ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Τμήμα Πληροφορικής και Επικοινωνιών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατασκευή μοντέλων Data Mining με Γενικευμένα Νευρωνικά Δίκτυα Παλινδρόμησης GRNN σε βάσεις δεδομένων Oracle

Αναστασιάδης Γιώργος

ΑΕΜ 2151

Επιβλέπων: Κόκκινος Ιωάννης

Σέρρες, 2012

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΞΟΡΥΞΗ ΓΝΩΣΗΣ -DATA MINING 2

1.1 ΕΙΣΑΓΩΓΗ 2

1.2 ΒΑΣΙΚΟΙ ΣΤΟΧΟΙ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ 3

1.3 ΜΕΘΟΔΟΙ DATA MINING 4

1.4 ΑΝΑΛΥΣΗ ΕΡΓΑΣΙΩΝ ΕΞΟΡΥΞΗΣ ΓΝΩΣΗΣ 4

1.5 Η KDD ΔΙΑΔΙΚΑΣΊΑ 6

1.6 ΑΛΓΟΡΙΘΜΟΙ ΜΑΘΗΣΗΣ ΚΑΙ ΚΑΤΑΣΚΕΥΗΣ ΜΟΝΤΕΛΩΝ 7

1.7 ΜΕΘΟΔΟΙ ΑΞΙΟΛΟΓΗΣΗΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ 10

1.8 ΟΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ORACLE 14

2 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ 17

2.1 ΕΙΣΑΓΩΓΗ ΣΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ 17

2.2 ΔΟΜΗ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ 19

2.3 ΕΚΠΑΙΔΕΥΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ 21

2.4 BACK PROPAGATION NEURAL NETWORK 22

2.5 PROBABILISTIC NEURAL NETWORKS (PNN) 23

3 GENERAL REGRESSION NEURAL NETWORK 25

3.1 ΑΝΑΛΥΣΗ ΠΑΛΙΝΔΡΟΜΗΣΗΣ-REGRESSION ANALYSIS 25

3.2 ΓΕΝΙΚΕΥΜΕΝΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ 26

3.3 ΜΟΝΤΕΛΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ 27

3.4 ΓΕΝΙΚΗ ΠΑΛΙΝΔΡΟΜΗΣΗ 29

3.5 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ GRNN 31

3.6 ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ GRNN: ΕΠΙΛΟΓΗ ΜΕΤΑΒΛΗΤΗΣ ΣΙΓΜΑ 33

4 ΠΕΡΙΓΡΑΦΗ ΠΕΙΡΑΜΑΤΩΝ 34

4.1 ΠΕΙΡΑΜΑΤΑ ΤΑΞΙΝΟΜΗΣΗΣ (CLASSIFICATION) 34

4.2 ΠΕΙΡΑΜΑΤΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ (REGRESSION) 38

5 ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ 41

6 ΒΙΒΛΙΟΓΡΑΦΙΑ 67

ΠΕΡΙΛΗΨΗ

Το θέμα της πτυχιακής εργασίας είναι: “ Κατασκευή μοντέλων Data Mining με Γενικευμένα Νευρωνικά Δίκτυα Παλινδρόμησης GRNN σε βάσεις δεδομένων Oracle ” Data Mining είναι η τεχνική ανακάλυψης γνώσης από δεδομένα που χρησιμοποιεί δένδρα αποφάσεων, παλινδρόμηση, λογικούς κανόνες, νευρωνικά δίκτυα, στατιστικές μεθόδους διάκρισης ή κοντινότερους γείτονες ή τεχνικές Bayes για να κατατάξει τα δεδομένα σε προκαθορισμένες κλάσεις. Η ανάλυση παλινδρόμησης είναι ένα στατιστικό εργαλείο για τη διερεύνηση των σχέσεων μεταξύ των μεταβλητών. Χρησιμοποιεί πολλές διαφορετικές τεχνικές, συμπεριλαμβανομένων των Νευρωνικών Δικτύων.

Το General Regression Neural Network (GRNN) εκτελεί γενικές και μη γραμμικές παλινδρομήσεις και εκτίμηση εξόδου συνεχούς συνάρτησης. Υλοποιεί τον εκτιμητή παλινδρόμησης Nadaraya-Watson και βασίζεται στους μη παραμετρικούς στατιστικούς εκτιμητές πυκνότητας Parzen kenels. Μπορεί να εκτελέσει διεργασίες παλινδρόμησης και να κατασκευάσει ένα μοντέλο παλινδρόμησης για έρευνα της συσχέτισης μεταξύ μιας εξαρτώμενης μεταβλητής και μιας ή περισσοτέρων ανεξάρτητων μεταβλητών. Η αρχιτεκτονική του GRNN αποτελείται από 4 επίπεδα. Το πρώτο επίπεδο είναι το επίπεδο εισόδου. Αποτελείται από τόσους νευρώνες όσους και τα διανύσματα εισόδου. Το δεύτερο επίπεδο είναι το επίπεδο προτύπων. Σε αυτό το επίπεδο περιέχονται όλα τα δείγματα εκπαίδευσης. Στο τρίτο επίπεδο, που ονομάζεται επίπεδο άθροισης, περιέχεται ο παρονομαστής και ο αριθμητής του εκτιμητή Nadaraya-Watson, Το τέταρτο επίπεδο είναι η έξοδος διαιρείται ο αριθμητής με τον παρονομαστή. Το GRNN μπορεί να λύσει οποιοδήποτε πρόβλημα που μπορεί να λυθεί από ένα στατιστικό μοντέλο παλινδρόμησης χωρίς να πρέπει να ταιριάξει τα δεδομένα εκπαίδευσης σε ένα συγκεκριμένο τύπο παραμετρικού μοντέλου που δίνεται εκ των προτέρων.

Κατά την εκπαίδευση του δικτύου GRNN πρέπει να βρεθεί η άγνωστη παράμετρος σίγμα. Η επιλογή της παραμέτρου αυτής είναι πολύ σημαντική. Για την επιλογή της χρησιμοποιούνται cross-validation και Holdout μέθοδοι. Η παράμετρος πλάτους σίγμα καθορίζει το παράθυρο Parzen όπου με την χρήση του παραθύρου Parzen μπορει να εκτιμηθεί η επιφάνεια regression.

Στην πτυχιακή υλοποιήθηκε το Generalized Regression Nevral Network σε περιβάλλον Borland C++ Builder και επιτρέπει την κατασκευή και μελέτη διαφόρων μοντέλων data mining από Oracle Server. Επίσης υλοποιήθηκε μία μέθοδος εύρεσης βέλτιστων παραμέτρων πλάτους διακύμανσης σ των συναρτήσεων του νευρωνικού δικτύου.

1 ΕΞΟΡΥΞΗ ΓΝΩΣΗΣ -DATA MINING

1.1 Εισαγωγή

Ο όρος εξόρυξη δεδομένων (data mining) αναφέρεται γενικά στην διαδικασία της ημιαυτόματης ανάλυσης μεγάλων βάσεων δεδομένων για εύρεση χρήσιμων μοτίβων. Είναι μία σειρά από τεχνικές που βασίζονται σε ανάπτυξη αλγορίθμων και είναι χρήσιμες σε πολλούς κλάδους όπως οι: οικονομία, βιοστατιστική, δημογραφία, μετεωρολογία και γεωλογία.

Η διαδικασία εξόρυξης γνώσης χρησιμοποιεί δύο συστήματα : (α) την επαλήθευση και (β) την ανακάλυψη. Στον πρώτο σύστημα, γίνεται επαλήθευση των υποθέσεων του χρήστη από το σύστημα ενώ, το δεύτερο σύστημα, βρίσκει νέους κανόνες και πρότυπα μέσα από αυτόνομες διαδικασίες.

Οι μέθοδοι επαλήθευσης ασχολούνται με την εκτίμηση μιας υπόθεσης που προτείνεται από μια εξωτερική πηγή. Αυτές οι μέθοδοι περιλαμβάνουν περισσότερο παραδοσιακές μεθόδους από τη Στατιστική και σχετίζονται λιγότερο με την εξόρυξη δεδομένων απ' ότι οι μέθοδοι ανακάλυψης, οι οποίες εντοπίζουν αυτόματα πρότυπα στα δεδομένα. Το στάδιο της ανακάλυψης χωρίζεται στην περιγραφή και την πρόβλεψη που είναι και οι δύο στόχοι της εξόρυξης δεδομένων. Δηλαδή, η αναγνώριση των προτύπων που επικρατούν σε ένα μεγάλο σύνολο δεδομένων και η δημιουργία προβλέψεων όσον αφορά τη μελλοντική αξία ή συμπεριφορά κάποιων μεταβλητών.

Τα 4 μέρη της ΕΔ είναι η περιγραφική μοντελοποίηση, η μοντελοποίηση πρόβλεψης , η ανάλυση σαφήνειας και η ανίχνευση παρεκτροπών.

Μοντέλο περιγραφής: Στόχος ενός μοντέλου περιγραφής είναι να γίνει περιγραφή όλου του συνόλου δεδομένων ή της διαδικασίας που παράγει τα δεδομένα. Η σημαντικότερη εφαρμογή των περιγραφικών μοντέλων είναι η συσταδοποίηση, η οποία επιχειρεί να βρει ομάδες παρατηρήσεων που είναι κοντά μεταξύ τους ως προς τα χαρακτηριστικά που περιλαμβάνουν.

Μοντέλο Πρόβλεψης: Στοχεύει στη δυνατότητα πρόγνωσης της τιμής μιας μεταβλητής (απόκριση) μέσα από τις τιμές άλλων μεταβλητών (επεξηγηματικές) που είναι γνωστές. Εάν η μεταβλητή απόκρισης είναι ή μπορεί να θεωρηθεί κατηγορική, τότε είμαστε σε θέση να εφαρμόσουμε μια μέθοδο ταξινόμησης. Όμως, αν έχουμε συνεχή απόκριση, τότε προχωράμε σε παλινδρόμηση. Επιθυμώντας να διευκρινίσουμε τη διαφορά μεταξύ ταξινόμησης και συσταδοποίησης, πρέπει να σημειώσουμε ότι στην ταξινόμηση επιχειρείται η περιγραφή μιας λειτουργίας που αντιστοιχεί (ταξινομεί) ένα στοιχείο σε μια εκ των κατηγοριών οι οποίες είναι ήδη προκαθορισμένες.

Ανάλυση Σαφήνειας: χρησιμοποιείται ώστε να ανακαλυφθούν πρότυπα που περιγράφουν χαρακτηριστικά συνάφειας μεταξύ των δεδομένων. Τα πρότυπα αυτά απεικονίζονται συνήθως στα πλαίσια κανόνων συνεπαγωγής (implication rules) ή υποομάδων των χαρακτηριστικών. Η χαρακτηριστικότερη εφαρμογή και η αιτία από την οποία ξεκίνησαν οι κανόνες συνάφειας (ή κανόνες «if – then») είναι η «ανάλυση του καλαθιού αγοράς»(market basket analysis). Άλλες εφαρμογές πραγματοποιούνται στην προώθηση προϊόντων ή στην τοποθέτησή τους στα ράφια καταστημάτων, στη διαχείριση αποθεμάτων κ.λπ. Στους κανόνες συνάφειας δίνεται και εκτίμηση για το πόσο πιθανό να συμβεί αυτή η σχέση αιτίας – αποτελέσματος,

Ανίχνευση Παρεκτροπών: ανήκουν εργασίες εντοπισμού παρατηρήσεων των οποίων τα χαρακτηριστικά διαφέρουν σημαντικά από αυτά του υπόλοιπου συνόλου δεδομένων (έκτροπες παρατηρήσεις ή outliers). Στόχος είναι η υψηλού επιπέδου ανίχνευση πιθανών ανωμαλιών, διατηρώντας όμως χαμηλά ποσοστά λανθασμένης προειδοποίησης. Ως εφαρμογή μπορούμε να αναφέρουμε τον προσδιορισμό απειλής στην έγκριση δανείων ή πιστωτικών καρτών από μια τράπεζα.

1.2 Βασικοί στόχοι εξόρυξης δεδομένων

Η επιστήμη της εξόρυξης δεδομένων στοχεύει να καλύψει το κενό που υπάρχει στην εκμετάλλευση της χρήσιμης πληροφορίας. Πιο συγκεκριμένα, οι βασικοί στόχοι της εξόρυξης δεδομένων είναι οι εξής :

- Το φιλτράρισμα άχρηστων πληροφοριών.
- Η απομόνωση χρήσιμης πληροφορίας και προτύπων δεδομένων.
- Η εύρεση σχέσεων και συσχετίσεων μεταξύ δεδομένων.
- Η ταξινόμηση δεδομένων σε κατηγορίες.
- Η ομαδοποίηση δεδομένων βάσει κοινών ιδιοτήτων.
- Η σύνοψη και η εμφάνιση των επεξεργασμένων πληροφοριών σε μορφή κατανοητή από τον άνθρωπο.

Στην ουσία, η εξόρυξη δεδομένων στοχεύει στην απόσταση έμμεσων, και προηγουμένως αγνώστων, ορθών, κατανοητών και ενδιαφερόντων πληροφοριών από ένα τεράστιο, αταξινόμητο σύνολο δεδομένων.

1.3 Μέθοδοι Data Mining

Οι αλγόριθμοι εξόρυξης γνώσης μπορεί να θεωρηθεί ότι αποτελούνται από τρία μέρη:

Μοντέλο: Ο σκοπός του αλγορίθμου είναι να ταιριάζει το μοντέλο στα δεδομένα

Προτίμηση: πρέπει να χρησιμοποιούνται κάποια κριτήρια για να ταιριάξει ένα μοντέλο έναντι ενός άλλου

Αναζήτηση: Όλοι οι αλγόριθμοι απαιτούν μια τεχνική για να κάνουν αναζήτηση στα δεδομένα

Υπάρχουν δύο είδη τεχνικών εξόρυξης γνώσης:

Α) Προβλεπτικά Μοντέλα (predictive model)

Κάνει πρόβλεψη συμπεριφοράς κάποιων μεταβλητών που οι οποίες βασίζονται στη συμπεριφορά άλλων μεταβλητών. Η πρόβλεψη μπορεί να στηρίζεται σε ιστορικά δεδομένα. Περιλαμβάνει την Κατηγοριοποίηση, Παλινδρόμηση, Ανάλυση Χρονοσειρών, Πρόβλεψη.

Β) Περιγραφικά Μοντέλα (descriptive model)

Βρίσκει πρότυπα που ανήκουν στα δεδομένα και ερευνά τις ιδιότητες των υπό εξέταση δεδομένων, σε αντίθεση με τα προβλεπτικά μοντέλα που προβλέπουν νέες, και εξηγεί τη συμπεριφορά τους. Περιλαμβάνει την Συσταδοποίηση, Παρουσίαση συνόψεων, Κανόνες Συσχετίσεων, Ανακάλυψη ακολουθιών.

1.4 Ανάλυση Εργασιών εξόρυξης γνώσης από τα δεδομένα

1.4.1 Προβλεπτικά Μοντέλα

Κατηγοριοποίηση (classification) Οι αλγόριθμοι ταξινόμησης εφαρμόζονται σε δεδομένα τα οποία έχουν ξανά ταξινομηθεί σε συγκεκριμένες κλάσεις με στόχο την εξαγωγή κανόνων οι οποίοι μπορεί μετέπειτα να χρησιμοποιηθούν για ταξινόμηση νέων δεδομένων στις ίδιες κλάσεις. Αναφέρεται συχνά σαν εποπτευόμενη μάθηση γιατί οι κλάσεις καθορίζονται πριν εξεταστούν τα δεδομένα. Βασίζεται σε ιστορικά στοιχεία όπου φαίνεται η επίδραση των μεταβλητών εισόδου στην μεταβλητή στόχευσης. Ουσιαστικά το σύστημα εκπαιδεύεται από τα ιστορικά στοιχεία

Παλινδρόμηση (regression) Είναι παρόμοια με την ταξινόμηση, με την διαφορά ότι αντιστοιχίζει τα αντικείμενα από ένα σύνολο δεδομένων στην τιμή μίας μεταβλητής πρόβλεψης. Η παλινδρόμηση προϋποθέτει ότι τα σχετικά δεδομένα ταιριάζουν με μερικά γνωστά είδη συνάρτησης και μετά καθορίζει την

καλύτερη συνάρτηση αυτού του είδους που μοντελοποιεί τα δεδομένα που έχουν δοθεί.

Ανάλυση Χρονοσειρών (time series analysis), Αναλύει και προβλέπει γεγονότα που βασίζονται σε χρονολογική σειρά. Η ανάλυση και η μελέτη των δεδομένων αυτών καθώς μεταβάλλονται στο χρόνο γίνεται σε ίσα χρονικά διαστήματα. Το αποτέλεσμα των χρονοσειρών εξάγεται σε ένα διάγραμμα χρονοσειρών. Χρησιμοποιούνται μονάδες μέτρησης απόστασης για να καθορίσουν την ομοιότητα ανάμεσα σε διαφορετικές χρονοσειρές, ακολούθως εξετάζεται η δομή της χρονοσειράς για να την κατηγοριοποίηση. Αφού γίνουν τα πιο πάνω τότε εξάγονται τα διαγράμματα χρονοσειρών όπου μπορούν να χρησιμοποιηθούν για την πρόβλεψη μελλοντικών τιμών.

Πρόβλεψη (prediction): Δίνεται μια τιμή σε μια μελλοντική κατάσταση παρά σε μια τρέχουσα. Εδώ αναφερόμαστε σε ένα είδος εφαρμογής παρά σε μια προσέγγιση μοντελοποίησης.

1.4.2 Περιγραφικά Μοντέλα

Συσταδοποίηση(clustering) Τα δεδομένα που χρησιμοποιούνται για μάθηση δεν είναι ξανά ταξινομημένα. Η τεχνική ομαδοποίησης χωρίζει ουσιαστικά ένα σύνολο εγγραφών σε ομάδες έτσι ώστε οι εγγραφές που βρίσκονται στην ίδια ομάδα να έχουν περισσότερες ομοιότητες μεταξύ τους, με βάση ορισμένα προκαθορισμένα κριτήρια, απ'ότι με εγγραφές άλλων ομάδων. Οι περισσότεροι αλγόριθμοι βασίζονται σε ένα σύνολο επαναλήψεων και σταματούν όταν το μοντέλο συγκλίνει, δηλαδή όταν τα σύνολα κάθε ομαδοποίησης γίνουν διακριτά

Παρουσίαση συνόψεων (summarization) Χρησιμοποιούνται για να χαρακτηριστούν τα περιεχόμενα της βάσης δεδομένων απεικονίζοντας τα δεδομένα σε υποσύνολα τους με συνοδευτικές απλές περιγραφές.

Κανόνες Συσχετίσεων (link analysis) αναφέρεται στη διαδικασία εκείνη της εξόρυξης γνώσης που αποκαλύπτει συσχετίσεις μεταξύ των δεδομένων. Παρέχουν έναν συνοπτικό τρόπο για να εκφραστούν οι ενδεχομένως χρήσιμες πληροφορίες που γίνονται εύκολα κατανοητές από τους τελικούς χρήστες.

Ανακάλυψη Ακολουθιών (sequential analysis) χρησιμοποιείται για να καθοριστούν σειριακά πρότυπα στα δεδομένα. Τα πρότυπα αυτά βασίζονται σε μια χρονική ακολουθία ενεργειών και συσχετίζονται τα δεδομένα που εξάγονται με βάση το χρόνο.

1.5 Η KDD διαδικασία (Knowledge Discovery in Databases)

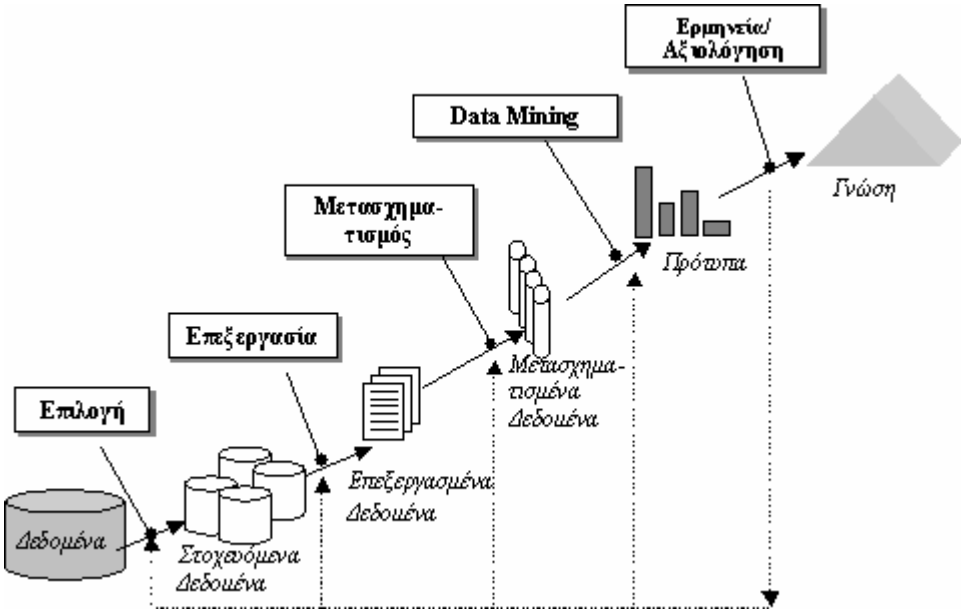
Η Ανακάλυψη Γνώσης μέσα από Βάσεις Δεδομένων (Knowledge Discovery in Databases - KDD) αποτελεί ένα ταχέως αναπτυσσόμενο επιστημονικό πεδίο που αποσκοπεί στην ανακάλυψη κρίσιμων “κρυμμένων” πληροφοριών μέσα από τις βάσεις δεδομένων. Οι εφαρμογές της ανακάλυψης γνώσης αυξάνονται συνεχώς λόγω της έκρηξης της πληροφορίας που παρατηρείται τα τελευταία χρόνια και της ανάγκης για ανακάλυψη χρήσιμων πληροφοριών από την πληθώρα των διαθέσιμων δεδομένων.

Η KDD διαδικασία είναι μια αλληλεπιδραστική και επαναληπτική διαδικασία, η οποία περιλαμβάνει πλήθος βημάτων στα οποία χρειάζεται πολλές φορές να παρέμβει και ο άνθρωπος λαμβάνοντας κρίσιμες αποφάσεις.

