

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ  
ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΑΣ  
ΕΡΓΑΣΤΗΡΙΟ ΕΝΣΤΡΜΑΤΗΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΣ



## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και  
Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του  
Πανεπιστημίου Πατρών

ΑΝΔΡΕΑ ΜΠΑΜΠΟΥΡΗ του ΣΩΤΗΡΙΟΥ

Α.Μ.: 227320

Ανάπτυξη βιβλιοθήκης λογισμικού  
για εφαρμογές μηχανικής μάθησης  
σε παράλληλα υπολογιστικά συστήματα

Επιβλέπων  
Δρ.-Μηχ. Ευάγγελος Δερματάς

Πάτρα, Οκτώβριος 2018



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ  
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ  
ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΑΣ  
ΕΡΓΑΣΤΗΡΙΟ ΕΝΣΤΡΜΑΤΗΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΣ



## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και  
Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του  
Πανεπιστημίου Πατρών

ΑΝΔΡΕΑ ΜΠΑΜΠΟΥΡΗ του ΣΩΤΗΡΙΟΥ

Α.Μ.: 227320

Ανάπτυξη βιβλιοθήκης λογισμικού  
για εφαρμογές μηχανικής μάθησης  
σε παράλληλα υπολογιστικά συστήματα

Επιβλέπων  
Δρ.-Μηχ. Ευάγγελος Δερματάς

Πάτρα, Οκτώβριος 2018



# ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η διπλωματική εργασία με θέμα

«Ανάπτυξη βιβλιοθήκης λογισμικού  
για εφαρμογές μηχανικής μάθησης  
σε παράλληλα υπολογιστικά συστήματα»

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και  
Τεχνολογίας Υπολογιστών

**ΑΝΔΡΕΑ ΜΠΑΜΠΟΥΡΗ του ΣΩΤΗΡΙΟΥ**

A.M.: 227320

παρουσιάστηκε δημόσια και εξετάστηκε στο  
Τμήμα Ηλεκτρολόγων Μηχανικών και  
Τεχνολογίας Υπολογιστών στις

05 / 10 / 2018

Ο Επιβλέπων

Ευάγγελος Δερματάς  
Αναπληρωτής Καθηγητής

Ο Διευθυντής του Τομέα

Θεόδωρος Αντωνακόπουλος  
Καθηγητής



Αριθμός Διπλωματικής Εργασίας: **227320/2018**

Θέμα: «**Ανάπτυξη βιβλιοθήκης λογισμικού για εφαρμογές μηχανικής μάθησης σε παράλληλα υπολογιστικά συστήματα**»

Φοιτητής: **Ανδρέας Μπαμπούρης**

Επιβλέπων: **Δρ.-Μηχ. Ευάγγελος Δερματάς**

## Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η προγραμματιστική υλοποίηση βασικών τεχνικών μηχανικής μάθησης με τέτοιο τρόπο που να γίνεται εκμετάλλευση των αρχιτεκτονικών παράλληλης επεξεργασίας που συναντώνται στους σύγχρονους υπολογιστές. Έχουν υλοποιηθεί μοντέλα και μέθοδοι όπως αυτά της γραμμικής παλινδρόμησης, της λογιστικής παλινδρόμησης, των τεχνητών νευρωνικών δικτύων εμπρόσθιας τροφοδότησης, του απλού ταξινομητή Bayes, της εκμάθησης δέντρων αποφάσεων, και της ομαδοποίησης  $k$ -μέσων. Έγινε χρήση της γλώσσας προγραμματισμού C++ μαζί με τη βιβλιοθήκη γραμμικής άλγεβρας Eigen, ενώ για την παραλληλοποίηση των αλγορίθμων χρησιμοποιήθηκε το πρότυπο OpenMP. Στα πλαίσια της εργασίας έγιναν δοκιμές σε πραγματικά δεδομένα, και η προκύπτουσα βιβλιοθήκη λογισμικού μπορεί να χρησιμοποιηθεί για την ανάπτυξη πραγματικών εφαρμογών.



Diploma Thesis Number: **227320/2018**

Topic: “**Development of a software library for machine learning applications on parallel computers**”

Student: **Andreas Bampouris**

Supervisor: **Dr.-Eng. Evangelos Dermatas**

## **Abstract**

The objective of this diploma thesis is a software implementation of certain basic machine learning techniques in such way that they are able to make use of the parallel computing architectures present in modern computers. Models and methods implemented include: linear regression, logistic regression, feedforward neural networks, the naive Bayes classifier, decision tree learning, and  $k$ -means clustering. The programming language used is C++ along with the linear algebra library Eigen, while the OpenMP standard was used to introduce parallelization. Real data from various sources were used for the purpose of testing the library’s capabilities, and the end result can be used to develop real applications.



*Στη μητέρα μου Αναστασία*



## Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Δρ.-Μηχ. Ευάγγελο Δερματά, τόσο για τις συμβουλές και την καθοδήγηση κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας, όσο και για το ενδιαφέρον που μου ενέπνευσε για τη μηχανική μάθηση στα πλαίσια των σπουδών μου, μέσω των μαθημάτων Αναγνώρισης Προτύπων. Ευχαριστώ κατ' επέκταση τους καθηγητές του τμήματός μας για το πλούσιο πρόγραμμα σπουδών που είχα την ευκαιρία να παρακολουθήσω και το οποίο με κατέστησε ικανό να ασχοληθώ σε βάθος με θέματα όπως η μηχανική μάθηση, η επεξεργασία δεδομένων, και ο παράλληλος προγραμματισμός. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, και ιδιαίτερα τον πατέρα μου, για την υπομονή και για τη στήριξη που μου προσέφεραν όσο διόλευα στην παρούσα εργασία.



# Περιεχόμενα

<b>1 Πρόλογος</b>	<b>1</b>
1.1 Μηχανική μάθηση . . . . .	3
1.2 Παράλληλα υπολογιστικά συστήματα . . . . .	4
1.2.1 Παράλληλος προγραμματισμός . . . . .	6
1.2.2 Το πρότυπο OpenMP . . . . .	8
1.3 Βιβλιοθήκη λογισμικού . . . . .	11
1.3.1 Βιβλιοθήκες μηχανικής μάθησης στη βιομηχανία . . . . .	12
1.4 Η βιβλιοθήκη «metis» της εργασίας . . . . .	14
1.4.1 Διάρθρωση της αναφοράς . . . . .	15
<b>2 Θεωρητικό υπόβαθρο</b>	<b>17</b>
2.1 Δεδομένα και μηχανική μάθηση . . . . .	17
2.1.1 Προετοιμασία των δεδομένων . . . . .	19
2.2 Γραμμική παλινδρόμηση . . . . .	20
2.2.1 Το γραμμικό μοντέλο . . . . .	21
2.2.2 Μέθοδος ελαχίστων τετραγώνων . . . . .	23
2.2.3 Απλή γραμμική παλινδρόμηση . . . . .	24
2.3 Λογιστική παλινδρόμηση . . . . .	24
2.3.1 Το λογιστικό μοντέλο . . . . .	25
2.3.2 Μέθοδος επικλινούς καθόδου . . . . .	27
2.3.3 Τεχνική επιλογής One-vs-Rest . . . . .	28
2.4 Τεχνητά νευρωνικά δίκτυα . . . . .	29
2.4.1 Δομή του TNΔ . . . . .	30
2.4.2 Η εμπρόσθια τροφοδότηση . . . . .	33

2.4.3	Η οπισθοδρομική διάδοση του σφάλματος . . . . .	33
2.4.4	Επιλογή της συνάρτησης ενεργοποίησης . . . . .	34
2.5	Απλός ταξινομητής Bayes . . . . .	35
2.5.1	Πιθανοτικό μοντέλο . . . . .	36
2.5.2	Multinomial ταξινομητής . . . . .	37
2.5.3	Gaussian ταξινομητής . . . . .	38
2.6	Εκμάθηση δέντρων αποφάσεων . . . . .	39
2.6.1	Χαρακτηριστικά του δέντρου . . . . .	40
2.6.2	Εκπαίδευση ενός δέντρου αποφάσεων . . . . .	41
2.6.3	Επιλογή του «καλύτερου» χαρακτηριστικού . . . . .	41
2.7	Ομαδοποίηση $k$ -μέσων . . . . .	43
2.7.1	Αλγόριθμος ομαδοποίησης . . . . .	44
2.7.2	Επιλογή των αρχικών σημείων . . . . .	46
2.7.3	Επιλογή του $k$ . . . . .	47
<b>3</b>	<b>Σχεδιασμός του API</b>	<b>49</b>
3.1	Κλάση DataSet . . . . .	49
3.1.1	Κλάση DataLabeled . . . . .	52
3.2	Κλάση LinearRegression . . . . .	53
3.3	Κλάση LogisticRegression . . . . .	54
3.4	Κλάση MLPClassifier . . . . .	56
3.5	Κλάση NaiveBayes . . . . .	58
3.5.1	Κλάση MultinomialNB . . . . .	60
3.5.2	Κλάση GaussianNB . . . . .	61
3.6	Κλάση DecisionTree . . . . .	61
3.6.1	Κλάση DTNode . . . . .	63
3.7	Κλάση KMeans . . . . .	64
<b>4</b>	<b>Λεπτομέρειες υλοποίησης</b>	<b>67</b>
4.1	Ανάγνωση και προετοιμασία των δεδομένων . . . . .	67
4.2	Γραμμική παλινδρόμηση . . . . .	69
4.3	Ταξινόμηση μέσω λογιστικής παλινδρόμησης . . . . .	71

4.4 Ταξινόμηση με TNΔ multilayer perceptron . . . . .	73
4.5 Ταξινόμηση με απλό ταξινομητή Bayes . . . . .	76
4.6 Ταξινόμηση με δέντρο αποφάσεων . . . . .	78
4.7 Ομαδοποίηση $k$ -μέσων . . . . .	80
<b>5 Εφαρμογές και αξιολόγηση</b>	<b>83</b>
5.1 Χρήση της γραμμικής παλινδρόμησης για προβλέψεις επί της προόδου διαβητικών ασθενών . . . . .	83
5.2 Ταξινόμηση λουλουδιών του γένους της ίριδος μεταξύ τριών ειδών . . . . .	85
5.2.1 Χρήση απλού ταξινομητή Bayes . . . . .	85
5.2.2 Χρήση ταξινομητή λογιστικής παλινδρόμησης . . . . .	86
5.2.3 Σύγκριση αποτελεσμάτων . . . . .	87
5.3 Ταξινόμηση χειρόγραφων αριθμητικών χαρακτήρων του συνόλου MNIST . . . . .	88
5.3.1 Χρήση ταξινομητή λογιστικής παλινδρόμησης . . . . .	88
5.3.2 Χρήση TNΔ εμπρόσθιας τροφοδότησης . . . . .	90
5.4 Ταξινόμηση της κατάστασης της ασθένειας καρκινοπαθών ασθενών . . . . .	92
5.4.1 Χρήση του απλού ταξινομητή Bayes . . . . .	92
5.4.2 Χρήση ταξινομητή δέντρου αποφάσεων . . . . .	93
5.4.3 Σύγκριση των αποτελεσμάτων . . . . .	94
5.5 Ομαδοποίηση του συνόλου δεδομένων Iris με ομαδοποιητή $k$ -μέσων . . . . .	94
<b>6 Επίλογος</b>	<b>99</b>
6.1 Συνέχιση ανάπτυξης της βιβλιοθήκης . . . . .	99
6.2 Συμπεράσματα . . . . .	100
<b>Βιβλιογραφία</b>	<b>100</b>
<b>Α' Αντιστοίχηση με ορολογία διεύθυνούς βιβλιογραφίας</b>	<b>3</b>
<b>Β' Σύνολα δεδομένων</b>	<b>5</b>
B'.1 Σύνολο δεδομένων Diabetes . . . . .	5
B'.2 Σύνολο δεδομένων banknote authentication . . . . .	6
B'.3 Σύνολο δεδομένων Iris . . . . .	6

B'.4 Σύνολο δεδομένων Wine . . . . .	7
B'.5 Σύνολο δεδομένων MNIST . . . . .	7
B'.6 Σύνολο δεδομένων Breast Cancer . . . . .	8





# Κεφάλαιο 1

## Πρόλογος

Η λέξη «μῆτις» είναι ένα ουσιαστικό ομόρριζο του ρήματος «μετρώ». Συνδέεται με τη σοφία, την επιδεξιότητα, και την ικανότητα να δίνει κανείς συμβουλές που προκύπτουν ως αποτέλεσμα «μετρημένης» σκέψης. Από τη λέξη αυτή παίρνει το όνομά της η βιβλιοθήκη μηχανικής μάθησης «metis» που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Οι λεπτομέρειες του σχεδιασμού και της ανάπτυξής της θα παρουσιαστούν αναλυτικά, αφού πρώτα γίνει μια μικρή εισαγωγή σε θέματα της μηχανικής μάθησης και της παράλληλης υπολογιστικής.



Από την αρχαιότητα ακόμα, η εμφύσηση νοημοσύνης και συνείδησης σε τεχνητά όντα έχει παρουσιαστεί ως σταθερή προσδοκία της ανθρωπότητας. Τα αυτόματα του θεού Ηφαίστου όπως ο Τάλως παρουσιάζονται στην αρχαία ελληνική μυθολογία ως τεχνητά όντα με ικανότητα νόησης, ενώ παρόμοιοι μύθοι συναντώνται σε όλη την υφήλιο σε πολλές περιόδους της ανθρώπινης ιστορίας. Η ιδέα της μηχανοποίησης της ανθρώπινης σκέψης θα απασχολήσει φιλοσόφους και επιστήμονες για αιώνες, και με την άφιξη του ηλεκτρονικού υπολογιστή τη δεκαετία του 1950 θα αρχίσει να θεμελιώνεται η επιστήμη της τεχνητής νοημοσύνης.

Ήταν το 1950 όταν ο Alan Turing πρότεινε το λεγόμενο «Turing test» με σκοπό τη μέτρηση της νοημοσύνης μιας μηχανής. Τα προηγούμενα χρόνια είχαν κάνει την εμφάνισή τους οι πρώτοι υπολογιστές, ενώ η κατανόηση όλο και περισσότερων μαθηματικών εννοιών προχωρούσε με γοργούς ρυθμούς. Η προσδοκία συνδυασμού των δύο με σκοπό την επίτευξη δημιουργίας μηχανών που να μπορούν να σκέφτονται ενέπνευσε έργα επιστημονικής φαντασίας από οραματιστές όπως ο Isaac Asimov και ο Arthur C. Clarke, ενώ οι πρώτες εφαρμογές άρχισαν να κάνουν την εμφάνισή τους με τη μορφή ανταγωνιστικών προγραμμάτων για παιχνίδια όπως η ντάμα και το σκάκι.



Σχήμα 1.1: Ιδέες εμπνευσμένες από την τεχνητή νοημοσύνη έκαναν αισθητή την παρουσία τους τόσο στη λογοτεχνία όσο και στον κινηματογράφο κατά τις δεκαετίες του 1950 και 1960.

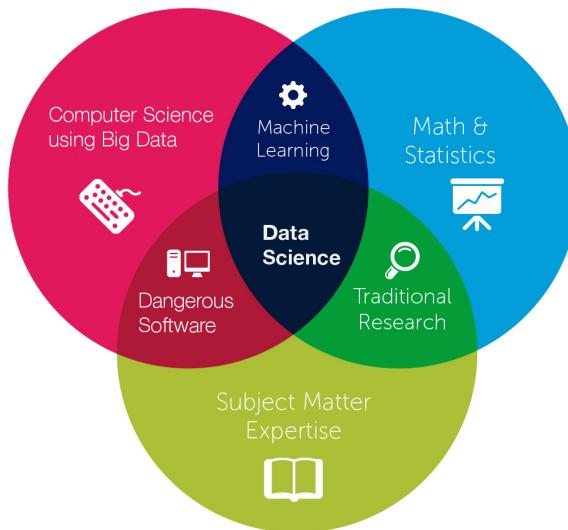
Τις επόμενες δεκαετίες, παράλληλα με την ανάπτυξη της πληροφορικής, μέθοδοι από ένα μεγάλο εύρος επιστημονικών πειδίων αρχίζουν να επανεξετάζονται με σκοπό τη χρήση τους στα πλαίσια της τεχνητής νοημοσύνης. Τεχνικές προερχόμενες από την πιθανοθεωρία και τη στατιστική όπως το θεώρημα του Bayes αποβαίνουν ιδιαίτερα αποτελεσματικές, όπως και βασικές μέθοδοι που αναπτύσσονται στα πλαίσια της αναγνώρισης προτύπων. Ωστόσο, η πορεία της επιστήμης της τεχνητής νοημοσύνης θα έχει πολλά σκαμπανεβάσματα μέχρι την ανακάλυψη της μεθόδου της οπισθοδρομικής διάδοσης τη δεκαετία του 1980. Η ανακάλυψη αυτή θα αλλάξει την προσέγγιση που ακολουθούνταν μέχρι εκείνη την εποχή έτσι ώστε να βασίζεται λιγότερο σε τυποποίηση και μοντελοποίηση της γνώσης, και περισσότερο στην μάθηση μέσω δεδομένων.

Το πεδίο της μηχανικής μάθησης υπόσχεται να γίνει ένα από τα βασικά μέσα επίτευξης τεχνητής νοημοσύνης και αναζωπυρώνει το ενδιαφέρον, ιδιαίτερα τις δύο τελευταίες δεκαετίες. Μέθοδοι όπως αυτές των νευρωνικών δικτύων και των μηχανών διανυσμάτων υποστήριξης λαμβάνουν χυρίαρχο ρόλο, και μέσω της «βαθιάς» μάθησης

η βελτίωση της ακρίβειας προβλέψεων συνεχίζεται ανελλιπώς. Σε συνδυασμό με τη μεγάλη αύξηση της υπολογιστικής ισχύος τον 21ο αιώνα μέσω εξαιρετικά γρήγορων πολυπύρηνων επεξεργαστών, η μηχανική μάθηση αποδεικνύεται καρποφόρα, με επιτυχημένες εφαρμογές τόσο στην ακαδημία, όσο και στη βιομηχανία.

## 1.1 Μηχανική μάθηση

Σκοπός του επιστημονικού πεδίου της μηχανικής μάθησης είναι η ανάπτυξη τεχνικών και μεθόδων που προσδίδουν στους υπολογιστές τη δυνατότητα μάθησης, δηλαδή τη δυνατότητα βελτίωσης της συμπεριφοράς και της απόδοσής τους πάνω σε μια εργασία χωρίς να έχει γίνει ρητός προγραμματισμός για αυτή εκ των προτέρων. Η μηχανική μάθηση μπορεί να υεωρηθεί ένα μέσο επίτευξης τεχνητής νοημοσύνης και συνδέεται άμεσα με πεδία όπως η αναγνώριση προτύπων, η στατιστική, και η ανάλυση σημάτων, από τα οποία υιοθετεί και αναπροσαρμόζει τεχνικές.



Σχήμα 1.2: Η μηχανική μάθηση αποτελεί συστατικό μέρος της επιστήμης των δεδομένων.

Η βασική ιδέα πίσω από τις μεθόδους μηχανικής μάθησης είναι η σύνθεση μοντέλων για συγκεκριμένα φαινόμενα, χρησιμοποιώντας δεδομένα που έχουν συλλεχθεί για αυτά από την έρευνα των αντίστοιχων επιστημονικά πεδίων στα οποία εμπίπτουν. Τα μοντέλα που προκύπτουν μπορούν στη συνέχεια να χρησιμοποιηθούν για να γίνουν προβλέψεις και να ληφθούν διαφόρων τύπων συμπεράσματα. Υπάρχουν πολλές διαφορετικές προσεγγίσεις παραγωγής μοντέλων, οι οποίες χρησιμεύουν για την ανάπτυξη πολλών διαφορετικών τύπων εφαρμογών.

Στο υψηλότερο επίπεδο, οι τεχνικές μηχανικής μάθησης ταξινομούνται πιο συχνά

σε δύο μεγάλες κατηγορίες που σχετίζονται με τη φύση των δεδομένων που χρησιμοποιούνται για να γίνει μάθηση:

- στην *επιβλεπόμενη μάθηση*, όπου ο υπολογιστής έχει μπροστά του ένα σύνολο δεδομένων που αποτελείται από ένα ζεύγος δεδομένων εισόδου και εξόδου, και στόχος του είναι η μοντελοποίηση της σχέσης μεταξύ τους,
- στη *μη-επιβλεπόμενη μάθηση*, όπου τα δεδομένα εκπαιδευσης αποτελούνται μόνο από δεδομένα εισόδου, στερούνται δηλαδή κάποιου συμπεράσματος ή «ετικέτας», και στόχος είναι η εύρεση κάποιας πιθανής δομής στα δεδομένα.

Ένα διαφορετικό είδος διάκρισης των τεχνικών μηχανικής μάθησης γίνεται βάσει του είδους των εφαρμογών στις οποίες πρόκειται να χρησιμοποιηθούν:

- στην *ταξινόμηση* κάθε παράδειγμα στα δεδομένα ανήκει σε μία ή περισσότερες κλάσεις, και στόχος του μοντέλου που εκπαιδεύεται είναι να ανακαλύψει τη σχέση μεταξύ των μεταβλητών εισόδου και των κλάσεών τους έτσι ώστε να μπορεί να κάνει προβλέψεις πάνω σε νέα δεδομένα εισόδου των οποίων η κλάση είναι άγνωστη,
- στην *παλινδρόμηση* μοντελοποιείται η σχέση μεταξύ των μεταβλητών εισόδου και εξόδου σε δεδομένα που έχουν συνεχείς αριθμητικές τιμές εκατέρωθεν, με στόχο ο προβλεπτής που εκπαιδεύεται να μπορεί να κάνει κάποια υπόθεση για την έξοδο μιας νέας εισόδου,
- στην *ομαδοποίηση* (ή *συσταδοποίηση*) δίνεται ως είσοδος ένα σύνολο από δεδομένα κάποιου αριθμού χαρακτηριστικών για τα οποία δεν καταγράφεται κάποιο συμπέρασμα, και επιδιώκεται η δημιουργία ομάδων από αυτά τα δεδομένα,
- στη *μείωση της διαστασιμότητας* στόχος είναι η απλοποίηση ενός συνόλου δεδομένων μέσω της αφαίρεσης διαστάσεων που δε συμβάλλουν σημαντικά στην ερμηνεία του φαινομένου, ή μέσω της δημιουργίας ενός μικρότερου αριθμού νέων διαστάσεων που προκύπτουν ως μετασχηματισμός των παλαιών.

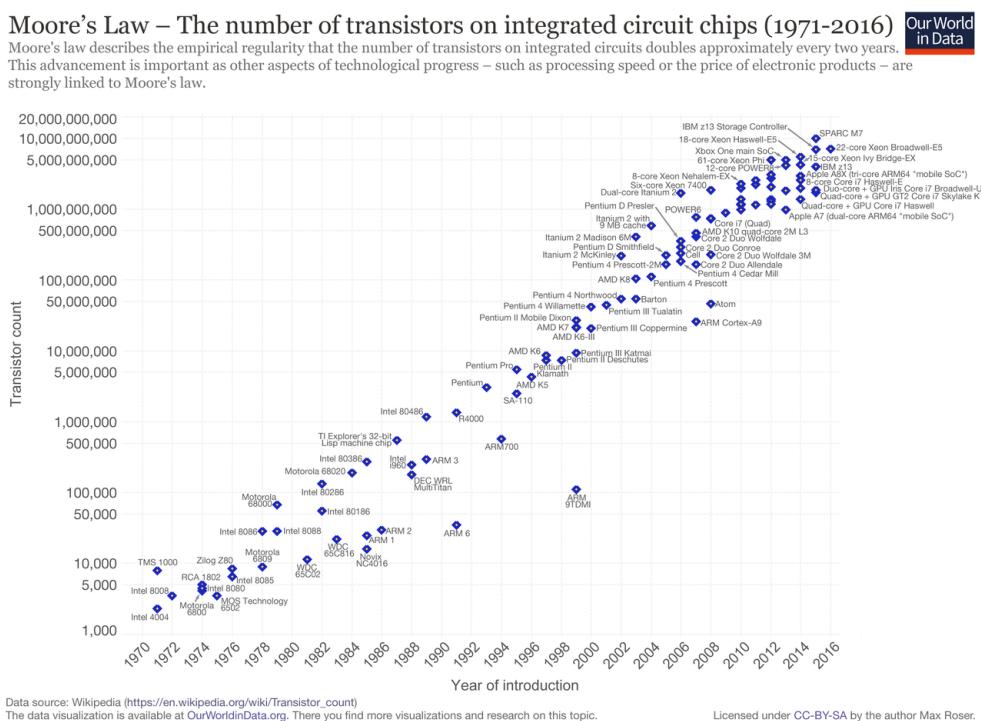
Η διαθεσιμότητα μεγάλου όγκου δεδομένων που αντιπροσωπεύουν καλά το φαινόμενο προς μελέτη, η κατάλληλη προεπεξεργασία πριν γίνει χρήση τους, και η επιλογή του κατάλληλου αλγορίθμου μηχανικής μάθησης είναι το κλειδί για την παραγωγή ενός επιτυχημένου προβλεπτικού μοντέλου.

## **1.2 Παράλληλα υπολογιστικά συστήματα**

Το λογισμικό των υπολογιστών για πολλά χρόνια ήταν φτιαγμένο έτσι ώστε να εκτελείται με αποκλειστικά σειριακό τρόπο. Ένα πρόβλημα διαιρείται σε κομμάτια-εντολές,

οι οποίες εκτελούνται η μία μετά την άλλη στον επεξεργαστή. Όταν τελειώσει η εκτέλεση μιας εντολής, ξεκινάει η εκτέλεση της επόμενης· με άλλα λόγια, δύο εντολές δεν εκτελούνται ποτέ ταυτόχρονα. Αυτή η προσέγγιση δουλεύει καλά, αλλά σπάνια είναι η ιδανική.

Για πολλές δεκαετίες, η ταχύτητα επεξεργασίας καθιστάται από τη συχνότητα του επεξεργαστή στον οποίο εκτελούνται οι εντολές, και έτσι η αύξηση της συχνότητας οδηγούσε σε ανάλογη αύξηση της ταχύτητας επεξεργασίας. Σύμφωνα με το νόμο του Moore, αναμένεται διπλασιασμός των τρανζίστορ ενός πυκνού ολοκληρωμένου κυκλώματος κάθε δύο χρόνια, πράγμα που σημαίνει συνεχή αύξηση της συχνότητας και της ταχύτητας υπολογισμού. Ωστόσο, κάποια χαρακτηριστικά του φυσικού κόσμου οριοθετούν το πόσο μπορεί να αυξηθεί πλέον η συχνότητα σε ένα επεξεργαστή. Τόσο η επίτευξη δημιουργίας ολοκληρωμένων κυκλωμάτων στα οποία η απόσταση μεταξύ δύο τρανζίστορ δεν μπορεί να μειωθεί πολύ περισσότερο, όσο και οι περιορισμοί που εισάγουν η αύξηση της θερμοκρασίας και της κατανάλωσης σε υψηλότερες συχνότητες, έγιναν τα βασικά κίνητρα πίσω από την αναζήτηση εναλλακτικών προσεγγίσεων για την αύξηση της ταχύτητας επεξεργασίας, και τελικά προέκυψαν οι αρχιτεκτονικές παράλληλης επεξεργασίας με χρήση περισσότερων της μίας μονάδων επεξεργασίας.



**Σχήμα 1.3:** Ο νόμος του Moore περιγράφει το ρυθμό αύξησης των τρανζίστορ στους επεξεργαστές σε σχέση με το χρόνο.

Τα φυσικά όρια βέβαια δεν είναι ο μόνος λόγος ανάπτυξης αυτών των αρχιτεκτονικών. Ορισμένα προβλήματα του πραγματικού κόσμου παρουσιάζουν εκ φύσεως

κάποιο είδος συγχρονισμού στα χαρακτηριστικά τους το οποίο μπορεί να αποδοθεί καλύτερα μέσω παράλληλων αρχιτεκτονικών. Για παράδειγμα, ο σχηματισμός γαλαξιών, τα καιρικά φαινόμενα, και η κυκλοφορία έχουν πολλά χαρακτηριστικά που ενδείκνυνται να μοντελοποιηθούν εν παραλλήλω.

Αυτός ο συνδυασμός συνθηκών έφερε άνοδο στο λεγόμενο παράλληλο προγραμματισμό, ή αλλιώς στον κλάδο της παράλληλης υπολογιστικής. Βασική ιδέα είναι η διαιρεση του προβλήματος σε κομμάτια που μπορούν να επεξεργάζονται ταυτόχρονα διαφορετικές μονάδες επεξεργασίας. Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές παράλληλης επεξεργασίας, βάσει των εντολών που εκτελούνται σε κάθε μονάδα επεξεργασίας και της πρόσβασης στη μνήμη που έχει η κάθε μονάδα.

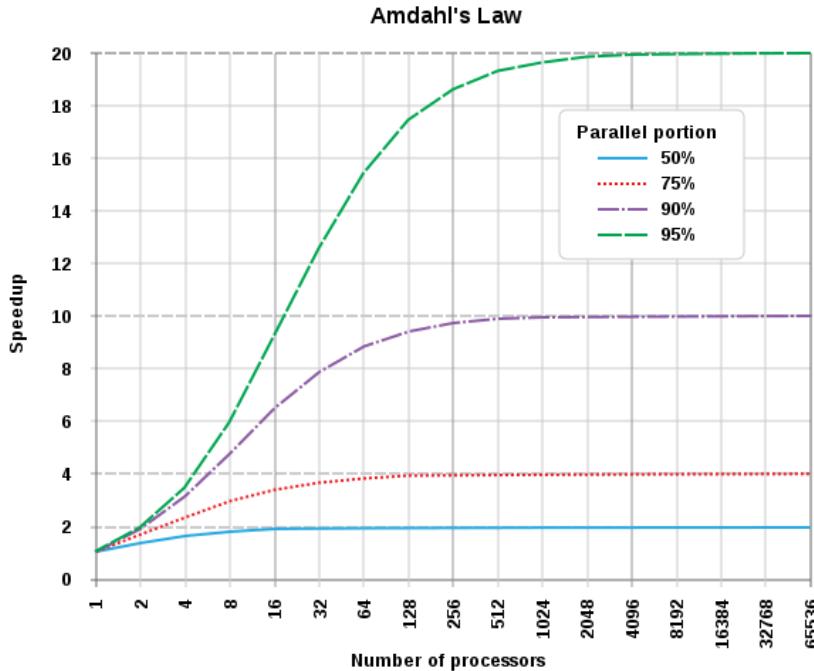
Εκτός των παραδοσιακών επεξεργαστών (CPU), η τελευταία δεκαετία έχει φέρει μεγάλη ανάπτυξη και στις μονάδες επεξεργασίας γραφικών (GPU). Αυτό είχε ως αποτέλεσμα την ανάπτυξη της λεγόμενης γενικού σκοπού υπολογιστικής σε μονάδες επεξεργασίας γραφικών (GPGPU), που στοχεύει στην επίλυση προβλημάτων όχι απαραίτητα σχετικών με γραφικά, και τα οποία τυπικά επιλύονται με εκτέλεση σε CPU, κάνοντας χρήση GPU. Το πλεονέκτημα των μονάδων GPU είναι ότι αν και η συχνότητα της κάθε μίας μονάδας ξεχωριστά είναι χαμηλότερη σε σχέση με αυτή μιας μονάδας CPU, υπάρχουν παρόλα αυτά πολλές περισσότερες από αυτές σε μία κάρτα γραφικών.

Μια σχέση συνεχούς ανάδρασης λαμβάνει χώρα μεταξύ των αρχιτεκτονικών παράλληλης επεξεργασίας που δημιουργήθηκαν αρχικά από ανάγκη και των προβλημάτων που επωφελούνται από αυτή, και έτσι η ανάπτυξη της παράλληλης υπολογιστικής βρίσκεται σε μεγάλη άνοδο τα τελευταία χρόνια. Ένα από αυτά τα επιστημονικά πεδία που επωφελούνται σημαντικά από την ύπαρξη πόρων παράλληλης επεξεργασίας είναι και η μηχανική μάθηση. Πολλές τεχνικές μηχανικής μάθησης παρουσιάζουν φυσικό παραλληλισμό μέσα τους, λόγω των πολύπλοκων μαθηματικών πράξεων σε αυτές και της επαναληπτικής φύσης τους.

### **1.2.1 Παράλληλος προγραμματισμός**

Όσο οι εφαρμογές εκτελούνται σε ένα μόνο επεξεργαστή, η απόδοσή τους εξαρτιόταν καθαρά από την ταχύτητα αυτού του επεξεργαστή, χωρίς να υπάρχει ιδιαίτερη απαίτηση σε ό,τι αφορά τον προγραμματισμό του λογισμικού πέραν της σωστής υλοποίησης του εκάστοτε αλγορίθμου. Η άφιξη των πολυπύρηνων επεξεργαστών έβαλε όμως τέλος στην πλήρη εξάρτηση από το υλικό, και δημιούργησε την ανάγκη βαθύτερης μελέτης της εφαρμογής που παράγεται με σκοπό την πλήρη εκμετάλλευση των πόρων του συστήματος στο οποίο θα εκτελεστεί αυτή.

Ο νόμος του Amdahl είναι ένα από τα σημαντικότερα κριτήρια παραλληλοποίησης μιας εφαρμογής. Σύμφωνα με αυτόν, η μέγιστη δυνατή επιτάχυνση μιας εφαρμογής μέσω παραλληλοποίησης περιορίζεται από τον υπόλοιπο σειριακό κώδικα. Με άλλα λόγια, το όφελος παραλληλοποίησης μεγιστοποιείται παραλληλοποιώντας όσο το δυνατόν περισσότερο κώδικα, και δίνοντας ιδιαίτερη προσοχή στα παραλληλοποιήσιμα τμήματα που είναι υπεύθυνα για το μεγαλύτερο μέρος του χρόνου εκτέλεσης.



Σχήμα 1.4: Σύμφωνα με το νόμο του Amdahl η μέγιστη δυνατή επιτάχυνση της εκτέλεσης ενός προγράμματος μέσω παραλληλοποίησης οριοθετείται από το ποσοστό μη-παραλληλοποίησιμου κώδικα σε αυτόν.

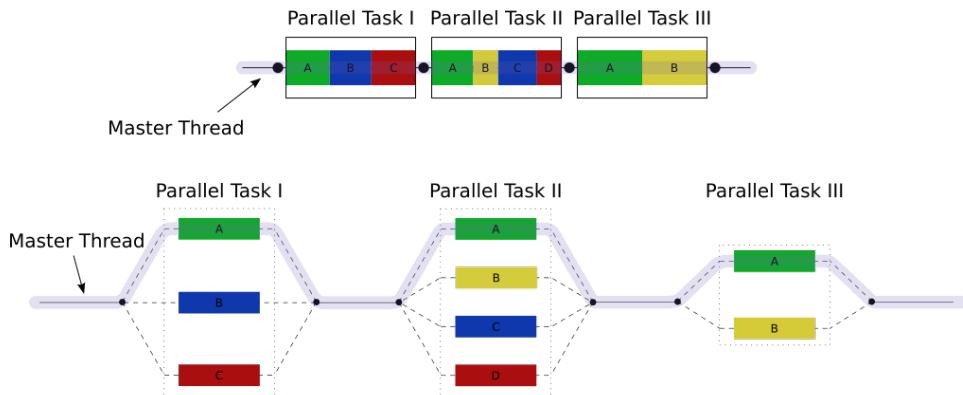
Τα θεωρητικά οφέλη που μπορούν να προκύψουν από την παραλληλοποίηση μιας εφαρμογής είναι αυτόδηλα. Ωστόσο, οι περισσότερες γλώσσες προγραμματισμού στερούνται έμφυτης λειτουργικότητας που να επιτρέπει στον προγραμματιστή να κάνει αυτή την παραλληλοποίηση πράξη. Ένας αριθμός προγραμματιστικών μοντέλων για γλώσσες όπως η C και η Fortran έχουν δημιουργηθεί για να καλύψουν αυτό το κενό. Καθένα από αυτά μπορεί να χρησιμοποιεί διαφορετικές συντακτικές συμβάσεις και να στοχεύει στην υλοποίηση διαφορετικών αρχιτεκτονικών παραλληλισμού μεταξύ διαφορετικών τύπων πολυπύρηνους επεξεργαστές.

Τα πρότυπα αυτά παρέχουν έναν αριθμό εργαλείων που μπορεί να χρησιμοποιηθούν από τον προγραμματιστή για να δηλώσει τις διεργασίες της εφαρμογής που προορίζονται προς παράλληλη εκτέλεση. Τα εργαλεία αυτά μπορεί να έχουν τη μορφή οδηγιών για το μεταφραστή και ρουτίνων βιβλιοθήκης που να αποσαφηνίζουν σε ένα συμβατό μεταφραστή ποιές περιοχές του προγράμματος να μεταφράσει σε παράλληλο

κώδικα μηχανής. Στη βιομηχανία σήμερα χρησιμοποιούνται κατά κόρον πρότυπα όπως το OpenMP, που αποβλέπει κυρίως στην εκτέλεση παράλληλου κώδικα σε CPU, τα CUDA και OpenCL που στοχεύουν στην εκμετάλλευση των πόρων GPU, ενώ έχουν αρχίσει να κάνουν την εμφάνισή τους και πρότυπα που εκμεταλλεύονται αμφότερους πολυπύρηνους επεξεργαστές και επιταχυντές υπολογισμών βασισμένους σε GPU, όπως το OpenACC. Στα πλαίσια της εργασίας θα γίνει χρήση του προτύπου OpenMP.

