Defining the boundary condition
def boundary(x, on_initial):
  return on_initial and np.isclose(x[0], 0)
# Setting up the geomentry and bountary condition
geom = dde.geometry.Interval(0, 1) # epresents a one-dimention interval from 0 to 1
bc = dde.DirichletBC(geom, lambda x: 1, boundary)
# Creating the data object. Combines the geometry, the ODE and the BC.
data = dde.data.PDE(geom, ode, bc, num_domain=10, num_boundary=10, solution=analytic_solution)
# Defining the neutral network architecture
layer_size = [1] + [50] * 3 + [1] #The network has an input layer with one neuron, three hidden layers
with 50 neurons each, and an output layer with one neuron.
activation = "tanh" # Activation function used in the hidden layer
initializer = "Glorot uniform" # The weight initialization method is Glorot uniform
net = dde.maps.FNN(layer_size, activation, initializer) # Creates a FNN
# Creating and compiling the model
model = dde.Model(data, net) # Combines the data object and neutral network architecture
model.compile("adam", lr=0.001, metrics=["l2 relative error"], loss_weights=[1])
# Training the model
losshistory, train_state = model.train(epochs=10000)
# Evaluate the solution at test points
x_test = np.linspace(0, 3, 500).reshape(-1, 1) #less test points
y_test = model.predict(x_test)
# Plot and save the results
dde.saveplot(losshistory, train_state, issave=True, isplot=True)
```

```
# Plotting the predicted and analytic solution
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_test, label='Predicted Solution')
plt.plot(x_test, analytic_solution(x_test), label='Analytic Solution')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.grid(True)
plt.show()
```

--Σχήμα 3.3 και 3.4

```
  # Importing Libraries
import deepxde as dde
import matplotlib.pyplot as plt
import paddle
import numpy as np
# Defining the ODE
def ode(x, y):
  xt = x.detach().numpy()
  return dde.grad.jacobian(y, x) + y/5 - paddle.exp(-x/5) * paddle.cos(x)
# Defining the analytical solution
def analytic_solution(x):
  return np.exp(-x/5) * np.sin(x)
# Defining the boundary condition
def boundary(x, on_initial):
  return on_initial and np.isclose(x[0], 0)
# Setting up the geometry and boundary condition
geom = dde.geometry.Interval(0, 4)  # Represents a one-dimensional interval from 0 to 2
bc = dde.DirichletBC(geom, lambda x: 0, boundary)
# Creating the data object. Combines the geometry, the ODE, and the BC.
data = dde.data.PDE(geom, ode, bc, num_domain=10, num_boundary=10, solution=analytic_solution)
# Defining the neutral network architecture
layer_size = [1] + [50] * 3 + [1]  # The network has an input layer with one neuron, three hidden layers
with 50 neurons each, and an output layer with one neuron.
activation = "tanh"  # Activation function used in the hidden layer
initializer = "Glorot uniform"  # The weight initialization method is Glorot uniform
net = dde.maps.FNN(layer_size, activation, initializer)  # Creates a FNN
# Creating and compiling the model
model = dde.Model(data, net)  # Combines the data object and neutral network architecture
```

```
model.compile("adam", lr=0.001, metrics=["l2 relative error"], loss_weights=[1])
# Training the model
losshistory, train_state = model.train(epochs=10000)
# Evaluate the solution at test points
x_test = np.linspace(0, 3, 500).reshape(-1, 1)  # Fewer test points (you can increase this if needed)
y_test = model.predict(x_test)  # Convert NumPy array to Paddle Tensor

# Plot and save the results
dde.saveplot(losshistory, train_state, issave=True, isplot=True)
# Plotting the predicted and analytic solution
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_test, label='Predicted Solution')
plt.plot(x_test, analytic_solution(x_test), label='Analytic Solution')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.grid(True)
plt.show()
```

--Σχήμα 3.5 και 3.6

```
# Importing Libraries
import deepxde as dde
import matplotlib.pyplot as plt
import numpy as np
import paddle

# Defining the analytical solution
def func(x):
  return np.hstack((np.sin(x), 1+x**2))

# Defining the ODE system
def ode_system(x, y):
  y1, y2 = y[:, 0:1], y[:, 1:2]
  dy1_dx = dde.grad.jacobian(y, x, i=0)
  dy2_dx = dde.grad.jacobian(y, x, i=1)
  return [dy1_dx - paddle.cos(x) - y1**2 - y2 + (1 + x**2 + paddle.sin(x)**2),
      dy2_dx - 2*x + (1 + x**2) * paddle.sin(x) - y1*y2]

# Defining the boundary condition
def boundary(_, on_initial):
  return on_initial
```

```
# Setting up the geometry and boundary condition
geom = dde.geometry.TimeDomain(0, 3)
ic1 = dde.icbc.IC(geom, lambda x: 0, boundary, component=0)
ic2 = dde.icbc.IC(geom, lambda x: 1, boundary, component=1)
# Creating the data object. Combines the geometry, the ODE, and the BC.
data = dde.data.PDE(
    geom,
    ode_system,
    [ic1, ic2],
    10, 2,
    solution=func,
    num_test=500
)

# Defining the neutral network architecture
layer_size = [1] + [50] * 3 + [2]
activation = "tanh"
initializer = "Glorot uniform"
net = dde.maps.FNN(layer_size, activation, initializer)
model = dde.Model(data, net)
model.compile("adam", lr=0.0001, metrics=["l2 relative error"])#, loss_weights=[1, 1])
# Training the model
losshistory, train_state = model.train(iterations=18000)
# Evaluate the solution at test points
x_test = np.linspace(0, 3, 500).reshape(-1, 1) #less test points
y_test = model.predict(x_test)
# Plot and save the results
dde.saveplot(losshistory, train_state, issave=True, isplot=True)
# Plotting the predicted and analytic solution
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_test, label='Predicted Solution')
plt.plot(x_test, func(x_test), label='Analytic Solution')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.legend()
plt.grid(True)
plt.show()
```