ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΕΚΠΑΙΔΕΥΣΗ ΑΛΓΟΡΙΘΜΩΝ ΤΑΞΙΝΟΜΗΣΗΣ ΓΙΑ ΧΑΡΑΚΤΗΡΙΣΜΟ ΑΔΡΟΝΙΚΩΝ ΠΙΔΑΚΩΝ ΣΤΟΝ ΑΝΙΧΕΥΤΗ CMS ΤΟΥ CERN

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΕΡΔΙΚΗΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΙΒΛΕΠΩΝ : ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΟΥΣΟΥΡΗΣ

ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2017

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 3 ΜΑΘΗΜΑΤΙΚΗ ΘΕΩΡΙΑ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΚΕΦΑΛΑΙΟ 4 ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

ΠΑΡΑΡΤΗΜΑΤΑ

ABSTRACT

The subject of this thesis is the study of the supervised machine learning classification algorithms. In particular, the algorithms which were studied and applied, analyzing their parameters in detail, were the linear discriminant, the Fisher method, the least square method for classification, the k nearest neighbors' method, support vector machine, neural networks, decision trees and the combination of classifiers (Boosting). The data we used were related to to high-energy physics, an object which requires big data analysis. More specifically, the data came from the compact muon solenoid (CMS) detector of the European Organization for Nuclear Research (CERN). The analysis was made both theoretically, studying the mathematical theory describing these algorithms, and practically by programming these methods with the TMVA software of ROOT and MATLAB. The code written for the study of the above methods with TMVA / ROOT is given in the annexes at the end.

ΕΥΧΑΡΙΣΤΙΕΣ

Η εργασία αυτή πραγματοποιήθηκε το ακαδημαϊκό έτος 2016-2017 στα πλαίσια εκπόνησης της διπλωματικής εργασίας των προπτυχιακών σπουδών της σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών (*Σ.Ε.Μ.Φ.Ε.*) του Εθνικού Μετσόβιου Πολυτεχνείου. Ο επιβλέπων καθηγητής της διπλωματικής εργασίας ήταν ο Κωνσταντίνος Κουσουρής, στον οποίο οφείλω τις ευχαριστίες μου για την συνεχή στήριξη και την βοήθεια καθόλη την διάρκεια του έτους. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου, καθώς και τους φίλους μου που με συμβουλεύουν και συμπαραστέκονται όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία γίνεται μια μελέτη των αλγορίθμων ταξινόμησης στην μηχανική μάθηση με επίβλεψη. Ειδικότερα οι αλγόριθμοι που μελετήθηκαν και εφαρμόστηκαν, εξετάζοντας αναλυτικά τις παραμέτρους τους είναι**:** ο απλός γραμμικός ταξινομητής, η μέθοδος Fisher, η μέθοδος ελαχίστων τετραγώνων για ταξινόμηση, η μέθοδος k πλησιέστερων γειτόνων, η μηχανή διανυσματικής στήριξης, τα νευρωνικά δίκτυα, τα δέντρα απόφασης και ο συνδυασμός ταξινομητών (Boosting).
Τα δεδομένα που χρησιμοποιήθηκαν σχετίζονται με την φυσική υψηλών ενεργειών, αντικείμενο το οποίο απαιτεί ανάλυση μεγάλου όγκου δεδομένων. Πιο συγκεκριμένα τα δεδομένα προήλθαν απο τον ανιχνευτή *compact muon solenoid* (CMS) του ευρωπαικού κέντρου πυρηνικών ερευνών (CERN). Η ανάλυση έγινε τόσο σε θεωρητικό επίπεδο, μελετώντας την μαθηματική θεωρία που περιγράφει τους αλγόριθμους αυτούς, όσο και σε προγραμματιστικό με τα λογισμικά TMVA του ROOT και MATLAB. Ο κώδικας που γράφτηκε για την μελέτη των παραπάνω μεθόδων στο TMVA/ROOT δίνεται στα παραρτήματα.

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 Η ΕΝΝΟΙΑ ΤΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Η μηχανική μάθηση (machine learning) είναι το πεδίο της επιστήμης των υπολογιστών που τους προσδίδει την ικανότητα της εκμάθησης. Ο πρωτοπόρος στην επιστήμη της τεχνητής νοημοσύνης Arthur Samuel εφηύρε τον όρο μηχανική μάθηση το 1959. Ο τομέας αυτός προέκυψε από την ανάπτυξη και τον συνδυασμό δύο προγενέστερων τομέων, αυτών της αναγνώρισης προτύπων (pattern recognition) και της υπολογιστικής θεωρίας της μάθησης (computational theory of learning) στην τεχνητή νοημοσύνη (artificial intelligence). Συνεπώς η μηχανική μάθηση ερευνά την κατασκευή και χρήση αλγορίθμων οι οποίοι είναι ικανοί να μαθαίνουν από τα δεδομένα ώστε να παίρνουν αυτόνομες αποφάσεις. Η παραπάνω ιδιότητα των αλγορίθμων αυτών διαφέρει ριζικά από την απλή εκτέλεση πάγιων προγραμματιστικών εντολών. [6]

1.1.1 Ορισμός μηχανικής μάθησης

Ένας σαφής ορισμός έχει δοθεί από τον Tom M. Mitchell ο οποίος διατύπωσε πως: Ένα πρόγραμμα υπολογιστή λέγεται πως μαθαίνει από εμπειρία Ε ως προς κλάση εργασιών Τ και μέτρου επίδοσης Ρ, εάν η επίδοση του στην κλάση εργασιών Τ με βάση την αποτίμηση του μέτρου Ρ, βελτιώνεται με την ίδια εμπειρία Ε. [5]

1.1.2 Κατηγορίες μηχανικής μάθησης

Η κατηγοριοποίηση των διεργασιών της μηχανικής μάθησης μπορεί να γίνει με βάση έναν από τους δύο άξονες, τον τρόπο εκμάθησης του αλγορίθμου καθώς και το είδος του σήματος εξόδου από το σύστημα μηχανικής μάθησης.

Με βάση τον τρόπο εκμάθησης του αλγορίθμου έχουμε τρεις κατηγορίες:

>Την επιβλεπόμενη μάθηση (Supervised learning). Στην περίπτωση αυτή στο σύνολο δεδομένων περιέχονται τα στοιχεία εισόδου (inputs) αλλά και και τα επιθυμητά αποτελέσματα για αυτά (targets). Τα δοθέντα αποτελέσματα έχουν τον ρόλο του “δασκάλου”, δηλαδή συντελούν στην εύρεση ενός γενικού κανόνα αντιστοίχισης εισόδων με αποτελέσματα.

>Την μη επιβλεπόμενη μάθηση (Unsupervised learning). Εδώ το σύνολο δεδομένων δεν παριέχει απαντήσεις, δηλαδή δεν υπάρχει εμπειρία

στον αλγόριθμο, οπότε η πρόκληση είναι να βρεθούν κρυφές δομές στα δεδομένα εισόδου.

>Την ενισχυτική μάθηση (Reinforcement learning). Αυτού του είδους η μάθηση περιλαμβάνει αλληλεπίδραση του αλγόριθμου με ένα δυναμικό περιβάλλον. Έτσι έχουμε συνεχώς νέα δεδομένα χωρίς να είναι εξαρχής σαφές εάν ο αλγόριθμος προσεγγίζει τον στόχο του.

Με βάση το σήμα εξόδου του συστήματος έχουμε:

>Την ταξινόμηση (Classification), που αντιστοιχεί τα δεδομένα εισόδου σε κλάσεις και επειδή υπάρχουν οι απαντήσεις υπάγεται στην μάθηση με επίβλεψη.

>Την παλινδρόμηση (Regression), που ανάλογα τον τύπο του μοντέλου της παράγεται συνεχής έξοδος και επίσης υπάγεται στην μάθηση με επίβλεψη.

>Την συσταδοποίηση (Clustering), όπου τα δεδομένα εισόδου δεν περιλαμβάνουν απαντήσεις και πρέπει να κατηγοριοποιηθούν σε ομάδες (clusters), οι οποίες δεν είναι γνωστές από πριν. Η συσταδοποίηση είναι κλασικό παράδειγμα μη επιβλεπόμενης μάθησης.

>Την εκτίμηση πυκνότητας (Density Estimator), όπου μας δίνει την κατανομή των δεδομένων εισόδου.

>Την μείωση διαστάσεων (Dimensionality reduction), όπου απλοποιούνται τα δεδομένα εισόδου με την απεικόνιση τους σε ένα υπερεπίπεδο λιγότερων διαστάσεων.

1.1.3 Εφαρμογές της μηχανικής μάθησης

Μερικές ενδεικτικές από τις αναρίθμητες εφαρμογές της μηχανικής μάθησης εμπίπτουν στους παρακάτω τομείς:

Υπολογιστική όραση

Ανάκτηση πληροφορίας

Βελτιστοποίηση

Αναγνώριση ομιλίας

Αυτόνομα οχήματα

Αναγνώριση εικόνας

Αναγνώριση χειρόγραφων

Ιατρική διάγνωση

Βιοπληροφορική

Διαδικτυακή αναζήτηση

Εξατομικευμένη διαφήμιση	*Προστασία δεδομένων*
Χρηματιστηριακή ανάλυση	*Προγραμματισμός λογισμικών*
Αναγνώριση διαδικτυακής απάτης	*Ηλεκτρονικά παιχνίδια*
Πρόβλεψη συμπεριφοράς	*Μετάφραση*
Ανάλυση χρονοσειρών	*Δυναμική τιμολόγηση*

1.1.4 Χρονολόγιο ανακαλύψεων

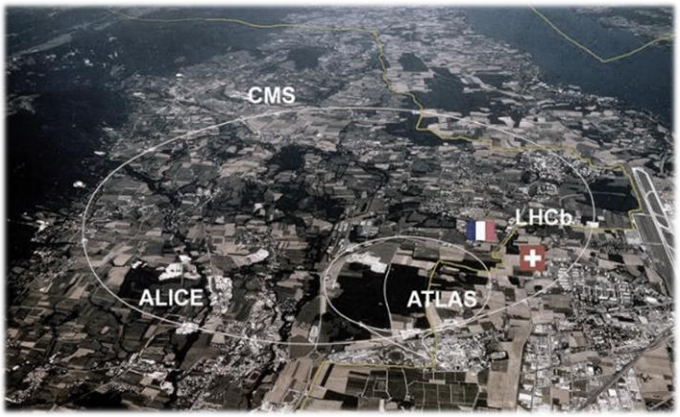
Χρονολόγιο με τα βασικότερες ανακαλύψεις αλγορίθμων και θεωρημάτων που συνέβαλαν στην ανάπτυξη της μηχανικής μάθησης:

1763	*Θεώρημα Bayes*
1805	*Μέθοδος ελαχίστων τετραγώνων*
1913	*Μαρκοβιανές αλυσίδες*
1957	*Αλγόριθμος Perceptron*
1967	*Μέθοδος πλησιέστερων γειτόνων (kNN)*
1980	*Neocogniton (πολυστρωματικό νευρωνικό δίκτυο – ANN)*
1982	*Αναδρομικό νευρωνικό δίκτυο (Hopfield)*
1986	*Αλγόριθμος οπισθοδιάδοσης (Backpropagation)*
1989	*Ενισχυτική μάθηση (Reinforcement learning)*
1992	*Μηχανές διανυσματικής στήριξης (SVM)*
1995	*Αλγόριθμος Random forest*
1997	*Long-short term memory αναδρομικά νευρωνικά δίκτυα, (LSTM)*

1.2 Ο ΜΕΓΑΛΟΣ ΑΔΡΟΝΙΚΟΣ ΕΠΙΤΑΧΥΝΤΗΣ LHC ΚΑΙ ΤΟ ΠΕΙΡΑΜΑ CMS

1.2.1 THE LARGE HADRON COLLIDER

Ο Large Hadron Collider (LHC) βρίσκεται στις εγκαταστάσεις του Ευρωπαικού Κέντρου Πυρηνικών Ερευνών (CERN) και αποτελεί τον μεγαλύτερο επιταχυντή σωματιδίων που έχει κατασκευαστεί ποτέ. Τέθηκε για πρώτη φορά σε λειτουργία τον Σεπτέμβρη του 2008, έχει κυκλικό σχήμα με περίμετρο 27 χιλιομέτρων και βρίσκεται στα Γάλλο-Ελβετικά σύνορα μεταξύ 50 και 175 μέτρων κάτω από την γή.

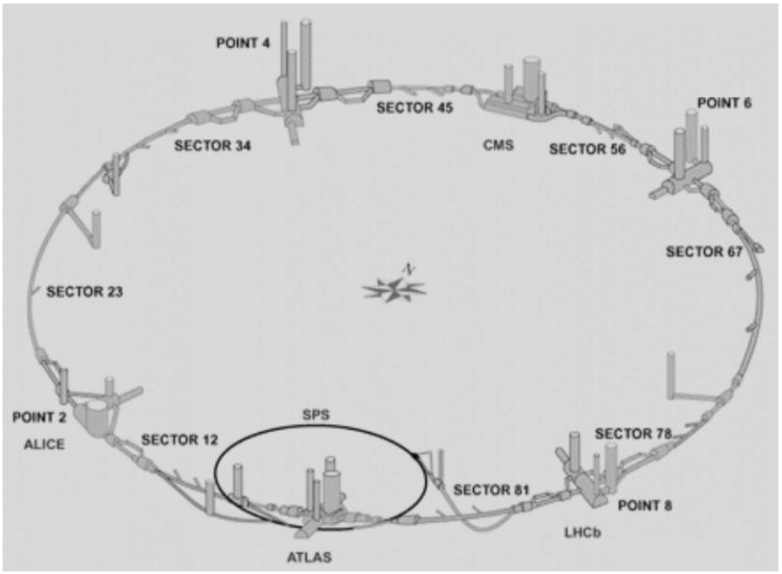


Στον LHC έχουμε επιτάχυνση 2 δεσμών πρωτονίων που κινούνται σε αντίθετες κατευθύνσεις με ταχύτητες που προσεγγίζουν την ταχύτητα του φωτός. Οι δέσμες καθοδηγούνται από ένα ισχυρό μαγνητικό πεδίο (8,3 Tesla) που διατηρείται από υπεραγώγιμους ηλεκτρομαγνήτες. Οι ηλεκτρομαγνήτες είναι κατασκευασμένοι από πηνία ειδικού ηλεκτρικού καλωδίου που λειτουργεί σε υπεραγώγιμη κατάσταση, διοχετεύοντας αποτελεσματικά την ηλεκτρική ενέργεια χωρίς αντίσταση ή απώλεια ενέργειας. Αυτό απαιτεί ψύξη των μαγνητών στους 1,85 Kelvin, θερμοκρασία χαμηλότερη από την μέση θερμοκρασία στο διάστημα. Για τον λόγο αυτό μεγάλο μέρος του επιταχυντή συνδέεται με σύστημα διανομής υγρού ηλίου, το οποίο ψύχει τους μαγνήτες. Στον LHC συγκρούσεις έχουμε σε 4 προκαθορισμένα σημεία, όπου έχουν τοποθετηθεί ανιχνευτές σωματιδίων. Ο επιταχυντής έχει δυνατότητα να επιτύχει συγκρούσεις με ενέργεια 13TeV στο σύστημα κέντρου μάζας, πράγμα που τον καθιστά και τον ισχυρότερο επιταχυντή μέχρι σήμερα. Στόχος του είναι η μελέτη της δομής, της συμπεριφοράς, καθώς και των αλληλεπιδράσεων των στοιχειωδών σωματιδίων. [14]

- Ανιχνευτικές διατάξεις του LHC

Ο επιταχυντής έχει 8 σημεία πρόσβασης. Συγκρούσεις παρατηρούνται σε 4 προκαθορισμένα σημεία από αυτά, πάνω στον δακτύλιο του επιταχυντή, όπου βρίσκονται τοποθετημένοι οι ανιχνευτές. Αυτοί είναι οι ακόλουθοι:

> A Toroidal LHC Apparatus (ATLAS)	point 1
> Compact Muon Solenoid (CMS)	point 5
> LHC Beauty (LHCb)	point 2
> A Large Ion Collider Experiment (ALICE)	point 8



1.2.2 COMPACT MUON SOLENOID

Ο Compact Muon Solenoid (CMS) είναι ένας ανιχνευτής γενικού ενδιαφέροντος σχεδιασμένος να ανιχνεύει όλα τα στοιχειώδη σωματίδια. Το σύνολο των ανιχνευτικών διατάξεων του σχηματίζουν ένα κύλινδρο (barrel), με πολλές ομοαξονικές στην δέσμη κυλινδρικές επιφάνειες. Η ανάγκη ανίχνευσης όλων των σωματιδίων κατά την σύγκρουση των πρωτονίων στο κέντρο του CMS επιβάλλει την επιπλέον προσαρμογή δύο κάθετων δίσκων (endcaps) στον άξονα της δέσμης, ώστε ο ανιχνευτής να είναι ερμητικά κλειστός. Ο ανιχνευτής έχει μήκος 28,7 μέτρα, διάμετρο 15 μέτρα και ζυγίζει περίπου 14 χιλιάδες τόνους. Οι δύο ιδιαιτερότητες του ανιχνευτή αυτού είναι πως έχει την δυνατότητα να ανιχνεύει μιόνια με μεγάλη ακρίβεια, καθώς και ο

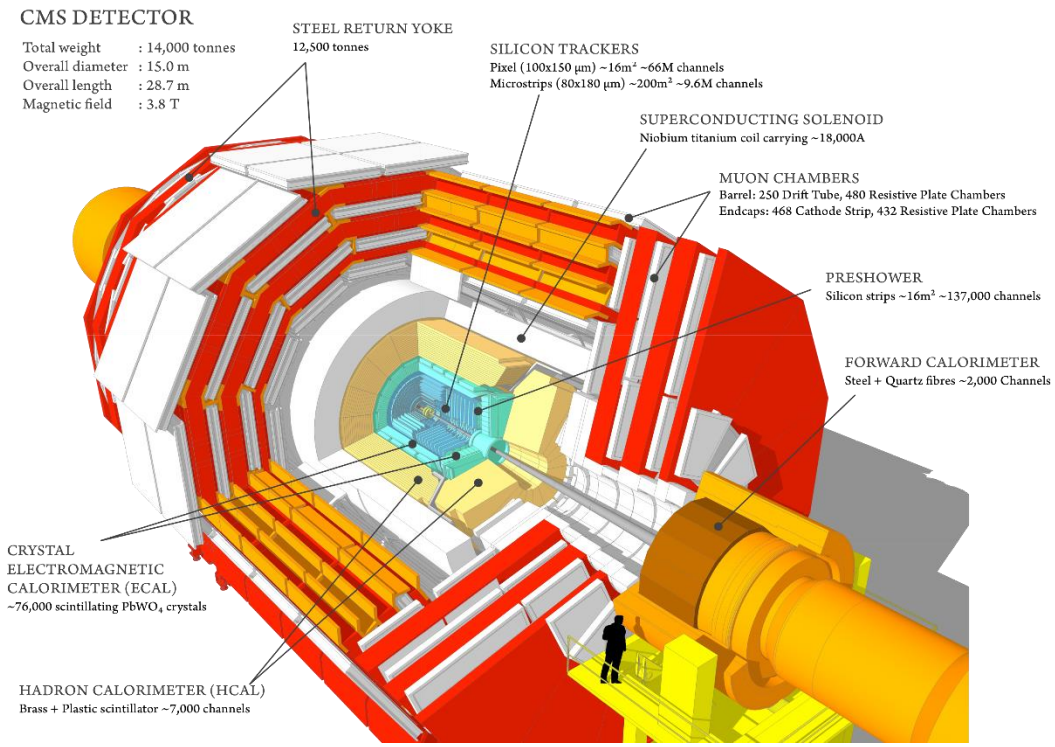
ισχυρός σωληνοειδής μαγνήτης του. Αυτός έχει μήκος 13 μέτρα και παράγει μαγνητικό πεδίο έντασης 3,8 Tesla παράλληλα στον άξονα της δέσμης, ώστε να καμπυλώνει τις τροχιές των φορτισμένων σωματιδίων. [14]

Ο ανιχνευτής CMS αποτελείται από τις εξής ανιχνευτικές διατάξεις:

> Το σύστημα ανίχνευσης τροχιών (Tracker Detector), που δίνει πληροφορίες για το είδος και την ορμή των σωματιδίων που προέρχονται από την σύγκρουση των δεσμών.
> Το ηλεκτρομαγνητικό θερμιδόμετρο (ECAL), που συμβάλει στην ανίχνευση φωτονίων και ηλεκτρονίων.
> Το αδρονικό θερμιδόμετρο (HCAL), που συμβάλει στην ανίχνευση και μελέτη των ουδέτερων αλλά και φορτισμένων αδρονίων.
> Ένα μιονικό σύστημα υψηλής απόδοσης (Muon Detectors) για μελέτη και ανίχνευση μιονίων.

Συνεπώς ο ανιχνευτής CMS αποτελείται από πολλά επίπεδα, με κάθε ένα από αυτά να αποτελεί έναν υπο-ανιχνευτή με συγκεκριμένο ρόλο. Η διάταξη τους λοιπόν ξεκινώντας από την δέσμη προς τα έξω έχει την εξής σειρά:

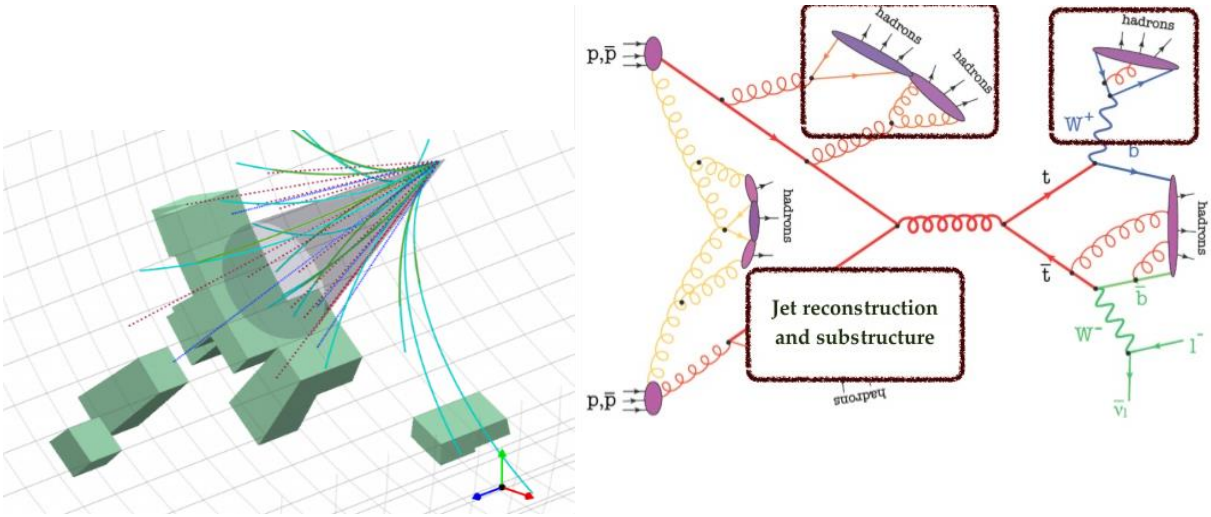
Tracker -> ECAL -> HCAL -> Μαγνήτης -> Ανιχνευτές μιονίων



1.3 ΑΔΡΟΝΙΚΟΙ ΠΙΔΑΚΕΣ

1.3.1 Ορισμός των αδρονικών πιδάκων (jets)

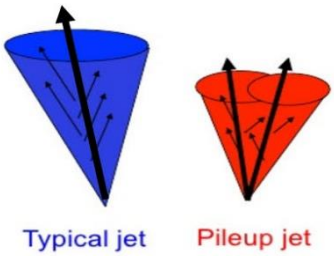
Ένα jet είναι ένας στενός κώνος από αδρόνια και άλλα σωματίδια που παράγονται από την αδρονοποίηση ενός κουάρκ ή γλουονίου. Τα σωματίδια που φέρουν χρώμα, όπως τα κουάρκ, δεν μπορούν να υπάρχουν σε ελεύθερη μορφή εξαιτίας του περιορισμού (confinement) της κβαντικής χρωμαδυναμικής (QCD) που επιτρέπει μόνο άχρωμες καταστάσεις. Όταν ένα σωματίδιο από παρτόνια (κουαρκς η γλουόνια) απομακρύνεται ένα συστατικό του παίρνει μαζι του χρώμα. Για να υπακούσουν στον περιορισμό, αυτά τα συστατικά δημιουργούν άλλα έγχρωμα αντικείμενα γύρω τους για να σχηματίσουν άχρωμες τελικές καταστάσεις (αδρόνια). Το σύνολο αυτών των αντικειμένων ονομάζεται jet. Τα jets μετρώνται σε ανιχνευτές σωματιδίων και μελετώνται για να προσδιοριστούν οι ιδιότητες των αρχικών κουάρκ.



Τα jets μπορούν να θεωρηθούν και ως τοπικά clusters ενέργειας συνοδευόμενα από τροχιές σωματιδίων. Για το λόγο αυτό μπορούμε να τα διαχωρίσουμε σε δύο κατηγορίες:

> Prompt Jets *(αυτά που προέρχονται από την κύρια σύγκρουση)*

> Pileup Jets *(αυτά που προέρχονται από τις δευτερεύουσες συγκρούσεις)*



1.3.2 Ανακατασκευή των αδρονικών πιδάκων (jets)

Η ανακατασκευή των jet από τα δεδομένα των υποανιχνευτών παίζει πολύ σημαντικό ρόλο. Λόγω της διαδικασίας του καταιγισμού, μικρές αλλαγές στην αρχή της αδρονοποίησης είναι πιθανό να οδηγήσουν σε σοβαρές αποκλίσεις. Επιπρόσθετα τα τελικά αποτελέσματα αλλοιώνονται λόγω του ηλεκτρονικού θορύβου, μετατρέποντας την διαδικασία ταυτοποίησης σε ακόμα δυσκολότερη υπόθεση. Για την επίλυση των παραπάνω προβλημάτων έχουν δημιουργηθεί διάφοροι αλγόριθμοι επανακατασκευής των jet, με πιο διαδεδομένο τον k_T algorithm .

- k_T algorithm

>Ορίζουμε την απόσταση d_{ij} μεταξύ δύο σωματιδίων i και j ως:

$$d_{ij} = \min\left(k_{Ti}^{2p}, k_{Tj}^{2p}\right)\frac{\Delta_{ij}}{D}$$

$$\text{όπου} \quad \Delta_{ij}^2 = (y_i - y_j)^2 + (\varphi_i - \varphi_j)^2$$

(για τον ορισμό του χώρου y,φ -> βλ. σελίδες 22 και 23)

> Ορίζουμε την απόσταση d_{iB} μεταξύ σωματιδίου i και της δέσμης (B) ως:

$$d_{iB} = k_{Ti}^{2p}$$

>Υπολογισμός όλων των αποστάσεων και εύρεση της μικρότερης.

>>Εάν η μικρότερη είναι μία d_{ij}, πρόσθεση των ορμών των δύο σωματιδίων, ανανέωση των αποστάσεων και εύρεση της επόμενης μικρότερης.

>>Εάν η μικρότερη είναι μία d_{iB}, τότε αφαίρεση του σωματιδίου i και σχηματισμός του τελικού jet.

>Επανάληψη της διαδικασίας εώς ότου να εισαχθούν σε clusters όλα τα σωματίδια στο jet.

Τα D και p είναι παράμετροι του αλγόριθμου, το p στον k_T αλγόριθμο είναι 2 και το D^2 είναι η ελάχιστη απόσταση Δ^2 για οποιαδήποτε τελικά jets.