### 1.2.2 Το πρότυπο OpenMP

Το *OpenMP* είναι ένα API που στοχεύει στη διευκόλυνση του προγραμματισμού εφαρμογών που χρησιμοποιούν πολλαπλά νήματα. Προσφέρει μία σειρά οδηγιών για το μεταφραστή, ρουτινών βιβλιοθήκης, και μεταβλητών περιβάλλοντος που μπορούν εύκολα να χρησιμοποιηθούν από έναν προγραμματιστή C, C++ και Fortran στα πλαίσια της ανάπτυξης παράλληλων εφαρμογών εντός ενός χώρου διαμοιραζόμενης μνήμης. Στόχος του OpenMP είναι η επίτευξη παραλληλοποίησης της εφαρμογής με όσο το δυνατόν λιγότερες αλλαγές στον κώδικα της ενός νήματος εκδοχής του, και έχει δημιουργηθεί ως μία προτυποποίηση των πρακτικών παράλληλου προγραμματισμού σε χώρο διαμοιραζόμενης μνήμης που χρησιμοποιούνταν τα προηγούμενα χρόνια. Την ανάπτυξή του συντονίζει το OpenMP Architecture Review Board, και σε αυτή συμμετέχουν ηγέτες της βιομηχανίας όπως η Intel, η AMD και η IBM.



Σχήμα 1.5: Η διαδικασία εκτέλεσης μεταξύ πολλών νημάτων περιλαμβάνει τη δημιουργία νέων νημάτων από το master νήμα όταν αυτό κρίνεται απαραίτητο.

Η φιλοσοφία σχεδίασης του OpenMP στηρίζεται στην παραλληλοποίηση του υπάρχοντος συγχρονισμού μεταξύ πολλαπλών νημάτων. Εφ' όσον έχει λάβει την αντίστοιχη εντολή, το νήμα που έχει οριστεί ως master κάνει fork έναν αριθμό slave νημάτων και το σύστημα διανέμει την εργασία μεταξύ αυτών. Τα κομμάτια του κώδικα που προορίζονται προς παράλληλη εκτέλεση μαρκάρονται αντίστοιχα με χρήση οδηγιών που θα ερμηνεύσει ο μεταφραστής. Μοναδική απαίτηση του OpenMP είναι ένας

συμβατός μεταφραστής που να μπορεί να υλοποιήσει τη λειτουργικότητα των οδηγιών OpenMP που λαμβάνει. Οι περισσότεροι από τους πιο γνωστούς μεταφραστές όπως ο GCC, ο LLVM και οι μεταφραστές της Intel και της IBM παρέχουν υποστήριξη για όλες ή τις περισσότερες λειτουργίες του OpenMP.

Οι πρακτικές ανάπτυξης παράλληλων εφαρμογών μέσω OpenMP έχουν ως εξής:

- Αναπτύσσονται αρχικά οι αλγόριθμοι της εφαρμογής χωρίς να λαμβάνεται ενεργά υπόψιν η πρόθεση παραλληλοποίησής τους, και γίνεται μια αρχική αξιολόγηση της ορθότητας και της επίδοσης της εφαρμογής.
- Αφού επαληθευτεί ότι η εφαρμογή παράγει τα αναμενόμενα αποτελέσματα, εντοπίζονται τα κομμάτια που παίρνουν περισσότερο χρόνο κατά την εκτέλεση. Εκτός από χειροκίνητη επιμεώρηση του κώδικα, αυτή τη διαδικασία μπορούν να διευκολύνουν και εργαλεία όπως το PGI Profiler.
- Επιλέγονται οι περιοχές προς παραλληλοποίηση λαμβάνοντας υπόψιν το νόμο του Amdahl.
- Το OpenMP χρησιμοποιείται για παραλληλοποίηση των επιλεγμένων περιοχών του κώδικα. Αυτό σημαίνει μικρές αλλαγές στην υλοποίηση και προσθήκη των αντίστοιχων οδηγιών που καθιστούν ικανό το μεταφραστή να παραγάγει κώδικα μηχανής έτοιμο για εκτέλεση μεταξύ πολλών νημάτων.
- Η εφαρμογή αξιολογείται εκ νέου όσον αφορά την ορθότητα των αποτελεσμάτων και την απόδοσή της και η διαδικασία επαναλαμβάνεται έως ότου επιτευχθεί το επιθυμητό αποτέλεσμα.

Στα πλαίσια της χρήσης του με τις γλώσσες C / C++, το OpenMP έρχεται μαζί με μία βιβλιοθήκη `omp.h`, που εφόσον συμπεριληφθεί, παρέχει ορισμένες συναρτήσεις βασικότερες από τις οποίες είναι οι παρακάτω:

- `void omp_set_num_threads(n)`: Ορίζει τον αριθμό των νημάτων που γίνονται διαθέσιμα στην εφαρμογή.
- `int omp_get_thread_num()`: Επιστρέφει το ID του νήματος που κάλεσε την παρούσα συνάρτηση.
- `int omp_get_num_threads()`: Επιστρέφει τον αριθμό των νημάτων που τρέχουν τη δεδομένη στιγμή.
- `int omp_get_max_threads()`: Επιστρέφει τον μέγιστο αριθμό νημάτων που μπορούν να χρησιμοποιηθούν.
- `int omp_num_procs()`: Επιστρέφει τον αριθμό των φυσικών μονάδων επεξεργασίας.

- `bool omp_in_parallel()`: Επιστρέφει το αν η εκτέλεση του κώδικα βρίσκεται σε παράλληλη περιοχή.
- `double omp_get_wtime()`: Επιστρέφει το χρόνο τη δεδομένη στιγμή εκτέλεσης.

Ο βασικός τρόπος με τον οποίο μια περιοχή κώδικα σημαδεύεται προς παραλληλοποίηση είναι με χρήση των παρακάτω οδηγιών, χάρη στις οποίες ο μεταφραστής θα γνωρίζει πότε να παραγάγει παράλληλο κώδικα:

- `#pragma omp parallel`: Το block κώδικα που ακολουθεί θα εκτελεστεί από όλα τα διαθέσιμα νήματα. Με την επιπλέον χρήση του όρου `num_threads(n)` δίπλα στην οδηγία, ορίζεται ότι η εκτέλεση θα γίνει μόνο σε  $n$  νήματα.
- `#pragma omp for`: Με δεδομένο ότι εκτελείται μία παράλληλη περιοχή του κώδικα που έχει ξεκινήσει με την προηγούμενη οδηγία, γνωστοποιείται στον μεταφραστή ότι η εργασία του ακόλουθου βρόχου `for` μπορεί να γίνει εν παραλλήλω, ορίζοντας συγκεκριμένα νήματα ως υπεύθυνα για συγκεκριμένες επαναλήψεις.
- `#pragma omp parallel for`: Συνδυασμός των δύο παραπάνω.

Η φύση του OpenMP ως ένα πρότυπο που ακολουθεί το μοντέλο διαμοιραζόμενης μνήμης έχει ως αποτέλεσμα την εμφάνιση συνθηκών ανταγωνισμού οι οποίες μπορούν να προκαλέσουν αλλαγή του αποτελέσματος της εκτέλεσης ανάλογα με τη χρονοδρομολόγηση των νημάτων εκτέλεσης. Πρέπει να δίνεται μεγάλη προσοχή στις περιπτώσεις που πολλά νήματα επεξεργάζονται τα ίδια δεδομένα. Προς εξυπηρέτηση αυτής της ανάγκης υπάρχουν και οδηγίες μεταφραστή που είναι υπεύθυνες για το συγχρονισμό:

- `#pragma omp critical`: Η περιοχή του παράλληλου κώδικα που ακολουθεί θα εκτελείται μόνο από ένα νήμα τη φορά.
- `#pragma omp atomic`: Κάνει ακριβώς ότι και το `critical`, όμως αφορά μόνο την ανάγνωση ή την ανανέωση του περιεχομένου μιας θέσης μνήμης.
- `reduction`: Τίθενται ως επιπρόσθετος όρος στον ορισμό μιας παράλληλης περιοχής και ενημερώνει το μεταφραστή ότι ο πολλά νήματα μπορεί να αλλάξουν μια συγκεκριμένη μεταβλητή ταυτόχρονα. Αν υπάρχει ενδεχόμενο αύξησης της τιμής μιας μεταβλητής `var` για παράδειγμα από πολλά νήματα, θα δοθεί η οδηγία `#pragma omp parallel for reduction(+:var)`.
- `#pragma omp barrier`: Η εκτέλεση της περιοχής κώδικα μετά από μία οδηγία `barrier` δεν θα ξεκινήσει μέχρι να φτάσουν όλα τα νήματα σε αυτό το σημείο.
- `#pragma omp single`: Η περιοχή του παράλληλου κώδικα που ακολουθεί θα εκτελεστεί μόνο από ένα νήμα.

Ο κώδικας που παράγεται έχει μεγάλη φορητότητα τόσο ως προς το μεταφραστή στον οποίο θα μεταφραστεί όσο και ως προς το υλικό στο οποίο πρόκειται να εκτελεστεί. Έτσι, μπορεί να αναπτυχθεί μια παράλληλη εφαρμογή σε έναν προσωπικό υπολογιστή με διπύρηνο επεξεργαστή η οποία δύναται αργότερα να επωφεληθεί από εκτέλεση σε ένα workstation με επεξεργαστές 32 ή 64 πυρήνων που στοχεύουν σε επιστημονικούς υπολογισμούς. Αυτό κάνει το OpenMP ένα ιδιαίτερα ευέλικτο εργαλείο.

### 1.3 Βιβλιοθήκη λογισμικού

Η ανάπτυξη εφαρμογών οποιουδήποτε τύπου απαιτεί συνήθως τη χρήση δομών και μεθόδων που ενδέχεται να έχουν χρησιμοποιηθεί στο παρελθόν κατά την ανάπτυξη κάποιας άλλης εφαρμογής. Ιδιαίτερα σε περιπτώσεις στις οποίες οι συγκεκριμένες δομές και μέθοδοι εμπεριέχουν μέσα τους ιδιαίτερα μεγάλη πολυπλοκότητα, η επανυλοποίησή τους μπορεί να αποτελέσει τροχοπέδη για την ανάπτυξη μιας απλής εφαρμογής. Την ανάγκη δυνατότητας επαναχρησιμοποίησης και διακίνησης υλοποιήσεων λογισμικού έρχονται να καλύψουν οι βιβλιοθήκες λογισμικού.

Μια βιβλιοθήκη λογισμικού αποτελεί μία συλλογή υλοποιημένων δομών και μεθόδων με σαφώς ορισμένα χαρακτηριστικά ως προς τον τρόπο χρήσης και λειτουργίας τους. Προσφέρουν στον προγραμματιστή εύχρηστες διεπαφές, επιτρέποντάς του να μην χρειάζεται να απασχοληθεί με τις λεπτομέρειες υλοποίησης των μεθόδων που χρησιμοποιεί, αλλά μόνο την είσοδο και την έξοδό τους. Οι βιβλιοθήκες λογισμικού στηρίζονται στην ιδέα της διανομής, της επαναχρησιμοποίησης, και της συνεχούς ανάπτυξης από ομάδες ενός ή περισσοτέρων προγραμματιστών, συχνά στα πλαίσια του ανοιχτού λογισμικού.

Διαφορετικές βιβλιοθήκες λογισμικού διαφέρουν ως προς την γλώσσα στην οποία είναι φτιαγμένες, και τις πλατφόρμες στις οποίες προορίζονται να χρησιμοποιηθούν. Έτσι, μπορεί να υπάρχουν βιβλιοθήκες C++, Python, ή Java, βιβλιοθήκες με υλοποιήσεις που στοχεύουν σε υπολογιστές χαμηλής κατανάλωσης ή σε μεγάλα workstations, και βιβλιοθήκες που κάνουν χρήση συγκεκριμένων αρχιτεκτονικών επεξεργασίας. Η ποιότητα των υλοποιήσεων που μπορεί να προσφέρει μία βιβλιοθήκη, καθώς και η ευχολία χρήσης της είναι επίσης κριτήρια χρήσης της έναντι μιας άλλης.

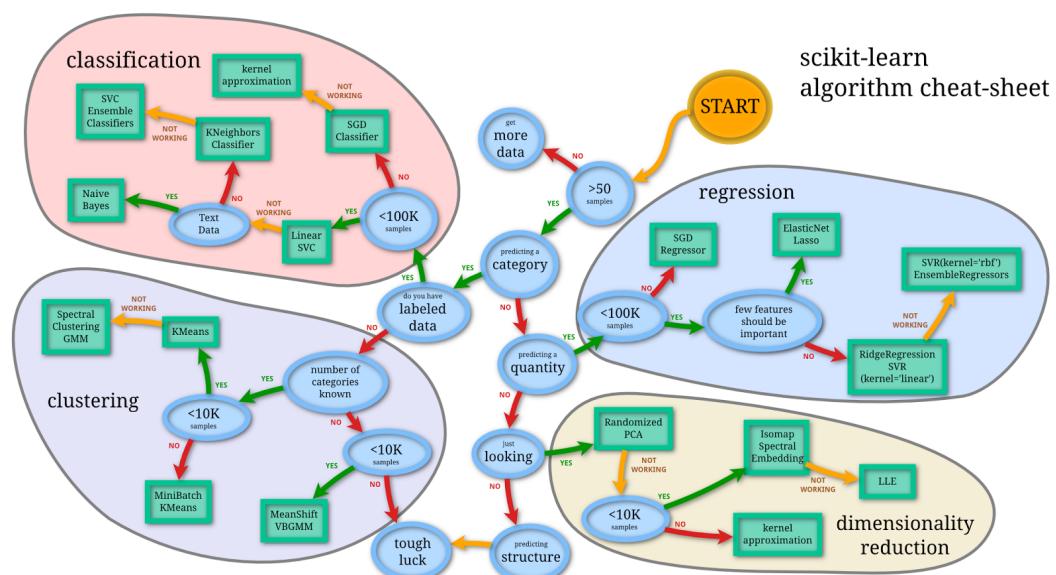
Καθώς η μηχανική μάθηση χρησιμοποιείται ολοένα και περισσότερο τόσο στην έρευνα όσο και στη βιομηχανία, προκύπτουν συνεχώς νέες βιβλιοθήκες λογισμικού που υλοποιούν μεθόδους μηχανικής μάθησης. Οι βιβλιοθήκες αυτές διαφέρουν συχνά ως προς τις εφαρμογές πάνω στις οποίες πρόκειται να χρησιμοποιηθούν, καθώς και σε τεχνικά ζητήματα όπως η γλώσσα προγραμματισμού για την οποία έχουν γραφτεί και

ο τρόπος υλοποίησης των αλγορίθμων που προσφέρουν.

### 1.3.1 Βιβλιοθήκες μηχανικής μάθησης στη βιομηχανία

Στα πλαίσια κατασκευής της βιβλιοθήκης της εργασίας δοκιμάστηκαν διάφορες υπάρχουσες βιβλιοθήκες με σκοπό τον προσδιορισμό κοινών σημείων τους, πρακτικών χρήσης που επικαλούνται, και φυσικά τη σύγχριση των αποτελεσμάτων και της επίδοσής τους. Δόθηκε ιδιαίτερη σημασία στις βιβλιοθήκες γενικού σκοπού scikit-learn και mlpack, ορισμένα χαρακτηριστικά των οποίων θα παρουσιαστούν παρακάτω.

Η scikit-learn είναι μια βιβλιοθήκη μηχανικής μάθησης για τη γλώσσα προγραμματισμού Python. Είναι ιδιαίτερα απλή, προσβάσιμη και χρήσιμη στα πλαίσια της εξόρυξης και ανάλυσης δεδομένων. Υλοποιεί μια σειρά αλγορίθμων ταξινόμησης, παλινδρόμησης και ομαδοποίησης, και αποτελεί συνεπώς μια βιβλιοθήκη γενικού σκοπού. Είναι γραμμένη κυρίως σε Python και χρησιμοποιεί για τις ανάγκες της τις αντίστοιχες αριθμητικές και επιστημονικές βιβλιοθήκες NumPy και SciPy. Παρέχεται υπό την άδεια BSD ως ανοιχτό λογισμικό.



Σχήμα 1.6: Μερικές από τις μεθόδους που παρέχει η βιβλιοθήκη scikit-learn.

Η mlpack έχει γραφτεί σε C++ και είναι επίσης μία βιβλιοθήκη γενικού σκοπού. Δίνει έμφαση στην ταχύτητα, την ευκολία χρήσης και την επεκτασιμότητα, παρέχοντας ένα απλό και συνεπές API που μπορεί να χρησιμοποιήσει χρήστης κάθε επιπέδου. Παρέχει ιδιαίτερα αποδοτικές υλοποιήσεις για πολλές από τις βασικές μεθόδους της βιομηχανίας, και χρησιμοποιεί τη βιβλιοθήκη Armadillo για τις πράξεις γραμμικής άλγεβρας. Εκτός από τη C++ παρέχει και binding για τις γλώσσες R και Python. Διανέμεται υπό την άδεια BSD ως ανοιχτό λογισμικό με συνεισφέροντες από όλο τον κόσμο.

## what does mlpack implement?

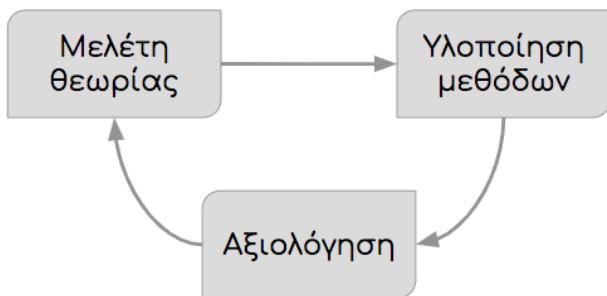
Below is a high-level list of the available functionality contained within **mlpack**, along with relevant links to papers, API documentation, tutorials, or other references.

- Collaborative filtering (with many decomposition techniques) ([api](#), [wiki](#))
- Decision stumps (one-level decision trees) ([api](#), [paper](#))
- Density estimation trees ([api](#), [tutorial](#), [paper](#))
- Euclidean minimum spanning tree calculation ([api](#), [tutorial](#), [paper](#))
- Gaussian mixture models ([api](#), [wiki](#))
- Hidden Markov models ([api](#), [wiki](#))
- Kernel Principal Components Analysis (optionally with sampling) ([api](#), [paper](#), [wiki](#))
- k-Means clustering (with several accelerated algorithms) ([api](#), [tutorial](#), [wiki](#))
- Least-angle regression (LARS/LASSO) ([api](#), [paper](#), [wiki](#))
- Linear regression (simple least-squares) ([api](#), [wiki](#))
- Local coordinate coding ([api](#), [paper](#))
- Locality-sensitive hashing for approximate nearest neighbor search ([api](#), [paper](#), [wiki](#))
- Logistic regression ([api](#), [wiki](#))
- Max-kernel search ([api](#), [tutorial](#), [paper](#))
- Naive Bayes classifier ([api](#), [wiki](#))
- Nearest neighbor search with dual-tree algorithms ([api](#), [tutorial](#), [paper](#), [wiki](#))
- Neighborhood components analysis ([api](#), [paper](#), [wiki](#))
- Non-negative matrix factorization ([api](#), [paper](#), [wiki](#))
- Perceptrons ([api](#), [wiki](#))
- Principal components analysis (PCA) ([api](#), [wiki](#))
- RADICAL (independent components analysis) ([api](#), [paper](#))
- Range search with dual-tree algorithms ([api](#), [tutorial](#), [paper](#), [wiki](#))
- Rank-approximate nearest neighbor search ([api](#), [paper](#))
- Sparse coding with dictionary learning ([api](#), [paper](#), [wiki](#))

Σχήμα 1.7: Μέθοδοι που παρέχει η βιβλιοθήκη mlpack σύμφωνα με την επίσημη ιστοσελίδα της.

## 1.4 Η βιβλιοθήκη «metis» της εργασίας

Για τη βιβλιοθήκη της εργασίας επιλέχθηκε να γίνει υλοποίηση ενός αριθμού βασικών και αντιπροσωπευτικών μεθόδων μηχανικής μάθησης. Ο τρόπος εργασίας απαρτίζεται από έναν βρόχο που περιλαμβάνει διαδοχικά μελέτη της θεωρίας, υλοποίηση κάποιου αλγορίθμου, αξιολόγησή του, και επανάληψη μέχρι να παραχθεί το επιθυμητό αποτέλεσμα.



Οι μέθοδοι που υλοποιήθηκαν είναι οι παρακάτω:

- Γραμμική παλινδρόμηση
- Λογιστική παλινδρόμηση
- Νευρωνικά δίκτυα εμπρόσθιας τροφοδότησης
- Απλός ταξινομητής Bayes
- Εκμάθηση δέντρων αποφάσεων
- Ομαδοποίηση  $k$ -μέσων

Οι υλοποιήσεις έγιναν στη γλώσσα C++ με χρήση των συμβάσεων του αντικειμενοστραφούς προγραμματισμού. Κατά το σχεδιασμό της βιβλιοθήκης έγινε προσπάθεια δημιουργίας ενός εύχρηστου API που παραμένει συνεπές μεταξύ των διαφορετικών κλάσεων και μεθόδων που παρέχονται. Επειδή πολλές μέθοδοι μηχανικής μάθησης απαιτούν πράξεις γραμμικής άλγεβρας, επιλέχθηκε να γίνει χρήση της βιβλιοθήκης Eigen, η οποία υλοποιεί αποδοτικά πράξεις όπως οι πολλαπλασιασμοί πινάκων, και της οποίας οι δομές χρησιμοποιούνται για αποθήκευση των δεδομένων. Για την παραλληλοποίηση των υλοποιημένων μεθόδων χρησιμοποιείται το πρότυπο OpenMP. Η ανάπτυξη της βιβλιοθήκης έγινε με τρόπο τέτοιο ώστε ο κώδικας της να είναι ευανάγνωστος και να διευκολύνεται η συντήρηση και η περαιτέρω ανάπτυξή του.

#### 1.4.1 Διάρθρωση της αναφοράς

Στα πλαίσια παρουσίασης της εργασίας που έγινε για τη δημιουργία της βιβλιοθήκης, το παρόν σύγγραμμα έχει χωριστεί σε έξι κεφάλαια.

- Στον **πρόλογο** γίνεται μία παρουσίαση βασικών θεμάτων που αφορούν τη μηχανική μάθηση και την παράλληλη υπολογιστική.
- Στο κεφάλαιο του **θεωρητικού υποβάθρου** αναπτύσσονται οι τεχνικές μηχανικής μάθησης που υλοποιούνται στη βιβλιοθήκη.
- Στο κεφάλαιο **σχεδιασμού του API** παρουσιάζονται οι κλάσεις που αναπτύχθηκαν και οι διεπαφές με τις οποίες μπορεί να τις χρησιμοποιήσει ο χρήστης.
- Στο κεφάλαιο των **λεπτομερειών υλοποίησης** εξηγούνται κομμάτια του κώδικα και δικαιολογούνται ορισμένες επιλογές.
- Στο κεφάλαιο των **εφαρμογών** παρουσιάζονται μερικές απλές εφαρμογές που αποσκοπούν στην επίδειξη της βιβλιοθήκης.
- Στον **επίλογο** αναφέρονται τα συμπεράσματα και η μελλοντική εργασία που θα γίνει πάνω στη βιβλιοθήκη.



# Κεφάλαιο 2

## Θεωρητικό υπόβαθρο

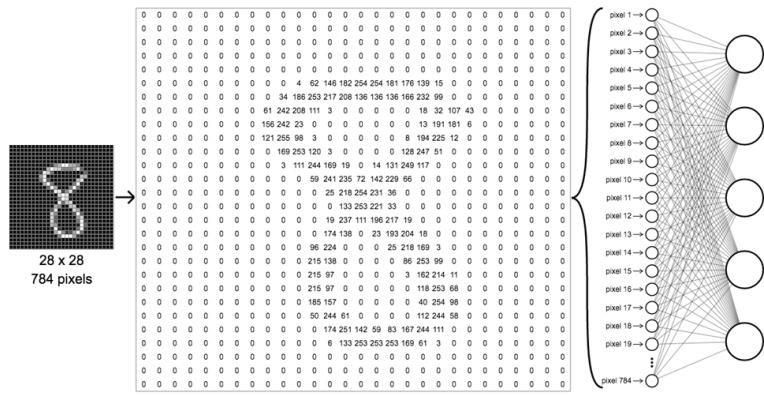
Η μελέτη και κατανόηση της θεωρίας πίσω από τις διάφορες δομές και αλγορίθμους μηχανικής μάθησης που υλοποιήθηκαν αποτέλεσε ένα από τα σημαντικότερα κομμάτια της εργασίας. Σε αυτό το κεφάλαιο θα γίνει μία εν συντομίᾳ ανάπτυξη αυτής της θεωρίας. Τα θέματα θα παρουσιαστούν κατά κύριο λόγο στα πλαίσια που σχετίζονται με το σχεδιασμό και την υλοποίηση της βιβλιοθήκης, συνεισφέροντας στην κατανόηση του περιεχομένου των επόμενων κεφαλαίων.

### 2.1 Δεδομένα και μηχανική μάθηση

Η ικανότητα ενός μοντέλου μηχανικής μάθησης να κάνει προβλέψεις για ορισμένα χαρακτηριστικά ενός συστήματος ή φαινομένου δεδομένων κάποιων άλλων χαρακτηριστικών του στηρίζεται στην υπόθεση ότι προηγούμενες εμπειρίες και παρατηρήσεις μπορούν να χρησιμοποιηθούν για τον σκοπό της μάθησης. Ως «μάθηση» αναφέρεται η εκπαίδευση ή η προσαρμογή ενός μαθηματικού μοντέλου πάνω σε ένα σύνολο δεδομένων το οποίο έχει προκύψει από αυτές τις εμπειρίες και παρατηρήσεις.

Τα δεδομένα που χρησιμοποιούνται σε διαφορετικές εφαρμογές ποικίλουν σε μορφή, ποσότητα και περιεχόμενο. Μπορούν να έχουν μορφή εικόνας, ήχου, κειμένου ή οποιουδήποτε άλλου μέσου, και η συλλογή τους είναι ευθύνη της επιστήμης στην οποία υπάγονται. Έτσι, τα δεδομένα μιας ιατρικής εφαρμογής μπορούν να έχουν συλλεχθεί από χλινικές μελέτες, μία χλιματολογική εφαρμογή μπορεί να έχει παρατηρήσεις για χαρακτηριστικά του καιρού με την πάροδο του χρόνου, ενώ μία διαστημική εφαρμογή ενδέχεται να χρησιμοποιεί ως δεδομένα εικόνες υψηλής ανάλυσης από τηλεσκόπια.

Για να γίνουν χρήσιμα στους διάφορους αλγορίθμους μηχανικής μάθησης, τα δεδομένα αυτά λαμβάνουν συνήθως κάποια αριθμητική αναπαράσταση που προκύπτει μετά από επεξεργασία τους. Η αριθμητική τιμή κάθε χαρακτηριστικού μπορεί να είναι



Σχήμα 2.1: Μία εικόνα μπορεί να αναπαρασταθεί διαφορετικά ως τιμές φωτεινότητας των pixel που την αποτελούν.

διακριτή ή συνεχής, και να αποτελεί ακέραιο ή πραγματικό αριθμό. Η δομή που λαμβάνουν τα δεδομένα είναι συνήθως αυτή ενός πίνακα με τόσες γραμμές όσες και τα παραδείγματα ή περιστατικά και τόσες στήλες όσες τα χαρακτηριστικά τα οποία μετρούνται. Ένα μοναδικό παράδειγμα αποτελεί ουσιαστικά ένα διάνυσμα με μέγεθος που καθορίζεται από τον αριθμό των χαρακτηριστικών του.

Στην επιβλεπόμενη μάθηση γίνεται συχνά διάκριση μεταξύ δεδομένων εισόδου και δεδομένων εξόδου. Τα δεδομένα εισόδου αναφέρονται και ως ανεξάρτητες ή επεξηγηματικές μεταβλητές, ενώ τα δεδομένα εξόδου ονομάζονται αλλιώς εξαρτημένες μεταβλητές ή μεταβλητές απόκρισης. Στην περίπτωση της ταξινόμησης, οι διάφορες κλάσεις ή κατηγορίες στις οποίες μπορούν να ανήκει το κάθε παράδειγμα, αποτελούν συνήθως την έξοδο ή «ετικέτα» του αντίστοιχου παραδείγματος. Σκοπός των αλγορίθμων μηχανικής μάθησης υπό επίβλεψη είναι η μοντελοποίηση της σχέση μεταξύ των δεδομένων εισόδου και εξόδου.

Προς εξυπηρέτηση της παραγωγής του καλύτερου δυνατού μοντέλου, είναι πολύ συχνό τα δεδομένα να διαιρούνται σε τρία υποσύνολα. Πρώτο είναι το σύνολο εκπαίδευσης, το οποίο χρησιμοποιείται για την εκπαίδευση αυτή καθαυτή. Δεύτερο είναι το σύνολο επικύρωσης, πάνω στο οποίο αξιολογείται το σύστημα που εκπαιδεύτηκε με διαφορετικές αρχιτεκτονικές και παραμέτρους. Η ικανότητα του μοντέλου που επιλέγεται τελικά να ανταποκριθεί στη γενική περίπτωση, επιβεβαιώνεται μέσω αξιολόγησής του πάνω στο συνόλου δοκιμής. Σε περίπτωση που τα παραδείγματα του συνόλου δεδομένων δε φτάνουν για την παραγωγή και των τριών υποσυνόλων, κάποιο από τα υποσύνολα επικύρωσης και δοκιμής παραλείπεται χωρίς να αλλάξει σημαντικά η διαδικασία εκπαίδευσης. Αυτό θα έχει φυσικά ως αποτέλεσμα την παραγωγή ενός λιγότερο έμπιστου μοντέλου.

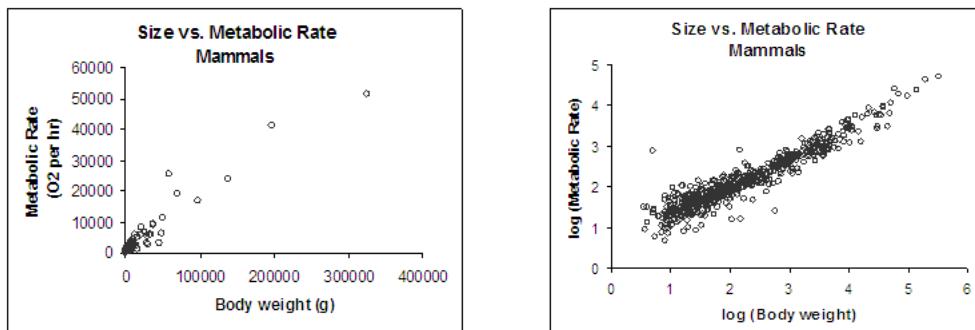
Η χρήση ενός μεγάλου και αντιπροσωπευτικού συνόλου δεδομένων για το φα-

νόμενο που μοντελοποιείται είναι το Άλφα και το Ωμέγα για την επιτυχημένη εκπαίδευση ενός μοντέλου μηχανικής μάθησης, ενώ η απουσία του αποτελεί εμπόδιο που δύσκολα ξεπερνιέται. Ο συνδυασμός δε ενός καλού συνόλου δεδομένων με τις κατάλληλες τεχνικές μηχανικής μάθησης δύναται να παραγάγει ένα μοντέλο που αντιπροσωπεύει σε σημαντικό βαθμό την πραγματικότητα και το οποίο μπορεί να κάνει ακριβείς προβλέψεις.

### 2.1.1 Προετοιμασία των δεδομένων

Μία σημαντική λεπτομέρεια που κάνει συχνά τη διαφορά στην επιτυχία εκπαίδευσης ενός συστήματος μηχανικής μάθησης είναι η επεξεργασία των δεδομένων πριν γίνει χρήση τους. Οι τύποι επεξεργασίας των δεδομένων ποικίλουν σε μεθοδολογία και στόχο, και δεν είναι πάντα προφανές ποιοί από αυτούς πρέπει να γίνουν. Η απεικόνιση των δεδομένων μπορεί να βοηθήσει στην επιλογή, και είναι συνηθισμένο να δοκιμάζονται διαφορετικές προσεγγίσεις έως ότου βρεθεί η πιο επιτυχημένη.

Ένα παράδειγμα στο οποίο η προεπεξεργασία των δεδομένων είναι συχνά αναπόφευκτη είναι τα γραμμικά μοντέλα. Η υπόθεση γραμμικότητας μεταξύ των επεξηγηματικών μεταβλητών που κάνουν αλγόριθμοι όπως αυτός της γραμμικής παλινδρόμησης σημαίνει ότι αναμένεται ο μέσος της μεταβλητής απόχρισης να είναι ένας γραμμικός συνδυασμός των παραμέτρων και των επεξηγηματικών μεταβλητών. Στην πραγματικότητα, αυτή η υπόθεση αποτελεί περιορισμό μόνο για τις παραμέτρους της γραμμικής προγνωστικής συνάρτησης, καθώς ακόμα και στην περίπτωση μη-γραμμικής σχέσης μεταξύ εισόδου και εξόδου, μπορεί να γίνει μετασχηματισμός των δεδομένων. Για παράδειγμα, μπορεί εύκολα να εφαρμοστεί λογαριθμικός μετασχηματισμός σε δεδομένα που φαίνεται να έχουν εκθετική σχέση.



Σχήμα 2.2: Λογαριθμικός μετασχηματισμός σε δεδομένα των οποίων η είσοδος έχει εκθετική σχέση με την έξοδο.

Ένας από τους πιο δημοφιλείς μετασχηματισμούς που μπορεί να γίνει στα δεδομένα είναι η κανονικοποίηση, και ιδιαίτερα το λεγόμενο standardization. Η εφαρμογή

του έχει σχεδόν πάντα σαν αποτέλεσμα την ταχύτερη και αποδοτικότερη εκπαίδευση του συστήματος μηχανικής μάθησης. Από μαθηματικής άποψης, το standardization αποτελεί ένα μετασχηματισμό στα δεδομένα κάθε ενός χαρακτηριστικού ως εξής:

$$x' = \frac{x - \mu}{\sigma}$$

Όπου:

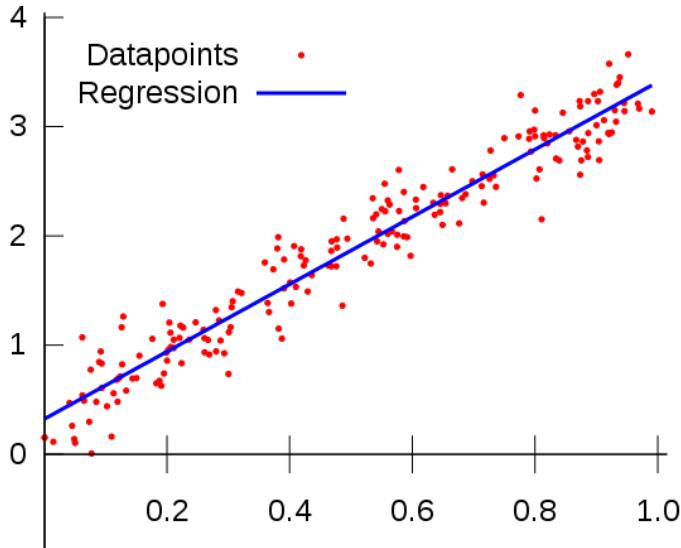
- $x$ , η τιμή μιας μεταβλητής ενός παραδείγματος των δεδομένων
- $x'$ , ο εκ νέου μετασχηματισμός αυτής της τιμής
- $\mu$ , ο μέσος όρος των τιμών του συγκεκριμένου χαρακτηριστικού επί όλου του συνόλου δεδομένων
- $\sigma$ , η τυπική απόκλιση του συγκεκριμένου χαρακτηριστικού επί όλου του συνόλου δεδομένων

Άλλοτε δε, είναι χρήσιμο να γίνει και ανακλιμακωποίηση των δεδομένων έτσι ώστε η κάθε μεταβλητή κάθε παραδείγματος να βρίσκεται γύρω από χοντινές τιμές.

## 2.2 Γραμμική παλινδρόμηση

Ένα από τα επιστημονικά πεδία του οποίου οι τεχνικές έχουν ιδιαίτερα μεγάλη χρηστικότητα στη μηχανική μάθηση είναι η στατιστική. Σε αυτή την ενότητα γίνεται ανάπτυξη της τεχνικής της γραμμικής παλινδρόμησης, που αποτελεί μία γραμμική προσέγγιση στη μοντελοποίηση της σχέσης μεταξύ μιας ή περισσοτέρων μεταβλητών εισόδου και κάποιας μεταβλητής εξόδου. Η συγκεκριμένη μέθοδος είναι μια τεχνική επιβλεπόμενης μάθησης και έχει μακρά ιστορία στα πλαίσια της στατιστικής. Με μια σειρά προσαρμογών αποτελεί ισχυρότατο εργαλείο και για τη μηχανική μάθηση.

Ως γραμμικό εννοείται ένα μοντέλο που επιχειρεί να ποσοτικοποιήσει την εξάρτηση μιας μεταβλητής απόχρισης ή εξόδου με μια σειρά επεξηγηματικών μεταβλητών ή μεταβλητών εισόδου, κάνοντας την υπόθεση γραμμικής σχέσης μεταξύ των δύο. Η εξάρτηση αυτή αποτυπώνεται μέσω της γραμμικής προγνωστικής συνάρτησης, της οποίας οι παράμετροι υπολογίζονται «προσαρμόζοντας» το μοντέλο σε ένα σύνολο παραδειγμάτων εκπαίδευσης. Με άλλα λόγια γίνεται υπολογισμός της ευθείας (ή σε περίπτωση περισσότερων διαστάσεων, του επιπέδου ή υπερεπιπέδου) που ταιριάζει καλύτερα στα δεδομένα και ελαχιστοποιεί την απόκλιση από αυτά. Όταν υπάρχει μία μεταβλητή εισόδου, γίνεται λόγος για απλή γραμμική παλινδρόμηση, ενώ σε διαφορετική περίπτωση πρόκειται για παλινδρόμηση πολλών εισόδων.