Τα βασικά βήματα της KDD διαδικασίας είναι τα ακόλουθα:

1. Ανάπτυξη και κατανόηση του πεδίου της εφαρμογής συμπεριλαμβανόμενης οποιασδήποτε σχετικής προηγούμενης γνώσης για το πρόβλημα καθώς επίσης και των στόχων / προσδοκιών των τελικών χρηστών.
2. Δημιουργία του στοχευόμενου συνόλου δεδομένων (target data), το οποίο θα περιλαμβάνει τα δεδομένα από τα οποία πρόκειται να εξαχθεί η γνώση. Το βήμα αυτό είναι εξαιρετικά κρίσιμο καθώς η ποιότητα των δεδομένων επηρεάζει την απόδοση του συστήματος ανακάλυψης γνώσης.
3. Καθαρισμός και επεξεργασία των δεδομένων (data cleaning). Το βήμα αυτό περιλαμβάνει βασικές λειτουργίες όπως η αντιμετώπιση του προβλήματος των δεδομένων με ελλιπείς τιμές κ.α.
4. Μείωση της ποσότητας των δεδομένων (data reduction). Το βήμα αυτό περιλαμβάνει την εύρεση χρήσιμων χαρακτηριστικών για την αναπαράσταση των δεδομένων του προβλήματος ανάλογα με τους στόχους της ανακάλυψης γνώσης, τη μείωση του πλήθους αυτών των χαρακτηριστικών κ.α.
5. Επιλογή των εργασιών εξόρυξης γνώσης (data mining) που θα χρησιμοποιηθούν για τις ανάγκες του προβλήματος, π.χ. ταξινόμηση, πρόβλεψη, ομαδοποίηση κ.α.
6. Επιλογή των αλγορίθμων εξόρυξης γνώσης (data mining) που θα χρησιμοποιηθούν για την αναζήτηση προτύπων στα δεδομένα. Το βήμα αυτό περιλαμβάνει την επιλογή του κατάλληλου μοντέλου, την επιλογή των κατάλληλων παραμέτρων του μοντέλου κ.α.

7. Data Mining: αναζήτηση στα δεδομένα των προτύπων που μας ενδιαφέρουν.
8. Ερμηνεία των προτύπων που ανακαλύφθηκαν από την KDD διαδικασία – πιθανόν να χρειαστεί να επιστρέψουμε και πάλι σε κάποιο από τα παραπάνω βήματα.
9. Ενοποίηση της γνώσης που έχει εξαχθεί: ενσωμάτωση αυτής της γνώσης στο σύστημα ή απλά κοινοποίησή της με την κατάλληλη τεκμηρίωση στα ενδιαφερόμενα μέλη. Το βήμα αυτό περιλαμβάνει και έλεγχο συγκρούσεων με την γνώση που επικρατούσε πριν.



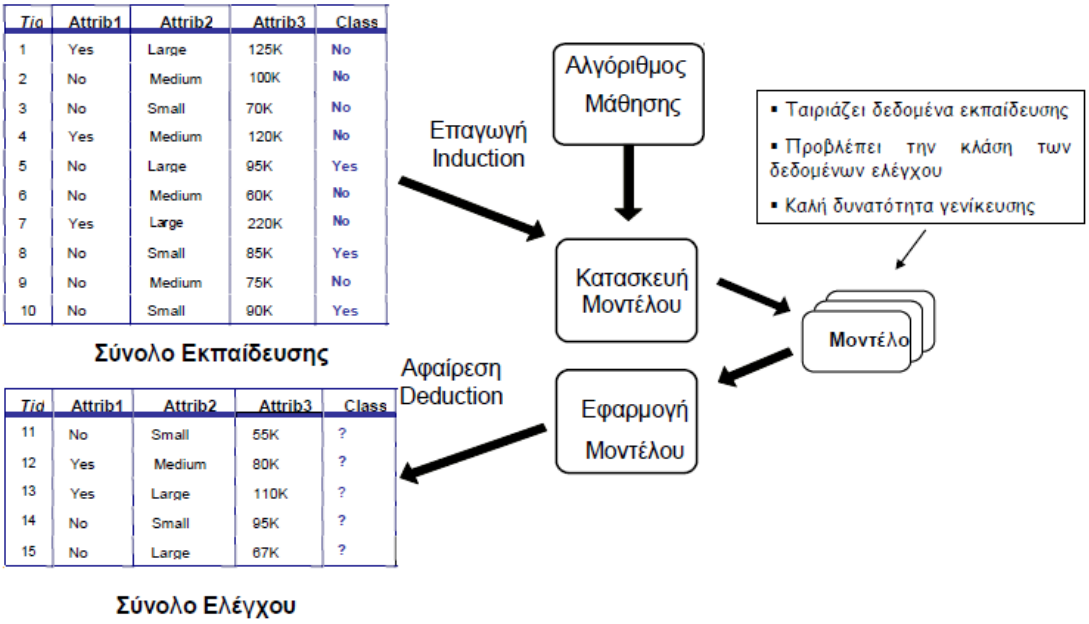
Από τα διάφορα βήματα της KDD διαδικασίας αυτό που συγκεντρώνει το μεγαλύτερο ενδιαφέρον είναι το βήμα του data mining. Αυτό όμως, δε σημαίνει πως τα υπόλοιπα βήματα δεν είναι σημαντικά, αντιθέτως η επιτυχής τους διεκπεραίωση επηρεάζει την επιτυχία ολόκληρης της KDD διαδικασίας.

1.6 Αλγόριθμοι μάθησης και κατασκευής μοντέλων

Οι τεχνικές κατηγοριοποίησης είναι καταλληλότερες για να προβλέπουν ή να περιγράφουν δεδομένα με δυαδικές ή κατηγορικές τιμές στα δεδομένα. Είναι λιγότερο αποτελεσματικές για συνεχείς τιμές στα δεδομένα, όπου για να εφαρμοστεί το μοντέλο θα πρέπει να γίνει διαχωρισμός των τιμών σε ένα συγκεκριμένο πλήθος κατηγοριών. Οι αλγόριθμοι κατηγοριοποίησης, οι γνωστοί ταξινομητές είναι μια συστηματική προσέγγιση για τη δημιουργία των μοντέλων από τα δεδομένα εισόδου. Τέτοια παραδείγματα αποτελούν ποικίλοι αλγόριθμοι μάθησης από τα Decision Trees, Nearest Neighbors, statistical analysis, neural networks, rule induction, support vector machines, Bayesian networks και πολλά άλλα κ.α.

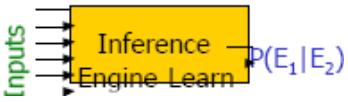
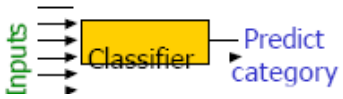
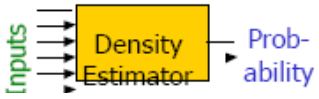
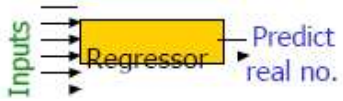
Το μοντέλο που θα δημιουργηθεί θα πρέπει επίσης να ταιριάζει τα δεδομένα εισόδου και να προβλέπει σωστά την έξοδο y και για εγγραφές οι οποίες δεν χρησιμοποιήθηκαν στην κατασκευή του μοντέλου. Επομένως ο στόχος ενός αλγορίθμου εκμάθησης και κατασκευής μοντέλων είναι το να δημιουργεί μοντέλα με ικανότητα γενίκευσης, δηλ. μοντέλα που προβλέπουν σωστά την έξοδο y για άγνωστα παραδείγματα x.

Η επίλυση των προβλημάτων κατηγοριοποίησης περιλαμβάνει δύο βασικά στάδια. Στο πρώτο στάδιο, το στάδιο της εκπαίδευσης, δημιουργούμε ένα μοντέλο από την αξιολόγηση και την ανάλυση των δεδομένων εκπαίδευσης. Αυτό το βήμα έχει σαν είσοδο τα δεδομένα εκπαίδευσης και σαν έξοδο ένα ορισμό του μοντέλου που αναπτύχθηκε. Το μοντέλο που δημιουργείται από αυτό το στάδιο είναι σε θέση να κατηγοριοποιεί τα δεδομένα εκπαίδευσης με όσο το δυνατό μεγαλύτερη ακρίβεια. Όταν είναι ήδη γνωστές οι κατηγορίες του συνόλου των δεδομένων εκπαίδευσης, δηλαδή το σύνολο των δεδομένων εκπαίδευσης περιλαμβάνει ένα χαρακτηριστικό το οποίο δείχνει την κλάση στην οποία κατηγοριοποιείται η κάθε εγγραφή, τότε το βήμα αυτό καλείται εποπτευμένη μάθηση (supervised learning), σε αντίθετη περίπτωση, δηλαδή όταν δεν είναι γνωστές οι κατηγορίες του συνόλου των δεδομένων εκπαίδευσης, τότε το βήμα αυτό καλείται μη εποπτευμένη μάθηση (unsupervised learning - clustering). Στο δεύτερο στάδιο, το στάδιο της εφαρμογής, εφαρμόζουμε το μοντέλο που αναπτύχθηκε στο προηγούμενο βήμα κατηγοριοποιώντας τις εγγραφές της υπό εξέταση Βάσης Δεδομένων .



Γενικό σχήμα ταξινόμησης - κατηγοριοποίησης.

Υπάρχουν πολλοί αλγόριθμοι μάθησης και κατασκευής μοντέλων για πολλά είδη εργασιών όπως:

 An Inference Engine Learns P(E1|E2)	Joint Density Estimation, Bayes Net Structure Learning
 A Classifier Predicts category	Decision Tree, Sigmoid Perceptron, Sigmoid Neural Network, Probabilistic Neural Network, Gauss/Joint BC, Gauss Naïve BC, N.Neigh, Bayes, Net Based BC, Cascade Correlation, Support Vector Machines k-nearest neighbour –kNN, weighted kNN, variable kNN, ID3, CHAID, CART, C4.5, QUEST, C5.0, SPRINT, SLIQ, dynamic classifier selection, adaptive classifier combination,
 A Density Estimator Computes Probability	Joint DE, Naïve DE, Gauss/Joint DE, Gauss Naïve DE, Bayes Net Structure Learning
 **A Regressor** Predicts a real value Y	Linear Regression, Polynomial Regression, RBFs, Multi-Layer Perceptron (MLP), Radial Basis Function Neural Network, **General Regression Neural Network (GRNN),** Locally Weighted Regression, Robust Regression, Cascade Correlation, Regression Trees, GMDH, Multi-linear Interpolation, Multivariate Adaptive Regression Splines (MARS)

1.7 Μέθοδοι Αξιολόγησης Αποτελεσμάτων

Στην εξόρυξη δεδομένων η πιο γνωστή μέθοδος είναι η ταξινόμηση προτύπων (classification). Η κατηγοριοποίηση χρησιμοποιείτε σε συστήματα που μας χρησιμεύουν στην καθημερινή μας ζωή. Σε ιατρικά, τραπεζικά και βιομηχανικά συστήματα , σε συστήματα που αφορούν την οικονομία μια χώρας ή κάποιου οργανισμού και σε πολλά άλλα συστήματα. Υπάρχουν πάρα πολλοί αλγόριθμοι κατηγοριοποίησης. Η αποτελεσματικότητα ενός αλγορίθμου μπορεί να εκτιμηθεί βάση της ακρίβεια (accuracy) της κατηγοριοποίησης. Λέγοντας αποτελεσματικότητα εννοούμε την ικανότητα του μοντέλου να προβλέπει την κατηγορία μιας νέας περίπτωσης και βάση αυτού να εξετάζεται η επίδοση των αλγορίθμων. Η εκτίμηση της ακρίβειας κατηγοριοποίησης είναι πολύ σημαντικό ζήτημα αφού κάτι τέτοιο δείχνει πόσο καλά ανταποκρίνεται ο αλγόριθμος μας για δεδομένα με τα οποία δεν έχει εκπαιδευτεί. Η εκτίμηση της ακρίβειας επιτρέπει και την σύγκριση των αλγόριθμων κατηγοριοποίησης.

Η ποιότητα των μοντέλων εξετάζεται με την εκτίμηση του σφάλματος γενίκευσης, δηλαδή την ικανότητα του μοντέλου να προβλέπει την κατηγορία μιας νέας περίπτωσης. Με βάση την απόδοση του κάθε ταξινομητή στα διαθέσιμα δεδομένα προκύπτει το εάν ένα μοντέλο καλύπτει καλά το σύνολο των δεδομένων λειτουργίας του. Κατά την μάθηση του νευρωνικού δικτύου μέσα από ένα σύνολο δεδομένων εκπαίδευσης, το νευρωνικό δίκτυο μαθαίνει τις εσωτερικές παραμέτρους του για μια συνάρτηση αντιστοίχησης της κάθε εισόδου στην εκτιμώμενη έξοδο. Σε αυτή τη μάθηση υπάρχουν δύο σφάλματα, το σφάλμα εκπαίδευσης (trainError) και το σφάλμα ελέγχου (testError).

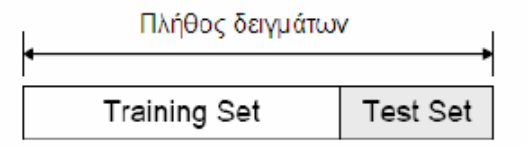
Το σφάλμα εκπαίδευσης που δείχνει πόσο καλά προσεγγίζει το μοντέλο τα Ntrain το πλήθος παραδειγμάτων εκπαίδευσης με τα οποία εκπαιδεύτηκε και είναι αυτό που μειώνεται κατά τη διάρκεια εκπαίδευσης, και το σφάλμα ελέγχου που δείχνει πόσο καλά γενικεύει το μοντέλο σε Ntest το πλήθος νέα παραδείγματα ελέγχου.

Οι πιο κάτω τεχνικές που θα αναφέρουμε επιτρέπουν την καλύτερη χρήση των δεδομένων για εκπαίδευση (training),εκτίμηση απόδοσης (testing),επιλογή μοντέλου (model selection)

1.7.1 Η μέθοδος Hold-Out

Στην μέθοδο αυτή το σύνολο δεδομένων διαχωρίζετε σε δύο ανεξάρτητα σύνολα δεδομένων. Το σύνολο δεδομένων εκπαίδευσης (training set) και το σύνολο δεδομένων ελέγχου (test set). Το σύνολο δεδομένων εκπαίδευσης χρησιμοποιείται για την εκπαίδευση του ταξινομητή. Το σύνολο δεδομένων ελέγχου χρησιμοποιείτε για δοκιμές ελέγχου δηλαδή για την εκτίμηση του

σφάλματος. Συνήθως κρατάμε για δεδομένα εκπαίδευσης Τα 2/3 του συνόλου δεδομένων και το υπόλοιπο 1/3 για δεδομένα ελέγχου.



Η μέθοδος hold-out είναι μια απλή μέθοδος η οποία είναι εύκολη στην υλοποίηση, παρόλα αυτά, έχει δύο βασικά μειονεκτήματα. Στην περίπτωση που το σύνολο δεδομένων είναι μικρό δεν μπορούμε να κρατήσουμε δείγματα για δοκιμή. Επίσης ένα σημαντικό μειονέκτημα είναι ότι μπορεί το σύνολο δεδομένων εκπαίδευσης να περιέχει λίγα δεδομένα που ανήκουν σε μια κλάση, αυτό θα έχει ως αποτέλεσμα την δημιουργία ενός μοντέλου το οποίο δεν θα μπορεί να κατατάσσει εσφαλμένα τα στιγμιότυπα του συνόλου ελέγχου που πραγματικά ανήκουν στην κλάση αυτήν. Σε αυτήν την περίπτωση η ακρίβεια του μοντέλου θα είναι υποεκτιμημένη, λόγω του ότι δεν κατατάσσει ορθά τα δεδομένα της συγκεκριμένης κλάσης. Για να αποφύγουμε το πιο πάνω πρόβλημα πρέπει κατά το διαχωρισμό του συνόλου δεδομένων σε δεδομένα εκπαίδευσης και δεδομένα ελέγχου να υπάρχει μια διαδικασία για να διαχωριστούν τα δεδομένα με την σωστή αναλογία στα δύο υποσύνολα. Η διαδικασία αυτή ονομάζεται stratification και χρησιμοποιείται για το σωστό διαχωρισμό των δεδομένων.

1.7.2 Οι μέθοδοι Cross Validation (διασταυρωμένης επικύρωσης)

Στις μεθόδους cross-validation χρησιμοποιείτε ολόκληρο το σύνολο δεδομένων για εκπαίδευση και για έλεγχο. Αυτό γίνετε για να αποφευχθούν τα δύο προβλήματα που συναντούμε στην μέθοδο hold-out. Αυτό θα έχει ως αποτέλεσμα να διεξάγονται πολλαπλάσια πειράματα εις βάρος του υψηλότερου υπολογιστικού κόστους .

Στις μεθόδους cross validation περιλαμβάνονται η μέθοδος Random Subsampling, η μέθοδος K-Fold Cross-Validation, η μέθοδος Leave-one-out Cross-Validation και Bootstrap.

1.7.3 Τυχαία δειγματοληψία (random subsampling)

Η μέθοδος ακολουθεί παρόμοια λογική με την hold-out. Οι δυο μέθοδοι διαφέρουν στον τρόπο που διαμερίζουν το σύνολο δεδομένων στα δύο τμήματα (σύνολο εκπαίδευση και σύνολο ελέγχου), ώστε να λυθεί το πρόβλημα που εμφανίζει η holdout.

Σε αυτήν την περίπτωση το σύνολο ελέγχου επιλέγεται ως τυχαίο δείγμα.

Εφαρμόζουμε τυχαία δειγματοληψία χωρίς επανατοποθέτηση και επιλέγουμε Ν πρότυπα για το σύνολο ελέγχου. Τα εναπομένοντα πρότυπα σχηματίζουν το σύνολο εκπαίδευσης. Με αυτήν την ενέργεια μειώνεται η επιρροή που μπορεί να επιφέρει η κατανομή των στιγμιότυπων στο σύνολο δεδομένων. Η προηγούμενη διαδικασία επαναλαμβάνεται κ φορές ώστε νε επιτευχθεί η μεγαλύτερη δυνατή μείωση της επιρροής.

1.7.4 k-fold cross-validation

Θεωρείται μια από τις πλέον αξιόπιστές μεθόδους για την αποτίμηση της ακρίβειας κατηγοριοποιητών. Στην περίπτωση αυτήν εφαρμόζεται ένας πιο συστηματικός τρόπος για να λαμβάνουμε τυχαία δείγματα για το σχηματισμό του συνόλου ελέγχου.

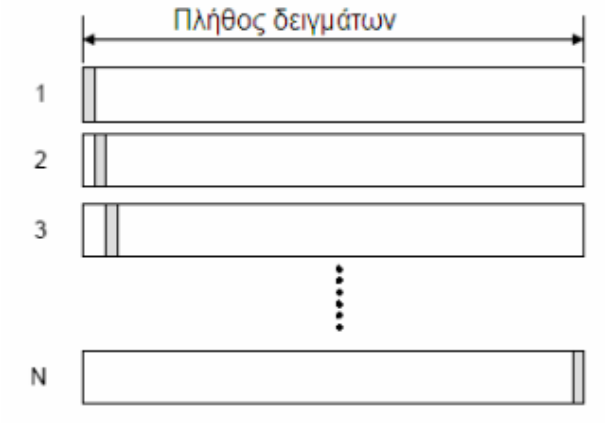
Εάν το σύνολο δεδομένων αποτελείται από Μ πρότυπα και έχουμε ορίσει ως αριθμό επαναλήψεων το k, τότε χωρίζουμε το σύνολο σε k ισομεγέθη τμήματα μεγέθους M/k το καθένα. Στην i-οστή επανάληψη, το i-οστό τμήμα λειτουργεί ως σύνολο ελέγχου, ενώ τα υπόλοιπα k – 1 τμήματα αποτελούν το σύνολο εκπαίδευσης.

Για μεγάλο Κ η εκτίμηση του σφάλματος είναι αρκετά ακριβής αλλά με μεγάλες αποκλίσεις. Επίσης αυξάνετε το υπολογιστικό κόστος. Για μικρό Κ μειώνεται το πλήθος των πειραμάτων και το υπολογιστικό κόστος και το εκτιμώμενο σφάλμα θα είναι μεγαλύτερο από το πραγματικό αλλά με μικρότερες αποκλίσεις.

Η συνηθέστερη τιμή για το k είναι το 10 (10-fold cross-validation).

1.7.5 Leave-one-out Cross-Validation (εξαίρεσε ένα)

Η Leave-one-out είναι ειδική περίπτωση της K-Fold Cross Validation. Για ένα σύνολο δεδομένων με Κ παραδείγματα εκτελούμε Κ πειράματα. Κάθε παράδειγμα αφήνετε με την σειρά του έξω από το σύνολο εκπαίδευσης, και ο αλγόριθμος στην συνέχεια εκπαιδεύεται στα υπόλοιπα Κ-1. Δηλαδή χρησιμοποιούνται Κ-1 δείγματα για εκπαίδευση και ένα μόνο για έλεγχο. Η παραλλαγή αυτή χρησιμοποιείται μόνο σε μικρά σύνολα δεδομένων.



