Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Εφαρμοσμένων Μαθηματικών
και Φυσικών Επιστημών
Τομέας Μαθηματικών

Αλγόριθμοι Επιβλεπόμενης Μάθησης για Ανάλυση Μεγάλου Όγκου Δεδομένων από Φορητές Συσκευές σε Κατανεμημένα Περιβάλλοντα Επεξεργασίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΓΓΕΛΟΣ ΓΕΩΡΓΟΥΛΗΣ

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Εφαρμοσμένων Μαθηματικών
και Φυσικών Επιστημών
Τομέας Μαθηματικών

Αλγόριθμοι Επιβλεπόμενης Μάθησης για Ανάλυση Μεγάλου Όγκου Δεδομένων από Φορητές Συσκευές σε Κατανεμημένα Περιβάλλοντα Επεξεργασίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΓΓΕΛΟΣ ΓΕΩΡΓΟΥΛΗΣ

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Ιουλίου 2020.

Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Γεώργιος Στάμου Αναπληρωτής Καθηγητής Ε.Μ.Π.	Πέτρος Στεφανέας Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

Άγγελος Γεωργούλης

Διπλωματούχος Σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η κατηγοριοποίηση σημάτων που βρίσκονται στη μορφή συντεταγμένων χώρου και προέρχονται από φορητές συσκευές. Η δυσκολία έγκειται στο γεγονός πως μια τέτοια διεργασία δεν είναι εφικτή σε ένα σύνηθες προσωπικό υπολογιστικό σύστημα με συμβατικές τεχνικές όταν τα δεδομένα εμπίπτουν στην κατηγορία των Μεγάλων Δεδομένων. Για το λόγο αυτό χρησιμοποιούμε το Apache Spark σε ένα κατανεμημένο υπολογιστικό περιβάλλον. Το κάθε σήμα κατηγοριοποιείται σε μία από έξι κλάσεις ανάλογα με τη φυσική δραστηριότητα στην οποία είχε υποβληθεί το άτομο από το οποίο λαμβάνονταν κάθε φορά οι μετρήσεις.

Χρησιμοποιούνται γνωστοί αλγόριθμοι επιβλεπόμενης μηχανικής μάθησης, όπως τα τυχαία δάση και η λογιστική παλινδρόμηση, για τη δημιουργία μοντέλων πρόβλεψης. Το κάθε μοντέλο εκπαιδεύεται σε ένα σύνολο δεδομένων εκπαίδευσης και παράλληλα βελτιστοποιείται, χρησιμοποιώντας ειδικές τεχνικές ώστε, δεδομένων των παρεχόμενων υπερπαραμέτρων, να εφαρμοστεί σε ένα σύνολο δεδομένων ελέγχου και να αποφέρει όσο το δυνατόν χαμηλότερο σφάλμα αναγνώρισης της εκάστοτε δραστηριότητας.

Λέξεις κλειδιά

Μηχανική Μάθηση, Κατανεμημένο Περιβάλλον, Μεγάλα Δεδομένα, Apache Spark.

Abstract

The purpose of this diploma dissertation is the classification of signals appearing in the form of 3-dimensional coordinates derived from mobile devices. The main difficulty in processing them stems from the fact that such a process is not easily feasible on commonplace personal computing system, using ordinary techniques, when the data are characterized as Big Data. For that reason, we use Apache Spark on a distributed (clustered) environment. Each signal is classified into one of six classes corresponding to a physical activity.

Supervised Machine Learning Algorithms are used, such as random forests and logistic regression, in order to create prediction models. Each model is trained using a training set and in parallel is being optimized using special techniques concerning the respective hyperparameters used, so that it returns the lowest possible recognition error for each activity being tested.

Key words

Machine Learning, Distributed Environment, Big Data, Apache Spark.

Ευχαριστίες

Ευχαριστώ τον Επιβλέποντα Καθηγητή αυτής της διατριβής, κ. Ανδρέα-Γεώργιο Σταφυλοπάτη, για την καθοδήγηση και την εμπιστοσύνη του.

Ευχαριστώ επίσης τον Αναπληρωτή Καθηγητή κ. Στάμου Γεώργιο και τον Επίκουρο Καθηγητή κ. Στεφανέα Πέτρο που μου έκαναν την τιμή να είναι μέλη της τριμελούς επιτροπής εξέτασης της παρούσας εργασίας.

Θέλω ακόμα να ευχαριστήσω θερμά τα μέλη Εργαστηριακού και Διδακτικού Προσωπικού κ. κ. Γεώργιο Αλεξανδρίδη και Γεώργιο Σιόλα για την πρόθυμη και πάντα αποτελεσματική βοήθειά τους, για τη συνεχή καθοδήγηση και τις χρήσιμες συμβουλές.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου που με υποστήριξε τόσο σην εκπόνηση της διπλωματικής μου, όσο και συνολικά στις σπουδές μου.

Άγγελος Γεωργούλης,

Αθήνα, 15η Ιουλίου 2020

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Listings

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός της εργασίας

Οι φορητές συσκευές έχουν ωριμάσει και χαρακτηρίζονται ως μια οικονομική και συνάμα αξιόπιστη πλατφόρμα συλλογής δεδομένων σε διαδεδομένα και πανταχού παρόντα συστήματα αίσθησης. Συγκεκριμένα πωλούνται μαζικά στην αγορά, είναι συνδεδεμένα με καθημερινές ανθρώπινες δραστηριότητες και υποστηρίζονται από ενσωματωμένες μονάδες επικοινωνίας και αίσθησης. Σύμφωνα με πρόσφατη μελέτη της Cisco Systems [Cisc16], μισό δις φορητών συσκευών πουλήθηκαν παγκοσμίως το 2015 και η κυκλοφορία κινητών δεδομένων αυξήθηκε κατά 74% παράγοντας 3,7 exabytes δεδομένων ανά μήνα. Τα *μεγάλα δεδομένα φορητών συσκευών* (Mobile Big Data - MBD) είναι μια ιδέα που περιγράφει ένα τεράστιο ποσό φορητών δεδομένων, τα οποία δε μπορούν να επεξεργαστούν από μία μόνο μηχανή. Τα MBD περιλαμβάνουν χρήσιμες πληροφορίες για την επίλυση πολλών προβλημάτων όπως ο *εντοπισμός απάτης* (fraud detection), η εστιασμένη διαφήμιση και το μάρκετινγκ, η υπολογιστική *επίγνωσης περιβάλλοντος* (context-aware) και θέματα του τομέα υγείας. Έτσι, η ανάλυση MBD αποτελεί ένα θέμα υψηλής ενασχόλησης και εστιάζει στην εξαγωγή χρήσιμης πληροφορίας και μοτίβων από ακατέργαστα φορητά δεδομένα.

Η *Μηχανική* και η *Βαθιά Μάθηση* αποτελούν ένα ισχυρό εργαλείο στην ανάλυση MBD. Όπως προκύπτει, όμως, από την κατάρα της διαστατικότητας και του μεγέθους των MBD, τα παραγόμενα μοντέλα είναι αργά και χρειάζονται από μερικές ώρες μέχρι και μέρες για να εκτελεστούν σε συμβατικά υπολογιστικά συστήματα. Αντίθετα, τα περισσότερα φορητά συστήματα έχουν καθυστερήσεις και οι αποφάσεις πρέπει να λαμβάνονται όσο το δυνατόν ταχύτερα ώστε να είναι υψηλή η ικανοποίηση του χρήστη.

Ο στόχος, λοιπόν, είναι η αντιμετώπιση της ολοένα και υψηλότερης ζήτησης κλιμακούμενων και προσαρμοστικών φορητών συστημάτων που επιτρέπει την ανάλυση MBD σε αποδοτικό χρόνο χρησιμοποιώντας μοντέλα τόσο μηχανικής όσο και βαθιάς Μάθησης σε ένα περιβάλλον κατανεμημένων υπολογιστικών συστημάτων χρησιμοποιώντας λογισμικό ανοιχτού κώδικα και συγκεκριμένα το Apache Spark [Apac20].

1.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της *Αναγνώρισης Προτύπων* και της *Υπολογιστικής Θεωρίας Μάθησης* στην *Τεχνητή Νοημοσύνη* (TN) [BRIT]. Το 1959, ο Άρθουρ Σάμουελ όρισε τη μηχανική μάθηση ως *«πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί»* [Simo13]. Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα [RonK98] και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Η μηχανική μάθηση είναι στενά συνδεδεμένη και συχνά συγχέεται με την *Υπολογιστική Στατιστική*, έναν κλάδο που επίσης επικεντρώνεται στην πρόβλεψη μέσω της χρήσης των υπολογιστών. Έχει ισχυρούς δεσμούς με τη *Μαθηματική Βελτιστοποίηση*, η οποία παρέχει μεθόδους, τη θεωρία και

τομείς εφαρμογής. Η μηχανική μάθηση εφαρμόζεται σε μια σειρά από υπολογιστικές εργασίες, όπου τόσο ο σχεδιασμός όσο και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Παραδείγματα εφαρμογών αποτελούν τα *φίλτρα ανεπιθύμητης ηλεκτρονικής αλληλογραφίας* (spam filters), η *οπτική αναγνώριση χαρακτήρων* (optical character recognition - OCR) [Wern], οι μηχανές αναζήτησης και η *υπολογιστική όραση*. Ο τομέας της *Εξόρυξης Δεδομένων* [Mann96] είναι ένας σχετικός κλάδος, ο οποίος εστιάζει στην *Εξερευνητική Ανάλυση Δεδομένων* (Exploratory Data Analysis) χρησιμοποιώντας *Μη-Επιβλεπόμενη Μάθηση* (Unsupervised Learning). Στην εφαρμογή της σε πρακτικά επιχειρηματικά προβλήματα, η μηχανική μάθηση αναφέρεται και ως *Αναλυτική Πρόβλεψης* (Predictive Analytics).

Στο πεδίο της ανάλυσης δεδομένων, η μηχανική μάθηση είναι μια μέθοδος που χρησιμοποιείται για την κατασκευή πολύπλοκων μοντέλων και αλγορίθμων που οδηγούν στην πρόβλεψη. Τα αναλυτικά μοντέλα επιτρέπουν στους ερευνητές, τους επιστήμονες δεδομένων, τους μηχανικούς και τους αναλυτές να παράγουν αξιόπιστες αποφάσεις και αποτελέσματα και να αναδείξουν αλληλοσυσχετίσεις μέσω της μάθησης από ιστορικές σχέσεις και τάσεις στα δεδομένα.

Ο Tom M. Mitchell πρότεινε το 1997 έναν πιο επίσημο ορισμό που χρησιμοποιείται ευρέως: *«Ένα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοσή του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E»* [Mitc97]. Αυτός ο ορισμός είναι σημαντικός για τον καθορισμό της μηχανικής μάθησης στη βάση ενός λειτουργικού πλαισίου και όχι με γνωστικούς όρους, ακολουθώντας έτσι την πρόταση του Alan Turing στην εργασία του «Υπολογιστικές μηχανές και Νοημοσύνη» [Turi50], ότι το ερώτημα αν μπορούν οι μηχανές να σκεφτούν, μπορεί να αντικατασταθεί με το ερώτημα αν μπορούν οι μηχανές να κάνουν αυτό που εμείς (ως σκεπτόμενες οντότητες) μπορούμε να κάνουμε.

Ως κλάδος της ΤΝ, η μηχανική μάθηση [Bish06] ασχολείται με τη μελέτη αλγορίθμων που βελτιώνουν τη συμπεριφορά τους σε κάποια εργασία που τους έχει ανατεθεί χρησιμοποιώντας την εμπειρία τους. Όσον αφορά τη σχεδίαση των συστημάτων μηχανικής μάθησης, για τα συστήματα που ανήκουν στη συμβολική ΤΝ, η δυνατότητα μάθησης προσδιορίζεται ως η ικανότητα απόκτησης επιπλέον γνώσης, που επιφέρει μεταβολές στην υπάρχουσα καταχωρημένη γνώση είτε αλλάζοντας τα χαρακτηριστικά της είτε αυξομειώνοντάς την. Στην περίπτωση των συστημάτων ΤΝ που ανήκουν στη μη-συμβολική ΤΝ (όπως η περίπτωση των τεχνητών νευρωνικών δικτύων), ως μάθηση προσδιορίζεται η δυνατότητα που διαθέτουν τα συστήματα στο να μετασχηματίζουν την εσωτερική τους δομή, παρά στο να μεταβάλλουν κατάλληλα τη γνώση που έχει καταχωρηθεί μέσα σε αυτά κατά το σχεδιασμό τους.

Αν και απέχουμε πάρα πολύ από τη δημιουργία μηχανών που μαθαίνουν τόσο καλά όσο ο άνθρωπος, για συγκεκριμένες περιοχές μάθησης έχουν αναπτυχθεί αλγόριθμοι οι οποίοι έχουν επιτρέψει την εμφάνιση σύγχρονων εμπορικών εφαρμογών με σημαντική επιτυχία. Επιπλέον, τα αποτελέσματα από τις εφαρμογές της ΤΝ αρχίζουν ήδη να είναι ορατά και να δίνουν απαντήσεις σε αναπάντητα, έως τώρα, ερωτήματα των άλλων κλάδων που διερευνούν την ικανότητα του ανθρώπου να μαθαίνει.

Ο τομέας της μηχανικής μάθησης αναπτύσσει, επίσης, επιτυχώς την *εξελικτική μάθηση* (evolutionary learning), η οποία μιμείται διαδικασίες φυσικής αναπαραγωγικής σε έμβια όντα. Χρησιμοποιείται κυρίως σε προβλήματα βελτιστοποίησης και εκεί κυριαρχούν οι *γενετικοί αλγόριθμοι* (genetic algorithms).

1.2.1 Είδη Μηχανικής Μάθησης

Εν γένει, ο τομέας της Μηχανικής Μάθησης αναπτύσσει τρεις κύριους τρόπους μάθησης [Duda00], ανάλογους με τους τρόπους με τους οποίους μαθαίνει ο άνθρωπος:

1. **Επιβλεπόμενη Μάθηση (Supervised Learning)** είναι η διαδικασία της εκμάθησης μιας συνάρτησης που αντιστοιχίζει μία είσοδο σε μία έξοδο, βασισμένη σε παραδείγματα ζευγαριών εισόδων-εξόδων. Δημιουργεί μια συνάρτηση από *δεδομένα εκπαίδευσης με ετικέτα* (labeled training data) αποτελούμενα από σύνολα παραδειγμάτων εκπαίδευσης [Russ10]. Κάθε παράδειγμα είναι ένα ζεύγος αποτελούμενο από το αντικείμενο εισόδου (συνήθως ένα διάνυσμα)

και μια επιθυμητή τιμή εξόδου. Ένας αλγόριθμος επιβλεπόμενης μάθησης αναλύει τα δεδομένα εκπαίδευσης και παράγει μία συνάρτηση που μπορεί να χρησιμοποιηθεί για την αντιστοίχιση νέων παραδειγμάτων. Ένα βέλτιστο σενάριο θα επέτρεπε στον αλγόριθμο να προβλέπει τις ετικέτες των κλάσεων για άγνωστες, μέχρι εκείνη τη στιγμή, εισόδους. Αυτό απαιτεί ο αλγόριθμος εκμάθησης να μπορεί να γενικεύσει, από τα δεδομένα εκπαίδευσης, σε μη παρατηρηθείσες καταστάσεις κατά έναν «λογικό» τρόπο.

2. **Μη-Επιβλεπόμενη Μάθηση (Unsupervised Learning)** είναι η περιοχή της μηχανικής μάθησης που αναζητά μοτίβα σε ένα σύνολο δεδομένων χωρίς την ύπαρξη ετικετών και με ελάχιστη ανθρώπινη επίβλεψη. Στην πράξη, μέσω της μη-επιβλεπόμενης μάθησης, μαθαίνονται τα στατιστικά χαρακτηριστικά των δεδομένων εισόδου [Tuck04].

3. **Αυτοενισχυόμενη Μάθηση (Reinforcement Learning)** είναι η περιοχή της μηχανικής μάθησης όπου η γνώση μοντελοποιείται υπό τη μορφή πρακτόρων που εξερευνούν το περιβάλλον τους με στόχο τη μεγιστοποίηση της ανταμοιβής τους. Διαφέρει από την επιβλεπόμενη μάθηση στο γεγονός πως δε χρειάζεται ζεύγη εισόδων-εξόδων με ετικέτ. Αντίθετα, επικεντρώνεται στην αναζήτηση μιας ισορροπίας ανάμεσα στην αναζήτηση της αχαρτογράφητης περιοχής και εκμετάλλευση της τρέχουσας γνώσης.

Μεταξύ της επιβλεπόμενης και της μη-επιβλεπόμενης μάθησης είναι η ημι-επιβλεπόμενη μάθηση, όπου το σύνολο δεδομένων προς εκπαίδευσης είναι ελλειπές, μιας και κάποιες (συχνά πολλές) από τις ετικέτες των δεδομένων απουσιάζουν. Η μεταγωγή είναι μια ειδική περίπτωση αυτής της αρχής, όπου το σύνολο των καταστάσεων του προβλήματος είναι γνωστό κατά το χρόνο εκμάθησης, όμως ένα μέρος των στόχων-ετικετών λείπουν.

Μεταξύ άλλων κατηγοριών μηχανικής μάθησης, υπάρχει ακόμα η διαδικασία εκμάθησης (meta learning) που μαθαίνει στην μηχανή (να αναπτύσσει) τις δικές της επαγωγικές μεθόδους, βασιζόμενη σε προηγούμενη εμπειρία. Η *αναπτυξιακή ρομποτική μάθηση* (developmental robotics), η οποία έχει αναπτυχθεί για την εκμάθηση από ρομπότ, δημιουργεί τη δική της ακολουθία μαθησιακών καταστάσεων, ώστε το ρομπότ να αποκτά συσσωρευτικά μια ποικιλία δεξιοτήτων μέσω της αυτόνομης εξερεύνησης και της κοινωνικής αλληλεπίδρασης με ανθρώπους εκπαιδευτές και χρησιμοποιώντας μηχανισμούς καθοδήγησης, όπως η ενεργητική μάθηση, η ωρίμανση και η μίμηση.

1.3 Βαθιά Μάθηση

Η *βαθιά μάθηση* (deep learning) ανήκει [Ripl96] σε μια ευρύτερη οικογένεια μεθόδων μηχανικής μάθησης που βασίζεται στα τεχνητά νευρωνικά δίκτυα με εκμάθηση χαρακτηριστικών. Και σε αυτή την περίπτωση η μάθηση μπορεί να διακριθεί σε επιβλεπόμενη, ημι-επιβλεπόμενη ή μη-επιβλεπόμενη [Schm].

Τα περισσότερα μοντέλα βαθιάς μάθησης βασίζονται στα τεχνητά νευρωνικά δίκτυα και συγκεκριμένα στα *συνελικτικά νευρωνικά δίκτυα* (convolutional neural networks - CNN), παρόλο που μπορούν να συμπεριλάβουν εξισώσεις προτασιακής λογικής ή λανθάνουσες μεταβλητές οργανωμένες ανά επίπεδα σε *βαθιά παραγωγικά μοντέλα* (deep generative models), όπως οι κόμβοι σε *βαθιά δίκτυα εμπιστοσύνης* (deep belief networks) και *βαθιές μηχανές Boltzmann* (deep Boltzmann machines).

Στη βαθιά μάθηση [LeCu15], κάθε επίπεδο μαθαίνει να μετασχηματίζει τα δεδομένα εισόδου του σε μια πιο αφηρημένη και σύνθετη αναπαράσταση. Σε μια εφαρμογή αναγνώρισης εικόνας, η *ακατέργαστη είσοδος* (raw input) μπορεί να είναι ένας πίνακας από πίξελ. Το πρώτο επίπεδο αναπαράστασης μπορεί να αφαιρέσει τα πίξελ και να κωδικοποιήσει γωνίες, το δεύτερο μπορεί να συνθέσει και να κωδικοποιήσει συστοιχίες γωνιών, το τρίτο μπορεί να κωδικοποιήσει μία μύτη και μάτια και το τέταρτο επίπεδο μπορεί να αναγνωρίσει πως η εικόνα περιέχει ένα πρόσωπο. Σημαντικό είναι πως μια διαδικασία βαθιάς μάθησης μπορεί να μάθει από μόνη της πού να τοποθετήσει και σε ποιο επίπεδο χαρακτηριστικά και αυτό κατά έναν βέλτιστο τρόπο. Αυτό βέβαια δε σημαίνει πως εξαλείφεται η ανάγκη για *επαναρύθμιση* (hard tuning) κάτι που αποδεικνύεται από το γεγονός πως διαφορετικοί αριθμοί επιπέδων και μεγέθη επιπέδων μπορούν να προσφέρουν διαφορετικούς βαθμούς αφαιρετικότητας.

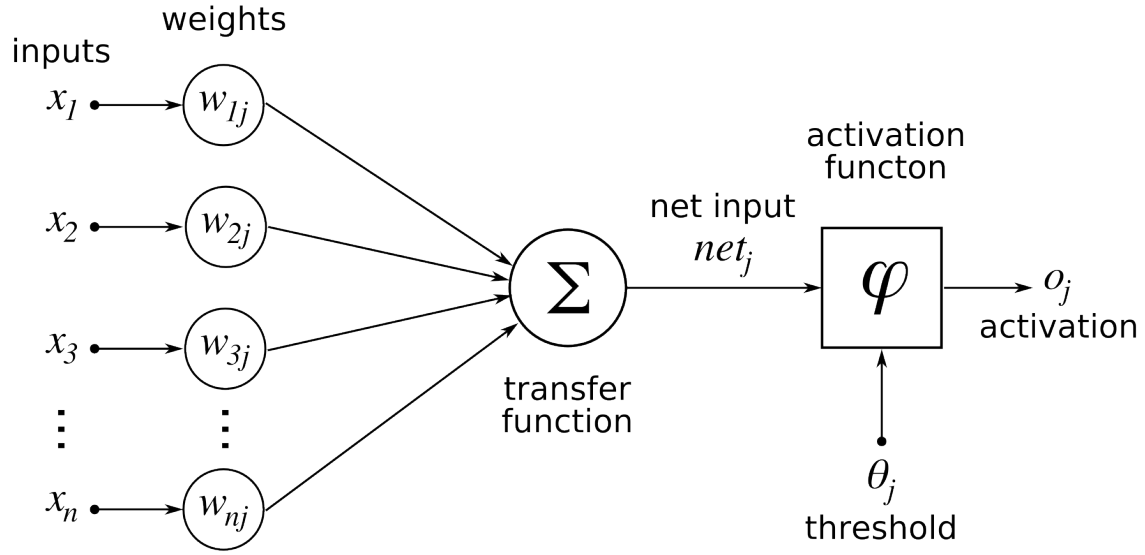
Η λέξη «βαθιά» στη «βαθιά μάθηση» αναφέρεται [Ripl96] στον αριθμό των επιπέδων δια μέσω των οποίων μετασχηματίζονται τα δεδομένα. Πιο συγκεκριμένα, τα συστήματα βαθιάς μάθησης έχουν ένα ουσιώδες βάθος που καθορίζεται από *μονοπάτι ανάθεσης πίστωσης* (credit assignment path - CAP). Το CAP είναι η αλυσίδα μετασχηματισμών από την είσοδο στην έξοδο. Τα CAP περιγράφουν πιθανές αιτιώδεις συνδέσεις μεταξύ εισόδου και εξόδου. Για ένα δίκτυο *πρόσθιας τροφοδότησης* (feedforward neural network), το βάθος των CAP είναι το βάθος του δικτύου και ο αριθμός των κρυφών επιπέδων συν ένα (το επίπεδο εξόδου είναι και εκείνο παραμετροποιήσιμο). Για *αναδρομικά νευρωνικά δίκτυα* (recurrent neural networks), στα οποία ένα σήμα μπορεί να περάσει μέσω ενός επιπέδου περισσότερες από μία φορές, το βάθος του CAP είναι πιθανότατα απεριόριστο. Παρότι δεν υπάρχει κάποιος ορισμός, κατά γενική ομολογία η βαθιά μάθησης σχετίζεται με βάθος CAP μεγαλύτερο από 2, κάτι που τη διαφοροποιεί από τη *ρηχή μάθηση* (shallow learning). Το βάθος CAP ίσο με 2 έχει αποδειχθεί πως είναι ένας καθολικός εκτιμητής με την έννοια ότι μπορεί να προσομοιώσει οποιαδήποτε συνάρτηση. Πέραν αυτού, περισσότερα επίπεδα δεν προσθέτουν κάτι παραπάνω στη δυνατότητα εκτίμησης της εξίσωσης του δικτύου. Τα βαθιά μοντέλα (CAP > 2) είναι ικανά να εξάγουν καλύτερα χαρακτηριστικά από τα ρηχά, κάτι που επιβεβαιώνει πως παραπάνω επίπεδα βοηθούν στην πιο αποτελεσματική εκμάθηση των χαρακτηριστικών.