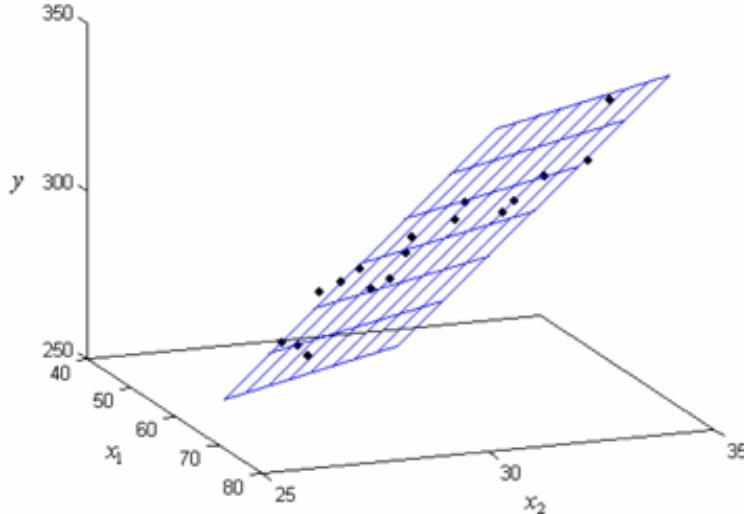
Σχήμα 2.3: Γραφμικό μοντέλο προσαρμοσμένο σε δεδομένα με μία επεξηγηματική μεταβλητή.

Παρ' όλη την απλότητα της προσέγγισης, η γραφμική παλινδρόμηση αποβαίνει συχνά ιδιαίτερα χρήσιμη, ιδίως σε προβλήματα για τα οποία διατίθεται μεγάλος αριθμός παραδειγμάτων και μπορεί να συντεθεί ένα αξιόπιστο μοντέλο. Το μοντέλο που προκύπτει ύστερα από την εκπαίδευση μπορεί να χρησιμοποιηθεί για πρόβλεψη της μεταβλητής απόχρισης ενός διανύσματος προγνωστικών μεταβλητών εισόδου των οποίων η έξοδος είναι άγνωστη. Μπορεί επιπλέον να χρησιμοποιηθεί για να επιδείξει πόσο ισχυρή ή ανίσχυρη είναι η γραφμική σχέση μιας ή περισσοτέρων επεξηγηματικών μεταβλητών με τη μεταβλητή απόχρισης, εξηγώντας έτσι την παραλλαγή των τιμών της απόχρισης σε σχέση με την παραλλαγή των τιμών της εισόδου και αναγνωρίζοντας το υποσύνολο των επεξηγηματικών μεταβλητών που περιέχουν περιττή για το φαινόμενο που μελετάται πληροφορία.

Τη πάρχει ένας αριθμός διαφορετικών μεθόδων εκπαίδευσης ενός γραφμικού μοντέλου, καθένας εκ των οποίων ενδέικνυται για διαφορετικές περιπτώσεις, χυρίως βάσει του μεγέθους του συνόλου δεδομένων και του αριθμού επεξηγηματικών μεταβλητών σε αυτό. Θα εξεταστεί η βασικότερη από αυτές, που είναι η μέθοδος των ελαχίστων τετραγώνων.

### 2.2.1 Το γραφμικό μοντέλο

Το γραφμικό μοντέλο χρησιμοποιεί για τις προβλέψεις του τη λεγόμενη γραφμική προγνωστική συνάρτηση:



Σχήμα 2.4: Επίπεδο που έχει προσαρμοστεί σε δεδομένα με δύο επεξηγηματικές μεταβλητές.

$$y = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

Διαθέτοντας ένα σύνολο δεδομένων  $n$  παραδειγμάτων  $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$ ,  $p$  διαστάσεων, το γραμμικό μοντέλο υποθέτει ότι η σχέση μεταξύ της εξαρτημένης μεταβλητής  $y$  και των ανεξάρτητων μεταβλητών  $\widehat{x}_i^T$ , είναι γραμμική, θέτοντας επιπλέον έναν όρο σφάλματος ή διαταραχής  $\varepsilon_i$  που προσθέτει θόρυβο σε αυτή τη γραμμική σχέση, και εξηγεί τη διακύμανση που παρατηρείται στην προσεγγιστικά γραμμική σχέση. Έτσι:

$$y_i = \theta_0 + \theta_1 x_{i1} + \dots + \theta_n x_{in} + \varepsilon_i$$

Η σε μορφή διανύσματος:

$$y_i = \widehat{x}_i^T \widehat{\theta} + \varepsilon_i$$

Όπου:

- $y_i$ , η εξαρτημένη μεταβλητή ή μεταβλητή απόκρισης του παραδείγματος  $i$
- $\widehat{x}_i^T$ , το διάνυσμα των εξαρτημένων ή επεξηγηματικών μεταβλητών του παραδείγματος  $i$
- $\widehat{\theta} = (\theta_0 \theta_1 \dots \theta_p)^T$ , το διάνυσμα των παραμέτρων της γραμμικής προγνωστικής συνάρτησης

- $\varepsilon_i$ , ο όρος σφάλματος ή διαταραχής για το παράδειγμα  $i$

Όπως φαίνεται, η διάσταση των διανυσμάτων  $\widehat{x}_i$  και  $\widehat{\theta}$  είναι  $(p + 1)$ , ενώ ο αριθμός των ανεξάρτητων μεταβλητών στην είσοδο είναι  $p$ . Η πρόσθετη παράμετρος  $\theta_0$  ονομάζεται «απόκλιση» και είναι σαν να πολλαπλασιάζεται με έναν επιπλέον όρο εισόδου  $x_0 = 1$ . Αυτή η πρόσθετη παράμετρος προσθέτει έναν παραπάνω βαθμός ελευθερίας που επιτρέπει τη μετακίνηση του γραμμικού μοντέλου, παραδείγματος χάριν μιας ευθείας στην περίπτωση της μίας διάστασης, πάνω ή κάτω.

Σκοπός της γραμμικής παλινδρόμησης είναι φυσικά η εύρεση των τιμών του διανύσματος παραμέτρων. Το μέγεθος κάθε συντελεστή  $\theta_j$  υποδηλώνει το βαθμό εξάρτησης της εξαρτημένης μεταβλητής  $y$  από την αντίστοιχη ανεξάρτητη μεταβλητή  $x_j$ . Αυτή η πληροφορία είναι χρήσιμη σε περίπτωση που επιδιώκεται μείωση των διαστάσεων του προβλήματος, αφαιρώντας τις ανεξάρτητες μεταβλητές των οποίων η αντίστοιχη παράμετρος τείνει στο 0.

### 2.2.2 Μέθοδος ελαχίστων τετραγώνων

Η πιο γνωστή μέθοδος υπολογισμού των παραμέτρων είναι αυτή των ελαχίστων τετραγώνων (συναντάται συχνά ως OLS στη διεύθυνη βιβλιογραφία), μία τεχνική που προέρχεται από τη στατιστική και έχει μελετηθεί και χρησιμοποιηθεί εκτεταμένα. Η μέθοδος των ελαχίστων τετραγώνων είναι αναλυτική λύση που δίνει απευθείας τις ακριβείς παραμέτρους του γραμμικού μοντέλου. Η μέθοδος ελαχιστοποιεί τα τετραγωνισμένα υπόλοιπα και οδηγεί σε μία λύση κλειστής μορφής για τις άγνωστες παραμέτρους:

$$\widehat{\theta} = (X^T X)^{-1} X^T y$$

$$\widehat{\theta} = (\sum \widehat{x_i x_i^T})^{-1} (\sum \widehat{x_i y_i})$$

Η μέθοδος των ελαχίστων τετραγώνων είναι εννοιολογικά απλή και εύκολα διαχειρίσιμη υπολογιστικά. Από προγραμματιστικής άποψης μπορεί να υλοποιηθεί εύκολα με βελτιστοποιημένα πακέτα γραμμικής άλγεβρας όπως το Eigen ή το Armadillo στην περίπτωση της C++.

### 2.2.3 Απλή γραμμική παλινδρόμηση

Στην περίπτωση της απλής γραμμικής παλινδρόμησης, δηλαδή την περίπτωση μιας μόνο επεξηγηματικής μεταβλητής και μιας μεταβλητής απόκρισης, προκύπτει μία ακόμα πιο απλή ειδική περίπτωση της μεθόδου ελαχίστων τετραγώνων. Γίνεται κατανοητό ότι σε αυτή την περίπτωση το μοντέλο που επιδιώκεται να προσαρμοστεί είναι το εξής:

$$y = \theta_0 + \theta_1 x_1$$

Όπως και στο OLS σκοπός είναι η ελαχιστοποίηση των τετραγωνισμένων υπολοίπων. Με αναλυτική αντιμετώπιση του προβλήματος προκύπτουν οι εξής εξισώσεις για τα  $\theta_0$  και  $\theta_1$ :

$$\theta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

Όπου:

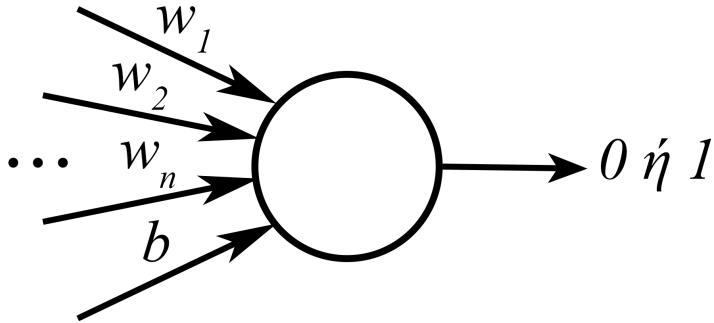
- $\bar{x}$ , ο μέσος όρος της επεξηγηματικής μεταβλητής
- $\bar{y}$ , ο μέσος όρος της μεταβλητής πρόβλεψης

### 2.3 Λογιστική παλινδρόμηση

Όπως και η γραμμική παλινδρόμηση, η λογιστική παλινδρόμηση είναι μια τεχνική που προέρχεται από τη στατιστική. Παρ' ότι το όνομά της υποδηλώνει ότι πρόκειται για τεχνική παλινδρόμησης, χρησιμοποιείται στην πραγματικότητα για διεργασίες ταξινόμησης, και συγχεκριμένα σε προβλήματα δυαδικής ταξινόμησης. Αποτελεί τεχνική επιβλεπόμενης μάθησης, που σημαίνει ότι η εκπαίδευση του μοντέλου γίνεται με ένα ζεύγος διανυσμάτων εισόδου και εξόδου.

Σκοπός της μεθόδου εκπαίδευσης της λογιστικής παλινδρόμησης είναι η εύρεση των παραμέτρων ενός λογιστικού μοντέλου, ενός δηλαδή μοντέλου που για μία δεδομένη είσοδο παράγει ως έξοδο μία πιθανότητα ο λογάριθμος της οποίας είναι γραμμικός συνδυασμός των ανεξάρτητων μεταβλητών εισόδου. Το λογιστικό μοντέλο υπολογίζει την πιθανότητα μιας δυαδικής απόκρισης βάσει ενός ή περισσοτέρων επεξηγηματικών

μεταβλητών. Αυστηρά μιλώντας, δεν κάνει ταξινόμηση από μόνος του ως έχει, αλλά μπορεί να οριστεί μια τιμή κατωφλίου για την πιθανότητα, πάνω από την οποία ο ταξινομητής παράγει θετικό αποτέλεσμα.



Σχήμα 2.5: Γραφική αναπαράσταση ενός ταξινομητή λογιστικής παλινδρόμησης.

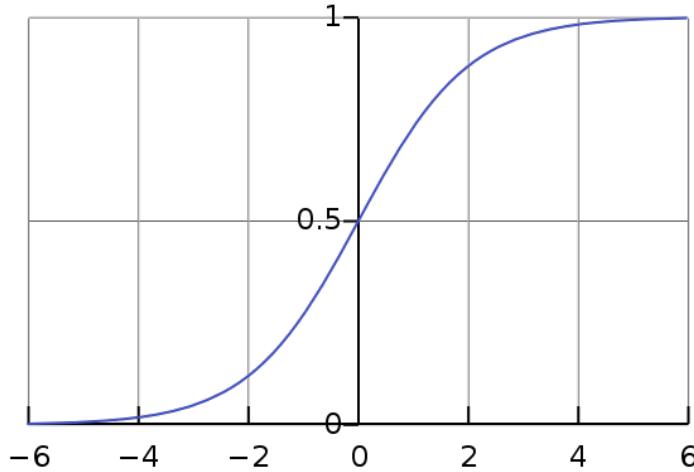
Η λογιστική παλινδρόμηση βρίσκει εφαρμογή σε περιπτώσεις όπου η έξοδος αντιπροσωπεύει κάποια κατηγορία όπως “νίκη ή ήττα”, “επιτυχία ή αποτυχία”, “υγιής ή άρρωστος”, και ούτω καθεξής. Δεδομένου ότι αντιστοιχεί συγκεκριμένη πιθανότητα σε αυτή την πρόβλεψη, μπορεί να γίνει αξιολόγηση του πόσο η παρουσία μιας συγκεκριμένης τιμής σε κάποια μεταβλητή μπορεί να αυξήσει κάποιον παράγοντα κινδύνου. Με τη μέθοδο One-vs-Rest που θα αναφερθεί παρακάτω, η λογιστική παλινδρόμηση μπορεί να χρησιμοποιηθεί και για ταξινόμηση μεταξύ περισσότερων των δύο κατηγοριών.

Σε αντίθεση με την γραμμική παλινδρόμηση, δεν υπάρχει κάποια αξιόλογη αναλυτική μέθοδος προσαρμογής του λογιστικού μοντέλου στα δεδομένα εκπαίδευσης. Αντίθετα, η εκπαίδευση γίνεται με χρήση τεχνικών βελτιστοποίησης όπως ο αλγόριθμος της επικλινούς καθόδου. Η λογιστική παλινδρόμηση αποτελεί σημαντική μέθοδο τόσο λόγω της πρακτικής χρησιμότητάς της, όσο και ως μέσο προόδου για την κατανόηση και ανάλυση των νευρωνικών δικτύων, δεδομένου ότι ένα λογιστικός παλινδρομητής αντιστοιχεί ουσιαστικά σε ένα νευρώνα.

### 2.3.1 Το λογιστικό μοντέλο

Στην ενότητα της γραμμικής παλινδρόμησης εξετάστηκε το γραμμικό μοντέλο, που χρησιμοποιείται για να κάνει προβλέψεις για κάποια μεταβλητή εξόδου που αποτελεί βαθμωτό μέγεθος. Στην περίπτωση του λογιστικού μοντέλου, η έξοδος δεν είναι βαθμωτό μέγεθος, αλλά πιθανότητα μεταξύ 0 και 1. Γίνεται λοιπόν κατανοητό ότι η συνάρτηση πρόβλεψης θα είναι μία μη-γραμμική συνάρτηση. Αυτή η συνάρτηση θα είναι η συνάρτηση που δίνει το όνομά της στη μέθοδο, δηλαδή η λογιστική, και συγκεκριμένα η ειδική της περίπτωση που ονομάζεται σιγμοειδής και ορίζεται ως εξής:

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



Σχήμα 2.6: Η σιγμοειδής συνάρτηση.

Το  $z$  ονομάζεται λογάριθμος της πιθανότητας κάποιου γεγονότος, και αποτελεί γραμμικό συνδυασμό των επεξηγηματικών μεταβλητών. Η τιμή του καθορίζεται από τις παραμέτρους που επιδιώκεται να προσδιοριστούν. Πιο συγκεκριμένα, αν διατίθεται ένα σύνολο  $n$  παραδειγμάτων  $p$  διαστάσεων,  $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$ , το  $z$  ορίζεται ως εξής:

$$z = b + w_1 x_{i1} + \dots + w_n x_{in}$$

Ή σε μορφή διανύσματος:

$$z_i = \hat{x}_i^T \hat{w} + b$$

Όπου:

- $\hat{x}_i^T = (x_{i1} \dots x_{ip})$ , το διάνυσμα των εξαρτημένων ή επεξηγηματικών μεταβλητών του παραδείγματος  $i$
- $\hat{w} = (w_1 \dots w_p)^T$ , το διάνυσμα των παραμέτρων της γραμμικής προγνωστικής συνάρτησης που περιγράφει τη σχέση μεταξύ του  $z$  και του  $x_i$
- $b$ , ο όρος απόκλισης

Σε αντίθεση με τη γραμμική παλινδρόμηση, στη βιβλιογραφία της λογιστικής παλινδρόμησης είναι πιο συνηθισμένο να γίνεται αντιμετώπιση του όρου  $b$  ξεχωριστά

από τις υπόλοιπες παραμέτρους  $w$ . Η πρόβλεψη του λογιστικού μοντέλου θα λαμβάνεται τελικά από την εξής συνάρτηση πρόβλεψης:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^z}$$

$$\hat{y} = \frac{1}{1 + e^{\hat{x}_i^T \hat{w} + b}}$$

Εδώ φαίνεται και η χρησιμότητα της σιγμοειδούς συνάρτησης, η οποία για πολύ μεγάλες θετικές τιμές θα δίνει εξόδους που τείνουν στο 0, ενώ για πολύ μεγάλες αρνητικές τιμές θα δίνει εξόδους που τείνουν στο 1. Γίνεται κατανοητό ότι σκοπός της λογιστικής παλινδρόμησης είναι ο καθορισμός των παραμέτρων  $\hat{w}$  και  $b$ , έτσι ώστε να προσδιοριστεί το λογιστικό μοντέλο.

### 2.3.2 Μέθοδος επικλινούς καθόδου

Η αξιολόγηση του πόσο καλά έχει προσαρμοστεί το λογιστικό μοντέλο πάνω στα δεδομένα εκπαίδευσης που διατίθενται γίνεται μέσω μιας συνάρτησης σφάλματος. Η επιλογή αυτής της συνάρτησης είναι κρίσιμη, καθώς η τιμή αυτής είναι που πρέπει να ελαχιστοποιηθεί μέσω του αλγορίθμου επικλινούς καθόδου. Δεδομένου ότι στόχος είναι η πρόβλεψη να βρίσκεται κοντά στη μεταβλητή απόκρισης, δηλαδή  $\hat{y} \approx y$ , μία προφανής επιλογή θα ήταν το τετραγωνισμένο σφάλμα μεταξύ της πρόβλεψης του λογιστικού μοντέλου και της μεταβλητής απόκρισης που είναι γνωστή από τα δεδομένα:

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

Ωστόσο η συγκεκριμένη συνάρτηση σφάλματος κάνει το πρόβλημα βελτιστοποίησης μη-κυρτό, καθώς έχει πολλά τοπικά ελάχιστα. Προκύπτει ότι μία καλή συνάρτηση σφάλματος είναι η εξής:

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

Μέσω της παραπάνω συνάρτησης σφάλματος που αναφέρεται σε ένα μοναδικό παράδειγμα εκπαίδευσης, θα προχύψει η λεγόμενη συνάρτηση κόστους, η οποία μετράει την απόδοση που έχουν τα επιλεγμένα  $w$  και  $b$  σε ολόκληρο το σύνολο εκπαίδευσης:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i, y_i)$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m (y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i))$$

Επιδιώκεται η μεγιστοποίηση της απόδοσης του λογιστικού μοντέλου, δηλαδή η ελαχιστοποίηση της συνάρτησης κόστους  $J$ . Αυτό ανάγεται σε πρόβλημα εύρεσης των ιδανικών  $w$  και  $b$ , που γίνεται μέσω του αλγορίθμου επικλινούς καθόδου, ενός επαναληπτικού αλγορίθμου διόρθωσης των τιμών του διανύσματος  $w$  και της τιμής του  $b$ . Σε κάθε επανάληψη του αλγορίθμου οι παράμετροι ανανεώνονται ως εξής:

$$w_i := w_i - \alpha \frac{\partial J(\hat{w}, b)}{\partial w_i}$$

$$b := b - \alpha \frac{\partial J(\hat{w}, b)}{\partial b}$$

Όπου:

- $\alpha$ , ένας προκαθορισμένος ρυθμός μάθησης

Ακολουθεί ότι για να προσδιοριστούν οι μαθηματικές πράξεις του επαναληπτικού βήματος, θα πρέπει να γίνει υπολογισμός των μερικών παραγώγων της συνάρτησης σφάλματος. Αυτές προκύπτει ότι είναι οι εξής:

$$\frac{\partial L(\hat{w}, b)}{\partial w_i} = x_i \frac{\partial \hat{y}_i}{\partial z}$$

$$\frac{\partial \hat{y}_i}{\partial z} = \hat{y}_i - y$$

### 2.3.3 Τεχνική επιλογής One-vs-Rest

Η μέθοδος της λογιστικής παλινδρόμησης στην απλή μορφή της χρησιμοποιείται για δυαδική ταξινόμηση, δηλαδή για προσδιορισμό του αν ένα παράδειγμα ανήκει σε μία

συγκεκριμένη κλάση ή όχι. Μπορεί ωστόσο με μια μικρή τροποποίηση να εφαρμοστεί και σε προβλήματα επιλογής μεταξύ πολλών κλάσεων.

Αυτό που γίνεται στην πράξη είναι να εκπαιδευτούν τόσα μοντέλα δυαδικής ταξινόμησης όσες και οι πιθανές κλάσεις στις οποίες μπορεί να ανήκει ένα παράδειγμα. Για ένα παράδειγμα προς ταξινόμηση, αφού γίνει υπολογισμός της πιθανότητας να ανήκει στην κάθε κλάση, επιλέγεται αυτή για την οποία παράγεται η μεγαλύτερη πιθανότητα.

## 2.4 Τεχνητά νευρωνικά δίκτυα

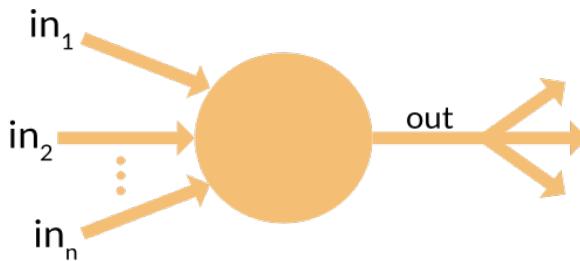
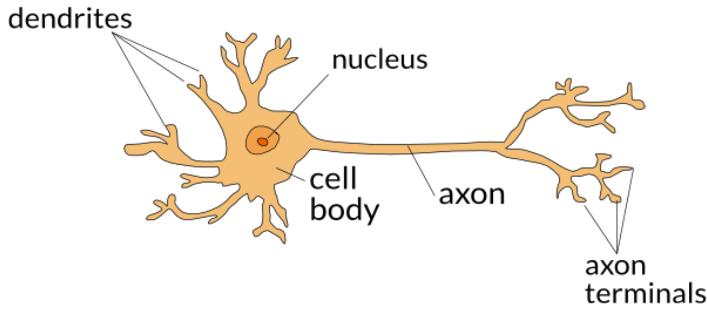
Πολλές προσεγγίσεις στη μηχανική μάθηση είναι εμπνευσμένες από παρατηρήσεις δομών και φαινομένων στον πραγματικό κόσμο. Μία από τις πλέον δημοφιλέστερες είναι και αυτή των τεχνητών νευρωνικών δικτύων, που προσεγγίζουν τον τρόπο λειτουργίας των βιολογικών νευρωνικών δικτύων που συναντώνται στον εγκέφαλο του ανθρώπου και άλλων ζώων.

Τα τεχνητά νευρωνικά δίκτυα (TNΔ για συντομία) έχουν την ικανότητα να μαθαίνουν από παραδείγματα και να βελτιώνουν συνεχώς τη συμπεριφορά τους, αντλώντας νόημα από περίπλοκα και αναχριβή δεδομένα με τρόπο που άλλες υπολογιστικές τεχνικές δεν μπορούν. Το γεγονός ότι δεν απαιτούν ρητό προγραμματισμό για τις διαφορετικές εφαρμογές στις οποίες στοχεύεται να χρησιμοποιηθούν, αλλά ούτε οποιαδήποτε γνώση για το πρόβλημα που επιλύουν εκ των προτέρων, τα κάνει ιδιαίτερα ευέλικτα και χρήσιμα σε ένα μεγάλο εύρος πεδίων.

Ένα TNΔ αποτελεί ουσιαστικά ένα υπολογιστικό σύστημα φτιαγμένο από μία συλλογή υψηλά διασυνδεδεμένων μονάδων επεξεργασίας, οι οποίες λαμβάνουν μια πληροφορία εισόδου, και μέσω αυτής παράγουν ένα συμπέρασμα. Κατ' αναλογία με την αντίστοιχη ορολογία στα βιολογικά νευρωνικά δίκτυα, αυτές οι μονάδες επεξεργασίας των TNΔ ονομάζονται νευρώνες, ενώ οι συνδέσεις μεταξύ τους συνάψεις.

Ένας νευρώνας λαμβάνει μία ή περισσότερες εισόδους και ύστερα από κάποια επεξεργασία, μεταδίδει μέσω των συνάψεων μία έξοδο. Το συμπέρασμα του δικτύου συνολικά καθορίζεται από την τιμή στην οποία καταλλήγουν ορισμένοι από τους νευρώνες. Η εκπαίδευση ενός TNΔ ανάγεται σε πρόβλημα εύρεσης του τρόπου επεξεργασίας των εισόδων κάθε νευρώνα έτσι ώστε να παράγονται κατά το δυνατόν ακριβέστερα συμπεράσματα για την κάθε είσοδο.

Το βασικό μέρος της θεωρίας πίσω από τα νευρωνικά δίκτυα έχει αναπτυχθεί από τη δεκαετία του 1980, με την ανακάλυψη της μεθόδου οπισθοδρομικής διάδοσης. Λόγω έλλειψης της απαιτούμενης υπολογιστικής ισχύος, η πρακτική τους χρησιμότητα βρισκόταν ωστόσο υπό αμφισβήτηση για πολλά χρόνια. Οι εξελίξεις των τελευταίων



Σχήμα 2.7: Σύγκριση ενός φυσικού νευρώνα με έναν τεχνητό.

ετών ωστόσο, τόσο στη θεωρητική έρευνα όσο και στην ανάπτυξη υλικού υψηλής υπολογιστικής ισχύος, έχουν αλλάξει πλήρως αυτή την πεποίθηση. Ιδιαίτερα την τελευταία δεκαετία, ένας αυξανόμενος αριθμός εφαρμογών, εμπορικών και μη, κάνουν χρήση τεχνητών νευρωνικών δικτύων σε μια πλειάδα επιστημονικών πεδίων. Στα πλαίσια δε της «βαθιάς» μάθησης, τα νευρωνικά δίκτυα αποτελούν αδιαμφισβήτητα μία από τις ισχυρότερες και περισσότερο υποσχόμενες προσεγγίσεις στη μηχανική μάθηση.

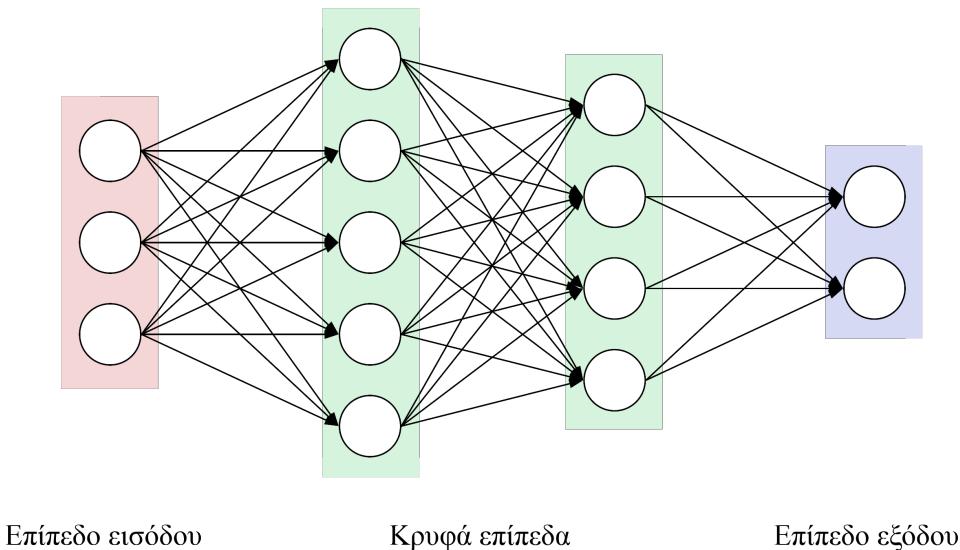
#### 2.4.1 Δομή του TNΔ

Η τοπολογία ενός TNΔ, δηλαδή ο τρόπος οργάνωσης των νευρώνων μέσα σε αυτό και ο τρόπος διασύνδεσης των νευρώνων μεταξύ τους, αποτελεί την ειδοποιό διαφορά μεταξύ των διαφορετικών τύπων. Διαφορετικές τοπολογίες μπορούν να αποβούν χρήσιμες σε διαφορετικά προβλήματα.

Ο πιο συνηθισμένος τρόπος οργάνωσης των νευρώνων είναι σε επίπεδα, με τους νευρώνες κάθε επιπέδου να συνδέονται μόνο με νευρώνες του προηγούμενου και του επόμενου επιπέδου. Γίνεται λόγος για το επίπεδο εισόδου, στους νευρώνες του οποίου τροφοδοτείται η είσοδος, για τα κρυφά επίπεδα, όπου γίνεται η βασική επεξεργασία των δεδομένων εισόδου, και για το επίπεδο εξόδου, στο οποίο εμφανίζονται τα συμπεράσματα. Όταν υπάρχουν περισσότερα από ένα επίπεδα κρυφών νευρώνων, το TNΔ ονομάζεται βαθύ νευρωνικό δίκτυο.

Η έξοδος ενός νευρώνα γίνεται είσοδος για κάποιον άλλο με τον οποίο είναι

συνδεδεμένος. Η σχέση αυτή εκφράζεται μέσω των συνάψεων, οι οποίες περιέχουν πληροφορία σχετική με το πόσο μεγάλη επιρροή έχει η κάθε είσοδος του νευρώνα στην έξοδό του, τον επονομαζόμενο συντελεστή βαρύτητας. Ένα TNΔ του οποίου κάθε νευρώνας είναι συνδεδεμένος με όλους τους νευρώνες του προηγούμενου και του επόμενου επιπέδου, ονομάζεται πλήρως διασυνδεδεμένο, ενώ σε διαφορετική περίπτωση πρόκειται για ένα συνελικτικό δίκτυο.



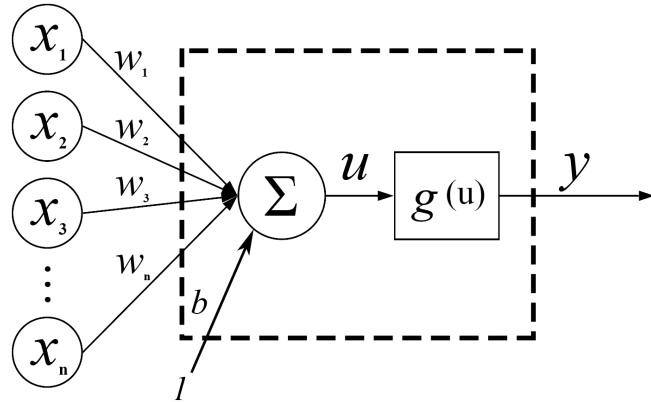
Σχήμα 2.8: Δομή ενός νευρωνικού δικτύου εμπρόσθιας τροφοδότησης τριών επιπέδων.

Γίνεται επιπλέον διάκριση μεταξύ των TNΔ βάσει της κατεύθυνσης διάδοσης της πληροφορίας. Στα επονομαζόμενα TNΔ εμπρόσθιας τροφοδότησης, η διάδοση γίνεται αποκλειστικά από το προηγούμενο επίπεδο προς το επόμενο, ενώ στα ανατροφοδοτούμενα TNΔ, οι συνάψεις μπορούν να είναι αμφικατευθυντικές.

Από μαθηματικής πλευράς, η έξοδος κάθε νευρώνα είναι ένας πραγματικός αριθμός, ενώ το σύνολο των εισόδων του είναι ένα σύνολο πραγματικών αριθμών που ονομάζεται διάνυσμα εισόδου. Ανεξαρτήτως τύπου και τοπολογίας του δικτύου, το έργο ενός τεχνητού νευρώνα παραμένει το ίδιο, δηλαδή ο μετασχηματισμός ενός διανύσματος εισόδου σε μία και μόνο έξοδο. Το διάνυσμα εισόδου προέρχεται από εξόδους προηγούμενων νευρώνων, ενώ η έξοδος που παράγεται είναι τυπικά ένας μη-γραμμικός μετασχηματισμός του που μεταδίδεται εν συνεχείᾳ στους επόμενους νευρώνες.

Πιο συγκεκριμένα, ο νευρώνας υπολογίζει ένα άθροισμα που αποτελεί γραμμικό μετασχηματισμό των τιμών της εισόδου, κάνοντας χρήση των συντελεστών βαρύτητας των συνάψεων, και παράγει μία έξοδο που προκύπτει ως μη-γραμμική συνάρτηση αυτού του αθροίσματος και είναι συνήθως μεταξύ 0 και 1. Αυτή η μη-γραμμική συνάρτηση ονομάζεται συνάρτηση μεταφοράς ή ενεργοποίησης και μπορεί να είναι η βηματική, η

σιγμοειδής, ή παρεμφερείς συναρτήσεις. Όπως φανεί παρακάτω, η επιλογή της είναι κρίσιμη για την ταχύτητα εκπαίδευσης του δικτύου και την απόδοσή του.



Σχήμα 2.9: Γραφική αναπαράσταση ενός τεχνητού νευρώνα.

Ένας νευρώνας κάνει ουσιαστικά το ίδιο έργο με το λογιστικό μοντέλο. Θεωρώντας  $y$  την έξοδο, η συμπεριφορά ενός νευρώνα μπορεί να περιγραφεί ως εξής:

$$y = g(u)$$

Με:

$$u = b + w_1x_1 + \dots + w_nx_n$$

Ή σε μορφή διανύσματος:

$$u = \hat{x}_i^T \hat{w} + b$$

Όπου:

- $\hat{x}^T = (x_1 \dots x_p)$ , ένα διάνυσμα εισόδου  $p$  διαστάσεων
- $\hat{w} = (w_1 \dots w_p)^T$ , το διάνυσμα των παραμέτρων της γραμμικής προγνωστικής συνάρτησης που περιγράφει τη σχέση μεταξύ του  $u$  και του  $\hat{x}$
- $b$ , ο όρος απόκλισης

Στα πλαίσια της εργασίας γίνεται λόγος για πλήρως διασυνδεδεμένα TNΔ εμπρόσθιας τροφοδότησης ενός ή παραπάνω χρυφών επιπέδων. Αυτού του είδους το TNΔ ονομάζονται στη βιβλιογραφία και ως multilayer perceptron, και γίνεται η σύμ-

βαση ότι ένα multilayer perceptron  $n$  επιπέδων περιέχει  $n - 1$  κρυφά επίπεδα συν το επίπεδο εξόδου.

### 2.4.2 Η εμπρόσθια τροφοδότηση

Η εμπρόσθια τροφοδότηση είναι ο τρόπος με τον οποίο το νευρωνικό δίκτυο μετασχηματίζει ένα διάνυσμα εισόδου σε ένα διάνυσμα εξόδου. Οι νευρώνες εισόδου τροφοδοτούνται από το λεγόμενο διάνυσμα εισόδου. Αυτό σημαίνει ότι η τιμή κάθε νευρώνα τίθεται ίση με την τιμή της αντίστοιχης μεταβλητής του διανύσματος εισόδου.

Επακολούθως γίνεται μετάδοση της πληροφορίας εισόδου στο πρώτο κρυφό επίπεδο, δηλαδή γίνεται αρχικά ένας γραμμικός μετασχηματισμός των εξόδων των νευρώνων εισόδου, και στη συνέχεια ένας μη-γραμμικός μετασχηματισμός της τιμής που προέκυψε από το προηγούμενο μέσω της συνάρτησης ενεργοποίησης. Με τον ίδιο ακριβώς τρόπο γίνεται εμπρόσθια μετάδοση και προς τα επόμενα κρυφά επίπεδα.

Αφού παραχθούν όλοι οι έξοδοι των νευρώνων του τελευταίου κρυφού επιπέδου, γίνεται το ίδιο ακριβώς πράγμα με μετάδοση προς τους νευρώνες εξόδου. Οι τιμές των νευρώνων εξόδου είναι το λεγόμενο διάνυσμα εξόδου, από το οποίο προκύπτει το συμπέρασμα στο οποίο καταλήγει το TNΔ, συνήθως με επιλογή της χλάσης που αντιστοιχεί στον νευρώνα που αποκτά τη μεγαλύτερη τιμή.

### 2.4.3 Η οπισθοδρομική διάδοση του σφάλματος

Όπως και στη λογιστική παλινδρόμηση, τρόπος εκπαίδευσης των νευρωνικών δικτύων είναι τυπικά ο αλγόριθμος της επικλινούς καθύδου. Γίνεται αρχικά η εμπρόσθια μετάδοση, και αφού υπολογιστεί το σφάλμα των νευρώνων εξόδου, γίνεται η λεγόμενη οπισθοδρομική διάδοσή του στα κρυφά επίπεδα, και τέλος στο επίπεδο εισόδου. Το σφάλμα για συγκεκριμένες τιμές των παραμέτρων  $w$  και  $b$  υπολογίζεται ακριβώς όπως στη λογιστική παλινδρόμηση:

$$J(w^{[1]}, b^{[1]}, \dots, w^{[m]}, b^{[m]}) = \frac{1}{n} \sum_{i=0}^n L(\hat{y}, y)$$

Όπου:

- $L(\hat{y}, y)$ , η συνάρτηση σφάλματος.
- $\hat{y}$ , το αναμενόμενο διάνυσμα εξόδου.
- $y$ , το διάνυσμα εξόδου που προβλέπει το νευρωνικό δίκτυο.

- $w^{[i]}$ , το διάνυσμα των παραμέτρων της γραμμικής προγνωστικής συνάρτησης για το επίπεδο  $i$  από τα  $m$  συνολικά επίπεδα.