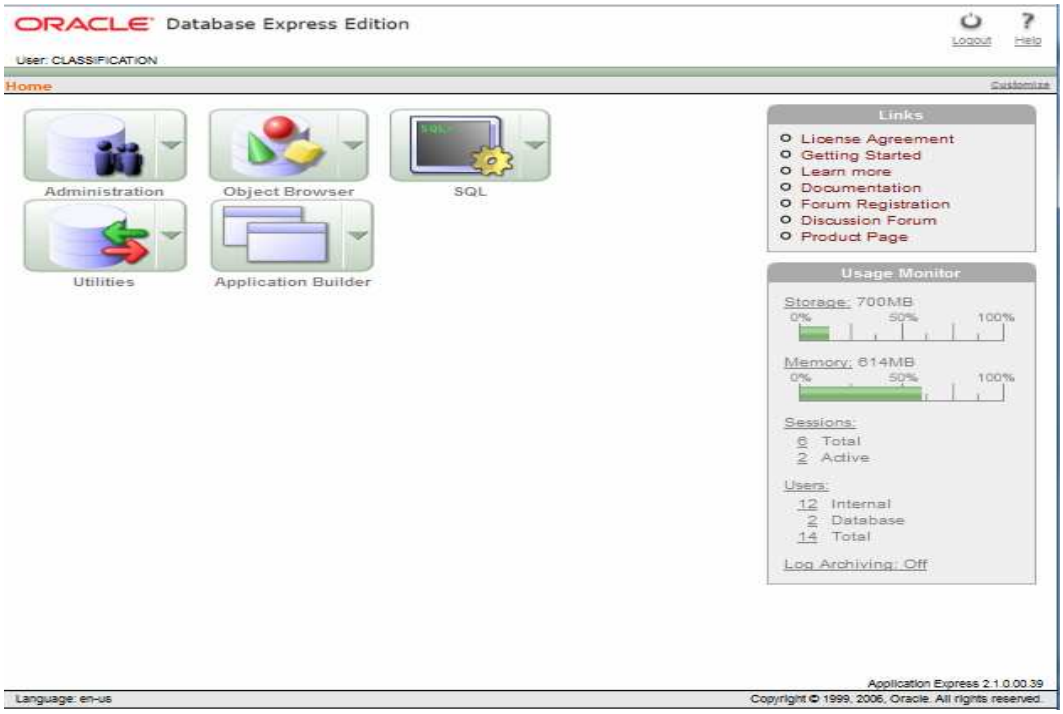
Η μέθοδος αυτή έχει πλεονέκτημα στο ότι αποφεύγετε η τυχαία δειγματοληψία. Επίσης χρησιμοποιείτε το μέγιστο δυνατό ποσό δεδομένων για εκπαίδευση. Μειονέκτημα αυτής της μεθόδου είναι έχει ψηλό υπολογιστικό κόστος .

1.7.6 Η μέθοδος bootstrap

Αποτελεί εναλλακτική μέθοδο της τυχαίας δειγματοληψίας. Ενώ η τυχαία δειγματοληψία σχηματίζει το σύνολο ελέγχου εφαρμόζοντας τυχαία δειγματοληψία χωρίς επανατοποθέτηση στο σύνολο δεδομένων, δηλαδή ένα στιγμιότυπο, εφόσον επιλεγόταν δεν ήταν δυνατό να ξαναεπιλεγεί, ο bootstrap εφαρμόζει τυχαία δειγματοληψία με επανατοποθέτηση.

1.8 ΟΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ORACLE

Το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων Oracle σχεδιάστηκε για να επιτρέψει την ταυτόχρονη πρόσβαση σε μεγάλες κατανεμημένες βάσεις δεδομένων. Ήταν το πρώτο εμπορικό προϊόν που εμφανίστηκε στην αγορά. Με την πάροδο του χρόνου η Oracle έχει αναπτυχθεί πέρα από το σχεσιακό διακομιστή βάσεων δεδομένων. Εκτός από εργαλεία που σχετίζονται για την ανάπτυξη και διαχείριση πουλά και έξυπνα εργαλεία όπως πολυδιάστατο σύστημα διαχείρισης βάσεων δεδομένων, εργαλεία ερωτημάτων και ανάλυσης, προϊόντα εξόρυξης δεδομένων και μια εφαρμογή διακομιστή. Πιο κάτω θα αναφέρουμε διάφορες λειτουργίες των βάσεων δεδομένων της Oracle.



Εργαλεία Σχεδίασης και Ερωτημάτων Βάσεων Δεδομένων

Η Oracle προσφέρει διάφορα εργαλεία για σχεδίαση ερωτήματα, δημιουργία αναφορών και ανάλυσης δεδομένων.

Εργαλεία Σχεδίασης

Τα περισσότερα εργαλεία σχεδίασης της Oracle περιλαμβάνονται στο Oracle Internets Development Suite. Αυτό είναι ένα πακέτο από εργαλεία για διάφορα θέματα ανάπτυξης εφαρμογών, όπως εργαλεία για ανάπτυξη φορμών, μοντελοποίηση δεδομένων, αναφορές και ερωτήματα.

Το βασικό εργαλείο σχεδίασης βάσεων δεδομένων του πακέτου είναι το Oracle Designer. Υποστηρίζει τεχνικές μοντελοποίησης όπως τα διαγράμματα E-R(οντότητας-σχέσης), μηχανική συστημάτων και ανάλυση και σχεδίαση αντικειμένων. Το πακέτο επίσης παρέχει εργαλεία για ανάπτυξης εφαρμογών για δημιουργία φορμών, αναφορών και εργαλείων, για διάφορα θέματα ανάπτυξης Java και XML.

Εργαλεία Ερωτημάτων

Η Oracle παρέχει εργαλεία για δημιουργία ερωτημάτων, αναφορών και ανάλυσης δεδομένων συμπεριλαμβανομένου και του διακομιστή OLAP.

Το Oracle Discoverer βασίζεται στο Web και είναι ένα εργαλείο ερωτημάτων, αναφορών, ανάλυσης και δημοσίευσης στο Web για τελικούς χρήστες και ανάλυση δεδομένων. Επιτρέπει στους χρήστες να βρίσκουν σύνολα αποτελεσμάτων, συγκεντρωτικά δεδομένα και να αποθηκεύουν υπολογισμούς ως αναφορές που μπορούν να δημοσιευτούν σε διάφορες μορφές, όπως λογιστικά φύλλα ή HTML.

Το Oracle Express Server είναι ένας πολυδιάστατος διακομιστής βάσεων δεδομένων. Υποστηρίζει μια μεγάλη ποικιλία από αναλυτικά ερωτήματα, όπως προβλέψεις, μοντελοποίηση και διαχείριση σεναρίων.

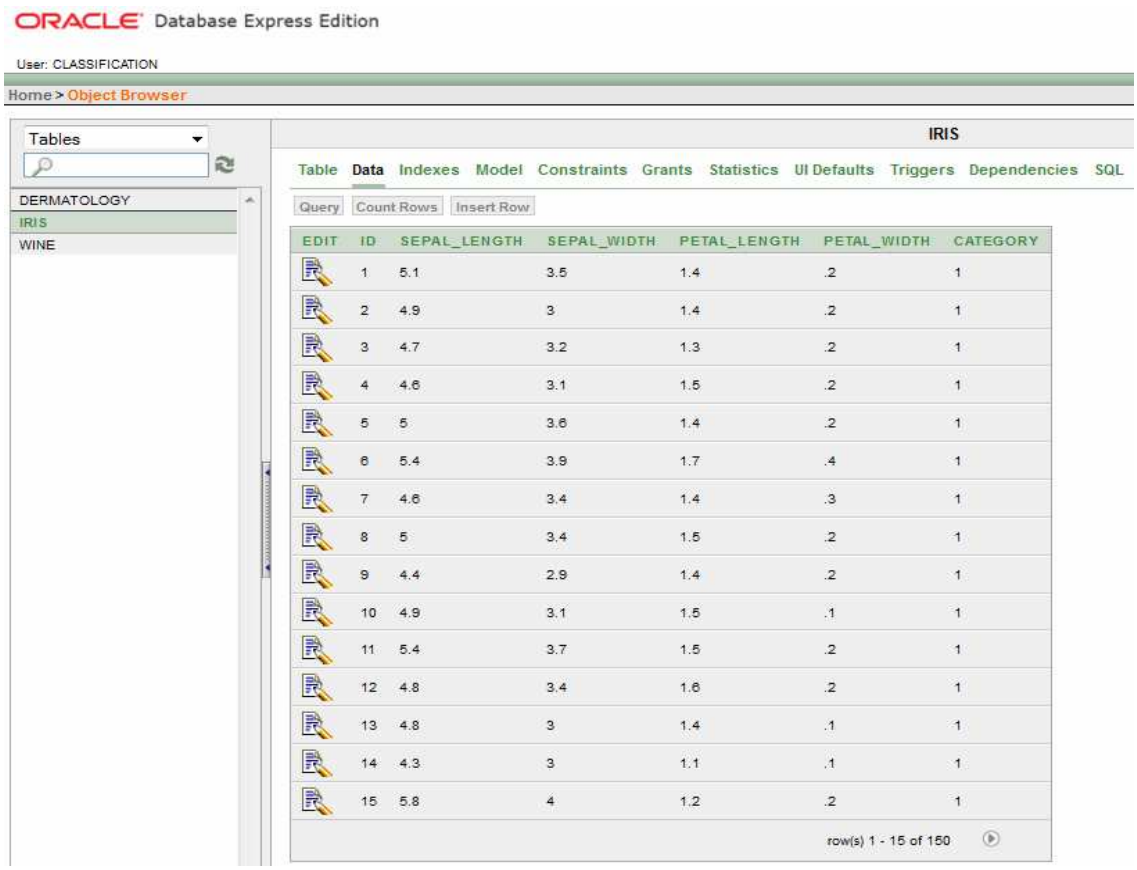
Με την εισαγωγή των OLAP υπηρεσιών, η Oracle σταμάτησε να υποστηρίζει ξεχωριστές μηχανές αποθήκευσης και μετακίνησε του περισσότερους υπολογισμούς στην SQL. Αυτό έχει ως αποτέλεσμα ένα μοντέλο όπυ όλα τα δεδομένα βρίσκονται στο σχεσιακό σύστημα διαχείρισης και όπου οι υπόλοιποι υπολογισμοί που δεν μπορούν να υπολογιστούν στην SQL γίνονται σε μια μηχανή υπολογισμών που τρέχει στο διακομιστή της βάσης δεδομένων.

Εργαλεία Διαχείρισης

Η Oracle παρέχει στους χρήστες διάφορα εργαλεία για την διαχείριση του συστήματος και ανάπτυξη εφαρμογών.

Oracle Enterprise Manager

Είναι το κύριο εργαλείο της Oracle για την διαχείριση βάσεων δεδομένων. Παρέχει ένα εύκολο γραφικό περιβάλλον και διάφορους οδηγούς για διαχείριση σχήματος, διαχείριση ασφάλειας, διαχείριση στιγμιότυπων, διαχείριση αποθήκευσης και χρονοδιαγράμματα εργασιών. Παρέχει παρακολούθηση απόδοσης και εργαλεία για να βοηθήσει ένα διαχειριστή να βελτιώσει την SQL εφαρμογή, τις διαδρομές πρόσβασης και τις παραμέτρους στιγμιότυπων και αποθήκευσης δεδομένων.



Database Resource Management

Ο διαχειριστής βάσεων δεδομένων θα πρέπει να μπορεί να ελέγχει πως κατανέμεται η δύναμη επεξεργασίας του υλικού μεταξύ χρηστών ή ομάδα χρηστών. Μερικές ομάδες μπορεί να εκτελούνε αλληλεπιδραστικά ερωτήματα, όπου ο χρόνος απόκρισης είναι κρίσιμος. Άλλοι μπορεί να εκτελούν χρονοβόρες αναφορές, που μπορεί να τρέχουν ως μαζικές εργασίες στο παρασκήνιο όταν ο φόρτος είναι χαμηλός. Η λειτουργία Database Resource Management επιτρέπει στον διαχειριστή της βάσης δεδομένων να διαιρεί του χρήστες σε ομάδες κατανάλωσης των πόρων, σε διαφορετικές προτεραιότητες και ιδιότητες.

2 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

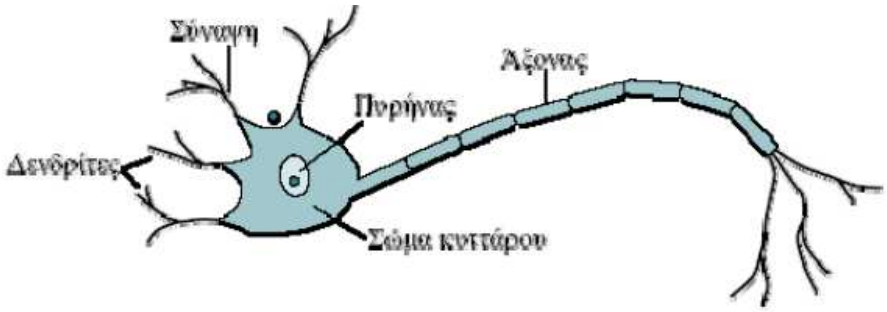
2.1 ΕΙΣΑΓΩΓΗ ΣΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Η ανάγκη του ανθρώπου να μοντελοποιήσει το σημαντικότερο του όργανο που δεν είναι άλλο φυσικά από τον ανθρώπινο εγκέφαλο, και το πώς αυτός λειτουργεί οδήγησε στη ανάγκη για τη δημιουργία των τεχνητών νευρωνικών δικτύων. Με τον όρο τεχνητό νευρωνικό δίκτυο αναφερόμαστε σε μια διασυνδεμένη ομάδα τεχνητών νευρώνων που χρησιμοποιώντας ένα μαθηματικό ή υπολογιστικό μοντέλο επεξεργάζεται πληροφορίες βασισμένη σε μια πολυσύνδετη προσέγγιση ενός προβλήματος και παράγει κάποια συγκεκριμένη έξοδο. Όπως και τα νευρωνικά δίκτυα του ανθρώπινου εγκεφάλου, τα τεχνητά νευρωνικά δίκτυα έχουν την ικανότητα να μαθαίνουν μέσω παραδειγμάτων, να εκπαιδεύονται και κατά συνέπεια να αντιδρούν ορθότερα σε οποιαδήποτε μελλοντική είσοδο δεχτούν.

Τα νευρωνικά δίκτυα αποτελούν μια νέα φυσική επιστήμη καθώς έχουν γίνει γνωστά και αναπτυχθεί τις τελευταίες δεκαετίες. Αυτή η φυσική επιστήμη αναπτύχτηκε ραγδαία καθώς πολλοί επιστήμονες ασχολούνται με αυτή επιφέροντας σημαντικά επιτεύγματα συμβάλλοντας έτσι στο να γίνουν γνωστά τα νευρωνικά δίκτυα. Οι επιστήμονες στην περιοχή των νευρωνικών δικτύων προέρχονται από διάφορους κλάδους των θετικών επιστημών όπως την Ιατρική, την επιστήμη Μηχανικών, τη Φυσική, τη Χημεία, τα Μαθηματικά, την επιστήμη Υπολογιστών, την Ηλεκτρολογία κτλ. Αυτό δείχνει ότι για την ανάπτυξή τους απαιτούνται ταυτόχρονα γνώσεις και θέματα από πολλές επιστήμες, ενώ το ίδιο ισχύει και για τις τεχνικές και τις μεθόδους που χρησιμοποιούνται. Το κύριο χαρακτηριστικό τους είναι ότι οι αρχές τους και οι λειτουργίες τους βασίζονται και εμπνέονται από το νευρικό σύστημα των ζώντων οργανισμών. Τα νευρωνικά δίκτυα εξελίχθηκαν και επεκτάθηκαν και πέρα από βιολογικούς οργανισμούς , δημιουργώντας μια νέα περιοχή αποκομμένη από την βιολογία. Χρησιμοποιούνται για να λύσουν οποιαδήποτε προβλήματα στους ηλεκτρονικούς υπολογιστές. Η λειτουργία τους προσπαθεί να συνδυάσει τον τρόπο σκέψης του ανθρώπινου εγκεφάλου με αυτή της μαθηματικής σκέψης. Χρησιμοποιούνται για περίπλοκες μαθηματικές συναρτήσεις και για κάθε είδους μαθηματικά εργαλεία.

Τα πρώτα που μελετήθηκαν είναι τα βιολογικά νευρωνικά δίκτυα. Αυτά βρίσκονται σε όλους τους ζώντες οργανισμούς. Σε αυτά ένα νευρικό σύστημα το οποίο είναι υπεύθυνο για πολλές διεργασίες . Το νευρικό σύστημα αποτελείται από πολλά νευρωνικά δίκτυα το οποίο είναι υπεύθυνο για μια διαφορετική λειτουργία όπως η επαφή με τον έξω κόσμο, η μάθηση, η μνήμη κτλ. Η κεντρική μονάδα του νευρικού συστήματος είναι ο εγκέφαλος , ο οποίος

είναι νευρωνικό δίκτυο. Κάθε νευρωνικό δίκτυο αποτελείται από πολλές μονάδες που ονομάζονται νευρώνες. Οι νευρώνες είναι μικρές ανεξάρτητες μονάδες του δικτύου, οι οποίες συνεχώς και ασταμάτητα επεξεργάζονται πληροφορίες, παίρνοντας και στέλνοντας πληροφορίες σε άλλους νευρώνες.



Τα βιολογικά νευρωνικά δίκτυα εκτελούν πολύ περίπλοκες αλλά συνάμα πολύ χρήσιμες διεργασίες για την καθημερινή ζωή του ανθρώπου. Τις διεργασίες αυτές ο ανθρώπινος εγκέφαλος τις εκτελεί με ελάχιστη ή και μηδαμινή προσπάθεια π.χ. η αναγνώριση μιας εικόνας. Ένας υπολογιστής δεν μπορεί με την ίδια ευκολία να εκτελέσει αυτές τις διεργασίες για το λόγο ότι η δομή του ανθρώπινου εγκεφάλου είναι πολύ διαφορετική από αυτή του υπολογιστή. Για να φτιαχτεί ένας τέτοιος υπολογιστής που να μπορεί να έχει παρόμοια δομή με αυτή του ανθρώπινου εγκεφάλου έγιναν κάποιες σκέψεις για να δημιουργηθούν κάποια μοντέλα του ανθρώπινου νευρωνικού συστήματος τα οποία θα περιέχουν όλα τα χαρακτηριστικά για να εκτελούν όλες τις διεργασίες που εκτελεί ο ανθρώπινος εγκέφαλος. Τα δίκτυα αυτά ονομάζονται τεχνητά νευρωνικά δίκτυα (ΤΝΔ). Τα ΤΝΔ παίρνουν γνώσεις με την εξάσκηση και την εμπειρία, όπως ακριβώς οι άνθρωποι. Η διαφορά τους με τα βιολογικά νευρωνικά δίκτυα είναι στο ότι δεν ακολουθούν ορισμένους προκαθορισμένους κανόνες που είναι το χαρακτηριστικό των υπολογιστών.

Σκοπός των ΤΝΔ είναι να δειχθεί αν τα μοντέλα που βασίζονται σε αυτές τις απλές αρχές μπορούν να εκτελέσουν τους υπολογισμούς που ξέρουμε ότι εκτελεί ο ανθρώπινος εγκέφαλος. Δηλαδή, αν μπορούν να μεταδώσουν πληροφορίες σχετικά με τα ερεθίσματα που λαμβάνουν και να μετασχηματίσουν ερεθίσματα σε μια απόκριση ανάλογη με αυτή που εμείς παράγουμε όταν διαβάζουμε μια λέξη δυνατά, ή αναγνωρίζουμε ένα πρόσωπο από μία νέα γωνία, ή εξάγουμε ένα συμπέρασμα.

Η θεωρία των ΤΝΔ έχει να αντιμετωπίσει μια πρόκληση. Η πρόκληση αυτή έχει να κάνει με την εύρεση των κατάλληλων αλγορίθμων εκπαίδευσης των δικτύων και ανάκλησης της πληροφορίας που αυτά περιέχουν έτσι ώστε να προσαρμόζουν ευφυείς διαδικασίες όπως η μάθηση, η γενίκευση, η ομαδοποίηση προτύπων κτλ. Για να επιτευχθεί αυτός ο στόχος απαιτείται ο ορισμός του κατάλληλου περιβάλλοντος εκπαίδευσης για παράδειγμα αν το

δίκτυο θα εκπαιδεύεται με επίβλεψη, δηλαδή με την χρήση κάποιον δεδομένων οδηγών, ή αν το δίκτυο θα αφήνεται να οργανωθεί μόνο του αυτόματα και με ποιο συγκεκριμένο κριτήριο η στόχο.

Η τεχνητή νοημοσύνη έχει σαν αρχή την ύπαρξη ενός υλικού στρώματος στο οποίο θα μπορούν να εκτελούνται διάφορες λειτουργίες όπως η αναγνώριση εικόνων, η αναγνώριση φωνής, η κατανόηση και η παραγωγή της γλώσσας, η λήψη αποφάσεων με δυνατότητα την κατάστρωση στρατηγικής για την επιλογή της καλύτερης με βάση διάφορα κριτήρια κόστους, η μάθηση και η αυτό προσαρμογή σε νέο περιβάλλον και νέες καταστάσεις κ.α. Η μελέτη της τεχνητής νοημοσύνης έχει σαν στόχο την ανάπτυξη αλγορίθμων που θα μιμούνται τις λειτουργίες που αναφέρονται πιο πάνω, δηλαδή θα κάνουν αναγνώριση φωνής, προσώπων , περιβάλλοντος, θα αναπτύσσουν βέλτιστες στρατηγικές για ένα πρόβλημα, θα εκτελούν συλλογισμούς και θα καταλήγουν σε λογικά συμπεράσματα και θα μαθαίνουν από την εμπειρία τους.

2.2 ΔΟΜΗ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

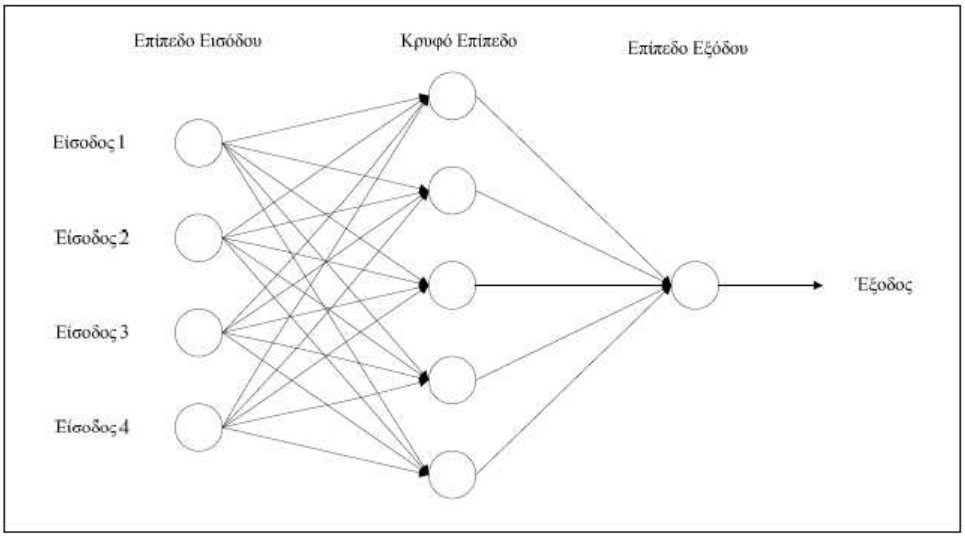
Τα τεχνητά νευρωνικά δίκτυα είναι μοντέλα που μιμούνται την λειτουργία και την δομή των βιολογικών νευρωνικών δικτύων. Το αντικείμενο των τεχνιτών νευρωνικών δικτύων είναι η ανάπτυξη και η μελέτη μαθηματικών αλγορίθμων που μιμούνται την αρχιτεκτονική και το πρότυπο των βιολογικών νευρωνικών δικτύων.