Στην περίπτωση της επιβλεπόμενης μάθησης, οι μέθοδοι βαθιάς μάθησης μπορούν να εξαλείψουν την ανάγκη για *μηχανική χαρακτηριστικών* (feature engineering), μεταφράζοντας τα δεδομένα σε συμπαγείς ενδιάμεσες αναπαραστάσεις όμοιες *με κύριους άξονες* (principal components), και αντλούν δομημένα επίπεδα που αφαιρούν τον πλεονασμό από την αναπαράσταση.

Οι αλγόριθμοι βαθιάς μάθησης μπορούν να εφαρμοστούν και σε εργασίες μάθησης χωρίς επίβλεψη. Αυτό αποτελεί ένα σημαντικό όφελος, καθώς τα δεδομένα χωρίς ετικέτες είναι άφθονα σε σχέση με τα δεδομένα με ετικέτες. Παραδείγματα βαθιών αρχιτεκτονικών που μπορούν να εκπαιδευτούν με τέτοιο τρόπο αποτελούν τα βαθιά δίκτυα εμπιστοσύνης.

1.3.1 Τεχνητά Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα [Hayk99] είναι δίκτυα από απλούς υπολογιστικούς κόμβους (νευρώνες), διασυνδεδεμένους μεταξύ τους. Είναι εμπνευσμένα από το κεντρικό νευρικό σύστημα του ανθρώπου και των ζώων, το οποίο και προσπαθεί να προσομοιώσει.



Σχήμα 1.1: Νευρώνας ενός τεχνητού νευρωνικού δικτύου (Πηγή: `https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png`)

Οι νευρώνες αποτελούν τα δομικά στοιχεία του δικτύου (Σχήμα 1.1) [Δι07]. Κάθε τέτοιος κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε

από το περιβάλλον), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου. Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες. Οι νευρώνες εισόδου δεν επιτελούν κανέναν υπολογισμό, μεσολαβούν απλώς ανάμεσα στις περιβαλλοντικές εισόδους του δικτύου και στους υπολογιστικούς νευρώνες. Οι νευρώνες εξόδου διοχετεύουν στο περιβάλλον τις τελικές αριθμητικές εξόδους του δικτύου. Οι υπολογιστικοί νευρώνες πολλαπλασιάζουν κάθε είσοδό τους με το αντίστοιχο συναπτικό βάρος και υπολογίζουν το ολικό άθροισμα των γινομένων. Το άθροισμα αυτό τροφοδοτείται ως όρισμα στη συνάρτηση ενεργοποίησης, την οποία υλοποιεί εσωτερικά κάθε κόμβος. Η τιμή που λαμβάνει η συνάρτηση για το εν λόγω όρισμα είναι και η έξοδος του νευρώνα για τις τρέχουσες εισόδους και βάρη.

Εάν x_{ki} είναι η i-οστή είσοδος του k-οστού νευρώνα, w_{ki} το i-οστό συναπτικό βάρος του k-οστού νευρώνα και $\phi(\cdot)$ η συνάρτηση ενεργοποίησης του νευρώνα, τότε η έξοδος o_j του δίνεται από την Εξίσωση 1.1

$$o_j = \phi\left(\sum_{i=0}^{N} x_{ki} w_{ki} + \theta_j\right) \tag{1.1}$$

Η παράμετρος θ_j είναι γνωστή και ως *πόλωση* (bias) ή *κατώφλι* (threshold) και μπορούμε να θεωρήσουμε ότι αντιστοιχεί σε είσοδο η τιμή της οποίας είναι πάντα ίση με τη μονάδα. Εάν το συνολικό άθροισμα από τις υπόλοιπες εισόδους του νευρώνα είναι μεγαλύτερο από την τιμή αυτή, τότε ο νευρώνας ενεργοποιείται. Εάν είναι μικρότερο, τότε ο νευρώνας παραμένει ανενεργός. Η ιδέα προέκυψε από τα βιολογικά νευρικά κύτταρα.

Όπως είναι φανερό, οι αριθμοί οι οποίοι συναποτελούν το διάνυσμα εισόδου (κάθε στοιχείο του διανύσματος τροφοδοτείται κατά τη λειτουργία του δικτύου σε έναν νευρώνα εισόδου), αλλά και οι αριθμοί οι οποίοι συναποτελούν το διάνυσμα εξόδου (κάθε στοιχείο του οποίου εμφανίζεται, μετά το πέρας του ολικού υπολογισμού, σε έναν νευρώνα εξόδου), περιγράφουν χαρακτηριστικά του προς επίλυση προβλήματος. Συνήθως αυτό που μας ενδιαφέρει είναι το δίκτυο να απεικονίζει με ορθό τρόπο διανύσματα εισόδου σε κατάλληλα διανύσματα εξόδου, το πρόβλημα δηλαδή είναι η υλοποίηση μίας συνάρτησης πολλαπλών μεταβλητών, κατά κανόνα περίπλοκης και με άγνωστο ακριβή τύπο. Τέτοιες απεικονίσεις έχουν εφαρμογή σε πολλούς τομείς της επιστήμης και της τεχνολογίας, αφού λειτουργούν ως αριθμητικά μοντέλα για πολλά διαφορετικά προβλήματα. Το ίδιο δίκτυο μπορεί να υλοποιήσει άπειρες διαφορετικές απεικονίσεις, μία για κάθε διαφορετική επιλογή συνόλου συναπτικών βαρών.

Το κύριο χαρακτηριστικό των νευρωνικών δικτύων είναι η εγγενής ικανότητα μάθησης. Ως μάθηση μπορεί να οριστεί η σταδιακή βελτίωση της ικανότητας του δικτύου να επιλύει κάποιο πρόβλημα (π.χ. η σταδιακή προσέγγιση μίας συνάρτησης). Η μάθηση επιτυγχάνεται μέσω της εκπαίδευσης, μίας επαναληπτικής διαδικασίας σταδιακής προσαρμογής των παραμέτρων του δικτύου (συνήθως των βαρών και της πόλωσής του) σε τιμές κατάλληλες ώστε να επιλύεται με επαρκή επιτυχία το προς εξέταση πρόβλημα. Αφού ένα δίκτυο εκπαιδευτεί, οι παράμετροί του συνήθως «παγώνουν» στις κατάλληλες τιμές και από εκεί κι έπειτα είναι σε λειτουργική κατάσταση. Το ζητούμενο είναι το λειτουργικό δίκτυο να χαρακτηρίζεται από μία ικανότητα γενίκευσης: αυτό σημαίνει πως δίνει ορθές εξόδους για εισόδους καινοφανείς και διαφορετικές από αυτές με τις οποίες εκπαιδεύτηκε.

Τέλος, όσον αφορά τις συναρτήσεις ενεργοποίησης $\phi(\cdot)$, οι πιο διαδεδομένες μορφές στη βιβλιογραφία είναι η *βηματική* (step), η *γραμμική* (linear), η *σιγμοειδής* (sigmoid), η *λογιστική σιγμοειδής* και η *υπερβολική εφαπτομένη*.

Κεφάλαιο 2

Μηχανική Μάθηση σε Κατανεμημένα Περιβάλλοντα

2.1 Κατανεμημένος Υπολογισμός

Ο *κατανεμημένος υπολογισμός* (distributed computing) αποτελεί [Cham18] έναν τομέα της επιστήμης των υπολογιστών που μελετά τα κατανεμημένα συστήματα. Ένα κατανεμημένο σύστημα είναι ένα σύστημα, τα συστατικά του οποίου βρίσκονται σε διαφορετικούς υπολογιστές συνδεδεμένους σε ένα δίκτυο, οι οποίοι επικοινωνούν και συντονίζουν τις ενέργειές τους διαβιβάζοντας μηνύματα μεταξύ τους. Τα συστατικά αλληλεπιδρούν μεταξύ τους ώστε να επιτύχουν έναν κοινό υπολογιστικό στόχο. Υπάρχουν πολλά παραδείγματα κατανεμημένων συστημάτων, από *αρχιτεκτονικές προσανατολισμένες στις υπηρεσίες* (service oriented architectures) μέχρι και *μαζικά online παιχνίδια πολλών παικτών*.

Ένα πρόγραμμα υπολογιστή που τρέχει μέσα σε ένα κατανεμημένο σύστημα καλείται κατανεμημένο πρόγραμμα. Υπάρχουν διαφορετικοί τρόποι υλοποιήσεων για τον μηχανισμό διαβίβασης μηνυμάτων, λ.χ. μέσω χρήσης του πρωτοκόλλου HTTP, της αρχιτεκτονικής RPC και ουρών μηνυμάτων.

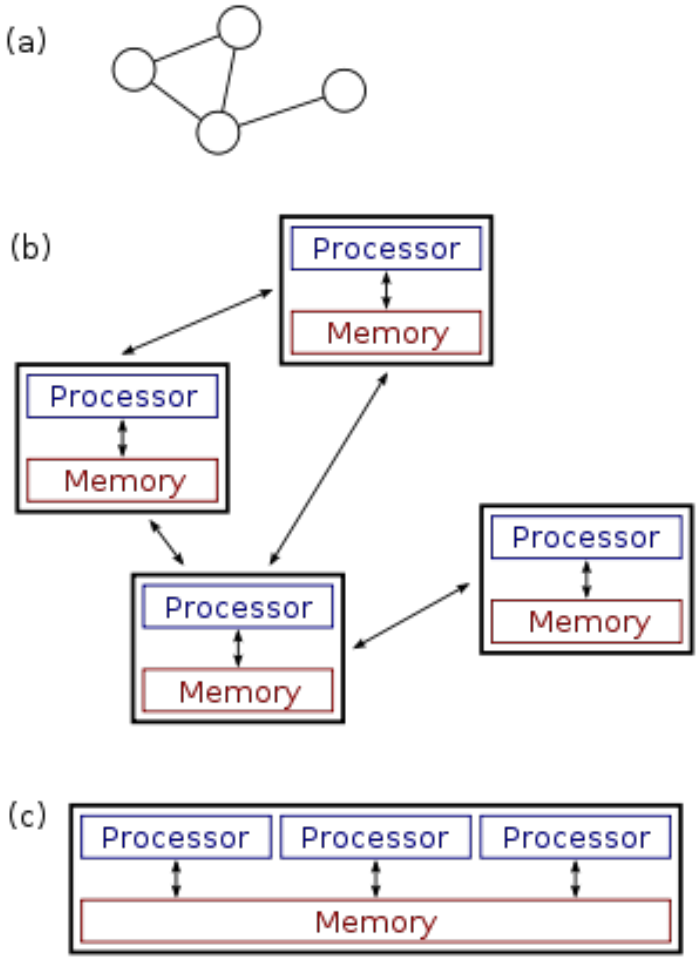
Ο κατανεμημένος υπολογισμός αναφέρεται επίσης στη χρήση των κατανεμημένων συστημάτων για την επίλυση υπολογιστικών προβλημάτων. Στον κατανεμημένο υπολογισμό, ένα πρόβλημα διαχωρίζεται σε πολλές εργασίες, κάθε μία εκ των οποίων επιλύεται από έναν ή περισσότερους υπολογιστές, η επικοινωνία των οποίων γίνεται με τη διαβίβαση μηνυμάτων.

2.1.1 Παράλληλος και Κατανεμημένος Υπολογισμός

Τα κατανεμημένα συστήματα αποτελούν ομάδες υπολογιστών συνδεδεμένων σε ένα δίκτυο, οι οποίοι μοιράζονται ένα κοινό στόχο για την εργασία τους [Tane02]. Οι όροι ταυτόχρονος υπολογισμός, παράλληλος υπολογισμός και κατανεμημένος υπολογισμός [Cham18] έχουν πολλά κοινά στοιχεία μεταξύ τους και δεν είναι εύκολη η διαφοροποίησή τους. Το ίδιο σύστημα μπορεί να χαρακτηριστεί και παράλληλο και κατανεμημένο. Οι επεξεργαστές σε ένα τυπικό κατανεμημένο σύστημα τρέχουν ταυτόχρονα παράλληλα. Ο παράλληλος υπολογισμός μπορεί να γίνει αντιληπτός ως μια πολύ κοντινή μορφή του κατανεμημένου υπολογισμού, και ο κατανεμημένος υπολογισμός μπορεί να γίνει κατανοητός ως μία πολύ μακρινή μορφή του παράλληλου υπολογισμού. Εκτός των άλλων, είναι δυνατό να ταξινομήσει κανείς τα ταυτόχρονα συστήματα ως παράλληλα ή κατανεμημένα χρησιμοποιώντας τα ακόλουθα κριτήρια:

- Στον παράλληλο υπολογισμό, όλοι οι επεξεργαστές μπορούν να έχουν πρόσβαση σε μια κοινή μνήμη για την ανταλλαγή πληροφοριών μεταξύ επεξεργαστών.
- Στον κατανεμημένο υπολογισμό, κάθε επεξεργαστής έχει τη δική του ιδιωτική μνήμη (κατανεμημένη). Η πληροφορία ανταλλάσσεται μεταβιβάζοντας μηνύματα μεταξύ των επεξεργαστών.

Στο Σχήμα 2.1 καταδεικνύεται η διαφορά μεταξύ κατανεμημένων και παράλληλων συστημάτων. Πιο συγκεκριμένα, στην περίπτωση (a) βλέπουμε μια σχηματική όψη ενός τυπικού κατανεμημένου συστήματος. Το σύστημα αναπαρίσταται ως μια τοπολογία δικτύου στην οποία κάθε κόμβος είναι ένας υπολογιστής και κάθε γραμμή είναι ένας δίαυλος επικοινωνίας που συνδέει τους κόμβους. Στην περίπτωση (b) βλέπουμε το ίδιο κατανεμημένο σύστημα με μεγαλύτερη λεπτομέρεια. Κάθε υπολογιστής



Σχήμα 2.1: Κατανεμημένα εναντίον Παράλληλων Συστημάτων

έχει τη δική του τοπική μνήμη και η πληροφορία μπορεί να ανταλλαγεί μόνο περνώντας μηνύματα από τον έναν κόμβο στον άλλο χρησιμοποιώντας τους διαθέσιμους διαύλους επικοινωνίας. Τέλος στην περίπτωση (c) βλέπουμε ένα παράλληλο σύστημα στο οποίο κάθε επεξεργαστής έχει άμεση πρόσβαση στην κοινή μνήμη.

Η κατάσταση γίνεται πιο περίπλοκη χρησιμοποιώντας τους παραδοσιακούς όρους του παράλληλου και του κατανεμημένου αλγορίθμου, καθώς δεν ταιριάζουν ακριβώς με τους ανωτέρω ορισμούς των παράλληλων και των κατανεμημένων συστημάτων. Παρόλ' αυτά, ως κανόνας, ο παράλληλος υπολογισμός υψηλών επιδόσεων σε έναν πολυεπεξεργαστή με κοινόχρηστη μνήμη, χρησιμοποιεί παράλληλους αλγορίθμους [Lync96], ενώ ο συντονισμός μεγάλων σε κλίμακα κατανεμημένων συστημάτων χρησιμοποιεί κατανεμημένους αλγόριθμους.

2.2 Κατανεμημένο Σύστημα Αρχείων

Τα *κατανεμημένα συστήματα αρχείων* (distributed file systems - DFS) δεν περιορίζονται [Gull15] στα αποθηκευτικά μέσα ενός μόνο υπολογιστικού κόμβου, αλλά χρησιμοποιούν ένα πρωτόκολλο δικτύου για να διαμοιράζουν αντίγραφα της πληροφορίας και σε άλλους υπολογιστικούς κόμβους του δικτύου [Silb09]. Αυτά είναι γνωστά ως *συστήματα αρχείων δικτύου* (network file systems), παρόλο που δεν είναι είναι τα μόνα συστήματα αρχείων που χρησιμοποιούν το δίκτυο για να στείλουν δεδομένα. Τα DFS μπορούν να περιορίσουν την πρόσβαση στο σύστημα αρχείων κάτι που εξαρτάται από τις λίστες πρόσβασης ή τις δυνατότητες τόσο των *εξυπηρετητών* (servers) όσο και των *πελατών* (clients), κάτι που στη συνέχεια εξαρτάται από το πώς είναι σχεδιασμένο το πρωτόκολλο.

Η διαφορά μεταξύ ενός DFS και μιας κατανεμημένης αποθήκης δεδομένων έγκειται στο ότι το πρώτο επιτρέπει στα αρχεία να προσπελαστούν χρησιμοποιώντας τις ίδιες διεπαφές και την ίδια σημασιολογία με τα τοπικά αρχεία, όπως προσάρτηση/αποπροσάρτηση, λίστα καταλόγων, διάβασμα-/εγγραφή σε όρια byte. Αντίθετα, η δεύτερη απαιτεί τη χρήση διαφορετικού API ή βιβλιοθήκης και έχει διαφορετική σημασιολογία.

2.2.1 Σχεδιαστικοί Στόχοι

Τα DFS στοχεύουν στη διαφάνεια όσον αφορά μια σειρά από χαρακτηριστικά. Αυτό σημαίνει πως είναι επιθυμητό να είναι διακριτά από τα προγράμματα πελάτη και να βλέπουν ένα σύστημα που μοιάζει με ένα τοπικό σύστημα αρχείων [Magn15]. Στο παρασκήνιο, το DFS χειρίζεται τοπικά αρχεία, μεταφέρει δεδομένα και πιθανώς παρέχει και άλλα χαρακτηριστικά, τα οποία αναφέρονται στη λίστα που ακολουθεί.

- **Διαφάνεια Πρόσβασης:** Το πρόγραμμα πελάτη δεν έχει πληροφορία σχετικά με το ότι τα αρχεία είναι κατανεμημένα και συνεπώς μπορεί να αποκτήσει πρόσβαση σε αυτά με τον ίδιο τρόπο που θα προσπέλαζε τοπικά αρχεία.
- **Διαφάνεια Θέσης:** Υπάρχει ένας σταθερός χώρος ονομάτων που περιλαμβάνει τοπικά και απομακρυσμένα αρχεία. Το όνομα ενός αρχείου δε δίνει την τοποθεσία του.
- **Διαφάνεια Συγχρονισμού:** Όλα τα προγράμματα πελάτη βλέπουν την ίδια εικόνα της κατάστασης του συστήματος αρχείων. Αυτό σημαίνει πως αν μια διαδικασία μεταβάλλει ένα αρχείο, τότε κάθε άλλη διεργασία στο ίδιο σύστημα ή σε απομακρυσμένα συστήματα που έχουν πρόσβαση στα αρχεία, θα δουν τις αλλαγές κατά έναν συνεκτικό τρόπο.
- **Διαφάνεια Σφάλματος:** Ο πελάτης και τα προγράμματά του πρέπει να λειτουργούν σωστά μετά από ένα σφάλμα του εξυπηρετητή.
- **Ετερογένεια:** Η υπηρεσία αρχείων πρέπει να παρέχεται σε διαφορετικές πλατφόρμες υλικού και λογισμικού.
- **Επεκτασιμότητα:** Το σύστημα αρχείων πρέπει να δουλεύει καλά σε μικρά περιβάλλοντα και να μπορεί να επεκταθεί με ευκολία σε μεγαλύτερα.
- **Διαφάνεια Αναπαραγωγής:** Τα προγράμματα πελάτη πρέπει να μη γνωρίζουν την αναπαραγωγή αρχείων που επιτελείται σε πολλαπλούς εξυπηρετητές για την υποστήριξη της επεκτασιμότητας.
- **Διαφάνεια Μεταφοράς:** Τα αρχεία θα πρέπει να μπορούν να μετακινούνται μεταξύ διαφορετικών εξυπηρετητών χωρίς τη γνώση του προγράμματος-πελάτη.

2.2.2 Διατηρήσιμη Μνήμη

Η *διατηρήσιμη μνήμη* (persistent memory) είναι μια μέθοδος [Cham18] για αποδοτική αποθήκευση δομών δεδομένων που μπορούν να συνεχίσουν να είναι προσβάσιμα με τη χρήση εντολών μνήμης ή προγραμματιστικών διεπαφών (application programming interface - API) μνήμης, ακόμα και μετά από τη διεργασία που τα δημιούργησε ή που τα μετέβαλε.

Συχνά συγχέεται με τη *μη-πτητική μνήμη τυχαίας προσπέλασης* (non-volatile random-access memory - NVRAM), ωστόσο η διατηρήσιμη μνήμη είναι πιο στενά συνδεδεμένη με την ιδέα της επιμονής, δίνοντας έμφαση στην κατάσταση του προγράμματος που πραγματοποιεί έξοδο εκτός της ζώνης σφάλματος της διεργασίας που το δημιούργησε. Να διευκρινιστεί πως μία διεργασία είναι ένα πρόγραμμα που εκτελείται. Επίσης, η ζώνη σφάλματος μιας διεργασίας είναι εκείνο το υποσύνολο των καταστάσεων του προγράμματος που μπορεί να καταστραφεί από τη διεργασία, αν αυτή συνεχίσει να εκτελείται μετά από την πραγματοποίηση ενός σφάλματος.