Το σφάλμα αυτό μεταδίδεται οπισθοδρομικά προς τους νευρώνες των προηγούμενων επιπέδων με τους οποίους συνδέεται, σύμφωνα με τη μερική παράγωγό τους ως προς τα βάρη των συνάψεων με αυτούς τους νευρώνες:

$$\frac{\partial J(w^{[1]}, b^{[1]}, \dots, w^{[m]}, b^{[m]})}{\partial w_i}$$

$$\frac{\partial J(w^{[1]}, b^{[1]}, \dots, w^{[m]}, b^{[m]})}{\partial b_i}$$

Τέλος, γίνεται ενημέρωση των παραμέτρων όπως και στη λογιστική παλινδρόμηση:

$$w_i := w_i - \alpha \frac{\partial J(\hat{w}, b)}{\partial w_i}$$

$$b_i := b_i - \alpha \frac{\partial J(\hat{w}, b)}{\partial b}$$

#### 2.4.4 Επιλογή της συνάρτησης ενεργοποίησης

Η επιλογή της συνάρτησης ενεργοποίησης είναι χρίσμα για την επιτυχία και την ταχύτητα εκπαίδευσης του TNΔ. Διαφορετικές συναρτήσεις ενεργοποίησης λειτουργούν καλύτερα σε διαφορετικά προβλήματα και διαφορετικά δεδομένα εισόδου, και η ανάγκη δοκιμής περισσότερων από μία δε σπανίζει. Γενικά η συνάρτηση ενεργοποίησης θα συμβολίζεται ως  $g(z)$ .

Μία συχνή επιλογή που συναντάται στη βιβλιογραφία λόγω της ιστορίας της από τη λογιστική παλινδρόμηση είναι η σιγμοειδής συνάρτηση:

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

Αν και η σιγμοειδής είναι μια καλή επιλογή που μπορεί να παράξει αξιόλογα αποτελέσματα, προκύπτει ότι υπάρχουν καλύτερες επιλογές. Η  $\tanh$  είναι μία απ' αυτές, και ουσιαστικά αποτελεί μια μεταποιημένη έκδοση της σιγμοειδούς. Ορίζεται ως εξής:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Σχεδόν σε όλες τις περιπτώσεις η χρήση της  $\tanh$  είναι προτιμότερη σε σχέση με τη σιγμοειδή. Αξίζει να σημειωθεί ότι διαφορετικά επίπεδα του νευρωνικού δικτύου μπορούν να κάνουν χρήση διαφορετικών συναρτήσεων ενεργοποίησης, και πολλές φορές, συγκεκριμένα στην περίπτωση της δυαδικής ταξινόμησης, η σιγμοειδής χρησιμεύει για το επίπεδο εξόδου.

Η καλύτερη επιλογή για τη μετάδοση από την είσοδο στους χρυφούς νευρώνες και μεταξύ των χρυφών νευρώνων είναι συνήθως η λεγόμενη συνάρτηση Rectified Logic Unit, ή ReLU function, η οποία ορίζεται ως εξής:

$$\text{ReLU}(z) = z^+ = \max(0, z)$$

Η συγκεκριμένη συνάρτηση έχει παράγωγο 0 για τιμές κάτω από 0 και 1 για τιμές ίσες ή άνω του 0.

## 2.5 Απλός ταξινομητής Bayes

Μία απλή και εύχρηστη τεχνική ταξινόμησης είναι ο απλός ταξινομητής Bayes. Όπως δηλώνει και το όνομά του, είναι βασισμένος στο θεώρημα του Bayes που προέρχεται από την πιθανοθεωρία, και χρησιμοποιείται για να προβλέψει την πιθανότητα ενός συμβάντος βάσει προηγούμενης γνώσης για συνθήκες που ενδέχεται να σχετίζονται με αυτό. Βασικό χαρακτηριστικό του είναι η υπόθεση ανεξαρτησίας που κάνει μεταξύ των χαρακτηριστικών κάθε χλάσης, και συνεπώς μεταξύ των προγνωστικών μεταβλητών. Με άλλα λόγια, υποθέτει ότι η παρουσία ενός χαρακτηριστικού είναι ασυσχέτιστη με την παρουσία οποιουδήποτε άλλου χαρακτηριστικού. Για παράδειγμα, ένα φρούτο μπορεί να θεωρηθεί μήλο αν είναι κόκκινο και στρογγυλό, αλλά δεν θα ληφθεί υπόψιν κάποια πιθανή συσχέτιση μεταξύ χρώματος και σχήματος.

Το μεγάλο πλεονέκτημα του απλού ταξινομητή Bayes, εκτός από την ευκολία που υπάρχει στην κατανόησή του, είναι η ευκολία και η ταχύτητα κατασκευής του. Αυτή γίνεται υπό επίβλεψη και σε αντίθεση με τις περισσότερες μενόδους μηχανικής μάθησης, δεν περιέχει κάποιο επαναληπτικό βήμα. Γίνεται συνεπώς πολύ γρήγορα και ενδείκνυται για μεγάλα σύνολα δεδομένων, χρησιμεύοντας συχνά στην αξιολόγηση αρχικών υποθέσεων. Προκύπτει δε ότι σε περίπτωση που η υπόθεση ανεξαρτησίας

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Σχήμα 2.10: Ο απλός ταξινομητής Bayes μπορεί εύκολα να εφαρμοστεί σε ένα σύνολο δεδομένων διαχριτών μεταβλητών εισόδου, στην οποία περίπτωση υπολογίζονται οι πίνακες συχνοτήτων για κάθε χαρακτηριστικό.

μεταξύ των προγνωστικών μεταβλητών ισχύει σε σημαντικό βαθμό, οι προβλέψεις ενός απλού ταξινομητή Bayes μπορούν να ανταγωνιστούν ή και να ξεπεράσουν αντίστοιχες προβλέψεις ακόμα και πιο εκλεπτυσμένων μοντέλων.

Ο απλός ταξινομητής Bayes βρίσκει τις απαρχές του στην κοινότητα ανάκτησης κειμένου εδώ και κάμποσες δεκαετίες, και παραμένει ιδιαίτερα χρήσιμος σε τέτοιου είδους εφαρμογές. Χρησιμοποιείται συχνά στην ταξινόμηση κειμένου ως έγκυρο ή ανεπιθύμητο, στην κατηγοριοποίηση περιεχομένου κειμένου μεταξύ κατηγοριών όπως αθλητικά, πολιτικά, χλπ., χρησιμοποιώντας τις συχνότητες λέξεων ως χαρακτηριστικά, καθώς και σε άλλες εφαρμογές. Τα τελευταία χρόνια πρωταγωνιστεί ως μέθοδος στην ανάπτυξη μοντέλων αυτόματης ιατρικής διάγνωσης.

### 2.5.1 Πιθανοτικό μοντέλο

Για την ταξινόμηση ενός παραδείγματος με  $n$  χαρακτηριστικά σε μία από  $k$  κλάσεις, ο απλός ταξινομητής Bayes κάνει κατά βάση δύο πράγματα. Βρίσκει αρχικά την πιθανότητα αυτό το παράδειγμα να ανήκει σε έκαστη από τις  $k$  κλάσεις, και ύστερα παράγει ως έξοδο την κλάση με τη μέγιστη πιθανότητα.

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k | \bar{x})$$

Όπου:

- $\bar{x}$ , το διάνυσμα των προγνωστικών μεταβλητών της εισόδου  $\bar{x} = (x_1, \dots, x_n)$

- $C_k$ , μία εκ των  $k$  κλάσεων στις οποίες μπορεί να ανήκει το παράδειγμα

Κάνοντας την υπόθεση ανεξαρτησίας μεταξύ των προγνωστικών μεταβλητών, η παραπάνω πιθανότητα υπολογίζεται βάσει του θεωρήματος του Bayes ως εξής:

$$p(C_k|\bar{x}) = \frac{p(C_k)p(\bar{x}|C_k)}{p(\bar{x})}$$

Όπου:

- $p(C_k|\bar{x})$ , η εκ των υστέρων πιθανότητα μιας κλάσης δεδομένου διανύσματος προγνωστικών μεταβλητών, η λεγόμενη posterior πιθανότητα
- $p(C_k)$ , η εκ των προτέρων πιθανότητα μιας κλάσης, η λεγόμενη prior πιθανότητα
- $p(\bar{x}|C_k)$ , η πιθανότητα ενός προβλεπτή δεδομένης κλάσης, το λεγόμενο likelihood
- $p(\bar{x})$ , η εκ των προτέρων πιθανότητα ενός προβλεπτή

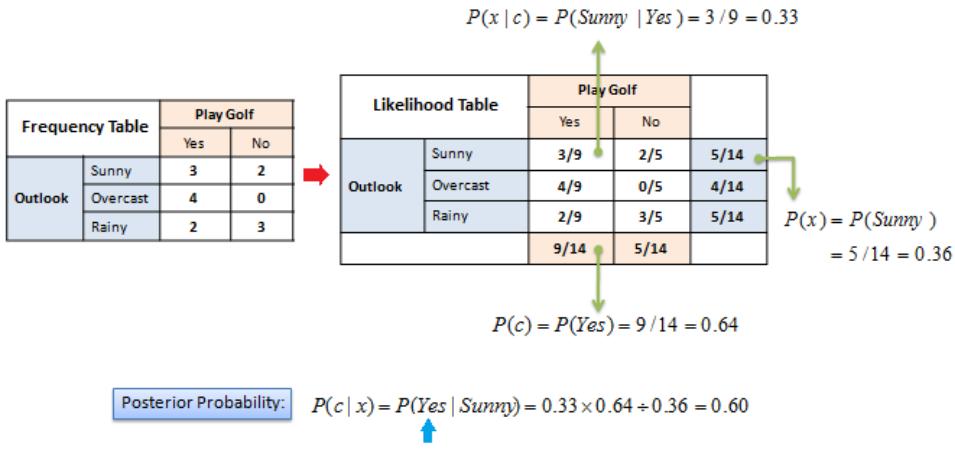
Ο απλός ταξινομητής Bayes μπορεί να χρησιμοποιηθεί τόσο σε φαινόμενα των οποίων οι προγνωστικές μεταβλητές είναι διαχριτές, όσο και σε φαινόμενα με συνεχείς τιμές στους προβλεπτές. Και στις δύο περιπτώσεις, η πρόβλεψη της εκ των υστέρων πιθανότητας μιας κλάσης δεδομένου διανύσματος προγνωστικών μεταβλητών γίνεται υπολογίζοντας το εξής γινόμενο:

$$p(C_k|\bar{x}) = p(x_1|C_k)p(x_2|C_k)...p(x_n|C_k)p(C_k)$$

Οι δύο περιπτώσεις διαφέρουν ωστόσο σε ορισμένες λεπτομέρειες σχετικές με την εκπαίδευσή τους, καθώς και στον τρόπο με τον οποίο παράγουν την πρόβλεψη.

### 2.5.2 Multinomial ταξινομητής

Η εκπαίδευση του πιθανοτικού μοντέλου στην περίπτωση διαχριτών τιμών στις προβλεπτικές μεταβλητές συνεπάγεται στον υπολογισμό των πινάκων των πιθανοτήτων που είναι απαραίτητες για την πρόβλεψη. Αρχικά γίνεται δόμηση του πίνακα συχνοτήτων κάθε κλάσης, ο οποίος μπορεί να χρησιμοποιηθεί για τον υπολογισμό της πιθανότητας  $p(C_k)$ . Στη συνέχεια, υπολογίζεται ο πίνακας συχνοτήτων για κάθε χαρακτηριστικό κάθε μεταβλητής έναντι κάθε κλάσης, και ο πίνακας των εκ των προτέρων συχνοτήτων κάθε χαρακτηριστικού κάθε προγνωστικής μεταβλητής. Οι δύο αυτοί πίνακες μπορούν να χρησιμοποιηθούν στη συνέχεια για τον υπολογισμό των πιθανοτήτων  $p(\bar{x})$  και  $p(\bar{x}|C_k)$ .



Σχήμα 2.11: Η κατασκευή του πίνακα συχνοτήτων κάθε προβλεπτή δεδομένης κλάσης συνεπάγονται εύκολα την κατασκευή του αντίστοιχου πίνακα πιθανοτήτων.

Οι εκ των υστέρων πιθανότητες που παράγονται από τις προβλέψεις του μοντέλου μπορούν να κανονικοποιηθούν έτσι ώστε η τιμή τους να βρίσκεται μεταξύ 0 και 1. Κάτι τέτοιο δεν χρίνεται όμως απαραίτητο δεδομένου του κανόνα επιλογής που τέθηκε, ο οποίος συνεπάγεται ότι η έκβαση της πρόβλεψης καθορίζεται απλώς από την μέγιστη πιθανότητα. Έχοντας τις σχετικές πιθανότητες, μπορεί να αποδοθεί μια εκτίμηση για την αξιοπιστία της πρόβλεψης, ωστόσο αυτή δεν πρέπει να ληφθεί τοις μετρητοίς, καθώς δεν είναι απολύτως φερέγγυα.

Αξίζει να σημειωθεί η περίπτωση του λεγόμενου προβλήματος μηδενικής συχνότητας. Αν υπάρχει κάποια τιμή ενός χαρακτηριστικού που δεν οδηγεί ποτέ σε κάποια κλάση, μία απλή λύση προς αποφυγή μαθηματικών ασυνεπειών είναι η προσθήκη του 1 σε κάθε συνδυασμό τιμής-κλάσης του πίνακα συχνοτήτων.

### 2.5.3 Gaussian ταξινομητής

Όταν γίνεται χειρισμός δεδομένων με συνεχείς αριθμητικές τιμές στις προγνωστικές μεταβλητές, μια τυπική υπόθεση είναι ότι οι συνεχείς αυτές τιμές που συσχετίζονται με κάθε κλάση είναι κατανεμημένες σύμφωνα με την Γκαουσιανή κατανομή. Πρέπει αρχικά να υπολογιστούν ο μέσος  $\mu$  και η τυπική απόκλιση  $\sigma$ . Αφού διαχωριστούν τα δεδομένα βάσει της κλάσης τους, ο υπολογισμός για κάθε μεταβλητή της κλάσης γίνεται ως εξής:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2}$$

Σύμφωνα την παραπάνω εικασία, δεδομένης κλάσης  $C_k$  ο υπολογισμός της πιθανότητας κατανομής μιας τιμής  $u$  κάποιας προγνωστικής μεταβλητής  $x_i$  γίνεται θέτοντας στην εξίσωση της κανονικής κατανομής την συγκεκριμένη τιμή και τις αντίστοιχες τιμές  $\sigma_k$  και  $\mu_k$  που αντιστοιχούν στην κλάση.

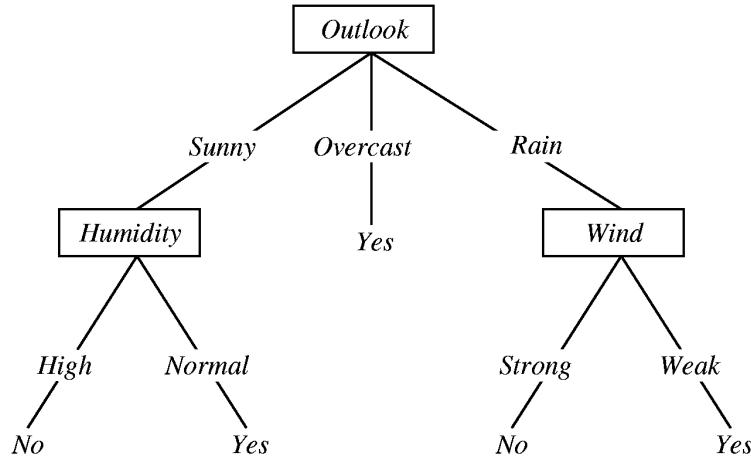
$$p(x_i = u | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(u-\mu_k)^2}{2\sigma_k^2}}$$

## 2.6 Εκμάθηση δέντρων αποφάσεων

Από τις πλέον δημοφιλέστερες δομές που χρησιμοποιούνται για τη λήψη αποφάσεων είναι τα δέντρα αποφάσεων. Τα δέντρα αποφάσεων αποτελούν δομές δέντρου που λειτουργούν με παρόμοιο τρόπο με τα διαγράμματα ροής. Υποθέτοντας δεδομένα εισόδου ενός αριθμού επεξηγηματικών μεταβλητών, κάθε εσωτερικός κόμβος αντιστοιχεί σε μία από αυτές τις επεξηγηματικές μεταβλητές, και συνδέεται κάτωθι με τόσους κόμβους όσες και οι πιθανές τιμές της μεταβλητής ή οι πιθανές απαντήσεις σε ένα ερώτημα που αφορά την τιμή της μεταβλητής. Υπ' αυτή την έννοια κάθε εσωτερικός κόμβος είναι μία «δοκιμή» πάνω σε κάποιο χαρακτηριστικό των δεδομένων και σκοπός είναι η άφιξη σε ένα φύλλο του δέντρου που αντιπροσωπεύει μία τιμή της μεταβλητής απόκρισης, και συνεπώς ένα συμπέρασμα.

Η εκμάθηση δέντρων αποφάσεων αφορά τις τεχνικές ανάπτυξης ή «βλάστησης» του δέντρου που μπορεί στη συνέχεια να χρησιμοποιηθεί με σκοπό την ταξινόμηση ή την παλινδρόμηση. Αυτό γίνεται προσαρμόζοντας το δέντρο πάνω σε ένα σύνολο δεδομένων με ορισμένες τιμές στις μεταβλητές εισόδου και τα αντίστοιχα συμπεράσματα στα οποία αυτές οδηγούν. Έχει δημιουργηθεί ένας αριθμός αλγορίθμων που εκπληρώνουν αυτό το σκοπό.

Το βασικό πλεονέκτημα των δέντρων αποφάσεων είναι η ευκολία κατανόησή τους. Η δομή τους είναι εύκολα αναγνώσιμη, και κάθε βήμα που ακολουθείται για τη λήψη μιας απόφασης είναι ορατό και εύκολα ερμηνεύσιμο από τον χρήστη, αποτελούν δηλαδή ένα μοντέλο «λευκού κουτιού». Γι' αυτό το λόγο, η προσέγγιση χρησιμοποιείται σε πολλά επιστημονικά πεδία, και ιδιαίτερα στα πλαίσια της εξόρυξης δεδομένων.



Σχήμα 2.12: Ακολουθώντας τη διαδρομή που καθορίζει ένα διάνυσμα εισόδου, το δέντρο αποφάσεων μπορεί εύκολα να οδηγήσει στην πιθανότερη έκβαση για τη μεταβλητή εξόδου.

### 2.6.1 Χαρακτηριστικά του δέντρου

Ένα δέντρο αποφάσεων είναι μια απλή αναπαράσταση του τρόπου με τον οποίο ένα διάνυσμα εισόδου με συγκεκριμένες τιμές σε κάθε επεξηγηματική μεταβλητή μπορεί να οδηγήσει σε μία έξοδο. Γενικά τα δέντρα αποφάσεων, που αναφέρονται ομαδικά στη διεθνή βιβλιογραφία ως classification and regression trees (CART) χωρίζονται σε δύο βασικές κατηγορίες:

- Τα δέντρα ταξινόμησης, όπου η πρόβλεψη στην οποία οδηγεί το δέντρο είναι η επιλογή μιας κλάσης από έναν αριθμό πιθανών κλάσεων.
- Τα δέντρα παλινδρόμησης, όπου η πρόβλεψη στην οποία οδηγεί το δέντρο είναι ένας πραγματικός αριθμός.

Στα πλαίσια της εργασίας θα γίνει αναφορά σε δέντρα ταξινόμησης και σε εκπαίδευση με σύνολα δεδομένων που έχουν μία μόνο μεταβλητή απόκρισης η οποία θα αναφέρεται ως «έκβαση», και κάθε τιμή της οποίας θα αντιπροσωπεύει μία πιθανή κλάση. Επιπλέον θα γίνει η υπόθεση ότι τα χαρακτηριστικά των μεταβλητών εισόδου μπορούν να λάβουν μόνο έναν πεπερασμένο αριθμό διαχριτών τιμών.

Ένα δέντρο αποφάσεων αποτελείται κατά βάση από τη ρίζα του και από τους κόμβους στους οποίους αυτή διακλαδώνεται. Κάθε κόμβος είναι συνυφασμένος με ένα χαρακτηριστικό εισόδου και διακλαδώνεται σε τόσους άλλους κόμβους όσες και οι πιθανές τιμές που μπορεί να πάρει το χαρακτηριστικό αυτό. Τα τόξα δε που συνδέουν έναν κόμβο με τα παιδιά του συνδέονται με κάθε μία από τις πιθανές τιμές που μπορεί να πάρει το χαρακτηριστικό.

### 2.6.2 Εκπαίδευση ενός δέντρου αποφάσεων

Η εκπαίδευση ενός δέντρου αποφάσεων στηρίζεται στη διαδοχική διχοτόμηση του πηγάιου συνόλου δεδομένων ενός κόμβου σε υποσύνολα βάση ενός χαρακτηριστικού εισόδου. Με τη δημιουργία του, ένας κόμβος αντιπροσωπεύει ένα υποσύνολο του συνόλου δεδομένων και ο ίδιος διαιρείται πάλι σε νέους κόμβους χρησιμοποιώντας ένα άλλο χαρακτηριστικό. Πιο αναλυτικά, η εκπαίδευση βήμα προς βήμα έχει ως εξής:

- Δημιουργία ενός δέντρου με έναν κόμβο ο οποίος ορίζεται ως ρίζα.
- Αν όλα τα παραδείγματα ανήκουν σε μία κλάση, τότε το δέντρο αποτελείται μόνο από τη ρίζα, και η έκβαση είναι πάντα η συγκεκριμένη κλάση.
- Αν δεν υπάρχουν άλλες επεξηγηματικές μεταβλητές, τότε το δέντρο αποτελείται μόνο από τη ρίζα, και η έκβαση είναι η πιο συχνή κλάση μεταξύ των υπαρχόντων.
- Διαφορετικά:
  - Γίνεται επιλογή του «καλύτερου» χαρακτηριστικού βάσει του οποίου μπορούν να διχοτομηθούν τα δεδομένα.
  - Ορισμός του συγκεκριμένου χαρακτηριστικού ως χαρακτηριστικό διχοτόμησης στον παρόντα κόμβο, και για κάθε πιθανή τιμή του χαρακτηριστικού:
    - \* Δημιουργία ενός νέου κόμβου που τίθεται ως ο κλάδος του παραπάνω κόμβου και αντιστοιχεί σε μία πιθανή τιμή.
    - \* Ορισμός του υποσυνόλου που λαμβάνει την παραπάνω τιμή στο συγκεκριμένο χαρακτηριστικό ως σύνολο του νέου κόμβου.
    - \* Αν το υποσύνολο αυτό είναι άδειο, τότε δεν διχοτομείται περαιτέρω και έκβαση θεωρείται η πιο συχνή κλάση στα παραδείγματα.
    - \* Διαφορετικά ο αλγόριθμος ξεκινάει εκ νέου με διχοτόμηση του νέου κόμβου που δημιουργήθηκε.

Η διαδικασία επαναλαμβάνεται έως ότου να μην μπορεί να γίνει περαιτέρω διαίρεση κανενός κόμβου, είτε λόγω έλλειψης άλλων χαρακτηριστικών εισόδου είτε επειδή οι μεταβλητές απόκρισης των παραδειγμάτων του υποσυνόλου έχουν όλες την ίδια τιμή, και συνεπώς μπορεί να γίνει απολύτως βέβαιη πρόβλεψη. Ο αλγόριθμος αυτός είναι ένα παράδειγμα «άπληστου» αλγορίθμου και αποτελεί τον πιο κοινό τρόπο εκμάθησης ενός δέντρου αποφάσεων από δεδομένα.

### 2.6.3 Επιλογή του «καλύτερου» χαρακτηριστικού

Η επιλογή του χαρακτηριστικού βάσει του οποίου θα γίνει η διχοτόμηση είναι βασικό σημείο του αλγορίθμου, και ο τρόπος προσδιορισμού του αποτελεί την ειδοποιό

διαφορά μεταξύ των διαφορετικών εκδοχών του αλγορίθμου ανάπτυξης ενός δέντρου αποφάσεων. Συνήθως αυτή γίνεται υπολογίζοντας ένα συγκεκριμένο μέτρο για κάθε υποψήφιο χαρακτηριστικό διχοτόμησης και επιλέγοντας το χαρακτηριστικό εκείνο που μεγιστοποιεί ή ελαχιστοποιεί αυτό το μέτρο.

Δύο μέτρα που χρησιμοποιούνται συχνά είναι το λεγόμενο gini impurity στην περίπτωση του αλγορίθμου CaRT και το κέρδος πληροφορίας στον ID3. Στην συγκεκριμένη εργασία θα γίνει λόγος για τον ID3, που αποτελεί έναν από τους συνηθέστερους αλγορίθμους επιλογής, και του οποίου οι μεταγενέστερες εκδόσεις όπως ο C4.5 και C5.0 βρίσκονται μεταξύ των ισχυρότερων αλγορίθμων ταξινόμησης στη βιομηχανία.

Το κέρδος πληροφορίας είναι ένα μέτρο που βασίζεται στην ιδέα της εντροπίας που προέρχεται από τη θεωρία πληροφορίας. Η εντροπία μετρά ουσιαστικά το μέγεθος της αβεβαιότητας σε ένα σύνολο  $S$ :

$$H(S) = \sum_{x \in X} -p(x)\log_2 p(x)$$

Όπου:

- $S$ , το σύνολο για το οποίο υπολογίζεται η εντροπία, δηλαδή το υποσύνολο που αντιστοιχεί σε κάθε κόμβο
- $X$ , το σύνολο των κλάσεων στο  $S$
- $p(x)$ , το ποσοστό των παραδειγμάτων του  $S$  που ανήκουν στην κλάση  $x$

Η ισότητα  $H(S) = 0$  αντιστοιχεί στην περίπτωση όπου όλα τα παραδείγματα του συνόλου  $S$  ανήκουν στην ίδια κλάση. Αντίστοιχα, όταν  $H(S) = 1$ , τα παραδείγματα του υποσυνόλου είναι εξίσου πιθανό να ανήκουν σε μία οποιαδήποτε κλάση από το σύνολο. Είναι προφανές ότι όσο χαμηλότερη είναι η εντροπία που αντιστοιχεί σε ένα υποσύνολο, τόσο μεγαλύτερη βεβαιότητα υπάρχει για την έκβαση στην οποία οδηγεί η συγκεκριμένη διακλάδωση του δέντρου.

Το δε κέρδος πληροφορίας μετράει το πόσο μειώνεται η εντροπία ύστερα από τη διχοτόμηση του συνόλου  $S$  βάσει ενός συγκεκριμένου χαρακτηριστικού  $A$ :

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

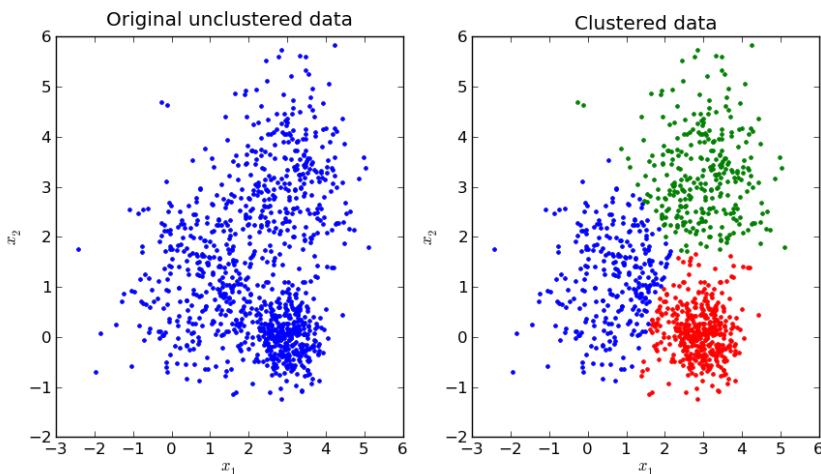
Όπου:

- $H(S)$ , η εντροπία του κόμβου γονέα που αντιπροσωπεύει το σύνολο  $S$

- $T$ , τα υποσύνολα που δημιουργούνται για κάθε παιδί κόμβο στο οποίο διαχλαδώνεται ο παραπάνω κόμβος
- $p(t)$ , το ποσοστό των παραδειγμάτων του συνόλου  $S$  που βρίσκονται στο υποσύνολο  $t$
- $H(t)$ , η εντροπία του κόμβου γονέα που αντιπροσωπεύει το υποσύνολο  $t$

## 2.7 Ομαδοποίηση $k$ -μέσων

Τα περισσότερα δεδομένα στον κόσμο δεν έχουν «ετικέτα». Αυτό σημαίνει ότι μπορεί να υπάρχει διαθέσιμη μια συλλογή παραδειγμάτων ενός φαινομένου με μετρήσιες για ορισμένα από τα χαρακτηριστικά του, χωρίς όμως να καταγράφεται η κατηγορία στην οποία ανήκει το κάθε παράδειγμα ή τα συμπεράσματα στα οποία οδηγεί. Για να γίνουν αυτά τα δεδομένα χρήσιμα, θα πρέπει να βρεθεί ένας τρόπος αποσαφήνισης της κατηγορίας κάθε παραδειγματος. Το έργο αυτό επιδιώκουν να εκπληρώσουν οι μέθοδοι μη-επιβλεπόμενης μάθησης. Πιο συγκεκριμένα, στο παρόν κεφάλαιο θα εξεταστεί η μέθοδος της ομαδοποίησης  $k$ -μέσων, η οποία οδηγεί σε κατάτμηση ενός χώρου με τόσες διαστάσεις όσες και τα χαρακτηριστικά του φαινομένου σε έναν αριθμό υποχώρων, και επομένως του συνόλου δεδομένων σε ίσο αριθμό ομάδων.



Σχήμα 2.13: Παράδειγμα δεδομένων πριν και μετά την ομαδοποίηση  $k$ -μέσων.

Η μέθοδος της ομαδοποίησης  $k$ -μέσων προέρχεται από τη θεωρία σημάτων και χρησιμοποιείται ευρέως στα πλαίσια της εξόρυξης δεδομένων. Θεωρώντας κάθε παράδειγμα ως ένα σημείο στο χώρο, το αποτέλεσμα της μεθόδου είναι η διαίρεση του χώρου σε  $k$  υποχώρους, και θένας από τους οποίους ορίζει μία ομάδα. Η επιδίωξη αυτή συνδέεται άμεσα με την εύρεση των λεγόμενων κεντροειδών, σημείων που χαρακτηρίζουν κάθε μια από αυτές τις ομάδες. Όπως υποδηλώνει και το όνομά του, ένα

κεντροειδές αποτελεί το κέντρο του υποχώρου του, και υπολογίζεται ως ο μέσος των σημείων που ανήκουν στην αντίστοιχη ομάδα. Ο προσδιορισμός της ομάδας στην οποία ανήκει ένα νέο σημείο χωρίς ετικέτα γίνεται μέσω εντοπισμού του κέντρου που βρίσκεται πιο κοντά του. Όπως θα φανεί παρακάτω, η επιλογή του αριθμού  $k$  των ομάδων στις οποίες θα διαχωριστεί ο χώρος είναι κρίσιμη, καθώς έτσι θα προκύψει ο αριθμός των ετικετών για τα δεδομένα.

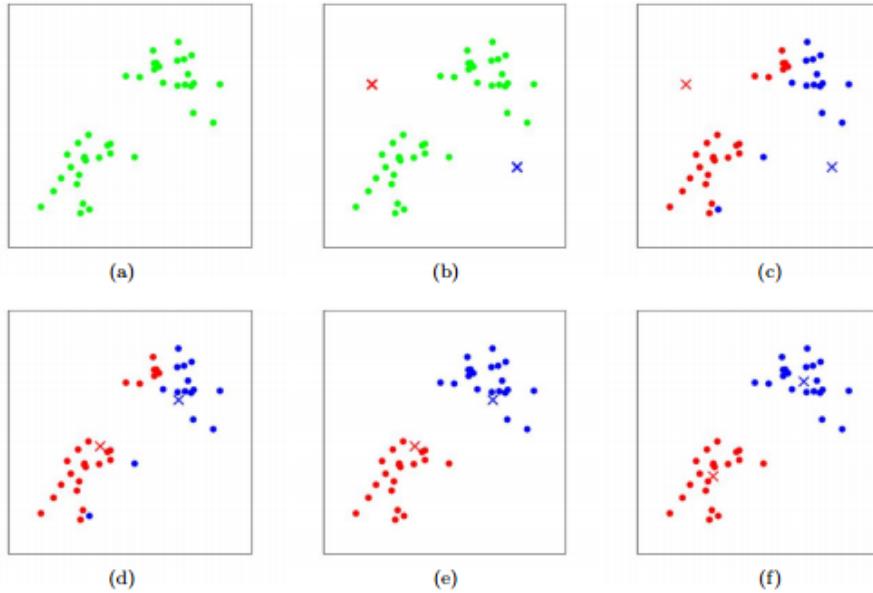
Η ομαδοποίηση  $k$ -μέσων βρίσκει εφαρμογή σε μια πληθώρα πεδίων. Χρησιμοποιείται σε χρηματιστηριακές αναλύσεις, στην υπολογιστική όραση, στην αστρονομία, αλλά ακόμα και στην ασφάλεια υπολογιστικών συστημάτων. Αποτελεί συχνά βήμα προεπεξεργασίας των δεδομένων που ακολουθείται από χρήση άλλων αλγορίθμων μηχανικής μάθησης.

### 2.7.1 Αλγόριθμος ομαδοποίησης

Κεντρικό χαρακτηριστικό της μεθόδου ομαδοποίησης  $k$ -μέσων είναι η αλληλεξάρτηση μεταξύ των κέντρων και των αντίστοιχων ομάδων που σχηματίζονται από αυτά. Όπως αναφέρθηκε, τα κέντρα προκύπτουν ως οι μέσοι όροι των σημείων της αντίστοιχης ομάδας τους. Μια επαναληπτική διαδικασία διαδοχικού επαναπροσδιορισμού ομάδων και κέντρων λαμβάνει χώρα έως ότου επέλθει σύγκλιση, δηλαδή σταθερές ομάδες που δεν αλλάζουν κατά την επόμενη επανάληψη.

Πρώτο βήμα είναι η επιλογή του αριθμού  $k$  των ομάδων στις οποίες θα διαιρεθεί ο χώρος. Η επιλογή αυτού του αριθμού είναι απαραίτητη προϋπόθεση για να γίνει η ομαδοποίηση, καθώς ο αλγόριθμος δεν είναι ικανός να τον βρει από μόνος του εκ των προτέρων. Ο αριθμός  $k$  εξαρτάται από τα δεδομένα εισόδου και από τα κριτήρια βάσει των οποίων είναι επιθυμητό να γίνει ο διαχωρισμός. Ακολουθεί ότι από διαφορετικά  $k$  θα προκύψουν διαφορετικές ομάδες. Ορισμένες φορές ο αριθμός αυτός μπορεί να εκτιμηθεί εύκολα από τον προγραμματιστή της εφαρμογής, αλλά σε πιο πολύπλοκα προβλήματα, και ιδιαίτερα όσο αυξάνονται οι διαστάσεις και δεν υπάρχει δυνατότητα απεικόνισης των δεδομένων, ο αριθμός  $k$  δεν είναι καθόλου προφανής. Μια καλή προσέγγισή του μπορεί να προκύψει μέσω της χρήσης συγκεκριμένης μεθόδου που θα αναφερθεί παρακάτω.

Κατά την έναρξη του αλγορίθμου αυτού καθαυτού, δεδομένου ότι δεν είναι γνωστές οι  $k$  ομάδες, είναι άγνωστα και τα κέντρα τους. Πρώτη εργασία λοιπόν είναι να τεθούν  $k$  σημεία ως κεντροειδή. Μια απλή τεχνική που ακολουθείται συχνά είναι να επιλεγούν  $k$  τυχαία σημεία από το σύνολο των δεδομένων, αλλά υπάρχουν και καλύτερες προσεγγίσεις όπως ο αλγόριθμος  $k$ -means++ που θα παρουσιαστεί αργότερα. Έχοντας αυτά, μπορεί να γίνει μία αρχική εκτίμηση των ομάδων.



Σχήμα 2.14: Διαδικασία διαδοχικού επαναπροσδιορισμού κέντρων και ομάδων.

Ο καθορισμός των ομάδων γίνεται υπολογίζοντας την απόσταση κάθε σημείου από κάθε κέντρο και τοποθετώντας το στην ομάδα που ορίζει το κοντινότερό του κέντρο. Ένα σημείο του συνόλου δεδομένων τοποθετείται στην ομάδα από της οποίας το κέντρο έχει την ελάχιστη ευχλείδεια απόσταση:

$$\operatorname{argmin}_{c_i \in C} \operatorname{dist}(c_i, x)$$

με:

$$\operatorname{dist}(c_i, x) = \sqrt{\sum_{j=0}^n (c_{ij} - x_j)^2}$$

όπου:

- $c_i$ , ένα κεντροειδές από το σύνολο  $C$  των κεντροειδών
- $x$ , ένα σημείο  $n$  διαστάσεων του συνόλου δεδομένων που ομαδοποιούμε

Έχοντας πλέον προσδιορίσει τις ομάδες, μπορεί να γίνει ο επαναπροσδιορισμός των κέντρων τους. Αυτό γίνεται υπολογίζοντας τον μέσο όρο των σημείων - παραδειγμάτων που έχουν τοποθετηθεί σε κάθε μια από τις ομάδες:

$$c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$

όπου:

- $S_i$ , μία από τις  $k$  ομάδες στις οποίες έχουν ομαδοποιηθεί τα παραδείγματα
- $x_j$ , κάθε σημείο της ομάδας  $S_i$

Τα παραπάνω βήματα επαναλαμβάνονται έως ότου υπάρξει σύγκλιση, δηλαδή μέχρι τα νέα κέντρα που υπολογίζονται να μην διαφέρουν από αυτά του προηγούμενου βήματος.

### 2.7.2 Επιλογή των αρχικών σημείων

Δεδομένων αρκετών επαναλήψεων, η σύγκλιση του αλγορίθμου ομαδοποίησης  $k$ -μέσων είναι εγγυημένη. Ωστόσο, λόγω της τυχαιότητας που υπάρχει στην επιλογή των αρχικών κεντροειδών, δεν αποκλείεται το ενδεχόμενο σύγκλισης σε μη-ιδανικά κέντρα, δηλαδή σύγκλιση σε ένα τοπικό ελάχιστο. Γι' αυτό το λόγο κρίνεται συχνά απαραίτητο να τρέξει ο αλγόριθμος περισσότερες από μία φορές και να διατηρηθούν τα κεντροειδή που ομαδοποιούν τα δεδομένα με τον βέλτιστο τρόπο.