ΚΕΦΑΛΑΙΟ 2
ΠΕΡΙΓΡΑΦΗ ΜΕΤΑΒΛΗΤΩΝ ΧΑΡΑΚΤΗΡΙΣΜΟΥ

2. ΠΕΡΙΓΡΑΦΗ ΜΕΤΑΒΛΗΤΩΝ ΧΑΡΑΚΤΗΡΙΣΜΟΥ [12] , [13]

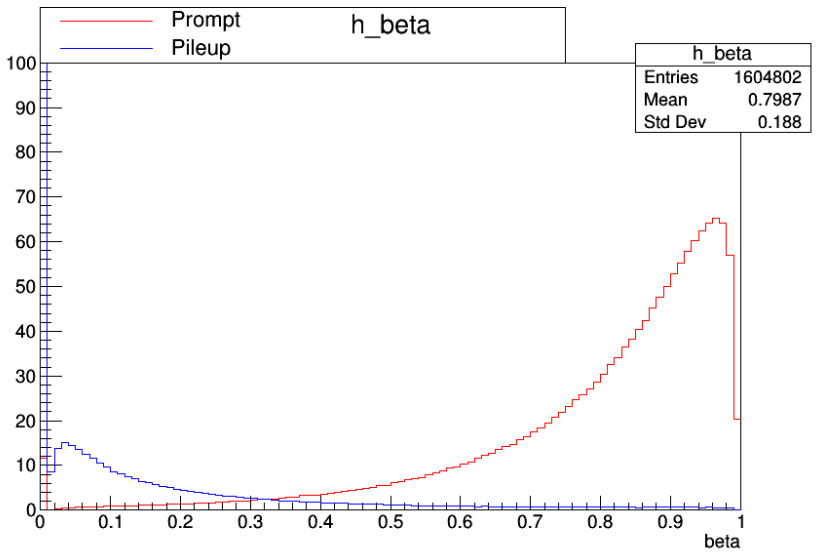
Η κατάλληλη επιλογή των μεταβλητών ταξινόμησης (predictors) είναι ιδιαίτερα σημαντική καθώς με βάση αυτές γίνονται οι διακρίσεις, άρα παίρνονται και οι αποφάσεις για το σύνολο δεδομένων. Κάθε μεταβλητή για ένα συγκεκριμένο σύνολο δεδομένων έχει μια διακριτική ικανότητα, η οποία όταν είναι μεγαλύτερη από αυτές των υπολοίπων, μας καθιστά την μεταβλητή αυτή ιδιαίτερα σημαντική για την ταξινόμηση. Οι μεταβλητές των δεδομένων μας , τα οποία έχουν προέλθει από προσομοίωση (Monte Carlo), έχουν επιλεγεί στην παρούσα διπλωματική να είναι 14 και παρακάτω αναλύεται η φυσική τους σημασία.

2.1.Μεταβλητή “beta”

Η μεταβλητή beta ορίζεται ως το ποσοστό της εγκάρσιας ορμής (P_T) των φορτισμένων σωματιδίων του jet που προέρχονται από την κύρια σύγκρουση, προς την συνολική εγκάρσια ορμή. Δηλαδή:

$$\beta = \frac{\Sigma_{i \in PV} \; P_{Ti}}{\Sigma_k \; P_{Tk}}$$

με το i να αναφέρεται στην κύρια σύγκρουση και το k σε όλα τα στοιχεία του jet.

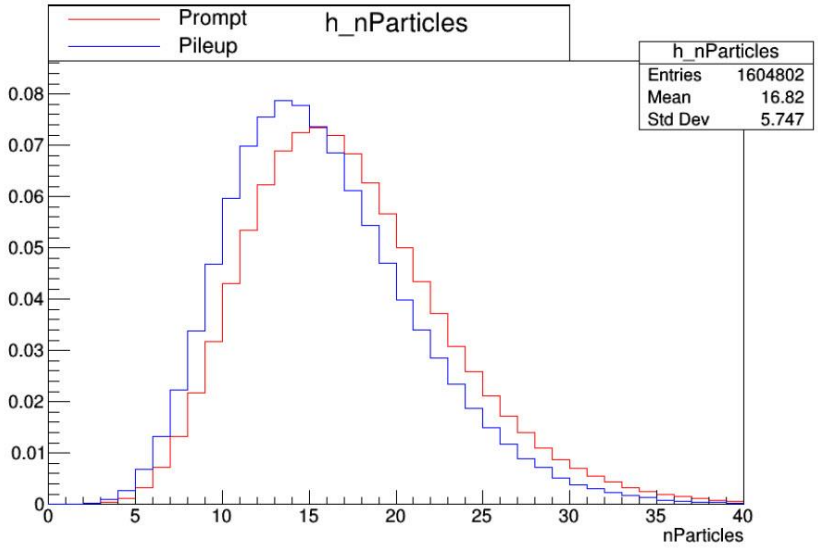


Απεικόνιση της μεταβλητής beta στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

2.2. Μεταβλητή “nPatricles”

Η μεβλητή αυτή μας δίνει τον συνολικό αριθμό των σωματιδίων στο jet.

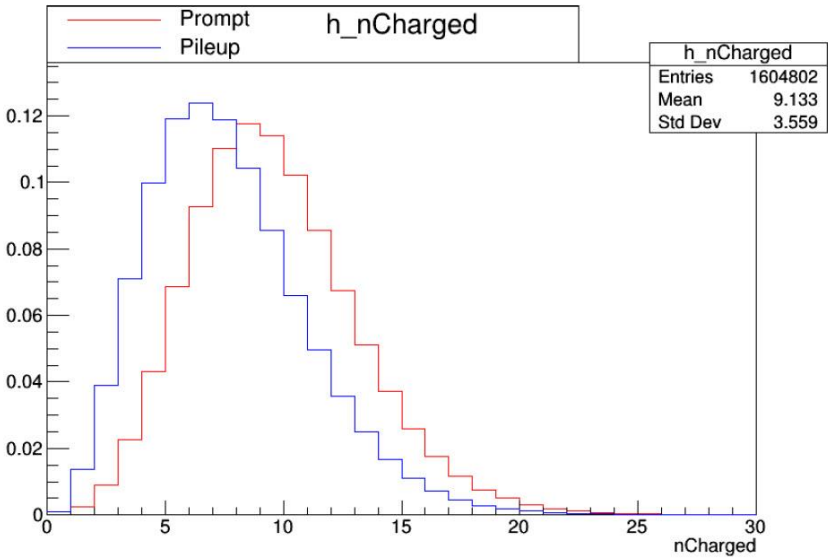
nParticles = nCharged + nNeutral



Απεικόνιση της μεταβλητής nParticles στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

2.3. Μεταβλητή “nCharged”

Η μεβλητή αυτή μας δίνει τον αριθμό των φορτισμένων σωματιδίων στο jet.



Απεικόνιση της μεταβλητής nCharged στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

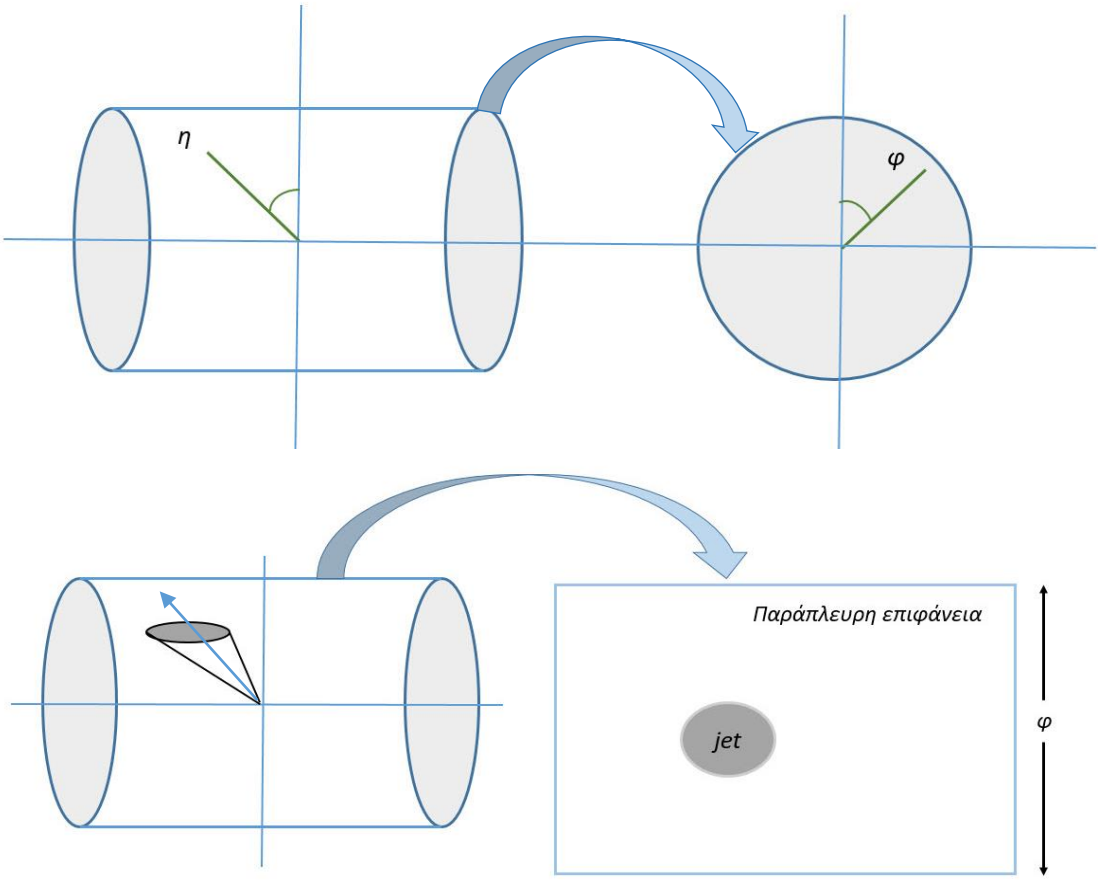
2.4. Μεταβλητή “dR2Mean”

Η μεταβλητή αυτή ορίζεται από την σταθμισμένη μέση τιμή του τετραγώνου της απόστασης όλων των στοιχείων του jet από τον άξονα του jet. Συνεπώς περιγράφεται από την σχέση:

$$\text{dR2Mean: } \langle \Delta R^2 \rangle = \frac{\sum_i \Delta R_i^2 P_{Ti}{}^2}{\sum_i P_{Ti}{}^2}$$

$$\text{οπου} \quad \Delta R^2 = (\Delta\eta)^2 + (\Delta\phi)^2$$

Τα μεγέθη η και φ είναι χωρικές συντεταγμένες που συμβάλλουν στον προσδιορισμό της θέσης του jet μέσα στον ανιχνευτή. Στις παρακάτω δύο εικόνες μπορούμε να δούμε τις γωνίες στις οποίες αναφερόμαστε.

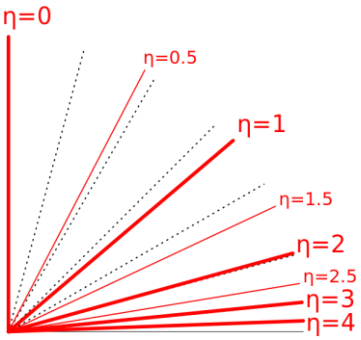


Το μέγεθος φ είναι η αζιμουθιακή γωνία και μετριέται σε rad. Οι τιμές που παίρνει είναι μεταξύ –π και π.

Το μέγεθος η λέγεται ψευδωκύτητα (pseudorapidity) και όπως προαναφέρθηκε είναι μια χωρική συντεταγμένη που περιγράφει την γωνία της ορμής ενός σωματιδίου σε σχέση με τον άξονα της δέσμης (beam axis).

Η σχέση που περιγράφει την ψευδωκύτητα είναι:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right)$$



Όπου θ είναι η πολική γωνία, δηλαδή αυτή μεταξύ της ορμής του σωματιδίου **p** και του θετικού ημιάξονα της δέσμης. Επιλύοντας την παραπάνω εξίσωση ως προς θ έχουμε:

$$\theta = 2\cdot\arctan(e^{-\eta})$$

Η ψευδωκύτητα μπορεί επίσης να δοθεί σε συνάρτηση με την ορμή ως εξής:

$$\eta = \frac{1}{2}\ln\left(\frac{|\mathbf{p}|+p_L}{|\mathbf{p}|-p_L}\right) = \operatorname{arctanh}\left(\frac{p_L}{|\mathbf{p}|}\right)$$

Με $|\mathbf{p}|$ το μέτρο της τρισδιάστατης ορμής και p_L η συνιστώσα της ορμής κατά μήκος του άξονα της δέσμης (longitudinal momentum).

Οι τιμές που παίρνει το η είναι από -∞ εώς +∞ με βάση την μαθηματική περιγραφή του, όμως στην πράξη σε έναν ανιχνευτή είναι από -5 εως +5.

Η φυσική σημασία της ψευδωκύτητας (η) έγκειται στο γεγονός ότι με βάση κάποιες προσεγγίσεις μας δίνει την ωκύτητα (y rapidity). Πράγματι όταν ένα σωματίδιο έχει ταχύτητα κοντά στην ταχύτητα του φωτός ή έχει αμελητέα μάζα, τότε μπορούμε να κάνουμε την ακόλουθη προσέγγιση:

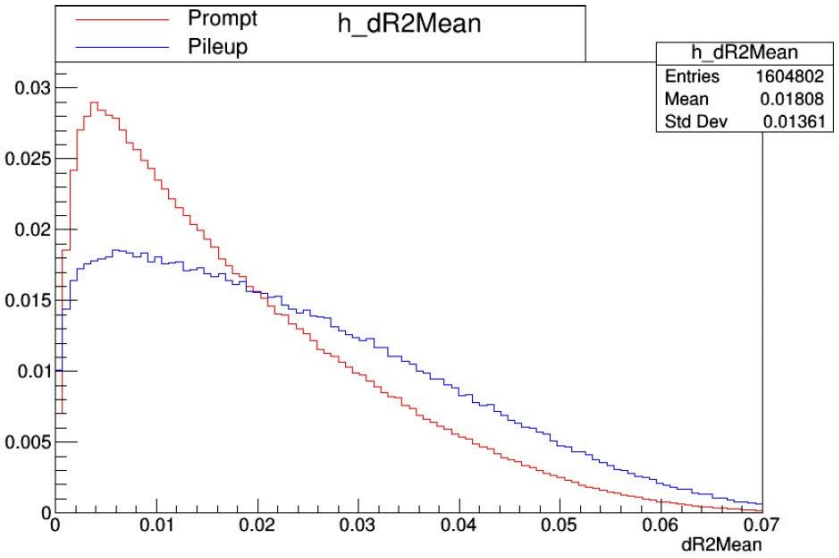
$$m << p \Rightarrow E \approx p \Rightarrow \eta \approx y$$

Επομένως ο ορισμός της ωκύτητας στην σωματιδιακή φυσική είναι:

$$y = \frac{1}{2}\ln\left(\frac{E+p_L}{E-p_L}\right)$$

Επίσης η ωκύτητα σε συνάρτηση της ψευδωκύτητας μπορεί να δοθεί ως εξής:

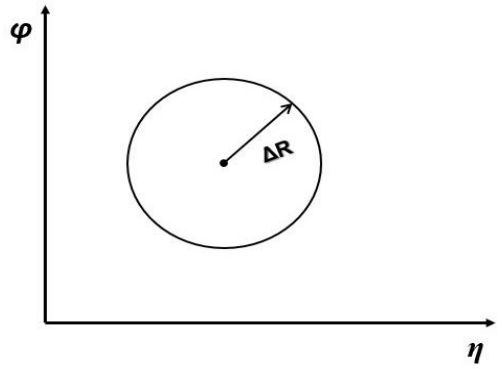
$$y = \frac{1}{2}\ln\left(\frac{\sqrt{m^2+P_T^2\cosh^2\eta}+P_T\sinh\eta}{\sqrt{m^2+P_T^2}}\right)$$



Απεικόνιση της μεταβλητής dR2Mean στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

2.5 - 2.8. Μεταβλητές “frac01” εως “frac04”

Η μεταβλητή frac01 ορίζεται από το ποσοστό της εγκάρσιας ορμής P_T των συστατικών που βρίσκονται σε απόσταση ΔR < 0,1 από τον άξονα του jet. Η ορμή P_T (transverse momentum) είναι η συνιστώσα της τρισδιάστατης ορμής κάθετη στον άξονα της δέσμης. Ετσι ορίζεται ως :



$$p_T = \sqrt{(p_X)^2 + (p_Y)^2}$$

Επίσης όπως προαναφέρθηκε η απόσταση από τον άξονα του jet ορίζεται με τον εξης τρόπο:

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$$

Έτσι με όμοιο τρόπο ορίζονται και οι υπόλοιπες τρεις μεταβλητές frac02, frac03 και frac04 με μόνη διαφορά πως θέλουμε διαφορετική απόσταση ΔR κάθε φορά. Οι αποστάσεις για την κάθε μία είναι:

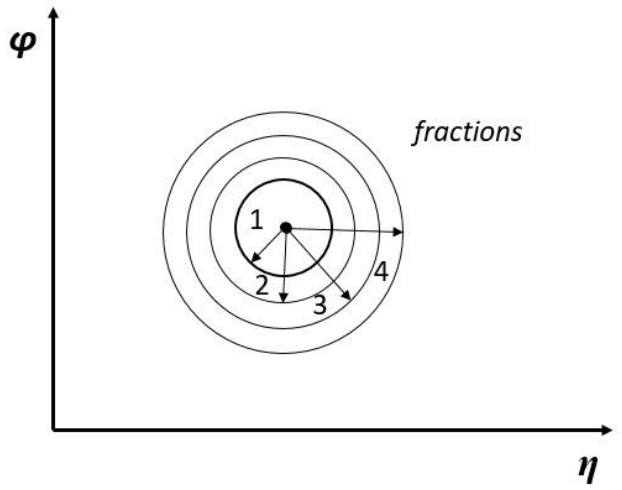
frac01 : 0,0 < ΔR < 0,1

frac02 : 0,1 < ΔR < 0,2

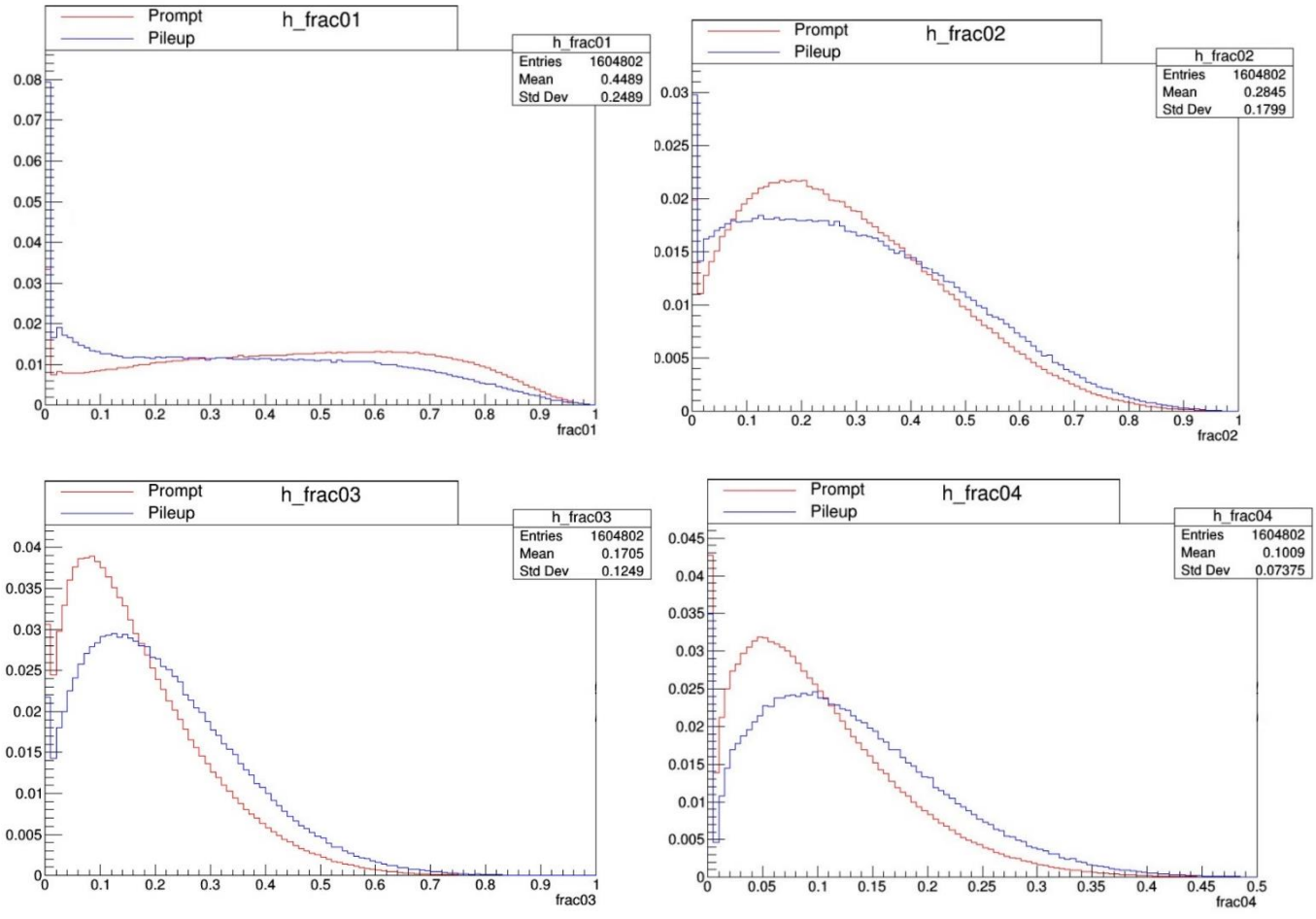
frac03 : 0,2 < ΔR < 0,3

frac04 : 0,3 < ΔR < 0,4

Συνεπώς από τους ορισμούς που δόθηκαν προκύπτει η ακόλουθη απεικόνιση:



Απεικόνιση των μεταβλητών frac01 εώς frac04 στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

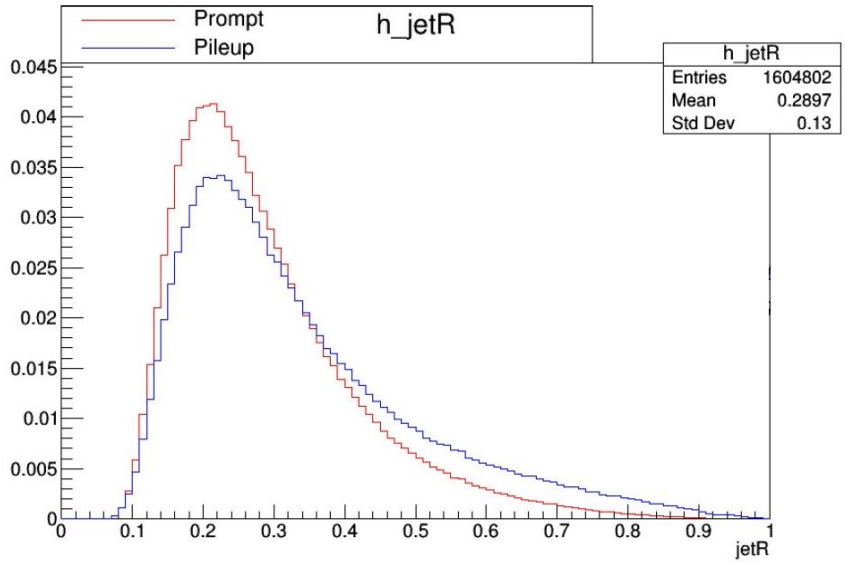


2.9. Μεταβλητή “jetR”

Η μεταβλητή jetR ορίζεται ως το ποσοστό της εγκάρσιας ορμής P_T ενός jet που προέρχεται από το στοιχείο με την μεγαλύτερη εγκάρσια ορμή, δηλαδή ορίζεται ως:

$$\text{jetR} = \frac{P_{Tk}}{\sum_i P_{Ti}}$$

οπου k το στοιχείο με την μεγαλύτερη εγκάρσια ορμή και το i να τρέχει πάνω σε όλα τα στοιχεία του jet.



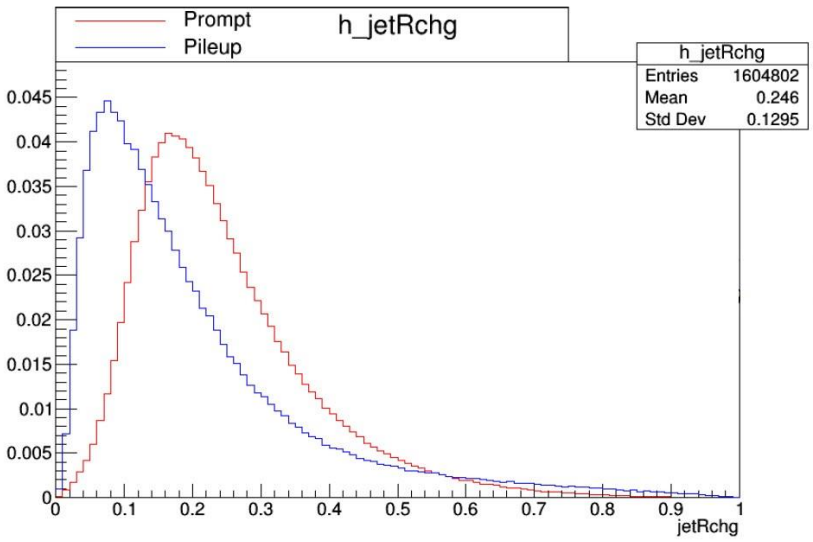
Απεικόνιση της μεταβλητής jetR στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

2.10. Μεταβλητή “jetRchg”

Η μεταβλητή jetRchg ορίζεται ως το ποσοστό της εγκάρσιας ορμής P_T ενός jet που προέρχεται από το φορτισμένο στοιχείο με την μεγαλύτερη εγκάρσια ορμή, δηλαδή ορίζεται ως:

$$\text{jetRchg} = \frac{P_{Tj}}{\sum_i P_{Ti}}$$

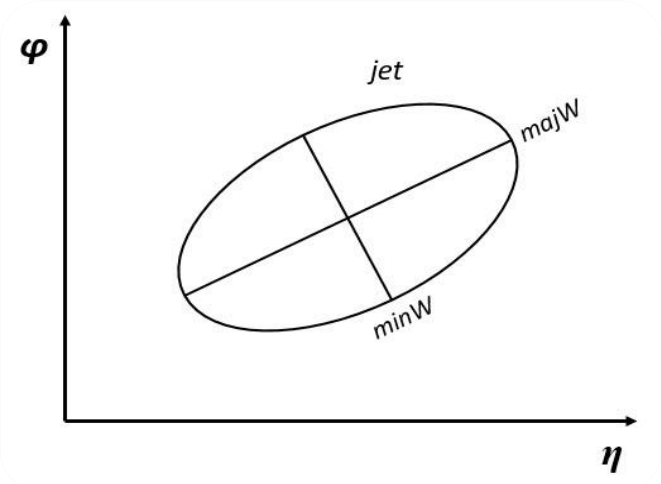
οπου j το φορτισμένο στοιχείο με την μεγαλύτερη εγκάρσια ορμή και επίσης το i να τρέχει πάνω σε όλα τα στοιχεία του jet.