Το κύριο χαρακτηριστικό της διατηρήσιμης μνήμης είναι η αποδοτική πρόσβαση στη μνήμη. Μπορεί να αξιοποιηθεί χρησιμοποιώντας εντολές μικροεπεξεργαστή, όπως η *load* και η *store*. Μπορεί επίσης και με τη χρήση API που εφαρμόζουν απομακρυσμένη απευθείας πρόσβαση μνήμης. Μπορούν να χρησιμοποιηθούν και άλλες εντολές χαμηλής αδράνειας που επιτρέπουν διαχείριση bytes για την πρόσβαση στη μνήμη.

Οι δυνατότητές της ξεπερνούν την ιδιότητα της μη-πτητικότητας για τα αποθηκευμένα bit. Για παράδειγμα, η απώλεια σημαντικών μεταδεδομένων, όπως καταχωρήσεις σελίδων πινάκων ή άλλες κατασκευές που μεταφράζουν εικονικές διευθύνσεις σε φυσικές, μπορούν να καταστήσουν ανθεκτικά bit μη-πτητικά. Με αυτό υπόψιν, η διατηρήσιμη μνήμη παρομοιάζει πιο αφηρημένες μορφές υπολογιστικών μονάδων αποθήκευσης, όπως τα συστήματα αρχείων. Για την ακρίβεια, περίπου όλες οι υπάρχουσες τεχνολογίες διατηρήσιμης μνήμης εφαρμόζουν τουλάχιστον ένα βασικό σύστημα αρχείων ,που μπορεί να χρησιμοποιηθεί για τη συσχέτιση ονομάτων ή αναγνωριστών με αποθηκευμένα εύρη.

2.3 Hadoop

Το Apache Hadoop [Shva10] είναι μια συλλογή εργαλείων λογισμικού ανοιχτού κώδικα που διευκολύνουν τη χρήση ενός δικτύου πολλών υπολογιστών ώστε να λύσουν προβλήματα που έχουν σχέση με τεράστιες ποσότητες δεδομένων και υπολογισμών. Παρέχει μια δομή λογισμικού για κατανεμημένη αποθήκευση και επεξεργασία μεγάλων δεδομένων χρησιμοποιώντας το προγραμματιστικό μοντέλο MapReduce. Όλες οι ενότητες του Hadoop είναι σχεδιασμένες με πρωταρχική υπόθεση ότι τα σφάλματα υλικού είναι συχνά και θα πρέπει να χειρίζονται αυτόματα από τη δομή (framework).

Ο πυρήνας του Hadoop αποτελείται από ένα τμήμα αποθήκευσης, γνωστό ως Hadoop Distributed File System (HDFS), και ένα επεξεργαστικό τμήμα που είναι το υπολογιστικό μοντέλο MapReduce. Το Hadoop διαχωρίζει τα αρχεία σε μεγάλα μπλοκ και τα διανέμει στους κόμβους του cluster [Evan16]. Μετά, μεταφέρει κώδικα προς εκτέλεση σε κόμβους, ώστε τα δεδομένα να επεξεργάζονται παράλληλα. Αυτή η προσέγγιση εκμεταλλεύεται την τοποθεσία των δεδομένων (data locality), όπου οι κόμβοι χειρίζονται τα δεδομένα στα οποία έχουν πρόσβαση. Αυτό επιτρέπει στο σύνολο δεδομένων να μπορεί να τεθεί υπό επεξεργασία πιο γρήγορα και πιο αποδοτικά από ότι θα είχε σε μία πιο συμβατική υπερυπολογιστική αρχιτεκτονική, η οποία βασίζεται σε ένα παράλληλο σύστημα αρχείων, όπου ο υπολογισμός και τα δεδομένα διαβιβάζονται μέσω ενός δικτύου υψηλής ταχύτητας.

Η βάση της δομής του Apache Hadoop αποτελείται από τα ακόλουθα κομμάτια [Chou13]:

- **Hadoop Common:** Περιλαμβάνει βιβλιοθήκες και εργαλεία που χρειάζονται από άλλα κομμάτια του Hadoop.
- **Hadoop Distributed File System (HDFS):** Ένα κατανεμημένο σύστημα αρχείων που αποθηκεύει δεδομένα σε διαφορετικούς υπολογιστικούς κόμβους σε ένα cluster.
- **Hadoop YARN:** Μια πλατφόρμα που είναι υπεύθυνη για τη διαχείριση υπολογιστικών πόρων στο cluster και που τους χρησιμοποιεί για να προγραμματίσει εφαρμογές του χρήστη.
- **Hadoop MapReduce:** Μια υλοποίηση του προγραμματιστικού μοντέλου MapReduce για επεξεργασία δεδομένων μεγάλης κλίμακας.

2.3.1 Αρχιτεκτονική

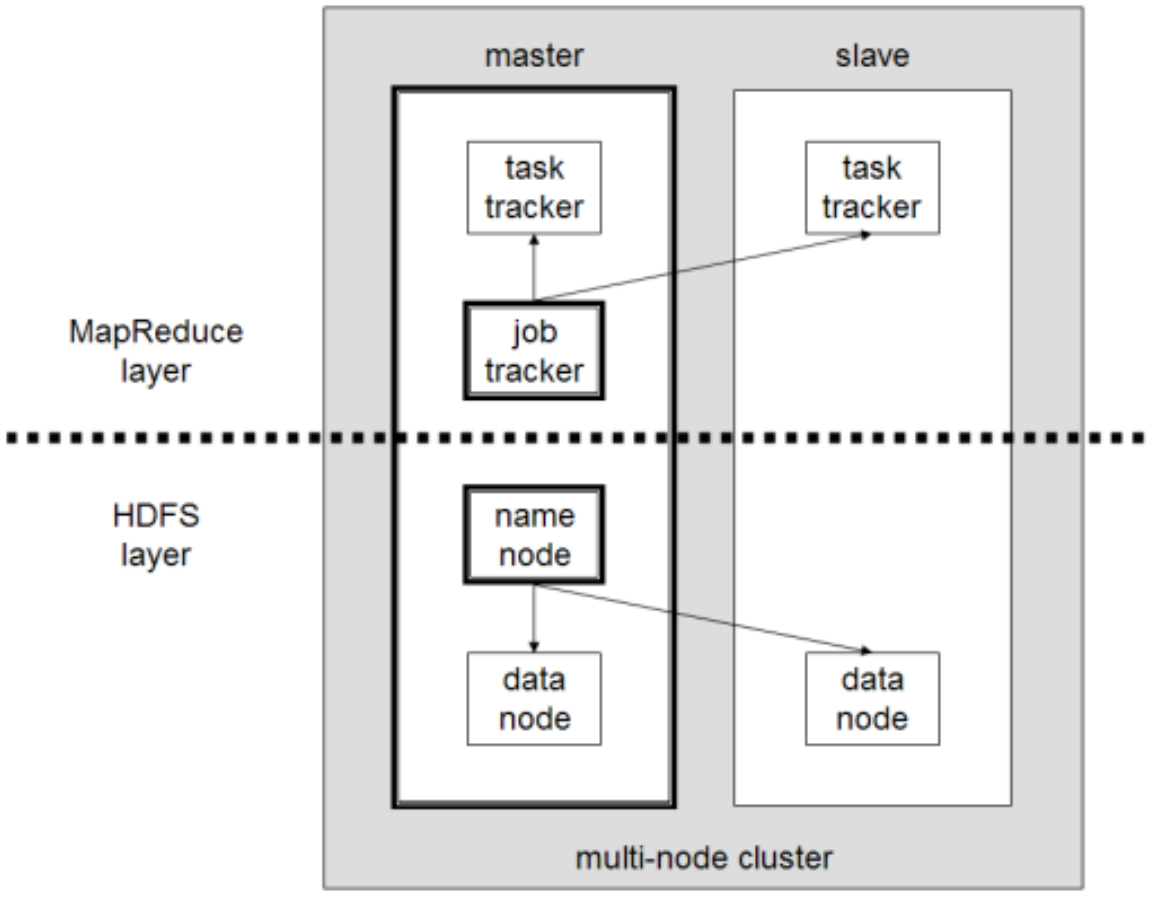
Το Hadoop αποτελείται από το πακέτο Hadoop Common, το οποίο παρέχει ένα σύστημα αρχείων και αφαιρετικά επίπεδα λειτουργικού συστήματος, μια μηχανή MapReduce (είτε τη MapReduce/MR1 είτε τη YARN/MR2) και το HDFS.

Για αποτελεσματικό προγραμματισμό των εργασιών [Cham18], κάθε σύστημα αρχείων συμβατό με Hadoop θα πρέπει να παρέχει επίγνωση της τοποθεσίας, η οποία είναι το όνομα του rack, συγκεκριμένα ο διακόπτης του δικτύου όπου βρίσκεται ο worker κόμβος. Οι εφαρμογές Hadoop μπορούν

να χρησιμοποιήσουν αυτήν την πληροφορία για να εκτελέσουν κώδικα πάνω στον κόμβο όπου βρίσκονται τα δεδομένα, και αν αποτύχουν να μειώσουν το traffic του ίδιου rack/διακόπτη. Το HDFS χρησιμοποιεί αυτή τη μέθοδο όταν αντιγράφονται δεδομένα για λόγους πλεονασμού (redundancy). Αυτή η προσέγγιση μειώνει τις συνέπειες που θα είχε η αποσύνδεση ενός κόμβου. Αν προκύψει κάποιο σφάλμα, τότε τα δεδομένα θα παραμείνουν διαθέσιμα.

Ένας μικρό Hadoop cluster [Gull15] περιλαμβάνει έναν κόμβο master και πολλαπλούς κόμβους workers. Ο κόμβος master αποτελείται από τους Job Tracker, τον Task Tracker, τον Name Node και τον Data Node. Ένας κόμβος worker ή slave λειτουργεί και ως Data Node και ως Task Tracker, παρόλο που είναι πιθανό να υπάρχουν κόμβοι workers μόνο για δεδομένα και μόνο για υπολογισμούς. Αυτοί χρησιμοποιούνται συνήθως σε μη-τυπικές εφαρμογές.

Σε ένα μεγάλο cluster, η διαχείριση των κόμβων HDFS γίνεται μέσω ενός αποκλειστικού εξυπηρετητή που καλείται Name Node και που μπορεί να φιλοξενήσει το ευρετήριο του συστήματος αρχείων και ένος δευτερεύοντα εξυπηρετητή Name Node, που μπορεί να αποθηκεύει στιγμιότυπα του πρώτου, προλαμβάνοντας κατ' αυτόν τον τρόπο την καταστροφή του συστήματος αρχείων και την απώλεια δεδομένων. Όμοια, ένας αυτόνομος εξυπηρετητής Job Tracker μπορεί να διαχειριστεί τον προγραμματισμό των εργασιών στους κόμβους. Όταν χρησιμοποιείται το Hadoop με ένα άλλο σύστημα αρχείων, τότε ο Name Node, ο δευτερεύοντας Name Node και ο Data Node του HDFS αντικαθίστανται από τα ισοδύναμα στοιχεία του άλλου συστήματος αρχείων.



Σχήμα 2.2: Απεικόνιση ενός Hadoop cluster με πολλούς κόμβους

Ακολουθεί μια συνοπτική περιγραφή των κόμβων που αναφέρθηκαν παραπάνω και επιπρόσθετων που εμφανίζονται στο Σχήμα 2.2

- **Name Node:** Το HDFS απαιτεί την ύπαρξη ενός Name Node που αποκαλείται Master Node. Αυτός ανιχνεύει αρχεία, διαχειρίζεται το σύστημα αρχείων και έχει μεταδεδομένα όλων των

αρχείων που είναι αποθηκευμένα σε αυτόν. Για την ακρίβεια, περιλαμβάνει λεπτομέρειες όπως τον αριθμό των μπλοκ, των θέσεων των Data Nodes που είναι αποθηκευμένα τα δεδομένα, το μέρος όπου είναι αποθηκευμένα τα αντίγραφα και άλλες λεπτομέρειες. Έχει απευθείας επαφή με το πρόγραμμα-πελάτη.

- **Data Node:** Ένας Data Node αποθηκεύει δεδομένα ως μπλοκ. Είναι επίσης γνωστός ως κόμβος slave και αποθηκεύει τα ακριβή δεδομένα στο HDFS, το οποίο είναι υπεύθυνο για την επικοινωνία με τα προγράμματα πελάτη. Κάθε Data Node στέλνει ένα μήνυμα (hearbeat) στον Name Node κάθε 3 δευτερόλεπτα και διευκρινίζει το ότι είναι ενεργός. Κατά αυτόν τον τρόπο, όταν ο Name Node δεν λάβει heartbeat από έναν Data Node για 2 λεπτά, θα θεωρήσει τον Data Node «νεκρό» και θα αρχίσει τη διαδικασία του να φορτώσει αντίγραφα από κάποιον άλλον Data Node.

- **Secondary Name Node:** Ο συγκεκριμένος κόμβος κρατάει είναι υπεύθυνος για τον έλεγχο των μεταδεδομένων του συστήματος. Ουσιαστικά είναι ένας βοηθητικός κόμβος του Name Node.

- **Job Tracker:** Λαμβάνει τα αιτήματα για εκτέλεση εργασιών MapReduce από το πρόγραμμα-πελάτη. Μιλάει με τον Name Node ώστε να ξέρει τη θέση των δεδομένων που θα χρειαστούν για την επεξεργασία. Ο Name Node απαντάει με μεταδεδομένα των απαιτούμενων δεδομένων προς επεξεργασία.

- **Task Tracker:** Είναι slave για τον Job Tracker. Λαμβάνει τον εκτελέσιμο κώδικα από τον Job Tracker και εφαρμόζει την επεξεργασία πάνω στο κατανεμημένο σύστημα αρχείων.

2.4 Spark

Το Apache Spark [Zaha16] αποτελεί μία δομή ανοιχτού κώδικα κατανεμημένου υπολογισμού γενικού σκοπού. Παρέχει μια διεπαφή για τον προγραμματισμό ολόκληρων clusters με παραλληλισμό δεδομένων και ανοχή σε σφάλματα. Αναπτύχθηκε αρχικά από το Πανεπιστήμιο της Καλιφόρνια και συγκεκριμένα στο εργαστήριο AMPLab και ο κώδικάς του παραχωρήθηκε αργότερα ως δωρεά στο Apache Software Foundation, το οποίο το συντηρεί μέχρι και σήμερα.

Έχει τις αρχιτεκτονικές του βάσεις στο Resilient Distributed Dataset (RDD), ένα πολυσύνολο δεδομένων μόνο για ανάγνωση, τα οποία είναι κατανεμημένα σε έναν cluster και τα οποία διατηρούν ανοχή στα σφάλματα. Το DataFrame API εκδόθηκε ως ένα αφαιρετικό επίπεδο πάνω από το RDD, ακολουθούμενο από το Dataset API. Η τεχνολογία RDD ακόμα αποτελεί τη βάση του Dataset API.

Το Spark και τα RDD αναπτύχθηκαν το 2012 ως απάντηση στους περιορισμούς του υπολογιστικού παραδείγματος του MapReduce , το οποίος επιβάλλει μια συγκεκριμένη ροή δεδομένων στα κατανεμημένα προγράμματα: τα προγράμματα MapReduce διαβάζουν δεδομένα εισόδου από τον δίσκο, αντιστοιχίζουν μια συνάρτηση στα δεδομένα, συσσωρεύουν τα αποτελέσματα της αντιστοίχισης και αποθηκεύουν τα αποτελέσματα της συσσώρευσης στον δίσκο. Τα Spark RDDs λειτουργούν ως ένα σύνολο εργασίας για κατανεμημένα προγράμματα και προσφέρουν μια περιορισμένη μορφή κατανεμημένης κοινόχρηστης μνήμης.

Διευκολύνουν την εφαρμογή τόσο επαναληπτικών αλγορίθμων, που προσπελάζουν το σύνολο δεδομένων τους πολλαπλές φορές μέσα στον βρόχο, όσο και διερευνητικής ανάλυσης δεδομένων [Tan17]. Η ταχύτητα εκτέλεσης τέτοιων εφαρμογών μπορεί να γίνει αρκετές τάξεις μεγέθους γρηγορότερα συγκριτικά με το MapReduce. Ανάμεσα στους επαναληπτικούς αλγορίθμους βρίσκονται οι αλγόριθμοι εκπαίδευσης των συστημάτων μηχανικής μάθησης, οι οποίοι αποτέλεσαν την αρχική ώθηση για την ανάπτυξη του Apache Spark.

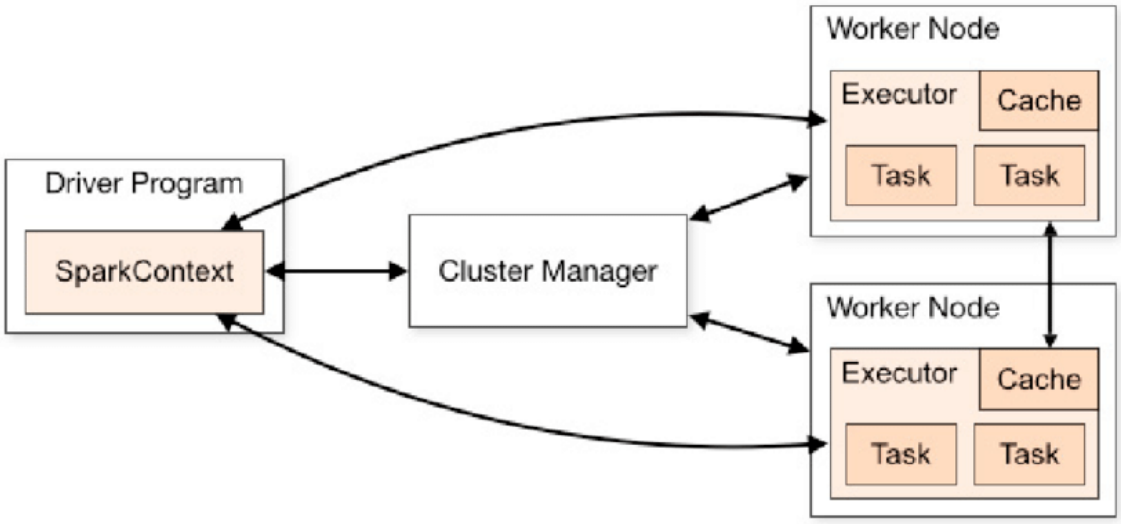
Το Apache Spark απαιτεί [Cham18] έναν διαχειριστή cluster και ένα κατανεμημένο σύστημα αποθήκευσης. Για διαχείριση cluster, υποστηρίζει αυτόνομους clusters Hadoop YARN και Apache Mesos ή Kubernetes. Για κατανεμημένη αποθήκευση, μπορεί να αποκτήσει σύνδεση με μια πληθώρα συστημάτων όπως το Alluxio, το HDFS, το MapR File System (MapR-FS), το Cassandra, το OpenStack Swift, το Amazon S3 και άλλες λύσεις. Υποστηρίζει επίσης μία ψευδο-κατανεμημένη λειτουργία, η

οποία χρησιμοποιείται συχνά μόνο για ανάπτυξη ή έλεγχο, όπου η κατανεμημένη αποθήκευση δεν είναι αναγκαία και μπορεί να χρησιμοποιηθεί το τοπικό σύστημα αρχείων. Σε ένα τέτοιο σενάριο, το Spark τρέχει σε μία μόνο μηχανή με έναν executor ανά πυρήνα της CPU.

2.4.1 Spark Core

Ο πυρήνας (Spark Core) αποτελεί [Cham18] τη βάση της όλης βιβλιοθήκης (Σχήμα 2.3). Παρέχει την κατανεμημένη αποστολή εργασιών, δρομολόγηση και βασικές λειτουργίες I/O, οι οποίες γίνονται ορατές μέσω μιας εφαρμογής προγραμματιστικής διεπαφής επικεντρωμένη στην αφαιρετικότητα των RDD. Αυτή η διεπαφή αντικατοπτρίζει ένα προγραμματιστικό μοντέλο υψηλής τάξης: ένα πρόγραμμα οδηγός καλεί παράλληλες διαδικασίες όπως ένας χάρτης, ένα φίλτρο ή μια μεταβολή ενός RDD, περνώντας μία συνάρτηση στο Spark, η οποία προγραμματίζει την εκτέλεση παράλληλα στον cluster. Αυτές οι διεργασίες, παίρνουν RDD ως εισόδους και παράγουν νέα RDD. Η ανοχή σε σφάλματα επιτυγχάνεται παρακολουθώντας τη σειρά των διεργασιών που παράγουν ένα RDD ώστε να μπορεί να ανακατασκευαστεί στην περίπτωση της απώλειας δεδομένων.

Εκτός από το συναρτησιακό προγραμματιστικό παράδειγμα με τη χρήση RDD [Gull15], το Spark προσφέρει δύο περιορισμένες μορφές κοινόχρηστων μεταβλητών: τις μεταβλητές broadcast που αναφέρονται σε δεδομένα μόνο προς ανάγνωση που χρειάζεται να είναι διαθέσιμα σε όλους τους κόμβους, και *συσσωρευτών* (accumulators), που μπορούν να χρησιμοποιηθούν για να προγραμματίσουν *μειώσεις* (reductions) στα δεδομένα.



Σχήμα 2.3: Ο Πυρήνας του Spark

2.4.2 Spark SQL

Η Spark SQL είναι ένα τμήμα του Spark Core που εισάγει μια αφηρημένη έννοια δεδομένων ονόματι DataFrames, η οποία παρέχει υποστήριξη για δομημένα και ημι-δομημένα δεδομένα. Παρέχει μια domain-specific language (DSL) για να διαχειρίζεται τα DataFrames σε Scala, Java ή Python. Επίσης, παρέχει και υποστήριξη SQL με διεπαφές γραμμής εντολών και έναν server ODBC/JDBC. Παρόλο που τα DataFrames δεν έχουν τον έλεγχο τύπων κατά τον χρόνο μεταγλώττισης, όπως τα RDD, από την έκδοση 2.0 του Spark, ο τύπος DataSet υποστηρίζεται και από την Spark SQL.