Την πάρχει ωστόσο και ο αλγόριθμος επιλογής των αρχικών κεντροειδών  $k$ -means++, ο οποίος μειώνει την πιθανότητα επιλογής κακών αρχικών κεντροειδών και υπόσχεται επιπλέον ταχύτερη σύγκλιση. Η βασική ιδέα πίσω από αυτόν τον αλγόριθμο επιλογής είναι η εξάπλωση των αρχικών κέντρων σε όλο το χώρο. Το πρώτο κέντρο επιλέγεται τελείως τυχαία από τα δεδομένα, ενώ τα υπόλοιπα επιλέγονται τυχαία με πιθανότητα ανάλογη της τετραγωνικής απόστασής τους από το κοντινότερο ήδη επιλεγμένο κέντρο. Πιο συγκεκριμένα, ο αλγόριθμος έχει ως εξής:

- Επιλογή ενός τυχαίου σημείου από το σύνολο δεδομένων και ορισμός του ως το πρώτο κεντροειδές.
- Για κάθε άλλο σημείο  $x$ , υπολογισμός της  $D(x)$ , απόστασης από το κοντινότερο σημείο που έχει ήδη επιλεγεί.
- Επιλογή ενός σημείου από το σύνολο δεδομένων χρησιμοποιώντας σταθμισμένη κατανομή πιθανότητας όπου το σημείο  $x$  επιλέγεται με πιθανότητα ανάλογη του  $D(x)^2$ .
- Επανάληψη των βημάτων 2 και 3 έως ότου έχουν επιλεγεί τα  $k$  αρχικά κεντροειδή.

Αξίζει να σημειωθεί ότι αν και η συγκεκριμένη μέθοδος κάνει την αρχικοποίηση αρκετά πιο ακριβή υπολογιστικά, ο αλγόριθμος ομαδοποίησης  $k$ -μέσων που τρέχει στη συνέχεια, συγκλίνει πιο γρήγορα απ' ό,τι στην περίπτωση τυχαίας επιλογής των αρχικών κέντρων, και με σημαντικά μικρότερη πιθανότητα σύγκλισης σε τοπικό ελάχιστο.

### 2.7.3 Επιλογή του $k$

Υπάρχουν φορές που η εφαρμογή κάνει τον αριθμό των ομάδων  $k$  προφανή. Παραδείγματος χάριν, διαθέτοντας μετρήσεις για αριθμητικά σύμβολα, είναι γνωστό εκ των προτέρων ότι θα σχηματίζονται δέκα ομάδες (από 0 έως 9), ενώ αν σκοπός της εφαρμογής είναι η επιγραφή δεδομένων που αφορούν την κατάσταση υγείας ασθενών καθένας εκ των οποίων πάσχει από μία ασθένεια ενός συνόλου πέντε πιθανών, θα σχηματίζονται προφανώς πέντε ομάδες. Σε περίπτωση δύο ή τριών διαστάσεων, η απεικόνιση των δεδομένων μπορεί επίσης να αποβεί χρήσιμη για τον προσδιορισμό του  $k$ .

Παρόλα αυτά, δεν είναι λίγες οι περιπτώσεις όπου δεν μπορεί να γίνει καμία αρκετά καλή υπόθεση για τον αριθμό αυτών των ομάδων. Δυστυχώς δεν υπάρχει κάποια αναλυτική μέθοδος προσδιορισμού του ακριβούς αριθμού, πράγμα που αφήνει ως μοναδική επιλογή τη δοκιμή πολλών διαφορετικών  $k$  και χρήση κάποιου μέτρου που να αξιολογεί πόσο καλή ομαδοποίηση κάνει καθένα από αυτά. Το μέτρο που χρησιμοποιείται πιο συχνά είναι η μέση απόσταση των σημείων δεδομένων από τα κεντροειδή της ομάδας τους, που υπολογίζεται μέσω του συνολικού αυθοίσματος των τετραγώνων (TSS) για κάθε ομάδα:

$$\bar{d} = \frac{1}{|S|} \sum_{i=1}^k TSS_i$$

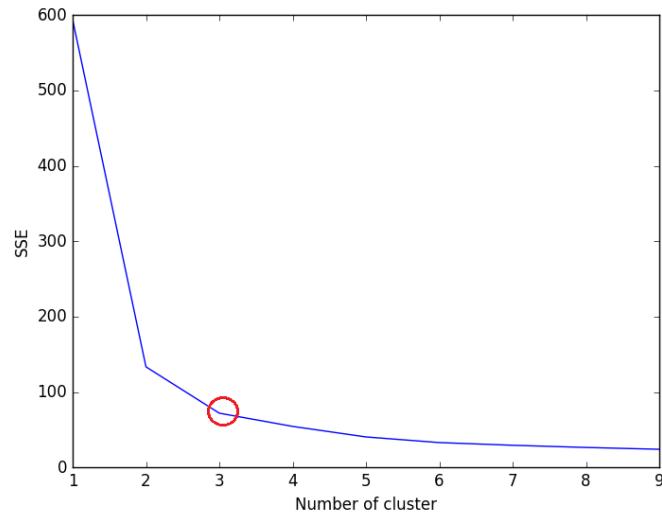
με:

$$TSS_i = \sum_{x_i \in S_i} (x_i - c_i)^2$$

όπου:

- $S$ , το σύνολο των παραδειγμάτων
- $S_i$ , μία από τις  $k$  ομάδες στις οποίες έχουν ομαδοποιηθεί τα παραδείγματα
- $x_i$ , σημείο της ομάδας  $S_i$
- $c_i$ , το κέντρο της ομάδας  $S_i$

Είναι προφανές ότι αυτό το μέτρο θα μειώνεται συνεχώς όσο αυξάνεται το  $k$ , έως ότου γίνει μηδενισμός του όταν ο αριθμός των ομάδων ταυτιστεί απόλυτα με τον αριθμό των σημείων στα δεδομένα, και επομένως επέλθει πλήρης ταύτιση μεταξύ των σημείων στα δεδομένα και των κέντρων. Αυτό είναι μια περίπτωση υπερπροσαρμογής που πρέπει να αποφευχθεί, και κατ' αυτό τον τρόπο γίνεται κατανοητό ότι το μέτρο αυτό από μόνο του δεν επαρκεί.



Σχήμα 2.15: Η μέθοδος του αγκώνα μπορεί να εφαρμοστεί φτιάχνοντας τη γραφική παράσταση μεταξύ του συνολικού αυθοίσματος τετραγώνων και του αριθμού των ομάδων.

Ο τρόπος με τον οποίο μπορεί να γίνει χρήση της μέσης απόστασης για προσδιορισμό του  $k$  είναι η λεγόμενη μέθοδος του αγκώνα, όπου γίνεται σχεδιασμός του ρυθμού μείωσης της μέσης απόστασης καθώς αυξάνεται το  $k$ . Το σημείο όπου σταματάει η οζεία μείωση της μέσης απόστασης είναι και το σημείο στο οποίο βρίσκεται το ιδανικό  $k$ .

# Κεφάλαιο 3

## Σχεδιασμός του API

Κατά τη χρήση μιας βιβλιοθήκης για την κατασκευή εφαρμογών, ο προγραμματιστής αλληλεπιδρά κυρίως με τη διεπαφή προγραμματισμού εφαρμογών, δηλαδή το λεγόμενο API. Ένα API αποτελείται από τον ορισμό των υπορουτίνων και των εργαλείων που θα χρησιμοποιηθούν για την ανάπτυξη εφαρμογών. Το API της βιβλιοθήκης «metis» έχει σχεδιαστεί έτσι ώστε να είναι εύκολο στη χρήση, και να παραφένει συνεπές μεταξύ των διαφορετικών δυνατοτήτων που προσφέρονται. Κάθε μέθοδος μηχανικής μάθησης στη βιβλιοθήκη αντιπροσωπεύεται μέσω μιας κλάσης, της οποίας οι βασικές παράμετροι, μεταβλητές και συναρτήσεις θα παρουσιαστούν στο παρόν κεφάλαιο.

### 3.1 Κλάση DataSet

Η αποθήκευση των συνόλων δεδομένων που προορίζονται να χρησιμοποιηθούν για εκπαίδευση μοντέλων μηχανικής μάθησης είναι ιδιαίτερα συνηθισμένο να γίνεται σε μορφή κειμένου. Η μορφή αυτή προκύπτει συνήθως ως αποτέλεσμα επεξεργασίας ενός άλλου μέσου, όπως οπτικού ή ακουστικού, και η παρούσα βιβλιοθήκη δεν θα ασχοληθεί με την παραγωγή της. Έχει ωστόσο παραχθεί προς διευκόλυνση του προγραμματιστή μία κλάση ανάγνωσης αρχείων κειμένου με σκοπό την προετοιμασία των συνόλων δεδομένων που είναι απαραίτητα για τους αλγορίθμους που υλοποιούνται.

Η κλάση `DataSet` έχει τη δυνατότητα να αναγιγνώσκει ένα αρχείο η τοποθεσία του οποίου της γνωστοποιείται κατά την αρχικοποίηση, λαμβάνοντας επιπλέον ως είσοδο τον διαχωριστικό χαρακτήρα μεταξύ των εγγραφών:

- `filePath`, αλφαριθμητικό που περιέχει την τοποθεσία του αρχείου προς ανάγνωση.
- `separator`, ο χαρακτήρας που διαχωρίζει μεταξύ τους τις τιμές των πεδίων στο

DataSet
<pre> - data : MatrixXd - categories : vector&lt;vector&lt;string&gt;&gt; - idOfCategory : vector&lt;map&lt;string, unsigned int&gt;&gt; - means : VectorXd - stDev : VectorXd - nInstances : unsigned int - nAttributes : unsigned int - nCategories : vector&lt;unsigned int&gt; - filePath : string - separator : char - nRows : unsigned int - nColumns : unsigned int  - readThrough() : void + create(attributes : vector&lt;unsigned int&gt;, categoricalAttr : vector&lt;unsigned int&gt;, dictionary : vector&lt;map&lt;string, unsigned int&gt;&gt;) : void + print() : void + producePlotInstructions(unsigned int, unsigned int) : void + instances() : unsigned int + attributes() : unsigned int + categories(unsigned int) : unsigned int + shuffle(VectorXi) : void + shrink(unsigned int, unsigned int) : void + convertToBinaryAttributes() : void + applyStandardization() : void + applyLogTransform() : void + applyExpTransform() : void + scaleSquaredLengthToOne() : void + DataSet(filePath : string, separator : char) + DataSet(data : MatrixXd) </pre>

Σχήμα 3.1: Διάγραμμα της κλάσης DataSet

αρχείο. Ο χαρακτήρας που διαχωρίζει μεταξύ τους τα διαφορετικά παραδείγματα θεωρείται ότι είναι πάντα η αλλαγή γραμμής.

Το σύνολο δεδομένων δημιουργείται αφού ο χρήστης γνωστοποιήσει τον τρόπο οργάνωσης του αρχείου και τα πεδία που είναι χρήσιμα για την δεδομένη εφαρμογή που παράγεται. Αυτό γίνεται με κλήση της αντίστοιχης μεθόδου:

- `create(param)`, δέχεται ως είσοδο τα πεδία του αρχείου που πρόκειται να χρησιμοποιηθούν ως χαρακτηριστικά του συνόλου δεδομένων και ποιά από αυτά αποτελούν ονομαστικές και όχι αριθμητικές τιμές. Προαιρετικά δέχεται και ένα χάρτη αντιστοίχισης ονομαστικών τιμών με αριθμητικές. Η μέθοδος υλοποιεί τη

διαδικασία μετατροπής των σχετικών δεδομένων του αρχείου από αλφαριθμητικά σε αριθμητικές τιμές.

Τα δεδομένα αποθηκεύονται σε μεταβλητές τύπου `MatrixXd` που παρέχει η βιβλιοθήκη Eigen. Εκτός από αυτά, στα πεδία της κλάσης διατηρούνται χρήσιμες πληροφορίες για το σύνολο δεδομένων. Συγκεκριμένα, οι μεταβλητές μέλη της κλάσης είναι οι εξής:

- `data`, ο πίνακας `MatrixXd` που περιέχει τα δεδομένα αυτά καθαυτά.
- `categories`, ένα διάνυσμα που περιέχει τις πιθανές τιμές για κάθε κατηγορία ονομαστικών ή διαχριτών τιμών.
- `idOfCategory`, χάρτης αντιστοίχισης ονομαστικών τιμών με αριθμητικές.
- `means`, η μέση τιμή για κάθε κατηγορία των δεδομένων.
- `stDev`, η τυπική απόκλιση για κάθε κατηγορία των δεδομένων.
- `nInstances`, ο αριθμός των παραδειγμάτων.
- `nAttributes`, ο αριθμός των κατηγοριών σε κάθε παράδειγμα.
- `nCategories`, ο αριθμός των πιθανών τιμών κάθε κατηγορίας ονομαστικών ή διαχριτών τιμών.

Ως γνωστόν, σε πολλούς αλγορίθμους μηχανικής μάθησης είναι συχνά χρήσιμο ή ακόμα και απαραίτητο να μπορούν να γίνουν μαθηματικοί και μη μετασχηματισμοί πάνω στα δεδομένα πριν αυτά χρησιμοποιηθούν για εκπαίδευση κάποιου μοντέλου. Έτσι, εκτός από τις μεθόδους δημιουργίας των συνόλων δεδομένων, η κλάση είναι επιπλέον εφοδιασμένη με έναν αριθμό μεθόδων που εφαρμόζουν απλούς μετασχηματισμούς στα δεδομένα:

- `shuffle(param)`, ανακατεύει τα δεδομένα βάσει της σειράς που αναδεικνύεται σε ένα διάνυσμα ακεραίων αριθμών που δέχεται ως είσοδο.
- `shrink(param)`, συρρικνώνει το σύνολο δεδομένων διατηρώντας μόνο της τιμές μεταξύ των δύο δεικτών που δέχεται ως είσοδο.
- `convertToBinaryAttributes()`, μετατρέπει μία κατηγορία με  $n$  πιθανές τιμές σε  $n$  κατηγορίες με πιθανές τιμές το 0 και το 1.
- `applyStandardization()`, εφαρμόζει standardization στα δεδομένα.
- `applyLogTransform()`, εφαρμόζει λογαριθμικό μετασχηματισμό στα δεδομένα.
- `applyExpTransform()`, εφαρμόζει εκθετικό μετασχηματισμό στα δεδομένα.
- `scaleSquaredLengthToOne()`, αλλάζει την τετραγωνισμένη ευχλείδια απόσταση των δεδομένων σε 1.

### 3.1.1 Κλάση DataLabeled

DataSet
- input : DataSet *
- output : DataSet *
+ setInput(DataSet *) : void
+ setOutput(DataSet *) : void
+ shuffle() : void
+ split(double) : DataLabeled *
+ instances() : unsigned int
+ inputs() : unsigned int
+ outputs() : unsigned int
+ categories(unsigned int) : unsigned int
+ classes(unsigned int) : unsigned int
- produceRegressionPlotInstructions(unsigned int, unsigned int) : void
- produceClusteringPlotInstructions(unsigned int, unsigned int) : void
+ producePlotInstructions(vector<unsigned int>, vector<unsigned int>) : void
+ DataLabeled(input : DataSet *, output : DataSet *)
+ DataLabeled(input : DataSet *, labels : VectorXi)

Σχήμα 3.2: Διάγραμμα της κλάσης DataLabeled

Στα πλαίσια της επιβλεπόμενης μάθησης γίνεται διαχείριση και επεξεργασία ενός ζεύγους δεδομένων εισόδου και εξόδου ή, διαφορετικά, ενός αριθμού παραδειγμάτων και των ετικετών τους. Γι' αυτό το σκοπό έχει δημιουργηθεί και η κλάση DataLabeled η οποία δίνει επιπλέον στον προγραμματιστή της εφαρμογής τη δυνατότητα να εφαρμόσει ορισμένες τροποποιήσεις ταυτοχρόνως στα δεδομένα εισόδου και εξόδου.

Η κλάση μπορεί να αρχικοποιηθεί με δύο τρόπους. Πρώτος είναι φυσικά με τους δείκτες προς τα δεδομένα εισόδου και εξόδου ως παραμέτρους. Ο δεύτερος, που μπορεί να χρησιμεύσει στην εκ νέου δημιουργία ζεύγους δεδομένων εισόδου και εξόδου σε περιπτώσεις αλγορίθμων ομαδοποίησης όπως η ομαδοποίηση  $k$ -μέσων δέχεται τις εξής παραμέτρους:

- **input**, δείκτης προς τα δεδομένα εισόδου.
- **labels**, ένα διάνυσμα `VectorXi` ακεραίων αριθμών, καθένας από τους οποίους αντιπροσωπεύει την κλάση στην οποία ανήκει το αντίστοιχο παράδειγμα.

Τλοποιείται επιπλέον ένας αριθμός μεθόδων που απευθύνονται ταυτόχρονα στα δεδομένα εισόδου και εξόδου:

- `shuffle()`, ανακατεύει τα δεδομένα εισόδου και εξόδου αλλάζοντας τη σειρά των γραμμών των πινάκων στους οποίους βρίσκονται με κοινό τρόπο.
- `split(param)`, διαιρεί τα δεδομένα και επιστρέφει δείκτη προς μία νέα περίσταση της κλάσης `DataLabeled` που έχει τόσα δεδομένα όσα ορίζονται από το ποσοστό που δέχεται ως είσοδο.
- `instances()`, επιστρέφει τον αριθμό των παραδειγμάτων.
- `inputs()`, επιστρέφει τον αριθμό των χαρακτηριστικών εισόδου.
- `outputs()`, επιστρέφει τον αριθμό των χαρακτηριστικών εξόδου.
- `categories(param)`, επιστρέφει τον αριθμό των διαχριτών τιμών που μπορεί να λάβει το χαρακτηριστικό εισόδου που τίθεται ως είσοδος.
- `classes(param)`, επιστρέφει τον αριθμό των διαχριτών τιμών που μπορεί να λάβει το χαρακτηριστικό εξόδου που τίθεται ως είσοδος.

## 3.2 Κλάση LinearRegression

LinearRegression
<pre> - coeff : MatrixXd - intercept : VectorXd - nAttributes : unsigned int - nOutputs : unsigned int  + fit(DataLabeled *) : void - simpleLinearRegression(DataLabeled *) : void - ordinaryLeastSquares(DataLabeled *) : void + predict(MatrixXd *) : MatrixXd + score(DataLabeled *) : VectorXd + getCoefficients() : MatrixXd + getIntercepts() : VectorXd + LinearRegression() </pre>

Σχήμα 3.3: Διάγραμμα κλάσης `LinearRegression`

Η κλάση `LinearRegression` της βιβλιοθήκης αναπαριστά ένα γραμμικό μοντέλο και υλοποιεί τη λειτουργικότητα που απαιτείται για την προσαρμογή του πάνω σε ένα σύνολο δεδομένων εκπαίδευσης βάσει των τεχνικών γραμμικής παλινδρόμησης που έχουν περιγραφεί. Ένα στιγμιότυπο της κλάσης μπορεί ύστερα από εκπαίδευση να κάνει προβλέψεις για την τιμή μιας ή περισσοτέρων εξαρτημένων μεταβλητών λαμβάνοντας ως είσοδο ένα διάνυσμα ανεξάρτητων μεταβλητών. Η αρχικοπόίηση ενός αντικειμένου της κλάσης δεν απαιτεί κάποια ιδιαίτερη παράμετρο.

Κατά την προσαρμογή του μοντέλου στα δεδομένα γίνεται προσδιορισμός των παραμέτρων της γραμμικής προγνωστικής συνάρτησης, οι οποίες αποτελούν τις βασικές μεταβλητές μέλη της κλάσης:

- `coeff`, αποτελεί έναν πίνακα πραγματικών αριθμών με τόσες στήλες όσες και τα χαρακτηριστικά εξόδου, κάθε γραμμή του οποίου αποτελεί το διάνυσμα των παραμέτρων της γραμμικής προγνωστικής συνάρτησης για το αντίστοιχο χαρακτηριστικό εξόδου.
- `intercept`, αποτελεί ένα διάνυσμα πραγματικών αριθμών κάθε τιμή του οποίου αποτελεί την απόκλιση για το αντίστοιχο χαρακτηριστικό εξόδου.

Η προσαρμογή του μοντέλου στην περίπτωση μιας μόνο επεξηγηματικής μεταβλητής, δηλαδή ενός μοναδικού χαρακτηριστικού στα δεδομένα εισόδου, γίνεται με χρήση της τεχνικής της απλής γραμμικής παλινδρόμησης, ενώ σε περίπτωση περισσότερων της μίας επεξηγηματικών μεταβλητών, χρησιμοποιείται η μέθοδος των ελαχίστων τετραγώνων. Δημιουργούνται τόσα γραμμικά μοντέλα όσες και οι μεταβλητές εξόδου.

Το μοντέλο προσαρμόζεται με κλήση της μεθόδου `fit(param)`, ενώ πέραν αυτής υπάρχουν μέθοδοι που χρησιμοποιούνται για να γίνουν προβλέψεις σε άγνωστα δεδομένα και αξιολόγηση του μοντέλου που έχει δημιουργηθεί. Οι βασικές μέθοδοι πρόσβασης φαίνονται παρακάτω:

- `fit(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων εκπαίδευσης που αποτελείται από ένα ζεύγος δεδομένων εισόδου και εξόδου, και προσδιορίζει τις παραμέτρους του γραμμικού μοντέλου.
- `predict(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων εισόδου και κάνει προβλέψεις για τις τιμές των εξόδων.
- `score(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων δοκιμής και επιστρέφει το μέσο τετραγωνικό σφάλμα για την κάθε μεταβλητή εξόδου.

### 3.3 Κλάση LogisticRegression

Το λογιστικό μοντέλο και η διαδικασία εκπαίδευσής του στη βιβλιοθήκη της εργασίας υλοποιούνται μέσω της κλάσης `LogisticRegression`. Ένας στιγμιότυπο της εν λόγω κλάσης περιγράφει έναν ταξινομητή λογιστικής παλινδρόμησης που αφού εκπαίδευτεί, μπορεί να χρησιμοποιηθεί για να κάνει προβλέψεις για την κατηγορία ενός αταξινόμητου παραδείγματος δεχόμενος το διάνυσμα των επεξηγηματικών μεταβλητών του ως είσοδο. Η κλάση υλοποιεί τον αλγόριθμο εκπαίδευσης του ταξινομητή χρησιμοποιώντας την

LogisticRegression
<pre> - coeff : MatrixXd - intercept : VectorXd - iterations : unsigned int - learnRate : double - batchSize : unsigned int - nAttributes : unsigned int - nClasses : unsigned int - verbose : bool  + fit(DataLabeled *, DataLabeled *) : double - batchGradientDescent(c : unsigned int) : void - stochasticGradientDescent(c : unsigned int) : void + predictProbabilities(MatrixXd *) : MatrixXd + predict(data : Eigen::MatrixXd *) : VectorXi + score(data : DataLabeled *) : double + LogisticRegression(iterations : unsigned int, learnRate : double, batchSize : unsigned int) : double </pre>

Σχήμα 3.4: Διάγραμμα της κλάσης LogisticRegression

μέθοδο εκπαίδευσης για τη λογιστική παλινδρόμηση που έχει περιγραφεί στο Κεφάλαιο 2, δηλαδή τον αλγόριθμο επικλινούς καθόδου. Η αρχικοποίηση της κλάσης λαμβάνει τις εξής παραμέτρους:

- **iterations**, ο μέγιστος αριθμός επαναλήψεων που θα κάνει ο αλγόριθμος εκπαίδευσης.
- **learnRate**, ο ρυθμός μάθησης.
- **batchSize**, ο αριθμός παραδειγμάτων που χρησιμοποιούνται σε κάθε βήμα εκπαίδευσης.

Εκπαίδευση του ταξινομητή σημαίνει ουσιαστικά προσδιορισμό των παραμέτρων του λογιστικού μοντέλου, δηλαδή των παραμέτρων της γραμμικής προγνωστικής συνάρτησης που όταν εφαρμοστεί στη σιγμοειδή συνάρτηση παράγει την πιθανότητα να ανήκει ένα παράδειγμα σε μια συγκεκριμένη κατηγορία. Ο επαναληπτικός αλγόριθμος της επικλινούς καθόδου επιχειρεί να βρει τις παραμέτρους που ελαχιστοποιούν τη συνάρτηση κόστους διορθώνοντας σε κάθε επανάληψή του τις παραμέτρους έτσι ώστε να μειώνεται η απόκλιση των προβλέψεων από τις αναμενόμενες εξόδους. Αυτοί οι παράμετροι αποτελούν και τις βασικές μεταβλητές μέλη της κλάσης:

- **coeff**, αποτελεί έναν πίνακα πραγματικών αριθμών που διατηρεί τις τιμές των παραμέτρων της γραμμικής προγνωστικής συνάρτησης που χρησιμοποιούνται για τον υπολογισμό της τιμής των λογαρίθμων των πιθανοτήτων. Σε κάθε γραμμή του αποθηκεύονται οι αντίστοιχες παράμετροι για μία κλάση.

- `intercept`, αποτελεί ένα διάνυσμα πραγματικών αριθμών που αποθηκεύει την τιμή της απόκλισης. Κάθε τιμή αντιστοιχεί στην απόκλιση για την αντίστοιχη κλάση.

Μια σημαντική λεπτομέρεια στην εκπαίδευση είναι ο αριθμός των παραδειγμάτων εκπαίδευσης που χρησιμοποιούνται σε κάθε επανάληψη, δηλαδή ο αριθμός των παραδειγμάτων που χρησιμοποιούνται για να διορθωθούν οι παράμετροι σε κάθε επανάληψη του αλγορίθμου. Στην υλοποίηση που έγινε υπάρχει δυνατότητα χρήσης ενός μόνο παραδείγματος ανά επανάληψη, (χρήσης δηλαδή της στοχαστικής επικλινούς καθόδου), όλων των παραδειγμάτων, ή ενός αριθμού (mini-batch) παραδειγμάτων που καθορίζει ο χρήστης. Κάθε περίπτωση έχει προφανώς διαφορετικά πλεονεκτήματα και μειονεκτήματα. Οι μέθοδοι πρόσβασης της κλάσης είναι οι εξής:

- `fit(param)`, η βασική μέθοδος εκπαίδευσης. Δέχεται ως είσοδο ένα σύνολο δεδομένων εκπαίδευσης και προαιρετικά ένα σύνολο δοκιμής. Κατά την εκτέλεσή της προσδιορίζονται οι παράμετροι του λογιστικού μοντέλου.
- `predictProbabilities(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και επιστρέφει την πιθανότητα να ανήκει το κάθε παράδειγμα σε κάθε μία από τις πιθανές κλάσεις.
- `predict(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και επιστρέφει την κλάση που είναι πιο πιθανό να ανήκει το κάθε παράδειγμα χρησιμοποιώντας την μέθοδο One-vs-Rest.
- `score(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων με ετικέτα και επιστρέφει το ποσοστό των παραδειγμάτων που ταξινομήθηκαν σωστά.

## 3.4 Κλάση `MLPClassifier`

Η κλάση `MLPClassifier` υλοποιεί έναν ταξινομητή multilayer perceptron, που αποτελεί ένα πλήρως διασυνδεδεμένο TNΔ εμπρόσθιας τροφοδότησης. Υπάρχει η επιθυμία ένα αντικείμενο της κλάσης να έχει τη δυνατότητα να δέχεται ως είσοδο έναν αριθμό παραδειγμάτων με ετικέτα, και βάσει αυτών να εκπαιδεύει ένα νευρωνικό δίκτυο έτσι ώστε να τα ταξινομεί όσο το δυνατόν καλύτερα. Θα πρέπει ύστερα να υπάρχει δυνατότητα ταξινόμησης δεδομένων χωρίς ετικέτα.

Ένα TNΔ της κλάσης `MLPClassifier` μπορεί να έχει ένα ή περισσότερα χρυφά επίπεδα, και κατά τη μετάδοση από ένα επίπεδο προς το επόμενο μπορεί να χρησιμοποιηθεί διαφορετική συνάρτηση ενεργοποίησης. Η αρχικοποίηση της κλάσης γίνεται με τις εξής παραμέτρους:

```

MLPClassifier
- coeff : vector<MatrixXd>
- intercept : vector<VectorXd>
- topology : vector<unsigned int>
- activation : vector<unsigned int>
- batchSize : unsigned int
- learnRate : double
- iterations : unsigned int
- nLayers : unsigned int
- nAttributes : unsigned int
- nClasses : unsigned int
- verbose : bool
+ train(DataLabeled *, DataLabeled *) : double
- batchGradientDescent() : void
- stochasticGradientDescent() : void
- combineWeights() : void
+ predictProbabilities(MatrixXd *) : MatrixXd
+ predict(MatrixXd *) : VectorXi
+ score(DataLabeled *) : double
- activationFunction(MatrixXd *, unsigned int) : void
- activationSigmoid(MatrixXd *) : void
- activationReLU(MatrixXd *) : void
- derivativeFunction(MatrixXd *, MatrixXd *, unsigned int) : void
- derivativeSigmoid(MatrixXd *) : void
- derivativeReLU(MatrixXd *) : void
+ MLPClassifier(hiddenLayers : vector<unsigned int>,
activationFunction : vector<unsigned int>, batchSize : unsigned
int, learnRate : double, iterations : unsigned) : double

```

Σχήμα 3.5: Διάγραμμα της χλάσης MLPClassifier

- **hiddenLayers**, ένα διάνυσμα με τόσους ακέραιους αριθμούς όσα και τα κρυφά επίπεδα. Καθένας από αυτούς συμβολίζει τον αριθμό νευρώνων του αντίστοιχου κρυφού επιπέδου.
- **activationFunction**, ένα διάνυσμα με τόσους ακέραιους αριθμούς όσα και τα επίπεδα, πλην του επιπέδου εισόδου. Κάθε ακέραιος συμβολίζει μία συνάρτηση ενεργοποίησης και αντιστοιχεί στη μετάδοση από το προηγούμενο επίπεδο προς το επόμενο. Το 0 αντιστοιχεί στην σιγμοειδή συνάρτηση, και το 1 στη ReLU.
- **batchSize**, ακέραιος αριθμός που αντιστοιχεί στο μέγεθος του mini-batch. Ο αριθμός δηλαδή των παραδειγμάτων που χρησιμοποιούνται σε κάθε βήμα εκπαίδευσης.
- **learnRate**, πραγματικός αριθμός που αντιστοιχεί στο ρυθμό μάθησης.
- **iterations**, ακέραιος αριθμός που αντιστοιχεί στο μέγιστο αριθμό επαναλήψεων.

Εκπαίδευση του νευρωνικού δικτύου συνεπάγεται προσδιορισμό των συντελεστών βαρύτητας των συνάψεων που συνδέουν μεταξύ τους, τους νευρώνες δύο διαδοχικών επιπέδων. Όπως έχει περιγραφεί στο Κεφάλαιο 2, αυτό γίνεται μέσω τη μεθόδου οπισθοδρομικής διάδοσης του σφάλματος, μιας επαναληπτικής μεθόδου επαναπροσδιορισμού των συντελεστών βαρύτητας. Τα βασικά χαρακτηριστικά της κλάσης του TNΔ περιγράφονται από τις εξής μεταβλητές:

- **coeff**, τόσοι πίνακες πραγματικών αριθμών όσα και τα χρυφά επίπεδα του δικτύου. Καθένας από αυτούς αποθηκεύει τους συντελεστές βαρύτητας των συνάψεων που συνδέουν τους νευρώνες δύο διαδοχικών επιπέδων.
- **intercept**, τόσα διανύσματα πραγματικών αριθμών όσα και τα επίπεδα του δικτύου. Καθένα από αυτά περιέχει την απόκλιση που αντιστοιχεί στις συνάψεις που συνδέουν τους νευρώνες δύο διαδοχικών επιπέδων.
- **topology**, ένα διάνυσμα με τόσους ακέραιους αριθμούς όσα και τα επίπεδα του TNΔ. Κάθε ακέραιος αντιστοιχεί στον αριθμό νευρώνων του αντίστοιχου επιπέδου.

Η εκπαίδευση ζεκινάει με κλήση της μεθόδου **train(param)**, ενώ παρέχονται επιπλέον μέθοδοι για να γίνονται οι προβλέψεις. Οι βασικές μέθοδοι πρόσβασης της κλάσης είναι οι εξής:

- **train(param)**, η βασική μέθοδος εκπαίδευσης. Δέχεται ως είσοδο ένα σύνολο δεδομένων εκπαίδευσης και προαιρετικά ένα σύνολο δοκιμής. Κατά την εκτέλεσή της προσδιορίζονται οι συντελεστές βαρύτητας των συνάψεων μεταξύ των νευρώνων κάθε ζεύγους διαδοχικών επιπέδων.
- **predictProbabilities(param)**, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και επιστρέφει την πιθανότητα να ανήκει το κάθε παράδειγμα σε κάθε μία από τις πιθανές κλάσεις.
- **predict(param)**, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και επιστρέφει την κλάση που είναι πιο πιθανό να ανήκει το κάθε παράδειγμα.
- **score(param)**, δέχεται ως είσοδο ένα σύνολο δεδομένων με ετικέτα και επιστρέφει το ποσοστό των παραδειγμάτων που ταξινομήθηκαν σωστά.

### 3.5 Κλάση NaiveBayes

Η ταξινόμηση με χρήση του απλού ταξινομητή Bayes έχει περιγραφεί στο Κεφάλαιο 2, και η υλοποίησή της γίνεται μέσω της κλάσης **template NaiveBayes**, η οποία όπως

```

<<abstract>>
NaiveBayes <T>

- prior : VectorXd
- nAttributes : unsigned int
- nClasses : unsigned int

+ fit(DataLabeled *) : void
+ findPrior(MatrixXd *) : VectorXd
+ findEvidence(unsigned int, MatrixXd *) : VectorXd
+ findLikelihood(unsigned int, MatrixXd *, unsigned int) :
VectorXd
+ findLogLikelihood(unsigned int, MatrixXd *, unsigned int) :
VectorXd
+ findPosterior(MatrixXd *) : VectorXd
+ findRelativePosterior(MatrixXd *) : VectorXd
+ findLogPosterior(MatrixXd *) : VectorXd
+ predict(MatrixXd *, bool) : VectorXi
+ score(DataLabeled *, bool) : double

```

Σχήμα 3.6: Διάγραμμα κλάσης NaiveBayes &lt;T&gt;

Θα φανεί παρακάτω υλοποιείται ανεξάρτητα για την περίπτωση του multinomial και του Gaussian ταξινομητή. Ένα αντικείμενο της κλάσης θα μπορεί να εκπαιδευτεί πάνω σε ένα σύνολο παραδειγμάτων με ετικέτα και εν συνεχείᾳ θα μπορεί να χρησιμοποιηθεί για προσδιορισμό της ετικέτας οποιουδήποτε αγνώστου παραδείγματος.

Όπως έχει προαναφερθεί, ο απλός ταξινομητής Bayes βασίζεται στο θεώρημα του Bayes, και συνεπώς εκπαιδευσή του συνεπάγεται την εύρεση των παραμέτρων εκείνων που είναι απαραίτητοι για τον υπολογισμό των πιθανοτήτων. Οι παράμετροι αυτοί διαφέρουν μεταξύ της περίπτωσης του multinomial και του Gaussian ταξινομητή, αλλά κοινή και στις δύο περιπτώσεις είναι η εκ των προτέρων πιθανότητα της κάθε κλάσης. Έτσι, η μεταβλητή μέλος που κληρονομούν και οι δύο υλοποιήσεις του template είναι:

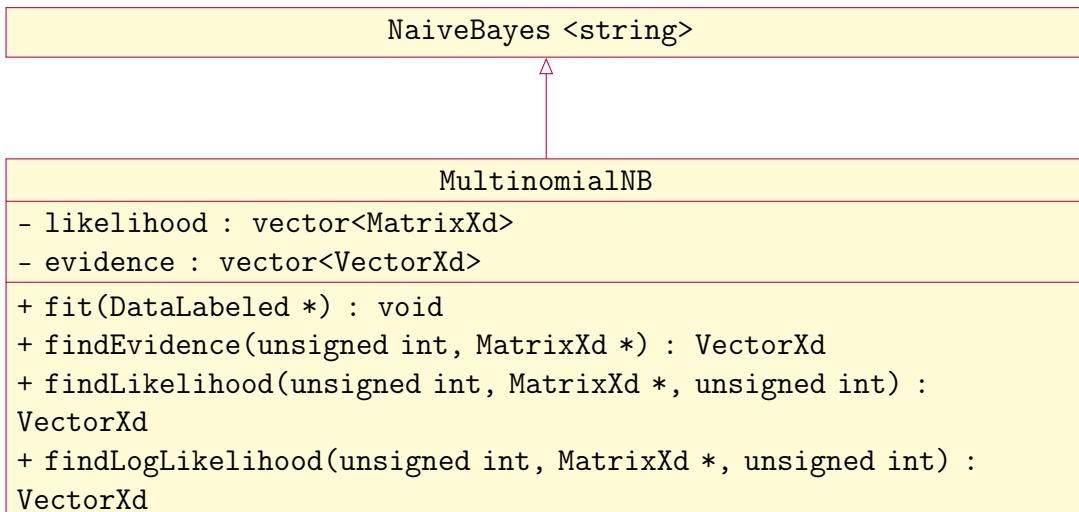
- **prior**, η συχνότητα εμφάνισης της κάθε κλάσης στα δεδομένα εισόδου.

Ένας εκπαιδευμένος απλός ταξινομητής Bayes θα πρέπει να μπορεί να παραγάγει την πιθανότητα να ανήκει ένα παράδειγμα σε κάθε μία από τις πιθανές κλάσεις. Οι μέθοδοι πρόσβασης είναι οι εξής:

- **fit(param)**, εκπαιδεύει το μοντέλο, προσδιορίζει δηλαδή τις παραμέτρους που είναι απαραίτητες για τον υπολογισμό των πιθανοτήτων. Η υλοποίησή της γίνεται ανεξάρτητα στη MultinomialNB και την GaussianNB.
- **findPrior(param)**, επιστρέφει την εκ των προτέρων πιθανότητα εμφάνισης της κάθε κλάσης στα δεδομένα εισόδου.