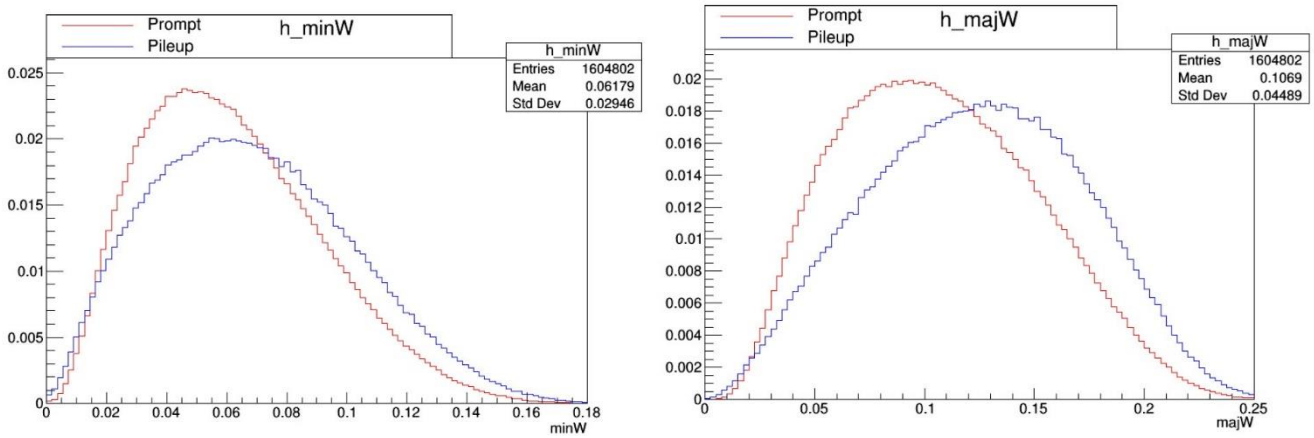


Απεικόνιση της μεταβλητής jetRchg στην περιοχή 0< η< 2,5 με το λογισμικό ROOT.

2.11 – 2.12. Μεταβλητές “minW” και “majW”

Όπως έχει προαναφερθεί το jet έχει την μορφή ενός κυκλικού κώνου με κορυφή το σημείο της πρωταρχικής σύγκρουσης (prompt jet) ή κάποιας δευτερεύουσας σύγκρουσης (pileup jet) και βάση την προβολή του στον κύλινδρο του ανιχνευτή, που λόγω της γεωμετρίας στο ανάπτυγμα της παράπλευρης επιφάνειας προκύπτει έλλειψη. Αφού λοιπόν στο ανάπτυγμα της παράπλευρης επιφάνειας του κυλίνδρου το jet προβάλλεται ως έλλειψη, θα πρέπει να ορισθούν δύο άξονες που να την προσδιορίζουν. Ετσι έχουμε τον μεγάλο άξονα της έλλειψης majW και τον μικρό άξονα της έλλειψης minW, όπως βλέπουμε και στην δίπλα εικόνα.



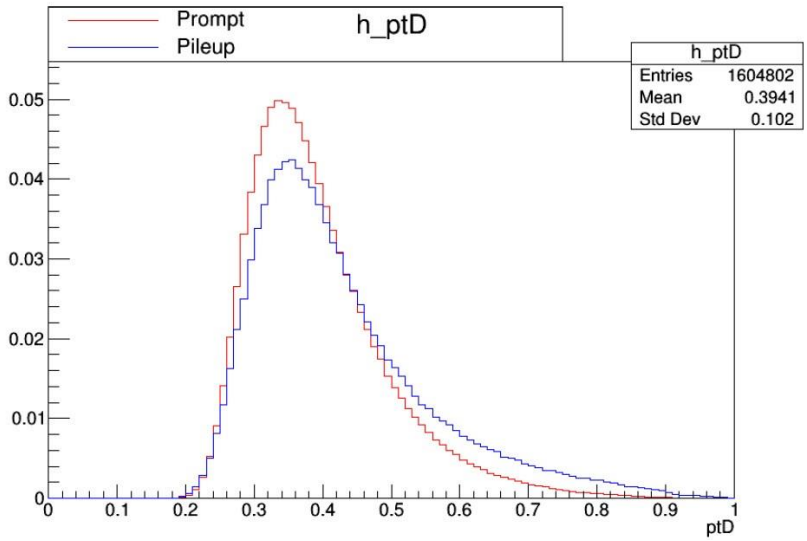


Απεικόνιση των μεταβλητών minW και majW στην περιοχή 0< η< 2,5 με το λογισμικό ROOT.

2.13. Μεταβλητή “ptD”

Η μεταβλητή ptD ορίζεται ως η τετραγωνική ρίζα του αθροίσματος του τετραγώνου της εγκάρσιας ορμής κάθε σωματιδίου, προς το συνολικό άθροισμα της εγκάρσιας ορμής για όλα τα στοιχεία του jet.

$$ptD = \frac{\sqrt{\sum_i P_{Ti}^2}}{\sum_i P_{Ti}}$$

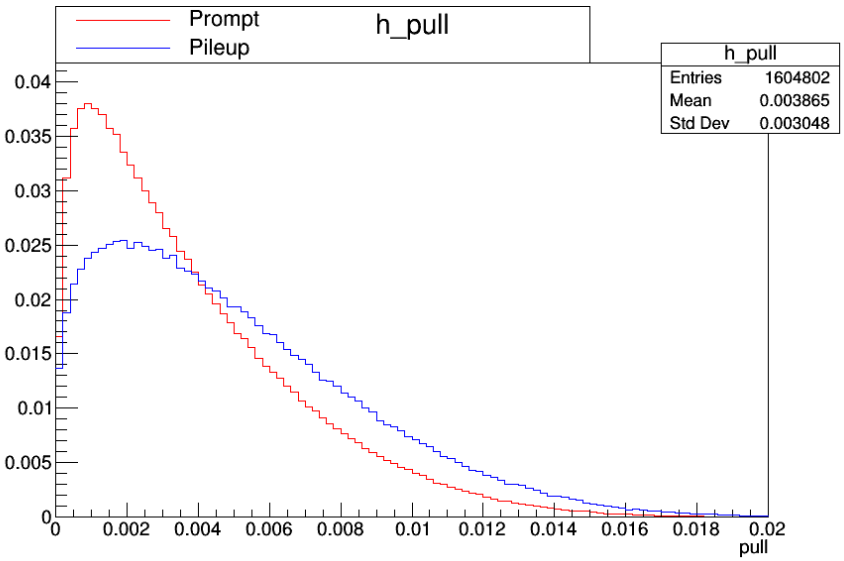


Απεικόνιση της μεταβλητής ptD στην περιοχή 0< η< 2,5 με το λογισμικό ROOT.

2.14. Μεταβλητή “pull”

Τέλος η μεταβλητή pull ορίζεται ως η ρίζα του αθροίσματος των τετραγώνων των μέσων τιμών των Δη και Δφ, δηλαδή:

$$\text{pull} = \sqrt{<\Delta\eta>^2 + <\Delta\varphi>^2}$$



Απεικόνιση της μεταβλητής pull στην περιοχή 0< η< 2,5 με το λογισμικό ROOT

ΚΕΦΑΛΑΙΟ 3
ΜΑΘΗΜΑΤΙΚΗ ΘΕΩΡΙΑ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

3.1 ΑΠΛΟΣ ΓΡΑΜΜΙΚΟΣ ΤΑΞΙΝΟΜΗΤΗΣ
(linear discriminant)

Στην απλή περίπτωση όπου έχουμε δυο κλάσεις C_1 και C_2 θεωρούμε μια γραμμική συνάρτηση των διανυσμάτων εισαγωγής **x** (input vectors) ως εξής:

$$y(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + w_o \quad (1)$$

με **w** το διάνυσμα από τα βάρη (weight vector) και w_o η μεροληψία (bias). Έτσι λοιπόν το **x** διάνυσμα εκχωρείται στην κλάση C_1 εάν $y(\mathbf{x}) \geq 0$ αλλιώς στην κλάση C_2. Το αντίστοιχο όριο απόφασης ορίζεται από την σχέση:

$$y(\mathbf{x}) = 0 \quad (2)$$

η οποία αποτελεί ένα (D-1) διαστάσεων υπερεπίπεδο , όπου D η διάσταση του χώρου που ορίζουν τα διανύσματα εισαγωγής (input space).

Θεωρώντας δυο σημεία $\mathbf{x_A}$, $\mathbf{x_B}$ που βρίσκονται στην επιφάνεια απόφασης, τα οποία θα πρέπει να επαληθεύουν την (2), παρατηρούμε:

$$y(\mathbf{x_A}) = y(\mathbf{x_B}) = 0 \Rightarrow \mathbf{w}^T \cdot (\mathbf{x_A} - \mathbf{x_B}) = 0 \quad (3)$$

Ως εκ τούτου το διάνυσμα **w** είναι ορθογώνιο με κάθε διάνυσμα που εφάπτεται στην επιφάνεια απόφασης. Συνεπώς το **w** ορίζει τον προσανατολισμό της επιφάνειας απόφασης και όπως θα δούμε στην συνέχεια το w_0 την θέση. Έτσι για ένα τυχαίο σημείο **x** πάνω στην επιφάνεια ισχύει πως:

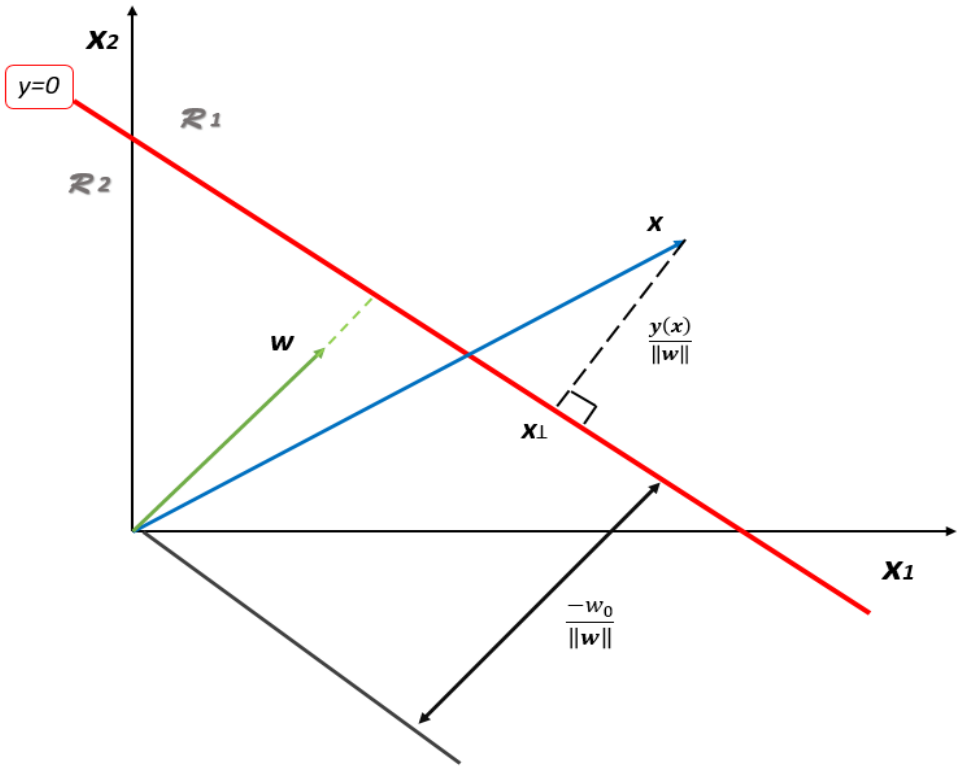
$$\frac{\mathbf{w}^T \cdot \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|} \quad (4)$$

Συνοψίζοντας για ένα αυθαίρετο σημείο **x** με $\mathbf{x}_\perp$ την ορθογώνια προβολή του πάνω στην επιφάνεια απόφασης έχουμε:

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \Rightarrow \mathbf{x} \cdot \mathbf{w}^T + w_o = \mathbf{x}_\perp \cdot \mathbf{w}^T + r\frac{\mathbf{w} \cdot \mathbf{w}^T}{\|\mathbf{w}\|} + w_o \Rightarrow$$

$$\Rightarrow y(\mathbf{x}) = y(\mathbf{x}_\perp) + r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} \Rightarrow r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|} \quad (5)$$

*(r -> η κάθετη απόσταση του **x** από την επιφάνεια απόφασης, βλ. εικόνα.)*


x2
y=0
R1
R2
x
w
y(x)/||w||
x⊥
x1
−w0/||w||

3.2 ΜΕΘΟΔΟΣ FISHER
(Fisher's linear discriminant analysis)

Ένας τρόπος να δούμε τους γραμμικούς ταξινομητές είναι ως μοντέλα μείωσης διαστάσεων. Έτσι έχοντας ένα D- διαστατο χώρο που ορίζεται από τα διανύσματα εισαγωγής (input space), γίνεται να τον προβάλουμε σε μια διάσταση (IR^D -> IR) με την χρήση της:

$$y(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} \tag{6}$$

Εάν βάλουμε ένα όριο στο y και ορίσουμε ότι σε περίπτωση που $y(\mathbf{x}) \geq -w_o$ το **x** ανήκει στην κλάση C_1 ,αλλιώς στην κλάση C_2 ,όπως με τον απλό γραμμικό ταξινομητή. Φυσικά η προβολή σε μια διάσταση έχει ως αποτέλεσμα σημαντική απώλεια πληροφορίας, αφού οι κλάσεις που σε D διαστάσεις ήταν καλώς διαχωρισμένες, ενώ στη μια υπάρχει περίπτωση επικαλύψεων. Παρ'ολα αυτά με την κατάλληλη προσαρμογή του **w, γίνεται** να διαλέξουμε την προβολή που μεγιστοποιεί τον διαχωρισμό.

Θεωρώντας λοιπόν 2 κλάσεις C_1 και C_2, με N_1 και N_2 διανύσματα αντίστοιχα η κάθε μια, έχουμε τα μέσα διανύσματα μ_1 και μ_2 τα οποία είναι:

$$\boldsymbol{\mu_1} = \frac{1}{N1} \sum \mathbf{x_n} \qquad \boldsymbol{\mu_2} = \frac{1}{N2} \sum \mathbf{x_n} \tag{7}$$

Ο απλούστερος τρόπος να ξεχωρίσουμε τις κλάσεις που έχουμε προβάλει πάνω σε μια διάσταση, είναι να μεγιστοποιήσουμε την διαφορά μεταξύ των μέσων διανυσμάτων τους, $|\mu_1 - \mu_2|$-> max άρα:

$$\mu_1 - \mu_2 = \mathbf{w}^T \cdot (\boldsymbol{\mu_1} - \boldsymbol{\mu_2}) \tag{8}$$

$$\mu_i = \mathbf{w}^T \cdot \boldsymbol{\mu_i} \tag{9}$$

Με μ_i τη μέση τιμή που ανήκει στην κλάση C_i . Η ιδέα που πρότεινε ο Fisher ήταν να μεγιστοποιήσουμε μια συνάρτηση η οποία θα δίνει την μέγιστη απόσταση μεταξύ των μέσων τιμών μετά την προβολή καθώς και την ελάχιστη διασπορά κάθε κλάσης γύρω από τις μέσες τιμές. Εδώ θα πρέπει να δηλώσουμε πως διασπορά κάθε κλάσης των διανυσμάτων που έχουν προβληθεί σε μια διάσταση είναι:

$$\boldsymbol{\sigma_i}^2 = \sum (y_n - \mu_i)^2 \tag{10}$$

Επίσης ορίζουμε την συνολική διασπορά κάθε κλάσης για όλο το σύνολο δεδομένων ως : $(\sigma_1^2+\sigma_2^2)$. Το κριτήριο του Fisher προκύπτει από το πηλίκο της μεταξύ των κλάσεων διασπορα προς την συνολική διασπορά, δηλαδή:

$$J(\mathbf{w}) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \quad \textit{(Fisher criterion)} \qquad (11)$$

Ακόμα αν ορίσουμε $\mathbf{S}_B$ την διασπορά μεταξύ των κλάσεων και $\mathbf{S}_W$ την συνολική διασπορά εντός των κλάσεων το παραπάνω κριτήριο γίνεται:

$$J(\mathbf{w}) = \frac{(\mathbf{w})^T \cdot \mathbf{S}_B \cdot \mathbf{w}}{(\mathbf{w})^T \cdot \mathbf{S}_W \cdot \mathbf{w}} \qquad (12)$$

$$(\mu_1-\mu_2)^2=[\mathbf{w}^T\cdot(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)]\cdot[\,\mathbf{w}^T\cdot(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)]^T=\mathbf{w}^T\cdot[\,(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\cdot(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T]\cdot\mathbf{w}=\mathbf{w}^T\,\mathbf{S_B}\,\mathbf{w} \qquad (13)$$

$$\sigma_i^2=\sum(y-\mu_i)^2=\sum|\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_i)|^2=\sum\,[\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_i)]\cdot[\,\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_i)]^T=$$

$$=\sum\,\mathbf{w}^T(\mathbf{x}-\boldsymbol{\mu}_i)\cdot(\mathbf{x}-\boldsymbol{\mu}_i)^T\mathbf{w}=\mathbf{w}^T\cdot[\,\sum(\mathbf{x}-\boldsymbol{\mu}_i)\cdot(\mathbf{x}-\boldsymbol{\mu}_i)^T\,]\cdot\mathbf{w} \qquad (14)$$

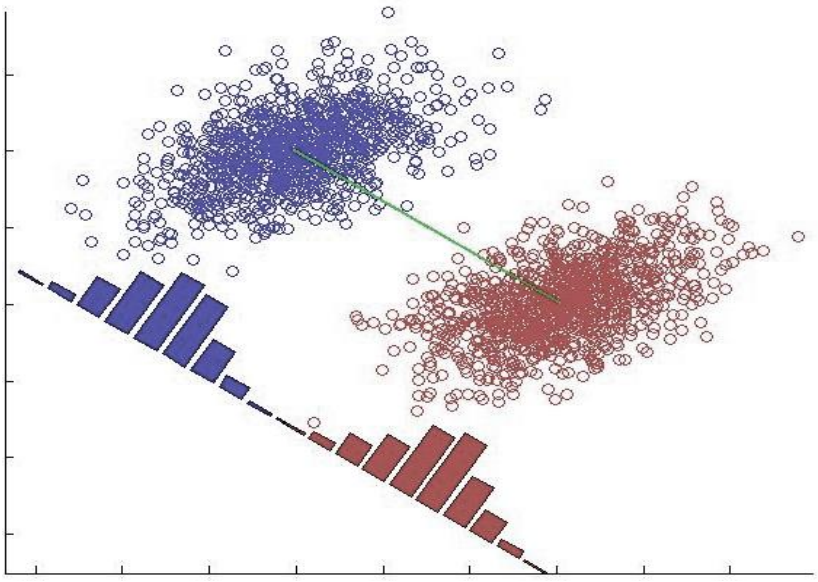
$$\mathbf{S}_w=\sum(\mathbf{x}-\boldsymbol{\mu}_1)\cdot(\mathbf{x}-\boldsymbol{\mu}_1)^T+\sum(\mathbf{x}-\boldsymbol{\mu}_2)\cdot(\mathbf{x}-\boldsymbol{\mu}_2)^T \qquad (15)$$

Επειδή όπως προαναφέρθηκε θέλουμε να μεγιστοποιήσουμε την συνάρτηση προκύπτει:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}=0 \;\Rightarrow$$

$$\Rightarrow (\mathbf{w}^T\cdot\mathbf{S}_B\cdot\mathbf{w})\cdot\mathbf{S}_w\cdot\mathbf{w}=(\mathbf{w}^T\cdot\mathbf{S}_w\cdot\mathbf{w})\cdot\mathbf{S}_B\cdot\mathbf{w} \;\Rightarrow$$

$$\Rightarrow \quad w\sim\mathbf{S}_w^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) \qquad (16)$$



3.3 ΜΕΘΟΔΟΣ ΕΛΑΧΙΣΤΩΝ ΤΕΤΡΑΓΩΝΩΝ ΓΙΑ ΤΑΞΙΝΟΜΗΣΗ (least square method)

Θεωρώντας ένα γενικό πρόβλημα ταξινόμησης με K κλάσεις και D διαστάσεων χώρο που ορίζουν τα δεδομένα εισαγωγής (input space), με κάθε κλάση C_k να περιγράφεται από το δικό της γραμμικό μοντέλο έτσι ώστε:

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \cdot \mathbf{x} + w_{ko} \quad , k= 1,\ldots,K \tag{17}$$

Για να συμβολίσουμε όλες τις κλάσεις μαζί κάνουμε χρήση του συμβολισμού:

$$y(\mathbf{x}) = \widetilde{\mathbf{W}}^T \cdot \tilde{\mathbf{x}} \tag{18}$$

με $\widetilde{W}$ ένα πίνακα όπου η k-οστη στήλη του περιέχει το D+1 διαστάσεων διάνυσμα $\widetilde{w}_k = (w_{ko}, \mathbf{w}_k^T)^T$ και $\tilde{\mathbf{x}}$ το αντίστοιχο αλλαγμένο διάνυσμα εισόδου $(x_o, \mathbf{x}^T)^T$ με αυθαίρετη εισαγωγή $x_o = 1$.

Κάνοντας χρήση ενός σύνολου δεδομένων $\{\mathbf{x_n}, \mathbf{t_n}\}$ με n= 1,…,N και N το πλήθος των δεδομένων εισαγωγής (inputs) ορίζουμε τον πίνακα **T** του οποίου η n-ιοστη γραμμή είναι το διάνυσμα $\mathbf{t}_n^T$. Έτσι μαζί με ένα πίνακα $\widetilde{\mathbf{X}}$ του οποίου η n-ιοστη γραμμή είναι το διάνυσμα $\tilde{\mathbf{x}}_n^T$ είμαστε σε θέση να ορίσουμε την συνάρτηση κόστους του αθροίσματος των τετράγωνων ως εξής:

$$J(\widetilde{\mathbf{W}}) = \frac{1}{2} Tr\{ (\widetilde{\mathbf{X}} \cdot \widetilde{\mathbf{W}} - \mathbf{T})^T \cdot (\widetilde{\mathbf{X}} \cdot \widetilde{\mathbf{W}} - \mathbf{T}) \} \tag{19}$$

Προφανώς ο σκοπός μας είναι να ελαχιστοποιήσουμε την συνάρτηση κόστους, άρα θέλουμε:

$$\frac{\partial J(\widetilde{\mathbf{W}})}{\partial \widetilde{\mathbf{W}}} = 0 \Rightarrow \widetilde{W} = (\widetilde{\mathbf{X}}^T \cdot \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \cdot \mathbf{T} \tag{20}$$

Στο σημείο αυτό ορίζουμε τον ψευδο-αντιστροφο πίνακα του $= \widetilde{\mathbf{X}}$ τον οποίο θα συμβολίζουμε ως $\widetilde{\mathbf{X}}^+$ και θα είναι ίσος με: $\widetilde{\mathbf{X}}^+ = (\widetilde{\mathbf{X}}^T \cdot \widetilde{\mathbf{X}})^{-1} \cdot \widetilde{\mathbf{X}}^T$

Συνεπώς η εξίσωση (4) γίνεται : $\widetilde{\mathbf{W}} = \widetilde{\mathbf{X}}^+ \cdot \mathbf{T}$ (21)

Έτσι η συνάρτηση διαχωρισμού μπορεί να γραφτεί ως εξής:

$$y(\mathbf{x}) = \widetilde{\mathbf{W}}^T \cdot \tilde{\mathbf{x}} = \mathbf{T}^T \cdot (\widetilde{\mathbf{X}}^+)^T \cdot \tilde{\mathbf{x}} \tag{22}$$

> Μια σημαντική ιδιότητα των λύσεων της μεθόδου αυτής είναι ότι εάν για κάθε target vector ισχύει πως $\mathbf{a}^T \cdot \mathbf{t_n} + b = 0$, με a,b σταθερές τότε για κάθε τιμή του **x** με τα ίδια a,b το μοντέλο πρόβλεψης θα είναι:

$$\mathbf{a}^T \cdot y(\mathbf{x}) + b = 0 \tag{23}$$

3.4 ΜΕΘΟΔΟΣ Κ ΠΛΗΣΙΕΣΤΕΡΩΝ ΓΕΙΤΟΝΩΝ kNN (k Nearest Neighbors)

(kernel density estimators)

Σε μια μικρή περιοχή R, η συνάρτηση μάζας πιθανότητας που σχετίζεται με την περιοχή είναι :

$$P = \int_R p(x)\, dx \tag{24}$$

Έχοντας λοιπόν ένα σύνολο δεδομένων N παρατηρήσεων που περιγράφονται από την p(x), ξέρουμε ότι κάθε δεδομένο έχει πιθανότητα P να περιέχεται μέσα στην περιοχή R. Έτσι ο συνολικός αριθμός των δεδομένων K που θα περιέχεται στην περιοχή R περιγράφεται από την διωνυμική κατανομή:

$$B(K|N,P)=\frac{N!}{N!(N-K)!}\, P^K (1-P)^{N-K} \tag{25}$$

Η συνάρτηση μέσης τιμής για τα δεδομένα της περιοχής R καθώς και η διασποράς τους δίνεται από:

$$E(K|N)=P \qquad\qquad Var[K|N] = \frac{P(1-P)}{N}$$

Για μεγάλα N όμως K-> N·P και εάν η περιοχή R είναι μικρή τότε P=p(x)·V, με V τον όγκο της περιοχής και p(x) την τοπική εκτίμηση πυκνότητας. Συνεπώς από τα παραπάνω έχουμε ότι:

$$p(x)= \frac{K}{N\cdot V} \tag{26}$$

Αντί να παίρνουμε ένα σταθερό V και να αλλάζουμε συνεχώς την τιμή του K, είναι χρησιμότερο να βρίσκουμε κάθε φορά τον κατάλληλο όγκο V για δεδομένο αριθμό στοιχείων K. Συνεπώς θέτουμε μια μικρή σφαίρα με κέντρο ένα σημείο x στο οποίο υπολογίζουμε την πυκνότητα πιθανότητας p(x) και ρυθμίζουμε την ακτίνα της σφαίρας ώστε να περιέχει ακριβώς K στοιχεία.

(k Nearest Neighbors)

Στη μέθοδο αυτή η τιμή του K ρυθμίζει την εξομάλυνση και φυσικά υπάρχει ένα βέλτιστο K ώστε να μην είναι ούτε πολύ μεγάλο , ούτε πολύ μικρο. Έτσι στα προβλήματα ταξινόμησης εφαρμόζουμε την τεχνική αυτή ξεχωριστά σε κάθε κλάση και μετά κάνουμε χρήση του θεωρήματος του Bayes. Θεωρούμε λοιπόν ένα σύνολο δεδομένων N_k σημείων στην κλάση C_k με N σημεία συνολικά, ώστε$\sum_k N_k$=N. Αν θελήσουμε να ταξινομήσουμε ένα νέο σημείο x,

σχεδιάζουμε μια σφαίρα με κέντρο το x περιέχοντας ακριβώς K σημεία ανεξάρτητα της κλάσης τους. Η σφαίρα με όγκο V περιέχει K_k σημεια από την κλάση C_k και από την (26) έχουμε την εκτίμηση της πυκνότητας πιθανότητας δεσμευμένη με την κλάση k (posterior probability) ως εξης:

$$p\,(x|C_k) = \frac{K_k}{N_k \cdot V} \tag{27}$$

Καθώς επίσης η τοπική εκτίμηση πυκνότητας είναι:

$$p\,(x) = \frac{K}{N \cdot V} \tag{28}$$

και η πιθανότητα να ανήκει ένα στοιχείο στην κλάση k (prior probability):

$$p\,(C_k) = \frac{N_k}{N} \tag{29}$$

Συνδυάζοντας τις τρεις παραπάνω εξισώσεις στο θεώρημα του Bayes:

$$p\,(C_k|x) = \frac{p(x \vee C_k)\, p(C_k)}{p(x)} = \frac{K_k}{K} \tag{30}$$

Εάν θέλουμε να ελαχιστοποιήσουμε την πιθανότητα της λάθος ταξινόμησης, θα πρέπει να θέσουμε το τεστ σημείο x στην κλάση που έχει την μεγαλύτερη posterior πιθανότητα, που αντιστοιχεί στην μεγαλύτερη τιμή $K_κ/K$. Έτσι για να ταξινομήσουμε ένα νέο σημείο, βρίσκουμε τους K πλησιέστερους γείτονες από το σύνολο εκπαίδευσης και αναθέτουμε το νέο σημείο στην κλάση που έχει τους περισσότερους αντιπρόσωπους στο πλήθος των K γειτόνων. Στην περίπτωση του K=1 έχουμε τον κανόνα του πλησιέστερου γείτονα όπου το νέο στοιχείο x πηγαίνει στην κλάση του κοντινότερου γείτονα. Μια ενδιαφέρουσα ιδιότητα της περίπτωσης όπου το K=1, είναι ότι το ποσοστό σφάλματος ακόμα και όταν το N-> ∞, δεν μπορεί να ξεπεράσει το διπλάσιο του ελαχίστου που επιτυγχάνεται για τον βέλτιστο ταξινομητή.