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
```

```
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

# Use an example JSON as the data source
df = spark.read.json("examples/src/main/resources/people.json")

# Print the schema in a tree format
df.printSchema()
# root
# |-- age: long (nullable = true)
# |-- name: string (nullable = true)

# Count people by age
df.groupBy("age").count().show()
# +----+-----+
# | age|count|
# +----+-----+
# |  19|    1|
# |null|    1|
# |  30|    1|
# +----+-----+
```

Listing 2.1: Βασικό Παράδειγμα Spark SQL

2.4.3 Spark MLlib

Η Spark MLlib [Gull15] είναι μια δομή κατανεμημένης μηχανικής μάθησης πάνω από τον Spark Core, η οποία σε μεγάλο βαθμό, εξαιτίας της κατανεμημένης αρχιτεκτονικής βασισμένη σε μνήμη του Spark, είναι πιο γρήγορη από άλλες αντίστοιχες βιβλιοθήκες όπως το Apache Mahout [Owen12] και το Vowpal Wabbit [Lang07]. Στα παρακάτω σημεία συνοψίζονται κάποιες από τις λειτουργικότητες της Spark MLlib:

- υπολογισμός συσχετίσεων, έλεγχος υποθέσεων, δειγματοληψία, δημιουργία τυχαίων δεδομένων
- ταξινόμηση και παλινδρόμηση με τη χρήση μηχανών διανυσμάτων υποστήριξης, απλού μπεϋζιανού ταξινομητή, δένδρων αποφάσεων, τυχαίων δασών
- Αλγόριθμοι συστάσεων συνεργατικής διήθησης και παραγοντοποίησης πινάκων
- Αλγόριθμοι συσταδοποίησης όπως ο k-μέσων και η λανθάνουσα κατανομή του Dirichlet
- Τεχνικές μείωσης διαστάσεων όπως η ανάλυση ιδιαζουσών τιμών και η ανάλυση κυρίων συνιστωσών
- Εξαγωγή χαρακτηριστικών και συναρτήσεις μετασχηματισμού

2.5 H2O

Η H2O είναι μια ανοιχτού κώδικα, κατανεμημένη, γρήγορη και κλιμακούμενη πλατφόρμα μηχανικής μάθησης και προγνωστικής αναλυτικής που επιτρέπει τη δημιουργία μοντέλων μηχανικής μάθησης σε μεγάλα δεδομένα και παρέχει τη δυνατότητα εύκολης υλοποίησης των μοντέλων αυτών σε επιχειρηματικό περιβάλλον [H2Oa].

Ο πυρήνας της H2O είναι γραμμένος σε Java. Στο εσωτερικό της, χρησιμοποιείται μια αποθήκη κατανεμημένων κλειδιών και τιμών για την πρόσβαση και την αναφορά δεδομένων, μοντέλων, αντικειμένων, κλπ., κατά μήκος όλων των κόμβων και μηχανών. Οι αλγόριθμοι υλοποιούνται πάνω στην

κατανεμημένη MapReduce δομή της H2O και χρησιμοποιεί τη δομή Fork/Join της Java για πολυνηματισμό. Τα δεδομένα διαβάζονται παράλληλα και κατανέμονται στον cluster και αποθηκεύονται στη μνήμη σε μια μορφή στηλών και μάλιστα συμπιεσμένα. Ο parser των δεδομένων την H2O έχει ενσωματωμένη νοημοσύνη ώστε να προβλέπει το *σχήμα* (schema) των εισερχόμενων δεδομένων και υποστηρίζει τη λήψη δεδομένων από πολλαπλές πηγές και διάφορες μορφές.

Το API της H2O επιτρέπει την πρόσβαση σε όλες τις δυνατότητες της H2O από ένα εξωτερικά προγράμματα-πελάτη. Χρησιμοποιείται από τη διαδικτυακή διεπαφή της H2O, ονόματι Flow UI, τον σύνδεσμο με την R (H2O-R) και τον σύνδεσμο με την Python (H2O-Python).

Η ταχύτητα, η ποιότητα, η ευκολία στη χρήση και η ανάπτυξη μοντέλων αιχμής για τους διάφορους αλγορίθμους μάθησης με επίβλεψη και μάθησης χωρίς επίβλεψη κάνουν την H2O ένα περιζήτητο API για την επιστήμη των μεγάλων δεδομένων.

2.5.1 Υποστηριζόμενοι Αλγόριθμοι

Η H2O υποστηρίζει μια πληθώρα ευρέως χρησιμοποιούμενων αλγορίθμων μηχανικής μάθησης, όπως αυτοί που φαίνονται στην παρακάτω λίστα:

- **Στατιστική Ανάλυση**: γενικευμένα γραμμικά μοντέλα, απλός μπεϋζιανός κανόνας
- **Βαθιά Νευρωνικά Δίκτυα**: βαθιά δίκτυα πρόσθιας τροφοδότησης
- **Συσταδοποίηση**: Υποστήριξη αλγορίθμου k-μέσων
- **Μείωση διαστατικότητας**: ανάλυση κυρίων συνιστωσών και γενικευμένα μοντέλα χαμηλής τάξης
- **Εύρεση ανωμαλιών**: αυτοκωδικοποιητές

2.5.2 Παρόμοιες Επεκτάσεις

Η εταιρεία που δημιούργησε την H2O προσφέρει και άλλες λύσεις για την επεξεργασία μεγάλων δεδομένων όλες βασισμένες σε αυτήν [Pand20].

- **H2O Sparkling Water:** Επιτρέπει στους χρήστες να συνδυάσουν τους γρήγορους και επεκτάσιμους αλγορίθμους μηχανικής μάθησης της H2O με τις δυνατότητες του Spark. Είναι ιδανικό για χρήστες που έχουν την ανάγκη να διαχειριστούν μεγάλα cluster για την επεξεργασία των δεδομένων τους και που θέλουν να μεταφέρουν δεδομένα από το Spark στην H2O (ή το αντίστροφο).
- **H2O4GPU:** Είναι ένα πακέτο για μηχανική μάθηση με επιτάχυνση GPU χρησιμοποιώντας API στην Python και την R ώστε να μπορεί ο οποιοσδήποτε να αξιοποιήσει GPU για τη δημιουργία ισχυρών μοντέλων μηχανικής μάθησης.
- **H2O Driverless AI:** Αυτοματοποιεί πλήρως κάποιες από τις πιο δύσκολες και απαιτητικές διαδικασίες στην εφαρμοσμένη επιστήμη δεδομένων, όπως η μηχανική χαρακτηριστικών και η ρύθμιση των μοντέλων.

2.6 Χαρακτηριστικά Κατανεμημένης Βαθιάς Μάθησης

Η κατανεμημένη βαθιά μάθηση είναι μια υποκατηγορία της γενικής κατανεμημένης μηχανικής μάθησης, η οποία έχει γίνει πλέον εξέχουσα εξαιτίας της αποτελεσματικότητάς της σε διάφορες εφαρμογές [Petu18].

2.6.1 Παραλληλία Δεδομένων

Η παραλληλία των δεδομένων είναι μια τεχνική παραλληλοποίησης της οποίας η χρήση γίνεται εφικτή από τον διαχωρισμό των δεδομένων. Στον κατανεμημένο υπολογισμό παράλληλων δεδομένων, αρχικά διαιρούμε τα δεδομένα σε λίγα *διαμερίσματα* (partitions), με τον αριθμό αυτών να είναι ίσος με τον αριθμό των υπολογιστικών κόμβων. Ύστερα, αφήνουμε κάθε κόμο να πάρει από ένα ανεξάρτητο partition και αφήνουμε τον κάθε έναν να εκτελέσει υπολογισμούς σε αυτά τα δεδομένα. Επειδή έχουμε πολλαπλούς κόμβους που σκανάρουν παράλληλα τα δεδομένα, πρέπει να μπορούμε να σκανάρουμε περισσότερα δεδομένα από όταν χρησιμοποιούμε έναν μόνο κόμβο. Έτσι αυξάνεται η απόδοση των συστημάτων παράλληλου υπολογισμού.

Στην κατανεμημένη μηχανική μάθηση, όπου ο στόχος μας είναι να επιταχύνουμε τη σύγκλιση του μοντέλου που εκπαιδεύουμε χρησιμοποιώντας πολλαπλούς κόμβους, η εφαρμογή της παραλληλίας των δεδομένων είναι διαισθητική. Αφήνουμε κάθε worker να εκτελέσει την εκπαίδευση στο δικό του partition δεδομένων και να παράγει στη συνέχεια ένα σύνολο ενημερωμένων παραμέτρων. Μετά, αφήνουμε όλους τους κόμβους να συγχρονίσουν τις παραμέτρους τους μέσω δικτυακής επικοινωνίας μέχρι να φτάσουν σε μια κοινή απόφαση. Όσο ο συγχρονισμός δεν παίρνει πολλή ώρα και βλέπουμε βελτίωση σε σχέση με τα αποτελέσματα ενός κόμβου, έχουμε πετύχει τον στόχο μας.

2.6.2 Παραλληλία Μοντέλων

Συγκριτικά με τον παραλληλισμό δεδομένων, ο παραλληλισμός μοντέλων είναι μια πολύ πιο περίπλοκη και αφηρημένη έννοια. Γενικά, εδώ αντί να διαχωρίσουμε τα δεδομένα, προσπαθούμε να διαχωρίσουμε το ίδιο το μοντέλο μηχανικής μάθησης ώστε να κατανέμει το φόρτο εργασίας σε πολλαπλούς υπολογιστικούς κόμβου. Για παράδειγμα, ας θεωρήσουμε ότι έχουμε ένα πρόβλημα παραγοντοποίησης πίνακα, όπου ο πίνακας είναι πάρα πολύ μεγάλος και θέλουμε να μάθουμε κάθε παράμετρο αυτού. Για να εφαρμόσουμε παραλληλία μοντέλου, πρέπει να διαχωρίσουμε τον πίνακα σε πολλά μικρά μπλοκ (μικρότερους πίνακες), και μετά να αφήσουμε κάθε κόμβο να επεξεργαστεί μερικούς. Κατά αυτόν τον τρόπο, είμαστε σε θέση να εκμεταλλευτούμε την επιπλέον μνήμη των πολλαπλών κόμβων, αν η μνήμη σε έναν κόμβο δεν είναι αρκετή ώστε να αποθηκεύσει όλες τις παραμέτρους του πίνακα. Αφού διαφορετικοί κόμβοι έχουν διαφορετικά βάρη εργασιών που αντιστοιχούν σε διαφορετικά μπλοκ του πίνακα, θα πρέπει να παρατηρούμε μια επιτάχυνση όταν εκτελούν τους υπολογισμούς παράλληλα.

Η δυσκολία έγκειται στο πώς μπορούμε να διαχωρίσουμε το μοντέλο, εφόσον έχουμε πολλά μοντέλα μηχανικής μάθησης και κάθε ένα μοντέλο έχει τα δικά του χαρακτηριστικά και τις δικές του αναπαραστάσεις, και έτσι δεν υπάρχει κάποια αρχή να εφαρμόσουμε για τον παραλληλισμό μοντέλων.

2.6.3 Προβλήματα της Κατανεμημένης Μηχανικής Μάθησης

Η παραλληλία δεδομένων είναι αρκετά αποτελεσματική όταν το μέγεθος των δεδομένων εκπαίδευσης είναι μεγάλο, καθώς μπορούμε να σκανάρουμε τα δεδομένα πολύ πιο γρήγορα από ότι με παραπάνω κόμβους. Είναι πιο εύκολα εφαρμόσιμη σε περιπτώσεις όπου το μέγεθος των μοντέλων είναι πολύ μεγάλο για έναν κόμβο, αφού μας επιτρέπει να διαχωρίσουμε τα μοντέλα και να χρησιμοποιήσουμε την επιπλέον μνήμη κατά μήκος πολλαπλών κόμβων.

Ιδανικά, στον κατανεμημένο υπολογισμό, θέλουμε να έχουμε τόσο πιο πολλές επιταχύνσεις όσο ο αριθμός των μηχανών που χρησιμοποιούμε (συνήθως αποκαλούμε αυτό το χαρακτηριστικό μετρική επεκτασιμότητα). Για την ακρίβεια, αν χρησιμοποιήσουμε K μηχανές και τα σύστημά μας μπορούν να σκανάρουν δεδομένα K φορές πιο γρήγορα από ότι σε έναν κόμβο ανά μονάδα χρόνου, τότε το σύστημά μας έχει επεκτασιμότητα K ή γραμμική επεκτασιμότητα. Η γραμμική επεκτασιμότητα είναι ιδανική σε συστήματα τέτοιου τύπου.

Ωστόσο, εξαιτίας των καθυστερήσεων που προέρχονται από τις ενέργειες συγχρονισμού, παίρνει πολύ περισσότερο χρόνο η ολοκλήρωση μιας επανάληψης εκπαίδευσης σε ένα κατανεμημένο

υπολογιστικό cluster από ότι σε έναν μόνο κόμβο. Πρέπει να ξοδέψουμε παραπάνω χρόνο για τον συγχρονισμό κατά μήκος πολλών κόμβων στο τέλος του υπολογισμού, ώστε να είμαστε βέβαιοι για τη σύγκλιση της διεργασίας εκπαίδευσης.

Πρακτικά, ο συγχρονισμός μπορεί να πάρει τόσο χρόνο όσο ο υπολογισμός ή ίσως και παραπάνω. Ένας από τους κύριους λόγους για αυτό αποτελεί το γεγονός πως κάποιοι κόμβοι στον cluster τρέχουν πιο αργά από άλλους κόμβους και για να μπορέσουν να συγχρονιστούν μαζί τους πρέπει οι πιο γρήγοροι κόμβοι να περιμένουν μέχρι οι εργασίες του υπολογισμού να τελειώσουν. Σε τέτοιες περιπτώσεις, η επιλογή του να βάλουμε K μηχανές μαζί πιθανότατα θα καταλήξει σε αρνητική επεκτασιμότητα, κάτι το οποίο είναι τεράστια σπατάλη χρόνου και χρημάτων.

Κεφάλαιο 3

Αλγόριθμοι

Στη μηχανική μάθηση, η ταξινόμηση πολλαπλών κλάσεων είναι το πρόβλημα της ταξινόμησης δειγμάτων σε περισοτερες από δύο κατηγορίες. Παρόλο που πολλοί αλγόριθμοι ταξινόμησης επιτρέπουν εκ φύσεως τη χρήση περισσότερων από δύο κατηγοριών, άλλοι είναι εκ φύσεως δυαδικοί. Ωστόσο, υπάρχουν τεχνικές που επιτρέπουν τη μετατροπής τους σε ταξινομητές πολλαπλών κλάσεων.

3.1 Δέντρα Απόφασης

3.1.1 Γενικά

Η μάθηση με *δέντρα απόφασης* (decision trees - DT) [Quin86] αποτελεί μια μέθοδο χρησιμοποιούμενη ευρέως στην περιοχή της εξόρυξης δεδομένων. Ο στόχος είναι η δημιουργία ενός μοντέλου που προβλέπει την τιμή μιας μεταβλητής που έχει οριστεί ως στόχος και βασίζεται σε αρκετές μεταβλητές εισόδου.

Ένα DT είναι μια απλή αναπαράσταση παραδειγμάτων ταξινόμησης. Ας υποθέσουμε ότι όλα τα χαρακτηριστικά εισόδου έχουν απαριθμήσιμους διακριτούς τομείς, και ότι υπάρχει ένα μοναδικό χαρακτηριστικό-στόχος στη βάση του οποίου πραγματοποιείται η ταξινόμηση. 'Ενα DT είναι ένα δέντρο μέσα στο οποίο κάθε εσωτερικός κόμβος (όχι φύλλο) είναι ετικετοποιημένος με ένα χαρακτηριστικό εισόδου. Οι ακμές που διέρχονται από έναν κόμβο και είναι ετικετοποιημένες με ένα χαρακτηριστικό εισόδου ετικετοποιούνται με κάθε μία από τις πιθανές τιμές του χαρακτηριστικού στόχου ή η ακμή οδηγεί σε ένα κατώτερο κόμβο απόφασης ενός διαφορετικού χαρακτηριστικού εισόδου. Κάθε φύλλο του δέντρου είναι ετικετοποιημένο με μία κλάση ή μια πιθανότητα κατανομής των κλάσεων, δείχνοντας πως το σύνολο δεδομένων έχει ταξινομηθεί από ένα δέντρο είτε σε μία συγκεκριμένη κλάση είτε σε μια συγκεκριμένη κατανομή πιθανότητας, η οποία αν το DT είναι καλά δομημένο, τότε ρέπει προς συγκεκριμένα υποσύνολα των κλάσεων.

Ένα δέντρο χτίζεται διαχωρίζοντας το πηγαίο σύνολο, συγκροτώντας τη ρίζα του δέντρου, σε υποσύνολα, τα οποία συγκροτούν τα παιδιά διαδόχους. Ο διαχωρισμός βασίζεται σε ένα σύνολο κανόνων διαχωρισμού που με τη σειρά του βασίζεται σε χαρακτηριστικά ταξινόμησης. Η διαδικασία επαναλαμβάνεται σε κάθε επιθυμητό υποσύνολο κατά έναν αναδρομικό τρόπο. Η αναδρομή ολοκληρώνεται όταν το υποσύνολο σε έναν κόμβο έχει όλες τις τιμές του ίδιες με τη μεταβλητή στόχο ή όταν περαιτέρω διαχωρισμός δεν προσφέρει παραπάνω στις προβλέψεις. Η διαδικασία της από πάνω προς τα κάτω επαγωγής των δέντρων απόφασης αποτελεί ένα παράδειγμα άπληστου αλγορίθμου και είναι με διαφορά η πιο ευρέως χρησιμοποιούμενη στρατηγική για εκμάθηση DT από δεδομένα.

Στην εξόρυξη δεδομένων [Lior07], τα δεδομένα έχουν τη μορφή $(\mathbf{x}, Y) = (x_1, x_2, x_3, ..., x_k, Y)$, όπου η εξαρτημένη μεταβλητή Y είναι η μεταβλητή στόχος, που προσπαθούμε να γενικεύσουμε. Το διάνυσμα $\mathbf{x}$ αποτελείται από τα χαρακτηριστικά $x_1, x_2, x_3, ..., x_k$ που χρησιμοποιούνται για αυτή τη διεργασία.

Τα DT χωρίζονται σε δύο κύριες κατηγορίες:

- Δέντρα Ταξινόμησης, όταν το προβλεπόμενο αποτέλεσμα είναι η κλάση (διακριτή) στην οποία ανήκουν τα δεδομένα.

- Δέντρα Παλινδρόμησης, όταν το προβλεπόμενο αποτέλεσμα μπορεί να είναι πραγματικός αριθμός (π.χ. η τιμή ενός σπιτιού).

Υπάρχουν κάποιες τεχνικές, αποκαλούμενες *συλλογικές μέθοδο*ι (ensemble methods), που κατασκευάζουν περισσότερα από ένα DT. Αυτές είναι:

- **Boosted Trees:** Κατασκευάζει αυξανόμενα ένα σύνολο, εκπαιδεύοντας κάθε νέο παράδειγμα ώστε να δώσει έμφαση στα παραδείγματα εκπαίδευσης που προηγουμένως δε μοντελοποιήθηκαν σωστά. Ένα παράδειγμα αποτελεί ο αλγόριθμος AdaBoost. Μπορούν να χρησιμοποιηθούν για προβλήματα τύπου παλινδρόμησης και ταξινόμησης.

- **Bootstrap Aggregated ή Bagged Decision Trees:** Μια πρώτη έκδοση των ensemble αλγορίθμων που κατασκευάζει πολλαπλά DT χρησιμοποιώντας δειγματοληψία με επανατοποθέτηση δεδομένων εκπαίδευσης. Τυπικό παράδειγμα αποτελούν τα τυχαία δάση που θα μελετηθούν στην συνέχεια

- **Rotation Forest:** Εδώ κάθε DT εκπαιδεύεται χρησιμοποιώντας πρώτα PCA σε ένα τυχαίο υποσύνολο των χαρακτηριστικών εισόδου.

3.1.2 Βασικός Αλγόριθμος

Ο βασικός αλγόριθμος κατασκευής DT είναι ένας άπληστος αλγόριθμος [Brei93] που εκτελεί μία αναδρομική δυαδική διαμέριση του χώρου των χαρακτηριστικών. Το δέντρο προβλέπει την ίδια ετικέτα για κάθε βαθύτερο (αριστερότερο) φύλλο. Κάθε χώρισμα επιλέγεται άπληστα επιλέγοντας το καλύτερο χώρισμα από ένα σύνολο πιθανών χωρισμάτων, ώστε να μεγιστοποιήσει το *κέρδος πληροφορίας* (information gain) σε ένα κόμβο του δέντρου. Με άλλα λόγια, το χώρισμα σε κάθε κόμβο επιλέγεται από το σύνολο $\arg\max_s IG(D, s)$, όπου το $IG(D, s)$ είναι το κέρδος πληροφορίας όταν ένα χώρισμα s εφαρμοστεί σε ένα σύνολο δεδομένων D.

3.1.3 Ανομοιογένεια Κόμβου και Κέρδος Πληροφορίας

Η *ανομοιογένεια κόμβου* (node impurity) είναι ένα μέτρο της ομοιογένειας των ετικετών στον κόμβο. Το κέρδος πληροφορίας είναι η διαφορά ανάμεσα στην ανομοιογένεια του κόμβου πατέρα και του αθροίσματος με βάρη των ανομοιογενειών των δύο κόμβων παιδιών. Υποθέτοντας ότι ένα χώρισμα s αθροισμάτων του συνόλου δεδομένων D μεγέθους σε δύο σύνολα δεδομένων D_{left} και D_{right} μεγεθών N_{left} και N_{right} αντίστοιχα, το κέρδος πληροφορίας είναι (Εξίσωση 3.1):

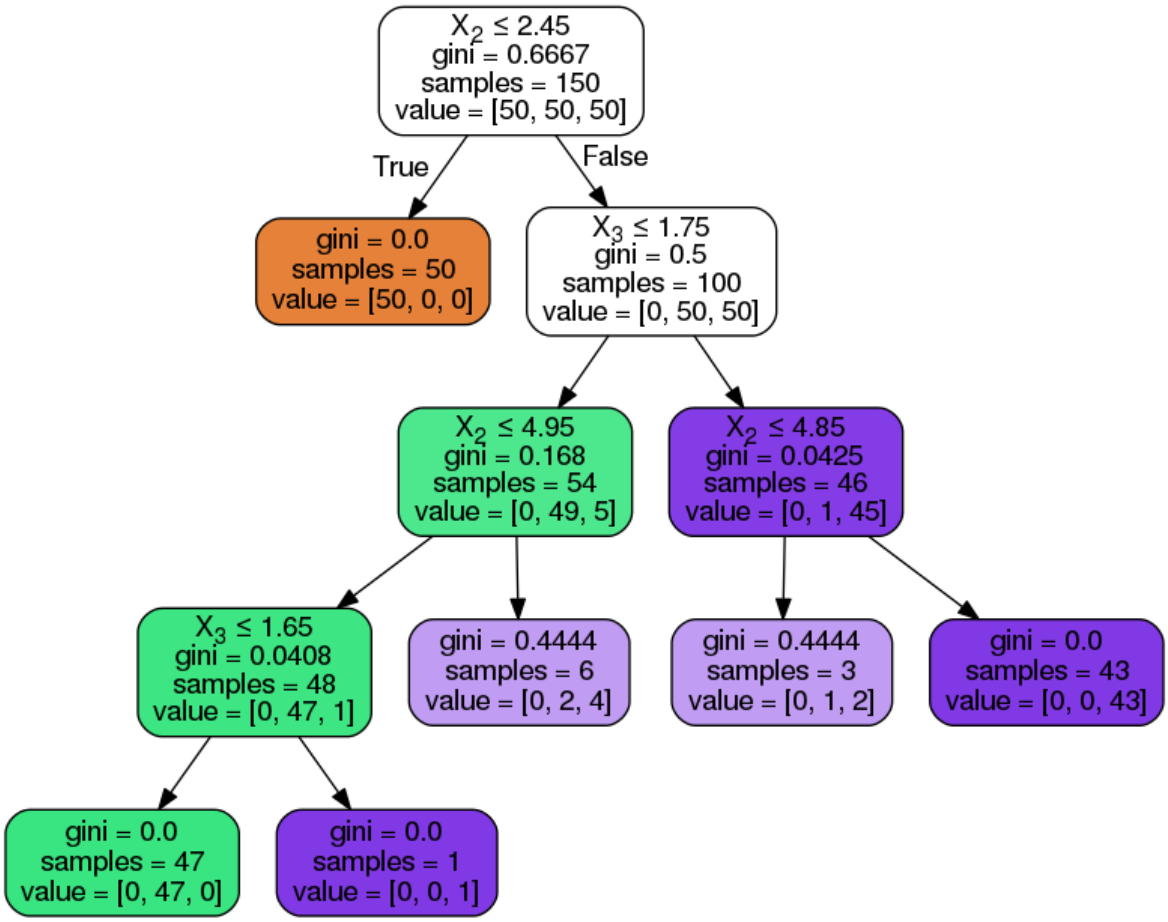
$$IG(D, s) = Impurity(D) - \frac{N_{left}}{N} Impurity\left(D_{left}\right) - \frac{N_{right}}{N} Impurity\left(D_{right}\right) \tag{3.1}$$

Στο Σχήμα 3.1 δίνεται ένα παράδειγμα δέντρου απόφασης.

3.2 Τυχαία Δάση

Σύμφωνα με την εργασία [Hast01], η εκμάθηση με δέντρα έρχεται όλο και πιο κοντά στην εκπλήρωση των απαιτήσεων για τη χρήση της ως μιας τυποποιημένης διαδικασίας για την εξόρυξη δεδομένων, καθώς είναι αμετάβλητη υπό κλίμακα και διάφορους άλλους μετασχηματισμούς των τιμών των χαρακτηριστικών, είναι ισχυρή για την συμπερίληψη άσχετων χαρακτηριστικών και παράγει εύκολα ελεγχόμενα μοντέλα. Ωστόσο, σπάνια είναι ακριβής.

Για την ακρίβεια, τα δέντρα που μεγαλώνουν πολύ σε βάθος τείνουν να μαθαίνουν σε ασυνήθιστα μοτίβα: *υπερ-προσαρμόζονται* (overfit) στο σύνολο εκπαίδευσης, έχουν χαμηλή *προκατάληψη* (bias), αλλά πολύ υψηλή *διακύμανση* (variance). Τα *τυχαία δάση* (random forests - RF) είναι ένας τρόπος της εξαγωγής του μέσου όρου πολλαπλών βαθιών DT, εκπαιδευμένα σε διαφορετικά κομμάτια του ίδιου



Σχήμα 3.1: Παράδειγμα Δένδρων απόφασης

συνόλου εκπαίδευσης και με στόχο τη μείωση της διακύμανσης. Αυτό βέβαια προσδίδει μια μικρή αύξηση της προκατάληψης και κάποια απώλεια της ερμηνείας, αλλά γενικά αυξάνει την απόδοση στο τελικό μοντέλο. Το Σχήμα 3.2 περιέχει ένα παράδειγμα RF.

3.2.1 Bagging

Ο αλγόριθμος εκπαίδευσης για τα RF εφαρμόζει την γενική τεχνική bootstrap aggregating ή bagging [Brei96]. Δοθέντος ενός συνόλου εκπαίδευσης $X = x_1, ..., x_n$ με στόχους $Y = y_1, ..., y_n$, το bagging επιλέγει επαναλαμβανόμενα (B φορές) ένα τυχαίο δείγμα με αντικατάσταση του συνόλου εκπαίδευσης και προσαρμόζει τα δέντρα σε αυτά τα δείγματα:

Για $b = 1, ..., B$:

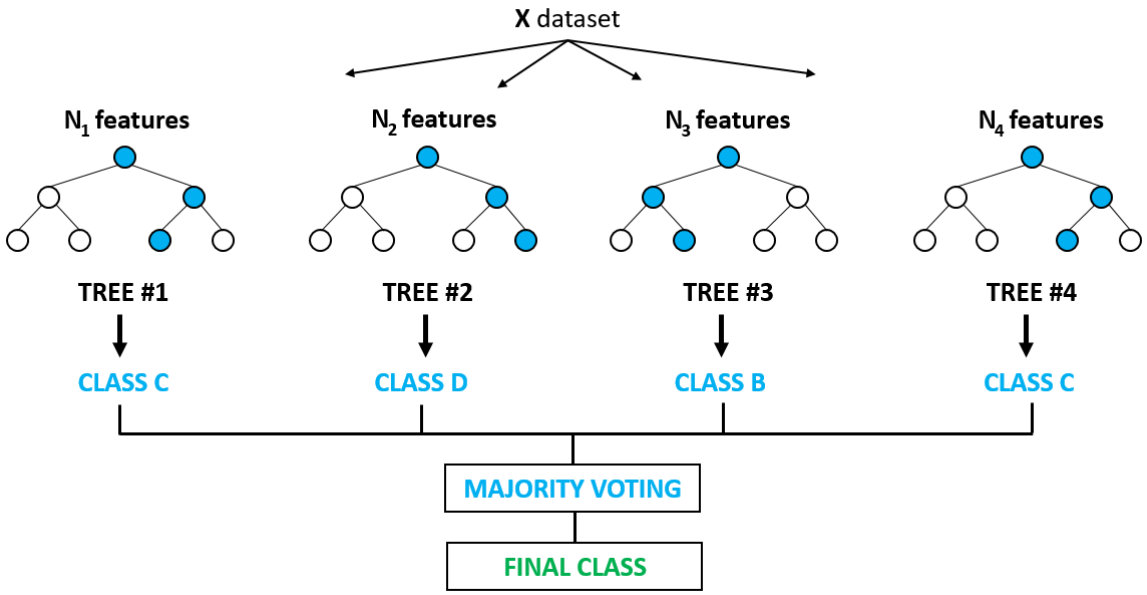
1. Κάνε δειγματοληψία με αντικατάσταση, n παραδειγμάτων εκπαίδευσης από τα X, Y και αποκάλεσέ τα X_b, Y_b.

2. Εκπαίδευσε ένα δέντρο ταξινόμησης ή παλινδρόμησης f_b στα X_b, Y_b.

Μετά την εκπαίδευση, οι προβλέψεις για τα δείγματα που δεν παρατηρήθηκαν, x', μπορούν να δημιουργηθούν εξάγοντας τον μέσο όρο των προβλέψεων από κάθε ξεχωριστό δέντρο παλινδρόμησης στο x' (Εξίσωση 3.2)

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x') \tag{3.2}$$

ή λαμβάνοντας την ψήφο της πλειοψηφίας στην περίπτωση των δέντρων ταξινόμησης.



Σχήμα 3.2: Παράδειγμα Τυχαίων Δασών

Αυτή η διαδικασία bootstrapping [Bish06] οδηγεί σε καλύτερη απόδοση του μοντέλου καθώς μειώνει την διακύμανση χωρίς να αυξάνει την προκατάληψη. Αυτό σημαίνει πως παρόλο που οι προβλέψεις μόνο ενός δέντρου είναι ευαίσθητες στο θόρυβο στο σύνολο εκπαίδευσης, ο μέσος όρος πολλών δέντρων δεν είναι, όσο όμως τα δέντρα δε συσχετίζονται. Η εκπαίδευση πολλών δέντρων σε ένα σύνολο εκπαίδευσης μπορεί να να δώσει υψηλά συσχετιζόμενα δέντρα (ακόμα και το ίδιο δέντρο πολλές φορές, αν ο αλγόριθμος εκπαίδευσης δεν είναι ντετερμινιστικός). Η δειγματοληψία bootstrap είναι ένας τρόπος αφαίρεσης της συσχέτισης των δέντρων ,δείχνοντάς τους διαφορετικά σύνολα εκπαίδευσης. Επιπλέον, μία εκτίμηση της αβεβαιότητας της πρόβλεψης μπορεί να γίνει με τη μορφή της τυπικής απόκλισης των προβλέψεων από όλα τα ξεχωριστά δέντρα παλινδρόμησης στο x' (Εξίσωση 3.3)

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}. \tag{3.3}$$

Ο αριθμός των δειγμάτων/δέντρων , είναι μια ελεύθερη παράμετρος. Τυπικά, λίγες εκατοντάδες μέχρι πολλές χιλιάδες δέντρα χρησιμοποιούνται, κάτι που εξαρτάται από το μέγεθος και τη φύση του συνόλου εκπαίδευσης. Ένας βέλτιστος αριθμός δέντρων , μπορεί να βρεθεί χρησιμοποιώντας διασταυρούμενη επικύρωση ή παρατηρώντας το μέσο σφάλμα πρόβλεψης σε κάθε παράδειγμα εκπαίδευσης x_i, χρησιμοποιώντας μόνο τα δέντρα που δεν είχαν x_i στο δείγμα bootstrap τους. Τα σφάλματα εκπαίδευσης και ελέγχου τείνουν να μειώνονται μετά από όταν κάποιος επαρκής αριθμός δέντρων έχει προσαρμοστεί.

3.2.2 Βασικός Αλγόριθμος

Η παραπάνω διαδικασία περιγράφει τον πρωτότυπο αλγόριθμο bagging για δέντρα. Τα RF διαφέρουν σε ένα μόνο πράγμα από τη γενική περίπτωση: χρησιμοποιούν μία τροποποιημένη έκδοση αλγορίθμου δέντρων που επιλέγει, σε κάθε υποψήφιο χώρισμα της διαδικασίας εκπαίδευσης, ένα τυχαίο υποσύνολο των χαρακτηριστικών. Η διαδικασία αυτή κάποιες φορές καλείται bagging χαρακτηριστικών. Ο λόγος που γίνεται αυτό είναι η συσχέτιση των δέντρων σε ένα σύνηθες δείγμα bootstrap: αν ένα ή λίγα από τα χαρακτηριστικά είναι πολύ ισχυρές προβλέψεις της μεταβλητής απόκρισης (στόχου),

τότε αυτά τα χαρακτηριστικά θα επιλεγούν σε πολλά από τα B δέντρα κάνοντάς τα συσχετιζόμενα. Μία ανάλυση του πώς το bagging και η τυχαία προβολή υποχώρου συμβάλλουν στην αύξηση της ακρίβειας κάτω από διάφορες συνθήκες δίνεται στην εργασία [Ho02].

Τυπικά, για ένα πρόβλημα ταξινόμησης με p χαρακτηριστικά, $\sqrt{p}$ (στρογγυλεμένα προς τα κάτω) χαρακτηριστικά χρησιμοποιούνται σε κάθε χώρισμα. Για προβλήματα παλινδρόμησης, προτείνονται $\frac{p}{3}$ (στρογγυλεμένα προς τα κάτω) με ένα ελάχιστο κόμβο μεγέθους 5. Στην πραγματικότητα, οι καλύτερες τιμές για αυτές τις παραμέτρους εξαρτώνται από το πρόβλημα και πρέπει να αντιμετωπίζονται ως μεταβλητές βελτιστοποίησης.

3.3 Λογιστική Παλινδρόμηση

Το *λογιστικό μοντέλο* (logistic ή logit model) χρησιμοποιείται για τη μοντελοποίηση της πιθανότητας μιας συγκεκριμένης δυαδικής κλάσης ή γεγονότος [Toll16]. Αυτό μπορεί να επεκταθεί ώστε να μοντελοποιήσει πολλές κλάσεις γεγονότων όπως ο καθορισμός του αν μια εικόνα περιέχει μια γάτα, έναν σκύλο, ένα λιοντάρι, κλπ. Κάθε αντικείμενο που ανιχνεύεται στην εικόνα αποκτά μια πιθανότητα μεταξύ 0 και 1 και το άθροισμα όλων κάνει 1.

Η λογιστική παλινδρόμηση είναι ένα στατιστικό μοντέλο που στη βασική του μορφή χρησιμοποιεί μια λογιστική συνάρτηση για να μοντελοποιήσει μια δυαδική εξαρτημένη μεταβλητή, παρόλο που υπάρχουν και πιο σύνθετες επεκτάσεις. Στην ανάλυση παλινδρόμησης, η *λογιστική παλινδρόμηση* (logistic ή logit regression) εκτιμάει τις παραμέτρους ενός λογιστικού μοντέλου (μια μορφή δυαδικής παλινδρόμησης). Μαθηματικά, ένα δυαδικό λογιστικό μοντέλο έχει μια εξαρτημένη μεταβλητή με δύο πιθανές τιμές που εκπροσωπούνται από μία συνάρτηση δείκτη, όπου οι δύο τιμές ετικετοποιούνται ως 0 και 1 [Hosm00]. Στο λογιστικό μοντέλο, η συνάρτηση log-odds ($ln[p/(1-p)]$, p πιθανότητα) για την τιμή με την ετικέτα 1 είναι ένας γραμμικός συνδυασμός ενός ή περισσότερων ανεξάρτητων μεταβλητών. Οι ανεξάρτητες αυτές μεταβλητές μπορούν να είναι δυαδικές ή συνεχείς. Η αντίστοιχη πιθανότητα μπορεί να είναι ανάμεσα στο 0 και το 1. Η συνάρτηση που μετατρέπει τη log-odds σε πιθανότητα είναι η λογιστική συνάρτηση, εξού και το όνομα. Η μονάδα μέτρησης της κλίμακας της log-odds καλείται logit, από logistic unit. Ανάλογα μοντέλα με μια διαφορετική σιγμοειδή συνάρτηση μπορούν να χρησιμοποιηθούν, όπως το μοντέλο probit. Το καθοριστικό χαρακτηριστικό του λογιστικού μοντέλου είναι ότι η αύξηση της μίας από τις ανεξάρτητες μεταβλητές επιδρά πολλαπλασιαστικά στις πιθανότητες του δοθέντος αποτελέσματος κατά έναν συνεχή ρυθμό, και αυτό καθώς κάθε ανεξάρτητη μεταβλητή έχει τη δική της παράμετρο. Για μια δυαδική εξαρτημένη μεταβλητή, αυτό γενικεύει το odds ratio, δηλαδή τη δύναμη της σχέσης μεταξύ δύο γεγονότων.

Σε ένα δυαδικό μοντέλο λογιστικής παλινδρόμησης, η εξαρτημένη μεταβλητή έχει δύο κατηγορικά επίπεδα. Έξοδοι με παραπάνω από δύο τιμές μοντελοποιούνται από μια πολυωνυμική λογιστική παλινδρόμηση και αυτό αν οι πολλαπλές κατηγορίες είναι διατεταγμένες. Το μοντέλο λογιστικής παλινδρόμησης από μόνο του μοντελοποιεί την πιθανότητα της εξόδου από την άποψη της εισόδου και δεν πραγματοποιεί στατιστική κατηγοριοποίηση. Δεν αποτελεί ταξινομητή, παρόλο που μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός, για παράδειγμα χρησιμοποιώντας μια τιμή αποκοπής και ταξινομώντας εισόδους με πιθανότητα μεγαλύτερη από την τιμή αποκοπής ως μία κλάση, και στην αντίθετη περίπτωση ως την άλλη κλάση. Αυτό αποτελεί έναν συνήθη τρόπο δημιουργίας ενός δυαδικού ταξινομητή. Οι συντελεστές γενικά δεν υπολογίζονται από μια έκφραση κλειστού τύπου, σε αντίθεση με τα γραμμικά ελάχιστα τετράγωνα.

3.3.1 Λογιστικό Μοντέλο

Ας θεωρήσουμε ένα λογιστικό μοντέλο με δοθείσες παραμέτρους [Bish06] και ας παρατηρήσουμε πώς οι συντελεστές μπορούν να εκτιμηθούν από δεδομένα. Έστω ένα μοντέλο με δύο παράγοντες πρόβλεψης, x_1 και x_2, και μια δυαδική (Bernoulli) μεταβλητή απάντησης Y, η οποία δηλώνεται ως $p = P(Y = 1)$. Υποθέτουμε γραμμική σχέση μεταξύ των μεταβλητών πρόβλεψης και της log-odds του γεγονότος όπου $Y = 1$. Αυτή η γραμμική σχέση μπορεί να γραφτεί στη μορφή της Εξίσωσης 3.4

[Walk67]

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{3.4}$$

όπου l είναι η log-odds, b η βάση του λογαρίθμου και β_i οι παράμετροι του μοντέλου. Μπορούμε να χρησιμοποιήσουμε δύναμη με βάση το e για να απομονώσουμε τις πιθανότητες (Εξίσωση 3.5)

$$\frac{p}{1-p} = b^{\beta_0+\beta_1 x_1+\beta_2 x_2} \tag{3.5}$$

Με απλές πράξεις αποδεικνύεται ότι η πιθανότητα του $Y = 1$ είναι:

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}} \tag{3.6}$$

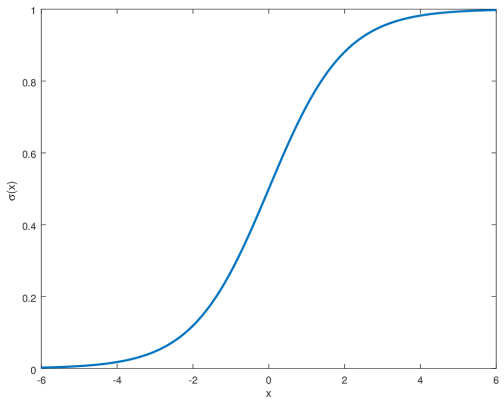
Η Εξίσωση 3.6 δείχνει πως όταν τα β_i γίνουν σταθερά, μπορούμε να υπολογίσουμε εύκολα για την κάθε παρατήρηση την πιθανότητα είτε για Y=1 είτε για Y=0. Η κύρια χρήση του λογιστικού μοντέλου είναι η τροφοδότηση με μια παρατήρηση (x_1, x_2), και η εξαγωγή μιας εκτίμησης με πιθανότητα p όπου $Y = 1$. Στις περισσότερες εφαρμογές, η βάση b του λογαρίθμου είναι συνήθως το e. Ωστόσο, σε κάποιες περιπτώσεις είναι ευκολότερη η αναπαράσταση των αποτελεσμάτων με χρήση της βάσης 2 ή 10.

3.3.2 Λογιστική Συνάρτηση

Η λογιστική συνάρτηση είναι μια σιγμοειδής συνάρτηση που παίρνει μια πραγματική είσοδο t και δίνει ως έξοδο μια τιμή μεταξύ 0 και1. Η τυπική λογιστική συνάρτηση $\sigma : \mathbb{R} \to (0, 1)$ ορίζεται ως εξής (Εξίσωση 3.7):

$$\sigma(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}} \tag{3.7}$$