- `findEvidence()`, επιστρέφει την πιθανότητα ένα χαρακτηριστικό να έχει μια συγκεκριμένη τιμή. Η υλοποίησή της γίνεται ανεξάρτητα στη `MultinomialNB` και την `GaussianNB`.
- `findLikelihood(param)`, επιστρέφει την εκ των προτέρων πιθανότητα ένα χαρακτηριστικό να έχει μια συγκεκριμένη τιμή δεδομένης της κλάσης στην οποία ανήκει. Η υλοποίησή της γίνεται ανεξάρτητα στη `MultinomialNB` και την `GaussianNB`.
- `findPosterior(param)`, επιστρέφει την εκ των υστέρων πιθανότητα να ανήκει ένα παράδειγμα σε κάθε πιθανή κλάση δεδομένων συγκεκριμένων τιμών στα χαρακτηριστικά εισόδου.
- `findRelativePosterior(param)`, επιστρέφει την εκ των υστέρων σχετική πιθανότητα να ανήκει ένα παράδειγμα σε κάθε πιθανή κλάση δεδομένων συγκεκριμένων τιμών στα χαρακτηριστικά εισόδου.
- `predict(param)`, λαμβάνει ως είσοδο ένα σύνολο παραδειγμάτων χωρίς ετικέτα και παράγει μία πρόβλεψη για την κλάση καθενός από αυτά.
- `score(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων με ετικέτα και επιστρέφει το ποσοστό των παραδειγμάτων που ταξινομήθηκαν σωστά.

### 3.5.1 Κλάση `MultinomialNB`

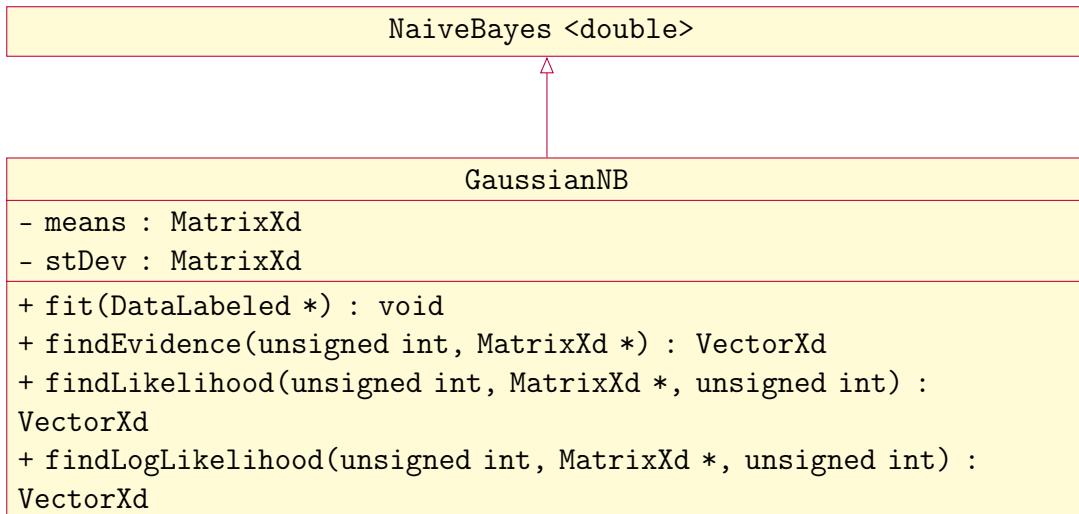


Σχήμα 3.7: Διάγραμμα της κλάσης `MultinomialNB`

Η συγκεκριμένη κλάση υλοποιεί την `NaiveBayes` με `T = string`. Στην περίπτωση του `multinomial` ταξινομητή οι παράμετροι που απαιτούνται για υπολογισμό των πιθανοτήτων είναι οι εξής:

- likelihood, η συχνότητα εμφάνισης μιας συγκεκριμένης τιμής σε ένα χαρακτηριστικό δεδομένης κλάσης.
- evidence, η συχνότητα εμφάνισης μιας συγκεκριμένης τιμής σε ένα χαρακτηριστικό.

### 3.5.2 Κλάση GaussianNB



Σχήμα 3.8: Διάγραμμα της κλάσης GaussianNB

Η συγκεκριμένη κλάση υλοποιεί την NaiveBayes με  $T = \text{double}$ . Στην περίπτωση του γκαουσιανού ταξινομητή οι παράμετροι που απαιτούνται για υπολογισμό των πιθανοτήτων είναι οι εξής:

- **means**, ο μέσος όρος.
- **stDev**, η τυπική απόκλιση.

## 3.6 Κλάση DecisionTree

Η υλοποίηση των δέντρων αποφάσεων στη βιβλιοθήκη στηρίζεται σε δύο βασικές κλάσεις: την `DecisionTree` και την `DTNode`. Η πρώτη αναπαριστά μια δομή δεδομένων δέντρου η κόμβοι του οποίου αντιπροσωπεύονται σαν περιστάσεις της δεύτερης κλάσης. Ένας ταξινομητής βασισμένος σε δέντρο αποφάσεων αναπαρίσταται ως μία περίσταση της κλάσης `DecisionTree` και τόσες περιστάσεις της `DTNode` όσοι και οι κόμβοι του δέντρου.

```

DecisionTree
- root : DTNode *
- nAttributes : unsigned int
- nOutcomes : unsigned int
+ fit(DataLabeled *) : double
+ predictProbabilities(MatrixXd *) : MatrixXd
+ predict(MatrixXd *) : VectorXi
+ score(DataLabeled *) : double
+ print() : void
+ DecisionTree()

```

Σχήμα 3.9: Διάγραμμα κλάσης DecisionTree

Κατά τη δημιουργία του, το δέντρο αποτελείται αρχικά μόνο από τη ρίζα του, δηλαδή έναν κόμβο ο οποίος δεν έχει διακλαδωθεί ακόμα. Αυτός ο κόμβος αποτελεί τη βασική μεταβλητή μέλος της κλάσης:

- **root**, δείκτης προς ένα αντικείμενο τύπου **DTNode** που λειτουργεί ως η ρίζα του δέντρου απόφασης. Η διάβαση του δέντρου γίνεται με χρήση αυτής της μεταβλητής.

Η λειτουργικότητα της κλάσης αφορά κυρίως τη βλάστηση του δέντρου, μία διεργασία που συνδέεται με την προσαρμογή του ταξινομητή στα δεδομένα εκπαίδευσης. Η βλάστηση γίνεται μέσω του αλγορίθμου ID3 και στην παρούσα υλοποίηση, μπορεί να γίνει διαχείριση μόνο χαρακτηριστικών εισόδου που λαμβάνουν ονομαστικές τιμές. Μπορεί να δημιουργηθεί δέντρο για οποιοδήποτε αριθμό χαρακτηριστικών εισόδου που λαμβάνουν έναν οποιοδήποτε πεπερασμένο αριθμό τιμών και οδηγούν σε δύο ή περισσότερες πιθανές εκβάσεις για ένα όμως μόνο χαρακτηριστικό εξόδου.

Εκτός από την ανάπτυξη του δέντρου, όλα πρέπει να μπορούν να γίνονται και προβλέψεις για δεδομένα χωρίς ετικέτα. Οι βασικές μέθοδοι πρόσβασης είναι οι εξής:

- **fit(param)**, δέχεται ως είσοδο ένα ζεύγος συνόλων δεδομένων εισόδου και εξόδου και δομεί ένα δέντρο αποφάσεων βάσει του αλγορίθμου ID3.
- **predictProbabilities(param)**, δέχεται ως είσοδο ένα σύνολο δεδομένων εισόδου χωρίς ετικέτες και επιστρέφει την πιθανότητα να ανήκει το κάθε παράδειγμα σε κάθε μια από τις πιθανές κλάσεις.
- **predict(param)**, δέχεται ως είσοδο ένα σύνολο δεδομένων εισόδου χωρίς ετικέτες και επιστρέφει την κλάση στην οποία είναι πιο πιθανό να ανήκει το κάθε παράδειγμα.
- **score(param)**, δέχεται ως είσοδο ζεύγος συνόλων δεδομένων εισόδου και εξόδου και επιστρέφει το ποσοστό των παραδειγμάτων που ταξινομήθηκαν σωστά.

- `print()`, τυπώνει το δέντρο σε μορφή κειμένου.

### 3.6.1 Κλάση DTNode

DTNode
<pre> - children : vector&lt;DTNode *&gt; - splitAttribute : unsigned int - splitValues : VectorXd - nChildren : unsigned int - entropy : double - outcomeFrequencies : VectorXd  - measureInfoGain(DataLabeled *, vector&lt;unsigned&gt;, unsigned int, vector&lt;vector&lt;unsigned&gt; *, VectorXd *) : double - findBestSplit(DataLabeled *, vector&lt;unsigned&gt;, vector&lt;unsigned int&gt;, vector&lt;vector&lt;unsigned&gt; *, VectorXd *) : unsigned int + split(DataLabeled *, vector&lt;unsigned&gt;, vector&lt;unsigned int&gt;) : void + traverse(VectorXd) : VectorXd + printSplit(unsigned int, unsigned int, double, vector&lt;vector&lt;string&gt;, vector&lt;string&gt;) : void + DTNode(double, ArrayXd) </pre>

Σχήμα 3.10: Διάγραμμα της κλάσης DTNode

Το μεγαλύτερο μέρος της λειτουργικότητας του ταξινομητή εμπεριέχεται στην κλάση DTNode. Εκεί υλοποιείται η διαδικασία επιλογής του κατάλληλου χαρακτηριστικού βάσει του οποίου πρέπει να διχοτομηθεί το σύνολο δεδομένων. Το κριτήριο που επιτάσσει ο αλγόριθμος ID3 είναι η μεγιστοποίηση του κέρδους πληροφορίας, ή διαφορετικά η ελαχιστοποίηση της εντροπίας. Όταν γίνεται διχοτόμηση του συνόλου γίνεται και αρχικοποίηση τόσων νέων κόμβων όσες οι πιθανές τιμές του χαρακτηριστικού που επιλέχθηκε. Η αρχικοποίηση γίνεται με τις εξής παραμέτρους που υπολογίζονται στο βήμα διχοτόμησης του κόμβου γονέα και χαρακτηρίζουν τον νέο κόμβο:

- `entropy`, η τιμή της εντροπίας στον συγκεκριμένο κόμβο.
- `outcomeFrequencies`, οι συχνότητες του κάθε πιθανού αποτελέσματος στον συγκεκριμένο κόμβο.

Όπως είναι γνωστό, η κόμβοι μιας δομής δεδομένων δέντρου διατηρούν δείκτες για τους κόμβους παιδιά τους, στα οποία διακλαδώνονται. Σε ένα δέντρο απόφασης, είναι επιπλέον απαραίτητα ορισμένα στοιχεία που χρησιμοποιούνται κατά τη διάβαση του δέντρου. Η πληροφορία για τα παιδιά του κάθε κόμβου που δημιουργούνται όταν αυτός διχοτομηθεί συνοψίζονται στις παρακάτω μεταβλητές μέλη:

- `nChildren`, ο αριθμός των παιδιών του συγκεκριμένου κόμβου.
- `children`, ένα διάνυσμα με δείκτες προς τις περιστάσεις της κλάσης `DTNode` που αντιπροσωπεύουν τα παιδιά του κόμβου.
- `splitAttribute`, το χαρακτηριστικό βάσει του οποίου διαχλαδώνεται ο κόμβος.
- `splitValues`, η τιμή του χαρακτηριστικού διαχλαδωσης για καθένα από τα παιδιά του κόμβου.

Η λειτουργικότητα του κόμβου αφορά εκτός από την δυνατότητα διχοτόμησης του εαυτού του και τις μεθόδους που είναι απαραίτητες για την διάβασή του. Συνολικά παρέχει τις παρακάτω μεθόδους πρόσβασης:

- `split(param)`, δέχεται ως εισόδους το υποσύνολο των δεδομένων που αφορούν τον συγκεκριμένο κόμβο και αφού προσδιοριστεί ποιό είναι το καλύτερο μεταξύ των υπόλοιπων χαρακτηριστικών βάσει των οποίων μπορεί να γίνει διαχλάδωση, δημιουργούνται τόσοι κόμβοι όσες οι πιθανές τιμές αυτού του χαρακτηριστικού, και καθένας από αυτούς τίθεται ως παιδί του παρόντος κόμβου.
- `traverse(param)`, δέχεται ως είσοδο ένα διάνυσμα με τιμές για κάθε χαρακτηριστικό, και οδηγεί προς το παιδί-κόμβο του που αντιστοιχεί στην τιμή της αντίστοιχης εισόδου.
- `printSplit(param)`, χρησιμοποιείται στα πλαίσια της τύπωσης του δέντρου.

### 3.7 Κλάση KMeans

Στη βιβλιοθήκη της εργασίας, η ομαδοποίηση  $k$ -μέσων υλοποιείται μέσω της κλάσης `KMeans`. Ένα αντικείμενο αυτής της κλάσης περιγράφει έναν ομαδοποιητή ο οποίος λαμβάνοντας ως είσοδο κάποιο σύνολο παραδειγμάτων χωρίς ετικέτες, δύναται να το ομαδοποιήσει σε έναν αριθμό ομάδων κάνοντας χρήση του αλγορίθμου ομαδοποίησης  $k$ -μέσων.

Ομαδοποίηση των δεδομένων πρακτικά σημαίνει παραγωγή μιας ετικέτας για κάθε παράδειγμα του συνόλου δεδομένων εισόδου, η οποία υποδεικνύει την ομάδα στην οποία ανήκει το συγκεκριμένο παράδειγμα. Ταυτόχρονα, σημαίνει ορισμό των ομάδων μέσω προσδιορισμού του αντίστοιχου κέντρου για κάθε μία από αυτές. Βασική παράμετρος του ομαδοποιητή θα αποτελεί ο αριθμός  $k$  που καθορίζει τον αριθμό των ομάδων στις οποίες θα διαχωριστούν τα δεδομένα. Η αρχικοποίηση της κλάσης λαμβάνει ως παραμέτρους:

- `nClusters`, τον αριθμό των ομάδων.

KMeans
<ul style="list-style-type: none"> <li>- centroids : MatrixXd</li> <li>- labels : VectorXi</li> <li>- nClusters : unsigned int</li> <li>- nDimensions : unsigned int</li> <li>- nInstances : unsigned int</li> <li>- maxIterations : unsigned int</li> </ul>
<ul style="list-style-type: none"> <li>+ cluster(data : DataSet *) : VectorXi</li> <li>+ predict(data : DataSet *) : VectorXi</li> <li>+ score(data : DataSet *) : double</li> <li>+ getCentroids() : MatrixXd</li> <li>+ getLabels() : VectorXi</li> <li>+ setMaxIterations(unsigned iterations) : void</li> <li>- initializeCentroidsRandom() : void</li> <li>- initializeCentroidsKmeansPP() : void</li> <li>- redetermineCentroids() : void</li> <li>- closestCentroid() : unsigned int</li> <li>+ KMeans(nClusters : unsigned int, randomInit : bool)</li> </ul>

Σχήμα 3.11: Διάγραμμα της κλάσης KMeans

- **randomInit**, μια μεταβλητή τύπου `bool` που καθορίζει αν η επιλογή των αρχικών κέντρων θα γίνει τυχαία.

Το σημείο επιλογής των αρχικών σημείων είναι κρίσιμο, καθώς θα καθορίσει πόσες επαναλήψεις θα χρειαστούν πριν επέλθει η σύγκλιση. Με σκοπό τη μείωση επαναλήψεων που θα χρειαστούν, στην παρούσα βιβλιοθήκη υλοποιείται ο αλγόριθμος επιλογής k-means++, που είναι η προεπιλογή που χρησιμοποιείται αν το `randomInit` δεν έχει τιμή `true`.

Οι συντεταγμένες των κεντροειδών είναι αυτές που ορίζουν πρακτικά τις ομάδες, και αυτά τα σημεία αποτελούν συνεπώς τη βασική πληροφορία που περιέχεται στην κλάση. Οι μεταβλητές μέλη της κλάσης είναι οι εξής:

- **centroids**, ένας πίνακας με τόσες γραμμές όσες οι ομάδες και τόσες στήλες όσες ο αριθμός των διαστάσεων κάθε σημείου.
- **labels**, αποτελεί ένα διάνυσμα ακεραίων αριθμών με τόσα στοιχεία όσα και τα παραδείγματα εισόδου, κάθε ένα απ' τα οποία υποδηλώνει την ομάδα του αντίστοιχου σημείου.
- **nClusters**, ο αριθμός των ομάδων.
- **nDimensions**, ο αριθμός των διαστάσεων κάθε παραδείγματος και κέντρου.
- **nInstances**, ο αριθμός των παραδειγμάτων εισόδου.
- **maxIterations**, ο μέγιστος αριθμός επαναλήψεων.

Οι βασικές μέθοδοι πρόσβασης είναι οι εξής:

- `cluster(param)`, η βασική μέθοδος του ομαδοποιητή, η οποία δέχεται ως όρισμα τα μη-ετικεταρισμένα δεδομένα και τα ομαδοποιεί προσδιορίζοντας τα κεντροειδή. Επιστρέφει επιπλέον τις ετικέτες των δεδομένων εισόδου.
- `predict(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και επιστρέφει την ομάδα κάθε παραδείγματος υπολογίζοντας το κοντινότερο κέντρο από αυτό.
- `score(param)`, δέχεται ως είσοδο ένα σύνολο δεδομένων χωρίς ετικέτα και αφού υπολογίσει την ομάδα κάθε παραδείγματος, επιστρέφει τη μέση απόσταση των σημείων από τα κέντρα της ομάδας τους.
- `getCentroids()`, επιστρέφει τα κεντροειδή.
- `getLabels()`, επιστρέφει τις ετικέτες των δεδομένων εισόδου που χρησιμοποιήθηκαν για τον προσδιορισμό των ομάδων
- `setMaxIterations(param)`, θέτει το μέγιστο αριθμό επαναλήψεων.

# Κεφάλαιο 4

## Λεπτομέρειες υλοποίησης

Στα προηγούμενα κεφάλαια παρουσιάστηκαν η θεωρία πίσω από τις μεθόδους της βιβλιοθήκης και το API που παρέχεται στο χρήστη για ανάπτυξη εφαρμογών. Όπως προαναφέρθηκε, η βιβλιοθήκη υλοποιήθηκε με χρήση της γλώσσας C++, μαζί με τη βιβλιοθήκη Eigen και το πρότυπο OpenMP. Στο παρόν κεφάλαιο θα αναλυθούν οι τεχνικές λεπτομέρειες πίσω από την υλοποίηση κάθε μεθόδου, συμπεριλαμβάνοντας κομμάτια κώδικα και δικαιολογώντας ορισμένες προγραμματιστικές επιλογές. Επιπλέον, θα εξηγηθούν οι προσεγγίσεις που ακολουθήθηκαν για την παραλληλοποίησή τους.

### 4.1 Ανάγνωση και προετοιμασία των δεδομένων

Η δομή δεδομένων μέσα στην οποία θα διατηρούνται τα δεδομένα που πρόκειται να χρησιμοποιηθούν από τους διάφορους αλγορίθμους είναι ένα θέμα στο οποίο δώθηκε αρκετή σημασία στα αρχικά στάδια ανάπτυξης της βιβλιοθήκης. Δοκιμάστηκε ένας αριθμός διαφορετικών προσεγγίσεων, όπως η αποθήκευσή τους σε απλούς πίνακες τύπου `double` και η χρήση της δομής `vector` της C++. Τελικώς όμως επιλέχθηκε να γίνει χρήση της βιβλιοθήκης Eigen, η οποία προσφέρει τις δομές `MatrixXd` και `VectorXd` και υλοποιεί επιπλέον με αποδοτικό τρόπο τόσο πράξεις γραμμικής άλγεβρας όσο και άλλους μαθηματικούς μετασχηματισμούς επάνω στα δεδομένα που είναι αποθηκευμένα σε αυτές τις δομές.

Τα δεδομένα της `MatrixXd` μεταβλητής `data` προκύπτουν από την ανάγνωση ενός αρχείου μέσω των γνωστών διεργασιών της γλώσσας C / C++. Με την αρχικοποίηση της κλάσης, η `private` μέθοδος `readThrough()` χρησιμοποιείται για προσδιορισμό του αριθμού των γραμμών και των στηλών πριν κληθεί η `create()` και αρχίσει

να γεμίζει ο πίνακας. Μια λεπτομέρεια που αξίζει να σημειωθεί είναι ότι γίνεται η υπόθεση ότι όλα τα παραδείγματα, καθένα από τα οποία ζεκινάει από νέα γραμμή, έχουν τον ίδιο αριθμό στηλών.

```

getline(dataFile, rowData);
_nRows = 1;
_nColumns = 0;

std::istringstream unscrambler(rowData);
while (getline(unscrambler, columnData, _separator))
    if (columnData[0] != 0) ++_nColumns;

while (getline(dataFile, rowData)) ++_nRows;

```

Ο αριθμός των στηλών καθορίζεται από ανάγνωση της πρώτης γραμμής.

Όπως φαίνεται και από το API, κατά τη δημιουργία του συνόλου δεδομένων λαμβάνονται ως είσοδος οι στήλες του αρχείου που θα χρησιμοποιηθούν ως χαρακτηριστικά, και επιπλέον αποσαφηνίζεται ποιές από αυτές περιέχουν ονομαστικές τιμές αντί αριθμητικών. Αν δοθεί ως είσοδος και ένας χάρτης αντιστοιχίσης των ονομαστικών τιμών με αριθμούς, αυτός θα χρησιμοποιηθεί κατά την ανάγνωση του αρχείου. Κάτι τέτοιο είναι χρήσιμο σε περίπτωση που υπάρχει ανάγκη οι ονομαστικές τιμές να αντιστοιχίζονται με τις αντίστοιχες αριθμητικές τους τιμές με κοινό τρόπο μεταξύ δύο ξεχωριστών συνόλων δεδομένων (αν παραδείγματος χάριν τα δεδομένα εκπαίδευσης και δοκιμής βρίσκονται σε ξεχωριστά αρχεία). Σε διαφορετική περίπτωση, ο χάρτης αυτός δημιουργείται σταδιακά κατά την ανάγνωση του αρχείου.

```

if (!attrIsCategorical[col])
    _data.coeffRef(row, attr) = strtod(colData.c_str(), &ch);

else {
    if (std::find(_categories[attr].begin(), _categories[attr].end(),
                  colData) == _categories[attr].end())
        _idOfCategory[attr].insert(std::pair<std::string,
                                   unsigned>(colData, _categories[attr].size()));
    _categories[attr].push_back(colData);
}
_data.coeffRef(row, attr) = (double)_idOfCategory[attr][colData];

```

Ο τρόπος διαχείρισης των ονομαστικών πεδίων διαφέρει από αυτόν των αριθμητικών.

Οι διάφοροι μετασχηματισμοί γίνονται εύκολα και αποδοτικά με χρήση των μεθόδων της Eigen. Ο λογαριθμικός μετασχηματισμός των δεδομένων για παράδειγμα

υλοποιείται με μία μόλις γραμμή.

```
// _data.array() = _data.array().log();
```

*H Eigen επιτρέπει πράξεις όπως η εφαρμογή της λογαριθμικής συνάρτησης στοιχείο-στοιχείο σε όλη την έκταση του πίνακα.*

Ένα άλλο σημείο ενδιαφέροντος είναι η υλοποίηση της μεθόδου `convertToBinaryAttributes()`. Η συγκεκριμένη μέθοδος μπορεί να κληθεί μόνο σε σύνολα δεδομένων με ένα μοναδικό χαρακτηριστικό το οποίο είναι διαχριτό και παίρνει *n* διαφορετικές τιμές. Θα δημιουργηθεί ένα νέο σύνολο δεδομένων με *n* χαρακτηριστικά τα οποία θα έχουν τιμές 0 ή 1. Η συγκεκριμένη μέθοδος είναι χρήσιμη σε περιπτώσεις όπως αυτή των νευρώνων εξόδου στα νευρωνικά δίκτυα.

```
double binLookup[_nCategories[0]][_nCategories[0]];
for (unsigned i = 0; i < _nCategories[0]; ++i) {
    for (unsigned j = 0; j < _nCategories[0]; ++j) {
        binLookup[i][j] = 0.0;
    }
    binLookup[i][i] = 1.0;
}

unsigned exampleClass;
Eigen::MatrixXd newData(_nInstances, _nCategories[0]);
for (unsigned i = 0; i < _nInstances; ++i) {
    for (unsigned c = 0; c < _nCategories[0]; ++c) {
        exampleClass = (unsigned) (_data.coeffRef(i, 0) + 0.5);
        newData.coeffRef(i, c) = binLookup[exampleClass][c];
    }
}
```

*Σε κάθε παράδειγμα το πεδίο που αντιστοιχεί στην ονομαστική τιμή που έχει προηγουμένως αυτό το παράδειγμα τίθεται σε 1, ενώ όλα τα υπόλοιπα πεδία του τίθενται σε 0.*

## 4.2 Γραμμική παλινδρόμηση

Το γραμμικό μοντέλο και η διαδικασία προσαρμογής του στα δεδομένα όπως περιγράφεται από τις τεχνικές γραμμικής παλινδρόμησης υλοποιείται από την κλάση `LinearRegression`. Για την εκπαίδευση έχει επιλεγεί η μέθοδος των ελαχίστων τετραγώνων και η απλή γραμμική παλινδρόμηση, που αποτελεί εξειδίκευσή της για την περίπτωση ύπαρξης μίας μόνο μεταβλητής εισόδου. Η υλοποίηση και των δύο είναι εξαιρετικά απλή κάνοντας χρήση των μεθόδων γραμμικής άλγεβρας που παρέχει η βι-

βιβλιοθήκη Eigen.

Η εκπαίδευση ξεκινάει με κλήση της μεθόδου `fit(param)`, και εντός αυτής επιλέγεται μία από τις `private` μεθόδους `simpleLinearRegression(param)` ή `ordinaryLeastSquares(param)`.

```
if (_nAttributes == 1) simpleLinearRegression(data);
else ordinaryLeastSquares(data);
```

Η απλή γραμμικής παλινδρόμηση χρησιμοποιείται στην περίπτωση ύπαρξης μίας ανεξάρτητης μεταβλητής.

Οι δύο αυτές αναλυτικές μέθοδοι είναι ιδιαίτερα απλές και έχουν περιγραφεί στο Κεφάλαιο 2. Η υλοποίησή τους γίνεται με αποδοτικό τρόπο χάρη στη χρήση της βιβλιοθήκης Eigen που παρέχει όλες τις πράξεις γραμμικής άλγεβρας που απαιτούνται υλοποιημένες διανυσματικά έτσι ώστε να εκμεταλλεύονται τις SIMD δυνατότητες των σύγχρονων επεξεργαστών.

```
#pragma omp parallel for
for (unsigned o = 0; o < _nOutputs; ++o) {
    _coeff.col(o) = (in->transpose() * (*in)).inverse() *
        (in->transpose()) * out->col(o);
    _intercept.coeffRef(o) = out->col(o).mean();
}
```

Τλοποίηση της μεθόδου των ελαχίστων τετραγώνων.

Θα εκπαιδευτούν ουσιαστικά τόσα γραμμικά μοντέλα όσα και οι εξαρτημένες μεταβλητές, οι παράμετροι καθενός από τα οποία τοποθετούνται στην κάθε στήλη της μεταβλητής μέλους `coeff`. Σε περίπτωση ενός μεγάλου συνόλου δεδομένων με πολλές μεταβλητές εξόδου, υπάρχει δυνατότητα παράλληλης εκτέλεσης της εκπαίδευσης του γραμμικού μοντέλου που αντιστοιχεί στην κάθε εξαρτημένη μεταβλητή. Χάρη στην `pragma` οδηγία, το OpenMP θα διαχειριστεί την παραλληλοποίηση της διαδικασίας εφόσον έχουν τεθεί πολλαπλά νήματα εκτέλεσης με την αντίστοιχη εντολή.

Οι προβλέψεις του μοντέλου υπολογίζονται με χρήση της μεθόδου `predict(param)`, που δεν κάνει τίποτα άλλο παρά να υπολογίσει τις εξόδους του γραμμικού μοντέλου κάνοντας χρήση των αντίστοιχων εισόδων και των παραμέτρων που υπολογίστηκαν.

```
predictions = (*data) * _coeff.matrix();
```

```
// predictions.rowwise() += _intercept.transpose();
```

Για τον υπολογισμό των εξόδων γίνεται αρχικά πολλαπλασιασμός των δεδομένων εισόδου με τις παραμέτρους, και προστίθεται στη συνέχεια η απόκλιση.

## 4.3 Ταξινόμηση μέσω λογιστικής παλινδρόμησης

Η κλάση LogisticRegression που περιγράφηκε στο Κεφάλαιο 3 υλοποιεί την απαραίτητη δομή και λειτουργικότητα που απαιτείται για παραγωγή ενός ταξινομητή λογιστικής παλινδρόμησης. Η υλοποίησή της αφορά κατά κύριο λόγο την υλοποίηση της διαδικασίας εκπαίδευσης του λογιστικού μοντέλου. Η διαδικασία αυτή συγκεντρώνεται στη μέθοδο `fit()`, η οποία αφού αρχικοποιήσει τις μεταβλητές μέλη της κλάσης, ξεκινάει την προσαρμογή του μοντέλου.

```
#pragma omp parallel for
for (unsigned c = 0; c < _nClasses; ++c) {
    if (_batchSize == 1) stochasticGradientDescent(c);
    else batchGradientDescent(c);
}
```

Η εκπαίδευση ξεκινάει με την κλήση μιας από τις `private` μεθόδους υλοποίησης του αλγορίθμου επικλινούς καθόδου.

Ο βρόχος προσαρμόζει τόσα ανεξάρτητα λογιστικά μοντέλα όσες και οι πιθανές κλάσεις σε μία από τις οποίες μπορεί να ανήκει ένα παράδειγμα. Η ανεξαρτησία των μοντέλων μεταξύ τους και το υψηλό υπολογιστικό κόστος της εκπαίδευσης καθενός από αυτά κάνει το συγκεκριμένο σημείο ιδανικό για την παραλληλοποίηση του αλγορίθμου, που γίνεται με μία μόνο πρόσθετη γραμμή κώδικα, δηλαδή της οδηγίας `pragma`. Σε περίπτωση αρκετών κλάσεων, η χρήση της αρχιτεκτονικής ενός πολυπύρηνου επεξεργαστή θα αποβεί ιδιαίτερα αποδοτική και η εκπαίδευση θα επιταχυνθεί σημαντικά.

Η υλοποίηση του αλγορίθμου της επικλινούς καθόδου αυτού καθαιτού γίνεται στις `private` μεθόδους `stochasticGradientDescent()` και `batchGradientDescent()`. Η πρώτη παραλλαγή χρησιμοποιείται όταν η μεταβλητή `batchSize` είναι ίση με 1, και χρησιμοποιεί σε κάθε επανάληψη ένα μόνο παράδειγμα για τη διόρθωση των παραμέτρων. Η δεύτερη χρησιμοποιείται σε οποιαδήποτε άλλη περίπτωση, και κάνει χρήση ενός mini-batch από τόσα παραδείγματα όσα το `batchSize` για την διόρθωση των παραμέτρων σε κάθε επανάληψή της. Θέτοντας δε τη μεταβλητή σε 0 γίνεται χρήση όλων των παραδειγμάτων σε κάθε επανάληψη,

δηλαδή χρησιμοποιείται το full-batch.

Στην περίπτωση της `batchGradientDescent()` γίνεται χρήση των μεθόδων της βιβλιοθήκης Eigen, η οποία υλοποιεί πράξεις πινάκων, και ιδιαίτερα πολλαπλασιασμούς μεταξύ τους, με πολύ αποδοτικό τρόπο.

```
for (unsigned b = 0; b < g_nBatches; ++b) {

    likelihood = g_in->block(b * _batchSize, 0, _batchSize,
        _nAttributes) * _coeff.row(c).transpose();
    likelihood.array() += _intercept.coeff(c);

    likelihood.array() *= -1;
    likelihood = likelihood.array().exp();
    likelihood.array() += 1.0;
    likelihood = likelihood.array().inverse();

    likelihood -= g_out->block(b * _batchSize, c, _batchSize, 1);

    dw = likelihood.transpose() * g_in->block(b * _batchSize, 0,
        _batchSize, _nAttributes);
    dw.array() /= _batchSize;
    db = likelihood.sum();
    db /= _batchSize;

    _coeff.row(c) -= _learnRate * dw.transpose();
    _intercept.coeffRef(c) -= _learnRate * db;

}
```

Ο βρόχος που εκτελείται σε κάθε επανάληψη της επικλινούς μεθόδου.

Ιδιαίτερης σημασίας είναι η μέθοδος `block()` της κλάσης `MatrixXd` της βιβλιοθήκης Eigen, η οποία λειτουργεί σαν αναφορά σε κομμάτι του πίνακα των εισόδων και δεν προσθέτει κανένα επιπρόσθετο κόστος στην εκτέλεση. Αυτό σημαίνει ότι χωρίς να δημιουργηθούν αντίγραφα των παραδειγμάτων εκπαίδευσης, μπορεί να χρησιμοποιηθεί αποδοτικά μέρος τους στα mini-batches.

Όπως προαναφέρθηκε, εκπαίδεύονται τόσοι ανεξάρτητοι ταξινομητές όσες και οι πιθανές κλάσεις σε κάποια εκ των οποίων μπορεί να ανήκει ένα παράδειγμα. Στη μέθοδο `predict()` υλοποιείται η τεχνική One-vs-Rest. Κατά την ταξινόμηση, αφού υπολογιστούν οι πιθανότητες να ανήκει ένα παράδειγμα σε κάθε κλάση, το παράδειγμα ταξινομείται σε αυτή με τη μέγιστη πιθανότητα.

```

Eigen::VectorXi prediction(nInstances);
double predictedLogLikelihood;

for (unsigned i = 0; i < nInstances; ++i) {
    prediction.coeffRef(i) = 0;
    predictedLogLikelihood = probabilities.coeff(i, 0);
    for (unsigned c = 1; c < _nClasses; ++c) {
        if (probabilities.coeff(i, c) > predictedLogLikelihood) {
            prediction.coeffRef(i) = c;
            predictedLogLikelihood = probabilities.coeff(i, c);
        }
    }
}

```

Η μέθοδος `predict()` υλοποιεί την τεχνική *One-vs-Rest* αφού κάνει υπολογισμό της πιθανότητας να ανήκει ένα παράδειγμα σε κάθε κλάση.

## 4.4 Ταξινόμηση με TNΔ multilayer perceptron

Η κλάση `MLPClassifier` υλοποιεί ένα πλήρως διασυνδεδεμένο TNΔ εμπρόσθιας τροφοδότησης, βασικά χαρακτηριστικά του οποίου είναι οι νευρώνες και οι συνάψεις μεταξύ αυτών. Οι βασικές διεργασίες που υλοποιούνται για την εκπαίδευση ενός TNΔ είναι η εμπρόσθια τροφοδότηση και η οπισθοδρομική διάδοση του σφάλματος.