Με τον όρο ΤΝΔ εννοούμε ένα δίκτυο από τεχνητούς νευρώνες οργανωμένους σε επίπεδα (layers), δηλαδή ο κάθε τεχνητός νευρώνας θεωρείται ως ένα ανεξάρτητο υπολογιστικό στοιχείο. Τα νευρωνικά δίκτυα χρησιμοποιούν ένα σύνολο από στοιχεία επεξεργασίας (κόμβους) που είναι ανάλογοι με τους νευρώνες του ανθρώπινου μυαλού. Οι κόμβοι αυτοί διασυνδέονται μεταξύ τους σε ένα δίκτυο που μπορεί να αναγνωρίσει τα πρότυπα μόλις αυτά παρουσιαστούν μέσα σε ένα σύνολο δεδομένων. Δηλαδή το δίκτυο μπορεί να μαθαίνει από την εμπειρία όπως ακριβώς κάνουν και οι άνθρωποι.

Ένα ΤΝΔ μπορεί να απαρτίζεται από ένα επίπεδο εισόδου, ένα ή και περισσότερα κρυφά επίπεδα και ένα επίπεδο εξόδου. Η μονάδα επεξεργασίας μπορεί να θεωρηθεί ως ένας πραγματικός νευρώνας, ή ομάδα νευρώνων. Κάθε μονάδα αθροίζει πληροφορία από άλλες μονάδες, εκτελεί ένα απλό υπολογισμό σε αυτό το άθροισμα και περνά το αποτέλεσμα σε άλλες μονάδες. Η επίδραση μιας μονάδας σε μια άλλη εξαρτάται από τη βαρύτητα της μεταξύ τους διασύνδεσης.

Οι τεχνητοί νευρώνες ενός επιπέδου συνδέονται πλήρως ή και μερικώς με τους τεχνητούς νευρώνες του επόμενου επιπέδου. Αν η διάδοση του σήματος

πραγματοποιείται έτσι ώστε κάθε νευρώνας να έχει ως είσοδο την έξοδο ενός νευρώνα του προηγούμενου ή των προηγούμενων επιπέδων, τότε καλείται ΤΝΔ πρόσθιας τροφοδότησης (feedforward), τα οποία είναι η πιο διαδεδομένη κατηγορία νευρωνικών δικτύων, όπου επιτρέπουν την κίνηση των δεδομένων μόνο προς μια κατεύθυνση δηλαδή από μια είσοδο προς μια έξοδο. Υπάρχει και η δυνατότητα σύνδεσης ενός τεχνητού νευρώνα με προηγούμενα επίπεδα οπότε υπάρχει η δυνατότητα ανάδρασης (feedback) του σήματος.



Σε ένα νευρωνικό δίκτυο πρόσθιας τροφοδότησης, τα κύρια βήματα για την κατασκευή ενός μοντέλου ταξινόμησης, είναι τα εξής :

- Η αναγνώριση των χαρακτηριστικών εισόδου και εξόδου.
- Η κατασκευή ενός δικτύου με την κατάλληλη τοπολογία.
- Η επιλογή του σωστού συνόλου εκπαίδευσης, το οποίο περιλαμβάνει
- δεδομένα που είναι ορισμένα ανά ζεύγη.
- Η εκπαίδευση του δικτύου. Στην διάρκεια της φάσης αυτής, τα δεδομένα εισέρχονται στο νευρωνικό δίκτυο ένα ένα. Το νευρωνικό δίκτυο μαθαίνει συγκρίνοντας τα αποτελέσματα ταξινόμησης ενός αντικειμένου με την γνωστή πραγματική ταξινόμηση αυτού. Τα λάθη από την αρχική ταξινόμηση του πρώτου αντικειμένου χρησιμοποιούνται για να διορθωθεί το δίκτυο μέσω της τροποποίησης των συναρτήσεων των νευρώνων. Η παραπάνω διαδικασία είναι επαναληπτική.
- Ο έλεγχος του δικτύου χρησιμοποιώντας ένα σύνολο ελέγχου, το οποίο είναι ανεξάρτητο από το σύνολο εκπαίδευσης.

Στην συνέχεια, το μοντέλο που παράγεται από το δίκτυο εφαρμόζεται για να ταξινομήσει νέα δεδομένα. Αξίζει να σημειώσουμε ότι η εκπαίδευση ενός νευρωνικού δικτύου βασίζεται στον υπολογισμό των τιμών των βαρών που προαναφέρθηκαν. Ο πιο γνωστός αλγόριθμος, μεταξύ άλλων στον οποίο βασίζεται ο παραπάνω υπολογισμός, είναι ο αλγόριθμος ανάστροφης μετάδοσης (back propagation algorithm). Επίσης για την εκπαίδευση των νευρωνικών δικτύων μπορούν να χρησιμοποιηθούν τόσο γενετικοί αλγόριθμοι όσο και στατιστικές μέθοδοι Bayes.

2.3 ΕΚΠΑΙΔΕΥΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Μια από τις πιο βασικές ιδιότητες των Νευρωνικών Δικτύων είναι η ικανότητά τους για εκπαίδευση. Η εκπαίδευση αυτή επιτυγχάνεται μέσω της ανταλλαγής τιμών και βαρών, που αποσκοπεί στη βαθμιαία σύλληψη της πληροφορίας η οποία στη συνέχεια θα είναι διαθέσιμη προς ανάκτηση. Υπάρχουν, βέβαια, πολλοί αλγόριθμοι που η εφαρμογή τους έχει στόχο την προσαρμογή των τιμών των βαρών ενός Τεχνητού Νευρωνικού Δικτύου. Όλες οι μέθοδοι μάθησης μπορούν να καταταχτούν σε τρείς κατηγορίες : τη μάθηση με επίβλεψη και τη μάθηση χωρίς επίβλεψη.

Μάθηση γενικά καλείται η διαδικασία αναπροσαρμογής ενός συστήματος με στόχο να βελτιστοποιήσει κάποιο κριτήριο καταλληλότητας για την επίλυση ενός συγκεκριμένου προβλήματος. Ένα νευρωνικό δίκτυο λέμε ότι μαθαίνει ή εκπαιδεύεται όταν αλλάζει τις εσωτερικές του παραμέτρους (συνήθως τα συναπτικά βάρη του) με στόχο να ελαχιστοποίηση ένα κριτήριο όπως για παράδειγμα το μέσο τετραγωνικό σφάλμα.

Εκπαίδευση με Εποπτεία: Το σύνολο προτύπων περιέχει τόσο διανύσματα εισόδου όσο και διανύσματα εξόδου. Ο στόχος της εκπαίδευσης είναι η ελαχιστοποίηση του καθολικού σφάλματος και η προσαρμογή των τιμών των σωστών βαρών με τέτοιο τρόπο ώστε όταν διαβιβάζουμε στο δίκτυο τη σωστή είσοδο αυτό να αναπαράγει τη σωστή έξοδο.

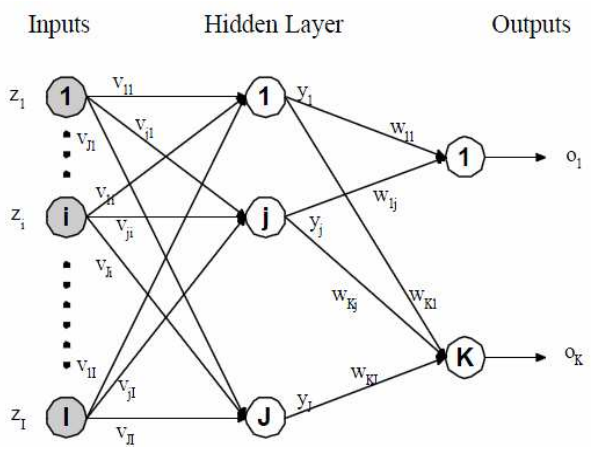
Εκπαίδευση χωρίς Εποπτεία: Το σύνολο προτύπων περιέχει μόνο διανύσματα εισόδου τα οποία ομαδοποιούνται με βάση τα κοινά τους χαρακτηριστικά. Ο στόχος είναι και εδώ η ελαχιστοποίηση μιας συνάρτησης σφάλματος.

Έχουν αναπτυχθεί αρκετοί τύποι ΤΝΔ οι οποίοι χρησιμοποιούνται σε ένα μεγάλο φάσμα εφαρμογών, όπως η διάγνωση ασθενειών, η αναγνώριση προτύπων, η εξόρυξη γνώσης, η σύνθεση μουσικής, η επεξεργασία εικόνας, ο έλεγχος ρομποτικής, η έγκριση πιστοληπτικής ικανότητας, η συμπίεση δεδομένων και πολλά άλλα.

2.4 BACK PROPAGATION NEURAL NETWORK

Το Back Propagation Neural Network αποτελείται από νευρώνες οργανωμένους σε ένα στρώμα εισόδου και ένα στρώμα εξόδου και πολλά κρυμμένα στρώματα νευρώνων. Οι νευρώνες εκτελούν ένα είδος υπολογισμού. Χρησιμοποιούν εισόδους για να υπολογιστεί μια έξοδος που αντιπροσωπεύει το σύστημα. Οι έξοδοι δίνονται σε έναν επόμενο νευρώνα. Το τόξο δείχνει σε πιο νευρώνα θα δοθεί η έξοδος με το κάθε τόξο να έχει ένα βάρος.

Αρχικά, το δίκτυο προβλέπει μια έξοδο για ένα διάνυσμα εισόδου για το οποίο η πραγματική έξοδος είναι γνωστή. Αυτός ο συνδυασμός της γνωστής εξόδου και διανύσματος εισόδου ονομάζεται δείγμα εκπαίδευσης. Η προβλεπόμενη έξοδος συγκρίνεται με την γνωστή τιμή. Τα βάρη για τα τόξα προσαρμόζονται ανάλογα με τη πρόβλεψη του πραγματικού αποτελέσματος. Στη πιο κάτω εικόνα φαίνεται η γενική τοπολογία ενός Back Propagation Neural Network.



Κάθε σήμα σταθμίζεται όπως αυτό δίνεται από το στρώμα εισόδου στο πρώτο κρυφό επίπεδο. Καθώς τα νέα σήματα φτάνουν σε ένα νευρώνα στο κρυφό επίπεδο όλα τα σήματα που λαμβάνονται από ένα νευρώνα αθροίζονται. Η διαδικασία αυτή μπορεί να αντιμετωπίζεται ως ένα διάνυσμα πολλαπλασιασμού του βάρος w και του διανύσματος σήματος yprevious του προηγούμενου στρώματος . Στο κρυφό επίπεδο τα νέα σήματα υπολογίζονται και δίνονται στο επόμενο κρυφό επίπεδο. Η διαδικασία αυτή συνεχίζεται έως ότου το στρώμα εξόδου έχει επιτευχθεί.

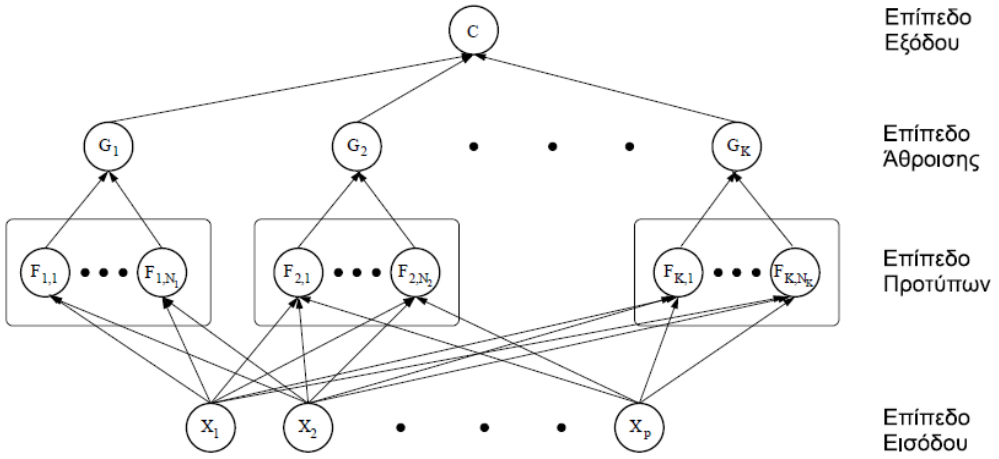
Ένα πλεονέκτημα του Back Propagation Neural Network είναι η ευελιξία του. Επίσης . Μπορούν να χρησιμοποιηθούν για την αναγνώριση προτύπων, καθώς και για τη λήψη αποφάσεων προβλημάτων. Ένα άλλο πλεονέκτημα είναι, όπως και για κάθε άλλο νευρωνικό δίκτυο ότι η διαδικασία είναι εξαιρετικά παράλληλη και συνεπώς η χρήση της παράλληλης επεξεργασίας είναι δυνατό και μειώνει τον απαιτούμενο χρόνο για υπολογισμούς . Επίσης έχει λίγες παραμέτρους για ρύθμιση.

Εκτός από πλεονεκτήματα το Back Propagation Neural Network έχει και σημαντικά μειονεκτήματα. Η κατάρτιση του δικτύου είναι εξαιρετικά χρονοβόρα. Όταν όμως καταρτιστεί το δίκτυο η εκτέλεση του είναι πολύ σύντομη. Σε μερικές περιπτώσεις μια δυσάρεστη πτυχή είναι ότι το μέγεθος των δεδομένων εκπαίδευσης είναι πολύ μεγάλο.

2.5 PROBABILISTIC NEURAL NETWORKS (PNN)

Τα Probabilistic Neural Networks (Πιθανοτικά Νευρωνικά Δίκτυα) αποτελούν μια κλάση ΤΝΔ που συνδυάζουν κάποια από τα επιθυμητά στοιχεία της στατιστικής αναγνώρισης προτύπων και των ΤΝΔ πρόσθιας τροφοδότησης και χρησιμοποιούνται κυρίως σε προβλήματα ταξινόμησης.

Τα PNN χρησιμοποιούν τον κανόνα ταξινόμησης του Bayes για την ταξινόμηση προτύπων, ενσωματώνουν επίσης τη μη παραμετρική εκτίμηση της συνάρτησης πυκνότητας πιθανότητας του Parzen και επιπλέον μπορεί να υπολογιστεί η συνεισφορά του κάθε νευρώνα του PNN στην τελική ταξινόμηση, κάτι που δεν είναι εφικτό στα περισσότερα νευρωνικά δίκτυα. Το κλασσικό PNN μπορεί να θεωρηθεί ως μια «ευφυής μνήμη» καθώς κάθε παρατήρηση του συνόλου εκπαίδευσης αποθηκεύεται σε ένα τεχνητό νευρώνα του PNN. Τα PNN απαιτούν ένα πολύ μικρό χρόνο εκπαίδευσης για την κατασκευή τους, καθώς δεν απαιτείται εκτίμηση των συντελεστών βάρους παρά μόνο μια απλή προσπέλαση όλων των παρατηρήσεων του συνόλου εκπαίδευσης. Βέβαια, αυτά τα επιθυμητά χαρακτηριστικά έρχονται μαζί με το κόστος των μεγάλων απαιτήσεων σε μνήμη και του σχετικά αργού χρόνου εκτέλεσης της ταξινόμησης μιας άγνωστης παρατήρησης σε κάποια από τις προκαθορισμένες κλάσεις.



Η δομή ενός PNN είναι παρόμοια με αυτή ενός ΤΝΔ πρόσθιας τροφοδότησης αλλά είναι πάντοτε περιορισμένη σε τέσσερα επίπεδα όπως φαίνεται και στο Σχήμα.

Τα επίπεδα του PNN είναι :

• Επίπεδο Εισόδου (Input Layer)

• Επίπεδο Προτύπων (Pattern Layer)

• Επίπεδο Άθροισης (Summation Layer)

• Επίπεδο Εξόδου (Output Layer)

Το επίπεδο εισόδου έχει ένα νευρώνα για κάθε χαρακτηριστικό γνώρισμα των δεδομένων εισόδου. Το επίπεδο προτύπων έχει έναν νευρώνα προτύπου για κάθε πρότυπο εκπαίδευσης. Κάθε νευρώνας προτύπου περνά το γινόμενο του διανύσματος βαρών και του δεδομένου παραδείγματος για την ταξινόμηση μέσω της συνάρτησης πυρήνα ενεργοποίησης. Το επίπεδο άθροισης λαμβάνει τα αποτελέσματα από τους νευρώνες προτύπων που συνδέονται με μια δεδομένη κατηγορία. Το επίπεδο εξόδου έχει τόσους νευρώνες όσες και οι υπάρχουσες κλάσεις/κατηγορίες. Σε αυτό το στρώμα επιλέγεται η κλάση/κατηγορία με τη μεγαλύτερη τιμή εξόδου.

3 GENERAL REGRESSION NEURAL NETWORK

3.1 ΑΝΑΛΥΣΗ ΠΑΛΙΝΔΡΟΜΗΣΗΣ-REGRESSION ANALYSIS

Η ανάλυση παλινδρόμησης είναι ένα στατιστικό εργαλείο για τη διερεύνηση των σχέσεων μεταξύ των μεταβλητών. Συνήθως, ο ερευνητής επιθυμεί να πληροφορηθεί την αιτιώδη επίδραση μιας μεταβλητής πάνω στην άλλη. Για παράδειγμα η επίδραση της αύξησης της τιμής από τη ζήτηση, ή η επίδραση των μεταβολών της προσφοράς χρήματος κατά το ποσοστό του πληθωρισμού. Για την διερεύνηση αυτών των ζητημάτων, ο ερευνητής συγκεντρώνει στοιχεία για τις υποκείμενες μεταβλητές που ενδιαφέρουν και χρησιμοποιεί παλινδρόμηση, για να εκτιμηθεί η ποσοτική επίδραση της αιτιώδους μεταβλητής από τη μεταβλητή που επηρεάζεται. Ο ερευνητής εκτιμά επίσης, τη «στατιστική σημαντικότητα» των εκτιμώμενων σχέσεων, δηλαδή, το βαθμό εμπιστοσύνης ότι η πραγματική σχέση είναι κοντά στην εκτιμώμενη σχέση.

Τα προβλήματα παλινδρόμησης ασχολούνται με την εκτίμηση μιας τιμής εξόδου με βάση τις τιμές εισόδου. Η παλινδρόμηση μπορεί να εφαρμοστεί εκτός από προβλήματα κατηγοριοποίησης και σε άλλες εφαρμογές, όπως η πρόβλεψη. Όταν η τεχνική της παλινδρόμησης εφαρμόζεται για να λύσει προβλήματα κατηγοριοποίησης, οι τιμές εισόδου είναι τα χαρακτηριστικά των δεδομένων και οι τιμές εξόδου αναπαριστούν το χαρακτηριστικό της κατηγορίας. Η παλινδρόμηση μπορεί να εκτελεστεί χρησιμοποιώντας πολλούς διαφορετικούς τύπους τεχνικών, συμπεριλαμβανομένων των Νευρωνικών Δικτύων. Στην πραγματικότητα αυτό που κάνει είναι να δέχεται ένα σύνολο από δεδομένα και να ταιριάζει αυτά τα δεδομένα σε μια εξίσωση.

Η παλαιότερη μορφή της παλινδρόμησης ήταν η μέθοδος των ελαχίστων τετραγώνων ,η οποία δημοσιεύθηκε από τον Legendre το 1805, και από τον Gauss το 1809. Ο Legendre και ο Gauss εφάρμοσαν και δύο τη μέθοδος για το πρόβλημα του προσδιορισμού, από τις αστρονομικές παρατηρήσεις, τις τροχιές των φορέων σχετικά με τον Ήλιο. Ο Gauss δημοσίευσε την περαιτέρω ανάπτυξη της θεωρίας των ελαχίστων τετραγώνων το 1821, συμπεριλαμβανομένης μιας έκδοσης του θεωρήματος Gauss-Markov.

Ο πατέρας της ανάλυσης παλινδρόμησης είναι ο Carl F. Gauss (1777-1855), ένας Γερμανός μαθηματικός, γνωστός για ευρεία συμβολή του στην επιστήμη. Για τη διδακτορική του διατριβή (στο Πανεπιστήμιο του Γκέτινγκεν) υπέβαλε μια απόδειξη ότι κάθε αλγεβρική εξίσωση έχει τουλάχιστον μία ρίζα, ή λύση. Αυτό το θεώρημα, το οποίο είχε αμφισβητηθεί από μαθηματικούς για αιώνες,

εξακολουθεί να ονομάζεται «το θεμελιώδες θεώρημα της άλγεβρας». Το κύριο έργο του Gauss είναι τα μαθηματικά και η μαθηματική φυσική, αν έκανε πολύτιμη συμβολή τόσο θεωρητική και πρακτική αστρονομία.

Στη στατιστική, η ανάλυση παλινδρόμησης περιλαμβάνει τεχνικές για τη μοντελοποίηση και την ανάλυση πολλών μεταβλητών, όταν δίνεται έμφαση στη σχέση ανάμεσα σε μια εξαρτημένη μεταβλητή και μία ή περισσότερες ανεξάρτητες μεταβλητές. Πιο συγκεκριμένα, η ανάλυση παλινδρόμησης μας βοηθά να καταλάβουμε πώς η χαρακτηριστική αξία της εξαρτημένης μεταβλητής αλλάζει όταν κάθε μία από τις ανεξάρτητες μεταβλητές είναι ποικίλες, ενώ οι άλλες ανεξάρτητες μεταβλητές να παραμένουν σταθερές.

Σε όλες τις περιπτώσεις, η εκτίμηση είναι συνάρτηση των ανεξάρτητων μεταβλητών που ονομάζεται συνάρτηση παλινδρόμησης. Στην ανάλυση παλινδρόμησης, είναι επίσης ενδιαφέρον να χαρακτηρίζουν τη διακύμανση της εξαρτημένης μεταβλητής γύρω από τη συνάρτηση παλινδρόμησης, η οποία μπορεί να περιγραφεί από μια κατανομή πιθανότητας. Η ανάλυση παλινδρόμησης χρησιμοποιείται ευρέως για την πρόβλεψη, όπου η χρήση της έχει ουσιαστική επικάλυψη με τον τομέα της μηχανικής μάθησης.