Η KNN μέθοδος απαιτεί να σκαναριστούν όλα τα δεδομένα, πράγμα που αποτελεί υπολογιστική σπάταλη και μειονέκτημα σε μεγάλα σύνολα δεδομένων.

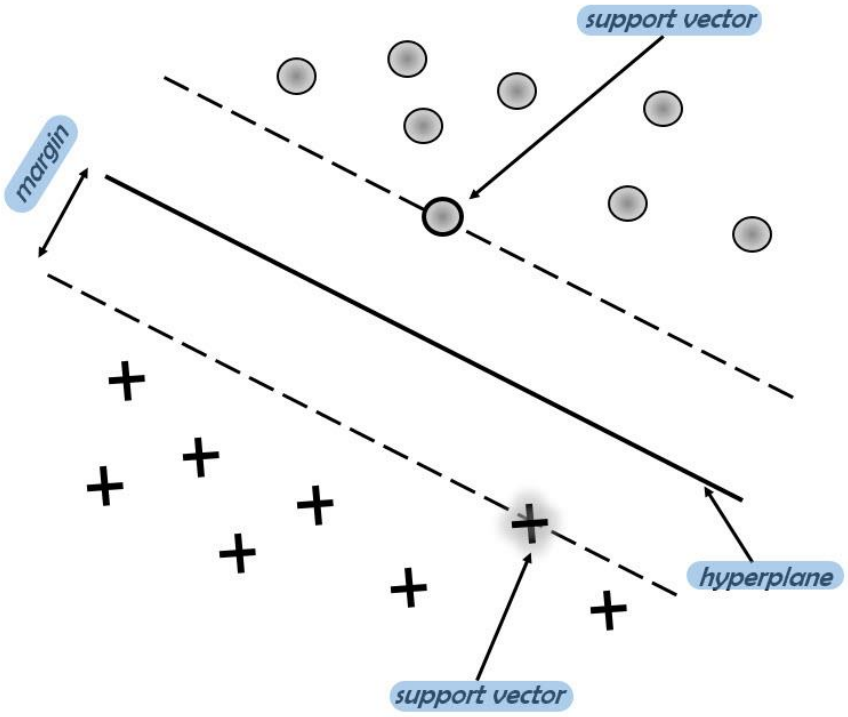
3.5 ΜΗΧΑΝΕΣ ΔΙΑΝΥΣΜΑΤΙΚΗΣ ΣΤΗΡΙΞΗΣ
(Support Vector machines – SVMs)

Τα SVMs αποτελούν μία εναλλακτική λογική για τον σχεδιασμό γραμμικών ταξινομητών. Στην απλή περίπτωση των δύο κλάσεων, έχοντας $\mathrm{x_i}, \mathrm{i} = 1,2 \ldots., \mathrm{N}$ τα διανύσματα χαρακτηριστικών του συνόλου εκπαίδευσης X. Καθένα από αυτά ανήκει σε κάποια από τις δύο διαχωρίσιμες κλάσεις w_1 ,w_2. Ο στόχος του ταξινομητή είναι να βρεθεί η επιφάνεια η οποία θα είναι μοναδική και θα έχει την μέγιστη απόσταση (κάθετη) και από τις δύο κλάσεις. Άρα η επιφάνεια (hyperplane) θα πρέπει να είναι της μορφής:

$$\mathrm{g}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{w}_0 = 0 \tag{31}$$

Για να βρούμε την ζητούμενη επιφάνεια πρώτα κατασκευάζουμε μια ζώνη, τα άκρα της οποίας τέμνουν την κλάση. Τα (support vectors) διανύσματα στήριξης είναι τα στοιχεία που βρίσκονται πάνω στις 2 ευθείες που ορίζουν την ζώνη διαχωρισμού. Είναι τα μόνα στοιχεία των κλάσεων που μας ενδιαφέρουν και μας καθορίζουν την τελική λύση.

Το περιθώριο (margin) που αφήνει η επιφάνεια από κάθε κλάση , θέλουμε να το μεγιστοποιήσουμε. Οπότε είναι λογικό να ψάξουμε την διεύθυνση που θα δίνει το μεγαλύτερο δυνατό περιθώριο.



Ποσοτικοποιώντας το περιθώριο έχουμε:

$$\text{margin} = 1/\|\mathbf{w}\| \tag{32}$$

Επειδή απαιτούμε :

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{w_o} \geq 1 \forall \mathrm{x} \in \omega 1 \tag{33}$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{w_o} \leq -1 \forall \mathrm{x} \in \omega 2 \tag{34}$$

μπορούμε να ορίσουμε τον αντίστοιχο δείκτη της κλάσης για κάθε $\mathrm{x_i}$ ώστε $\mathrm{y_i}$=+1 ή -1. Το ισοδύναμο πρόβλημα , ορίζοντας παράλληλα την συνάρτηση κόστους γίνεται:

$$\mathrm{J}(\mathbf{w}, \mathrm{w}_0) = \frac{1}{2}\|\mathrm{w}\|^2 \tag{35}$$

$$\mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + \mathrm{w}_0) \geq 1, \mathrm{i} = 1,2, \dots, \mathrm{N} \tag{36}$$

Προφανώς επειδή η ελαχιστοποίηση της νόρμας οδηγεί στην μεγιστοποίηση του περιθωρίου, θέλουμε να ελαχιστοποιήσουμε την συνάρτηση κόστους. Λόγω των περιορισμών εφαρμόζουμε πολλαπλασιαστές Lagrange ως εξής:

$$\mathrm{L}(\mathbf{w}, \mathrm{w}_0, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} - \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} \tag{37}$$

Για να έχουμε λύση λαμβάνουμε υπόψιν τους περιορισμούς. Θέλουμε:

$$\frac{\partial \mathrm{L}}{\partial \mathbf{w}} = 0 \tag{38}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{w}_0} = 0 \tag{39}$$

$$\lambda_{\mathrm{i}} \geq 0 \tag{40}$$

$$\lambda_{\mathrm{i}}[\mathrm{y_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + \mathrm{w}_0) - 1] = 0 \text{ , με} \mathrm{i} = 1,2, \dots, \mathrm{N} \tag{41}$$

όπου N: αριθμός διανυσμάτων εκπαίδευσης.

Τα στοιχεία λοιπόν που θα έχουν πολλαπλασιαστή Lagrange $\lambda_{\mathrm{i}} = 0$, θα είναι αυτά που δεν ανήκουν πάνω στις ευθείες και όπως φαίνεται στις εξισώσεις δεν επηρεάζουν την λύση μας.

Για λi≠0 έχουμε διανύσματα στήριξης $\mathbf{x}_i$ (support vectors), τα οποία μας δίνουν μοναδική λύση:

$$\mathbf{w} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_i y_i \mathbf{x}_i \tag{42}$$

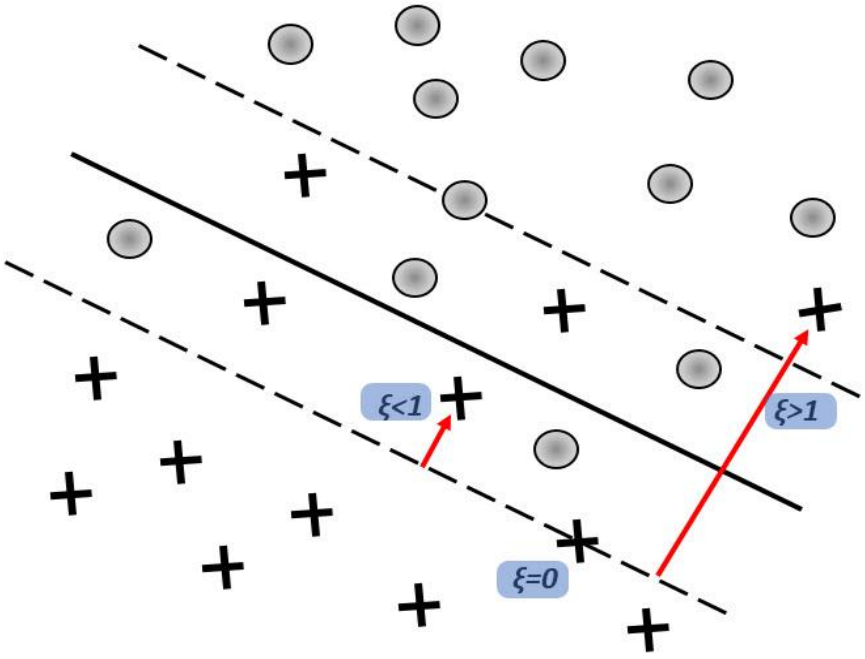
Στην περίπτωση που οι δυο κλάσεις είναι μη γραμμικώς διαχωρίσιμες έχουμε:

- Σημεία εκτός της ζώνης που είναι σωστά ταξινομημένα:
  $y_i(\mathbf{w}^T\mathbf{x}+w_0)\geq 1$
- Σημεία εντός της ζώνης που είναι σωστά ταξινομημένα:
  $0<y_i(\mathbf{w}^T\mathbf{x}+w_0)<1$
- Σημεία λάθος ταξινομημένα $y_i(\mathbf{w}^T\mathbf{x}+w_0)<0$

Για τις τρεις παραπάνω περιπτώσεις οι ανισώσεις μπορούν να γραφτούν σε μια μορφή ως εξής:

$$y_i \cdot(\mathbf{w}^T\mathbf{x}+w_0) \geq 1-\xi_i \quad (43)$$

όπου η ξ_i λέγεται χαλαρή μεταβλητή (slack variable) και σε κάθε μια από τις τρεις περιπτώσεις παίρνει τιμές $\xi_i=0$, $0\leq \xi_i \leq 1$ και $\xi_i >1$ αντίστοιχα.



Συνεπώς από τα παραπάνω μπορούμε να ορίσουμε μια συνάρτηση κόστους:

$$J = \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{N} I(\xi_i) \quad (44)$$

με C μια θετική σταθερά και $I(\xi_i) = \begin{cases} 1, & \xi > 1 \\ 0, & \xi \leq 1 \end{cases}$ (45)

Από την παραπάνω συνάρτηση κόστους σε συνδυασμό με την σχέση (43) προκύπτει μια λύση της μορφής:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \quad (46)$$

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T\mathbf{x} + w_0 \quad (47)$$

Γενικεύοντας έχουμε:

$$g(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}_i, \mathbf{z}) \quad (48)$$

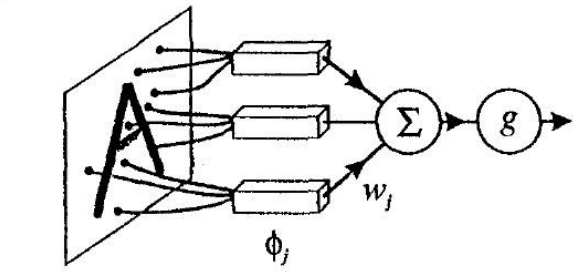
με K (kernel) συνάρτηση πυρήνα που μπορεί να είναι:

- Πολυωνυμική (polynomial) -> $(\mathbf{x}_i^T\mathbf{z}+1)^q$
- Ακτινωτής βάσης (radial basis function) -> $\exp(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2})$
- Υπερβολική εφαπτομένη (hyperbolic tangent) -> $\tanh(\beta \mathbf{x}_i^T\mathbf{z}+\gamma)$

[3]

3.6 ΜΟΝΟΣΤΡΩΜΑΤΙΚΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ
(The perceptron algorithm)

Νευρωνικά δίκτυα με ένα στρώμα και συνάρτηση ενεργοποίησης εισόδου, μελετήθηκαν για πρώτη φορά από τον Rosenblatt το 1962, ο οποίος τα αποκαλούσε perceptrons. Είναι γνωστό πως ένα νευρωνικό δίκτυο με μία στρώση νευρώνων έχει περιορισμένες δυνατότητες. Για να βελτιωθεί η επίδοση του perceptron, ο Rosenblatt έκανε χρήση μιας στρώσης προσαρμοσμένων δεδομένων (ϕ_j) που προέκυπταν από την μετατροπή των ακατέργαστων δεδομένων εισαγωγής.



Η έξοδος του perceptron δίνεται από :

$$y = g\left(\sum_{j=0}^{M} w_j \phi_j(\mathbf{x})\right) = g(\mathbf{w}^T\boldsymbol{\phi}) \qquad (49)$$

όπου το ($\boldsymbol{\phi}$) δηλώνει το διάνυσμα των προσαρμοσμένων δεδομένων από την συνάρτηση ενεργοποίησης εισόδου.

Η συνάρτηση ενεργοποίησης εξόδου έχει επιλεγεί για λόγους ευκολίας να είναι η αντισυμμετρική της συνάρτησης ενεργοποίησης εισόδου, είναι δηλαδή της μορφής:

$$g(x) = \begin{cases} -1, & x < 0 \\ +1, & x \geq 0 \end{cases}$$

Συνεπώς ο σκοπός είναι να δημιουργήσουμε ένα αποτελεσματικό σύστημα ταξινόμησης και γι αυτό θα πρέπει να οριστεί μια συνάρτηση σφάλματος που να προσδιορίζει τον αριθμό των λάθος ταξινομήσεων σε σχέση με το ολικό σύνολο εκπαίδευσης. Έχοντας λοιπόν $\mathbf{x}^n$ το διάνυσμα εισόδου, $\boldsymbol{\phi}^n$ το επεξεργασμένο διάνυσμα εισόδου και $\mathbf{t}^n$ το target vector τέτοιο ώστε:

$$\mathbf{t}^n = \begin{cases} -1, \ \text{για } x_j \in C1 \\ +1, \ \text{για } x_j \in C2 \end{cases}$$

Τότε θέλουμε:

$\mathbf{w}^T\boldsymbol{\phi}^n > 0$ για την κλάση C_1

και $\mathbf{w}^T\boldsymbol{\phi}^n < 0$ για την κλάση C_2

Άρα συνδυάζοντας τις παραπάνω δυο σχέσεις θέλουμε :

$$\mathbf{w}^T(\boldsymbol{\phi}^n t^n) > 0 \tag{50}$$

Κάπως έτσι λοιπόν προκύπτει η συνάρτηση σφάλματος ή αλλιώς κόστους, την οποία θέλουμε να ελαχιστοποιήσουμε, γνωστή ως perceptron criterion:

$$J(\mathbf{w}) = \sum_{\varphi \in M} w^T(\boldsymbol{\phi}^n t^n) \tag{51}$$

οπου το M είναι το σύνολο διανυσμάτων $\boldsymbol{\phi}^n$ τα οποία ταξινομήθηκαν λάθος από το διάνυσμα **w**

(perceptron learning)

Ο τρόπος εκμάθησης του perceptron δηλαδή ο τρόπος ελαχιστοποίησης της συνάρτησης κόστους με τελικό αποτέλεσμα την επαναπροσαρμογή των βαρών είναι:

$$J(\mathbf{w}) = \sum_n J^n(\mathbf{w}) \Rightarrow$$

$$w_j^{(\tau+1)} = w_j^{(\tau)} - \eta \cdot \frac{\partial \mathbf{J}^n}{\partial w_j} \Big|_{w^{(\tau)}} = 0 \tag{52}$$

όπου “η” η παράμετρος εκμάθησης
η οποία είναι πάντα θετικός αριθμός

Κάτω από συγκεκριμένες συνθήκες η τιμή του “η” συγκλίνει σε ένα σημείο στο οποίο η J ελαχιστοποιείται. Αρκετές φορές το “η” το ρυθμίζουμε να αλλάζει με τον χρόνο ως εξής:

$$\eta^{(\tau)} = \eta_0 / \tau \tag{53}$$

Συνεπώς λαμβάνοντας υπόψιν τον ορισμό της συνάρτησης κόστους προκύπτει ο ακόλουθος κανόνας (gradient descent rule):

$$w_j^{(\tau+1)} = w_j^{(\tau)} - \eta \cdot (\phi_j^n t^n) \tag{54}$$

Θεώρημα σύγκλισης του perceptron.

Για κάθε σύνολο δεδομένων το οποίο είναι γραμμικά διαχωρίσιμο, ο κανόνας εκμάθησης (gradient descent rule) μας εγγυάται πως θα βρεθεί λύση για πεπερασμένο αριθμό βημάτων.

Απόδειξη (Hertz et al. 1991):

Εάν το σύνολο δεδομένων είναι γραμμικά διαχωρίσιμο τότε υπάρχει τουλάχιστον ένα $\widehat{\mathbf{w}}$, για το οποίο όλα τα διανύσματα εκπαίδευσης είναι σωστά ταξινομημένα, ώστε:

$$\widehat{\mathbf{w}}^{T}\boldsymbol{\phi}^{n}t^{n}>0 \text{ , για όλα τα n.} \tag{55}$$

Η διαδικασία εκμάθησης ξεκινάει με ένα τυχαίο διάνυσμα των βαρών, τα οποία χωρίς βλάβη της γενικότητας μπορούμε να θεωρήσουμε μηδέν. Σε κάθε βήμα του αλγορίθμου τα βάρη προσαρμόζονται χρησιμοποιώντας την σχέση:

$$\mathbf{w}^{(\tau+1)}=\mathbf{w}^{(\tau)} - \eta\cdot(\boldsymbol{\phi}^{n}t^{n}) \tag{56}$$

με $\boldsymbol{\phi}^{n}$ ένα διάνυσμα που ταξινομήθηκε λάθος από τον perceptron.

Αφού τρέξουμε τον αλγόριθμο για λίγη ώρα, ο αριθμός των φορών που το κάθε διάνυσμα $\boldsymbol{\phi}^{n}$ εμφανιζεται και ταξινομείται λάθος είναι τ^{n}. Σε αυτό το σημείο το διάνυσμα των βαρών δίνεται από τη σχέση:

$$\mathbf{w}= \sum_{n} \tau^{n} \boldsymbol{\phi}^{n} t^{n} \Rightarrow$$

$$\widehat{\mathbf{w}}^{T} \mathbf{w}= \sum_{n} \tau^{n} \widehat{\mathbf{w}}^{T} \boldsymbol{\phi}^{n} t^{n} \geq \tau \min(\widehat{\mathbf{w}}^{T} \boldsymbol{\phi}^{n} t^{n}) \tag{57}$$

οπου $\tau =\sum_{n} \tau^{n}$ ο συνολικος αριθμος των φορων που ανανεωθηκαν τα βαρη.

Από τον κανόνα εκμάθησης έχουμε:

$$\left\|\mathbf{w}^{(\tau+1)}\right\|^{2} = \left\|\mathbf{w}^{(\tau)}\right\|^{2} +\|\boldsymbol{\varphi}^{n}\|^{2}(t^{n})^{2} + 2\, \mathbf{w}^{(\tau)T} \boldsymbol{\phi}^{n}t^{n} \leq \left\|\mathbf{w}^{(\tau)}\right\|^{2} +\|\boldsymbol{\varphi}^{n}\|^{2}(t^{n})^{2}$$

Η ανισότητα προκύπτει από το γεγονός ότι το $\boldsymbol{\phi}^{n}$ πρέπει να εχει ταξινομηθεί λάθος ώστε $\mathbf{w}^{(\tau)T}\boldsymbol{\phi}^{n}t^{n}<0$. Επίσης το $(t^{n})^{2}=+1$ αφού $(t^{n})=\pm 1$ και$\|\boldsymbol{\varphi}^{n}\|^{2}\leq\|\boldsymbol{\varphi}\|^{2}_{max}$ οπου το $\|\boldsymbol{\varphi}\|^{2}_{max}$ ειναι το μήκος του μεγαλύτερου διανύσματος εισαγωγής.

Επιπρόσθετα: : $\Delta\|\mathbf{w}\|^{2}= \left\|\mathbf{w}^{(\tau+1)}\right\|^{2} - \left\|\mathbf{w}^{(\tau)}\right\|^{2} \leq \|\boldsymbol{\varphi}\|^{2}_{max}$

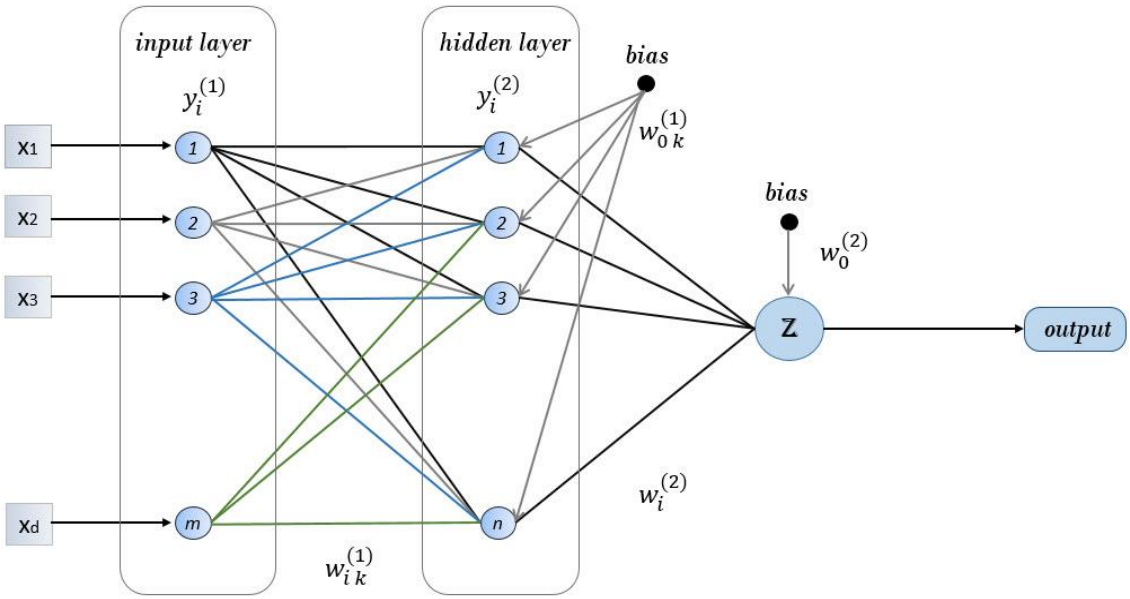
Άρα μετά από τ επαναπροσαρμογες των βαρών έχουμε:

$$\|\mathbf{w}\|^{2} \leq \tau\|\boldsymbol{\varphi}\|^{2}_{max} \tag{58}$$

Συνεπώς το μήκος $\|\mathbf{w}\|$ του διανύσματος των βαρών δεν αυξάνεται γρηγορότερα από $\tau^{1/2}$. Επίσης τώρα γνωρίζουμε πως το γινόμενο $\widehat{\mathbf{w}}^{T}\mathbf{w}$ είναι κάτω φραγμένο από μια γραμμική συνάρτηση του τ. Αφότου λοιπόν το τ δεν γίνεται να αυξάνεται επ' άπειρον, καθώς τότε τα αποτελέσματα μας γίνονται ασυμβίβαστα, θα πρέπει ο αλγόριθμος να συγκλίνει σε έναν πεπερασμένο αριθμό βημάτων. [1], [2]

3.7 ΠΟΛΥΣΤΡΩΜΑΤΙΚΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ (ARTIFICIAL NEURAL NETWORKS)

Τα νευρωνικά δίκτυα αποτελούν μια πολυστρωματική μορφή του perceptron. Στην βιβλιογραφία αναφέρονται ως artificial neural networks ή multilayer perceptrons ή απλα neural nets. Είναι ίσως από τις πιο ενδιαφέρουσες και ευρέως διαδεδομένες μεθόδους που εφαρμόζονται στην μηχανική μάθηση και ενας από τους λόγους είναι ότι αποτελούν μια προσομοίωση των νευρώνων του εγκεφάλου. Ετσι λοιπόν κάθε αναπαράσταση από ένα πλήθος διασυνδεδεμένων νευρώνων, όπου κάθε νευρώνας παράγει μια απόκριση στο δοθέν εισαγόμενο σήμα, μπορεί να θεωρηθεί νευρωνικό δίκτυο. Στην παρακάτω εικόνα βλέπουμε την δομή ενός νευρωνικού με μία κρυφή στρώση νευρώνων (hidden layer) και φυσικά την πρώτη φανερή στρώση (input layer). Το νευρωνικό δίκτυο της εικόνας έχει ροή προς τα εμπρός (Feedforward net) , πράγμα που σημαίνει πως δεν περιέχει βρόγχους (loops).



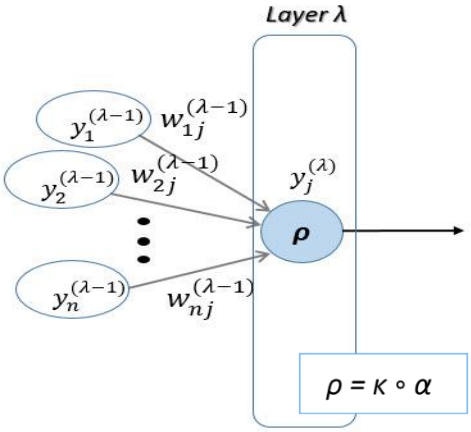
Συνεπώς τα δεδομένα εξέρχονται από τους νευρώνες – κόμβους της πρώτης στρώσης και αφου πολλαπλασιαστούν με τα κατάλληλα βάρη, εισέρχονται στους νευρώνες δεύτερης στρώσης (κρυφή στρώση). Εκεί επεξεργάζονται από την συνάρτηση σύναψης, προστίθεται ο συντελεστής μεροληψίας (bias) και επεξεργάζονται ξανά από την συνάρτηση ενεργοποίησης. Ετσι αφού δράσουν και οι δύο συναρτήσεις δημιουργείται η απόκριση του κρυφού στρώματος. Τέλος το σήμα αφού επεξεργαστεί με την ίδια διαδικασία από την συνάρτηση απόκρισης του στρώματος εξόδου, συνθέτει το τελικό σήμα εξόδου του

δικτύου. Η διαδικασία αυτή επαναλαμβάνεται στην περίπτωση που υπάρχουν περισσότερες από μια κρυφές στρώσεις νευρώνων. Για απλούστευση η μεροληψία γίνεται να σημειωθεί ως μία παραπάνω είσοδος, με δικό της συντελεστή βάρους και σταθερή τιμή εισόδου ίση με τη μονάδα.

Συνάρτηση απόκρισης νευρώνα

Η συνάρτηση απόκρισης νευρώνα (ρ) ελέγχει το σήμα που θα περάσει από τον νευρώνα, το οποίο θα αποτελέσει την έξοδό του. Συχνά χαρακτηρίζεται ως μια σύνθεση δύο συναρτήσεων, της συνάρτησης σύναψης (κ) και της συνάρτησης ενεργοποίησης (α), έτσι ώστε ρ = κ ◦ α.

Στην παρακάτω εικόνα βλέπουμε ένα μονό νευρώνα που βρίσκεται στην στρώση λ και έχει n συνδέσεις εισαγωγής δεδομένων. Οι συνδέσεις εισόδου φέρουν τον εκάστοτε συντελεστή βάρους w_{ij} , ο οποίος πολλαπλασιάζεται με το σήμα εισαγωγής. Καθώς το σήμα διαπερνά τον νευρώνα η συνάρτηση απόκρισης ορίζει πως θα διαμορφωθεί το τελικό σήμα δηλαδή το σήμα εξόδου.