Οι διεργασίες αυτές, ιδιαίτερα όταν γίνεται χρήση mini-batch, προσφέρονται για διανυσματοποίηση μονής εντολής-πολλαπλών δεδομένων (SIMD), δυνατότητα που προσφέρεται στις μενόδους της βιβλιοθήκης Eigen. Θεωρώντας ένα μοναδικό διάνυσμα εισόδου, με χρήση των δομών και μενόδων Eigen, η τιμή κάθε νευρώνα ενός επιπέδου αναπαρίσταται ως τιμή ενός διανύσματος τόσων πραγματικών αριθμών όσοι και οι νευρώνες του αντίστοιχου επιπέδου. Αντίστοιχα, για mini-batch  $n$  εισόδων, κάθε επίπεδο αποτυπώνεται ως ένας πίνακας  $n$  γραμμών και τόσων στηλών όσοι οι νευρώνες του επιπέδου. Η εμπρόσθια τροφοδότηση συνεπώς υλοποιείται ως ένας πολλαπλασιασμός μεταξύ του πίνακα δεδομένων εισόδου και των παραμέτρων `coeff` του αντίστοιχου επιπέδου και στη συνέχεια ως ένα μη-γραμμικός μετασχηματισμός αυτού του πολλαπλασιασμού.

```

probabilities[0] = _coeff[0] * data->transpose();
probabilities[0].colwise() += _intercept[0];
activationFunction(&probabilities[0], _activation[0]);

for (unsigned l = 1; l < _nLayers; ++l) {
    probabilities[l] = _coeff[l] * probabilities[l-1];
}

```

```

    probabilities[1].colwise() += _intercept[1];
    activationFunction(&probabilities[1], _activation[1]);
}

```

*H εμπρόσθια τροφοδότηση υλοποιείται μέσω ενός βρόχου.*

Το σημείο στο οποίο εφαρμόζεται η συνάρτηση ενεργοποίησης (ή αντίστοιχα η παράγωγος της συνάρτησης ενεργοποίησης στην οπισθοδρομική διάδοση), γίνεται κλήση μιας συνάρτησης η οποία λαμβάνει ως είσοδο το αποτέλεσμα του γραμμικού μετασχηματισμού του διανύσματος εισόδου, επιλέγει μεταξύ των υλοποιημένων συναρτήσεων ενεργοποίησης, και εφαρμόζει κάποιο μη-γραμμικό μετασχηματισμό σε αυτή την είσοδο. Η συγκεκριμένη συνάρτηση, όπως και οι συναρτήσεις ενεργοποίησης αυτές καθαυτές, έχει οριστεί ως *inline* για λόγους απόδοσης.

```

switch (actFunction) {
    case 0:
        activationSigmoid(activated);
        break;
    case 1:
        activationReLU(activated);
        break;
    default:
        activationSigmoid(activated);
        break;
}

```

*H επιλογή μεταξύ των συναρτήσεων ενεργοποίησης γίνεται βάσει της τιμής του *actFunction* που έχει οριστεί για το συγκεκριμένο επίπεδο κατά την αρχικοποίηση..*

Όπως και στην κλάση *LogisticRegression*, το *batchSize* καθορίζει την επιλογή μεταξύ της στοχαστικής ή της mini-batch επικλινούς καθόδου. Ωστόσο, η προσέγγιση που ακολουθείται για την παραλληλοποίηση της κλάσης *MLPClassifier* είναι διαφορετική. Εδώ χρησιμοποιείται η ιδέα της παραλληλισμού δεδομένων, στην οποία εκπαιδεύονται τόσα νευρωνικά δίκτυα όσα και τα νήματα εκτέλεσης.

```

#pragma omp parallel reduction(:g_counter)
{
    if (_batchSize == 1) stochasticGradientDescent();
    else batchGradientDescent();
}

```

*H παραλληλοποίηση της *MLPClassifier* κάνει χρήση της ιδέας του παραλληλισμού δεδομένων.*

Καθένα από αυτά εκπαιδεύεται σε ένα υποσύνολο του συνόλου δεδομένων εκ-

παίδευσης, και έτσι συνυπάρχουν τόσα σύνολα συντελεστών βαρύτητας όσα και τα νήματα. Η βασική αλλαγή στον κώδικα για υποστήριξη της παραλληλοποίησης είναι στον βρόχο επιλογής του παραδείγματος ή του mini-batch παραδειγμάτων που χρησιμοποιούνται σε κάθε βήμα του αλγορίθμου, και συγκεκριμένα στον αριθμό που προσαντέλεται σε κάθε βήμα του βρόχου, που αντί για 1 είναι ίσος με τον αριθμό των νημάτων εκτέλεσης. Έτσι, αν υπάρχουν για παράδειγμα δύο νήματα, το ένα διαχειρίζεται τα άρτια mini-batches και το άλλο τα περιττά.

```

for (unsigned b = th; b < g_nBatches; b += g_nThreads) {

    // Feed-forward
    linear[0] = g_coeff[th][0] * g_in->block(b * _batchSize, 0,
                                                _batchSize, _nAttributes).transpose();
    linear[0].colwise() += g_intercept[th][0];
    activated[0] = linear[0];
    activationFunction(&activated[0], _activation[0]);

    for (unsigned l = 1; l < _nLayers; ++l) {
        linear[l] = g_coeff[th][l] * activated[l-1];
        linear[l].colwise() += g_intercept[th][l];
        activated[l] = linear[l];
        activationFunction(&activated[l], _activation[l]);
    }
    // [...]
}

```

*Η εμπρόσθια τροφοδότηση γίνεται ανεξάρτητα μεταξύ πολλαπλών νημάτων.*

Μετά από έναν ορισμένο αριθμό ανανεώσεων των παραμέτρων που αποθηκεύεται στην `global` και κοινή μεταξύ των νημάτων μεταβλητή `g_counter`, υπολογίζεται ο μέσος όρος των συντελεστών βαρύτητας για κάθε επίπεδο. Αυτοί οι νέοι συντελεστές βαρύτητας που προκύπτουν ορίζονται στη συνέχεια και ως τοπικοί συντελεστές βαρύτητας για κάθε νήμα:

```

if (g_counter > 0 && (g_counter % 100) == 0) {
    #pragma omp barrier
    if (th == 0) {
        if (_verbose)
            std::cout << "Score on test set after " << g_counter << "
                           updates:\t" << score(testD) << std::endl;
        combineWeights();
    }
}

```

```
#pragma omp barrier
for (unsigned l = 0; l < _nLayers - 1; ++l) {
    g_coeff[th][l] = _coeff[l];
    g_intercept[th][l] = _intercept[l];
}
std::cout << g_counter_n[th] << std::endl;
```

Μετά από 100 ανανεώσεις, γίνεται υπολογισμός του μέσου όρου των συντελεστών βαρύτητας κάθε νήματος.

Στο παραπάνω απόσπασμα του κώδικα φαίνεται και η χρήση της δομής συγχρονισμού **barrier** της OpenMP. Με το συγκεκριμένο **pragma** η εκτέλεση των νημάτων σταματάει μόνο όταν έχουν φτάσει όλα στο συγκεκριμένο σημείο κώδικα.

Στα πλαίσια της ταξινόμησης ο τρόπος που κάνει προβλέψεις το μοντέλο δεν διαφέρει σε κάτι από τον τρόπο που κάνουν προβλέψεις άλλοι ταξινομητές.

## 4.5 Ταξινόμηση με απλό ταξινομητή Bayes

Η βασική ιδέα πίσω από τον απλό ταξινομητή Bayes είναι η εφαρμογή του θεωρήματος του Bayes στα πλαίσια της ταξινόμησης. Για την εφαρμογή του θεωρήματος γίνεται χρήση ορισμένων πιθανοτήτων οι οποίες υπολογίζονται με διαφορετικό τρόπο στην περίπτωση του multinomial και του γκαουσιανού ταξινομητή. Γι' αυτό τον τρόπο έχει ληφθεί η απόφαση υλοποίησης της κλάσης NaiveBayes ως ένα **template** κλάσης του οποίου οι **virtual** μέθοδοι θα υλοποιηθούν από τις εφαρμογές του **template**.

Όπως φαίνεται και από το API της κλάσης στο Κεφάλαιο 3, οι περισσότερες μέθοδοι είναι **pure virtual**, δηλαδή στερούνται υλοποίησης στην **NaiveBayes**, και υλοποιούνται στους ορισμούς των επακόλουθων κλάσεων που την κληρονομούν. Ωστόσο, η μέθοδος υπολογισμού της εκ των υστέρων πιθανότητας, βάσει τις οποίας γίνονται και οι προβλέψεις, είναι κοινή μεταξύ των διαφορετικών εκδοχών της **NaiveBayes**.

```
for (unsigned c = 0; c < _nClasses; ++c) {
    for (unsigned a = 0; a < _nAttributes; ++a)
        probabilities.array().row(c) *= findLikelihood(a, data,
                                                       c).transpose().array();
    probabilities.array().row(c) *= _prior.coeff(c);
}
```

Η εκ των υστέρων πιθανότητα υπολογίζεται σε ένα βρόχο.

Η πρόβλεψη μπορεί να γίνει χρησιμοποιώντας είτε την πραγματική εκ των υ-

στέρων πιθανότητα όπως φαίνεται παραπάνω, ή χρησιμοποιώντας τον λογάριθμο της πιθανότητας. Ο δεύτερος τρόπος έχει χαμηλότερο υπολογιστικό κόστος, αλλά παράγει ελαφρώς χειρότερες προβλέψεις.

Η υλοποίηση της `fit(param)` στην περίπτωση του multinomial απλού ταξινομητή Bayes της κλάσης `MultinomialNB` συνεπάγεται προσδιορισμό των τιμών των μεταβλητών `prior`, `evidence` και `likelihood` οι οποίες αντιπροσωπεύουν αντίστοιχα την εκ των προτέρων πιθανότητα εμφάνισης μιας κλάσης, μιας την πιθανότητα εμφάνισης συγκεκριμένης τιμής ενός χαρακτηριστικού, και την πιθανότητα εμφάνισης μιας συγκεκριμένης τιμής ενός χαρακτηριστικού δεδομένης κλάσης.

Ο τρόπος υπολογισμού είναι εξαιρετικά απλός και δεν διαφέρει σε τίποτα από απλή μέτρηση των συχνοτήτων εμφάνισης κάθε ενδεχομένου. Υπάρχει ωστόσο η ιδιαίτερη περίπτωση του προβλήματος μηδενικής συχνότητας, στην οποία για μία κλάση δεν υπάρχει περίπτωση εμφάνισης κάποιας συγκεκριμένης τιμής ενός χαρακτηριστικού. Σε περίπτωση εντοπισμού του προβλήματος μηδενικής συχνότητας αρκεί να προστεθεί ο αριθμός 1 σε κάθε συνδυασμό κλάσης και τιμής χαρακτηριστικού.

```
// zero-frequency problem
bool zfp = false;
for (unsigned a = 0; a < _nAttributes; ++a)
    for (unsigned r = 0; r < data->categories(a); ++r)
        for (unsigned c = 0; c < _nClasses; ++c)
            if (_likelihood[a].coeff(r, c) == 0)
                zfp = true;

if (zfp)
    for (unsigned a = 0; a < _nAttributes; ++a)
        _likelihood[a].array() += 1.0;
```

Αν εντοπιστεί ύπαρξη του zero-frequency problem, αυτό επιλύεται με την προσθήκη του αριθμού 1 στο `likelihood` κάθε χαρακτηριστικού.

Στην περίπτωση του γκαουσιανού απλού ταξινομητή Bayes στην κλάση `GaussianNB`, υπάρχουν συνεχείς τιμές σε κάθε χαρακτηριστικό, και επομένως δεν μπορεί να γίνει μέτρηση των συχνοτήτων πέραν αυτών της κάθε κλάσης. Ο υπολογισμός των πιθανοτήτων γίνεται σε αυτή την περίπτωση με υπολογισμό του μέσου όρου και της τυπικής απόκλισης και χρήσης της συνάρτησης Gauss για παραγωγή πιθανοτήτων σε κάθε περίπτωση.

```
Eigen::VectorXd likelihoods(data->rows());

likelihoods.array() = (data->col(attr).array() - _means.coeff(attr,
```

```

        givenClass)).square();
    likeliabilities.array() /= -2 * pow(_stDev.coeff(attr, givenClass), 2);
    likeliabilities.array() = likeliabilities.array().exp();
    likeliabilities.array() /= sqrt(2 * M_PI * pow(_stDev.coeff(attr,
        givenClass), 2));

    return likeliabilities;
}

```

Παραγωγή του likelihood χρησιμοποιώντας τη συνάρτηση του Gauss μαζί με τους μέσους όρους και τις τυπικές αποκλίσεις.

## 4.6 Ταξινόμηση με δέντρο αποφάσεων

Η υλοποίηση του ταξινομητή δέντρου αποφάσεων γίνεται μέσω των κλάσεων `DecisionTree` και `DTNode`. Το δέντρο αυτό καθαυτό αναπαρίσταται μέσω της πρότερης κλάσης, η οποία περιέχει πολλαπλές περιστάσεις της δεύτερης ως κόμβους.

Η διαδικασία ανάπτυξης του δέντρου ξεκινάει με τη μέθοδο `fit(param)`, η οποία λαμβάνει τα δεδομένα εκπαίδευσης, δημιουργεί τον κόμβο που λειτουργεί ως ρίζα του δέντρου, και εκκινεί τη διαδικασία διχοτόμησής του, που θα έχει τελικά ως αποτέλεσμα την δημιουργία του δέντρου.

```

outcomeFrequencies.setZero();
for (unsigned i = 0; i < nInstances; ++i)
    ++outcomeFrequencies.coeffRef((unsigned)(out->coeff(i, 0)));
outcomeFrequencies.array() /= nInstances;

_root = new DTNode(1, outcomeFrequencies);

std::vector<unsigned> subSet(nInstances);
for (unsigned i = 0; i < data->instances(); ++i)
    subSet[i] = i;

std::vector<unsigned> attributes(_nAttributes);
for (unsigned a = 0; a < _nAttributes; ++a)
    attributes[a] = a;

_root->split(data, subSet, attributes);
}

```

Για τη δημιουργία του κόμβου της ρίζας, γίνεται υπολογισμός των συχνοτήτων εμφάνισης της κάθε έκβασης στο σύνολο των δεδομένων, και τίθεται επιπλέον η εντροπία σε 1.

Η κλήση της μεθόδου `split(param)` ξεκινάει τη διαδικασία βλάστησης του δέντρου. Η διχοτόμηση ενός κόμβου έχει ως πρώτο βήμα τον προσδιορισμό του καλύτε-

ρου χαρακτηριστικού από αυτά που δεν έχουν ήδη χρησιμοποιηθεί από προηγούμενο κόμβο, και ακολουθείται από δημιουργία τόσων κόμβων παιδιών όσες και οι πιθανές τιμές αυτού του χαρακτηριστικού που επιλέχθηκε.

```

    _splitAttribute = findBestSplit(data, subSet, attributes, &newSubSets,
        &newEntropies);

    _nChildren = newSubSets.size();
    _children.resize(_nChildren);
    _splitValues.resize(_nChildren);

```

*H findBestSplit(param)* επιστρέφει το καλύτερο χαρακτηριστικό και επιπλέον υπολογίζει τα νέα υποσύνολα και την εντροπία καθενός από αυτά.

Η *private* μέθοδος *findBestSplit(param)* υλοποιεί το κριτήριο επιλογής του αλγορίθμου ID3. Η διαδικασία επιλογής αποτελείται από υπολογισμό του κέρδους πληροφορίας μετά από πιθανή επιλογή καθενός από τα εναπομέναντα χαρακτηριστικά και η διατήρηση αυτού που παράγει το μεγαλύτερο κέρδος, και συνεπώς ελαχιστοποιεί την εντροπία του κόμβου που θα παραχθεί. Η μέτρηση του κέρδους πληροφορίας γίνεται με την *private* μέθοδο *measureInfoGain(param)*. Η μέθοδος αυτή λαμβάνει ως είσοδο το υποσύνολο δεδομένων που εξετάζεται, το ενδεχόμενο χαρακτηριστικό το οποίο πρόκειται να διχοτομήσει αυτό το σύνολο, και επιστρέφει την τιμή του κέρδους πληροφορίας, τα νέα υποσύνολα που θα παραχθούν, και την τιμή εντροπίας του καθενός.

Όπως και στους περισσότερους ταξινομητές, η διαδικασία ταξινόμησης αποτελείται αρχικά από υπολογισμό της πιθανότητας να έχει ένα παράδειγμα την κάθε πιθανή έκβαση, και επιλογή της έκβασης εκείνης με την μέγιστη πιθανότητα. Ο υπολογισμός των πιθανοτήτων γίνεται μέσω διάβασης του δέντρου βάσει των χαρακτηριστικών εισόδου και επιστροφή των συχνοτήτων έκβασης του κόμβου άφιξης ως πιθανότητες. Η διάβαση ξεκινάει φυσικά από τη ρίζα του δέντρου και συνεχίζει μέχρι να γίνει είτε κατάληξη σε ένα φύλο του δέντρου.

```

Eigen::VectorXd metis::DTNode::traverse(Eigen::VectorXd data) const {
    for (unsigned c = 0; c < _nChildren; ++c)
        if (data.coeff(_splitAttribute) == _splitValues.coeff(c))
            return _children[c]->traverse(data);

    return _outcomeFrequencies;
}

```

*H διάβαση του δέντρου ακολουθεί τη διαδρομή που ορίζεται από το διάνυσμα εισόδου.*

Γίνεται κατανοητό ότι αν συναντηθεί άγνωστη τιμή για κάποιο χαρακτηριστικό, δεν επιλέγεται κανένα παιδί του κόμβου που αντιπροσωπεύει αυτό το χαρακτηριστικό, και γίνεται επιστροφή των τιμών συχνότητας αυτού του κόμβου.

## 4.7 Ομαδοποίηση $k$ -μέσων

Το API της KMeans έχει περιγραφεί στο Κεφάλαιο 3, και εδώ θα γίνει λόγος για ορισμένες από τις λεπτομέρειες που αφορούν την υλοποίησή της. Η βασική μέθοδος της KMeans είναι φυσικά η `cluster(param)`, εντός τις οποίας γίνονται οι επιμέρους διεργασίες ομαδοποίησης.

Αξίζει αρχικά να γίνει αναφορά σε μια σειρά `global` μεταβλητών που ορίζονται μια φορά εντός ενός ανώνυμου `namespace`, και στη συνέχεια χρησιμοποιούνται σε όλη την έκταση του προγράμματος για της ανάγκες προσωρινής αποθήκευσης δεδομένων.

```
namespace {
    Eigen::MatrixXd *g_data;
    unsigned g_nThreads;
    Eigen::VectorXd *g_distances;
    Eigen::MatrixXd *g_newCentroids;
    std::vector<unsigned> *g_newClusterSizes;
}
```

Ορισμένες μεταβλητές ορίζονται ως `global` σε ένα ανώνυμο `namespace` έτσι ώστε να υπάρχει πρόσβαση σε αυτές κατά τη διάρκεια της εκπαίδευσης από όλες τις μεθόδους.

Το μέγεθος αυτών των μεταβλητών είναι μεγάλο, και χρησιμοποιούνται πολλές φορές σε κάθε επανάληψη του αλγορίθμου. Προκειμένου να μην γίνεται δέσμευση και αποδέσμευση μνήμης σε κάθε επανάληψη, κάτι που θα καθυστερούσε την εκπαίδευση, ο ορισμός των μεταβλητών μόνο μία φορά εκ των προτέρων καθίσταται αναγκαίος. Όπως θα φανεί αργότερα δε, στα πλαίσια της παραλληλοποίησης της εκπαίδευσης, κάθε νήμα εκτέλεσης θα διατηρεί το δικό του αντίγραφο αυτών των προσωρινών μεταβλητών.

Αφού οριστούν οι παραπάνω μεταβλητές, η ροή εκτέλεσης της `cluster(param)` προχωρά στην αρχικοποίηση των κεντροειδών βάσει του αλγορίθμου επιλογής που έχει οριστεί. Κάθε ένας από αυτούς τους αλγορίθμους υλοποιείται από μία `private` μέθοδο, όπως η `initializeCentroidsKMeansPP()`. Οι συγκεκριμένες μέθοδοι, πάλι για λόγους απόδοσης, έχουν οριστεί ως `inline`, κάτι που θα επιταχύνει την εκτέλεσή τους μειώνοντας το μέγεθος του call stack.

Φτάνει τέλος το επαναληπτικό μέρος του αλγορίθμου, το οποίο εκτελείται έως ότου τα κέντρα μετά από μία επανάληψη δεν αλλάζουν, ή μέχρι να ολοκληρωθεί ένας μέγιστος αριθμός επαναλήψεων. Τα νέα κεντροειδή αποθηκεύονται στον πρώτο δείκτη της `global` μεταβλητής `g_newCentroids`, και συγχρίνονται με τα κεντροειδή της προηγούμενης επανάληψης.

```

    for (i = 0; i < _maxIterations; ++i) {
        redetermineCentroids();
        if (g_newCentroids[0].isApprox(_centroids)) break;
        else _centroids = g_newCentroids[0];
    }

```

Μετά από τον επαναπροσδιορισμό των κέντρων γίνεται έλεγχος ισότητά τους με τα κέντρα της προηγούμενης επανάληψης.

Το κάθε επαναληπτικό βήμα γίνεται με τη μέθοδο `redetermineCentroids()`, η οποία υλοποιεί το βήμα επαναπροσδιορισμού των ομάδων και των κέντρων τους. Εδώ είναι που χρησιμοποιούνται οι `global` μεταβλητές που ορίστηκαν προηγουμένως. Υπολογίζεται αρχικά το χοντινότερο κέντρο από κάθε παράδειγμα των δεδομένων εισόδου με την `closestCentroid()` και στη συνέχεια βρίσκεται ο μέσος όρος των τιμών κάθε ομάδας και προσδιορίζονται με αυτόν τον τρόπο τα νέα κέντρα.

```

#pragma omp parallel
{
    unsigned th = omp_get_thread_num();

    g_newCentroids[th].setZero();

    g_newClusterSizes[th].clear();
    g_newClusterSizes[th].resize(_nClusters);

    unsigned selectedCluster;

    #pragma omp for
    for (unsigned i = 0; i < _nInstances; ++i) {
        selectedCluster = closestCentroid(g_data->row(i));
        g_newCentroids[th].row(selectedCluster) += g_data->row(i);
        g_newClusterSizes[th][selectedCluster] += 1;
    }
}

```

Κάθε νήμα υπολογίζει την ομάδα στην οποία ανήκει το κάθε σημείο-παράδειγμα, και αποθηκεύει τα αποτελέσματα προσωρινά στο αντίστοιχο αντίγραφο των `global` μεταβλητών.

Αυτό το σημείο έχει και το υψηλότερο υπολογιστικός κόστος, και συνεπώς εδώ επιλέχθηκε να γίνει παραλληλοποίηση της εκτέλεσης. Η προσέγγιση που ακολουθήθηκε είναι ο διαχωρισμός της εργασίας που γίνεται επί του συνόλου δεδομένων μεταξύ πολλαπλών νημάτων, και όπως φαίνεται παραπάνω, χρειάστηκαν ελάχιστες αλλαγές στον κώδικα. Αφού ομαδοποιηθούν όλα τα παραδείγματα, τα αποτελέσματα προστίθενται μεταξύ τους και τα νέα κέντρα των ομάδων υπολογίζονται στο αντίγραφο του πρώτου νήματος.

```

for (unsigned t = 1; t < g_nThreads; ++t) {
    for (unsigned c = 0; c < _nClusters; ++c) {
        g_newCentroids[0].row(c) += g_newCentroids[t].row(c);
        g_newClusterSizes[0][c] += g_newClusterSizes[t][c];
    }
}

for (unsigned c = 0; c < _nClusters; ++c) {
    if (g_newClusterSizes[0][c] == 0) {
        g_newCentroids[0].row(c) = _centroids.row(c);
    } else
        g_newCentroids[0].row(c) /= g_newClusterSizes[0][c];
}

```

Στο σημείο προσθήκης των τοπικών κεντροειδών κάθε νήματος παύει η παράλληλη εκτέλεση των κώδικα.

# Κεφάλαιο 5

## Εφαρμογές και αξιολόγηση

Κατά την ανάπτυξη της βιβλιοθήκης, για να επιβεβαιωθεί η αρτιότητα της λειτουργίας των αλγορίθμων που υλοποιήθηκαν έγινε ανάπτυξη μερικών απλών εφαρμογών. Στο παρόν κεφάλαιο θα παρουσιαστούν αυτές ορισμένες από αυτές της εφαρμογές κάνοντας χρήση σύνολα δεδομένων που συλλέχθηκαν από το διαδίκτυο και στα οποία γίνεται αναφορά στο Παράρτημα Β'. Στόχος της παρουσίασης αυτών των παραδειγμάτων είναι η εξοικείωση του αναγνώστη με το API της βιβλιοθήκη και η αξιολόγηση της λειτουργικότητας των παρεχόμενων μεθόδων.

### 5.1 Χρήση της γραμμικής παλινδρόμησης για προβλέψεις επί της προόδου διαβητικών ασθενών

Η εφαρμογή της μηχανικής μάθησης στην ιατρική γίνεται με πολλούς τρόπους και αυξάνεται με γοργούς ρυθμούς. Το σύνολο δεδομένων Diabetes που αφορά κλινικές μετρήσεις για διαβητικούς ασθενείς περιγράφεται στο παράρτημα. Θα γίνει χρήση της γραμμικής παλινδρόμησης για την εκπαίδευση ενός μοντέλου επάνω σε αυτά τα δεδομένα, με σκοπό τη σύνθεση ενός μοντέλου που να μπορεί να χρησιμοποιηθεί για προβλέψεις σχετικές με την υγεία νέων ασθενών.

```
void demoLinReg() {  
  
    metis::DataLabeled *diabetes = metis::loadDiabetes();  
    diabetes->shuffle();  
    metis::DataLabeled *diabetesTest = diabetes->split(0.2);
```

```

metis::LinearRegression regressor = metis::LinearRegression();
regressor.fit(diabetes);

std::cout << "\nModel has been fitted:" << std::endl;
std::cout << regressor.getCoefficients() << std::endl;
std::cout << std::endl << regressor.getIntercept() << std::endl;

Eigen::MatrixXd prediction(diabetesTest->instances(),
    diabetesTest->outputs());
prediction = regressor.predict(diabetesTest->getInputs()->getData());

std::cout << "\nMean squared error:" << std::endl;
std::cout << regressor.score(diabetesTest) << std::endl;

}

```

Κώδικας της εφαρμογής *demoLinReg()*.

Αρχικά φορτώνεται το σύνολο δεδομένων Diabetes, το οποίο ύστερα από ανακάτεμα διαιρείται σε ένα σύνολο δοκιμής που αποτελείται από το 20% των παραδειγμάτων και ένα σύνολο εκπαίδευσης που αποτελείται από το 80% των παραδειγμάτων. Δημιουργείται στη συνέχεια ένα γραμμικός παλινδρομητής μέσω της κλάσης `LinearRegression` και με χρήση της μεθόδου `fit(param)` λαμβάνει χώρα η προσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης. Η μέθοδος `predict(param)` παράγει προβλέψεις πάνω στο σύνολο δοκιμής οι οποίες μπορούν εύκολα να τυπωθούν αν το επιθυμεί ο χρήστης. Το μέσο τετραγωνικό σφάλμα επιστρέφεται μέσω της μεθόδου `score(param)`.

```

Model has been fitted:
-3.77597 -13.4632
-406.382 -423.084
519.919 513.499
573.323 605.283
-594.773 -682.743
181.598 261.445
-72.5708 -47.5875
305.822 309.899
453.739 482.767
53.0837 63.668

152.568

```

```
162.613
```

```
Mean squared error:
```

```
2581.3
```

```
2643.3
```

Έξοδος της εφαρμογής `demoLinReg()`.

Βάσει του μέσου τετραγωνικού σφάλματος (MSE) μπορεί να γίνει η εκτίμηση ότι η προβλεπόμενη τιμή θα διαφέρει κατά  $\frac{\sqrt{MSE}}{2}$  από την πραγματική κατά μέσο όρο, δηλαδή κατά  $\pm 25.40$  για την πρώτη παράμετρο και  $\pm 25.70$  για τη δεύτερη, κάτι που μπορεί να επαληθευτεί τυπώνοντας τα αποτελέσματα με χρήση της εξόδου της μεθόδου `predict(param)`.

Αξίζει να σημειωθεί ότι η τιμή της απόκλισης μεταξύ αυτής και της πρώτης εξόδου είναι η αναμενόμενη. Η δεύτερη έξοδος είναι μία τεχνητή μεταβλητή και έχει προστεθεί στα πλαίσια της επίδειξης και συγκεκριμένα, είναι κοινή με την πρώτη με επιπλέον επιπρόσθετο έναν τυχαίο αριθμό μεταξύ 0 και 20. Επομένως το γραμμικό μοντέλο που την περιγράφει αναμένεται να είναι παρόμοιο με αυτό της πρώτης, αλλά μετατοπισμένο προς τα πάνω κατά  $\frac{0+20}{2=10}$ , πράγμα που πράγματι συμβαίνει κρίνοντας από το γεγονός ότι οι τιμές της απόκλισης διαφέρουν κατά  $162.613 - 152.568 = 10.045$ .

## 5.2 Ταξινόμηση λουλουδιών του γένους της ίριδος μεταξύ τριών ειδών

Η βοτανική μπορεί να χρησιμοποιήσει τη μηχανική μάθηση για γρήγορη και αξιόπιστη αναγνώριση μεταξύ ειδών λουλουδιών που ενδέχεται να έχουν πολύ μικρές διαφορές μεταξύ τους. Το σύνολο δεδομένων Iris έχει περιγραφεί στο παράτημα και θα χρησιμοποιηθεί για επίδειξη των δυνατοτήτων της βιβλιοθήκης. Θα γίνει σύγκριση μεταξύ του απλού ταξινομητή Bayes και ενός ταξινομητή λογιστικής παλινδρόμησης.

### 5.2.1 Χρήση απλού ταξινομητή Bayes

```
void demoGaussianNB(unsigned nThread) {
    metis::DataLabeled *iris = metis::loadIris();
    iris->shuffle();
    metis::DataLabeled *irisTest = iris->split(0.25);
```

```

    metis::GaussianNB gnb;
    gnb.fit(iris);

    Eigen::VectorXi prediction(irisTest->instances());
    prediction = gnb.predict(irisTest->getInputs()->getData());

    Eigen::MatrixXd predictionProb(iris->instances(), iris->classes(0));
    predictionProb = gnb.findPosterior(irisTest->getInputs()->getData());

    std::cout << "\nAccuracy on test set:" << std::endl;
    std::cout << gnb.score(irisTest) << std::endl;

}

```

Κώδικας της εφαρμογής *demoGaussianNB(param)*.

Αφού φορτωθούν τα δεδομένα, ανακατεύονται και διαιρούνται σε σύνολα εκπαίδευσης και δοκιμής που αποτελούνται από το 75% και 25% των παραδειγμάτων αντίστοιχα. Στην παρούσα περίπτωση ταξινομείται ένα σύνολο δεδομένων με συνεχείς τιμές, οπότε πρέπει να χρησιμοποιηθεί η γκαουσιανή εκδοχή του απλού ταξινομητή Bayes που υλοποιείται μέσω της κλάσης *GaussianNB*. Το μοντέλο προσαρμόζεται με κλήση της μεθόδου *fit(param)* η οποία λαμβάνει τα δεδομένα εισόδου ως παράμετρο. Τέλος γίνεται αξιολόγηση του μοντέλου στα δεδομένα δοκιμής καλώντας την *score(param)*.

### 5.2.2 Χρήση ταξινομητή λογιστικής παλινδρόμησης

```

void demoLogRegIris(unsigned iterations, double learnRate, unsigned
batchSize) {

    metis::DataLabeled *iris = metis::loadIris();
    iris->shuffle();
    metis::DataLabeled *irisTest = iris->split(0.25);

    metis::LogisticRegression logreg(iterations, learnRate, batchSize);
    logreg.fit(iris);

    Eigen::VectorXi prediction(irisTest->instances());
    prediction = logreg.predict(irisTest->getInputs());

    Eigen::MatrixXd predictionProb(irisTest->instances(),
                                  irisTest->classes(0));
    predictionProb = logreg.predictProbabilities(irisTest->getInputs());
}

```

```

    std::cout << "\nCoefficients:" << std::endl;
    std::cout << logreg.getCoefficients() << std::endl;

    std::cout << "\nIntercepts:" << std::endl;
    std::cout << logreg.getIntercepts() << std::endl;

    std::cout << "\nAccuracy on test set:" << std::endl;
    std::cout << logreg.score(irisTest) << std::endl;
}

}

```

*Kώδικας της εφαρμογής `demoLogRegIris(param)`.*

Για τον ταξινομητή λογιστικής παλινδρόμησης χρησιμοποιείται η κλάση `LogisticRegression`. Ορίζεται ένας ταξινομητής λογιστικής παλινδρόμησης και αρχικοποιείται με τις τιμές των επαναλήψεων, το ρυθμό μάθησης και το μέγεθος του mini-batch. Στη συνέχεια γίνεται προσαρμογή του μοντέλου με χρήση της μεθόδου `fit(param)` που λαμβάνει ως είσοδο το σύνολο δεδομένων εκπαίδευσης. Υπάρχει δυνατότητα τύπωσης των παραμέτρων που προσδιορίζονται μέσω των `getCoefficients()` και `getIntercepts()`. Η αξιολόγηση γίνεται με χρήση της μεθόδου `score()`.

### 5.2.3 Σύγκριση αποτελεσμάτων

Πίνακας 5.1: Απόδοση και χρόνοι εκτέλεσης ταξινόμησης Iris

Μέθοδος	Παράμετροι	Ακρίβεια	Χρόνος
LogisticRegression	100, 0.01, 0	24.3%	0.001 sec
LogisticRegression	1000, 0.01, 0	81.1%	0.005 sec
LogisticRegression	10000, 0.01, 0	81.1%	0.006 sec
LogisticRegression	10, 0.1, 1	89.2%	0.0007 sec
LogisticRegression	100, 0.1, 1	91.9%	0.002 sec
LogisticRegression	100000, 0.01, 1	91.9%	1.81 sec
LogisticRegression	100000, 0.01, 40	91.9%	0.47 sec
LogisticRegression	100000, 0.01, 0	94.6%	0.40 sec
GaussianNB	-	94.6%	0.00002sec

Από τα αποτελέσματα που φαίνονται στον πίνακα, το πρώτο πράγμα που γίνεται προφανές είναι ότι για απλά προβλήματα όπως αυτό της ταξινόμηση τους Iris, στα οποία η υπόθεση της ανεξαρτησίας ισχύει σε σημαντικό βαθμό, ο απλός ταξινομητής Bayes παράγει πολύ καλά αποτελέσματα πολύ πιο γρήγορα από άλλους ταξινομητές όπως αυτός της λογιστικής παλινδρόμησης.

Για τον ταξινομητή γραμμικής παλινδρόμησης, μία γρήγορη προσέγγιση μπορεί να επιτευχθεί χρησιμοποιώντας τη στοχαστική επικλινή κάθισδο (δηλαδή μέγεθος mini-batch ίσο με το 1) για λίγες επαναλήψεις. Ωστόσο, όπως είναι αναμενόμενο, η αύξηση του μεγέθους του mini-batch μπορεί να μειώσει σημαντικά το χρόνο εκτέλεσης, κάνοντας επιπλέον δυνατή την απόκτηση ενός πιο ακριβούς αποτελέσματος στα πλαίσια χρήσης σταθερού ρυθμού μάθησης λόγω της πιο «λείας» σύγκλισης που επιτυγχάνεται χάρη στην επεξεργασία πολλών παραδειγμάτων ταυτόχρονα που λύνει το πρόβλημα του υπερακοντισμού.

## 5.3 Ταξινόμηση χειρόγραφων αριθμητικών χαρακτήρων του συνόλου MNIST

Η αναγνώριση εικόνας είναι ένα από τα βασικά πεδία εφαρμογής της μηχανικής μάθησης, και η αναγνώριση χειρόγραφων χαρακτήρων μία από τις πιο απλές περιπτώσεις της. Το σύνολο δεδομένων MNIST έχει περιγραφεί στο παράρτημα και θα χρησιμοποιηθεί στα πλαίσια επίδειξης της βιβλιοθήκης metis. Θα αναδειχθεί η απλότητα χρήσης των κλάσεων της βιβλιοθήκης και θα συγχριθεί ο βαθμός απόδοσης ενός ταξινομητή λογιστικής παλινδρόμησης και ενός νευρωνικού δικτύου εμπρόσθιας τροφοδότησης του σφάλματος χρησιμοποιώντας έναν αριθμό διαφορετικών παραμέτρων.

### 5.3.1 Χρήση ταξινομητή λογιστικής παλινδρόμησης

```
void demoLogRegMNIST(unsigned iterations, double learnRate, unsigned
batchSize, unsigned nThreads) {

    omp_set_num_threads(nThreads);

    metis::DataLabeled *mnist = metis::loadMNIST();
    metis::DataLabeled *mnistTest = metis::loadMNIST(true);

    metis::LogisticRegression logreg(iterations, learnRate, batchSize);
    logreg.fit(mnist, mnistTest, false);

    Eigen::VectorXd prediction(mnistTest->instances());
    prediction = logreg.predict(mnist->getInputs());

    Eigen::MatrixXd predictionProb(mnistTest->instances(),
        mnist->classes(0));
```

```

predictionProb = logreg.predictProbabilities(mnistTest->getInputs());

std::cout << "\nCoefficients:" << std::endl;
std::cout << logreg.getCoefficients() << std::endl;

std::cout << "\nIntercepts:" << std::endl;
std::cout << logreg.getIntercepts() << std::endl;

std::cout << "\nAccuracy on test set:" << std::endl;
std::cout << logreg.score(mnistTest) << std::endl;

}

```

*Κώδικας της εφαρμογής `demoLogRegMNIST(param)`.*

Για τον ταξινομητή λογιστικής παλινδρόμησης χρησιμοποιείται η κλάση `LogisticRegression`. Η αρχικοποίηση της κλάσης γίνεται ορίζοντας τον αριθμό των επαναλήψεων που θα γίνουν, το ρυθμό μάθησης και το μέγευθος του mini-batch. Στη συνέχεια καλείται η μέθοδος `fit(param)` με όρισμα οπωσδήποτε ένα σύνολο εκπαίδευσης και προαιρετικά ένα σύνολο δοκιμής. Τρίτο όρισμα είναι η παράμετρος `verbose`, που αν τεθεί `true` θα υπολογίζεται και θα τυπώνεται ο βαθμός απόδοσης του ταξινομητή στο σύνολο δοκιμής μετά από κάποιον αριθμό επαναλήψεων. Τέλος, η παρούσα εφαρμογή θα εκμεταλλευτεί τη δυνατότητα εκτέλεσης σε πολλαπλά νήματα.

Πίνακας 5.2: Απόδοση και χρόνοι εκτέλεσης ταξινόμησης MNIST με λογιστική παλινδρόμηση

Μέθοδος	Παράμετροι	Ακρίβεια	Χρόνος
LogisticRegression	1, 0.01, 60, 1	64.2%	0.801714 sec
LogisticRegression	5, 0.01, 60, 1	79.7%	3.82615 sec
LogisticRegression	5, 0.01, 60, 2	79.7%	2.30213 sec
LogisticRegression	1, 0.1, 60, 1	84.0%	0.802644 sec
LogisticRegression	1, 0.1, 1, 1	84.6%	3.02959 sec
LogisticRegression	1, 0.01, 1, 1	86.6%	3.08755 sec
LogisticRegression	5, 0.1, 1, 1	87.2%	15.0887 sec
LogisticRegression	5, 0.1, 1, 2	87.2%	15.0253 sec
LogisticRegression	5, 0.01, 1, 1	87.7%	15.3322 sec
LogisticRegression	5, 0.01, 1, 2	87.7%	14.6846 sec
LogisticRegression	5, 0.1, 60, 1	89.6%	3.84412 sec
LogisticRegression	5, 0.1, 60, 2	89.6%	2.00663 sec

Λόγω της πολυπλοκότητας του προβλήματος αναγνώρισης χαρακτήρων, η λογιστική παλινδρόμηση δυσκολεύεται να παραγάγει ακρίβεια πολύ μεγαλύτερη από 89%.

Όπως φαίνεται και από τα αποτελέσματα, η αύξηση των επαναλήψεων δεν ωφελεί ιδιαίτερα στην αύξηση της ακρίβειας όταν η ακρίβεια φτάσει πάνω από ένα επίπεδο.