3.2 ΕΙΣΑΓΩΓΗ ΣΤΑ ΓΕΝΙΚΕΥΜΕΝΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ

Το General Regression Neural Network (GRNN) εκτελεί γενικές και μη γραμμικές παλινδρομήσεις και εκτίμηση εξόδου συνεχούς συνάρτησης. Είναι ένα ισχυρότατο εργαλείο παλινδρόμησης με μια δυναμική δομή . Υλοποιεί τον εκτιμητή παλινδρόμησης/οπισθοδρόμισης Nadaraya-Watson. Το GRNN βασίζεται στους μη παραμετρικούς στατιστικούς εκτιμητές πυκνότητας Parzen kenels. Εφαρμόζεται για προβλήματα ταξινόμησης με ιδιαίτερη επιτυχία γιατί η ταξινόμηση είναι ένα είδος regression αλλά με διακριτές εξόδους. Μπορεί να εκτελέσει διεργασίες παλινδρόμησης και να κατασκευάσει ένα μοντέλο παλινδρόμησης για έρευνα της συσχέτισης μεταξύ μιας εξαρτώμενης μεταβλητής και μιας ή περισσοτέρων ανεξάρτητων μεταβλητών.

Το πρώτο στρώμα του GRNN αποτελείται από νευρώνες τόσους όσου είναι και το μέγεθος των διανυσμάτων εισόδου. Στο δεύτερο στρώμα, που είναι το στρώμα προτύπων, περιέχει όλα τα δείγματα προτύπων εκπαίδευσης. Στο τρίτο στρώμα, αλλιώς το στρώμα αθροίσματος περιέχεται ο παρονομαστής και ο αριθμητής του εκτιμητή Nadaraya-Watson έτσι ώστε στο τέταρτο στρώμα που είναι η έξοδος να είναι ένας σταθμισμένος μέσος όρος των τιμών των στόχων των περιπτώσεων εκπαίδευσης κοντά στο δοσμένο άγνωστο παράδειγμα εισόδου. Η επιλογή των παραμέτρων σίγμα (σ) των συναρτήσεων πυρήνα

καθορίζει το εύρος μιας περιοχής στο χώρο εισόδου στον οποίο κάθε συνάρτηση αποκρίνεται. Οι παράμετροι σίγμα επιλέγονται συνήθως με cross-validation ή με ειδικές μεθόδους.

Τα GRNN μπορούν να λύσουν οποιοδήποτε πρόβλημα που μπορεί να λυθεί από ένα στατιστικό μοντέλο παλινδρόμησης χωρίς να πρέπει να ταιριάξουν τα δεδομένα εκπαίδευσης σε ένα συγκεκριμένο τύπο παραμετρικού μοντέλου που δίνεται εκ των προτέρων. Πρέπει να σημειωθεί ότι με τη βοήθεια των GRNN μπορεί να υπολογιστεί όχι μόνο η υπό συνθήκη (conditional) μέση τιμή σε άγνωστες θέσεις y αλλά ακόμη και υψηλότερες k ροπές επίσης.

$Y^k(X) = \Sigma (Yi)^k \exp(-D(X,Xi)) / \Sigma \exp(-D(X,Xi))$ για i δείγματα

Για την εκπαίδευση του GRNN απαιτείτε ένα λεπτομερώς παρασκευασμένο καλά αντιπροσωπευτικό σύνολο δεδομένων για την εκτίμηση της άγνωστης συνάρτησης G. Αυτό είναι και το βασικό μειονέκτημα του GRNN.

Η απλότητα της δομής των δικτύων GRNN βοήθησε στην επίλυση πολλών προβλημάτων σε διάφορους τομείς όπως η ταξινόμηση προτύπων, η επεξεργασία εικόνας, διάγνωση ελαττωμάτων μηχανημάτων, οικονομικές προβλέψεις και πολλές εργασίες στην πρόβλεψη νομισματικής κρίσης, βιοιατρικές εφαρμογές και διάφορες άλλες.

3.3 ΜΟΝΤΕΛΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ

Η παλινδρόμηση είναι μια τεχνική μοντελοποίησης για την έρευνα της συσχέτισης μεταξύ μιας εξαρτώμενης μεταβλητής και μια η περισσοτέρων ανεξάρτητων μεταβλητών. Το GRNN εκτελεί διεργασίες παλινδρόμησης για να κατασκευάσει ένα τέτοιο μοντέλο παλινδρόμησης.

Κάθε μοντέλο παλινδρόμησης περιλαμβάνει τρεις μεταβλητές :

- Το διάνυσμα β όπου αντιστοιχεί στις άγνωστες παραμέτρους συσχέτισης
- Το διάνυσμα X που αντιστοιχεί στις ανεξάρτητες μεταβλητές.
- Η εξαρτώμενη μεταβλητή Y.

Το μοντέλο παλινδρόμησης συσχετίζει το Y σε μια συνάρτηση των X και β.

$Y \approx f(X, \beta)$

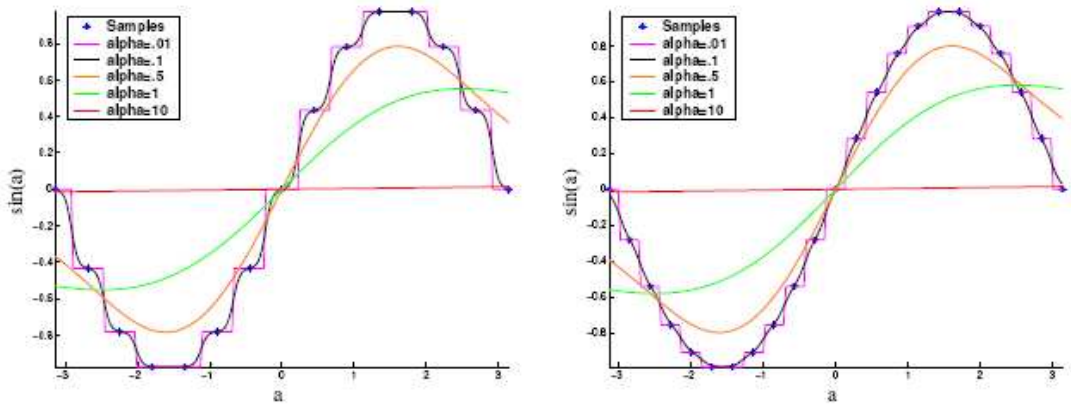
Για την ανάλυση παλινδρόμησης, καθορίζεται η μορφή της συνάρτησης παλινδρόμησης *f*. Μερικές φορές η μορφή της *f* βασίζεται στη γνώση για τη συσχέτιση μεταξύ του *Y* και του *X* που δεν στηρίζεται στα δεδομένα. Εάν τέτοια γνώση δεν είναι διαθέσιμη, τότε επιλέγεται μια κατάλληλη μορφή για την

συνάρτηση *f*. Γενικά για την μοντελοποίηση της απόκρισης Y ως συνάρτηση μιας ή περισσότερων μεταβλητών οδηγών (X_1, X_2,..., X_p) η συναρτησιακή μορφή είναι:

$$Y_i = f(X_i, \beta) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{pi} + \varepsilon$$

Εάν υπάρχει μόνο μια μεταβλητή, X, τότε μιλάμε συνήθως για απλή ανάλυση γραμμικής παλινδρόμησης. Ο όρος *ε* αναφέρεται ως "όρος τυχαίου σφάλματος".

Η Ανάλυση Παλινδρόμησης βοηθά να καταλάβουμε πώς αλλάζει η τιμή της εξαρτώμενης μεταβλητής *Y* όταν μεταβάλλεται μία από τις ανεξάρτητες μεταβλητές X, ενώ οι άλλες ανεξάρτητες μεταβλητές κρατιούνται σταθερές. Συνήθως, επιδιώκεται να εξακριβωθεί η αιτιώδης επίδραση μιας μεταβλητής επάνω σε άλλη. Για παράδειγμα η επίδραση της αύξησης τιμών προϊόντων με την προσφορά/ζήτηση. Η επίδραση της παροχής χρημάτων στο ρυθμό πληθωρισμού. Για τέτοια ζητήματα, συγκεντρώνονται τα δεδομένα που αφορούν τις μεταβλητές ενδιαφέροντος και υιοθετείται η παλινδρόμηση για να υπολογίσει την ποσοτική επίδραση των μεταβλητών επάνω στη μεταβλητή που επηρεάζουν. Αξιολογείται επίσης η "στατιστική σημασία" των κατ' εκτίμηση συσχετίσεων, δηλαδή ο βαθμός εμπιστοσύνης (confidence) ότι η αληθινή συσχέτιση είναι κοντά στην κατ' εκτίμηση. Η ανάλυση παλινδρόμησης για πρόβλεψη και πρόγνωση έχει ουσιαστική επικάλυψη με τον τομέα της μηχανικής μάθησης.



Εικόνα. Παραδείγματα της λειτουργίας παλινδρόμησης για την συνάρτηση ημιτόνου με τον εκτιμητή Nadaraya-Watson με χρήση αραιών δειγμάτων που αυξάνονται σταδιακά. Χρησιμοποιούνται γκαουσσιανοί πυρήνες με διαφορετικές παραμέτρους διακύμανσης (σ2) κλιμακωμένες με το Alpha. Τα μεγάλα πλάτη, Alpha = 10, πλησιάζουν έναν γραμμικό εκτιμητή, ενώ μικρότερα πλάτη εξειδικεύονται γύρω από κάθε δείγμα.

3.4 ΓΕΝΙΚΗ ΠΑΛΙΝΔΡΟΜΗΣΗ

Εάν το f(x,y) είναι η συνδυασμένη συνεχής συνάρτηση πυκνότητας πιθανότητας μιας διανυσματικής τυχαίας μεταβλητής, x, και μιας βαθμωτής μεταβλητής y και το X είναι μία δεδομένη μέτρηση της μεταβλητής x τότε η παλινδρόμηση (εκτίμηση) της τιμής της y, δίνεται από την υπό συνθήκη προσδοκία του y με δοσμένο το X (ή E[y|X])

$$\hat{Y}(X) = E[y \mid X] = \frac{\int_{-\infty}^{\infty} y f(X, y) dy}{\int_{-\infty}^{\infty} f(X, y) dy}$$

όπου

τα ολοκληρώματα τρέχουν στη μεταβλητή y

X είναι το διάνυσμα τιμών εισόδου,

$\hat{Y}(X)$ = E[y|X] είναι η αναμενόμενη τιμή εξόδου του y, με δοσμένο το διάνυσμα εισόδου X,

f(X,y) είναι η συνδυασμένη συνάρτηση πυκνότητας πιθανότητας (joint probability) X και y.

Τώρα εάν η σχέση μεταξύ του x και του y εκφράζεται σε μια συναρτησιακή μορφή με παραμέτρους τότε η παλινδρόμηση θα είναι παραμετρική. Αλλά σε γενικές περιπτώσεις η συνδυασμένη συνάρτηση πυκνότητας πιθανότητας f(x,y) είναι συνήθως άγνωστη και υπολογίζεται από τα δεδομένα παρατηρήσεων μόνο. Αυτή η μέθοδος εκτίμησης γνωστή ως μη-παραμετρική εκτίμηση πυκνότητας δεν έχει καμία σταθερή καθορισμένη μορφή για την κατ' εκτίμηση πυκνότητα. Η άγνωστη συνάρτηση πυκνότητας πιθανότητας γενικά υπολογίζεται εμπειρικά (μέσω των νευρώνων GRNN) από το δείγμα παρατηρήσεων των μεταβλητών x,y. Η γενική μορφή του εκτιμητή πυρήνων είναι η συνάρτηση:

$$f_n(x) = \frac{1}{n\lambda} \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{\sigma} \right)$$

όπου οι xi είναι ανεξάρτητες, όμοιας κατανομής τυχαίες μεταβλητές. Η συνάρτηση στάθμισης φ πρέπει να είναι φραγμένη και να ικανοποιεί κάποιους όρους [Specht, 1991]. Στα παρακάτω ως πυρήνας φ χρησιμοποιείται η Γκαουσιανή εκθετική συνάρτηση.

Βασισμένη στις τιμές δειγμάτων Xi και Yi των τυχαίων μεταβλητών x και y, το GRNN χρησιμοποιεί παράθυρα Parzen στην εκτίμηση της συνδυασμένης f(x,y)

$$\hat{f}(X,Y)=\frac{1}{(2\pi)^{(p+1)/2}\sigma^{(p+1)}}\frac{1}{n}\sum_{i=1}^{n}\exp\left[-\frac{(X-X^i)^T(X-X^i)}{2\sigma^2}\right]\exp\left[-\frac{(Y-Y^i)^2}{2\sigma^2}\right]$$

όπου το p είναι η διάσταση της διανυσματικής μεταβλητής x, το n είναι ο αριθμός παρατηρήσεων (τα πρότυπα εκπαίδευσης στο επίπεδο προτύπων), σ είναι το πλάτος του πυρήνα, Yi είναι η αντίστοιχη βαθμωτή έξοδος του κάθε διανύσματος Xi.

Με τη χρήση παραθύρων Parzen, ο εκτιμητής GRNN του y στο X είναι ο εξής Ŷ(X) :

$$\hat{Y}(X)=\frac{\sum_{i=1}^{n}\exp\left[-\frac{(X-X^i)^T(X-X^i)}{2\sigma^2}\right]\int_{-\infty}^{\infty}y\exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right]dy}{\sum_{i=1}^{n}\exp\left[-\frac{(X-X^i)^T(X-X^i)}{2\sigma^2}\right]\int_{-\infty}^{\infty}\exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right]dy}$$

Το ολοκλήρωμα $\int_{-\infty}^{\infty}y\exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right]dy$ δίνει Yi στον αριθμητή, ενώ το ολοκλήρωμα $\int_{-\infty}^{\infty}\exp\left[-\frac{(y-Y^i)^2}{2\sigma^2}\right]dy$ στον παρανομαστή δίνει μονάδα.

Η βαθμωτή συνάρτηση D_i^2 απόστασης είναι η $D_i^2=(X-X^i)^T(X-X^i)$

Αντικατάσταση του D_i^2 στην εξίσωση Ŷ(X) παράγει τον ακόλουθο εκτιμητή GRNN:

$$\hat{Y}(X)=\frac{\sum_{i=1}^{n}\exp\left[-\frac{D_i^2}{2\sigma^2}\right]Y^i}{\sum_{i=1}^{n}\exp\left[-\frac{D_i^2}{2\sigma^2}\right]}$$

Αυτή η έξοδος Ŷ(X) του GRNN, γνωστή ως εκτιμητής παλινδρόμησης Nadaraya-Watson, δίνει τον 'υπό συνθήκη μέσο' (conditional mean) της μεταβλητής y, με δοσμένη μία μέτρηση των μεταβλητών X και εφαρμόζεται σε προβλήματα με αριθμητικά δεδομένα.

Το GRNN μπορεί να χειριστεί και συνεχείς αριθμητικές μεταβλητές εξόδου και κατηγορικές μεταβλητές εξόδου με δύο κατηγορίες: για παράδειγμα ένα γεγονός ενδιαφέροντος (που κωδικοποιείται ως '1') ή όχι ('0'). Εάν η μεταβλητή εξόδου είναι δυαδική/διακριτή, τότε το GRNN υπολογίζει την πιθανότητα του γεγονότος ενδιαφέροντος. Εάν η μεταβλητή εξόδου είναι συνεχής, τότε υπολογίζει την τιμή της.

Υποθέτοντας ότι το σύνολο δεδομένων εκπαίδευσης έχει προέλθει από μετρήσεις των γνωρισμάτων Xi οι οποίες περιέχουν τυχαίο θόρυβο ei, τότε κάθε

υπό συνθήκη τιμή Z=Ŷ(X)= E[y|X] + ei, όπου τα σφάλματα ei για κάθε γνώρισμα Xi είναι ανεξάρτητα, έχουν μέση τιμή μηδέν και διακύμανση σ2(X). Άρα κάθε υπό συνθήκη μέση τιμή Zm=<Ŷ(X)> μπορεί να έχει και υπό συνθήκη διακύμανση (variance), VarZm, όπως και τυπική απόκλιση που μπορούν να υπολογισθούν από το σύνολο δεδομένων εκπαίδευσης. Η υπό συνθήκη διακύμανση της μέσης τιμής Zm=<Ŷ(X)> θα είναι [97]:

$$\text{Var Zm} = \sum_{i=1}^{n} \left(\frac{\sum_{i=1}^{n} (Z - Z_m)^2 \exp\left[- D_i^2 / (2\sigma^2)\right]}{\sum_{i=1}^{n} \exp\left[- D_i^2 / (2\sigma^2)\right]} \right) \frac{\exp\left[- D_i^2 / (2\sigma^2)\right]}{\sum_{i=1}^{n} \exp\left[- D_i^2 / (2\sigma^2)\right]}$$

Η εξίσωση μπορεί να τροποποιηθεί για να απεικονίσει και τον πρόσθετο θόρυβο στη δειγματοληψία (λάθη μέτρησης) στο νέο σημείο και το λάθος πρόβλεψης του εκτιμητή. Πρέπει να σημειωθεί ότι, γενικά, η παράμετρος εύρους sigma διαφέρει για τις εκτιμήσεις μέσης τιμής (παλινδρόμηση) από την εκτίμηση της υπό συνθήκη διακύμανσης.

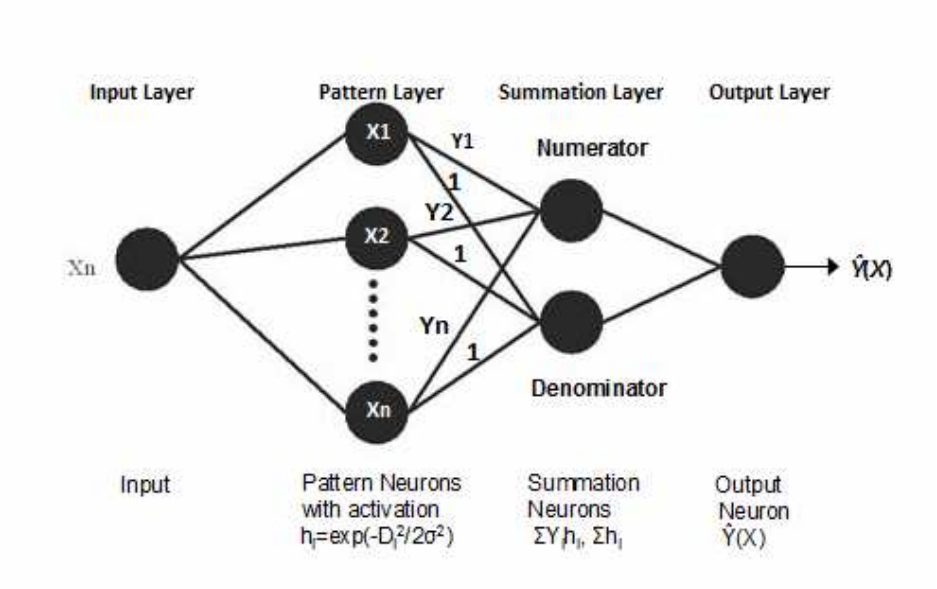
3.5 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ GRNN

Το GRNN χρησιμοποιείται για την επίλυση μη γραμμικών προβλημάτων παλινδρόμησης. Η εκτίμηση εξόδου, όταν είναι δεδομένο το διάνυσμα εισόδου x, δίνεται από τον τύπο $\boldsymbol{y = \frac{\sum yi \times hi}{\sum hi}}$ όπου στη τιμή yi ορίζεται η τιμή στόχος άμεσα από το σύνολο εκπαίδευσης. Τα αθροίσματα Σ γίνονται για όλα τα πρότυπα εκπαίδευσης (i=1..n) . Το h είναι η γκαουσιανή συνάρτηση πυρήνα και ισούται με $\boldsymbol{hi = e^{\frac{-Di^2}{2\sigma^2}}}$.

Η απόσταση Di, μεταξύ δείγματος εκπαίδευσης xi και του σημείου εισόδου x για την πρόβλεψη y, μετρά το πόσο καλά ένα δείγμα εκπαίδευσης μπορεί να αντιπροσωπεύσει τη θέση της πρόβλεψης x. Εάν η απόσταση Di, μεταξύ του δείγματος εκπαίδευσης και του σημείου πρόβλεψης είναι μικρή, το hi γίνεται μεγάλο. Για Di = 0, το hi παίρνει τιμή ένα και το σημείο αξιολόγησης αντιπροσωπεύεται καλύτερα από αυτό το δείγμα εκπαίδευσης. Η απόσταση σε όλα τα άλλα δείγματα εκπαίδευσης είναι μεγαλύτερη. Μια μεγαλύτερη απόσταση, Di, αναγκάζει τον όρο hi να γίνει μικρότερος και επομένως η συμβολή των άλλων δειγμάτων εκπαίδευσης στην πρόβλεψη είναι σχετικά μικρή.

Το Γενικευμένο Νευρωνικό Δίκτυο Παλινδρόμησης αποτελείται από τέσσερα επίπεδα όπου:

- Το πρώτο είναι το επίπεδο εισόδου. Αποτελείται με τόσους νευρώνες όσο το μέγεθος των διανυσμάτων εισόδου X. Συνδέεται πλήρως με το επίπεδο προτύπων που είναι το αμέσως επόμενο.
- Το δεύτερο επίπεδο είναι το επίπεδο προτύπων. Κάθε πρότυπο εκπαίδευσης αποτελείται από έναν νευρώνα i. Με τα πρότυπα ως κέντρα Ci υπολογίζει τις συναρτήσεις ενεργοποίησης hi=exp(-$D_i^2/2\sigma^2$).
- Το τρίτο επίπεδο είναι το επίπεδο άθροισης. Αποτελείται από δύο νευρώνες, τον αριθμητή(numerator) και τον παρονομαστή(denominator). Ο αριθμητής νευρώνας έχει συναπτικά βάρη ίσα με Yi (i=1,2…,n) και υπολογίζει έτσι το άθροισμα Σ yi hi. Ο παρονομαστής έχει βάρη ίσα με ένα, και παρέχει το άθροισμα Σ hi.
- Το τέταρτο επίπεδο αποτελείται από ένα νευρώνα. Ο νευρώνας εξόδου διαιρεί τον αριθμητή με τον παρονομαστή.