Μία αναπαράσταση της λογιστικής συνάρτησης στο διάστημα $(-6, 6)$ για την τιμή x, φαίνεται στο Σχήμα 3.3.



Σχήμα 3.3: Λογιστική Συνάρτηση στο διάστημα $(-6, 6)$

Ας υποθέσουμε ότι η t είναι μια γραμμική συνάρτηση μίας επεξηγηματικής μεταβλητής x. Μπορούμε να εκφράσουμε το t ως εξής (Εξίσωση 3.8):

$$t = \beta_0 + \beta_1 x \tag{3.8}$$

Και η γενική λογιστική συνάρτηση $p : R \to (0, 1)$ μπορεί να γραφτεί ως εξής (Εξίσωση 3.9):

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}} \tag{3.9}$$

Στο λογιστικό μοντέλο,η πιθανότητα $p(x)$ ερμηνεύεται ως η πιθανότητα η εξαρτημένη μεταβλητή να ισούται με μια επιτυχία και όχι μια αποτυχία. Είναι ξεκάθαρο πως οι μεταβλητές απόκρισης Y_i δεν είναι ομοιόμορφα κατανεμημένες. Η πιθανότητα $P(Y_i = 1 \mid X)$ διαφέρει από ένα σημείο X_i των δεδομένων σε ένα άλλο, παρόλο που είναι ανεξάρτητες, δοθέντος του πίνακα σχεδιασμού και των κοινών παραμέτρων β.

3.3.3 Η Συνάρτηση logit

Η συνάρτηση logit [Bish06] ορίζεται ως η ανάστροφη

$$g = \sigma^{-1}$$

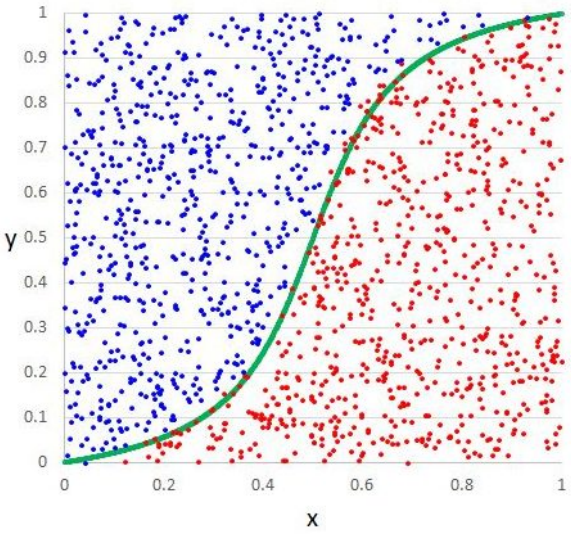
της τυπικής λογιστικής συνάρτησης. Είναι εύκολο να δούμε πως ικανοποιεί την Εξίσωση 3.10

$$g(p(x)) = \sigma^{-1}(p(x)) = \operatorname{logit} p(x) = \ln\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x \tag{3.10}$$

και αντίστοιχα, μετά από ύψωση στη δύναμη του e και στις δύο πλευρές, παίρνουμε τις πιθανότητες (Εξίσωση 3.11):

$$\frac{p(x)}{1-p(x)} = e^{\beta_0+\beta_1 x} \tag{3.11}$$

Το Σχήμα 3.4 απεικονίζει ένα παράδειγμα χρήσης της logit σε πρόβλημα ταξινόμησης.



Σχήμα 3.4: Συνάρτηση logit σε ταξινόμηση

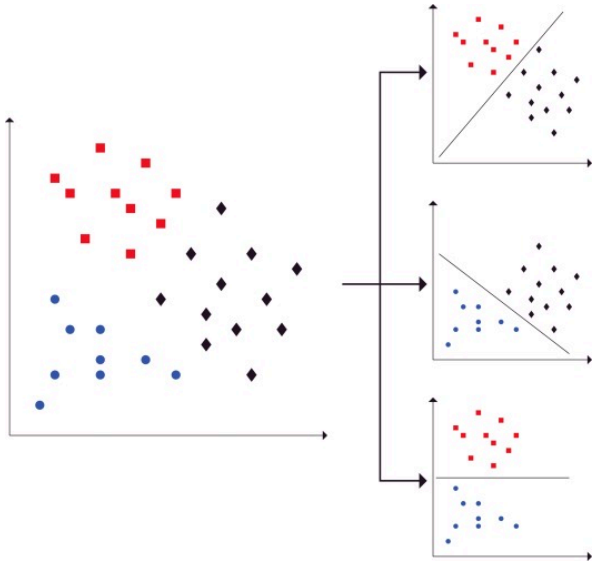
3.3.4 Ένας εναντίον των υπολοίπων

Η τεχνική παλινδρόμησης *ενός εναντίον των υπολοίπων* (one-vs-rest regression) περιλαμβάνει [Ripl96] την εκπαίδευση ενός ταξινομητή ανά κλάση, με δείγματα αυτής της κλάσης ως θετικά και όλων των άλλων ως αρνητικά δείγματα. Απαιτεί ο βασικός ταξινομητής να παράγει μια πραγματική τιμή αποτελέσματος εμπιστοσύνης, παρά απλά μια ετικέτα κλάσης. Ετικέτες διακριτών κλάσεων μπορούν από μόνες τους να οδηγήσουν σε αμφισημίες, όπου πολλαπλές κλάσεις προβλέπονται για ένα μόνο δείγμα. Ο αλγόριθμος εκπαίδευσης για έναν αλγόριθμο 1vR κατασκευασμένο από έναν αλγόριθμο δυαδικής ταξινόμησης L είναι ο ακόλουθος:

Λήψη αποφάσεων σημαίνει η εφαρμογή όλων των ταξινομητών σε ένα άγνωστο δείγμα x και η πρόβλεψη της ετικέτας k, για την οποία ο αντίστοιχος ταξινομητής αποδίδει το μεγαλύτερο αποτέλεσμα εμπιστοσύνης (Εξίσωση 3.12):

$$\hat{y} = \arg \max_{k \in \{1 \ldots K\}} f_k(x) \tag{3.12}$$

Παρόλο που η στρατηγική αυτή είναι δημοφιλής, είναι μια ευρετική μέθοδος που έχει προβλήματα. Αρχικά, η κλίμακα των τιμών εμπιστοσύνης μπορεί να διαφέρει ανά δυαδικό ταξινομητή. Δεύτερον, ακόμα και αν η κατανομή της κλάσης είναι ισορροπημένη στο σύνολο εκπαίδευσης, οι δυαδικοί ταξινομητές μπορεί να βλέπουν μη-ισορροπημένες κατανομές, επειδή τυπικά το σύνολο των αρνητικών που βλέπουν είναι πολύ μεγαλύτερο από το σύνολο των θετικών. Το Σχήμα 3.5 περιέχει ένα παράδειγμα παλινδρόμησης ενός εναντίων των υπολοίπων για 3 κλάσεις.



Σχήμα 3.5: Λογιστική Παλινδρόμηση ενός εναντίων των υπολοίπων για 3 κλάσεις

3.4 Πολυωνυμική Λογιστική Παλινδρόμηση

Η *πολυωνυμική λογιστική παλινδρόμηση* (multinomial logistic regression - MLR) είναι μια μέθοδος ταξινόμησης που γενικεύει τη λογιστική παλινδρόμηση σε προβλήματα πολλών κλάσεων, για παράδειγμα με παραπάνω από δύο πιθανά διακριτά αποτελέσματα. Είναι ένα μοντέλο που χρησιμοποιείται για την πρόβλεψη των πιθανοτήτων των διαφορετικών πιθανών αποτελεσμάτων μιας κατηγορικά κατανεμημένης εξαρτημένης μεταβλητής, δοθέντος ενός συνόλου ανεξάρτητων μεταβλητών, οι οποίες μπορούν να έχουν πραγματικές, δυαδικές, κατηγορικές και άλλες τιμές.

3.4.1 Προϋποθέσεις

Η MLR προϋποθέτει [Duda00] πως τα δεδομένα είναι συγκεκριμένα για κάθε περίπτωση. Κάθε ανεξάρτητη μεταβλητή έχει μια συγκεκριμένη τιμή για κάθε περίπτωση. Το μοντέλο, επίσης, προϋποθέτει ότι η εξαρτημένη μεταβλητή δε μπορεί σε καμία περίπτωση να προβλεφθεί τέλεια από τις ανεξάρτητες μεταβλητές. Όπως και με άλλους τύπους παλινδρόμησης, δεν υπάρχει η ανάγκη για τις ανεξάρτητες μεταβλητές να είναι στατιστικά ανεξάρτητες μεταξύ τους. Ωστόσο, η συγγραμικότητα θεωρείται σχετικά χαμηλή, καθώς γίνεται δύσκολη η διαφοροποίηση μεταξύ της επίδρασης πολλών μεταβλητών, αν συμβαίνει το αντίθετο.

Όταν η πολυωνυμική συνάρτηση logit χρησιμοποιείται για τη μοντελοποίηση επιλογών, βασίζεται στην *ανεξαρτησία των μη-σχετικών εναλλακτικών* (independence of irrelevant alternatives - IIA) που δεν είναι πάντα επιθυμητή. Αυτό δηλώνει πως οι πιθανότητες της επιλογής μιας κλάσης από μια άλλη δεν εξαρτώνται από την ύπαρξη ή την απουσία άλλων άσχετων εναλλακτικών. Για παράδειγμα,

οι σχετικές πιθανότητες της επιλογής αυτοκινήτου ή λεωφορείου για τη μετακίνηση στη δουλειά δεν αλλάζουν αν ένα ποδήλατο προστεθεί ως μια επιπλέον δυνατότητα. Αυτό επιτρέπει τη μοντελοποίηση K επιλεγμένων εναλλακτικών ως ένα σύνολο $K-1$ ανεξάρτητων δυαδικών επιλογών, όπου σε κάθε από τις ανεξάρτητες επιλέγεται ως σταθερή και οι άλλες $K-1$ συγκρίνονται με αυτήν, μία κάθε φορά. Η υπόθεση IIA είναι μια βασική υπόθεση στη θεωρία λογικής επιλογής, όμως πολλές μελέτες στην ψυχολογία δείχνουν πως οι άνθρωποι συχνά παραβιάζουν αυτή την προϋπόθεση όταν κάνουν επιλογές.

Αν η πολυωνυμική logit χρησιμοποιηθεί για τη μοντελοποίηση επιλογών, μπορεί σε κάποιες περιπτώσεις να προκαλέσει πάρα πολλή συγκράτηση των σχετικών επιλογών μεταξύ διαφορετικών εναλλακτικών. Αυτό το σημείο είναι εξαιρετικά σημαντικό αν η ανάλυση στοχεύει στην πρόβλεψη του πώς οι επιλογές θα άλλαζαν αν μια εναλλακτική εξαφανιζόταν. Άλλα μοντέλα όπως η εμφωλευμένη logit ή η πολυωνυμική probit μπορούν να χρησιμοποιηθούν σε ανάλογες περιπτώσεις καθώς επιτρέπουν την παραβίαση του IIA.

3.4.2 Μοντέλο

Η διαδικασία που ακολουθεί το μοντέλο είναι η ίδια με της λογιστικής παλινδρόμησης, με τη μόνη διαφορά να αποτελεί το γεγονός πως οι εξαρτημένες μεταβλητές είναι κατηγορικές και όχι δυαδικές. Για παράδειγμα, υπάρχουν K πιθανά αποτελέσματα και όχι μόνο δύο.

Γραμμική Πρόβλεψη

Όπως και σε άλλες μορφές γραμμικής παλινδρόμησης, η MLR [Bish06] χρησιμοποιεί μια γραμμική συνάρτηση πρόβλεψης $f(k,i)$ για να προβλέψει την πιθανότητα η παρατήρηση i να έχει αποτέλεσμα k, της μορφής της Εξίσωσης 3.13

$$f(k,i) = \beta_{0,k} + \beta_{1,k}x_{1,i} + \beta_{2,k}x_{2,i} + \cdots + \beta_{M,k}x_{M,i} \tag{3.13}$$

όπου $\beta_{m,k}$ είναι ένας συντελεστής της παλινδρόμησης που σχετίζεται με την m-οστή επεξηγηματική μεταβλητή και το k-οστό αποτέλεσμα. Οι συντελεστές και οι επεξηγηματικές μεταβλητές μαζεύονται σε διανύσματα μεγέθους $M+1$, ώστε η συνάρτηση πρόβλεψης να μπορεί να γραφτεί στην πιο συνεπτυγμένη μορφή της Εξίσωσης 3.14

$$f(k,i) = \beta_k \cdot \mathbf{x}_i \tag{3.14}$$

όπου $\boldsymbol{\beta}_k$ είναι το σύνολο των συντελεστών που σχετίζονται με το αποτέλεσμα k, και x_i (διάνυσμα γραμμή) είναι το σύνολο των επεξηγηματικών μεταβλητών που σχετίζονται με την παρατήρηση i.

Σύνολο ανεξάρτητων δυαδικών παλινδρομήσεων

Για να φτάσουμε στο πολυωνυμικό μοντέλο logit, πρέπει να σκεφτούμε πως για πιθανά αποτελέσματα, τρέχοντας $K-1$ ανεξάρτητα μοντέλα λογιστικής παλινδρόμησης, από τα οποία ένα αποτέλεσμα επιλέγεται ως σταθερό, τα άλλα $K-1$ αποτελέσματα παλινδρομούνται κόντρα στο σταθερό αποτέλεσμα. Αν το αποτέλεσμα K (το τελευταίο αποτέλεσμα) επιλεγεί ως το σταθερό, τότε

$$\begin{aligned}
\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} &= \boldsymbol{\beta}_1 \cdot \mathbf{X}_i \\
\ln \frac{\Pr(Y_i = 2)}{\Pr(Y_i = K)} &= \boldsymbol{\beta}_2 \cdot \mathbf{X}_i \\
&\cdots\cdots \\
\ln \frac{\Pr(Y_i = K-1)}{\Pr(Y_i = K)} &= \boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i
\end{aligned}$$

Ορίσαμε διαφορετικά σύνολα σταθερών παλινδρόμησης, ένα για κάθε δυνατό αποτέλεσμα. Αν υψώσουμε στη δύναμη του e και τις δύο πλευρές της κάθε εξίσωσης και λύσουμε ως προς τις πιθανότητες:

$$\begin{aligned}
\Pr(Y_i = 1) &= \Pr(Y_i = K) e^{\boldsymbol{\beta}_1 \cdot X_i} \\
\Pr(Y_i = 2) &= \Pr(Y_i = K) e^{\beta_2 \cdot X_i} \\
&\cdots\cdots \\
\Pr(Y_i = K-1) &= \Pr(Y_i = K) e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i}
\end{aligned}$$

Χρησιμοποιώντας το γεγονός πως το άθροισμα όλων των K πιθανοτήτων πρέπει να κάνει 1:

$$\Pr(Y_i = K) = 1 - \sum_{k=1}^{K-1} \Pr(Y_i = k) = 1 - \sum_{k=1}^{K-1} \Pr(Y_i = K) e^{\boldsymbol{\beta}_k \cdot X_i} \Rightarrow \Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot X_i}}$$

Μπορούμε να χρησιμοποιήσουμε το παραπάνω για να βρούμε τις άλλες πιθανότητες:

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot X_i}}$$

$$\cdots\cdots$$

$$\Pr(Y_i = K-1) = \frac{e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

Το γεγονός πως εκτελέσαμε πολλαπλές παλινδρομήσεις αποδεικνύει το λόγο για τον οποίο υποθέσαμε την ανεξαρτησία των άσχετων εναλλακτικών.

Log-γραμμικό μοντέλο

Η δυαδική λογιστική παλινδρόμηση ως ένα log-γραμμικό μοντέλο μπορεί να επεκταθεί σε ένα πολλαπλό μοντέλο. Μοντελοποιώντας τον λογάριθμο της πιθανότητας να δούμε μία δοθείσα έξοδο χρησιμοποιώντας τον γραμμικό ταξινομητή και έναν παράγοντα κανονικοποίησης, ο λαγάριθμος της συνάρτησης χωρισμού γίνεται:

$$\begin{aligned}
\ln \Pr(Y_i = 1) &= \boldsymbol{\beta}_1 \cdot \mathbf{X}_i - \ln Z \\
\ln \Pr(Y_i = 2) &= \boldsymbol{\beta}_2 \cdot \mathbf{X}_i - \ln Z \\
&\cdots\cdots \\
\ln \Pr(Y_i = K) &= \boldsymbol{\beta}_K \cdot \mathbf{X}_i - \ln Z
\end{aligned}$$

Όπως και στη δυαδική περίπτωση, χρειαζόμαστε έναν επιπλέον όρο $-\ln Z$ για να διασφαλίσουμε ότι όλες οι πιθανότητες σχηματίζουν μια κατανομή πιθανότητας, ώστε να έχουν άθροισμα 1:

$$\sum_{k=1}^{K} \Pr(Y_i = k) = 1$$

Ο λόγος που χρειάζεται να προσθέσουμε έναν όρο για να εξασφαλίσουμε την κανονικοποίηση, πέραν από τον συνήθη πολλαπλασιασμό, είναι γιατί έχουμε πάρει το λογάριθμο των πιθανοτήτων. Υψώνοντας στη δύναμη του e κάθε μέλος των ισοτήτων, μετατρέπουμε το προσθετικό μέρος σε πολλαπλασιαστικό και η πιθανότητα γίνεται το μέτρο του Gibbs:

$$\begin{aligned}
\Pr(Y_i = 1) &= \frac{1}{Z} e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i} \\
\Pr(Y_i = 2) &= \frac{1}{Z} e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i} \\
&\cdots\cdots \\
\Pr(Y_i = K) &= \frac{1}{Z} e^{\boldsymbol{\beta}_K \cdot \mathbf{X}_i}
\end{aligned}$$

Η ποσότητα Z καλείται η συνάρτηση κατάτμησης για την κατανομή. Μπορούμε να υπολογίσουμε την τιμή αυτής της συνάρτησης εφαρμόζοντας τον παραπάνω περιορισμό που απαιτεί όλες οι πιθανότητες να έχουν άθροισμα 1:

$$1 = \sum_{k=1}^{K} \Pr(Y_i = k) = \sum_{k=1}^{K} \frac{1}{Z} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i} = \frac{1}{Z} \sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}$$

Οπότε:

$$Z = \sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}$$

Ο συντελεστής είναι μια σταθερά με την έννοια ότι δεν είναι μια συνάρτηση του Y_i που είναι η μεταβλητή χάρη στην οποία ορίζεται η κατανομή πιθανότητας. Ωστόσο, δεν είναι μεταβλητή σε σχέση με τις επεξηγηματικές μεταβλητές ή τους άγνωστους συντελεστές β_k της παλινδρόμησης, οι οποίοι θα πρέπει να προσδιοριστούν μέσω μιας διαδικασίας βελτιστοποίησης.

Οι τελικές εξισώσεις των πιθανοτήτων είναι:

$$\begin{aligned}
\Pr(Y_i = 1) &= \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}} \\
\Pr(Y_i = 2) &= \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}} \\
&\cdots\cdots \\
\Pr(Y_i = K) &= \frac{e^{\boldsymbol{\beta}_K \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}
\end{aligned}$$

ή πιο γενικά:

$$\Pr(Y_i = c) = \frac{e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

Η συνάρτηση της Εξίσωσης 3.15

$$\text{softmax}(k, x_1, \ldots, x_n) = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}} \tag{3.15}$$

αποκαλείται συνάρτηση softmax. Ο λόγος είναι γιατί η εκθετοποίηση των μεταβλητών $x_1, \ldots, x_n$ μεγαλοποιεί τις διαφορές μεταξύ τους. Ως αποτέλεσμα, η $\text{softmax}(k, x_1, \ldots, x_n)$ θα επιστρέψει μια τιμή κοντά στο 0 όταν το x_k είναι αρκετά μικρότερο από το μέγιστο όλων των τιμών, και θα επιστρέψει 1 όταν εφαρμοστεί στη μέγιστη τιμή, εκτός και αν είναι πολύ κοντά στην επόμενη μεγαλύτερη τιμή.