Μια ιδιαίτερα αξιοσημείωτη παρατήρηση αφορά τη σχέση των χρόνων εκτέλεσης με το μέγεθος του mini-batch. Κάνοντας επεξεργασία 60 παραδειγμάτων σε κάθε επανάληψη, ο χρόνος εκτέλεσης μειώνεται δραματικά σχεδόν πέντε φορές. Αυτό είναι αποτέλεσμα της αρχιτεκτονικής μονής εντολής-πολλαπλών δεδομένων (SIMD) την οποία εκμεταλλεύεται η βιβλιοθήκη Eigen στις υλοποιήσεις της. Παρατηρείται δε σχεδόν υποδιπλασιασμός (52.2% του αρχικού) του χρόνου εκτέλεσης στην περίπτωση εκτέλεσης για πέντε επαναλήψεις με μέγεθος μινι-βατζη 60 και παράλληλη εκτέλεση σε 2 νήματα.

### 5.3.2 Χρήση TNΔ εμπρόσθιας τροφοδότησης

```

void demoMLPMNIST(std::vector<unsigned> hiddenLayers,
                    std::vector<unsigned> activationFunction,
                    unsigned batchSize, double learnRate, unsigned iterations,
                    unsigned nThreads) {

    omp_set_num_threads(nThreads);

    metis::DataLabeled *mnist = metis::loadMNIST();
    metis::DataLabeled *mnistTest = metis::loadMNIST(true);

    metis::MLPClassifier clf(hiddenLayers, activationFunction, batchSize,
                             learnRate, iterations);
    clf.train(mnist, mnistTest, true);

    Eigen::VectorXd predictions(mnistTest->instances());
    predictions = clf.predict(mnistTest->getInputs()).cast<double>();

    Eigen::MatrixXd predictionProb(mnistTest->instances(),
                                    mnist->classes(0));
    predictionProb = clf.predictProbabilities(mnistTest->getInputs());

    std::cout << "\nAccuracy on test set:" << std::endl;
    std::cout << clf.score(mnistTest) << std::endl;
}

```

Κώδικας της εφαρμογής *demoMLPMNIST(param)*.

Για τη δημιουργία ενός TNΔ τύπου multilayer perceptron χρησιμοποιείται η

αλάση `MLPClassifier`. Η αρχικοποίηση δέχεται ως πρώτο όρισμα ένα διάνυσμα ακεραίων αριθμών με τόσες τιμές όσοι και ο αριθμός των χρυφών επιπέδων, με την κάθε τιμή να αντιπροσωπεύει τον αριθμό των νευρώνων σε αυτό το επίπεδο. Δεύτερο όρισμα είναι ένα διάνυσμα οι τιμές του οποίου αντιπροσωπεύουν την συνάρτηση ενεργοποίησης από το παρόν επίπεδο προς το επόμενο, οπότε προφανώς θα έχει ίδιο αριθμό τιμών με το πρώτο όρισμα προσαυξημένο κατά 1. Όπως έχει αναφερθεί και στο Κεφάλαιο 3, με το 0 επιλέγεται η σιγμοειδής συνάρτηση, ενώ με το 1 η ReLU. Τέλος, ορίζονται πάλι το μέγεθος του mini-batch, ο ρυθμός μάθησης και ο αριθμός επαναλήψεων που θα γίνει. Η εφαρμογή αυτή θα χρησιμοποιήσει επίσης πολλαπλό αριθμό νημάτων.

**Πίνακας 5.3:** Απόδοση και χρόνοι εκτέλεσης ταξινόμησης MNIST

Μέθοδος	Παράμετροι	Ακρίβεια	Χρόνος
MLPClassifier	{100}, {0,0}, 1, 0.01, 60, 1	24.1%	1.16917 sec
MLPClassifier	{1000}, {0,0}, 1, 0.01, 60, 1	40.6%	15.671 sec
MLPClassifier	{100}, {0,0}, 1, 0.01, 10, 1	55.9%	2.57799 sec
MLPClassifier	{1000,400}, {0,0,0}, 1, 0.01, 60, 1	56.5%	22.946 sec
MLPClassifier	{1000}, {0,0}, 5, 0.01, 60, 1	64.1%	69.9861 sec
MLPClassifier	{1000}, {0,0}, 5, 0.01, 60, 2	70.0%	46.0529 sec
MLPClassifier	{1000}, {0,0}, 1, 0.01, 10, 1	72.3%	49.7746 sec
MLPClassifier	{1000,400}, {0,0,0}, 5, 0.01, 60, 1	78.3%	118.157 sec
MLPClassifier	{1000,400}, {0,0,0}, 5, 0.01, 60, 1	78.9%	117.618 sec
MLPClassifier	{100}, {1,0}, 1, 0.01, 60, 1	79.2%	1.17063 sec
MLPClassifier	{1000,400}, {0,0,0}, 1, 0.01, 10, 1	80.7%	79.9612 sec
MLPClassifier	{1000}, {1,0}, 1, 0.01, 60, 1	88.2%	14.6084 sec
MLPClassifier	{100}, {1,0}, 1, 0.01, 10, 1	90.6%	3.45157 sec
MLPClassifier	{1000,400}, {1,1,0}, 1, 0.01, 10, 1	91.4%	83.9604 sec
MLPClassifier	{1000,400}, {1,1,0}, 1, 0.01, 60, 1	91.5%	28.1584 sec
MLPClassifier	{1000,400}, {1,1,0}, 15, 0.001, 120, 1	91.8%	287.183 sec
MLPClassifier	{1000,400}, {1,1,0}, 15, 0.001, 120, 2	92.5%	156.958 sec
MLPClassifier	{1000}, {1,0}, 5, 0.01, 60, 2	92.6%	48.3774 sec
MLPClassifier	{1000}, {1,0}, 5, 0.01, 60, 1	93.4%	70.2143 sec
MLPClassifier	{1000}, {1,0}, 1, 0.01, 10, 1	93.6%	49.2433 sec
MLPClassifier	{1000,400}, {1,1,0}, 5, 0.01, 60, 2	94.5%	74.7876 sec
MLPClassifier	{1000,400}, {1,1,0}, 5, 0.01, 60, 1	94.6%	139.797 sec
MLPClassifier	{1000,400}, {1,1,0}, 10, 0.01, 60, 1	94.8%	237.947 sec
MLPClassifier	{1000,400}, {1,1,0}, 10, 0.01, 60, 2	95.1%	149.212 sec

Κοιτώντας τα αποτελέσματα, πρώτο πράγμα που μπορεί να παρατηρήσει κανείς είναι ότι η χρήση της συνάρτησης ενεργοποίησης ReLU παράγει δραματικά καλύτερα αποτελέσματα για κοινές κατά τα άλλα παραμέτρους. Η χρήση λίγων νευρώνων στο χρυφό επίπεδο μπορεί να αποβεί αποδοτικό ως μία πρώτη προσέγγιση, παράγοντας ακρίβεια 90.6% με μία μόλις επανάληψη και μέγεθος mini-batch 60 με 100 μόνο χρυφούς νευρώνες. Εντούτοις, η αύξηση του αριθμού των νευρώνων στα χρυφά επίπεδα έχει

σημαντική επίδραση στην ακρίβεια του παραγόμενου νευρωνικού δικτύου, με τον αριθμό 1000 να είναι μια καλή επιλογή δεδομένων των 784 νευρώνων εισόδου. Ένα TNΔ με ένα μόνο κρυφό επίπεδο κατάφερε να φτάσει ακρίβεια 93.6% στις δοκιμές που έγιναν.

Η χρήση βαθιού νευρωνικού δικτύου μπορεί βέβαια να αυξήσει περαιτέρω την ακρίβεια. Δημιουργώντας ένα δεύτερο κρυφό επίπεδο 400 νευρώνων, η ακρίβεια έφτασε στο 95.1% με τις παραμέτρους που χρησιμοποιήθηκαν στο παρόν κεφάλαιο, ενώ κατάφερε να φτάσει μέχρι και 97% σε διαφορετικές συνθήκες. Παρουσιάστηκε δε σημαντική μείωση στο χρόνο εκτέλεσης μέσω παράλληλης εκτέλεσης μεταξύ δύο νημάτων, πηγαίνοντας από 237.947 δευτερόλεπτα σε 149.212 (62.9% του αρχικού χρόνου) για τις κοινές κατά τα άλλα παραμέτρους.

## 5.4 Ταξινόμηση της κατάστασης της ασθένειας καρκινοπαθών ασθενών

Η μηχανική μάθηση χρησιμοποιείται κατά κόρον στον τομέα της ογκολογίας, τόσο για προβλέψεις σχετικές με τον εντοπισμό και την εξέλιξη της ασθένειας, όσο και για την ανάπτυξη τρόπων αντιμετώπισης. Το σύνολο Breast Cancer περιέχει δεδομένα σχετικά με τα χαρακτηριστικά ενός αριθμού ασθενών και της ασθένειας τους, και δεδομένων αυτών των χαρακτηριστικών αν υπήρξε επανεμφάνιση της ασθένειας. Θεωρώντας αυτή την έξοδο ως κλάση κάθισε παραδείγματος, ώστα επιχειρηθεί να γίνει ταξινόμηση των δεδομένων με απλό ταξινομητή Bayes και με δέντρο απόφασης.

### 5.4.1 Χρήση του απλού ταξινομητή Bayes

```

void demoMultinomialNB() {

    metis::DataLabeled *bc = metis::loadBC();
    bc->shuffle();
    metis::DataLabeled *bcTest = bc->split(0.1);

    metis::MultinomialNB mnb;
    mnb.fit(bc);

    Eigen::VectorXd prediction(bcTest->instances());
    prediction = mnb.predict(bcTest->getInputs()->getData());

    Eigen::MatrixXd predictionProb(bcTest->instances(),
        bcTest->classes(0));
}

```

```
    predictionProb = mnb.findPosterior(bcTest->getInputs()->getData());  
  
    std::cout << "\nAccuracy on test set:" << std::endl;  
    std::cout << mnb.score(bcTest) << std::endl;  
  
}
```

*Kώδικας της εφαρμογής `demoMultinomialNB()`.*

Γίνεται αρχικά φόρτωση των δεδομένων, τα οποία αφού ανακατευτούν, διαιρούνται σε σύνολο εκπαίδευσης και σύνολο δοκιμής. Θα χρησιμοποιηθεί ένα μικρό σύνολο δοκιμής, καθώς τα παραδείγματα που διατίθενται δεν είναι αρκετά. Για χρήση απλού ταξινομητή Bayes θα δημιουργηθεί μια περίσταση της κλάσης `MultinomialNB`, δεδομένου ότι τα δεδομένα είναι ονομαστικά και όχι αριθμητικά. Η προσαρμογή του μοντέλου γίνεται με τη μέθοδο `fit(param)`, και η αξιολόγηση με τη `score(param)`.

#### 5.4.2 Χρήση ταξινομητή δέντρου αποφάσεων

```
void demoDTTree() {  
  
    metis::DataLabeled *bc = metis::loadBC();  
    bc->shuffle();  
    metis::DataLabeled *bcTest = bc->split(0.1);  
  
    metis::DecisionTree tree;  
    tree.fit(bc);  
  
    std::cout << "\nTree:" << std::endl;  
    tree.print();  
  
    std::cout << "\nPredictions:" << std::endl;  
    std::cout << tree.predict(bcTest->getInputs()) << std::endl;  
  
    std::cout << "\nAccuracy on test set:" << std::endl;  
    std::cout << tree.score(bcTest) << std::endl;  
  
}
```

*Kώδικας της εφαρμογής `demoDTtree()`.*

Για τη δημιουργία ενός ταξινομητή δέντρου αποφάσεων γίνεται χρήση της κλάσης `DecisionTree`. Η δημιουργία του δέντρου ξεκινάει με χλήση της μεθόδου `fit(param)`, δίνοντάς της τα δεδομένα εισόδου ως παράμετρο. Μπορεί προαιρετικά να τυπωθεί το δέντρο σε μορφή κειμένου με την `print()`, ενώ οι προβλέψεις γίνονται

πάλι με την `predict(param)` και η αξιολόγηση με την `score(param)`.

### 5.4.3 Σύγκριση των αποτελεσμάτων

Πίνακας 5.4: Απόδοση και χρόνοι εκτέλεσης ταξινόμησης Breast Cancer

Μέθοδος	Σύνολο εκπαίδευσης	Ακρίβεια	Χρόνος
MultinomialNB	90%	71.4%	0.00005 sec
MultinomialNB	100%	76.9%	0.00005 sec
DecisionTree	90%	71.4%	0.001 sec
DecisionTree	100%	97.5%	0.00005 sec

Συγχρίνοντας τα αποτελέσματα, παρατηρείται χαμηλή σχετικά απόδοση όχι πάνω από 70% και για τους δύο ταξινομητές. Αυτό ενδέχεται να είναι αποτέλεσμα του γεγονότος ότι τα δεδομένα που παρέχονται δεν είναι αρκετά για τη δημιουργία ενός γενικού μοντέλου. Αυξάνοντας ωστόσο το σύνολο εκπαίδευσης στο 100% των παραδειγμάτων και χρησιμοποιώντας το ίδιο σύνολο για δοκιμή, μπορεί να προκύψει μία ακόμη ενδιαφέρουσα παρατήρηση. Ενώ ο ταξινομητής δέντρου αποφάσεων αυξάνει την απόδοσή του σε 97.5%, πράγμα απόλυτα αναμενόμενο δεδομένης της χρήσης του συνόλου εκπαίδευσης δέντρου αποφάσεων και για δοκιμή, η απόδοση του απλού ταξινομητή Bayes αυξάνεται ελάχιστα. Το συμπέρασμα που προκύπτει από αυτό είναι ότι η ανεξαρτησία μεταξύ των μεταβλητών εισόδου που προϋποθέτει το θεώρημα Bayes μάλλον δεν ισχύει σε σημαντικό βαθμό, και συνεπώς η εκλογή του απλού ταξινομητή Bayes για το συγκεκριμένο σύνολο δεδομένων δεν είναι ιδανική.

## 5.5 Ομαδοποίηση του συνόλου δεδομένων Iris με ομαδοποιητή $k$ -μέσων

Δεν είναι απίθανο να υπάρχει διαθέσιμο ένα σύνολο δεδομένων με μετρήσιες κάποιου φαινομένου, για παράδειγμα των χαρακτηριστικών ορισμένων λουλουδιών, αλλά καμία αναφορά στην κλάση που ανήκει καθένα τους. Στην παρούσα εφαρμογή θα γίνει χρήση του γνωστού συνόλου δεδομένων Iris χωρίς τη μεταβλητή εξόδου που αναφέρει το είδος του κάθε λουλουδιού και θα γίνει ομαδοποίησή τους με τον ομαδοποιητή  $k$ -μέσων.

```
void demoKMeansIris(unsigned nClusters, unsigned initMethod) {
    metis::DataLabeled *iris = metis::loadIris();
    metis::KMeans clusterer(nClusters, initMethod);
```

```

clusterer.cluster(iris->getInputs());

std::cout << "\nInput has been clustered with centroids:" << std::endl;
std::cout << clusterer.getCentroids() << std::endl;

std::cout << "\nMean distance:" << std::endl;
std::cout << clusterer.score(iris->getInputs()) << std::endl;

std::cout << "\nPredictions:" << std::endl;
std::cout << clusterer.getLabels().transpose() << std::endl;

}

```

*Κώδικας της εφαρμογής `demoKMeansIris(param)`*

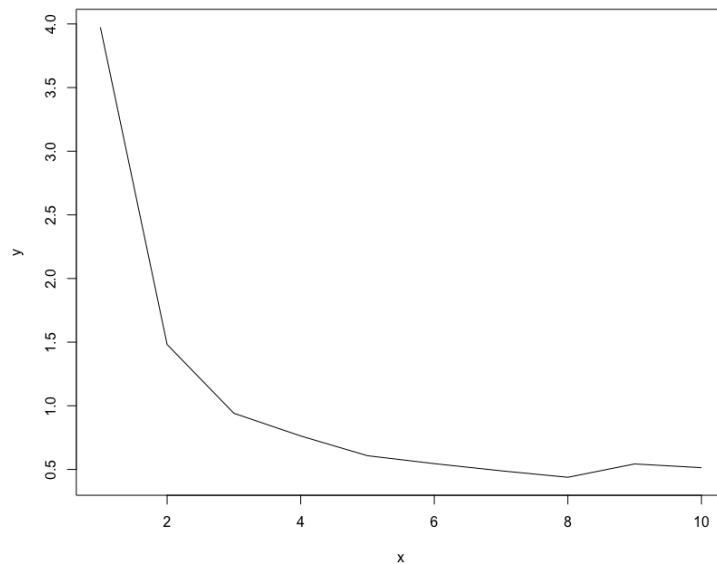
Γίνεται αρχικά φόρτωση των δεδομένων, και στη συνέχεια δημιουργείται ένας ομαδοποιητής  $k$ -μέσων. Δέχεται ως ορίσματα τον αριθμό των ομάδων που θα δημιουργηθούν και την μέθοδο με την οποία θα γίνει αρχικοποίηση. Κατόπιν της ομαδοποίησης υπάρχει δυνατότητα τύπωσης των κέντρων των ομάδων με την `getCentroids()` ενώ μπορεί να γίνει αξιολόγηση με την `score(param)` που επιστρέφει τη μέση απόσταση των σημείων από το κέντρο τους.

Γίνεται αρχικά δοκιμή ομαδοποίησης για όλες τις τιμές από 1 μέχρι 10, και καταγράφεται η μέση απόσταση των σημείων από το κέντρο τους.

Πίνακας 5.5: Αριθμός ομάδων για τα δεδομένα του Iris και μέση απόσταση για κάθε μία από αυτές.

Μέθοδος	Ομάδες	Μέση απόσταση
KMeans	1	3.97333 sec
KMeans	2	1.4816 sec
KMeans	3	0.941088 sec
KMeans	4	0.762816 sec
KMeans	5	0.607761 sec
KMeans	6	0.545443 sec
KMeans	7	0.489144 sec
KMeans	8	0.438299 sec
KMeans	9	0.543017 sec
KMeans	10	0.513779 sec

Σχεδιάζοντας τη σχέση μεταξύ των ομάδων και της μέση απόστασης σύμφωνα με τη μέθοδο του αγκώνα γίνεται εμφανές ότι ο σωστός αριθμός ομάδων βρίσκεται μάλλον κοντά στο 3, αφού η μείωση μετά από αυτό το σημείο είναι λιγότερο απότομη. Είναι βέβαια γνωστό ότι ο σωστός αριθμός ομάδων είναι τρεις, αφού τα είδη των λουλουδιών που υπάρχουν στα δεδομένα είναι τρία.



Σχήμα 5.1: Γραφική αναπαράσταση της σχέσης μεταξύ αριθμού ομάδων και μέσης απόστασης

Τυπώνοντας τις προβλέψεις του ομαδοποιητή για τρεις ομάδες και γνωρίζοντας ότι τα τρία είδη είναι σε σειρά, μπορεί εύκολα να παρατηρηθεί ότι τα περισσότερα παραδείγματα έχουν ταξινομηθεί σωστά, με εξαίρεση μια μικρή σύγχυση που υπάρχει μεταξύ του δεύτερου και του τρίτου είδους.

```

Convergence reached on iteration:
6

Input has been clustered with centroids:
 1.03015 0.0138378 0.940544 0.969016
 -0.167843 -0.966842 0.258754 0.17551
 -0.998721 0.892116 -1.29862 -1.25244

Mean distance:
0.941088

Predictions:
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 1 2 2 2 2 2 2 2 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1
 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1
 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 1 0 0 0 1 0 0 0

```

Απόσπασμα εξόδου της εφαρμογής `demoKMeansIris(param)` για  $k = 3$  και αρχικοποίηση με `kmeans++`.

## 5.5. ΟΜΑΔΟΠΟΙΗΣΗ ΤΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ IRIS ME ΟΜΑΔΟΠΟΙΗΤΗ K-ΜΕΣΩΝ

---

Στα παραπάνω παραδείγματα χρησιμοποιούταν ο αλγόριθμος kmeans++ για την αρχικοποίηση των κεντροειδών, και τα αποτελέσματα είναι γενικά συνεπή ως προς τα κέντρα που παράγονται και τη μέση απόσταση, όσες φορές και αν επαναληφθεί η διαδικασία. Αρχικοποιώντας τα τυχαία θέτοντας `initMethod = 0`, μπορεί να διαπιστωθεί ότι η ποιότητα των αποτελεσμάτων θα είναι λιγότερο αξιόπιστη, με τα κέντρα (και τις επαναλήψεις που απαιτούνται για σύγκλιση) να αλλάζουν από εκτέλεση προς εκτέλεση λόγω ενδεχόμενων κακών αρχικοποιήσεων που μπορεί να προκύψουν ορισμένες φορές.



# Κεφάλαιο 6

## Επίλογος

Μετά το πέρας της παρούσας εργασίας προέκυψε μια λειτουργική βιβλιοθήκη μηχανικής μάθησης που θα μπορούσε να χρησιμοποιηθεί για την ανάπτυξη πραγματικών εφαρμογών. Η ολοκλήρωση μιας μεγάλης προγραμματιστικής εργασίας που αποτελούνταν από μελέτη της βιβλιογραφίας, υλοποίηση περίπλοκων αλγορίθμων, και συνεχή αξιολόγηση των αποτελεσμάτων ήταν μια ιδιαίτερα πολύτιμη εμπειρία.

Από τη δουλειά που έγινε ενδυναμώθηκε τόσο η κατανόηση των τρόπων λειτουργίας βασικών τεχνικών μηχανικής μάθησης, όσο και οι τεχνικές δεξιότητες που αφορούν τον προγραμματισμό που στοχεύει στην εκμετάλλευση αρχιτεκτονικών παραλληληγενών επεξεργασίας. Επιπλέον, βιώθηκε από πρώτο χέρι η διαδικασία ανάπτυξης και συντήρησης μιας βιβλιοθήκης λογισμικού. Η όλη διαδικασία επικύρωσε και επαύξησε το ενδιαφέρον του γράφοντος για τη μηχανική μάθηση και την επιστήμη των δεδομένων.

### 6.1 Συνέχιση ανάπτυξης της βιβλιοθήκης

Η τωρινή μορφή της βιβλιοθήκης «metis» αποτελεί μια καλή βάση για μία εν δυνάμει ισχυρή γενικού σκοπού βιβλιοθήκη μηχανικής μάθησης. Η γνώση και η εμπειρία που αποκτήθηκαν από την ανάπτυξη της μπορούν να χρησιμοποιηθούν για επέκταση των δυνατοτήτων της με περισσότερες και πιο αποδοτικές μεθόδους. Μεταξύ των βελτιώσεων που στοχεύεται να γίνουν είναι:

- υλοποίηση τεχνικών όπως οι μηχανές διανυσμάτων υποστήριξης, η μέθοδος *k*-κοντινότερων γειτόνων και η ανάλυση κύριων συνιστώσων,
- παροχή περισσότερων τεχνικών προεπεξεργασίας των δεδομένων,
- αύξηση των αφαιρέσεων υλοποίησης με σκοπό την παραγωγή συνεπέστερου και πιο εύκολα επεκτάσιμου κώδικα,
- προσθήκη πιο εύχρηστων και άμεσων μεθόδων απεικόνισης.

Ο πηγαίος κώδικας της βιβλιοθήκης διατίθεται στο GitHub. Εκεί είναι όπου θα συνεχιστεί και η ανάπτυξή της ως λογισμικό ανοιχτού κώδικα.

<https://github.com/andbamp/metis>

## 6.2 Συμπεράσματα

Μια πλειάδα βιβλιοθηκών λογισμικού έχει κάνει την εμφάνισή του στη βιομηχανία και προσφέρει ένα μεγάλο αριθμό τεχνικών ανάλυσης δεδομένων και μηχανικής μάθησης. Για κάποιον που θέλει να ασχοληθεί σε βάθος με αυτά τα πεδία, η υλοποίηση των διάφορων δομών και αλγορίθμων που συνιστούν αυτές τις τεχνικές μπορεί να συνεισφέρει σημαντικά στην κατανόησή τους, στην απόκτηση της εμπειρίας που απαιτείται για να γίνει σωστή χρήση τους, και στην προσπάθεια βελτίωσή τους.

Από ιατρική διάγνωση μέχρι ταυτοποίηση προσώπου, και από χρηματοοικονομικές αναλύσεις μέχρι αναγνώριση σμηνών γαλαξιών, η μηχανική μάθηση χρησιμοποιείται για την ανάπτυξη κάθε λογής εφαρμογής. Σύντομα όλα τα επιστημονικά πεδία θα χρησιμοποιούν μεθόδους της με τον έναν τρόπο ή τον άλλο, και δεν χρειάζεται να είναι κανείς μηχανικός για να επιδιώξει την εξοικείωση με αυτή. Παράλληλα, η αυξανόμενη υπολογιστική ισχύ των σύγχρονων υπολογιστών επιτρέπει την εκπαίδευση προβλεπτικών μοντέλων πιο γρήγορα και αποδοτικά από ποτέ.

Η πρόοδος της τεχνολογίας μας έχει φέρει όλο και πιο κοντά στην πραγματοποίηση του ονείρου μηχανοποίησης της ανθρώπινης σκέψης. Μέσω της προσεκτικής μελέτης και εφαρμογής των τεχνολογιών που αναπτύσσονται στα πλαίσια επιδίωξης αυτού του στόχου, η μηχανική μάθηση δύναται να εξυπηρετήσει την ανθρωπότητα, κλείνοντας το χάσμα μεταξύ επιστημονικής φαντασίας και πραγματικότητας.

# Βιβλιογραφία

- [1] Ευάγγελος Δερματάς *Αναγνώριση Προτύπων*. (*Πανεπιστημιακές σημειώσεις*). Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, 1997.
- [2] Trevor Hastie, Robert Tibshirani, Jerome Friedman *The Elements of Statistical Learning*. [*Τα Στοιχεία της Στατιστικής Μάθησης*]. Εκδόσεις Springer (2η Έκδοση), 2016.
- [3] Christopher M. Bishop *Pattern Recognition and Machine Learning*. [*Αναγνώριση Προτύπων και Μηχανική Μάθηση*]. Εκδόσεις Springer, 2011.
- [4] J. R. Quinlan *Induction of Decision Trees*. [*Εισαγωγή των Δέντρων Αποφάσεων*]. <http://hunch.net/~coms-4771/quinlan.pdf>
- [5] David Arthur, Sergei Vassilvitskii *k-means++: The Advantages of Careful Seeding*. [*k-means++: Τα πλεονεκτήματα της προσεκτικής επιλογής αρχικών κέντρων*]. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

*ΒΙΒΛΙΟΓΡΑΦΙΑ*

---

# Παράρτημα Α'

## Αντιστοίχηση με ορολογία διεθνούς βιβλιογραφίας

- ανάλυση κύριων συνιστωσών principal component analysis (PCA)
- απλός ταξινομητής Βαψες naive Bayes classifier
- βαθιά μάθηση deep learning
- βιβλιοθήκη λογισμικού software library
- γραμμική παλινδρόμηση linear regression
- δεδομένα data
- δέντρο αποφάσεων decision tree
- διαμοιραζόμενη μνήμη shared memory
- διεπαφή προγραμματισμού εφαρμογών application programming interface
- εκμάθηση δέντρων αποφάσεων decision tree learning
- εμπρόσθια τροφοδότηση feedforward
- επιβλεπόμενη μάθηση supervised learning
- επικλινής κάθισμας gradient descent
- k*-κοντινότεροι γείτονες *k*-nearest neighbors
- λογιστική παλινδρόμηση logistic regression
- μέθοδος ελαχίστων τετραγώνων ordinary least squares
- μέθοδος του αγκώνα elbow method
- μείωση της διαστασιμότητας dimensionality reduction
- μεταβλητή περιβάλλοντος environment variable

**μεταφραστής** compiler

**μη-επιβλεπόμενη μάθηση** unsupervised learning

**μηχανές διανυσμάτων υποστήριξης** support vector machines

**μηχανική μάθηση** machine learning

**οδηγίες μεταφραστή** compiler instructions

**ομαδοποίηση** clustering

**ομαδοποίηση  $k$ -μέσων**  $k$ -mean clustering

**οπισθοδρομική διάδοση** backpropagation

**παλινδρόμηση** regression

**παράλληλη υπολογιστική** parallel computing

**παράλληλο υπολογιστικό σύστημα** parallel computer

**παράλληλος προγραμματισμός** parallel programming

**συγχρονισμός** concurrency

**ταξινόμηση** classification

**τεχνητό νευρωνικό δίκτυο** artificial neural network

**υπορουτίνα** subroutine

# Παράρτημα Β'

## Σύνολα δεδομένων

Μαζί με τη βιβλιοθήκη metis περιλαμβάνεται ένας μικρός αριθμός συνόλων δεδομένων που μπορούν να χρησιμοποιηθούν για επίδειξη των διαφόρων αλγορίθμων. Αυτά τα σύνολα δεδομένων-αιθύρματα αντιπροσωπεύουν μερικά από τα είδη προβλημάτων που μπορούν να αντιμετωπιστούν και από πραγματικές εφαρμογές. Η φόρτωσή τους μπορεί να γίνει εύκολα με μία γραμμή κώδικα.

Στο παρόν παράρτημα θα γίνει περιγραφή αυτών των συνόλων δεδομένων στα οποία γίνεται αναφορά και στο Κεφάλαιο 5 που περιλαμβάνει κάποιες απλές εφαρμογές που χρησιμοποιούν τις μεθόδους που δημιουργήθηκαν στα πλαίσια κατασκευής της βιβλιοθήκης.

### B'.1 Σύνολο δεδομένων Diabetes

Τα συγκεκριμένα δεδομένα μετρούν 10 μεταβλητές σχετικές με τα χαρακτηριστικά και την φυσική κατάσταση 442 διαβητικών ασθενών μία δεδομένη στιγμή, και περιέχουν επιπλέον μία μεταβλητή απόκρισης σχετική με την πρόοδο της ασθένειας ένα χρόνο μετά από αυτές τις μετρήσεις. Για ανάδειξη των δυνατοτήτων της κλάσης LinearRegression έχει επινοηθεί μία επιπλέον μεταβλητή εξόδου που απορρέει από την προηγούμενη προσθέτοντάς της μία τυχαία τιμή μεταξύ 0 και 20.

Τα δεδομένα εισόδου κανονικοποιούνται έτσι ώστε να έχουν μέσο όρο 0 και ώστε το άθροισμα των τετραγώνων τους να ισούται με 1. Η δημιουργία του συνόλου έχει ως εξής:

```
std::string filePath = "../data/diabetes.tab.txt";  
  
metis::DataSet *input = new metis::DataSet(filePath, '\t');
```

```

input->create({0,1,2,3,4,5,6,7,8,9});
input->applyStandardization();
input->scaleSquaredLengthToOne();

metis::DataSet *output = new metis::DataSet(filePath, '\t');
output->create({10,11});

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);

```

*H ανάγνωση του αρχείου ακολουθείται από δημιουργία των συνόλων εισόδου και εξόδου.*

## B'.2 Σύνολο δεδομένων banknote authentication

Το σύνολο banknote authentication αποτελεί ένα ιδιαίτερα απλό σύνολο δεδομένων που μετράει τις τιμές τεσσάρων χαρακτηριστικών 1372 χαρτονομισμάτων και αναφέρει επιπλέον αν καθένα από αυτά είναι αυθεντικό ή όχι. Και αυτό το σύνολο δεδομένων υπάρχει ως άθυρμα στη βιβλιοθήκη metis. Οι τιμές του κάθε χαρακτηριστικού εισόδου παίρνουν συνεχείς θετικές και αρνητικές τιμές οι οποίες θα κανονικοποιηθούν κατά την ανάγνωση και δημιουργία του συνόλου, η οποία γίνεται ως εξής:

```

metis::DataSet *input = new metis::DataSet(filePath, ',');
input->create({0,1,2,3});
input->applyStandardization();

metis::DataSet *output = new metis::DataSet(filePath, ',');
output->create({4}, {4});

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);

```

*Δημιουργία του συνόλου δεδομένων banknote authentication.*

## B'.3 Σύνολο δεδομένων Iris

Το σύνολο Iris περιέχει μετρήσεις για τέσσερα χαρακτηριστικά τριών διαφορετικών ειδών του λουλουδιού της ίριδας. Περιέχονται συνολικά 150 παραδείγματα, 50 για κάθε είδος, ενώ αναφέρεται αυτό καθαυτό και το είδος στο οποίο ανήκει το λουλούδι που καταγράφεται. Το συγκεκριμένο σύνολο χρησιμοποιείται συχνά στα αρχικά στάδια ανάπτυξης και υλοποίησης αλγορίθμων μηχανικής μάθησης που αποσκοπούν

σε ταξινόμηση ή ομαδοποίηση.

Η δημιουργία του συνόλου για χρήση με τις υλοποιήσεις της βιβλιοθήκης metis φαίνεται παρακάτω:

```
metis::DataSet *input = new metis::DataSet(filePath, ',');
input->create({0,1,2,3});
input->applyStandardization();

metis::DataSet *output = new metis::DataSet(filePath, ',');
output->create({4}, {4});

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);
```

Δημιουργία του συνόλου δεδομένων Iris.

## B'.4 Σύνολο δεδομένων Wine

Τα δεδομένα του συνόλου Wine είναι αποτέλεσμα χημικής ανάλυσης κρασίων που παρήχθησαν στην ίδια περιοχή της Ιταλίας αλλά προήλθαν από τρεις διαφορετικές καλλιέργειες. Μετράται η ποσότητα δεκατριών συστατικών μεταξύ 178 δειγμάτων κρασιού, και αναφέρεται φυσικά από ποιά καλλιέργεια προέρχεται το κάθε δείγμα. Η ανάγνωση γίνεται ως εξής:

```
metis::DataSet *input = new metis::DataSet(filePath, ',');
input->create({1,2,3,4,5,6,7,8,9,10,11,12,13});
input->applyStandardization();

metis::DataSet *output = new metis::DataSet(filePath, ',');
output->create({0}, {0});

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);
```

Δημιουργία του συνόλου δεδομένων Wine.

## B'.5 Σύνολο δεδομένων MNIST

Το σύνολο MNIST είναι ένα από τα πιο δημοφιλή σύνολα δεδομένων στη μηχανική μάθηση. Χρησιμοποιείται πολύ συχνά στα πλαίσια της αξιολόγησης αλγορίθμων ταξινόμησης. Το σύνολο MNIST αποτελείται από χειρόγραφα αριθμητικά ψηφία από 0 έως 9 τα οποία έχουν ψηφιοποιηθεί σε εικόνες μεγέθους 28x28 pixel.

Συγκεκριμένα, περιέχει 60,000 παραδείγματα που προορίζονται προς χρήση για εκπαίδευση, και άλλα 10,000 που προορίζονται για δοκιμή.

Το πρωτότυπο σύνολο αποτελείται από αρχεία εικόνας τύπου IDX αλλά στην εργασία χρησιμοποιείται μια μετατροπή των αρχείων αυτών σε μορφή κειμένου, με κάθε γραμμή του αρχείου κειμένου να αντιστοιχεί σε ένα παράδειγμα. Από την περιγραφή του συνόλου γίνεται κατανοητό ότι θα υπάρχουν 784 είσοδοι, μία για κάθε pixel, και 10 έξοδοι που θα αντιστοιχούν σε έκαστο από τα 10 πιθανά ψηφία. Η ανάγνωση του αρχείου και η επικείμενη δημιουργία του συνόλου δεδομένων γίνεται ως εξής:

```
metis::DataSet *input = new metis::DataSet(filePath, ',');
std::vector<unsigned> attr;
for (unsigned a = 1; a < 785; ++a) attr.push_back(a);
input->create(attr, {});
input->applyStandardization();

metis::DataSet *output = new metis::DataSet(filePath, ',');
std::vector<std::map<std::string, unsigned>> dict(1);
dict[0].insert(std::pair<std::string, unsigned>("0", 0));
dict[0].insert(std::pair<std::string, unsigned>("1", 1));
dict[0].insert(std::pair<std::string, unsigned>("2", 2));
dict[0].insert(std::pair<std::string, unsigned>("3", 3));
dict[0].insert(std::pair<std::string, unsigned>("4", 4));
dict[0].insert(std::pair<std::string, unsigned>("5", 5));
dict[0].insert(std::pair<std::string, unsigned>("6", 6));
dict[0].insert(std::pair<std::string, unsigned>("7", 7));
dict[0].insert(std::pair<std::string, unsigned>("8", 8));
dict[0].insert(std::pair<std::string, unsigned>("9", 9));
output->create({0}, {0}, dict);

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);
```

Δημιουργία του συνόλου δεδομένων MNIST.

Το πρώτο ψηφίο αποτελεί την μεταβλητή εξόδου, ενώ τα επόμενα 784 τις μεταβλητές εισόδου, δηλαδή την φωτεινότητα των pixel της εικόνας που έχουν μια τιμή από 0 έως 255. Στο σύνολο εφαρμόζεται και κανονικοποίηση.

## B'.6 Σύνολο δεδομένων Breast Cancer

Η μηχανική μάθηση βρίσκει πολύ συχνά εφαρμογή στην ογκολογία. Το σύνολο δεδομένων Breast Cancer παρέχει μετρήσεις για 201 ασθενείς καρκίνου μαστού,

μετρώντας 9 χαρακτηριστικά και παρέχοντας για την κάθε περίσταση μία κλάση. Τα 9 αυτά χαρακτηριστικά αντιμετωπίζονται ως ονομαστικά, αν και 3 από αυτά θα μπορούσαν να αναπαρασταθούν και με συνεχή τρόπο.

Λόγω του ότι στην παρούσα φάση η βιβλιοθήκη δεν υποστηρίζει σύνολα δεδομένων στα οποία λείπουν στοιχεία, το σύνολο δεδομένων έχει τροποποιηθεί ελαφρώς με σκοπό την αφαίρεση εννέα παραδειγμάτων στα οποία λείπουν μετρήσεις. Η ανάγνωσή του γίνεται ως εξής:

```
metis::DataSet *input = new metis::DataSet(filePath, ',');
input->create({1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8});

metis::DataSet *output = new metis::DataSet(filePath, ',');
output->create({0}, {0});

metis::DataLabeled *labeledData = new metis::DataLabeled(input, output);
```

*Δημιουργία του συνόλου δεδομένων Breast Cancer.*