Συναρτήσεις σύναψης

Οι συνηθέστερες συναρτήσεις σύναψης νευρώνα που χρησιμοποιούνται είναι:

Συνάρτηση αθροίσματος $k(y_i, w_{ij}) = \sum_{i=1}^{n} w_{ij}^{(\lambda)} y_i^{(\lambda)} + w_{0j}^{(\lambda)}$

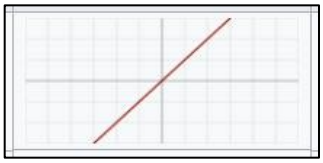
Αθροισμα τετραγώνων $k(y_i, w_{ij}) = \sum_{i=1}^{n} (w_{ij}^{(\lambda)} y_i^{(\lambda)})^2 + w_{0j}^{(\lambda)}$

Αθροισμα απόλυτων τιμών $k(y_i, w_{ij}) = \sum_{i=1}^{n} |w_{ij}^{(\lambda)} y_i^{(\lambda)}| + w_{0j}^{(\lambda)}$

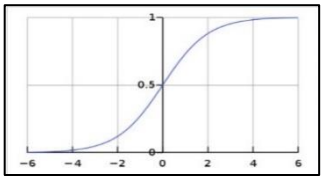
Συναρτήσεις ενεργοποίησης

Οι βασικότερες συναρτήσεις ενεργοποίησης ενός νευρώνα είναι:

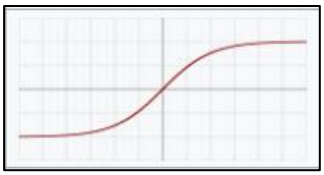
- *α(x) = x* *γραμμική (linear)*



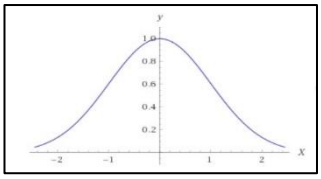
- *α(x) =* $\frac{1}{1+e^{-kx}}$ *σιγμοειδής (sigmoid)*



- *α(x) =* $\frac{e^{+x}-e^{-x}}{e^{+x}+e^{-x}}$ *υπερβολική εφαπτομένη (hyperbolic tangent - tanh)*



- *α(x) =* $e^{-x^2/2}$ *ακτινική (radial basis function -RBF)*



- *α(x) =* $\begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$ *βηματική (binary step)*

- *α(x) =* $\begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$ *γραμμικός ανορθωτής (rectified linear unit -ReLU)*

- *α(b,x)=* $\begin{cases} b(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$

εκθετική- γραμμική (exponential linear unit - ELU)

Μαθηματική περιγραφή απόκρισης

Θέλοντας λοιπόν να περιγράψουμε μαθηματικά τι συμβαίνει στο νευρωνικό δίκτυο μίας κρυφής στρώσης που βλέπουμε στην εικόνα (1), μπορούμε να ορίσουμε y_i την έξοδο του νευρώνα i της κρυφής στρώσης ως εξής:

$$y_i = \alpha_i \left(\sum_{j=1}^{d} w_{ji}^{(1)} x_j + w_{0i}^{(1)} \right) \quad (59)$$

Οπου α_i η συνάρτηση ενεργοποίησης του νευρώνα i, $w_{oi}^{(1)}$ η αντίστοιχη μεροληψία, x_j το σήμα εισόδου στο νευρώνα i, $w_{ji}^{(1)}$ ο συντελεστής βάρους απο τον νευρώνα j στον i και ως συνάρτηση σύναψης έγινε χρήση της αθροιστικής που είναι και η πιο διαδεδομένη. Η τελική έξοδος του δικτύου (z) δίνεται από την σχέση:

$$\mathbb{Z} = g \left(\sum_{i=1}^{m} w_i^{(2)} y_i + w_0^{(2)} \right) \quad (60)$$

Οπου g η συνάρτηση ενεργοποίησης του νευρώνα z, $w_{oi}^{(2)}$ η αντίστοιχη μεροληψία, y_i το σήμα εισόδου στο νευρώνα z, $w_i^{(2)}$ ο συντελεστής βάρους από το νευρώνα i στον z και ως συνάρτηση σύναψης έγινε πάλι χρήση της αθροιστικής. Συνεπώς από τις δύο παραπάνω εξισώσεις προκύπτει πως η συνολική απόκριση του νευρωνικού δικτύου ειναι:

$$\mathbb{Z} = g \left(\sum_{i=1}^{m} w_i^{(2)} [\alpha_i \left(\sum_{j=1}^{d} w_{ji}^{(1)} x_j + w_{0i}^{(1)} \right)] + w_0^{(2)} \right) \quad (61)$$

Θεώρημα Kolmogorov

Το 1957 ο Kolmogorov διατύπωσε ένα θεώρημα το οποίο έχει μια ενδιαφέρουσα συσχέτιση με τα νευρωνικά δίκτυα. Η ιστορία του συγκεκριμένου θεωρήματος ξεκινάει από το 1900, όταν ο Hilbert το συμπεριέλαβε σε μία λίστα με 23 άλυτα προβλήματα, τα οποία θα αποτελούσαν πρόκληση για τον 20^{ο} αιώνα. Το δέκατο τρίτο πρόβλημα από αυτά, αφορούσε το πότε μπορούμε να αναπαραστήσουμε συναρτήσεις πολλών μεταβλητών ως υπέρθεση συναρτήσεων λιγότερων μεταβλητών. Ετσι λοιπόν το 1957 ο ρώσος μαθηματικός Kolmogorov έδειξε ότι κάθε συνεχής συνάρτηση πολλών μεταβλητών για ένα κλειστό πεδίο ορισμού μπορεί να αναπαρασταθεί ως υπέρθεση ενός μικρού αριθμού συναρτήσεων μίας μεταβλητής. Στο θέμα των νευρωνικών δικτύων το παραπάνω θεώρημα βρίσκει εφαρμογή καθώς για κάθε συνεχή συνάρτηση απόκρισης νευρώνα y(**x**), από τις d μεταβλητές εισόδου x_i εώς την μία μεταβλητή εξόδου y, μπορούμε να δώσουμε την ακριβή αναπαράσταση ως ένα νευρωνικό δίκτυο τριών στρωμάτων. Το πρώτο κρυφό στρώμα θα πρέπει να έχει d(2d+1) μονάδες και το δεύτερο 2d+1. Στο παρακάτω δίκτυο απεικονίζεται η

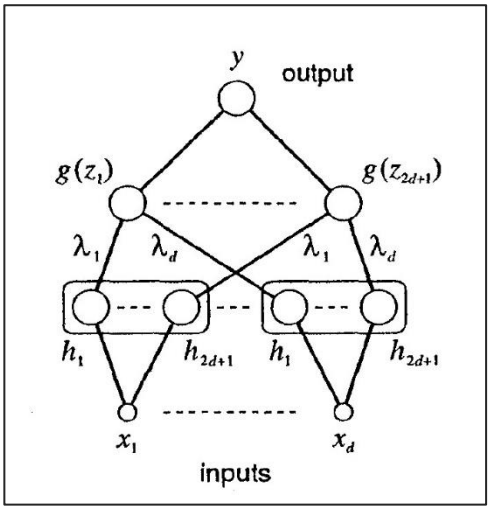
τοπολογική διάταξη που προκύπτει από την εφαρμογή του προαναφερθέντος θεωρήματος. Ετσι το αποτέλεσμα της σύναψης του j νευρώνα στην δεύτερη κρυφή στρώση μας δίνει:

$z_j = \sum_{i=1}^{d} \lambda_i h_j(x_i)$ όπου $0<\lambda_i<1$ είναι σταθερές (62)

Επίσης η έξοδος y του δικτύου είναι:

$$y = \sum_{j=1}^{2d+1} g(z_j) \quad (63)$$

Η συνάρτηση g θα πρέπει να είναι πραγματική και συνεχής, καθώς επίσης εξαρτάται από την y(**x**) που υποτίθεται ότι αναπαριστά, ενώ οι συναρτήσεις h_j δεν εξαρτώνται.



Network topology to implement Kolmogorov's theorem

Στο σημείο αυτό θα πρέπει να τονιστεί πως το θεώρημα αυτό εγγυάται την ύπαρξη ενός κατάλληλου δικτύου, όμως δεν μας δίνει τις συναρτήσεις h_j και g, ενώ παράλληλα δεν υπάρχει γνωστός μηχανισμός για την εύρεση τους. Συνεπώς το θεώρημα αν και σημαντικό είναι κάπως περιορισμένο στην πρακτική χρήση των νευρωνικών δικτύων κυρίως για δύο λόγους. Πρώτον οι συναρτήσεις h_j δεν είναι αρκετά ομαλές (smooth) και δεύτερον η συνάρτηση g εξαρτάται από την y(**x**), την οποία θέλουμε να αναπαραστήσουμε. [1]

Εκπαίδευση του νευρωνικού δικτύου

- Μέθοδος Οπισθοδιάδοσης (Back propagation)

Ο συνηθέστερος αλγόριθμος επαναπροσαρμογής των βαρών, που βελτιστοποιεί την επίδοση του δικτύου είναι η μέθοδος οπισθοδιάδοσης (BP). Ο αλγόριθμος αυτός ανήκει στην κατηγορία μεθόδων εκπαίδευσης με επίβλεψη (supervised learning methods), όπου το επιθυμητό αποτέλεσμα για κάθε δεδομένο εισαγωγής είναι γνωστό. Η έξοδος ενός νευρωνικού δικτύου (εδώ για λόγους απλούστευσης θεωρούμε πως έχουμε ένα κρυφό στρώμα με συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη και μία γραμμική συνάρτηση ενεργοποίησης στο στρώμα εξόδου) δίνεται από:

$$y_{NN} = \sum_{j=1}^{n_h} y_i^{(2)} w_{j1}^{(2)} = \sum_{j=1}^{n_h} \tanh\left(\sum_{i=1}^{n_{var}} x_i w_{ij}^{(1)}\right) w_{j1}^{(2)} \quad (64)$$

Όπου n_{var} και n_h είναι οι αριθμοί των νευρώνων του πρώτου φανερού στρώματος και του κρυφού στρώματος, αντίστοιχα. Το $w_{ij}^{(1)}$ είναι το βάρος μεταξύ του νευρώνα πρώτης στρώσης i και του νευρώνα κρυφής στρώσης j. Επίσης το $w_{j1}^{(2)}$είναι το βάρος μεταξύ του νευρώνα κρυφής στρώσης j και του νευρώνα εξόδου. Στην παραπάνω εξίσωση έγινε χρήση ως συνάρτηση σύναψης k το απλό άθροισμα.

Κατά την διάρκεια της διαδικασίας εκμάθησης το δίκτυο τροφοδοτείται από N γεγονότα εκπαίδευσης $\mathbf{x_\alpha}=(x_1,\ldots,x_{n_{var}})_\alpha$, α=1,..N. Για κάθε γεγονός εκπαίδευσης α η έξοδος του νευρωνικού δικτύου $y_{NN,a}$ υπολογίζεται και συγκρίνεται με την επιθυμητή έξοδο $\hat{y}_a$ ={1,0} (όπου στην ταξινόμηση έχουμε 1 για τα γεγονότα σήματος και 0 για το υπόβαθρο). Η συνάρτηση σφάλματος E, η οποία μετράει την ταύτιση της εξόδου του δικτύου με την επιθυμητή, δίνεται από:

$$E(\mathbf{x_1},\ldots,\mathbf{x_N}|\mathbf{w}) = \sum_{\alpha=1}^{N} E_\alpha(\mathbf{x_\alpha}|\mathbf{w}) = \sum_{\alpha=1}^{N} \frac{1}{2}(y_{NN,\alpha} - \hat{y}_a)^2 \quad (65)$$

Όπου το **w** δηλώνει το σύνολο των επαναπροσαρμοσμένων βαρών του δικτύου. Το σύνολο των βαρών που ελαχιστοποιεί τη συνάρτηση σφάλματος, μπορεί να βρεθεί κάνοντας χρήση της μεθόδου απότομης καθόδου (gradient descent), με δεδομένο ότι συνάρτηση απόκρισης νευρώνα είναι παραγωγίσιμη ως προς τα βάρη εισόδου. Ετσι λοιπόν ξεκινώντας από ένα τυχαίο σύνολο βαρών $w^{(\rho)}$, τα βάρη ανανεώνονται καθώς μετακινούνται μια μικρή απόσταση στον χώρο-w με κατεύθυνση $-\nabla_w E$.

$$w^{(\rho+1)} = w^{(\rho)} - \eta \cdot \nabla_w E \quad (66)$$

"η" ένας θετικός αριθμός που ονομάζεται ρυθμός εκμάθησης

Τα βάρη που συνδέονται με το εξωτερικό στρώμα ανανεώνονται με τον εξής τρόπο:

$$\Delta w_{j1}^{(2)} = -\eta \sum_{\alpha=1}^{N} \frac{\partial Ea}{\partial w_{j1}^{(2)}} = -\eta \sum_{\alpha=1}^{N} (y_{NN,\alpha} - \hat{y}_a)\, y_{j,a}^{(2)} \quad (67)$$

Ενώ τα βάρη που συνδέονται με τα κρυφά στρώματα ανανεώνονται σύμφωνα με:

$$\Delta w_{ij}^{(1)} = -\eta \sum_{\alpha=1}^{N} \frac{\partial Ea}{\partial w_{ij}^{(1)}} = -\eta \sum_{\alpha=1}^{N} (y_{NN,\alpha} - \hat{y}_a)\, y_{j,a}^{(2)} (1 - y_{j,a}^{(2)})\, w_{j1}^{(2)}\, x_{i,\alpha} \quad (68)$$

Οπου έγινε χρήση της σχέσης tanh'x= tanh x(1-tanh x).

Η παραπάνω μέθοδος εκπαίδευσης του δικτύου λέγεται μαζική εκμάθηση (batch learning) διότι το άθροισμα των σφαλμάτων όλου του συνόλου δεδομένων χρησιμοποιείται για να ανανεώσει τα βάρη. Μια εναλλακτική για τον τρόπο επαναπροσαρμογής των βαρών είναι η εκμάθηση ανα γεγονός (sequential learning or online learning). Τα βάρη σε αυτήν την περίπτωση περιγράφονται από τις ίδιες εξισώσεις απλά χωρίς το άθροισμα των γεγονότων. Στην περίπτωση αυτή είναι ιδιαίτερα σημαντικό να κάνουμε χρήση τυχαιοποιημένου δείγματος.

[1], [15]

- Μέθοδος BFGS (Broyden –Fletcher –Goldfarb –Shannon)

Η μέθοδος αυτή διαφέρει από την οπισθοδιάδοση στο γεγονός ότι γίνεται χρήση της δεύτερης παραγώγου της συνάρτησης σφάλματος ώστε να προσαρμόσει τα βάρη σύμφωνα με έναν αλγόριθμο τεσσάρων βημάτων:

1) Δύο διανύσματα D και Y υπολογίζονται. Το διάνυσμα των αλλαγών στα βάρη D αναπαριστά την εξέλιξη μεταξύ της (k-1) επανάληψης του αλγορίθμου και της (k). Κάθε σύναψη με βάρος ανταποκρίνεται σε ένα στοιχείο του διανύσματος. Το διάνυσμα Y είναι το διάνυσμα των σφαλμάτων κλίσης.

$$D_i^{(k)} = w_i^{(k)} - w_i^{(k-1)} \quad (69)$$

$$Y_i^{(k)} = g_i^{(k)} - g_i^{(k-1)} \quad (70)$$

Οπου το k είναι ο μετρητής των επαναλήψεων, το w_i το βάρος της ι-οστής σύναψης και g_i η ι-οστή κλίση συνάψεως.

2) Η προσέγγιση του ανάστροφου Hessian matrix (H^{-1}), κατά την επανάληψη k:

$$H^{-1\,(k)} = \{ D \cdot D^T (1 + Y^T \cdot H^{-1\,(k+1)} \cdot Y) \; / \; Y^T \cdot D \} - D \cdot Y^T \cdot H + H \cdot Y \cdot D^T + H^{-1\,(k-1)} \quad (71)$$

3) Εκτίμηση του διανύσματος αλλαγών στα βάρη από:

$$D^{(k)} = H^{-1\,(k)} \cdot Y^{(k)} \quad (72)$$

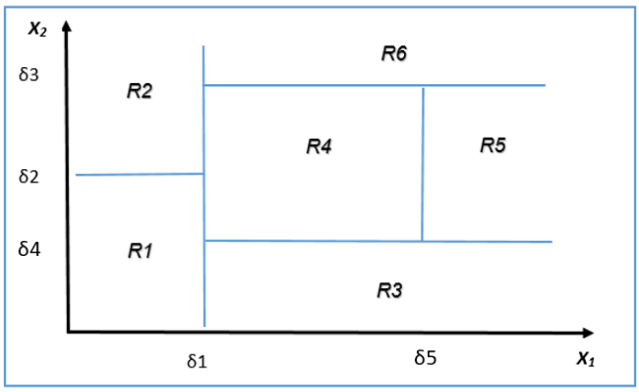
4) Υπολογισμός ενός νέου διανύσματος βαρών με την εφαρμογή του αλγορίθμου line search. Στον αλγόριθμο αυτό η συνάρτηση σφάλματος προσεγγίζεται τοπικά από μία παραβολή. Ο αλγόριθμος υπολογίζει την δεύτερη παράγωγο και αποφασίζει για το σημείο όπου αναμένεται το ελάχιστο της παραβολής. Το συνολικό σφάλμα υπολογίζεται για αυτό το σημείο και μετά ο αλγόριθμος υπολογίζει σημεία κατά μήκος της γραμμής που ορίζεται από την κατεύθυνση της κλίσης στον χώρο των βαρών ώστε να βρεθεί ολικό ελάχιστο. Τα βάρη στο ελάχιστο χρησιμοποιούνται στην επόμενη επανάληψη. Η παράμετρος εκμάθησης, η οποία ορίζει πόσο άλλαξαν τα βάρη σε κάθε επανάληψη, πολλαπλασιάζεται με το ποσοστό εκμάθησης όσο το σφάλμα του δικτύου με τα νέα βάρη είναι μικρότερο από αυτό με τα προηγούμενα. Εαν το σφάλμα του δικτύου με τα αλλαγμένα βάρη είναι μεγαλύτερο για την αρχική παράμετρο εκμάθησης, τότε το διαιρούμε με το ποσοστό εκμάθησης μέχρι να γίνει μικρότερο. Τέλος επειδή ο υπολογισμός του $H^{-1\,(k)}$ γίνεται λιγότερο ακριβής για αυξανόμενο αριθμό επαναλήψεων, θα πρέπει να γίνεται επαναφορά στον μοναδιαίο πίνακα σε κάθε ResetStep που ορίζουμε εμείς.

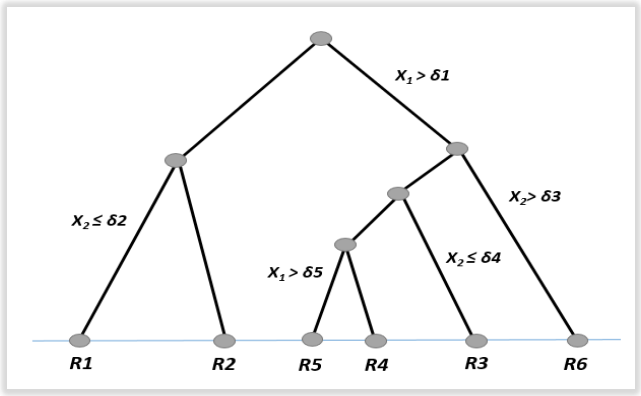
Η μέθοδος BFGS συγκριτικά με την BP χρειάζεται λιγότερες επαναλήψεις για την ολοκλήρωσή της, όμως επειδή η υπολογιστική ισχύς για την μία επανάληψη είναι ανάλογη του τετραγώνου του αριθμού των συνάψεων, για πολύ μεγάλα δίκτυα είναι περιορισμένη.

[1], [15]

3.8 ΔΕΝΤΡΑ ΑΠΟΦΑΣΗΣ
(decision trees)

Μια μεγάλη κατηγορία μη γραμμικών ταξινομητών αποτελούν αυτοί που εφαρμόζουν μια ακολουθία αποφάσεων, γνωστοί ως δέντρα απόφασης. Υπάρχουν πολλά απλά, αλλά ευρέως διαδεδομένα τέτοια μοντέλα τα οποία λειτουργούν διαχωρίζοντας τον χώρο των χαρακτηριστικών εισαγωγής (input space) σε κυβικές περιοχές. Οι ακμές των περιοχών αυτών θα πρέπει να είναι παράλληλες με τους άξονες και σε κάθε περιοχή ορίζουμε ένα μοντέλο – σταθερά. Η διαδικασία επιλογής ενός συγκεκριμένου μοντέλου, δοθέντος ενός νέου διανύσματος εισαγωγής (input vector) **x,** μπορεί να θεωρηθεί ως μια διαδοχική διαδικασία λήψης αποφάσεων. Αυτή η διαδοχική διαδικασία αντιστοιχεί σε ένα δυαδικό δέντρο, δηλαδή ένα δέντρο απόφασης που σε κάθε κόμβο χωρίζεται σε δυο κλαδιά. Στις παρακάτω 2 εικόνες βλέπουμε τον διαχωρισμό των περιοχών σε ένα δισδιάστατο χώρο που ορίζουν τα διανύσματα εισαγωγής, καθώς και το αντίστοιχο δέντρο απόφασης με τα κριτήρια διαχωρισμού σε κάθε κόμβο.





Για τα προβλήματα ταξινόμησης συνήθως αντιστοιχούμε κάθε περιοχή σε μια συγκεκριμένη κλάση. Μια χαρακτηριστική ιδιότητα των δέντρων απόφασης είναι ότι, επειδή αποτελούνται από ένα διαδοχικό σύνολο δυαδικών αποφάσεων πάνω στις εισαγόμενες μεταβλητές, είναι εύκολη η ανάλυση τους. Ο σκοπός ενός δυαδικού δέντρου απόφασης είναι η αύξηση της ομοιογένειας των υποσυνόλων που δημιουργούνται. Ο τρόπος λειτουργίας ενός δέντρου t διόδων είναι ο εξής:

$$\text{If } \{ \Delta(t) = I(t) - (N_{tY}/N_t)\cdot I(t_Y) - (N_{tN}/N_t)\cdot I(t_N) > 0 \} \text{ -> split} \quad (73)$$

else -> leaf.

Όπου:
> I(t) είναι η ομοιογένεια του κόμβου t.
> Οι δείκτες Y (yes) και N (no) συμβολίζουν την διαδρομή, δηλαδή την απάντηση στην ερώτηση -κριτήριο διαχωρισμού σε κάθε κόμβο t.
> Nt είναι το σύνολο των διανυσμάτων εισαγωγής στον κόμβο t και η πιθανότητα $P(R_i|t)$ ενός υποσυνόλου $\mathbf{x_t}$ να ανήκει στην κλάση -περιοχή R_i είναι ίση με N_{it}/N_t.
Συνεπώς από τα παραπάνω είναι σαφές ότι η σχέση (73) ορίζει, πως αν η διαφορά της ομοιογένειας είναι θετική ,τότε το δέντρο συνεχίζει , αλλιώς σταματάει (leaf).

Ορισμός της ομοιογένειας για ένα κόμβο:
(Δυο συναρτήσεις μπορούν να χρησιμοποιηθούν και κάθε φορά που κάνουμε εκπαίδευση του αλγορίθμου επιλέγουμε την κατάλληλη.)

$$I = p\cdot(1-p) \quad \text{(Gini Index)} \tag{74}$$
$$I = -\, p\cdot\ln p - (1-p)\cdot\ln(1-p) \quad \text{(Cross Entropy)} \tag{75}$$
[$p = P(R_i|t)$ όπως προαναφέρθηκε.]

Μια άλλη προσέγγιση σε αυτό τον αλγόριθμο είναι το κλάδεμα (pruning).
Αυτή η τεχνική βασίζεται σε ένα κριτήριο το οποίο εξισορροπεί το υπολειπόμενο σφάλμα έναντι της πολυπλοκότητας του μοντέλου.
Έτσι εάν ονομάσουμε το δέντρο που ξεκινάει το κλάδεμα ως T_o, τότε θα πρέπει να ορίσουμε και ένα μικρότερο δέντρο $T \subset T_o$ το οποίο θα προκύψει κλαδεύοντας κόμβους από το T_o . Θεωρώντας ότι οι τελικοί κόμβοι (leaves) δηλώνονται με $\tau = 1,\ldots,|T|$, όπου ο τελικός κόμβος τ αναπαριστά την περιοχή R_τ του χώρου των διανυσμάτων εισαγωγής (input space), έχουμε τη βέλτιστη πρόβλεψη για την περιοχή R_τ :

$$y_\tau = 1/N_\tau \cdot \sum_{i=1}^{n} t_I \tag{76}$$

Την αντίστοιχη συμβολή στην συνολική ομοιογένεια:

$$I_\tau(T) = \sum_{\kappa=1}^{\kappa} P\tau\kappa \cdot \ln(P\tau\kappa) \quad \text{(Negative cross entropy)} \tag{77}$$
$$I_\tau(T) = \sum_{\kappa=1}^{\kappa} P\tau\kappa \cdot (1 - P\tau\kappa) \quad \text{(Gini index)} \tag{78}$$

με Ρτκ μέρος των δεδομένων της περιοχής Rτ που καταχωρήθηκε στην κλάση C_k. Καθώς και το κριτήριο κλαδέματος:

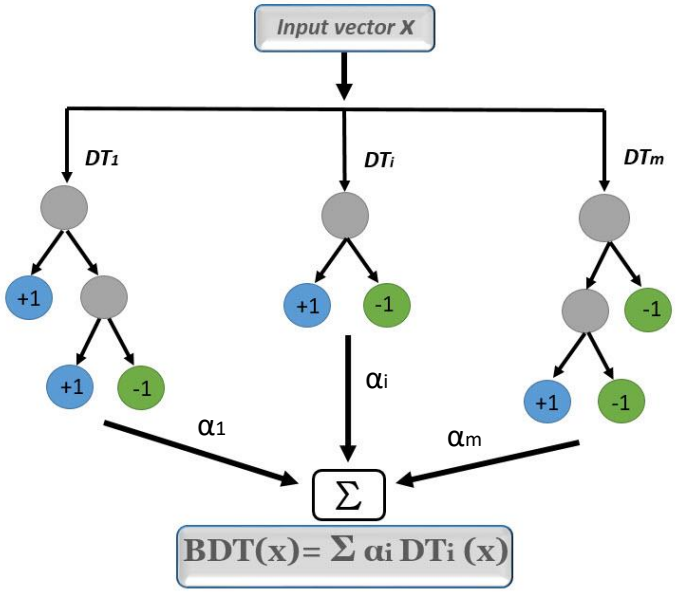
$$C(T) = \sum_{T=1}^{|T|} I\tau(T) + \lambda \cdot |T| \quad \text{(pruning criterion} \tag{79}$$

με λ μια παράμετρος (πολλαπλασιαστής Lagrange) που ρυθμίζει την ισορροπία μεταξύ I_τ και $|T|$.

3.9 ΣΥΝΔΥΑΣΜΟΣ ΤΑΞΙΝΟΜΗΤΩΝ (Boosting)

Η τεχνική του boosting είναι μια ισχυρή μέθοδος που προκύπτει από τον συνδυασμό πολλών απλών και συχνά ασθενών ταξινομητών με σκοπό να παράγει ένα καλύτερο αποτέλεσμα. Η πιο ευρέως διαδεδομένη μορφή της τεχνικής αυτής είναι ο αλγόριθμος Adaboost τον οποιο θα αναλύσουμε παρακάτω.