Η λειτουργία του GRNN είναι βασισμένη στην μη παραμετρική εκτίμηση πυκνότητας που χρησιμοποιείται συνήθως στη στατιστική. Ακριβέστερα, στην ανάλυση παλινδρόμησης (regression analysis) εξετάζεται ένα ζεύγος (X, Y), ενός διανύσματος X στο χώρο Rp και την αντίστοιχη συνεχή τιμή του Y στο χώρο R. Το πρόβλημα είναι να βρεθεί μια (μετρήσιμη) συνάρτηση φ: Rp → R που να αντιστοιχεί κάθε άγνωστη είσοδο X σε μία τιμή Y, έτσι ώστε το ρίσκο $E[\varphi(\vec{x}) - Y]^2$ να γίνεται ελάχιστο. Οι μη-παραμετρικές διαδικασίες και το GRNN πλησιάζουν την καλύτερη λύση καθώς το μέγεθος δειγμάτων εκπαίδευσης γίνεται μεγάλο.

3.6 ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ GRNN: ΕΠΙΛΟΓΗ ΤΗΣ ΜΕΤΑΒΛΗΤΗΣ ΣΙΓΜΑ (Σ)

Κατά την διάρκεια της εκπαίδευσης του δικτύου GRNN πρέπει να βρεθεί η άγνωστη παράμετρος σίγμα. Η επιλογή της παραμέτρου αυτής είναι πολύ σημαντική. Για την επιλογή της χρησιμοποιούνται cross-validation και Holdout μέθοδοι και η ποιότητα της εκπαίδευσης μελετάται με την βοήθεια στατιστικής ανάλυσης. Η παράμετρος πλάτους σίγμα καθορίζει το παράθυρο Parzen όπου με την χρήσης του παραθύρου Parzen μπορει να εκτιμηθεί η επιφάνεια reggeresion.

Το δίκτυο GRNN παράγει την εκτίμηση Ŷ(X) ως τον σταθμισμένο μέσο όρο των αποτελεσμάτων όλων των προτύπων Yi, όπου κάθε μία τιμή είναι σταθμισμένη εκθετικά σύμφωνα με την ευκλείδεια απόστασή της από το X. Όλα τα πρότυπα Xi της κάθε κλάσης k, συνεισφέρουν στη θέση πρόβλεψης του X κατά μία ποσότητα exp(-Di2/2σ2) όπου Di η απόσταση |Xi – X|. Όσο μεγαλύτερη η απόσταση Di τόσο μικρότερη η συνεισφορά. Όσο μεγαλύτερη η παράμετρος σίγμα, τόσο μεγαλύτερο το παράθυρο Parzen και τόσο μεγαλύτερη γίνεται η συνεισφορά περισσότερων γειτονικών σημείων Xi, καθώς απλώνεται η εμβέλειά τους. Όταν το σίγμα είναι μικρό, μόνο τα γειτονικά πρότυπα Xi διαδραματίζουν κάποιο ρόλο και η λύση συγκλίνει στην παρεμβολή (interpolation). Όταν το σίγμα τείνει στο μηδέν (σ→0), τότε δίνει πάντα το Yj του κοντινότερου Xj (δηλ., Ym → Yj εάν X → Xj). Εάν το σίγμα είναι μεγάλο τότε ακόμη και οι απόμακροι γείτονες επηρεάζουν την εκτίμηση στο X οδηγώντας σε μια πολύ ομαλή εκτίμηση. Έτσι όταν το σίγμα είναι μεγάλο υπάρχει ομαλοποίηση και η λύση συγκλίνει στην προσέγγιση.

Η χρήση της μεθόδου holdout προτείνετε για να επιλεγεί μια καλή τιμή της παραμέτρου σίγμα. Στη μέθοδο hold-out κάθε ένα πρότυπο Xi αφαιρείται από το στρώμα προτύπων του GRNN και περνά ως είσοδος από το δίκτυο για πρόβλεψη της τιμής εξόδου του Yi. Μετρείται το άθροισμα των τετραγωνικών σφαλμάτων SSE (sum of squared errors) για όλα. Κατόπιν η διαδικασία αυτή επαναλαμβάνεται για πολλές τιμές του σίγμα. Το σίγμα το οποίο το ποσό SSE είναι το ελάχιστο όλων των SSE είναι αυτό που πρέπει να χρησιμοποιηθεί για τις προβλέψεις.

Η εμπειρία χρήσης πολλών ετών έχει αποκαλύψει ότι η ακρίβεια πρόβλεψης του GRNN δεν είναι πολύ ευαίσθητη στη ρύθμιση της παραμέτρου σίγμα που πρέπει να επιλεγεί. Ως εκ τούτου στα νευρωνικά δίκτυα GRNN δεν χρειάζεται να κατασκευάσουμε και να επικυρώσουμε (με holdout, ή cross validation) πάρα πολλά μοντέλα πρόβλεψης.

4 ΠΕΡΙΓΡΑΦΗ ΠΕΙΡΑΜΑΤΩΝ

Στα πειράματα μελετήθηκαν τα νευρωνικά δίκτυα GRNN σε διάφορος πίνακες με δεδομένα από το UCI Repository of Machine Learning Databases. Ένα από τα σημαντικότερα ζητήματα σχεδιασμού ενός νευρωνικού δικτύου GRNN η εύρεση του βέλτιστου παράγοντα sigma.

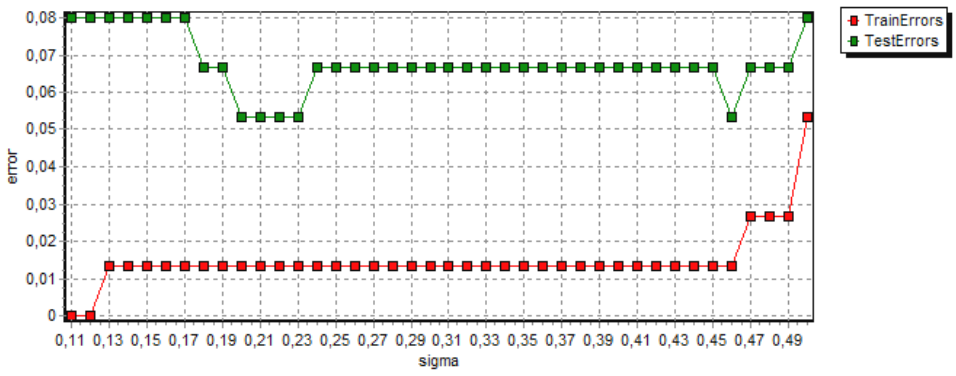
4.1 ΠΕΙΡΑΜΑΤΑ ΤΑΞΙΝΟΜΗΣΗΣ (CLASSIFICATION)

4.1.1 Iris Data Set

Αυτό το σύνολο δεδομένων αναφέρεται σε ένα τύπο λουλουδιού, τον κρίνο. Τα δεδομένα αυτά αναφέρονται σε 3 διαφορετικές κατηγορίες κρίνων (Iris setosa, Iris virginica και Iris versicolor). Κάθε κατηγορία περιέχει 50 παραδείγματα. Κάθε δείγμα αποτελείται από 4 χαρακτηριστικά που αφορούν το κρίνο.

Πιο κάτω βλέπουμε τις διάφορες τιμές του train error και test error καθώς και τις τιμές του σίγμα σε ένα εύρος τιμών από το 0,5 έως το 0,1 με το ενδιάμεσο βήμα να είναι 0,01. Το βέλτιστο σίγμα με βάση αυτές τι τιμές είναι σ=0,20 όπου σε αυτή την τιμή το σφάλμα ταξινόμησης ελαχιστοποιείται με τιμή testError=0,0533.

0,3	0,0133333333333333	0,0666666666666667
0,29	0,0133333333333333	0,0666666666666667
0,28	0,0133333333333333	0,0666666666666667
0,27	0,0133333333333333	0,0666666666666667
0,26	0,0133333333333333	0,0666666666666667
0,25	0,0133333333333333	0,0666666666666667
0,24	0,0133333333333333	0,0666666666666667
0,23	0,0133333333333333	0,0533333333333333
0,22	0,0133333333333333	0,0533333333333333
0,21	0,0133333333333333	0,0533333333333333
0,2	0,0133333333333333	0,0533333333333333
0,19	0,0133333333333333	0,0666666666666667
0,18	0,0133333333333333	0,0666666666666667
0,17	0,0133333333333333	0,08
0,16	0,0133333333333333	0,08
0,15	0,0133333333333333	0,08
0,14	0,0133333333333333	0,08
0,13	0,0133333333333333	0,08
0,12	0	0,08
0,11	0	0,08

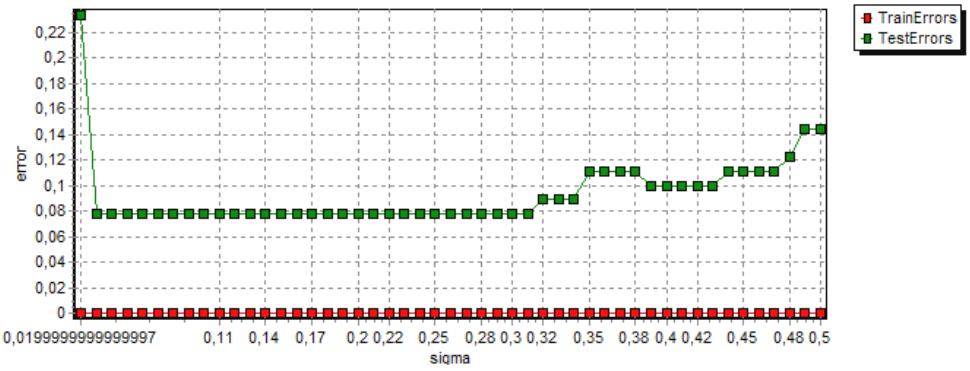


4.1.2 Wine Data Set

Σε αυτό το σύνολο δεδομένων παρουσιάζεται η χημική ανάλυση 178 κρασιών που παράγονται σε μια περιοχή στην Ιταλία. Τα κρασιά προέρχονται από τρείς διαφορετικές ποικιλίες όπου κάθε ποικιλία αντιστοιχείται σε μια κλάση. Η πρώτη κλάση περιέχει 51 δείγματα , η δεύτερη 49 και η τρίτη 78. Κάθε δείγμα αποτελείται από 13 χαρακτηριστικά.

Στο πιο κάτω πίνακα παρουσιάζονται τα train error και test error με το αντίστοιχο sigma. Οι τιμές του sigma ξεκινούν από 0,5 και φτάνουν έως και 0,01 με ενδιάμεσο βήμα το 0,01. Παρατηρούμαι πως το βέλτιστο σ(sigma) δηλαδή η τιμή για την ελαχιστοποίηση του σφάλματος ταξινόμησης είναι σ : 0.03 με αντίστοιχο test error : 0,077777777777778.

sigma	trainError	testError
0,25	0	0,077777777777778
0,24	0	0,077777777777778
0,23	0	0,077777777777778
0,22	0	0,077777777777778
0,21	0	0,077777777777778
0,2	0	0,077777777777778
0,19	0	0,077777777777778
0,18	0	0,077777777777778
0,17	0	0,077777777777778
0,16	0	0,077777777777778
0,15	0	0,077777777777778
0,14	0	0,077777777777778
0,13	0	0,077777777777778
0,12	0	0,077777777777778
0,11	0	0,077777777777778
0,09	0	0,077777777777778
0,08	0	0,077777777777778
0,07	0	0,077777777777778
0,06	0	0,077777777777778
0,05	0	0,077777777777778
0,04	0	0,077777777777778
0,03	0	0,077777777777778
0,02	0	0,233333333333333

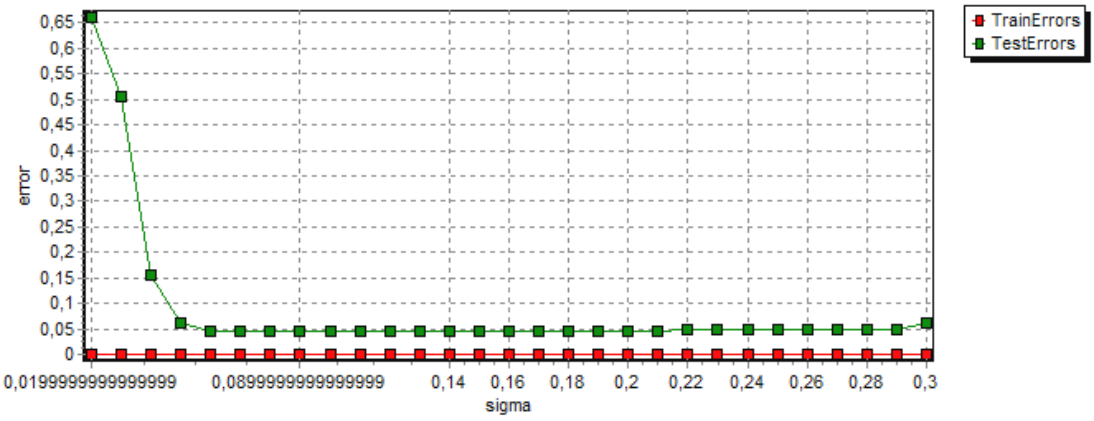


4.1.3 Dermatology data set

Η διάγνωση των Ερύθημα-επιθήλιο (erythemato-squamous) ασθενειών είναι ένα πραγματικό πρόβλημα στο κλάδο της δερματολογία. Όλοι μοιράζονται τα κλινικά χαρακτηριστικά γνωρίσματα του ερυθήματος και τις κλιμάκωσης , με τις πολύ μικρές διαφορές. Οι ασθένειες σε αυτήν την ομάδα είναι ψωρίαση, Σμηγματορροϊκή δερματίτιδα , Ομαλός λειχήνας, Πιτυρίαση rosea, Πιτυρίαση pilaris rubra.

Συνήθως μια βιοψία είναι απαραίτητη για τη διάγνωση αλλά δυστυχώς αυτές οι ασθένειες μοιράζονται πολλά ιστοπαθολογικά χαρακτηριστικά γνωρίσματα. Μια άλλη δυσκολία για τη διάγνωση είναι ότι μια ασθένεια μπορεί αρχικά να παρουσιάσει τα χαρακτηριστικά γνωρίσματα μιας άλλης ασθένειας και μπορεί να έχει τα χαρακτηριστικά γνωρίσματα στα ακόλουθα στάδια. Το σύνολο δεδομένων περιλαμβάνει 34 κατηγορίες με 358 καταχωρήσεις για την κάθε μια. Οι τιμές των ιστοπαθολογικών χαρακτηριστικών γνωρισμάτων καθορίζονται από μια ανάλυση των δειγμάτων κάτω από ένα μικροσκόπιο. Το χαρακτηριστικό γνώρισμα οικογενειακής ιστορίας έχει την αξία 1 εάν οποιαδήποτε από αυτές τις ασθένειες έχει παρατηρηθεί στην οικογένεια, και 0 ειδάλλως. Το χαρακτηριστικό γνώρισμα ηλικίας αντιπροσωπεύει απλά την ηλικία του ασθενή. Σε κάθε άλλο χαρακτηριστικό γνώρισμα (κλινικό και ιστοπαθολογικό) δόθηκε ένας βαθμός από το 0 έως 3. Εδώ, 0 δείχνουν ότι το χαρακτηριστικό γνώρισμα δεν ήταν παρόν, 3 δείχνουν το μεγαλύτερο ποσό πιθανό, και 1 ..2 δείχνει τις σχετικές ενδιάμεσες τιμές. Στον πίνακα καθώς και στην γραφική παράσταση που ακολουθεί παρατηρούμε το train error, το test error του σύνολο δεδομένων Dermatology, τις διάφορες τιμές που παίρνει το sigma, αρχίζοντας από την τιμή 0.64 μέχρι την τιμή 0.3 με ενδιάμεσο βήμα 0.01 και παρατηρούμαι πως το βέλτιστο σ(sigma) δηλαδή η τιμή για την ελαχιστοποίηση του σφάλματος ταξινόμησης είναι σ : 0.06 με αντίστοιχο test error : 0.044444.

sigma	trainError	testError
0,3	0	0,06111
0,29	0	0,05
0,28	0	0,05
0,27	0	0,05
0,26	0	0,05
0,25	0	0,05
0,24	0	0,05
0,23	0	0,05
0,22	0	0,05
0,21	0	0,044444
0,2	0	0,044444
0,19	0	0,044444
0,18	0	0,044444
0,17	0	0,044444
0,16	0	0,044444
0,15	0	0,044444
0,14	0	0,044444
0,13	0	0,044444
0,12	0	0,044444
0,11	0	0,044444
0,1	0	0,044444
0,09	0	0,044444
0,08	0	0,044444
0,07	0	0,044444
0,06	0	0,044444
0,05	0	0,061111
0,04	0	0,155555
0,03	0	0,505555
0,02	0	0,661111



4.2 ΠΕΙΡΑΜΑΤΑ ΠΑΛΙΝΔΡΟΜΗΣΗΣ (REGRESSION)

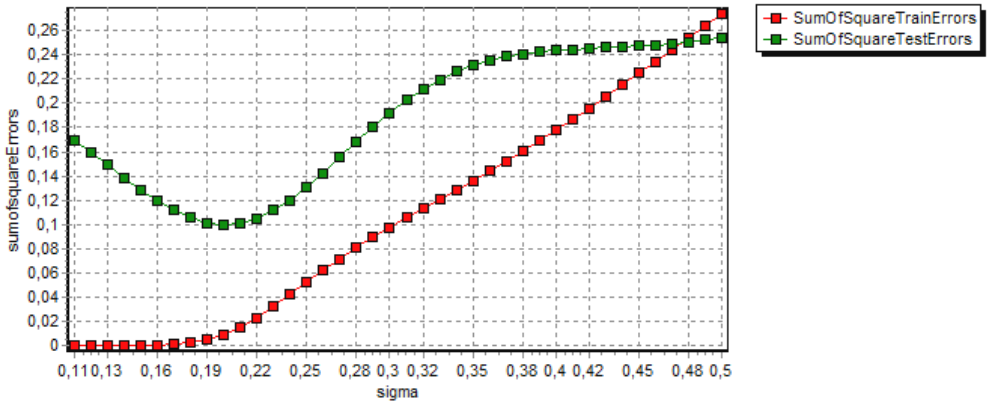
4.2.1 FOREST FIRES DATA

Οι δασικές πυρκαγιές προκαλούν σημαντικές περιβαλλοντικές ζημίες καθώς επίσης μπορούν να απειλήσουν και ανθρώπινες ζωές. Για την αποφυγή το πιο πάνω προβλημάτων έγινε προσπάθεια για την δημιουργία εργαλείων αυτόματης ανίχνευσης. Οι τρείς βασικές τάσεις είναι τα δορυφορικά δεδομένα, υπέρυθρες καπνού και οι τοπικοί αισθητήρες.

Σε αυτό ο σύνολο δεδομένων προτείνετε μια προσέγγιση που χρησιμοποιεί μετεωρολογικά δεδομένα που εντοπίστηκαν από τοπικούς αισθητήρες σε μετεωρολογικούς σταθμούς για δεδομένα που είναι γνωστά ότι επηρεάζουν τις δασικές πυρκαγιές.

Στον πίνακα καθώς και στην γραφική παράσταση που ακολουθεί παρατηρούμε το train error, το test error του σύνολο δεδομένων **forest fires**, τις διάφορες τιμές που παίρνει το sigma, αρχίζοντας από την τιμή 0.5 μέχρι την τιμή 0.1 με ενδιάμεσο βήμα 0.01 και παρατηρούμαι πως το βέλτιστο σ(sigma) δηλαδή η τιμή για την ελαχιστοποίηση του σφάλματος ταξινόμησης είναι σ : 0.2 με αντίστοιχο sumOfSquareTestErrors: 0,100026983207697.

sigma,	sumOfSquareTrainErrors	sumOfSquareTestErrors
0,3	0,0977092174738277	0,192053722791979
0,29	0,0894570719361061	0,180364596820771
0,28	0,0807747379370091	0,167896379364522
0,27	0,0715987421937506	0,15509152065274
0,26	0,0619394226384208	0,142480120148354
0,25	0,0519194394576775	0,130637910315171
0,24	0,0418039158920632	0,120134348803099
0,23	0,032001847805381	0,111480047239082
0,22	0,0230176183990194	0,105083276025128
0,21	0,0153466045623449	0,101221475677864
0,2	0,00934053118109026	0,100026983207697
0,19	0,00509908040996344	0,101481580609501
0,18	0,00244579006851936	0,105417450300286
0,17	0,00100582862275535	0,111531444130659
0,16	0,000344822624049873	0,119423483974258
0,15	9,60973090209802E-5	0,128655204657416
0,14	2,20499848466857E-5	0,138796085587244
0,13	4,89033659830414E-6	0,149411005500741
0,12	1,3987697589233E-6	0,15997760935982
0,11	4,60433659173248E-7	0,169800461289069



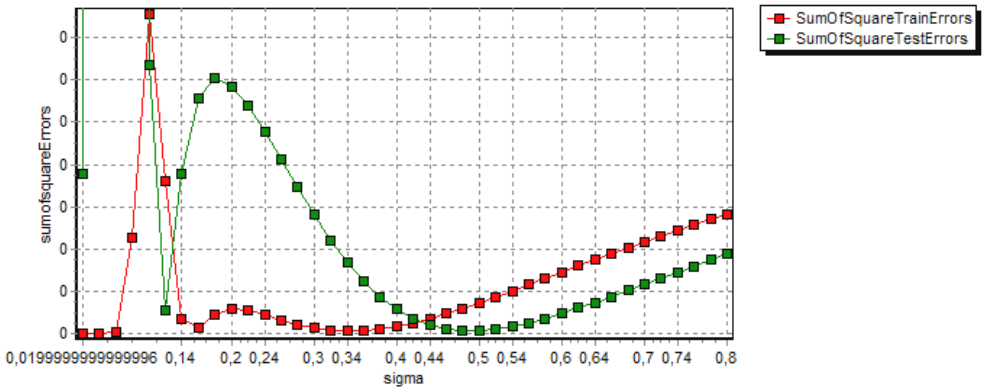
4.2.2 White wine Data Set

Σε αυτό το σύνολο δεδομένων συγκεντρώθηκαν δείγματα από άσπρο κρασί, παραγωγή της Πορτογαλικής εταιρίας "Vinho Verde". Τα κρασιά αξιολογήθηκαν από εμπειρογνώμονες κρασιού οι οποίοι τα αξιολόγησαν από το μηδέν (πολύ κακό) έως και το δέκα (πολύ εξαιρετικό). Η αξιολόγηση βασίζεται στα δεδομένα το αισθήσεων. Στα χαρακτηριστικά του συνόλου των δεδομένων δεν υπάρχουν δεδομένα σχετικά με την ιδιωτικότητα των κρασιών όπως τα είδη των σταφυλιών, την τιμή των κρασιών κ.τ.λ. αλλά μόνο φυσικοχημικά και αισθητικά χαρακτηριστικά.