Η συνάρτηση αυτή μπορεί να χρησιμοποιηθεί για την κατασκευή ενός βεβαρημένου μέσου όρου που συμπεριφέρεται ως μια ομαλή συνάρτηση (που μπορεί να διαφοριστεί εύκολα), και η οποία εκτιμά τη δείκτρια συνάρτηση (Εξίσωση 3.16)

$$f(k) = \left\{ \begin{array}{ll} 1, & k = \arg\max(x_1, \ldots, x_n) \\ 0, & otherwise \end{array} \right\} \tag{3.16}$$

Έτσι, μπορούμε να γράψουμε τις εξισώσεις πιθανοτήτων όπως στην Εξίσωση3.17 παρακάτω

$$\Pr(Y_i = c) = \text{softmax}(c, \boldsymbol{\beta}_1 \cdot \mathbf{X}_i, \ldots, \boldsymbol{\beta}_K \cdot \mathbf{X}_i) \tag{3.17}$$

Η συνάρτηση αυτή αποτελεί την αντίστοιχη της λογιστικής συνάρτησης στη δυαδική λογιστική παλινδρόμηση.

Αξιοσημείωτο κρίνεται πως όχι όλα τα διανύσματα συντελεστών $\boldsymbol{\beta}_k$ είναι μοναδικά αναγνωρίσιμα. Αυτό έχει σχέση με το ότι όλες οι πιθανότητες πρέπει να αθροίζουν στο 1, κάνοντας μία από αυτές πλήρως αποφασισμένη, ενώ όλες οι άλλες είναι άγνωστες. Ως αποτέλεσμα, υπάρχουν μόνο $k-1$ ξεχωριστά διαφοροποιήσιμες πιθανότητες, και για αυτό υπάρχουν $k-1$ ξεχωριστά διακεκριμένα διανύσματα συντελεστών. Ένας τρόπος για να το δούμε αυτό φαίνεται από το ότι αν προσθέσουμε ένα σταθερό διάνυσμα σε όλα διανύσματα συντελεστών, τότε οι εξισώσεις είναι πανομοιότυπες:

$$\frac{e^{(\boldsymbol{\beta}_c + C)\cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{(\boldsymbol{\beta}_k + C)\cdot \mathbf{X}_i}} = \frac{e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i} e^{C \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i} e^{C \cdot \mathbf{X}_i}} = \frac{e^{C \cdot \mathbf{X}_i} e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i}}{e^{C \cdot \mathbf{X}_i} \sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}} = \frac{e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

Ως αποτέλεσμα, είναι σύνηθες να θέσουμε $C = -\boldsymbol{\beta}_K$ (ή κάποιον άλλο από το διάνυσμα των συντελεστών). Ουσιαστικά, θέτουμε τη μεταβλητή ώστε ένα από τα διανύσματα να γίνει 0 και όλα τα άλλα να μετασχηματιστούν στη διαφορά μεταξύ των διανυσμάτων αυτών και των διανυσμάτων που επιλέγουμε.

Μαθηματικά, οι συντελεστές μετασχηματίζονται ως εξής:

$$\begin{aligned} \boldsymbol{\beta}'_1 &= \boldsymbol{\beta}_1 - \boldsymbol{\beta}_K \\ &\ldots\ldots \\ \boldsymbol{\beta}'_{K-1} &= \boldsymbol{\beta}_{K-1} - \boldsymbol{\beta}_K \\ \boldsymbol{\beta}'_K &= 0. \end{aligned}$$

Αυτό οδηγεί στις ακόλουθες εξισώσεις:

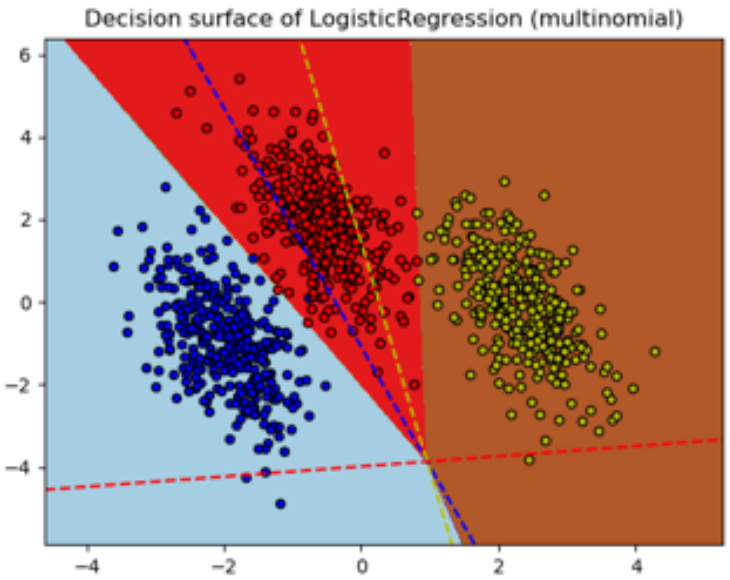
$$\begin{aligned} \Pr(Y_i = 1) &= \frac{e^{\boldsymbol{\beta}'_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}'_k \cdot \mathbf{X}_i}} \\ &\ldots\ldots \\ \Pr(Y_i = K-1) &= \frac{e^{\boldsymbol{\beta}'_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}'_k \cdot \mathbf{X}_i}} \\ \Pr(Y_i = K) &= \frac{1}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}'_k \cdot \mathbf{X}_i}}. \end{aligned}$$

Elastic Net

Η Elastic Net αποτελεί μια παράμετρο του αλγορίθμου η οποία χρησιμοποιείται για τη διασφάλιση της *υπερπροσαρμογής* (overfitting) του μοντέλου στα δεδομένα εκμάθησης. Αποτελεί τον όρο λ στην Εξίσωση 3.18

$$\min_{\beta, \beta_0} - \left[\sum_{i=1}^{L} w_i \cdot \log P(Y = y_i | \mathbf{x}_i) \right] + \lambda \left[\frac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1 \right] \tag{3.18}$$

η οποία εξίσωση με τη σειρά της εκφράζει την ποινή για όταν συμβαίνει κάποια αστοχία. Στο Σχήμα 3.6 παρουσιάζεται ένα παράδειγμα MLP ταξινόμησης ενός χώρου δεδομένων που χαρακτηρίζεται από 3 κλάσεις.



Σχήμα 3.6: Πολυωνυμική λογιστική παλινδρόμηση για 3 κλάσεις

3.5 Πολυεπίπεδο Perceptron

Τα *πολυεπίπεδα perceptron* (multilayer perceptron - MLP) αποτελούνται [Ripl96] από πολλαπλά επίπεδα υπολογιστικών μονάδων, διασυνδεδεμένα κατά έναν τρόπο πρόσθιας τροφοδότησης. Κάθε νευρώνας σε ένα επίπεδο έχει κατευθυνόμενες συνδέσεις προς τους νευρώνες του επόμενου επιπέδου. Σε πολλές εφαρμογές οι νευρώνες αυτών των δικτύων χαρακτηρίζονται από μια σιγμοειδή συνάρτηση ενεργοποίησης.

Το θεώρημα καθολικής προσέγγισης για νευρωνικά δίκτυα δηλώνει πως κάθε συνεχής συνάρτηση που αντιστοιχίζει διαστήματα πραγματικών αριθμών σε κάποιο διάστημα εξόδου πραγματικών αριθμών, μπορεί να προσεγγιστεί αυθαίρετα από ένα πολυστρωματικό perceptron με ένα μόνο κρυμμένο επίπεδο.

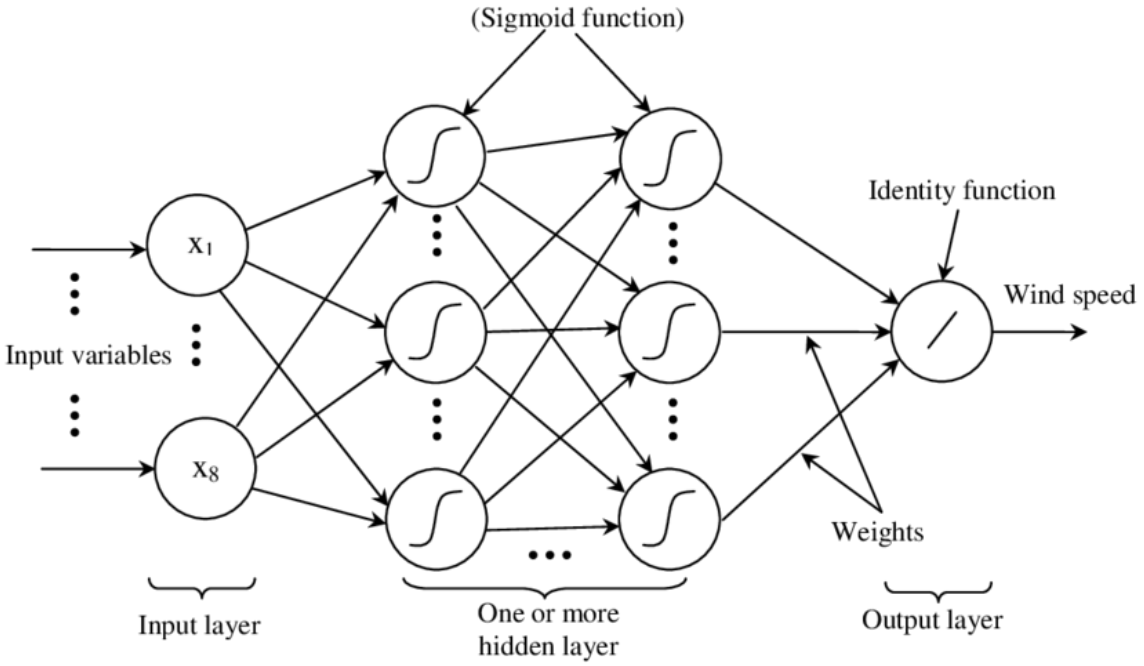
Το Σχήμα 3.7 αναπαριστά ένα πολυεπίπεδο perceptron και για την ακρίβεια ενός με τουλάχιστον τρία επίπεδα (ένα επίπεδο εισόδου, ένα εξόδου και τουλάχιστον ένα κρυμμένο επίπεδο).

3.5.1 Συνάρτηση Ενεργοποίησης

Αν ένα MLP έχει μια γραμμική συνάρτηση ενεργοποίησης σε όλους τους νευρώνες που αντιστοιχίζει τις εισόδους με βάρη στις εξόδους κάθε νευρώνα, τότε από την γραμμική άλγεβρα ξέρουμε ότι κάθε αριθμός επιπέδων μπορεί να μειωθεί και να φτάσουμε σε ένα μοντέλο εισόδου-εξόδου με δύο στρώματα. Στα MLP, ωστόσο, μπορούν να χρησιμοποιηθούν και μη-γραμμικές συναρτήσεις ενεργοποίησης

$$y(v_i) = \tanh(v_i) \quad \text{και} \quad y(v_i) = (1 + e^{-v_i})^{-1} \tag{3.19}$$

Η πρώτη συνάρτηση που απεικονίζεται στην Εξίσωση 3.19 είναι η υπερβολική εφαπτομένη [Ripl96] με τιμές μεταξύ -1 και 1, ενώ η άλλη είναι η λογιστική συνάρτηση που έχει ανάλογη μορφή, αλλά



Σχήμα 3.7: Γενικό παράδειγμα Multilayer Perceptron

τιμές μεταξύ 0 και 1. Εδώ το y_i είναι η έξοδος του i-οστού κόμβου (νευρώνα) και το v_i είναι η εισόδος του (το άθροισμα με τα βάρη που προέρχονται από το προηγούμενο επίπεδο)

Στις σύγχρονες εφαρμογές της βαθιάς μάθησης παρατηρείται μεγαλύτερη χρήση της συνάρτησης ReLU και αυτό ως ένα μέτρο αντιμετώπισης των αριθμητικών προβλημάτων που σχετίζονται με τις σιγμοειδείς. Έχουν προταθεί και εναλλακτικές συναρτήσεις ενεργοποίησης, όπως οι συναρτήσεις rectifier και softplus. Ακόμα πιο εξειδικευμένες συναρτήσεις αποτελούν οι συναρτήσεις ακτινικής βάσης.

3.5.2 Επίπεδα και Εκπαίδευση

Ένα MLP αποτελείται από τρία ή περισσότερα επίπεδα (ένα επίπεδο εισόδου και ένα επίπεδο εξόδου με ένα ή περισσότερα κρυμμένα επίπεδα) μη-γραμμικώς ενεργοποιημένων κόμβων. Αφού τα MLP είναι εσωτερικά διασυνδεδεμένα, κάθε κόμβος σε ένα επίπεδο συνδέεται με ένα συγκεκριμένο βάρος w_{ij} σε κάθε κόμβο του επόμενου επιπέδου.

Η εκπαίδευση γίνεται με τη μεταβολή των βαρών, αφού προηγουμένος έχει παρασχεθεί στο δίκτυο ένα δείγμα των δεδομένων εισόδων μαζί με την πραγματική κλάση του. Η εκπαίδευση βασίζεται στο σφάλμα της εξόδου συγκριτικά με το αναμενόμενο αποτέλεσμα. Αυτό αποτελεί ένα παράδειγμα μάθησης με επίβλεψη και πραγματοποιείται με μια μέθοδο αποκαλούμενη προς τα πίσω διάδοσης του σφάλματος (backpropagation of error), η οποία είναι μια γενίκευση του αλγορίθμου των ελαχίστων τετραγώνων στο γραμμικό perceptron.

Μπορούμε να παρουσιάσουμε το βαθμό του σφάλματος ενός κόμβου j στο n-οστό παράδειγμα εκπαίδευσης με την Εξίσωση 3.20

$$e_j(n) = d_j(n) - y_j(n) \tag{3.20}$$

όπου το d είναι η τιμή στόχος και y είναι η έξοδος του δικτύου. Τα βάρη των κόμβων μπορούν να προσαρμοστούν βάσει των συνδέσεων που ελαχιστοποιούν το σφάλμα σε όλο το σύνολο της εξόδου και δίνεται από την Εξίσωση 3.21

$$\mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n) \tag{3.21}$$

Για την ενημέρωση των βαρών χρησιμοποιούμε την μέθοδο της *κατάβασης κλίσης* (gradient descent), για την οποία υπολογίζουμε την παράγωγο της ενέργειας ως προς την είσοδο (τα βάρη) των νευρώνων. Για έναν κόμβο εξόδου, η ποσότητα αυτή παίρνει τη μορφή της Εξίσωσης 3.22

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n)) \tag{3.22}$$

όπου ϕ' είναι η παράγωγος της συνάρτησης ενεργοποίησης.

Στην περίπτωση των νευρώνων των κρυφών επιπέδων, η μερική παράγωγος γίνεται (Εξίσωση 3.23)

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n) \tag{3.23}$$

Αυτό εξαρτάται από την αλλαγή των βαρών των k-οστών κόμβων, οι οποίοι αντιπροσωπεύουν το στρώμα εξόδου. Έτσι, για να αλλάξουμε τα βάρη των κρυφών στρωμάτων, τα βάρη του στρώματος εξόδου αλλάζουν ανάλογα και σύμφωνα με την παράγωγο της συνάρτησης ενεργοποίησης.

Κεφάλαιο 4

Πειραματική Διαδικασία

4.1 Το πρόβλημα

Τα επιταχυνσιόμετρα αποτελούν αισθητήρες που μετρούν την απόλυτη επιτάχυνση ενός αντικειμένου σε σχέση με την κίνηση και τη βαρυτική δύναμη. Οι σύγχρονες φορητές συσκευές είναι εξοπλισμένες με μικρά κυκλώματα επιταχυνσιόμετρων, τα οποία παράγονται από ηλεκτρομηχανικά ευαίσθητα στοιχεία και παράγουν ηλεκτρικά σήματα κάθε φορά που παρατηρούν μια μηχανική κίνηση. Η απόλυτη επιτάχυνση διαφέρει από την συντεταγμένη επιτάχυνση της κλασικής μηχανικής. Η δεύτερη μετράει το ρυθμό αλλαγής της ταχύτητας, ενώ η πρώτη μετράει την επιτάχυνση σε σχέση με μια ελεύθερη πτώση. Ως γνωστόν, η απόλυτη επιτάχυνση ενός αντικειμένου σε μία ελεύθερη πτώση είναι μηδέν.

Ας σκεφτούμε μια φορητή συσκευή με έναν ενσωματωμένο αισθητήρα επιταχυνσιόμετρου που παράγει δείγματα επιτάχυνσης. Η αναγνώριση δραστηριότητας εφαρμόζεται σε ένα πλαίσιο δεδομένων χρονικής σειράς χρησιμοποιώντας ένα συρόμενο και επικαλυπτόμενο παράθυρο. Ο αριθμός των δειγμάτων χρονικής σειράς εξαρτάται από τη συχνότητα δειγματοληψίας του επιταχυνσιόμετρου και από το χρονικό μήκος του παραθύρου. Στη χρονική στιγμή t, ο ταξινομητής αναγνώρισης κίνησης με συνάρτηση $f : x_t \rightarrow S$ αντιστοιχίζει τα δεδομένα επιτάχυνσης x_t που βρίσκονται στο πλαίσιο δεδομένων με την πιο πιθανή ετικέτα δραστηριότητας από ένα σύνολο υποστηριζόμενων ετικετών δραστηριοτήτων $S = 1, 2, \ldots, N$, όπου N είναι ο αριθμός των υποστηριζόμενων δραστηριοτήτων σε ένα συστατικό ανίχνευσης.

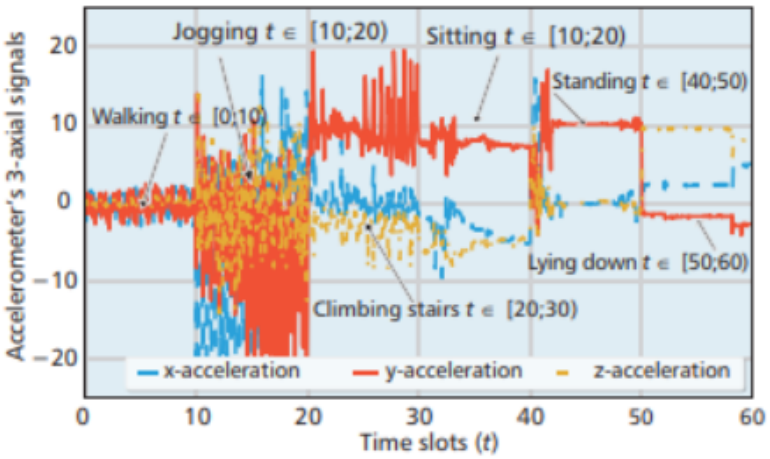
Συμβατικές προσεγγίσεις στην αναγνώριση δραστηριότητας απαιτούν την κατασκευή με το χέρι των χαρακτηριστικών (π.χ. στατιστικά χαρακτηριστικά), τα οποία είναι ακριβά στο σχεδιασμό, απαιτούν τέλεια γνώση τομέα και γενικεύουν σε κακό επίπεδο για την υποστήριξη περισσότερων δραστηριοτήτων. Για να το αποφύγει αυτό, ένα μοντέλο (μηχανικής ή/και) βαθιάς μάθησης για αναγνώριση δραστηριότητας μαθαίνει όχι μόνο μια αντιστοίχιση μεταξύ ακατέργαστων δεδομένων επιτάχυνσης και αντίστοιχης ετικέτας δραστηριότητας, αλλά και ένα σύνολο χρήσιμων χαρακτηριστικών που είναι ανώτερα από τα χαρακτηριστικά που κατασκευάζονται με το χέρι.

4.2 Η συλλογή δεδομένων

Χρησιμοποιήσαμε τη συλλογή δεδομένων Actitracker [Kwap11], το οποίο περιλαμβάνει δείγματα επιταχυνσιόμετρων 6 συνήθων δραστηριοτήτων (περπάτημα, τζόκινγκ, ανέβασμα σκάλας, καθήμενος, όρθιος και ξαπλωμένος) από 563 χρήστες. Το Σχήμα 4.1 απεικονίζει ένα παράδειγμα της απόκρισης του επιταχυνσιομέτρου για κάποιες από τις εξεταζόμενες δραστηριότητες.

Είναι εμφανές πως σήματα υψηλής συχνότητας δειγματοληπτούνται για δραστηριότητες με ενεργή κίνηση του σώματος (π.χ. περπάτημα, τζόκινγκ, ανέβασμα σκάλας). Από την άλλη, σήματα χαμηλής συχνότητας μαζεύονται κατά τη διάρκεια ημι-στατικών κινήσεων του σώματος (καθήμενος, όρθιος και ξαπλωμένος).

Τα δεδομένα συλλέχθηκαν χρησιμοποιώντας κινητά τηλέφωνα με συχνότητα δειγματοληψίας ίση με 20 Hertz και περιλαμβάνουν τόσο ετικετοποιημένα όσο και μη-ετικετοποιημένα δεδομένα 2.980.765 και 38.209.772 δειγμάτων αντίστοιχα. Αυτό αποτελεί ένα ρεαλιστικό παράδειγμα που δεί-



Σχήμα 4.1: Σήμα επιταχυνσιόμετρου για διαφορετικές δραστηριότητες

Πίνακας 4.1: Χαρακτηριστικά του συνόλου δεδομένων Actitracker [Kwap11]

Αριθμός δειγμάτων	2.980.765
Αριθμός χρηστών	563
Αριθμός χαρακτηριστικών	6
Απουσιάζουσες τιμές	Όχι

Πίνακας 4.2: Κατανομή κλάσεων

Κλάση	**Πλήθος δειγμάτων**	**Ποσοστό**
Περπάτημα	1,255,923	42.1%
Τζόκινγκ	438,871	14.7%
Ανέβασμα σκάλας	57,425	1.9%
Καθήμενος	663,706	22.3%
Όρθιος	288,873	9.7%
Ξαπλωμένος	275,967	9.3%

χνει τον περιορισμένο αριθμό δεδομένων με ετικέτα σε σχέση με τα δεδομένα χωρίς ετικέτα, αφού τα πρώτα απαιτούν την επέμβαση του ανθρώπινου παράγοντα. Τα δεδομένα πλαισιώθηκαν χρησιμοποιώντας μια συνάρτηση παραθύρου 10 δευτερολέπτων που παράγει 200 δείγματα χρονοσειρών.

Ο Πίνακας 4.1 συνοψίζει τα χαρακτηριστικά του συνόλου δεδομένων ενώ ο Πίνακας 4.2 συνοψίζει την κατανομή των δειγμάτων ανά δραστηριότητα (κλάση).

4.3 Διαδικασία

Μιας και το αντικείμενο της παρούσας διπλωματικής εργασίας ήταν η μελέτη αλγορίθμων επιβλεπόμενης μάθησης, χρησιμοποιήσαμε μόνο το κομμάτι των δεδομένων που είχαν ετικέτα (Πίνακας 4.1).

Αρχικά, προσπαθήσαμε να μετατρέψουμε τα δεδομένα σε μορφή *libsvm* [Chan11], δηλαδή στη μορφή:

```
<label>  <index1>:<value1>  <index2>:<value2>  …
.
.
.
```

Listing 4.1: Δομή αρχείου libsvm

Κάθε γραμμή περιέχει ένα δείγμα και τελειώνει με ένα χαρακτήρα νέας γραμμής (‘\n’). Για κατηγοριοποίηση, το `<label>` είναι ένας ακέραιος που δείχνει την ετικέτα της κλάσης (υποστηρίζονται πολλαπλές κλάσεις). Για παλινδρόμηση, το `<label>` είναι η τιμή στόχος που μπορεί να είναι οποιοσδήποτε πραγματικός αριθμός. Το ζεύγος `<index>:<value>` δίνει την τιμή ενός χαρακτηριστικού: το `<index>` είναι ένας ακέραιος που ξεκινάει από το 1 και το `<value>` είναι ένας πραγματικός αριθμός. Η μόνη εξαίρεση συναντάται στην περίπτωση του προ-υπολογισμένου πυρήνα, όπου το `<index>` ξεκινάει από το 0. Οι δείκτες (`<index>`) θα πρέπει να είναι αυστηρά σε αύξουσα σειρά. Οι ετικέτες στο αρχείο των παραδειγμάτων για πρόβλεψη χρησιμοποιούνται μόνο για τον υπολογισμό της ακρίβειας των σφαλμάτων. Αν είναι άγνωστα, τότε πρέπει να γεμίσουμε την πρώτη στήλη με οποιουσδήποτε αριθμούς.

Η μετατροπή αυτή όμως απέτυχε καθώς υπήρχαν γραμμές με ελλιπή δεδομένα παρόλο που οι δημιουργοί της συλλογής των δεδομένων δήλωναν ρητά το αντίθετο. Με κατάλληλη προεπεξεργασία, καθαρίσαμε τις λανθασμένες γραμμές, αντικαταστήσαμε τις κατηγορικές μεταβλητές με αριθμούς που δηλώνουν την κλάση, ρυθμίσαμε σωστά τους τύπους των στηλών, και συνεχίσαμε ολοκληρώνοντας επιτυχώς τη μετατροπή των δεδομένων στη μορφή libsvm.

Στη συνέχεια προχωρήσαμε στην εφαρμογή και εκτέλεση των αλγορίθμων που αναφέρθηκαν στο Κεφάλαιο 3. Εκ πρώτης όψεως, τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά, ειδικά σε σύγκριση με όσα ανέφεραν οι δημιουργοί της συλλογής δεδομένων [Kwap11]. Για το λόγο αυτό, προχωρήσαμε στην αναζήτηση των βέλτιστων παραμέτρων για τις μεθόδους που εξετάσαμε.

4.4 Αναζήτηση βέλτιστων παραμέτρων

Όπως αναφέρθηκε και στην Ενότητα 4.3, τα αρχικά αποτελέσματα της εκτέλεσης των αλγορίθμων που εξετάζουμε δεν ήταν ενθαρρυντικά. Αυτό οφείλεται στο γεγονός ότι οι μέθοδοι επιβλεπόμενης μάθησης που εξετάσαμε είναι παραμετρικές, για τις οποίες χρησιμοποιήσαμε κάποιες προκαθορισμένες τιμές, που εκ των πραγμάτων αποδείχθηκε ότι δεν ήταν καλές επιλογές. Συνεπώς κρίθηκε αναγκαίο να προχωρήσουμε στην αναζήτηση των βέλτιστων παραμέτρων για κάθε μοντέλο που εξετάζουμε. Για το σκοπό αυτό, υπάρχουν πολλές τεχνικές στη βιβλιογραφία, όπως η *τυχαία αναζήτηση*, η *μπεϋζιανή βελτιστοποίηση* και η *εξελικτική βελτιστοποίηση*. Στο πλαίσιο αυτής της εργασίας, ωστόσο, επιλέξαμε την *αναζήτηση πλέγματος*.

4.4.1 Αναζήτηση Πλέγματος

Η αναζήτηση πλέγματος αποτελεί μια από τις πιο κλασσικές μεθόδους βελτιστοποίησης υπερπαραμέτρων και επί της ουσίας πραγματοποιεί μια εξαντλητική αναζήτηση σε ένα υποσύνολο του χώρου των υπερπαραμέτρων ενός αλγορίθμου εκμάθησης, το οποίο ορίζουμε εμείς. Ένας τέτοιος αλγόριθμος αναζήτησης πρέπει να προσανατολιστεί από κάποια μετρική απόδοσης που συνήθως μετριέται με τη χρήση *διασταυρούμενης επικύρωσης* στο σύνολο εκπαίδευσης. Στην περίπτωσή μας, η μετρική που χρησιμοποιήσαμε ήταν η *ακρίβεια* (accuracy).

Από τη στιγμή που ο χώρος των παραμέτρων ενός αλγορίθμου εκμάθησης μπορεί να περιλαμβάνει πραγματικές τιμές ή χώρους τιμών χωρίς όρια για συγκεκριμένες παραμέτρους, μπορεί να είναι απαραίτητος ο χειροκίνητος ορισμός ορίων πριν την εφαρμογή της αναζήτησης πλέγματος. Για παράδειγμα, ένας τυπικός ταξινομητής SVM με πυρήνα ακτινικής συνάρτησης βάσης έχει το λιγότερο δύο υπερ-παραμέτρους που πρέπει να προσαρμοστούν για καλή απόδοση στα δεδομένα:

1. τον όρο κανονικοποίησης C και
2. την υπερπαράμετρο πυρήνα γ

Και οι δύο υπερ-παράμετροι είναι συνεχείς, οπότε για την εφαρμογή της αναζήτηση πλέγματος πρέπει να επιλεγεί ένα πεπερασμένο σύνολο λογικών τιμών για κάθε μία, όπως λ.χ. αυτό που περιγράφεται στην Εξίσωση 4.1

$$C \in \{10, 100, 1000\}, \quad \gamma \in \{0.1, 0.2, 0.5, 1.0\} \tag{4.1}$$

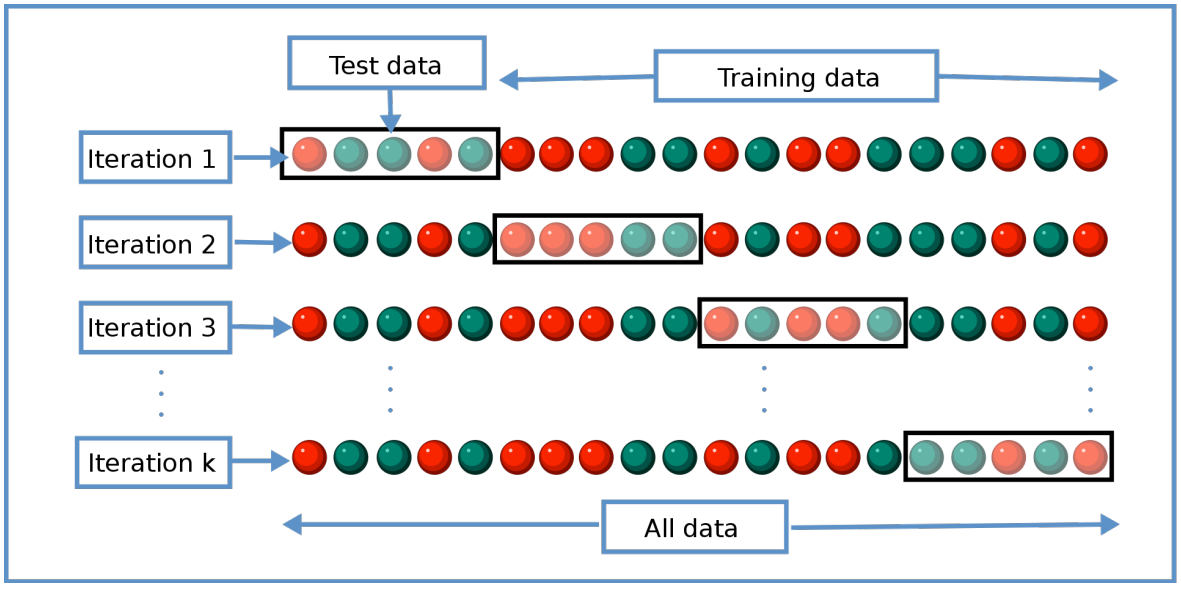
Στη συνέχεια η αναζήτηση πλέγματος εκπαιδεύει το SVM με κάθε ζεύγος (C, γ) στο καρτεσιανό γινόμενο των δύο αυτών συνόλων και προσδιορίζει την απόδοση σε ένα *σύνολο επικύρωσης* (validation set) ή χρησιμοποιώντας εσωτερικά διασταυρούμενη επικύρωση στο σύνολο εκπαίδευσης, στην οποία περίπτωση εκπαιδεύονται πολλαπλά SVM ανά ζεύγος. Τελικά, ο αλγόριθμος αναζήτησης πλέγματος επιστρέφει τις τιμές εκείνες για τις οποίες επιτεύχθηκε το υψηλότερο σκορ στη διαδικασία επικύρωσης.

Η αναζήτηση πλέγματος έχει το μειονέκτημα όχι πάσχει από την *κατάρα της διαστατικότητας* (curse of dimensionality), αλλά από την άλλη είναι εύκολα παραλληλοποιήσιμη, εξαιτίας των ρυθμίσεων των υπερπαραμέτρων που εξάγει και οι οποίες τυπικά είναι ανεξάρτητες μεταξύ τους.

4.4.2 Διασταυρούμενη επικύρωση k-μερών

Σε αυτή τη μέθοδο, το αρχικό δείγμα χωρίζεται τυχαία σε k μέρη ίδιου μεγέθους. Ένα από αυτά τα μέρη κρατιέται ως σύνολο ελέγχου για να δοκιμαστεί το μοντέλο, και τα υπόλοιπα $k-1$ μέρη χρησιμοποιούνται ως δεδομένα εκπαίδευσης. Η διαδικασία αυτή επαναλαμβάνεται k φορές, με κάθε ένα από τα k μέρη χρησιμοποιείται ακριβώς μία φορά ως δεδομένα ελέγχου (Σχήμα 4.2). Στα k αποτελέσματα υπολογίζεται ο μέσος όρος ώστε να έχουμε μία εκτίμηση. Το πλεονέκτημα αυτής της μεθόδου σε επαναλαμβανόμενα τυχαία μέρη είναι ότι όλες οι παρατηρήσεις χρησιμοποιούνται και για εκπαίδευση και για επιβεβαίωση, και κάθε παρατήρηση χρησιμοποιείται για επιβεβαίωση ακριβώς μία φορά.

Για παράδειγμα, θέτουμε $k = 2$. Στη διασταυρούμενη επικύρωση 2 μερών, ανακατεύουμε τυχαία το σύνολο δεδομένων και το χωρίζουμε σε δύο σύνολα, d_0 και d_1, ιδίου μεγέθους. Μετά εκπαιδεύουμε πάνω στο d_0 και ελέγχουμε πάνω στο d_1, συνεχίζοντας με την εκπαίδευση στο d_1 και τον έλεγχο στο d_0.



Σχήμα 4.2: Απεικόνιση διασταυρούμενης επικύρωσης k-μερών

Η τιμή που επιλέξαμε για το k στη δική μας περίπτωση ισούται με 5 και αποτελεί μια πρακτική που είναι ευρέως διαδεδομένη στα προβλήματα μηχανικής μάθησης. Μια άλλη τιμή που επίσης χρησιμοποιείται συχνά είναι το 10.

Στρωματοποιημένη διασταυρούμενη επικύρωση k-μερών

Σε αυτή την περίπτωση τα μέρη επιλέγονται έτσι ώστε η μέση τιμή απόκρισης να είναι περίπου ίση σε κάθε ένα από αυτά. Στην περίπτωση της δυαδικής ταξινόμησης, αυτό σημαίνει πως κάθε μέρος περιέχει περίπου τις ίδιες αναλογίες των δύο τύπων ετικετών κλάσεων.

Συνοπτικά, αυτό που κάναμε ήταν, χρησιμοποιώντας $k = 5$, να εκτελέσουμε μία επαναληπτική διαδικασία κατά την οποία κάθε φορά κρατούσαμε το πρώτο 80% των συνολικών δεδομένων. Στη συνέχεια, κρατήσαμε το υπόλοιπο 20% ως σύνολο σύνολο ελέγχου για κάθε ένα από τα 5 μέρη.

4.4.3 Τιμές Υπερ-Παραμέτρων

Στους Πίνακες 4.3-4.7 εμφανίζονται, για κάθε αλγόριθμο ξεχωριστά, οι υπερ-παράμετροι, οι τιμές αυτών που χρησιμοποιήθηκαν κατά την πειραματική διεργασία που εκτελέσαμε και μια σύντομη εξήγηση αυτών των επιλογών.

Πίνακας 4.3: Εύρη υπερ-παραμέτρων για τα δέντρα απόφασης

Υπερπαράμετρος	**Εύρος**
Μετρικές μη-καθαρότητας κόμβων	Gini, εντροπία
Ελάχιστο πληροφοριακό κέρδος	$0.0, 0.01, 0.1$
Μέγιστο βάθος δέντρου	$5, 10, 15$
Ελάχιστος αριθμός παραδειγμάτων ανά κόμβο	$1, 2$

Πίνακας 4.4: Εύρη υπερ-παραμέτρων για τα τυχαία δάση

Υπερπαράμετρος	**Εύρος**
Πλήθος δέντρων	$10, 20$
Μέγιστο βάθος δέντρου	$5, 10, 15$

Πίνακας 4.5: Εύρη υπερ-παραμέτρων για τη λογιστική παλινδρόμηση «ενός-εναντίων-υπολοίπων» με χρήση Elastic Net

Υπερπαράμετρος	**Εύρος**
Παράγοντας ομαλοποίησης	$0.3, 0.5$
Μέγιστο πλήθος επαναλήψεων	$10, 50, 100$

Πίνακας 4.6: Εύρη υπερ-παραμέτρων για MLR

Υπερπαράμετρος	**Εύρος**
Παράμετρος Elastic Net	$0.0, 0.4, 0.8$
Παράγοντας ομαλοποίησης	$0.3, 0.5$
Μέγιστο πλήθος επαναλήψεων	$10, 50, 100$
Χρήση όρου τομής	Ναι, Όχι

Πίνακας 4.7: Εύρη υπερ-παραμέτρων για MLP

Υπερπαράμετρος	**Εύρος**
Πλήθος κρυφών επιπέδων	$1, 2, 3, 4$
Πλήθος νευρώνων ανά κρυφό επίπεδο	1 ως 10
Μέγεθος δέσμης δειγμάτων	$128, 512$
η	$10.01, 0.03, 0.05$

4.5 Βέλτιστες υπερ-παράμετροι και αποτελέσματα

Στους Πίνακες 4.8-4.12 εμφανίζονται, για κάθε αλγόριθμο ξεχωριστά, οι βέλτιστες υπερ-παράμετροι καθώς και το σφάλμα ελέγχου ανά μέρος και συνολικά. Να σημειωθεί πως το σφάλμα είναι ίσο με 100(%) μείον την ακρίβεια.

Πίνακας 4.8: Βέλτιστες υπερ-παράμετροι και σφάλμα ελέγχου για τα δέντρα απόφασης

Μέρος	Μετρική μη-καθαρότητας κόμβων	Ελάχιστο πληροφοριακό κέρδος	Μέγιστο βάθος δέντρου	Ελάχιστος αριθμός παραδειγμάτων ανά κόμβο	Σφάλμα Ελέγχου
1	Gini	0.0	15	1	22.95%
2	Gini	0.0	15	2	22.97%
3	Gini	0.0	15	2	23.11%
4	Εντροπία	0.0	0	2	23.02%
5	Gini	0.0	15	1	23.07%

Μέσος όρος σφάλματος ελέγχου: 23.02%

Πίνακας 4.9: Βέλτιστες υπερ-παράμετροι και σφάλμα ελέγχου για τα τυχαία δάση

Μέρος	Μετρική μη-καθαρότητας κόμβων	Πλήθος Δέντρων	Μέγιστο βάθος δέντρου	Ελάχιστος αριθμός παραδειγμάτων ανά κόμβο	Σφάλμα Ελέγχου
1	Gini	20	15	2	23.41%
2	Gini	15	20	2	23.20%
3	Gini	15	20	2	23.11%
4	Gini	10	15	2	23.23%
5	Gini	20	15	2	23.16%

Μέσος όρος σφάλματος ελέγχου: 23.22%

Πίνακας 4.10: Βέλτιστες υπερ-παράμετροι και σφάλμα ελέγχου για τη λογιστική παλινδρόμηση «ενός-εναντίων-υπολοίπων» με χρήση Elastic Net

Μέρος	Παράμετρος Elastic Net	Παράγοντας ομαλοποίησης	Μέγιστο πλήθος επαναλήψεων	Χρήση όρου τομής	Σφάλμα Ελέγχου
1	0.4	0.3	10	ΝΑΙ	57.87%
2	0.4	0.3	10	ΝΑΙ	57.86%
3	0.4	0.3	10	ΝΑΙ	57.89%
4	0.4	0.3	10	ΝΑΙ	57.89%
5	0.4	0.3	10	ΝΑΙ	57.86%

Μέσος όρος σφάλματος ελέγχου: 57.87%

Πίνακας 4.11: Βέλτιστες υπερ-παράμετροι και σφάλμα ελέγχου για MLR

Μέρος	**Όριο ανοχής σφάλματος**	**Παράγοντας ομαλοποίησης**	**Μέγιστο πλήθος επαναλήψεων**	**Χρήση όρου τομής**	**Σφάλμα Ελέγχου**
1	10^{-5}	0.5	10	NAI	57.87%
2	10^{-5}	0.5	10	NAI	57.86%
3	10^{-5}	0.5	10	NAI	57.89%
4	10^{-5}	0.5	10	NAI	57.89%
5	10^{-5}	0.5	10	NAI	57.86%

Μέσος όρος σφάλματος ελέγχου: 57.87%

Πίνακας 4.12: Βέλτιστες υπερ-παράμετροι και σφάλμα ελέγχου για MLP

Μέρος	**Μέγιστο πλήθος επαναλήψεων**	**Νευρώνες κρυφών επιπέδων**	**Μέγεθος δέσμης δειγμάτων**	η	**Όριο ανοχής σφάλματος**	**Μέθοδος βελτιστο-ποίησης**	**Σφάλμα ελέγχου**
1	100	[3, 7, 7, 6]	512	0.05	10^{-6}	l-bfgs	42.88%
2	100	[3, 7, 7, 6]	512	0.03	10^{-6}	l-bfgs	42.89%
3	100	[3, 7, 7, 6]	128	0.01	10^{-6}	l-bfgs	42.91%
4	100	[3, 7, 7, 6]	512	0.03	10^{-6}	l-bfgs	42.94%
5	100	[3, 7, 7, 6]	128	0.03	10^{-6}	l-bfgs	42.75%

Μέσος όρος σφάλματος ελέγχου: 42.87%

Κεφάλαιο 5

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

5.1 Συμπεράσματα

Οι δημιουργοί της συλλογής δεδομένων Actitracker [Kwap11] έχουν δημοσιεύσει τα ακόλουθα αποτελέσματα (Πίνακας 5.1)

Πίνακας 5.1: Αποτελέσματα αλγορίθμων επιβλεπόμενης μάθησης της εργασίας [Kwap11]

Αλγόριθμος	**Σφάλμα Ελέγχου**
MLP	32.2%
Instance-based learning	31.6%
Τυχαία Δάση	24.1%
Βαθύ MLP (5 επίπεδα με 200 νευρώνες το καθένα)	14.4%

Οι τεχνικές που εμείς χρησιμοποιήσαμε είναι συγκρίσιμες σε αποτέλεσμα μόνο με την πρώτη και την τρίτη τεχνική από τις προαναφερόμενες, καθώς μελετήσαμε αλγορίθμους μάθησης με επίβλεψη. Όσον αφορά, λοιπόν, τη σύγκριση των αποτελεσμάτων φαίνεται πως υστερούμε στους αλγορίθμους MLP, MLR με Elastic Net, λογιστική παλινδρόμηση, ενώ τα αποτελέσματά μας ήταν καλύτερα για τις μεθόδους των δέντρων απόφασης και των τυχαίων δασών. Τα αποτελέσματα αυτά, προφανώς, απαιτούν περαιτέρω διερεύνηση μέσω της αύξησης του εύρους των υπερπαραμέτρων που εξετάστηκαν.

5.2 Μελλοντικές Κατευθύνσεις

Η υλοποίηση που ακολουθήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας αποτελεί μία πρώτη προσέγγιση στο πρόβλημα της αναγνώρισης κίνησης και συνεπώς υπάρχει περιθώριο για αρκετές και διαφορετικές μελλοντικές επεκτάσεις και βελτιώσεις.

Μερικές ενδιαφέρουσες προτάσεις συνοψίζονται στα παρακάτω σημεία

- Η εξερεύνηση τεχνικών που ανήκουν στην περιοχή της μάθησης χωρίς επίβλεψη, όπως η *συσταδοποίηση* (clustering).
- Η χρήση άλλων βιβλιοθηκών αντί για την Spark MLlib, όπως η H2O ή μια έκδοσή της που συνδυάζει το Spark, την H2O Sparkling Water.
- Η χρήση της γλώσσας προγραμματισμού Java αντί της Python ως γλώσσα υλοποίησης, μιας και το λογισμικό του cluster είναι γραμμένο σε Java, οπότε έτσι «γλυτώνουμε» τα επιπρόσθετα επίπεδα επεξεργασίας και μεταγλώττισης.
- Η πιο πληρέστερη εξέταση τεχνικών βαθιάς μάθησης
- Τέλος, θα μπορούσαμε να κατασκευάσουμε ένα σύστημα, το οποίο θα λαμβάνει δεδομένα σε πραγματικό χρόνο και παράλληλα θα προσπαθεί να κάνει προβλέψεις.

Βιβλιογραφία

[AAAI12] *"The Impact of Personalization on Smartphone-Based Activity Recognition"*, AAAI-12 Workshop on Activity Context Representation: Techniques and Languages, Toronto, CA, 2012.

[Alpa10] Ethem Alpaydin, *Introduction to machine learning*, MIT Press, Cambridge, Mass, 2010.

[Alsh16] Mohammad Abu Alsheikh, Dusit Niyato, Shaowei Lin, Hwee pink Tan and Zhu Han, "Mobile big data analytics using deep learning and apache spark", *IEEE Network*, vol. 30, no. 3, pp. 22–29, may 2016.

[Apac20] ApacheFoundation, ""Apache Spark–Lightning-Fast Cluster Computing"", Internet, May 2020.

[Bish06] Christofer Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag New York, 2006.

[Brei93] Leo Breiman, *Classification and regression trees*, Chapman & Hall, New York, 1993.

[Brei96] Leo Breiman, "Bagging predictors", *Machine Learning*, vol. 24, no. 2, pp. 123–140, aug 1996.

[BRIT] "http://www.britannica.com/EBchecked/topic/1116194/machine-learning", Internet.

[Cham18] Bill Chambers, *Spark: The Definitive Guide Big Data Processing Made Simple*, OReilly Media, Inc., 2018.

[Chan11] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines", *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, May 2011.

[Chou13] Harsh Chouraria, ""MR2 and YARN Briefly Explained"", Internet, October 2013.

[Cisc16] Cisco, ""Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2015–2020"", Technical report, Cisco Systems, 2016.

[Duda00] Richard Duda, *Pattern Classification*, Wiley, 2000.

[Evan16] Chris Evans, ""Big data storage: Hadoop storage basics"", Internet, October 2016.

[Gull15] Mohammed Guller, *Big Data Analytics with Spark*, Apress, Berkeley, CA, 2015.

[H2Oa] H2O, Internet.

[H2Ob] H2O, Internet.

[Hast01] Trevor Hastie, *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*, Springer, New York, 2001.

[Hayk99] Simon Haykin, *Neural networks : a comprehensive foundation*, Prentice Hall, Upper Saddle River, N.J, 1999.

[Ho02] Tin Kam Ho, "A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors", *Pattern Analysis & Applications*, vol. 5, no. 2, pp. 102–112, jun 2002.

[Hosm00] David Hosmer, *Applied logistic regression*, Wiley, New York, 2000.

[KDD210] *"Activity Recognition using Cell Phone Accelerometers,"*, Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10), Washington DC, 2010.

[Kwap11] Jennifer R Kwapisz, Gary M Weiss and Samuel A Moore, "Activity recognition using cell phone accelerometers", *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[Lang07] John Langford, Lihong Li and Alexander Strehl, "Vowpal wabbit open source project", *URL https://github. com*, 2007.

[LeCu15] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, may 2015.

[Lior07] Oded Maimon Lior Rokach, *Data Mining with Decision Trees*, World Scientific Publishing Company, 2007.

[Lync96] Nancy Lynch, *Distributed algorithms*, Morgan Kaufmann Publishers, San Francisco, Calif, 1996.

[Magn15] L Magnoni, "Modern Messaging for Distributed Sytems", *Journal of Physics: Conference Series*, vol. 608, p. 012038, may 2015.

[Mann96] Heikki Mannila, "«Data mining: machine learning, statistics, and databases». Int'l Conf. Scientific and Statistical Database Management. IEEE Computer Society.", Internet, 1996.

[Mitc97] T.M. Mitchell, *Machine Learning*, McGraw-Hill international editions - computer science series, McGraw-Hill Education, 1997.

[Owen12] Sean Owen and Sean Owen, *Mahout in action*, vol. 10, Manning Shelter Island, NY, 2012.

[Pand20] Parul Pandey, "Democratising Machine learning with H2O", Internet, April 2020.

[Petu18] Inc. Petuum, "Intro to Distributed Deep Learning Systems", Internet, February 2018.

[Quin86] J. R. Quinlan, "Induction of decision trees", *Machine Learning*, vol. 1, no. 1, pp. 81–106, mar 1986.

[Ripl96] Brian Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[RonK98] Foster Provost Ron Kohavi, "«Glossary of terms»", Internet, 1998.

[Russ10] Stuart Russell, *Artificial intelligence : a modern approach*, Prentice Hall, Upper Saddle River, New Jersey, 2010.

[Schm] Juergen Schmidhuber, "Deep Learning in Neural Networks: An Overview".

[Shva10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia and Robert Chansler, "The hadoop distributed file system", in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pp. 1–10, Ieee, 2010.

[Silb09] Peter; Gagne Greg Silberschatz, Abraham; Galvin, *"Operating System Concepts, 8th Edition"*, John Wiley & Sons, Inc., 2009.

[Simo13] Phil Simon, *Too Big to Ignore*, John Wiley & Sons Inc, 2013.

[Tan17] Ying Tan, *Data mining and big data : second International Conference, DMBD 2017, Fukuoka, Japan, July 27-August 1, 2017. Proceedings*, Springer, Cham, 2017.

[Tane02] Andrew Tanenbaum, *Distributed systems : principles and paradigms*, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[Toll16] Juliana Tolles and William J. Meurer, "Logistic Regression", *JAMA*, vol. 316, no. 5, p. 533, aug 2016.

[Tuck04] Allen Tucker, *Computer science handbook*, Chapman & Hall/CRC, Boca Raton, Fla, 2004.

[Turi50] Alan M. Turing, "Computing Machinery and Intelligence", *Mind*, vol. 59, no. October, pp. 433–60, 1950.

[Walk67] Strother H. Walker and David B. Duncan, "Estimation of the Probability of an Event as a Function of Several Independent Variables", *Biometrika*, vol. 54, no. 1/2, p. 167, jun 1967.

[Wern] Brankov Yourganov Wernick, Yang and Strother, *Machine Learning in Medical Imaging*, chapter 27, pp. 25–38, IEEE Signal Processing Magazine.

[WSDM11] *"Design Considerations for the WISDM Smart Phone-Based Sensor Mining Architecture"*, Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data (at KDD-11), San Diego, CA, 2011.

[Zaha16] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker and Ion Stoica, "Apache Spark: A Unified Engine for Big Data Processing", *Commun. ACM*, vol. 59, no. 11, p. 56–65, October 2016.

[Δι07] Διαμαντάρας, *Τεχνητά Νευρωνικά Δίκτυα*, Κλειδάριθμος, 2007.

Παράρτημα Α

Ευρετήριο Ακρωνυμίων και Συντμήσεων

Με αλφαβητική σειρά, ως προς τη σύντμηση

Α.1 Ελληνικών όρων

ΤΝ: Τεχνητή Νοημοσύνη

Α.2 Αγγλικών όρων

API: Application Programming Interface (Προγραμματιστική Διεπαφή)
CAP: Credit Assignment Path (Μονοπάτι Ανάθεσης Πίστωσης)
CNN: Convolutional Neural Networks (Συνελικτικά Νευρωνικά Δίκτυα)
DT: Decision Trees (Δέντρα απόφασης)
DFS: Distributed File System (Κατανεμημένο Σύστημα Αρχείων)
DSL: Domain Specific Language
HDFS: Hadoop Distributed File System
IIA: Independence of Irrelevant Alternatives (Ανεξαρτησία των μη-σχετικών εναλλακτικών)
MBD: Mobile Big Data (Μεγάλα Δεδομένα Φορητών Συσκευών)
MLR: Multinomial Logistic Regression (Πολυωνυμική Λογιστική Παλινδρόμηση)
MLP: Multilayer Perceptron (Πολυεπίπεδο Perceptron)
NVRAM: Non-Volatile Random-Access Memory (μη-πτητική μνήμη τυχαίας προσπέλασης)
OCR: Optical Character Recognition (Οπτική Αναγνώριση Χαρακτήρων)
RDD: Resilient Distributed Dataset
RF: Random Forest (Τυχαίο δάσος)