Για την επεξήγηση της μεθόδου θα γίνει αναφορά αρχικά στην εφαρμογή της στα δέντρα απόφασης (boosted decision trees), η οποία και εφαρμόστηκε στα πλαίσια αυτής τη διπλωματικής. Έτσι ο σκοπός είναι η τροποποίηση του δέντρου απόφασης ώστε να απαλλαγούμε από τυχαίες διακυμάνσεις του συνόλου εκπαίδευσης. Αυτό επιτυγχάνεται με την κατασκευή ενός “δάσους” από δέντρα, τα οποία εκπαιδεύτηκαν από τα ίδια δεδομένα με διαφορετικά βάρη κάθε φορά. Στο τέλος παίρνουμε το βεβαρυμένο μέσο όρο όλων αυτών των δέντρων μικρού βάθους που δημιουργήσαμε (βλ. εικόνα).



(Adaboost)

Έστω ϕ(**x;θ**) το αποτέλεσμα ενός ταξινομητή, όπου **x** είναι το διάνυσμα χαρακτηριστικών μεταβλητών και **θ** ένα διάνυσμα παραμέτρων. Θέλουμε να κατασκευάσουμε έναν άλλο ταξινομητή f(x)= sign(F(x)), όπου:

$$F(\mathbf{x}) = \sum_k \alpha_k \cdot \phi(\mathbf{x};\boldsymbol{\theta}_k) \quad (80)$$

Συνεπώς με τα ίδια δεδομένα εκπαιδεύουμε πολλούς ταξινομητές και κατόπιν παίρνουμε ένα άθροισμα αυτών με τα κατάλληλα βάρη α_k. Η μέθοδος boosting όπως προαναφέρθηκε είναι η αλληλουχία εκπαιδεύσεων ενός ταξινομητή όπου κάθε φορά τα δεδομένα έχουν διαφορετικά βάρη. Ο βασικότερος αλγόριθμος της είναι ο Adaboost στον οποίο χρησιμοποιείται η εξής συνάρτηση κόστους:

$$J(\alpha_k;\boldsymbol{\theta}_k) = \sum_{i=1}^{N} \exp[-y_i \cdot F(\mathbf{x}_i)] \quad (81)$$

Όπως βλέπουμε στην συγκεκριμένη συνάρτηση κόστους, στα λάθος ταξινομημένα διανύσματα ($y_i \cdot F(\mathbf{x}_i)<0$), επιβάλλεται πολύ μεγαλύτερη ποινή από αυτά που ταξινομούνται σωστά ($y_i \cdot F(\mathbf{x}_i)>0$). Ωστόσο η απευθείας βελτιστοποίηση αυτής της συνάρτησης είναι αρκετά δύσκολη γι αυτό θα πρέπει να οριστεί η βοηθητική έννοια του μερικού αθροίσματος F_m:

$$F_m(\mathbf{x})=\sum_{k=1}^{m} a_k \cdot \phi(\mathbf{x};\boldsymbol{\theta}_k)$$

όποτε ισχύει πως: $F_m(\mathbf{x})= F_{m-1} + \alpha_m \phi(\mathbf{x};\boldsymbol{\theta}_m)$ (αναδρομική σχέση) (82)

Συνεπώς σύμφωνα με την αναδρομική σχέση του μερικού αθροίσματος η συνάρτηση κόστους μπορεί να ξαναγραφτεί ως εξής:

$$J(\alpha_m;\boldsymbol{\theta}_m) = \sum_{i=1}^{N} \exp(-y_i \cdot [\, F_{m-1}(\mathbf{x}) + \alpha_m \phi(\mathbf{x};\boldsymbol{\theta}_m)]) \Rightarrow$$

$$\Rightarrow J(\alpha_m;\boldsymbol{\theta}_m) = \sum_{i=1}^{N} w_i^{(m)} \exp -y_i \cdot \alpha_m \phi(\mathbf{x};\boldsymbol{\theta}_m)] \quad (83)$$

$$\text{με } w_i^{(m)} = \exp[-y_i \cdot F_{m-1}(\mathbf{x})]$$

Δηλαδή κάθε διάνυσμα εκπαίδευσης $\mathbf{x}_i$ αποκτά ένα βάρος w_i το οποίο σχετίζεται με το αποτέλεσμα της ταξινόμησης στο προηγούμενο βήμα. Η εκπαίδευση του ταξινομητή στο βήμα m ισοδυναμεί με την εύρεση των παραμέτρων $\boldsymbol{\theta}_m$ που ισοδυναμεί με την ελαχιστοποίηση του σφάλματος ταξινόμησης P_m.

$$P_m = \sum_{y_i \phi(x_i;\, \theta_m)<0}^{N} w_i^{(m)} \quad \text{και} \quad 1-P_m = \sum_{y_i \phi(x_i;\theta_m)>0}^{N} w_i^{(m)} \quad (84)$$

Προκειμένου να ερμηνεύσουμε τα βάρη $w_i^{(m)}$ ως πιθανότητες απαιτούμε να είναι κανονικοποιημένα σε κάθε βήμα m. Έτσι κάνοντας χρήση της πιθανότητας P_m η συνάρτηση κόστους γράφεται:

$$J(\alpha_m;\boldsymbol{\theta}_m) = e^{-a_m}(1-P_m) + e^{a_m} \cdot P_m \quad (85)$$

η οποία ελαχιστοποιείται για:

$$a_m = \frac{1}{2} \ln\left(\frac{1-P_m}{P_m}\right) \quad (86)$$

Τέλος τα βάρη $w_i^{(m+1)}$ για το επόμενο βήμα μπορούν να υπολογιστούν κάνοντας χρήση των ελάχιστων τιμών των πιθανοτήτων P_m και των α_m που υπολογίστηκαν πριν:

$$w_i^{(m+1)} = \frac{1}{Zm} \exp[-y_i \cdot F_m(\mathbf{x}_i)] = \frac{1}{Zm} w_i^{(m)} \exp -y_i \cdot \alpha_m \phi(\mathbf{x}_i ; \boldsymbol{\theta}_m)] \quad (87)$$

[7], [9]

3.10 ΚΡΙΤΗΡΙΑ ΑΞΙΟΛΟΓΗΣΗΣ ΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΕΚΠΑΙΔΕΥΣΗΣ (Confusion Matrix - ROC Curve)

Ο **Confusion Matrix** είναι ένας πίνακας που χρησιμοποιείται για να περιγράψει την απόδοση ενός μοντέλου ταξινόμησης, το οποίο εφαρμόστηκε σε ένα σύνολο δεδομένων που ήταν γνωστές οι πραγματικές τιμές. Η έννοια του πίνακα αυτού είναι αρκετά απλή, όμως οι ορολογίες που εμφανίζονται σε αυτόν μπορούν να εκφραστούν με αρκετούς τρόπους και αυτό μπορεί να προκαλέσει σύγχυση. Γενικά ο confusion matrix αποτελείται από δυο γραμμές και δυο στήλες που αναφέρουν τα εξής τέσσερα χαρακτηριστικά: false positive, false negative, true positive και true negative. Επίσης μπορεί να σχηματιστεί και με ποσοστά τις αντί των αριθμητικών τιμών. Η χρήση του πίνακα μας δίνει την δυνατότητα μιας πιο λεπτουμερους ανάλυσης από την απλή αναλογία των σωστών ταξινομήσεων προς όλες (accuracy). Η αναλογία των σωστών ταξινομήσεων προς όλες δεν αποτελεί ένα αξιόπιστο μέτρο αξιολόγησης της απόδοσης καθώς μπορεί να μας παραπλανήσει στην περίπτωση που το δείγμα δεν είναι ισορροπημένο (δηλαδή όταν ο αριθμός των παρατηρήσεων για κάθε κλάση διαφέρει σημαντικά).

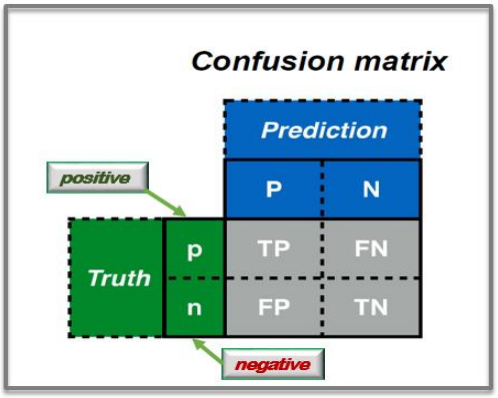
Στο σημείο αυτό θα πρέπει να ορίσουμε τα τέσσερα στοιχεία του πίνακα:

True positives (TP): Οι περιπτώσεις που προβλέψαμε θετικές (positive) και ήταν και στην πραγματικότητα θετικές.

True negatives (TN): Οι περιπτώσεις που προβλέψαμε αρνητικές (negative) και ήταν και στην πραγματικότητά αρνητικές.

False positives (FP): Οι περιπτώσεις που προβλέψαμε θετικές και ήταν στην πραγματικότητα αρνητικές.

False negatives (FN): Οι περιπτώσεις που προβλέψαμε αρνητικές και ήταν στην πραγματικότητα θετικές.



Η καμπύλη **ROC** (Receiver operating characteristic) χρησιμοποιήθηκε πρώτη φορά κατά την διάρκεια του δευτέρου παγκοσμίου πολέμου στην ανάλυση των σημάτων που λάμβαναν με ραντάρ. Μετά την μάχη στο Pearl Harbor το 1941, ο αμερικανικός στρατός ξεκίνησε μια νέα έρευνα για να αυξήσει την πιθανότητα σωστής ανίχνευσης των Ιαπωνέζικων αεροσκαφών από τα σήματα ραντάρ. Έτσι γι αυτό τον σκοπό μέτρησαν την ικανότητα των λειτουργιών των ραντάρ να κάνουν σωστούς διαχωρισμούς, την οποία αποκαλούσαν ‘’Receiver Operating Characteristics”. Στην δεκαετία του πενήντα οι καμπύλες ROC εφαρμόστηκαν σε πολλούς τομείς όπως την ψυχιατρική, την διαγνωστική ιατρική, την επιδημιολογία ,την ραδιολογία, ενώ αργότερα βρήκε εφαρμογή στην εκτίμηση των τεχνικών της μηχανικής μάθησης. Ο πρώτος που εφάρμοσε την καμπύλη αυτή στην αξιολόγηση διαφορετικών αλγορίθμων ταξινόμησης ήταν ο Spackman.

Για να αναλυθεί ο τρόπος που προκύπτει η καμπύλη ROC θα πρέπει να αναφερθούν κάποιες έννοιες που βασίζονται στα στοιχεία του confusion matrix:

True positive rate (TPR) = $\frac{\Sigma TruePositive}{\Sigma ConditionPositive} = \frac{TP}{TP+FN}$ *(sensitivity)*

True negative rate (TNR) = $\frac{\Sigma TrueNegative}{\Sigma ConditionNegative} = \frac{TN}{FP+TN}$ *(specificity)*

False positive rate (FPR) = $\frac{\Sigma FalsePositive}{\Sigma ConditionNegative} = \frac{FP}{FP+TN}$ *(fall-out)*

False negative rate (FNR) = $\frac{\Sigma FalseNegative}{\Sigma ConditionPositive} = \frac{FN}{TP+FN}$ *(miss rate)*

Positive Likelihood ratio (LR+) = $\frac{TPR}{FPR} = \frac{sensitivity}{1-specificity}$

Accuracy (ACC) = $\frac{TP+TN}{Total}$

Positive Predicted Value (PPV) = $\frac{\Sigma TruePositive}{\Sigma PredictedConditionPositive} = \frac{TP}{TP+FP}$ *(precision)*

F_1 score = $\frac{2}{\frac{1}{TPR}+\frac{1}{PPV}}$ *(evaluation metric)*

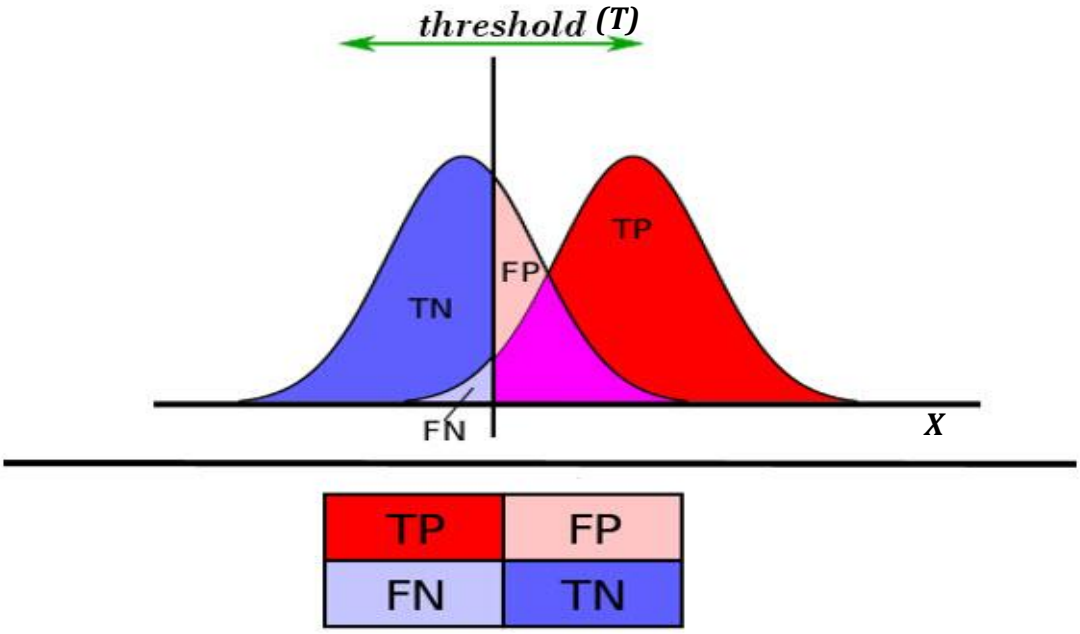
Για να σχεδιάσουμε την καμπύλη ROC λοιπόν θα πρέπει να σχεδιάσουμε την ευαισθησία (TPR) σε συνάρτηση με το συμπλήρωμα της εξειδίκευσης (FPR = 100-specificity) για διαφορετικά σημεία κατωφλίου. Κάθε σημείο στην καμπύλη ROC αναπαριστά ένα λόγο ευαισθησίας/εξειδίκευσης που αντιστοιχεί σε ένα συγκεκριμένο κατώφλι διαχωρισμού. Ένα τεστ με τέλειο διαχωρισμό (χωρίς αλληλεπικαλύψεις στις δυο κατανομές) έχει μια καμπύλη ROC που περνά από το υψηλότερο σημείο στην αριστερή γωνία και έχει 100% ευαισθησία και εξειδίκευση. [2],[17],[9]

(Ερμηνεία καμπύλης με βάση τις κατανομές)

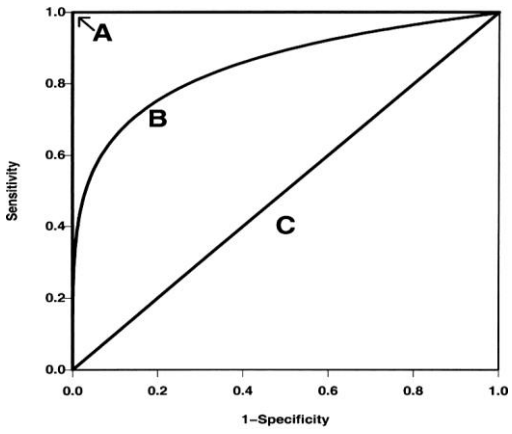
Στην ταξινόμηση δυο κλάσεων, η πρόβλεψη για κάθε παράδειγμα γίνεται συνήθως με βάση μια τυχαία συνεχή μεταβλητή X, η οποία υπολογίζεται για κάθε παράδειγμα και λειτουργεί ως μετρικό. Έτσι με δεδομένη την παράμετρο κατωφλίου T, εάν X>T τότε το στοιχείο ταξινομείται ως θετικό αλλιώς αρνητικό. Το X έχει πυκνότητα πιθανότητας $f_1(x)$ εάν ανήκει όντως στην θετική κλάση και $f_0(x)$ αλλιώς. Συνεπώς τα ποσοστά TPR και FPR δίνονται από :

$$TPR(T) = \int_T^{\infty} f_1(x)\,dx \quad \text{και} \quad FPR(T) = \int_T^{\infty} f_0(x)\,dx$$

Η καμπύλη ROC λοιπόν απεικονίζει παραμετρικά τα δυο αυτά ποσοστά καθώς το κατώφλι T αλλάζει τιμές, όπως βλέπουμε στην εικόνα παρακάτω.



Τέλος ένας τρόπος σύγκρισης των καμπύλων ROC μεταξύ τους είναι ο υπολογισμός του χωρίου κάτω από την καμπύλη ως ποσοστό του συνολικού χωρίου του διαγράμματος (AUC – area under the curve). Για παράδειγμα στις καμπύλες που βλέπουμε στο παρακάτω διάγραμμα η A εχει AUC=1 , η C AUC=0,5 ενώ η B κάπου ενδιάμεσα.



ΚΕΦΑΛΑΙΟ 4
ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

4.1 ΕΡΓΑΛΕΊΑ ΑΝΆΛΥΣΗΣ ΔΕΔΟΜΈΝΩΝ

Το κύριο εργαλείο με το οποίο έγινε η ανάλυση των δεδομένων είναι το TMVA του ROOT, όμως επειδή τα αποτελέσματα δεν ήταν τα επιθυμητά σε όλες τις μεθόδους, έγινε χρήση και του Matlab σε ρόλο επαλήθευσης.

Το **ROOT** είναι ένα δωρεάν και open-source λογισμικό για ανάλυση δεδομένων, το οποίο αναπτύχθηκε στο CERN, το ερευνητικό κέντρο στο οποίο εφευρέθηκε και το World Wide Web. Η ανάπτυξη του ROOT οφείλεται στην αναγκαιότητα αντιμετώπισης των προκλήσεων που δημιούργησαν τα σύγχρονα πειράματα φυσικής υψηλών ενεργειών, στα οποία παράγονται και αναλύονται αρκετά Petabytes δεδομένων σε ετήσια βάση.

Το **TMVA** (toolkit for multivariate analysis) παρέχει ένα ενσωματωμένο στο ROOT περιβάλλον για εφαρμογή της μηχανικής μάθησης στην επεξεργασία δεδομένων μέσω τεχνικών ταξινόμησης και παλινδρόμησης. Το TMVA έχει σχεδιαστεί ειδικά για τις ανάγκες των εφαρμογών της φυσικής υψηλών ενεργειών, όμως δεν περιορίζεται μόνο σε αυτές καθώς είναι κατάλληλο για ένα μεγάλο εύρος προβλημάτων μηχανικής μάθησης.

Τόσο στο ROOT όσο και στο TMVA ο προσδιορισμός των μεθόδων, η δήλωση των ορισμάτων τους, τα ιστογράμματα των μεταβλητών εισαγωγής καθώς και πολλές άλλες τροποποιήσεις – προεργασίες των δεδομένων έγιναν με αρχεία κειμένου σε **C++**. Τα αρχεία αυτά που έτρεχαν το πρόγραμμα κάνοντας την ανάλυση παρατίθενται στα παραρτήματα στο τέλος.

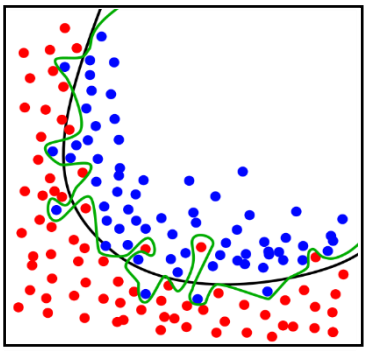
Το **Matlab** (matrix laboratory) είναι ένα πρόγραμμα αριθμητικού υπολογισμού με βάση τις πράξεις πινάκων. Αυτό που το καθιστά ισχυρό σαν λογισμικό είναι το πλήθος των ‘‘εργαλειοθηκών’’ που παρέχει. Συνεπώς το γεγονός ότι περιλαμβάνει εργαλειοθήκη για μηχανική μάθηση και ειδικότερα για ταξινόμηση, σε συνδυασμό με την δυνατότητα να δέχεται προγραμματιστικές εντολές, συντέλεσε στην επιλογή του ως εναλλακτικό- επιπρόσθετο εργαλείο μελέτης για την παρούσα διπλωματική.

4.2 ΟΡΙΣΜΑΤΑ ΤΩΝ ΜΕΘΟΔΩΝ ΠΟΥ ΜΕΛΕΤΗΘΗΚΑΝ

Neural Networks	***Boosted Decision Trees***	***Support Vector Machine***	***k Nearest Neighbors***	***Fisher***
NCycles (num. of iterations)	*NTrees (num.of trees)*	*Kernel (Poln,Gaus)*	*nkNN (num.of neighb.)*	
HiddenLayers (net architecture)	*Max Depth*		*Kernel (Poln, Gaus)*	
TrainingMethod (BP,GA,BFGS)	*SeparationType (Gini Index, Cross Entropy)*			
LearningRate	*BoostType (Adaboost,Grad)*			
BPMode (Sequential, Batch)	*Learning rate (Shrinkage/ AdaboostBeta)*			

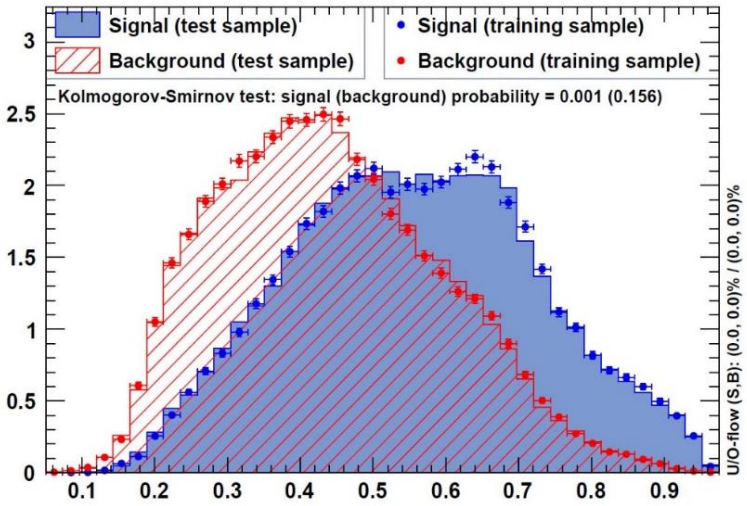
Εκτός από τις παραπάνω πέντε μεθόδους έγινε επίσης εφαρμογή του γραμμικού ταξινομητή (LD), της εύρεσης της πιθανοφάνειας (PDE) καθώς και της τεχνικής boosting στην Fisher. Ο λόγος για τον οποίο οι παραπάνω μέθοδοι δεν συμπεριλήφθηκαν ούτε στον πίνακα με τα ορίσματα αλλά ούτε και σε αυτόν με τα αποτελέσματα παρακάτω, είναι διότι δεν είχαν τα αποτελέσματα καθόλου διαφοροποιήσεις από την απλή Fisher.

4.3 ΤΟ ΦΑΙΝΟΜΕΝΟ ΤΗΣ ΥΠΕΡΕΚΠΑΙΔΕΥΣΗΣ ΚΑΙ Η ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ



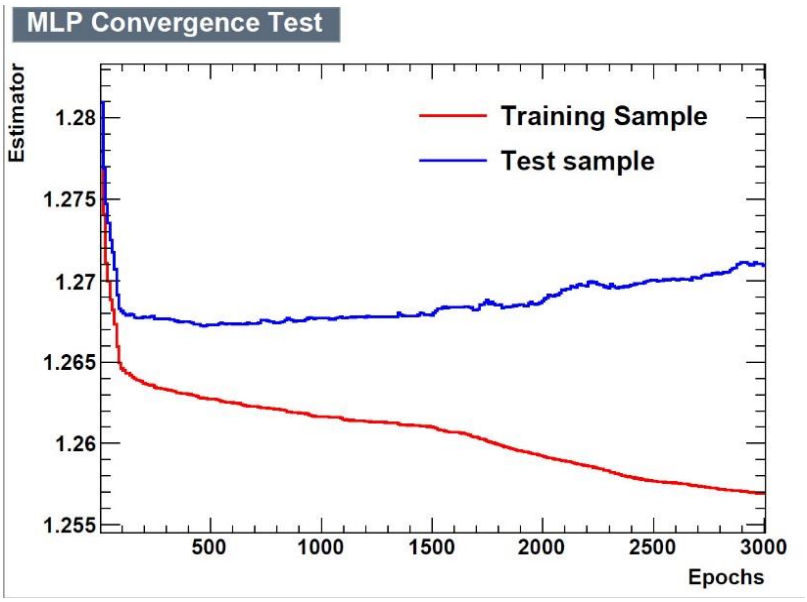
Όταν ένας ταξινομητής εξειδικευτεί υπερβολικά στο συγκεκριμένο δείγμα που του δόθηκε, τότε δεν ορίζει τον γενικό κανόνα διαχωρισμού άλλα έναν ειδικό ο οποίος είναι ιδιαίτερα αποτελεσματικός μόνο για το συγκεκριμένο δείγμα. Ετσι όπως βλέπουμε και στην εικόνα δίπλα η πράσινη γραμμή μπορεί να ξεχωρίσει τέλεια τα δύο είδη σημείων, αλλά σε καμία περίπτωση δεν θα είναι λειτουργική σε άλλο δείγμα.

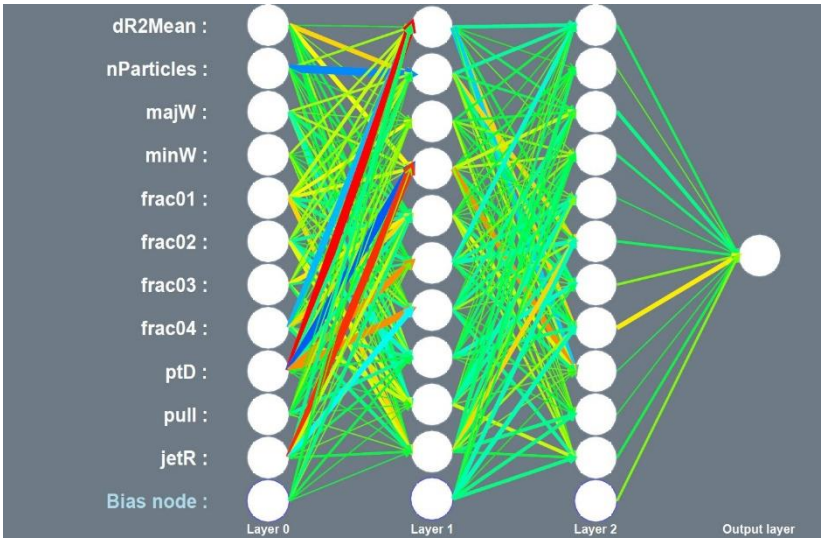
Από όλους τους ταξινομητές μεγαλύτερη ευαισθησία στην υπερεκπαίδευση (overtraining) έχουν τα νευρωνικά δίκτυα. Η τεχνική για τον εντοπισμό της υπερεκπαίδευσης βασίζεται στον διαχωρισμό του συνόλου δεδομένων σε δύο υποσύνολα, το test sample και το training sample. Ετσι μόλις ολοκληρωθεί η εκπαίδευση παίρνουμε τις κατανομές και τις συγκρίνουμε. Στην εικόνα κάτω παρατηρούμε πως στην μπλέ περιοχή που αναφέρεται στο σήμα, η καμπύλη για το test sample (σκιασμένη) και η καμπύλη για το training sample (bullets) διαφέρουν σε κάποια σημεία. Το γεγονός αυτό μας υποδηλώνει πως υπάρχει υπερεκπαίδευση.