Στον πίνακα καθώς και στην γραφική παράσταση που ακολουθεί παρατηρούμε το train error, το test error του σύνολο δεδομένων the White wine quality , τις διάφορες τιμές που παίρνει το sigma, αρχίζοντας από την τιμή 0.8 μέχρι την τιμή 0.01 με ενδιάμεσο βήμα 0.02 και παρατηρούμαι πως το βέλτιστο σ(sigma) δηλαδή η τιμή για την ελαχιστοποίηση του σφάλματος ταξινόμησης είναι σ : 0.5 με αντίστοιχο sumOfSquareTestErrors: 3,71015882782503E.

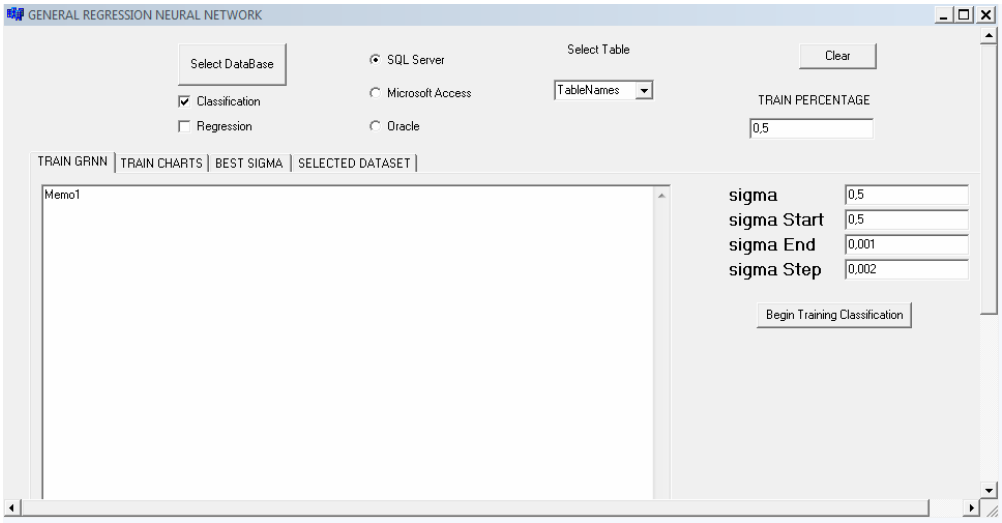
sigma	sumOfSquareTrainErrors,	sumOfSquareTestErrors
0,8	1,40986614114979E-6	9,36603212050531E-7
0,78	1,349459299366E-6	8,67325713768422E-7
0,76	1,28710729883938E-6	7,96820408677375E-7
0,74	1,22283858873184E-6	7,25338183119415E-7
0,72	1,15670450074178E-6	6,53199811015722E-7
0,7	1,08878392537197E-6	5,80810254804079E-7
0,68	1,01918874715245E-6	5,08675660349511E-7
0,66	9,48070145229429E-7	4,37423483999272E-7
0,64	8,75625878768073E-7	3,67826211627716E-7
0,62	8,02108689337002E-7	3,00829110547661E-7
0,6	7,27835961794571E-7	2,37582357862298E-7
0,58	6,53200783472574E-7	1,79477650333867E-7
0,56	5,78684513121876E-7	1,28188919706152E-7
0,54	5,04870886759161E-7	8,57158963749604E-8

0,52	4,32461495612512E-7	5,44277517153812E-8
0,5	3,62292088487729E-7	3,71015882782503E-8
0,48	2,95348455923668E-7	3,69467536777734E-8
0,46	2,32779496476951E-7	5,76004680984495E-8
0,44	1,75903315559066E-7	1,03072976349461E-7
0,42	1,26199876094749E-7	1,77612001161049E-7
0,4	8,52811969924817E-8	2,85448856301925E-7
0,38	5,48284667572213E-8	4,30386891229446E-7
0,36	3,64866534842204E-8	6,15204892259915E-7
0,34	3,17133014939176E-8	8,40883600643846E-7
0,32	4,15880092022251E-8	1,10572916953761E-6
0,3	6,65894120950038E-8	1,40455397895595E-6
0,28	1,06295193868607E-7	1,72813738651874E-6
0,26	1,58761982251876E-7	2,06311425693658E-6
0,24	2,18835676363661E-7	2,39198404837001E-6
0,22	2,73748919791064E-7	2,69155348503062E-6
0,2	2,9421320922958E-7	2,92468801838199E-6
0,18	2,28226047420739E-7	3,01360085226632E-6
0,16	6,06990760963792E-8	2,7791278046148E-6
0,14	1,66967062873821E-7	1,88333890726125E-6
0,12	1,80863419253359E-6	2,73032711433436E-7
0,1	3,7777271568713E-6	3,1771307355417E-6



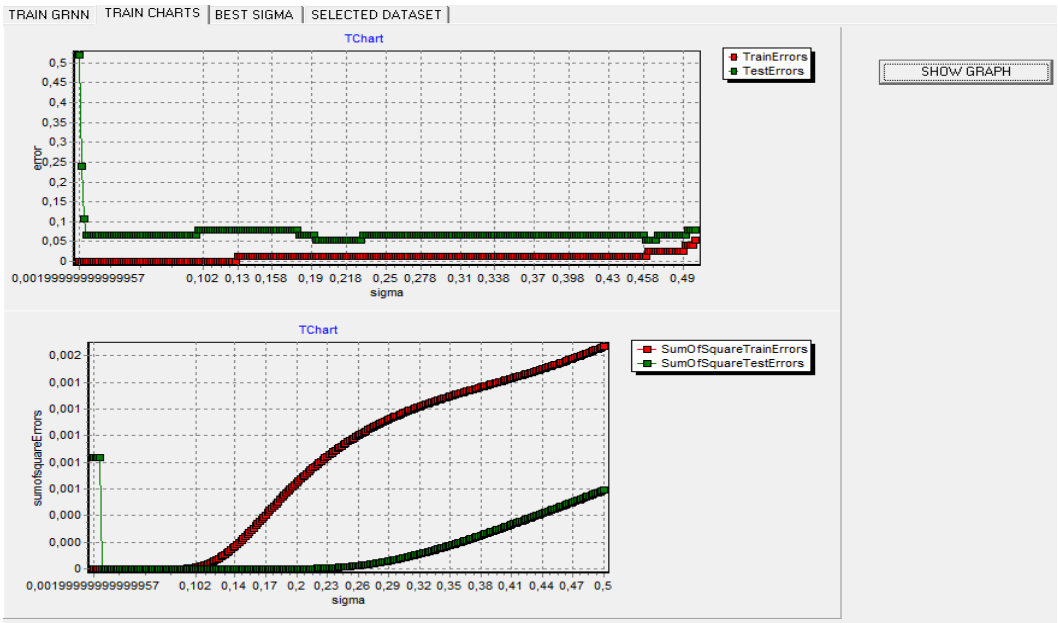
5 ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

TRAIN GRNN



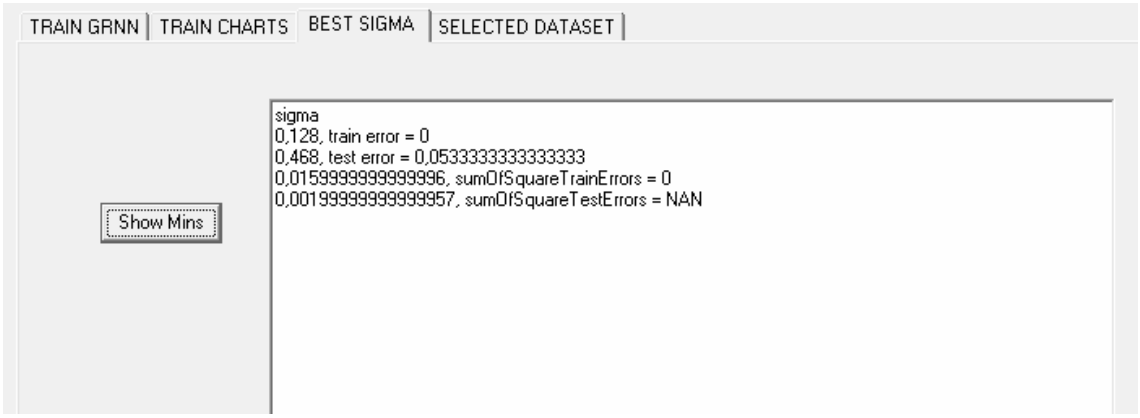
Στην πρώτη καρτέλα TRAIN GRNN μπορεί ο χρήστης να επιλέξει την βάση δεδομένων που επιθυμεί να συνδεθεί. Στη συνέχεια από το Compo Box μπορεί να επιλέξει με πιο σύνολο δεδομένων θέλει να εκπαιδεύσει το νευρωνικό δίκτυο. Ο χρήστης έχει την δυνατότητα να καθορίσει την περιοχή εύρους του sigma καθώς και το ποσοστό των δεδομένων εκπαίδευση και ελέγχου. Πατώντας το κουμπί Begin Train εμφανίζονται στο Memo το σφάλμα εκπαίδευσης(train error), σφάλμα ελέγχου (test error), άθροισμα των τετραγωνικών σφαλμάτων εκπαίδευσης(SSE train), άθροισμα των τετραγωνικών σφαλμάτων ελέγχου (SSE test)

TRAIN CHARTS



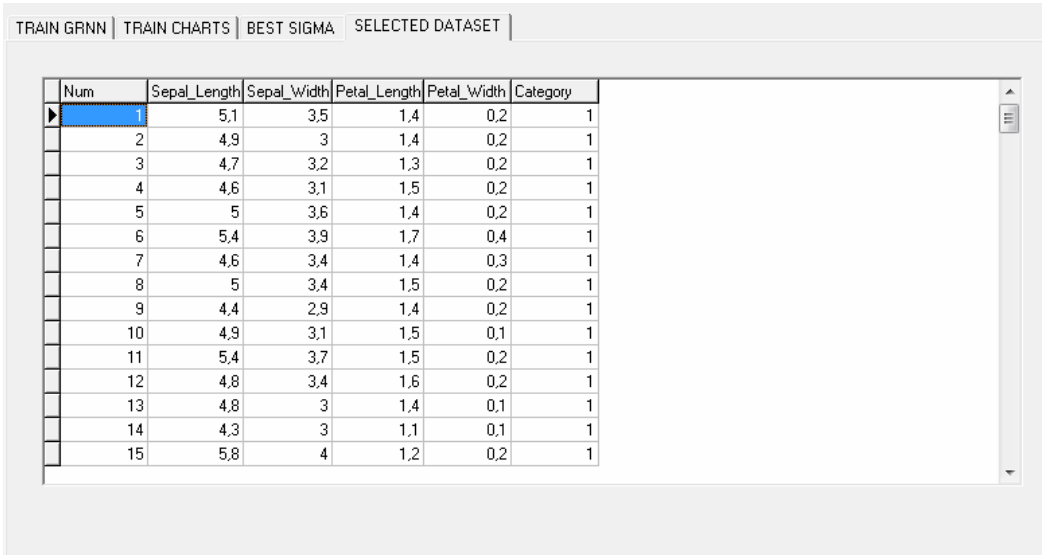
Στην καρτέλα TRAIN CHARTS πατώντας το κουμπί Show Charts εμφανίζονται οι γραφικές παραστάσεις των αποτελεσμάτων σε σχέση με το sigma.

BEST SIGMA



Στην καρτέλα BEST SIGMA εμφανίζονται τα ελάχιστα σφάλματα ελέγχου και εκπαίδευσης με το αντίστοιχο sigma.

SELECT DATASET



Num	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Category
1	5,1	3,5	1,4	0,2	1
2	4,9	3	1,4	0,2	1
3	4,7	3,2	1,3	0,2	1
4	4,6	3,1	1,5	0,2	1
5	5	3,6	1,4	0,2	1
6	5,4	3,9	1,7	0,4	1
7	4,6	3,4	1,4	0,3	1
8	5	3,4	1,5	0,2	1
9	4,4	2,9	1,4	0,2	1
10	4,9	3,1	1,5	0,1	1
11	5,4	3,7	1,5	0,2	1
12	4,8	3,4	1,6	0,2	1
13	4,8	3	1,4	0,1	1
14	4,3	3	1,1	0,1	1
15	5,8	4	1,2	0,2	1