Για να προσδιορίσουμε μετά από ποιά επανάληψη αρχίζει το φαινόμενο χρειάζεται να απεικονίσουμε σε διάγραμμα τα δύο υποσύνολα, όπως στην παρακάτω εικόνα. Στον x άξονα έχουμε τον αριθμό των επαναλήψεων και στον y την συνάρτηση σφάλματος. Προφανώς η συνάρτηση αυτή αποτελεί ένδειξη της απόστασης από το επιθυμητό αποτέλεσμα και ο σκοπός είναι η ελαχιστοποίηση της. Στο διάγραμμα λοιπόν βλέπουμε πως μετά την 500^{η}

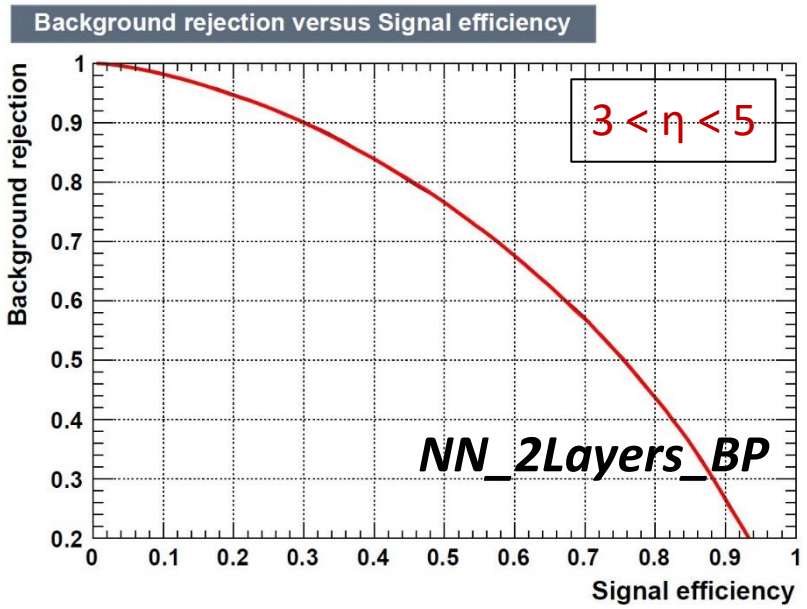
επανάληψη ενώ στο test sample η συνάρτηση σφάλματος παίρνει την ελάχιστη τιμή και μετά αυξάνει, στο training sample δεν αποκτά ποτέ ελάχιστο. Συνεπώς προκύπτει το συμπέρασμα ότι μετά την 500η επανάληψη αρχίζει η υπερεκπαίδευση και εκεί θα πρέπει να ορίσουμε την μέθοδο να σταματάει ώστε να έχουμε το καλύτερο δυνατό αποτέλεσμα.

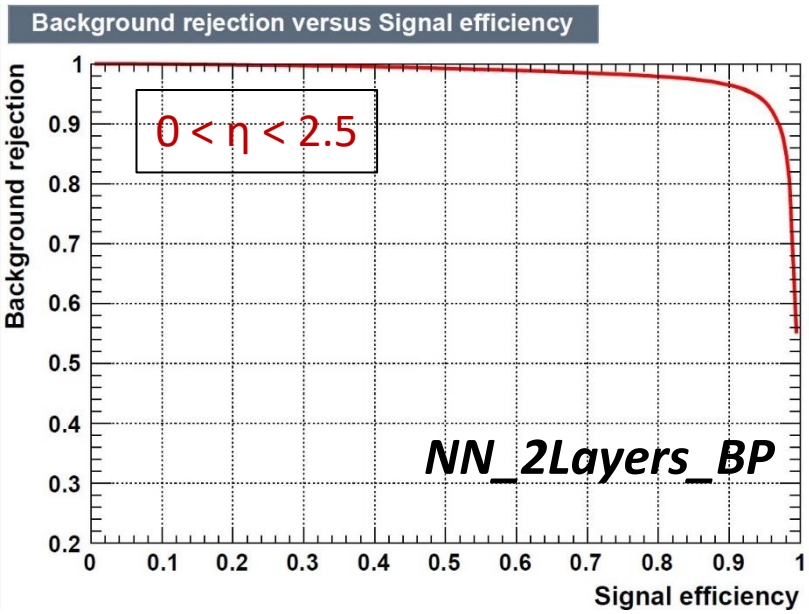




4.4 ΣΥΓΚΡΙΣΗ ΤΩΝ ΠΕΡΙΟΧΩΝ ΤΟΥ η (ΕΤΑ)

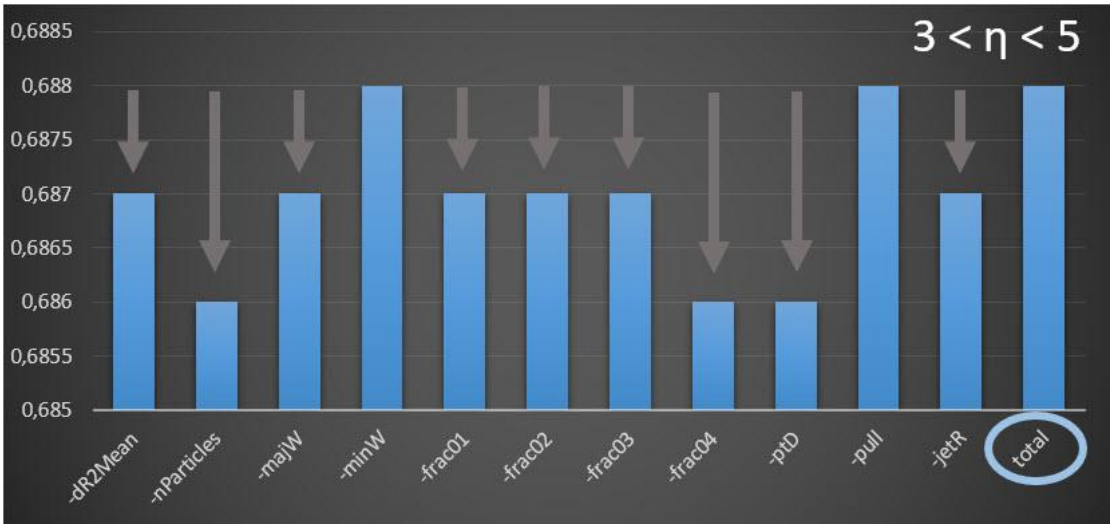
Οπως έχει προαναφερθεί η μελέτη του προβλήματος έγινε για τέσσερις περιοχές του η και πιο συγκεκριμένα εστίασε στις δύο ακραίες περιοχές την 0<η< 2,5 και την 3<η<5. Όπως βλέπουμε στις παρακάτω καμπύλες ROC για ένα νευρωνικό δίκτυο δύο στρωμάτων με εκπαίδευση οπισθοδιάδοσης (backpropagation) τα αποτελέσματα διαφέρουν σημαντικά. Στην περιοχή του 3<η<5 η επίδοση του ίδιου νευρωνικού δηλαδή το εμβαδό κάτω από την καμπύλη ROC μειώνεται δραματικά σε σχέση με την 0<η<2,5 . Η μείωση αυτή σχετίζεται με απώλεια διακριτικής ικανότητας δεδομένου πως στην περιοχή αυτή αφαιρέθηκαν τρεις μεταβλητές.

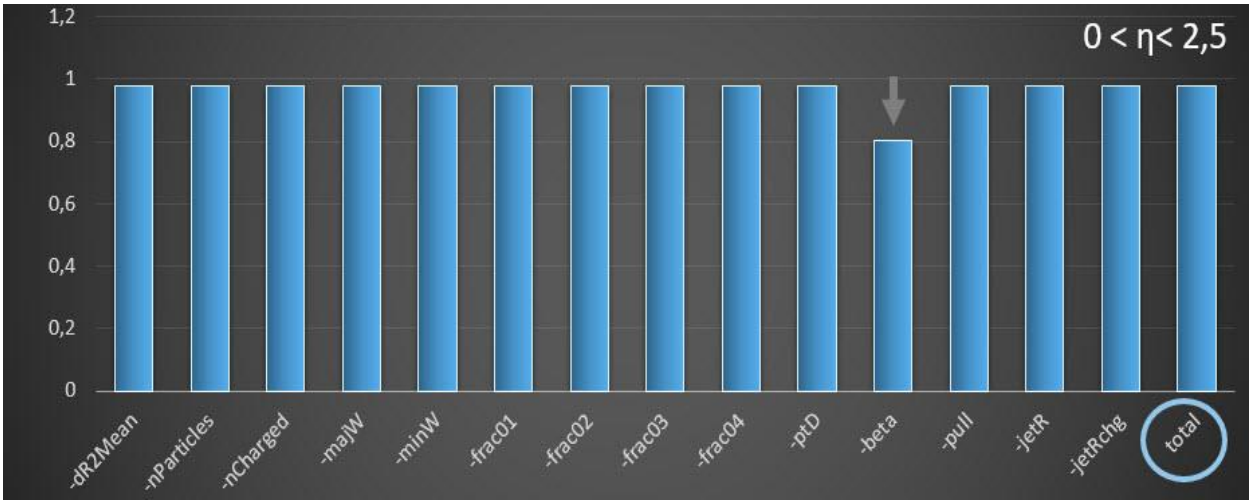




4.5 ΑΞΙΟΛΟΓΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Το γεγονός ότι η απώλεια τριών μεταβλητών μας μείωσε σε τέτοιο βαθμό την διαχωριστική ικανότητα του προβλήματος, μας οδήγησε στο να κατατάξουμε τις μεταβλητές με βάση την διακριτική τους ικανότητα. Ενας εύκολος τρόπος να γίνει αυτό αποτελεί το να εκπαιδεύσουμε ένα γραμμικό ταξινομητή, για λόγους εξοικονόμησης χρόνου, πολλές φορές αφαιρώντας μία μεταβλητή κάθε φορά. Όσο μεγαλύτερη είναι η πτώση στην συνολική επίδοση της μεθόδου από την αφαίρεση της μεταβλητής, τόσο μεγαλύτερη διακριτική ικανότητα έχει η μεταβλητή αυτή. Συνεπώς όπως βλέπουμε στο ραβδόγραμμα για την περιοχή 3<η<5 οι μεταβλητές nParticles, frac04 και ptD έχουν την μεγαλύτερη διακριτική ικανότητα. Στο ραβδόγραμμα για την περιοχή 0<η<2,5 όμως βλέπουμε πως η αφαίρεση της μεταβλητής beta προκαλεί ασύγκριτα μεγαλύτερη πτώση απο τις υπόλοιπες. Συνεπώς καταλήγουμε πως η beta είναι η μεταβλητή με την μεγαλύτερη διακριτική ικανότητα στο πρόβλημα μας.





4.6 ΣΥΓΚΕΝΤΡΩΤΙΚΟΣ ΠΙΝΑΚΑΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΑΙ ΠΑΡΑΤΗΡΗΣΕΙΣ

ΠΙΝΑΚΑΣ 1

Eta0to2p5	***total time(sec)***	***(min)***	***weights(bytes)***	***(MB)***	***ROC_curve***
Fisher	0,40	0	16126	0,0	0,964
Likelihood	1,98	0	4751344	4,5	0,945
KNN200	2460,56	41	38214697	36,4	0,976
NN(BP)	747,94	12	49551	0,0	0,979
NN(GA)	/	/	/	/	/
NN(BFGS)	2930,95	49	49567	0,0	0,979
NN(BP)_N,N-1	1081,40	18	64911	0,1	0,979
NN(BFGS)_N,N-1	5401,39	90	64879	0,1	0,979
BDT-100	239,50	4	2429452	2,3	0,979
BDT_Gini_Index	243,20	4	2398657	2,3	0,979
BDT_Cross_Entropy	238,40	4	2418873	2,3	0,979
SVM	/	/	/	/	/

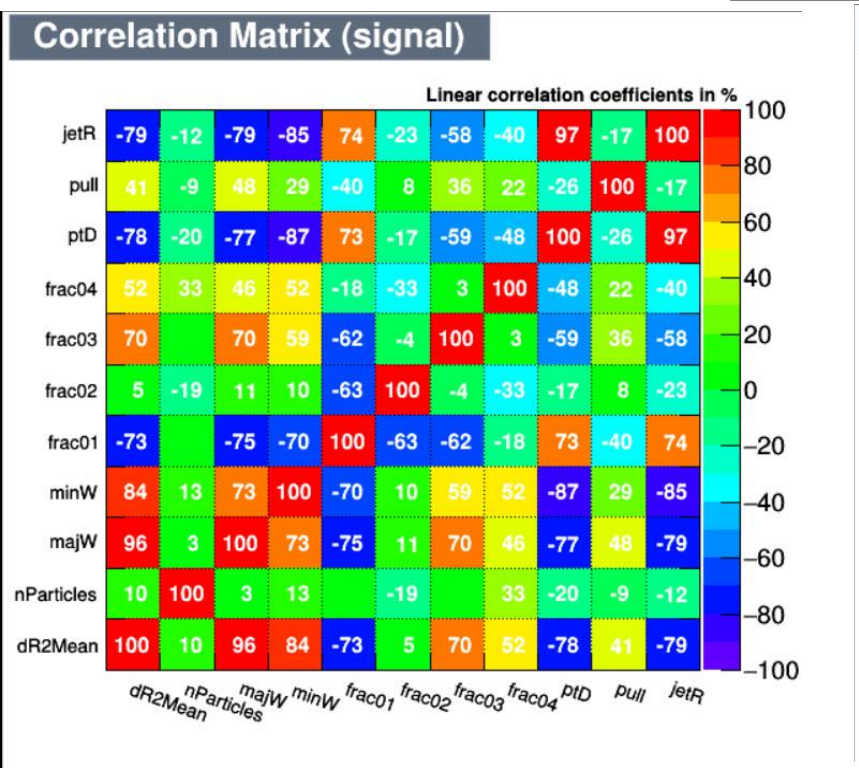
ΠΙΝΑΚΑΣ 2

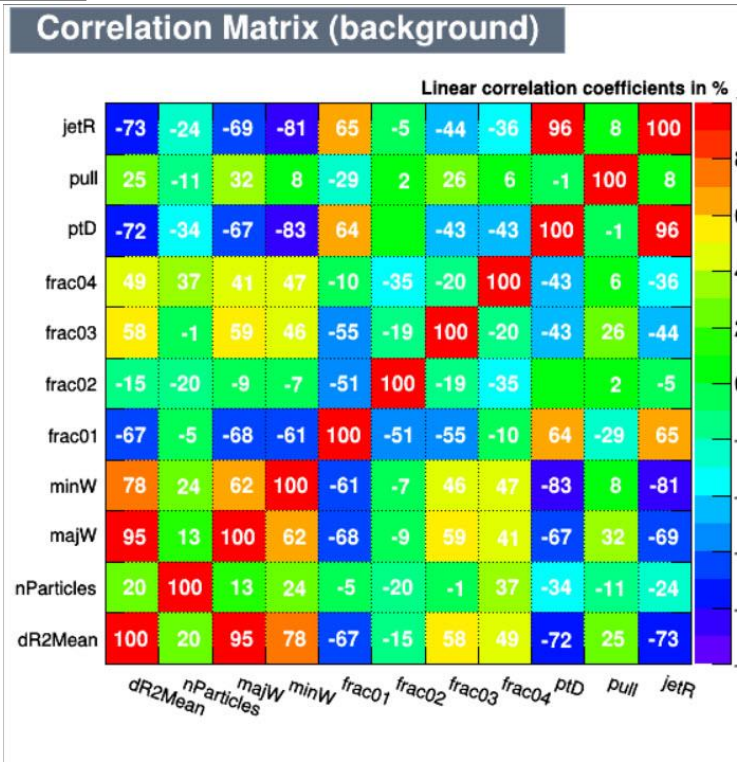
Eta3to5	***total time(sec)***	***(min)***	***weights(bytes)***	***(MB)***	***ROC_curve***
Fisher	0,33	0	14562	0,0	0,689
Likelihood	1,63	0	3726016	3,6	0,664
KNN500	1745,46	29	31313463	29,9	0,687
NN(BP)	578,74	10	40182	0,0	0,691
NN(GA)	/	/	/	/	/
NN(BFGS)	2230,73	37	40194	0,0	0,691
NN(BP)_N-6	385,54	6	34231	0,0	0,691
NN(BFGS)_N-6	1400,53	23	34225	0,0	0,691
NN(BP)_N+12	965,16	16	52201	0,0	0,691
NN(BFGS)_N+12	3871,14	65	52213	0,0	0,691
NN(BP)_Sequential	583,76	10	40183	0,0	0,691
NN(BP)_Batch	523,75	9	40224	0,0	0,393
NN_N-9,N-9	344,51	6	32397	0,0	0,69
NN_N,N-1	938,05	16	49208	0,0	0,691
BDT-100	202,90	3	2105050	2,0	0,687
SVM	/	/	/	/	/

ΠΑΡΑΤΗΡΗΣΕΙΣ :

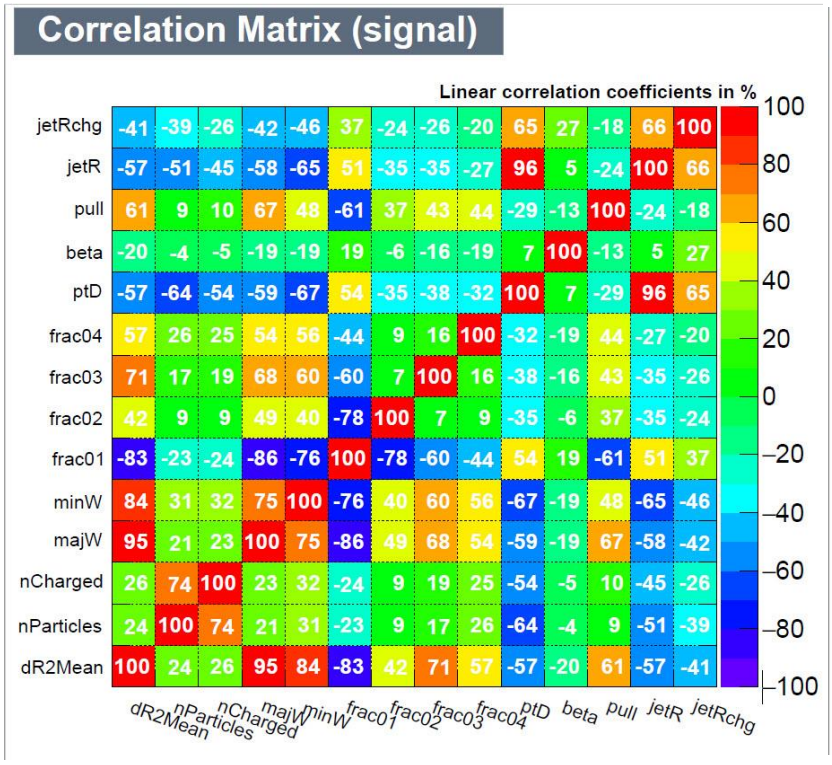
- Μεταξύ των δύο πινάκων, ο πίνακας 1 που αναφέρεται στην περιοχή 0<η<2,5 έχει καλύτερες επιδόσεις (area under the curve ROC) από τον 2, πράγμα αναμενόμενο καθώς στην περιοχή 3<η<5 δεν υπάρχει η μεταβλητή beta.
- Αναλύοντας τους δύο πίνακες είναι φανερό πως οι γραμμικές μέθοδοι είναι αρκετά αποτελεσματικές, παράγουν λίγα βάρη και απαιτούν ελάχιστο χρόνο εκπαίδευσης.
- Στο νευρωνικό δίκτυο από τις τρεις μεθόδους εκπαίδευσης (Backpropagation-BP, Genetic Algorithm-GA και Broyden-Fletcher-Goldfarb-Shannon- BFGS) η αποτελεσματικότερη προέκυψε η BP, ενώ από τους τρόπους λειτουργίας της (batch και sequential) αποτελεσματικότερη απεδείχθη η ακολουθιακή (seq.)
- Στο σημείο αυτό πρέπει να αναφερθεί πως όπως φαίνεται και στους πίνακες δεν υπήρξε ιδιαίτερο νόημα στην βαθύτερη ανάλυση της αρχιτεκτονικής των νευρωνικών δικτύων (deep learning) καθώς όλα οδηγούν στο ίδιο αποτέλεσμα. Το γεγονός αυτό μπορεί να εξηγηθεί από τους πίνακες συσχετίσεων του σήματος και του υποβάθρου στην επόμενη σελίδα, όπου παρατηρούμε πως δεν διαφοροποιούνται σημαντικά.
- Το Boosted Decision Tree - BDT αποδεικνύεται πως είναι μία αρκετά χρήσιμη μέθοδος για το συγκεκριμένο πρόβλημα καθώς με ελάχιστο χρόνο και μέτρια ποσότητα παραγωγής βαρών μπορεί να αποδώσει αρκετά καλά, είτε με την συνάρτηση ομοιογένειας gini index είτε με την cross entropy.

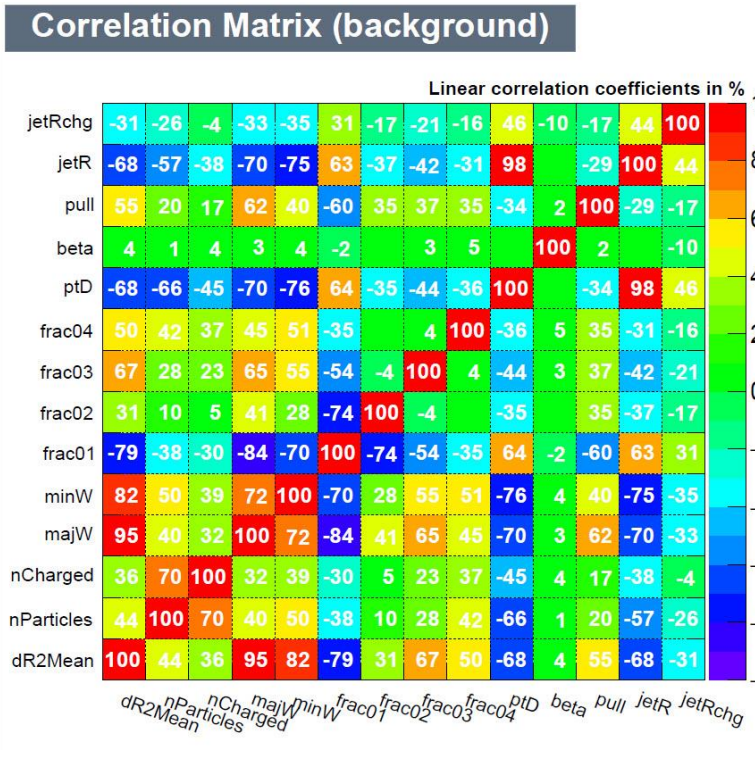
Eta 3 to 5





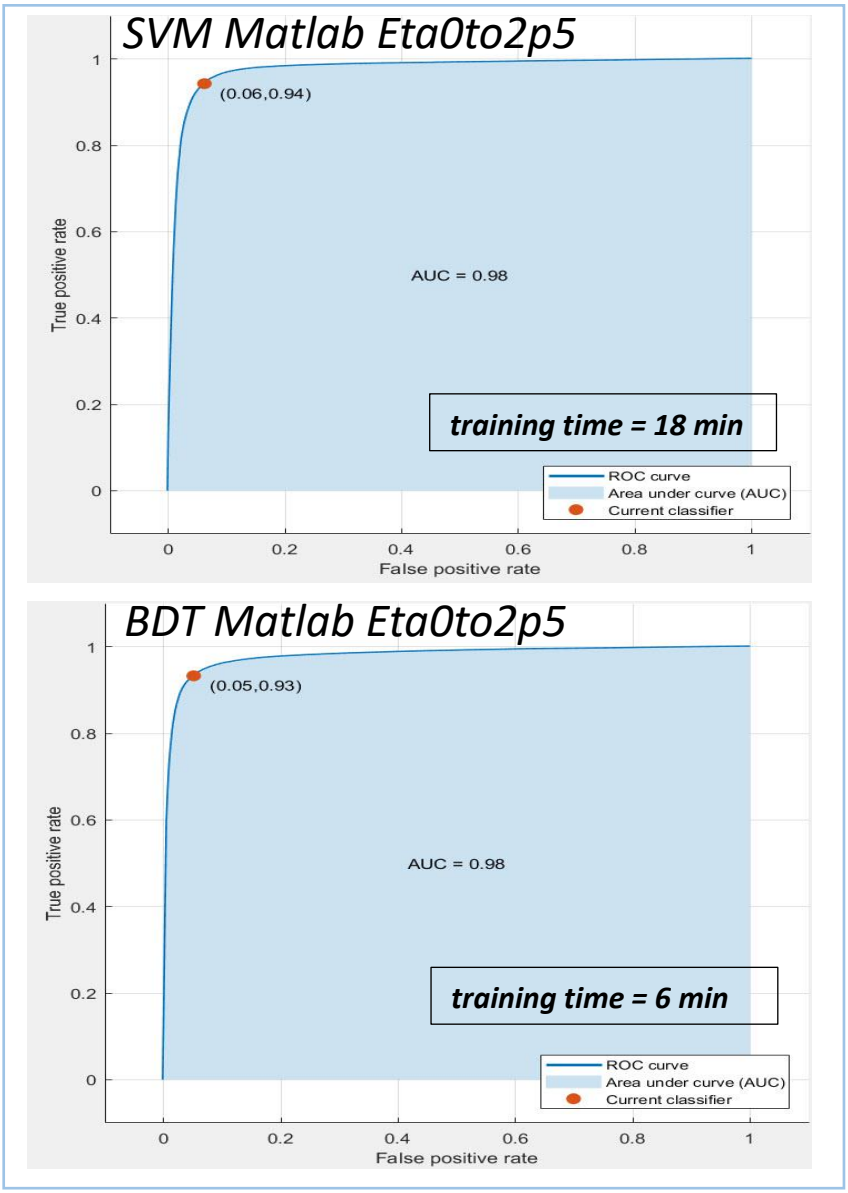
Eta 0 to 2.5





4.7 ΑΠΟΤΕΛΕΣΜΑΤΑ MATLAB

Όπως παρατηρούμε στους συγκεντρωτικούς πίνακες αποτελεσμάτων η μέθοδος SVM δεν έτρεχε με το TMVA του ROOT. Αυτό αποτέλεσε την κύρια αφορμή για την χρήση και του MATLAB στο πρόβλημα, ώστε να γίνει μία διερεύνηση. Συνεπώς το πρώτο βήμα ήταν η μετατροπή των δεδομένων από μορφή δέντρου σε πίνακα, το οποίο έγινε με κώδικα που δίνεται στο παράρτημα. Επειδή η SVM μέθοδος αποδείχθηκε ιδιαιτερα χρονοβόρα, ο τελικός πίνακας των δεδομένων εισαγωγής μειώθηκε στις τριάντα χιλιάδες μετρήσεις από τις διακόσιες χιλιάδες που δοκιμάστηκαν αρχικά. Τότε η μέθοδος εφαρμόστηκε με επιτυχία και τα αποτελέσματα δίνονται στην καμπύλη ROC παρακάτω. Επειδή η εφαρμογή του SVM στο MATLAB έγινε σε διαφορετικό υπολογιστή με καλύτερες δυνατότητες, δίνεται ως μέτρο σύγκρισης και η απόδοση ενός BDT στο ίδιο περιορισμένο δείγμα.



4.8 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

1. Στο συγκεκριμένο πρόβλημα για η < 2.5 έχουμε εξαιρετικό διαχωρισμό των prompt από τα pileup jets ενώ για η > 2.5 έχουμε ασθενέστερη ταξινόμηση λόγω έλλειψης πληροφορίας.

2. Η επιλογή έστω και μιας μεταβλητής με μεγάλη διαχωριστική ικανότητα βελτιώνει σημαντικά την απόδοση όλων των ταξινομητών

3. Όταν οι ταξινομητές έχουν παρόμοιες επιδόσεις, επιλέγουμε τους γραμμικούς καθώς κερδίζουμε σημαντικό χρόνο και χώρο σε MB

4. Εάν οι συσχετίσεις των μεταβλητών στο σήμα και στο υπόβαθρο δεν διαφέρουν αρκετά τότε το πρόβλημα δεν επιδέχεται βαθύτερη μάθηση (deep learning)

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα 1

Παράρτημα 2

Παράρτημα 3

Παράρτημα 4

Παράρτημα 5

Παράρτημα 6

Παράρτημα 7

ΠΑΡΑΡΤΗΜΑ 1

Στο παράρτημα αυτό δίνεται ο βασικός κώδικας (σε C++) με βάση τον οποίο έγινε η εκπαίδευση των αλγόριθμων ταξινόμησης. Επειδή έγινε χρήση πολλών μεθόδων με διαφορετικές παραμέτρους κάθε φορά, εφαρμόστηκε η τακτική: οι μέθοδοι που ήδη εφαρμόστηκαν να δηλώνονται ως σχόλια, ώστε να κρατούνται ως ιστορικό.

```
#include "TMVA/Factory.h"
#include "TMVA/DataLoader.h"
#include "TFile.h"
#include "TTree.h"

using namespace TMVA;
using namespace TMath;

void trainPUID(TString ETA);
void trainPUIDAll();

void trainPUIDAll()
{
  trainPUID("Eta0to2p5");
  trainPUID("Eta2p5to2p75");
  trainPUID("Eta2p75to3");
  trainPUID("Eta3to5");
}

void trainPUID(TString ETA)
{
  TFile *inf = TFile::Open("pileupID_Train.root");
  TTree *promptTree  = (TTree*)inf->Get("Prompt"+ETA);
  TTree *pileupTree = (TTree*)inf->Get("Pileup"+ETA);



  TFile *outf = new
TFile("pileupID_mva_file15_"+ETA+".root","RECREATE");
  TMVA::Factory* factory = new
TMVA::Factory("pileupID_"+ETA,outf,"!V:!Silent:Color:DrawProgressBar:
Transformations=I;G:AnalysisType=Classification");

  TMVA::DataLoader *dataloader = new TMVA::DataLoader("PUID");

  TTree *promptTreeTrain  = (TTree*)promptTree->CloneTree(100000);
  TTree *pileupTreeTrain = (TTree*)pileupTree->CloneTree(100000);

  dataloader->AddSignalTree(promptTreeTrain);
  dataloader->AddBackgroundTree(pileupTreeTrain);

  int N_Q(promptTreeTrain->GetEntries());
  int N_P(pileupTreeTrain->GetEntries());
  int N = TMath::Min(TMath::Min(N_Q,N_P),50000);

  const int NVARC = 14;
  TString VARC[NVARC] = {
      "dR2Mean",
      "nParticles",
```

```
      "nCharged",
      "majW",
      "minW",
      "frac01",
      "frac02",
      "frac03",
      "frac04",
      "ptD",
      "beta",
      "pull",
      "jetR",
      "jetRchg"
  };
  char TYPEC[NVARC] =
{'F','I','I','F','F','F','F','F','F','F','F','F','F','F'};

  const int NVARF = 11;
  TString VARF[NVARF] = {
      "dR2Mean",
      "nParticles",
      "majW",
      "minW",
      "frac01",
      "frac02",
      "frac03",
      "frac04",
      "ptD",
      "pull",
      "jetR"
  };
  char TYPEF[NVARC] = {'F','I','F','F','F','F','F','F','F','F','F'};

  if (ETA == "Eta3to5") {
    for(int i=0;i<NVARF;i++) {
      dataloader->AddVariable(VARF[i],TYPEF[i]);
    }
  }
  else {
    for(int i=0;i<NVARC;i++) {
      dataloader->AddVariable(VARC[i],TYPEC[i]);
    }
  }

  dataloader->AddSpectator("jetPt",'F');
  dataloader->AddSpectator("jetEta",'F');

  dataloader-
>PrepareTrainingAndTestTree("","SplitMode=Random:NormMode=NumEvents:!
V");

//file1
//factory->BookMethod(dataloader, TMVA::Types::kFisher,"Fisher");
//factory->BookMethod(dataloader, TMVA::Types::kLikelihood,
"Likelihood");

//other
//factory->BookMethod(dataloader, TMVA::Types::kSVM,"SVM");//failure
//factory->BookMethod(dataloader, TMVA::Types::kLD,"LD");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN10","nkNN=10");
```

```
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN20","nkNN=20");

//file2
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN30","nkNN=30");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN50","nkNN=50");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN70","nkNN=70");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN100","nkNN=100");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN200","nkNN=200");
//factory->BookMethod(dataloader,
TMVA::Types::kKNN,"kNN500","nkNN=500");

//file3
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP1","TrainingMethod=BP:HiddenLayers=N:NCycles=60
0:VarTransform=Norm");//
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP2","TrainingMethod=BFGS:HiddenLayers=N:NCycles=
600:VarTransform=Norm");//

//file4
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP3","TrainingMethod=BP:HiddenLayers=N,N-
1:NCycles=500:VarTransform=Norm");
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP4","TrainingMethod=BFGS:HiddenLayers=N,N-
1:NCycles=600:VarTransform=Norm");

//file5
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP5","TrainingMethod=BP:HiddenLayers=N-
6:NCycles=600:VarTransform=Norm");
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP6","TrainingMethod=BFGS:HiddenLayers=N-
6:NCycles=600:VarTransform=Norm");
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP7","TrainingMethod=BP:HiddenLayers=N+12:NCycles
=600:VarTransform=Norm");
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP8","TrainingMethod=BFGS:HiddenLayers=N+12:NCycl
es=600:VarTransform=Norm");

//file6
//factory-
>BookMethod(dataloader,TMVA::Types::kMLP,"MLP9","TrainingMethod=BP:BP
Mode=sequential:HiddenLayers=N:NCycles=600:VarTransform=Norm");
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP10","TrainingMethod=BP:BPMode=batch:HiddenLayer
s=N:NCycles=600:VarTransform=Norm");

//file7
//factory-
>BookMethod(dataloader,TMVA::Types::kMLP,"MLP11","TrainingMethod=BP:H
iddenLayers=N-9,N-9:NCycles=600:VarTransform=Norm");
```

```
//factory->BookMethod(dataloader,
TMVA::Types::kMLP,"MLP12","TrainingMethod=BP:HiddenLayers=N-
1,N:NCycles=600:VarTransform=Norm");

//file9
//factory->BookMethod(dataloader, TMVA::Types::kBDT,"BDT1");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT2","NTrees=200");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT3","BoostType=Grad");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT4","BoostType=Grad:Shrinkage=0.1");

//file10_AdaBoost_Learning_Rate_Beta
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT5","AdaBoostBeta=0.1");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT6","AdaBoostBeta=0.2");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT7","AdaBoostBeta=0.3");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT8","AdaBoostBeta=0.4");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT9","AdaBoostBeta=0.5");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT10","AdaBoostBeta=0.6");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT11","AdaBoostBeta=0.7");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT12","AdaBoostBeta=0.8");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT13","AdaBoostBeta=0.9");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT14","AdaBoostBeta=1.0");

//file11_GradBoost_Shrinkage
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT15","BoostType=Grad:Shrinkage=0.1");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT16","BoostType=Grad:Shrinkage=0.2");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT17","BoostType=Grad:Shrinkage=0.3");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT18","BoostType=Grad:Shrinkage=0.4");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT19","BoostType=Grad:Shrinkage=0.5");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT20","BoostType=Grad:Shrinkage=0.6");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT21","BoostType=Grad:Shrinkage=0.7");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT22","BoostType=Grad:Shrinkage=0.8");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT23","BoostType=Grad:Shrinkage=0.9");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT24","BoostType=Grad:Shrinkage=1.0");

//file12_SEPERATION_TYPE
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT_ST1","SeparationType=CrossEntropy:BoostType=Gr
ad:Shrinkage=0.1");
```

```
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT_ST2","SeparationType=GiniIndex:BoostType=Grad:
Shrinkage=0.1");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT_ST3","SeparationType=CrossEntropy:AdaBoostBeta
=0.1");
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT_ST4","SeparationType=GiniIndex:AdaBoostBeta=0.
1");

//file13_doboostmonitor
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT","SeparationType=GiniIndex:AdaBoostBeta=0.1:Do
BoostMonitor=True");

//file14_training_removing_vars_TRM
//factory->BookMethod(dataloader,
TMVA::Types::kBDT,"BDT_TRM1","AdaBoostBeta=0.1:NTrees=150");

//file15_Boosted
factory->BookMethod(dataloader,
TMVA::Types::kFisher,"BoostedFisher","Boost_AdaBoostBeta=0.1:Boost_Nu
m=200");
factory->BookMethod(dataloader,
TMVA::Types::kLikelihood,"BoostedLikelihood","Boost_AdaBoostBeta=0.1:
Boost_Num=200");



  factory->TrainAllMethods();
  factory->TestAllMethods();
  factory->EvaluateAllMethods();
  outf->Close();
  delete factory;
  delete dataloader;
}
```

ΠΑΡΑΡΤΗΜΑ 2

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που δημιουργεί τα ιστογράμματα και τα γεμίζει με τις μεταβλητές του προβλήματος.

```
#include "TFile.h"
#include "TTree.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TLegend.h"
#include <iostream>
using namespace std;
void SaveVariablesNew(TString TREENAME)
{
        TFile *inf = new TFile("pileupID_Train.root","read");
        TTree *tr = (TTree*)inf->Get(TREENAME);
        TFile *outf =new
TFile("histogram_"+TREENAME+".root","recreate");
        int N = tr->GetEntries();
        cout<<"number of entries: "<<N<<endl;
        int nParticles,nCharged;
        float
rho,jetPt,jetEta,dR2Mean,majW,minW,frac01,frac02,frac03,frac04,

ptD,beta,betaStar,pull,jetR,jetRchg,dRMatch,refdrjt;

        const int NVAR = 20;
        TH1F *h[NVAR];
        TCanvas *can[NVAR];
        TString VAR_NAME[NVAR] =
{"nParticles","nCharged","rho","jetPt","jetEta","dR2Mean","majW","min
W",

"frac01","frac02","frac03","frac04","ptD","beta","betaStar",

"pull","jetR","jetRchg","dRMatch","refdrjt"};
        int NBINS[NVAR] =
{40,30,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,10
0,100,100};
        float XMIN[NVAR] = {0,0,0,0,-
3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
        float XMAX[NVAR] =
{40,30,40,100,3,0.07,0.25,0.18,1,1,1,0.5,1,1,1,0.02,1,1,1,500};

        tr->SetBranchAddress("nParticles",&nParticles);
        tr->SetBranchAddress("nCharged",  &nCharged);
        tr->SetBranchAddress("rho",       &rho);
        tr->SetBranchAddress("jetPt",     &jetPt);
        tr->SetBranchAddress("jetEta",    &jetEta);
        tr->SetBranchAddress("dR2Mean",   &dR2Mean);
        tr->SetBranchAddress("majW",      &majW);
        tr->SetBranchAddress("minW",      &minW);
        tr->SetBranchAddress("frac01",    &frac01);
        tr->SetBranchAddress("frac02",    &frac02);
        tr->SetBranchAddress("frac03",    &frac03);
        tr->SetBranchAddress("frac04",    &frac04);
        tr->SetBranchAddress("ptD",       &ptD);
```

```
        tr->SetBranchAddress("beta",       &beta);
        tr->SetBranchAddress("betaStar",   &betaStar);
        tr->SetBranchAddress("pull",       &pull);
        tr->SetBranchAddress("jetR",       &jetR);
        tr->SetBranchAddress("jetRchg",    &jetRchg);
        tr->SetBranchAddress("dRMatch",    &dRMatch);
        tr->SetBranchAddress("refdrjt",    &refdrjt);

        TString title;
        //create histograms
        for(int ivar=0;ivar<NVAR;ivar++) {
          title = "h_"+VAR_NAME[ivar];
          h[ivar] = new
TH1F(title,title,NBINS[ivar],XMIN[ivar],XMAX[ivar]);
        }

        // fill histograms from this tree
        for(int i=0;i<N;i++) {
          tr->GetEntry(i);
          float var[NVAR] =
{(float)nParticles,(float)nCharged,rho,jetPt,jetEta,dR2Mean,majW,minW
,frac01,frac02,frac03,frac04,

ptD,beta,betaStar,pull,jetR,jetRchg,dRMatch,refdrjt};
          for(int ivar=0;ivar<NVAR;ivar++) {
            h[ivar]->Fill(var[ivar]);
          }

        }

        outf->Write();
        outf->Close();
}
```

ΠΑΡΑΡΤΗΜΑ 3

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που απεικονίζει την ίδια μεταβλητή (για Eta0to2p5) από διαφορετικά δέντρα στο ίδιο ιστόγραμμα.

```
#include "TFile.h"
#include "TTree.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TLegend.h"
#include <iostream>
using namespace std;
//draw the same variables from different trees on the same canvas.


void DrawSameVars_Eta0to2p5(TString VAR_NAME)
{
        const int NVAR = 2;
        TH1F  *h[NVAR];
        TFile *inf[NVAR];

        inf[0] = new TFile("histogram_PromptEta0to2p5.root");
        inf[1] = new TFile("histogram_PileupEta0to2p5.root");


     for(int i=0;i<2;i++) {
        h[i]=(TH1F*)inf[i]->Get("h_"+VAR_NAME);
        Double_t norm = 1.0;
        h[i]->Scale(norm/h[i]->Integral());
        }

        int i1 = h[0]->GetMaximumBin();
        float max1 = h[0]->GetBinContent(i1);
        int i2 = h[1]->GetMaximumBin();
        float max2 = h[1]->GetBinContent(i2);
        float max_total = TMath::Max(max1,max2);

        h[0]->SetLineColor(kRed);
        h[1]->SetLineColor(kBlue);
        TString title;
        title = "histo_"+VAR_NAME;
        float Y_edge = 1.1*max_total;

        TCanvas *can = new
TCanvas("Plot_"+VAR_NAME,"Plot_"+VAR_NAME,900,600);
        h[0]->GetXaxis()->SetTitle(VAR_NAME);
        h[0]->GetYaxis()->SetRangeUser(0,Y_edge);
        h[0]->Draw("hist");
        h[1]->Draw("same hist");

        TLegend *leg = new TLegend(0.1,0.9,0.7,1.0);
        leg->AddEntry(h[0],"Prompt","L");
        leg->AddEntry(h[1],"Pileup","L");
        leg->Draw();
        can->Print(TString(can->GetName())+".png");


}
```

ΠΑΡΑΡΤΗΜΑ 4

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που απεικονίζει την ίδια μεταβλητή (για Eta2p5to2p75) από διαφορετικά δέντρα στο ίδιο ιστόγραμμα.

```
#include "TFile.h"
#include "TTree.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TLegend.h"
#include <iostream>
using namespace std;
//draw the same variables from different trees on the same canvas.


void DrawSameVars_Eta2p5to2p75(TString VAR_NAME)
{
        const int NVAR = 2;
        TH1F  *h[NVAR];
        TFile *inf[NVAR];

        inf[0] = new TFile("histogram_PromptEta2p5to2p75.root");
        inf[1] = new TFile("histogram_PileupEta2p5to2p75.root");


     for(int i=0;i<2;i++) {
        h[i]=(TH1F*)inf[i]->Get("h_"+VAR_NAME);
        Double_t norm = 1.0;
        h[i]->Scale(norm/h[i]->Integral());
        }

        int i1 = h[0]->GetMaximumBin();
        float max1 = h[0]->GetBinContent(i1);
        int i2 = h[1]->GetMaximumBin();
        float max2 = h[1]->GetBinContent(i2);
        float max_total = TMath::Max(max1,max2);

        h[0]->SetLineColor(kRed);
        h[1]->SetLineColor(kBlue);
        TString title;
        title = "histo_"+VAR_NAME;
        float Y_edge = 1.1*max_total;

        TCanvas *can = new
TCanvas("Plot_"+VAR_NAME,"Plot_"+VAR_NAME,900,600);
        h[0]->GetXaxis()->SetTitle(VAR_NAME);
        h[0]->GetYaxis()->SetRangeUser(0,Y_edge);
        h[0]->Draw("hist");
        h[1]->Draw("same hist");

        TLegend *leg = new TLegend(0.1,0.9,0.7,1.0);
        leg->AddEntry(h[0],"Prompt","L");
        leg->AddEntry(h[1],"Pileup","L");
        leg->Draw();
        can->Print(TString(can->GetName())+".png");

}
```

ΠΑΡΑΡΤΗΜΑ 5

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που απεικονίζει την ίδια μεταβλητή (για Eta2p75to3) από διαφορετικά δέντρα στο ίδιο ιστόγραμμα.

```
#include "TFile.h"
#include "TTree.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TLegend.h"
#include <iostream>
using namespace std;
//draw the same variables from different trees on the same canvas.


void DrawSameVars_Eta2p75to3(TString VAR_NAME)
{
        const int NVAR = 2;
        TH1F  *h[NVAR];
        TFile *inf[NVAR];

        inf[0] = new TFile("histogram_PromptEta2p5to2p75.root");
        inf[1] = new TFile("histogram_PileupEta2p5to2p75.root");


     for(int i=0;i<2;i++) {
        h[i]=(TH1F*)inf[i]->Get("h_"+VAR_NAME);
        Double_t norm = 1.0;
        h[i]->Scale(norm/h[i]->Integral());
        }

        int i1 = h[0]->GetMaximumBin();
        float max1 = h[0]->GetBinContent(i1);
        int i2 = h[1]->GetMaximumBin();
        float max2 = h[1]->GetBinContent(i2);
        float max_total = TMath::Max(max1,max2);

        h[0]->SetLineColor(kRed);
        h[1]->SetLineColor(kBlue);
        TString title;
        title = "histo_"+VAR_NAME;
        float Y_edge = 1.1*max_total;

        TCanvas *can = new
TCanvas("Plot_"+VAR_NAME,"Plot_"+VAR_NAME,900,600);
        h[0]->GetXaxis()->SetTitle(VAR_NAME);
        h[0]->GetYaxis()->SetRangeUser(0,Y_edge);
        h[0]->Draw("hist");
        h[1]->Draw("same hist");

        TLegend *leg = new TLegend(0.1,0.9,0.7,1.0);
        leg->AddEntry(h[0],"Prompt","L");
        leg->AddEntry(h[1],"Pileup","L");
        leg->Draw();
        can->Print(TString(can->GetName())+".png");

}
```

ΠΑΡΑΡΤΗΜΑ 6

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που απεικονίζει την ίδια μεταβλητή (για Eτα3το5) από διαφορετικά δέντρα στο ίδιο ιστόγραμμα.

```
#include "TFile.h"
#include "TTree.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TLegend.h"
#include <iostream>
using namespace std;
//draw the same variables from different trees on the same canvas.

void DrawSameVars_Eta3to5(TString VAR_NAME)
{
        const int NVAR = 2;
        TH1F  *h[NVAR];
        TFile *inf[NVAR];

        inf[0] = new TFile("histogram_PromptEta2p5to2p75.root");
        inf[1] = new TFile("histogram_PileupEta2p5to2p75.root");

     for(int i=0;i<2;i++) {
        h[i]=(TH1F*)inf[i]->Get("h_"+VAR_NAME);
        Double_t norm = 1.0;
        h[i]->Scale(norm/h[i]->Integral());
        }

        int i1 = h[0]->GetMaximumBin();
        float max1 = h[0]->GetBinContent(i1);
        int i2 = h[1]->GetMaximumBin();
        float max2 = h[1]->GetBinContent(i2);
        float max_total = TMath::Max(max1,max2);

        h[0]->SetLineColor(kRed);
        h[1]->SetLineColor(kBlue);
        TString title;
        title = "histo_"+VAR_NAME;
        float Y_edge = 1.1*max_total;

        TCanvas *can = new
TCanvas("Plot_"+VAR_NAME,"Plot_"+VAR_NAME,900,600);
        h[0]->GetXaxis()->SetTitle(VAR_NAME);
        h[0]->GetYaxis()->SetRangeUser(0,Y_edge);
        h[0]->Draw("hist");
        h[1]->Draw("same hist");

        TLegend *leg = new TLegend(0.1,0.9,0.7,1.0);
        leg->AddEntry(h[0],"Prompt","L");
        leg->AddEntry(h[1],"Pileup","L");
        leg->Draw();

        can->Print(TString(can->GetName())+".png");
}
```

ΠΑΡΑΡΤΗΜΑ 7

Στο παράρτημα αυτό δίνεται ο κώδικας σε C++ που έκανε την μετατροπή των δεδομένων εισαγωγής από μορφή δέντρου σε πίνακα.

```
#include "TFile.h"
#include "TTree.h"
#include "TMath.h"
#include <iostream>
#include <fstream>
using namespace std;

void tree_to_matrix(TString ETA);
void tree_to_matrixAll();

void tree_to_matrixAll()
{
        tree_to_matrix("Eta0to2p5");
        tree_to_matrix("Eta2p5to2p75");
        tree_to_matrix("Eta2p75to3");
        tree_to_matrix("Eta3to5");
}

void tree_to_matrix(TString ETA)
{
        TFile *inf = new TFile("pileupID_Train.root","read");
        TTree *tr1 = (TTree*)inf->Get("Prompt"+ETA);
        TTree *tr2 = (TTree*)inf->Get("Pileup"+ETA);
        int N = tr1->GetEntries();
        int M = tr2->GetEntries();
         int NEV = TMath::Min(100000,TMath::Min(M,N));
        cout<<"number of entries: "<< N + M <<endl;
        float
dR2Mean,majW,minW,frac01,frac02,frac03,frac04,ptD,beta,pull,jetR,jetR
chg;
        int nParticles,nCharged;

 if (ETA == "Eta3to5")

{
        tr1->SetBranchAddress("dR2Mean"   ,&dR2Mean);
        tr1->SetBranchAddress("nParticles",&nParticles);
        tr1->SetBranchAddress("majW"      ,&majW);
        tr1->SetBranchAddress("minW"      ,&minW);
        tr1->SetBranchAddress("frac01"    ,&frac01);
        tr1->SetBranchAddress("frac02"    ,&frac02);
        tr1->SetBranchAddress("frac03"    ,&frac03);
        tr1->SetBranchAddress("frac04"    ,&frac04);
        tr1->SetBranchAddress("ptD"       ,&ptD);
        tr1->SetBranchAddress("pull"      ,&pull);
        tr1->SetBranchAddress("jetR"      ,&jetR);

        tr2->SetBranchAddress("dR2Mean"   ,&dR2Mean);
        tr2->SetBranchAddress("nParticles",&nParticles);
        tr2->SetBranchAddress("majW"      ,&majW);
        tr2->SetBranchAddress("minW"      ,&minW);
        tr2->SetBranchAddress("frac01"    ,&frac01);
```

```
        tr2->SetBranchAddress("frac02"    ,&frac02);
        tr2->SetBranchAddress("frac03"    ,&frac03);
        tr2->SetBranchAddress("frac04"    ,&frac04);
        tr2->SetBranchAddress("ptD"       ,&ptD);
        tr2->SetBranchAddress("pull"      ,&pull);
        tr2->SetBranchAddress("jetR"      ,&jetR);
 }

 else{

        tr1->SetBranchAddress("dR2Mean"   ,&dR2Mean);
        tr1->SetBranchAddress("nParticles",&nParticles);
        tr1->SetBranchAddress("nCharged"  ,&nCharged);
        tr1->SetBranchAddress("majW"      ,&majW);
        tr1->SetBranchAddress("minW"      ,&minW);
        tr1->SetBranchAddress("frac01"    ,&frac01);
        tr1->SetBranchAddress("frac02"    ,&frac02);
        tr1->SetBranchAddress("frac03"    ,&frac03);
        tr1->SetBranchAddress("frac04"    ,&frac04);
        tr1->SetBranchAddress("ptD"       ,&ptD);
        tr1->SetBranchAddress("beta"      ,&beta);
        tr1->SetBranchAddress("pull"      ,&pull);
        tr1->SetBranchAddress("jetR"      ,&jetR);
        tr1->SetBranchAddress("jetRchg"   ,&jetRchg);

        tr2->SetBranchAddress("dR2Mean"   ,&dR2Mean);
        tr2->SetBranchAddress("nParticles",&nParticles);
        tr2->SetBranchAddress("nCharged"  ,&nCharged);
        tr2->SetBranchAddress("majW"      ,&majW);
        tr2->SetBranchAddress("minW"      ,&minW);
        tr2->SetBranchAddress("frac01"    ,&frac01);
        tr2->SetBranchAddress("frac02"    ,&frac02);
        tr2->SetBranchAddress("frac03"    ,&frac03);
        tr2->SetBranchAddress("frac04"    ,&frac04);
        tr2->SetBranchAddress("ptD"       ,&ptD);
        tr2->SetBranchAddress("beta"      ,&beta);
        tr2->SetBranchAddress("pull"      ,&pull);
        tr2->SetBranchAddress("jetR"      ,&jetR);
        tr2->SetBranchAddress("jetRchg"   ,&jetRchg);
 }

        ofstream of1,of2;
        of1.open ("ETA_inputs_short.txt");
        of2.open ("ETA_targets_short.txt");
                for(int i=0;i<NEV;i++) {
                tr1->GetEntry(i);
                of1<<dR2Mean<<" "<<nParticles<<" "<<majW<<"
"<<minW<<" "<<frac01<<" "<<frac02<<" "<<frac03<<" "<<frac04<<"
"<<ptD<<" "<<pull<<" "<<jetR<<endl;
                of2<<1<<" "<<0<<endl;
                                        }
                for(int i=0;i<NEV;i++) {
                tr2->GetEntry(i);
                of1<<dR2Mean<<" "<<nParticles<<" "<<majW<<"
"<<minW<<" "<<frac01<<" "<<frac02<<" "<<frac03<<" "<<frac04<<"
"<<ptD<<" "<<pull<<" "<<jetR<<endl;
                of2<<0<<" "<<1<<endl;
                                        }
        of1.close();
        of2.close();
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Christopher M. Bishop ‘’**Neural Networks for Pattern Recognition**’’

[2] Christopher M. Bishop **‘’Pattern Recognition and Machine Learning**’’

[3] Vladimir Vapnik ’’**The Nature of Statistical Learning Theory**’’

[4] Kevin P.Murphy_’’**Machine Learning, A Probabilistic Perspective**’’

[5] Tom M. Mitchell ‘’**Machine Learning**’’

[6] Patrick Henry Winston ‘’**Artificial Intelligence**’’

[7] S. Theodoridis, K. Koutroumbas “**Pattern recognition**”

[8] Θ.Αλεξόπουλος, Α.Τζαμαριουδάκη “**Στατιστική Αναγνώριση Προτύπων**”

[9] Κ.Κουσουρής Σημειώσεις του μαθήματος “**Αναγνώριση Προτύπων και Νευρωνικά Δίκτυα**”

[10] Γ. Τσιπολίτης Σημειώσεις μαθήματος “**Τεχνολογία Ανιχνευτικών και Επιταχυντικών Διατάξεων**”

[11] Donald H. Perkins “**Εισαγωγή στη Φυσική Υψηλών Ενεργειών**”

[12] Philip Harris, Martina Malberti, Pasquale Musella, Salvatore Rappoccio, and Jan Steggemann “**CMS Analysis Note - Pileup Jet Identification**” CMS AN AN-13-186

[13] The CMS Collaboration ‘’**CMS Physics Analysis Summary - Pileup Jet Identification**’’ CMS PAS JME-13-005

[14] https://**cms.cern/detector**

[15] **TMVA 4** (Toolkit for Multivariate Data Analysis with ROOT) **Users Guide**

[16] **ROOT User's Guide: 6 Release Cycle**

[17] **Mathworks – Documentation** (Matlab support)

[18] http://www.**cplusplus**.com/