Στην τελευταία καρτέλα SELECTED DATASET εμφανίζονται όλα τα δεδομένα του αρχικού σύνολου δεδομένων που επιλέξαμε.

```
/*************************************************************************
**/
/*  GRNN   GENERAL REGRESSION NEURAL NETWORK                             */
/*  The general regression neural network model estimates an unknown y value. */
/*  The Training set contain  samples xi (i=1,2..,numSamples)            */
/*  Each sample is a vector containing vectorXSize variables.            */
/*  The usual Gaussian exponential distribution is  used  as a Parzen kernel  */
/*  The sum of squared differences (squared Euclidean distance) used as the   */
/*  argument to the exponential function.  Sigma is the scale parameter.      */
/*  It returns the estimated value y of of the most likely real number       */
/*  produced by the given input vector x .                                   */
/*************************************************************************
**/
//---------------------------------------------------------------------------

#include <vcl.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <conio.h>
#include <math.h>
#include <string.h>    // for strcpy
#include <ctype.h>
#include <time.h>

#include <functional>
#include <iostream>


#pragma hdrstop

#include "Form1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;


/*************global variables*************************/


//-----------for loading all data and labels ----------------------
   long dataSize;              // data size (number of rows)
   int dataColumns ;              // Number of dimensions or variables in x vector
```

```
   int dataClasses ;                // Number of Classes   (0 to numOfClasses-1)
   double **allData  = NULL ;   // allData
   double **allDataRegression = NULL ;
   int *allY = NULL ;     // all_Y class labels, used double for compatibility across PNN, RBF,
GRNN
   long int *numSamplesInClass ;// size of numOfClasses vector contain number
                        // of samples in each class population
//-------- for split (stratified) to train and test----------------------
   double **trainX = NULL;          // trainData[trainSize][dataColumns]
   double **testX = NULL ;          // testData[testSize][dataColumns]
   int *trainY = NULL ;      // trainLabels[trainSize]
   int *testY = NULL;            // testLabels[testSize]
   long *trainSamplesInClass = NULL;// number of train samples in each class
   float trainPercentage;
   long trainSize , testSize;       // trainSize and testSize

//--------------------for GRNN-------------------------------------
   double *summationNeurons = NULL , sigma ;
   double bestval , trainError, testError;
   int category , bestIndex;

   double sumOfSquareTrainErrors , sumOfSquareTestErrors, outputY;
   double sigmastart, sigmaend, sigmastep ;
   int sigmaCounter ;
   double ** sigmasErrors = NULL ;

  double* regressAllY = NULL ;
  double* regressTrainY = NULL ;
  double* regressTestY = NULL  ;


//----------------------
  int sizeError;
//-----------------------

/********************* function prototypes ************************/
void grnnSummations (int , long, double** , int*, double, double*,double* );
void grnnSummations (int , long, double** , double*, double, double*,double* );
double** allocate2DArray(long rows, long columns) ;
double** loadGRNNDataTable(int*, long*, int*, int**, long**, AnsiString, char ) ;

void InitializeTrainTest(int, long, int, long*, double **,  double***,
            double***, int**, int**, long**, long*, long*, float);

void StratifiedRandomSplitTT (int, int, long*, double**, int*, double**,  double**,
                        int*, int*, long*, float)   ;
```

```
double** loadGRNNDataForRegression(int*, long*, double**, AnsiString, char );
void InitializeRegression(int, long, double**, double*, double***, double***,
 double**, double**,long*,long*, float);
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

/*****************************************************************************/
 //-------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   long int i;
   int c , k , current;

    /*real loads input data file*/
   allData = loadGRNNDataTable( &dataColumns,  &dataSize,  &dataClasses, &allY,
                &numSamplesInClass, Form1->ComboTables->Text, 1) ;

 Memo1->Clear();
//---------------------------------------------------------------------------
   trainPercentage = Edit6->Text.ToDouble() ;

  InitializeTrainTest(dataColumns, dataSize, dataClasses, numSamplesInClass, allData,
    &trainX,  &testX,  &trainY,  &testY,  &trainSamplesInClass,  &trainSize,  &testSize,
trainPercentage);

  StratifiedRandomSplitTT(dataColumns, dataClasses, numSamplesInClass,  allData, allY,
          trainX, testX, trainY, testY, trainSamplesInClass, trainPercentage)  ;


//----------------apply GPNN -------------------------------------------
  summationNeurons = (double*)malloc(2*sizeof(double));
 sigma = Edit1->Text.ToDouble();

   sigmastart = Edit2->Text.ToDouble();
   sigmaend = Edit3->Text.ToDouble();
   sigmastep = Edit4->Text.ToDouble();
   Memo1->Lines->Add("trainSize ="  +  AnsiString(trainSize)  +  ",  testSize  ="
+AnsiString(testSize) );
   Memo1->Lines->Add("sigma,                                          trainError,testError,
sumOfSquareTrainErrors,sumOfSquareTestErrors");

   sigmaCounter = 0;
   while (sigmastart>=sigmaend) {
```

```
    sigmastart = sigmastart - sigmastep ;
    sigmaCounter++ ;
  }

  sigmasErrors = allocate2DArray(sigmaCounter, 5) ;

  //reset sigmastart
  sigmastart = Edit2->Text.ToDouble();
  current = 0;

 while (sigmastart>=sigmaend) {

   sigma = sigmastart ;
   testError = 0.0;
   sumOfSquareTestErrors = 0.0 ;


 for (i=0 ; i<testSize; i++) {

     summationNeurons[0] = summationNeurons[1] = 0.0 ;
     grnnSummations(dataColumns, trainSize, trainX, trainY, sigma , testX[i],
summationNeurons);

     outputY =  (summationNeurons[0]/ summationNeurons[1]) ;

     sumOfSquareTestErrors +=  ( (double)testY[i] - outputY ) *
                        ( (double)testY[i] - outputY ) ;

    if ((int)(floor(outputY+0.5)/1) != testY[i]) testError++ ;
   sumOfSquareTestErrors = sumOfSquareTestErrors/ dataSize;

 }//end for testsize

  trainError = 0.0 ;
  sumOfSquareTrainErrors = 0.0 ;
        // with holdout
 for (i=0 ; i<trainSize; i++) {

     summationNeurons[0] = summationNeurons[1] = 0.0 ;
     grnnSummations(dataColumns, trainSize, trainX, trainY, sigma , trainX[i],
summationNeurons);

     outputY =  (summationNeurons[0]/ summationNeurons[1]) ;
     sumOfSquareTrainErrors += ( (double)trainY[i] - outputY ) *
                        ( (double)trainY[i] - outputY ) ;
```

```
     if ((int)(floor(outputY+0.5)/1) !=trainY[i]) trainError++ ;
        sumOfSquareTrainErrors = sumOfSquareTrainErrors / dataSize;
  }//end for testsize

  trainError = trainError/(double)trainSize ;
  testError = testError/(double)testSize ;

    Memo1->Lines->Add(AnsiString(sigma)+" " + AnsiString(trainError) + " " +
                AnsiString(testError) + " " + AnsiString(sumOfSquareTrainErrors) +
                " " + AnsiString(sumOfSquareTestErrors) );

   sigmasErrors[current][0] = sigma ;
   sigmasErrors[current][1] = trainError ;
   sigmasErrors[current][2] = testError ;
   sigmasErrors[current][3] = sumOfSquareTrainErrors ;
   sigmasErrors[current][4] = sumOfSquareTestErrors ;

    current++;
   sigmastart -= sigmastep ;
 }//end while
sizeError= current;

// FREE ALL IN REVERSE ORDER

  if (summationNeurons != NULL) free (summationNeurons);

  if (testY != NULL) free (testY);
  if (trainY != NULL) free (trainY);
  if (testX != NULL)  {free(testX[0]); free(testX);} // by allocate2DArray
  if (trainX != NULL) {free(trainX[0]); free(trainX);} // by allocate2DArray
  if (trainSamplesInClass != NULL) free (trainSamplesInClass) ;

  if (regressAllY != NULL) free (regressAllY);
  if (regressTrainY != NULL) free (regressTrainY);
  if (regressTestY != NULL) free (regressTestY);

  if (numSamplesInClass != NULL) free (numSamplesInClass);
  if (allY != NULL) free (allY);
  if (allData != NULL) {free(allData[0]); free(allData);} // by allocate2DArray



 ShowMessage("OK");

}
//---------------------------------------------------------------------
```

```
/*****************************************************************************
 GRNN function returns the most likely y value estimation for an input vector x
 The difference of PNN and GRNN is only in the connection to the summation units
 In GRNN every pattern unit connects to both summation units
 For the Denominator unit a single sum is performed over all pattern contributions
 for the Numerator summation unit the weight connecting each pattern unit xi
 is equal to the value of the dependent variable yi for that pattern unit.
 For a given input vector x the grnn outputs the estimated value y for that x
   y(x)= Ói yi*exp(-D(x,xi)) / Ói exp(-D(x,xi))  for i samples,
       where D(x,xi)=Ój((xj-xij)/sigmaj)^2 for j dimensions
*****************************************************************************
/
void grnnSummations (
   int dataColumns ,            // Number of dimensions or variables in x vector
   long trainSize ,          // train set size
   double **trainData ,      // trainData[trainSize][dataColumns]
   int *trainY ,             // trainLabels[trainSize]
   double sigma ,               // sigma smoothing  parameter
   double *unknownX ,        // vectorXSize vector to be classified
   double *outputSums        // vector of all the summation units
   )
{
  long i;
  int d ;
  double diff, dist,  denominatorUnit, numeratorUnit ;

  //sigma *= (double) vectorXSize ;       // Keep sigma meaningful -> scaled it

  denominatorUnit = 0.0 ;   // Cumulate here for denominator summation unit
  numeratorUnit = 0.0 ;     // Cumulate here for numerator summation unit

  for (i=0 ; i<trainSize ; i++) { // Evaluate for each sample in pattern neurons
        dist = 0.0 ;                  // Will sum distance measure here
                                      // Use squared Euclidean distance
         for (d=0 ; d<dataColumns ; d++) { // for every dimension d Compute the distance
           diff = unknownX[d] - trainData[i][d] ;
           dist += diff * diff ;
           }

        dist /= sigma * sigma ;

       // Use exponential and sum all pattern contributions
     numeratorUnit  +=  (double)trainY[i] * exp ( - dist )  ;
     denominatorUnit += exp ( - dist ) ;

        // outputSums[trainY[i]] += exp ( - dist ) ;
```

```
   }//end for

   outputSums[0] += numeratorUnit ;
   outputSums[1] += denominatorUnit ;

}


/*****************************************************************************
*
 this common function used for allocate and return a 2D array[rows][columns]
 but not like an array of row elements points to other arrays (discontinuous)
 Insted it does some things to ensure the continuity :
 1) allocates first a 1D array [rows] of pointers (like the usual way)
 2) allocates a second continuous 1D array [row*col] for all data,
 3) initializes the second array (calloc set all elements to 0.0)
 4) returns the pointer of the second to the first element of first
 5) assign each element of the first array to point at each data 'row' (record)
    by doing some pointer arithmetic.
*****************************************************************************
/
double** allocate2DArray(long rows, long columns)
{
   long i;        //to cover both short and long arrays
   double** temp;       //temp points at first [][]
   temp = (double**)malloc(rows*sizeof(double*));   //allocate temp[][]
   assert(temp != NULL);
      // temp[0] points at first [0][], but allocates memory space for all
   temp[0] = (double*)calloc(rows*columns,sizeof(double));  //calloc initialize to zeros
   assert(temp[0] != NULL);
   //assign temp pointers [] to each 'row' []  ( = prev pointer + column number)
   for (i=1; i<rows; i++) temp[i]= temp[i-1] + columns ;

   return temp;
   //  free(temp[0]);
   //  free(temp);
}
/*****************************************************************************
  this function allocate memory for train test sets initializes
  InitializeTrainTest(vectorXSize, numOfClasses, numSamplesInClass, allData, allY,
         &trainX, &testX, &trainY, &testY, & trainSamplesInClass, percentage)
*****************************************************************************
/


void InitializeTrainTest(
```

```
   int dataColumns ,              // Number of dimensions or variables in x vector
   long dataSetSize ,          // data size
   int numOfClasses ,             // Number of Classes   (0 to numOfClasses-1)
   long int *numSamplesInClass ,// size of numOfClasses vector contain number
                        // of samples in each class population
   double **allData ,          // allData
   double ***trainData ,            // output: pointer to **trainData [trainSize][vectorXSize] for
memory allocation
   double ***testData ,        // output: pointer to **testData [testSize][vectorXSize] for memory
allocation
   int **trainY ,         // output: pointer to *train_Y [trainSize]for memory allocation
   int **testY,               // output: pointer to *test_Y [testSize] for memory allocation
   long  **trainSamplesInClass , // output: pointer to *trainSamplesInClass for memory
allocation
   long *trainSize ,           // output: pointer to trainSize
   long *testSize ,            // output: pointer to testSize
   float trainPercentage            // percentage % of train set portions
   )
{
    int c ;
    long i, testSetSize, trainSetSize = 0 ;

  printf("InitializeTrainTest for stratified random spliting\n");

    (*trainSamplesInClass) = (long*) malloc(numOfClasses*sizeof(long));

       for (c=0 ; c<numOfClasses ; c++){  // Evaluate for each class
        (*trainSamplesInClass)[c] = ( (float)numSamplesInClass[c]*trainPercentage ) /1 ;   /*float
to int truncate*/
        printf("trainSamplesInClass[%d]= %i \n", c,
              (*trainSamplesInClass)[c]) ;   //trainSamplesInClass[0][i]
            }

        // find total TrainSet size and total data set
       for (c=0 ; c<numOfClasses ; c++){
          trainSetSize += (*trainSamplesInClass)[c] ;
         }
       //  testSet size
        testSetSize = dataSetSize - trainSetSize;

       // allocate memory (continous) for trainData , trainY, testData, testY
         (*trainData) = allocate2DArray(trainSetSize, dataColumns) ;
         (*testData) =  allocate2DArray(testSetSize, dataColumns) ;

         (*trainY) = (int*) malloc(trainSetSize*sizeof(int));
         (*testY) = (int*) malloc(testSetSize*sizeof(int));
```

```
    *trainSize = trainSetSize ;
    *testSize = testSetSize ;
}
/*****************************************************************************
**
 SPLIT_TO_TRAIN_TEST WITH STRATIFIED RANDOM SAMPLING
 this function randomly choose a record for train set backets or test set backets
 until both become full with equall percentage for every class so as to
 implement stratification   (equal percentage for every class)
*****************************************************************************
*/

void StratifiedRandomSplitTT (
   int vectorXSize ,              // Number of dimensions or variables in x vector
   int numOfClasses ,             // Number of Classes   (0 to numOfClasses-1)
   long *numSamplesInClass ,   // size of numOfClasses vector contain number
                          // of samples in each class population
   double **allData ,          // allData
   int *allY ,                 // all_Y
   double **trainData ,        // trainData
   double **testData ,         // testData
   int *trainY ,               // train_Y
   int *testY,                    // test_Y
   long *trainSamplesInClass , // ((float)num_Samples[i]*percentage)/1   /*float to int truncate*/
   float trainPercentage       // percentage % of train set portions
   )
{
  long numOfcases, icase;
  int d,  iclass  ;
  double choise;
  long maxTrain, maxTest, currentAll = 0, currentTrain , currentTest;
  long lastTrain = 0,  lastTest =0;

  srand(11001);  // srand ( time(NULL));
  printf("stratified random spliting\n");
   for (iclass=0 ; iclass<numOfClasses ; iclass++) { // Evaluate for each class

    numOfcases = numSamplesInClass[iclass] ;// Number of data samples of this class
    maxTrain = trainSamplesInClass[iclass];
    maxTest= numOfcases - maxTrain;

    currentTrain =0;  //limit
    currentTest=0;   //limit
    for (icase=0 ; icase<numOfcases ; icase++) {     //Do all pattern cases in this class
        choise = (double)rand()/(double)RAND_MAX;  //rixnei to zari apo 0.0 eos 1.0
```

```
        if (choise<=trainPercentage && currentTrain<maxTrain ) {
        for (d=0 ; d<vectorXSize ; d++) { // for every dimension d
          trainData[lastTrain+currentTrain][d] = allData[currentAll][d] ;
           }
        trainY[lastTrain+currentTrain]= allY[currentAll];
        currentTrain++;
        }
       else  {
            if (currentTest<maxTest) {
               for (d=0 ; d<vectorXSize ; d++) { // for every dimension d
                testData[lastTest+currentTest][d] = allData[currentAll][d] ;
                }
                 testY[lastTest+currentTest]= allY[currentAll];
               currentTest++;
             }
            else {
               for (d=0 ; d<vectorXSize ; d++) { // for every dimension d
               trainData[lastTrain+currentTrain][d] = allData[currentAll][d] ;
                }
                trainY[lastTrain+currentTrain]= allY[currentAll];
                currentTrain++;
            }
         }
         currentAll++;

         }//end for cases
      lastTrain +=currentTrain ;
      lastTest +=currentTest ;


     }//end for classes


}
/*************************************************************************************
***********
 This function loads data from file specified by 'filename'
 counts the number of samples in each class
 normalizes all colums to (Xi-Xmin)/(Xmax-Xmin)
 and produce all outputs needed for initialization
*************************************************************************************
***********/
double** loadGRNNDataTable(
      int *dataColumns ,        // output: data dimension, except last class column
      long int *dataRows ,      // output: data size
```

```
      int *dataClasses ,       // output: number of data classes
      int **Yis ,              // output: pointer to *class labels 1D array for memory allocation
      long **numSamplesInClass, // output: pointer to *numSamplesInClass for mem allocation
      AnsiString table ,       // the table name to read
      char normalizedYesNo     // for 1 normalize each column
      )
{
   // 1) loads data points from user specified TABLE
   // 2) NORMALIZE THEM
   // 3) PREPARE LABELS FOR CLASSIFICATION
   int j, Columns, Classes, classIndex, firstClass, FieldCount;
   long int i, Rows;

   double** temp, b , *Xmin=NULL, *Xmax=NULL;

  AnsiString sql;

  // find Columns, Classes, Rows, FirstClass.
  Form1->ADOQuery1->SQL->Text = "Select category from " + table + " group by category";
  Form1->ADOQuery1->Open();

  Classes = Form1->ADOQuery1->RecordCount;
  Form1->ADOQuery1->Close() ;

  sql = "Select * from " + table + " order by category";
  Form1->ADOQuery1->SQL->Text = sql;
  Form1->ADOQuery1->Open();

  Rows = Form1->ADOQuery1->RecordCount;
  FieldCount = Form1->ADOQuery1->FieldCount ;
  Columns = FieldCount - 2 ;   //first is recordID, last is categoryID
  ShowMessage("the record size is " + AnsiString(Rows));

  firstClass = Form1->ADOQuery1->Fields->Fields[FieldCount-1]->Value ;

    // get memory for  arrays
   (*numSamplesInClass) = (long*)malloc(Classes* sizeof(long));
   for(j=0; j<Classes; j++)  (*numSamplesInClass)[j]=0;

   (*Yis) = (int*)malloc(Rows*sizeof(int));   // allocate Yis[iRows]

   temp = allocate2DArray(Rows, Columns) ;    // allocate temp[Rows][Columns]

   Xmin=(double*)malloc(Columns* sizeof(double));
   Xmax=(double*)malloc(Columns* sizeof(double));
```

```
  // read the first line
    for (j=0; j<Columns; j++){
        temp[0][j] = Form1->ADOQuery1->Fields->Fields[j+1]->Value;
        Xmin[j] = temp[0][j] ;
        Xmax[j] = temp[0][j] ;
     }
     b = Form1->ADOQuery1->Fields->Fields[FieldCount-1]->Value ;
     classIndex = b / 1 ;      // double to int truncate
     classIndex -= firstClass ;  //if firstClass=0 then is ok
     (*Yis)[0] = classIndex ;
     (*numSamplesInClass)[classIndex]++;

     //move to next line
     Form1->ADOQuery1->Next();

 for (i=1; i<Rows; i++) {
       for (j=0; j<Columns; j++) {
           temp[i][j] = Form1->ADOQuery1->Fields->Fields[j+1]->Value;
           if ( temp[i][j] < Xmin[j] ) Xmin[j] = temp[i][j] ;
           if ( temp[i][j] > Xmax[j] ) Xmax[j] = temp[i][j] ;
       }
       // scan a 'double' category label
       b = Form1->ADOQuery1->Fields->Fields[FieldCount-1]->Value;
       classIndex = b / 1 ;      // double to int truncate
       classIndex -= firstClass ;  //if firstClass=0 then is ok
       (*Yis)[i] = classIndex ;
       (*numSamplesInClass)[classIndex]++;

       //move to next line
       Form1->ADOQuery1->Next();
 }

  ShowMessage("load data file OK");

if (normalizedYesNo) {
    for (i=0; i<Rows; i++)
       for (j=0; j<Columns; j++)
          temp[i][j] = (temp[i][j] - Xmin[j])/(Xmax[j]-Xmin[j]);
  }

if (Xmin != NULL) free (Xmin);
if (Xmax != NULL) free (Xmax);

 *dataColumns = Columns ;
 *dataRows = Rows ;
 *dataClasses = Classes ;
```

```
    return temp;
  Form1->ADOQuery1->Close();
}
/*****************************************************************************
*****/
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  int i   ;

  Chart1->SeriesList->Series[0]->Clear();
    Chart1->SeriesList->Series[1]->Clear();

  //    Chart1->SeriesList->Series[0]->ColorEachPoint= true;
      for (i = 0; i < sigmaCounter; i++)
        Chart1->SeriesList->Series[0]->AddXY((const double)sigmasErrors[i][0] ,    //X
       (const double)sigmasErrors[i][1],                                  //Y
        AnsiString(sigmasErrors[i][0]),              //X Label
        clRed );                        //color

      for (i = 0; i < sigmaCounter; i++)
        Chart1->SeriesList->Series[1]->AddXY((const double)sigmasErrors[i][0] ,    //X
       (const double)sigmasErrors[i][2],                                  //Y
        AnsiString(sigmasErrors[i][0]),              //X Label
        clGreen );                      //color
        //clGreen

  Chart2->SeriesList->Series[0]->Clear();
  Chart2->SeriesList->Series[1]->Clear();

  //    Chart1->SeriesList->Series[0]->ColorEachPoint= true;
      for (i = 0; i < sigmaCounter; i++)
        Chart2->SeriesList->Series[0]->AddXY((const double)sigmasErrors[i][0] ,    //X
       (const double)sigmasErrors[i][3],                                  //Y
        AnsiString(sigmasErrors[i][0]),              //X Label
        clRed );                        //color

      for (i = 0; i < sigmaCounter; i++)
        Chart2->SeriesList->Series[1]->AddXY((const double)sigmasErrors[i][0] ,    //X
       (const double)sigmasErrors[i][4],                                  //Y
        AnsiString(sigmasErrors[i][0]),              //X Label
        clGreen );                      //color
        //clGreen
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
 if (sigmasErrors != NULL)      {free(sigmasErrors[0]); free(sigmasErrors);}  // by
allocate2DArray
    if (Form1->ADOQuery1->Active==true)   Form1->ADOQuery1->Close() ;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
RadioButton1->Visible=True;
RadioButton2->Visible=True;
RadioButton3->Visible=True;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
Label4->Visible=True;
Form1->ComboTables->Visible=true;
Form1->ADOConnection1->Connected=false;


        if (Form1->CheckBox2->Checked)
          {
            Form1->ADOConnection1->ConnectionString="Provider=SQLOLEDB.1;Data
Source=GIORGOS-PC;\
            Initial Catalog=Regression;Integrated Security=SSPI;";
            Form1->ADOConnection1->Connected=true;
            ADOConnection1->Open() ;
            Form1->ADOConnection1->GetTableNames(Form1->ComboTables->Items, false);
          }

           if (Form1->CheckBox1->Checked)
             {
              Form1->ADOConnection1->ConnectionString="Provider=SQLOLEDB.1;Data
Source=GIORGOS-PC;\
               Initial Catalog=Classification;Integrated Security=SSPI;";
               Form1->ADOConnection1->Connected=true;
               ADOConnection1->Open() ;
               Form1->ADOConnection1->GetTableNames(Form1->ComboTables->Items,
false);
             }
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
Form1->OpenDialog1->Execute();
Form1->ComboTables->Visible=true;
Form1->ADOConnection1->Connected=false;
Form1->ADOConnection1->ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;\
Data Source="+ Form1->OpenDialog1->FileName +";Persist Security Info=false;\
User ID=Admin";
Form1->ADOConnection1->Connected=true;
ADOConnection1->Open() ;
Form1->ADOConnection1->GetTableNames(Form1->ComboTables->Items, false);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
  double  min[4]    ;
  int rows;
  int i,j,minpos[4];
   for (j=1;j<5;j++)
   {

      min[j-1]=sigmasErrors[0][j];
      minpos[j-1]=0;
      for(i=1;i<sizeError;i++)
      {
        if(sigmasErrors[i][j]<min[j-1])
        {
              min[j-1]=sigmasErrors[i][j];
              minpos[j-1]=i;

        }

      }
   }
 Memo2->Clear();
 Memo2->Lines->Add("sigma");
 Memo2->Lines->Add(AnsiString(sigmasErrors[minpos[0]][0])+",        train        error        =
"+AnsiString(min[0]));
```

```
  Memo2->Lines->Add(AnsiString(sigmasErrors[minpos[1]][0])+", test error =
"+AnsiString(min[1]));
  Memo2->Lines->Add(AnsiString(sigmasErrors[minpos[2]][0])+", sumOfSquareTrainErrors =
"+AnsiString(min[2]));
  Memo2->Lines->Add(AnsiString(sigmasErrors[minpos[3]][0])+", sumOfSquareTestErrors =
"+AnsiString(min[3]));
}


void grnnSummationsRegression (
    int dataColumns ,            // Number of dimensions or variables in x vector
    long trainSize ,          // train set size
    double **trainData ,      // trainData[trainSize][dataColumns]
    double *trainY ,          // trainLabels[trainSize]
    double sigma ,              // sigma smoothing  parameter
    double *unknownX ,        // vectorXSize vector to be classified
    double *outputSums        // vector of all the summation units
    )
{
   long i;
   int d ;
   double diff, dist,  denominatorUnit, numeratorUnit ;

   //sigma *= (double) vectorXSize ;        // Keep sigma meaningful -> scaled it

   denominatorUnit = 0.0 ;   // Cumulate here for denominator summation unit
   numeratorUnit = 0.0 ;     // Cumulate here for numerator summation unit

   for (i=0 ; i<trainSize ; i++) { // Evaluate for each sample in pattern neurons
         dist = 0.0 ;                    // Will sum distance measure here
                                  // Use squared Euclidean distance
          for (d=0 ; d<dataColumns ; d++) { // for every dimension d Compute the distance
             diff = unknownX[d] - trainData[i][d] ;
             dist += diff * diff ;
             }

         dist /= sigma * sigma ;

        // Use exponential and sum all pattern contributions
       numeratorUnit  +=  (double)trainY[i] * exp ( - dist )  ;
       denominatorUnit += exp ( - dist ) ;

          // outputSums[trainY[i]] += exp ( - dist ) ;
   }//end for

   outputSums[0] += numeratorUnit ;
```

```
  outputSums[1] += denominatorUnit ;

}


//---------------------------------------------------------------------
/*********************************************************************
SPLIT_TO_TRAIN_TEST WITH RANDOM SAMPLING  FOR REGRESSION
 this function randomly choose a record for train set backets or test set backets
 until both become full with equall percentage
*********************************************************************
****/
//   InitializeRegression (dataColumns, dataSize, allData, regressAllY,
//    &trainX, &testX, &regressTrainY, &regressTestY,&trainSize, &testSize, trainPercentage);

 void InitializeRegression(
   int dataColumns ,              // Number of dimensions or variables in x vector
   long dataSize ,          // data size
   double **allData ,          // allData
   double * regressAllY ,
   double ***trainData , // output: pointer to **trainData [trainSize][vectorXSize] for mem alloc
   double ***testData , //output: pointer to **testData [testSize][vectorXSize] for mem alloc
   double **regressTrainY , // output: pointer to *train_Y [trainSize]for memory allocation
   double **regressTestY,      // output: pointer to *test_Y [testSize] for memory allocation
   long *trainSize ,           // output: pointer to trainSize
   long *testSize ,            // output: pointer to testSize
   float trainPercentage            // percentage % of train set portions
   )
{
  long testSetSize, trainSetSize = 0 ;
  int d   ;
  double choise;
  long  icase, currentAll, currentTrain , currentTest;

  trainSetSize = ( (float) dataSize *trainPercentage ) / 1 ;   /*float to int truncate*/
  testSetSize = dataSize - trainSetSize ;
      // allocate memory (continous) for trainData , trainY, testData, testY
    (*trainData) = allocate2DArray(trainSetSize, dataColumns) ;
    (*testData) =  allocate2DArray(testSetSize, dataColumns) ;

   (*regressTrainY) = (double*) malloc(trainSetSize*sizeof(double));
   (*regressTestY) = (double*) malloc(testSetSize*sizeof(double));

    *trainSize = trainSetSize ;
    *testSize = testSetSize ;
```

```
    srand(11001);  // srand ( time(NULL));
    currentTrain = currentTest = currentAll = 0;  //limit

    for (icase=0 ; icase<dataSize ; icase++) {     //Do all pattern cases
        choise = (double)rand()/(double)RAND_MAX;  //rixnei to zari apo 0.0 eos 1.0

        if (choise<=trainPercentage && currentTrain<trainSetSize ) {
        for (d=0 ; d<dataColumns ; d++) { // for every dimension d
          (*trainData)[currentTrain][d] = allData[currentAll][d] ;
          }
        (*regressTrainY)[currentTrain]= regressAllY[currentAll];
        currentTrain++;
        }
       else  {
            if (currentTest<testSetSize) {
               for (d=0 ; d<dataColumns ; d++) { // for every dimension d
                (*testData)[currentTest][d] = allData[currentAll][d] ;
                }
                 (*regressTestY)[currentTest]= regressAllY[currentAll];
               currentTest++;
            }
           else {
              for (d=0 ; d<dataColumns ; d++) { // for every dimension d
              (*trainData)[currentTrain][d] = allData[currentAll][d] ;
               }
               (*regressTrainY)[currentTrain]= regressAllY[currentAll];
               currentTrain++;
           }
        }
        currentAll++;

       }//end for cases
}
//---------------------------------------------------------
double** loadGRNNDataForRegression(
      int *dataColumns ,       // output: data dimension, except last class column
      long int *dataRows ,     // output: data size
      double **Yis ,           // output: pointer to *class labels 1D array for memory allocation
      AnsiString table ,       // the table name to read
      char normalizedYesNo     // for 1 normalize each column
      )
{
   // 1) loads data points from user specified TABLE
   // 2) NORMALIZE THEM

   int j, Columns,  classIndex, firstClass, FieldCount;
```

```
  long int i, Rows;

  double** temp, b , *Xmin=NULL, *Xmax=NULL;

 AnsiString sql;

 // find Columns, Rows.

Form1->ADOQuery2->SQL->Text="Select * from " + table + " order by ID" ;
Form1->ADOQuery2->Open();

 sql = "Select * from " + table + " order by ID";
 Form1->ADOQuery1->SQL->Text = sql;
 Form1->ADOQuery1->Open();
 Rows = Form1->ADOQuery1->RecordCount;
 FieldCount = Form1->ADOQuery1->FieldCount ;
 Columns = FieldCount - 2 ;   //first is recordID, last is regression value

 ShowMessage("the record size is " + AnsiString(Rows));


  (*Yis) = (double*)malloc(Rows*sizeof(double));   // allocate Yis[iRows]

  temp = allocate2DArray(Rows, Columns) ;    // allocate temp[Rows][Columns]

  Xmin=(double*)malloc(Columns* sizeof(double));
  Xmax=(double*)malloc(Columns* sizeof(double));

    // read the first line
      for (j=0; j<Columns; j++){
          temp[0][j] = Form1->ADOQuery1->Fields->Fields[j+1]->Value;
          Xmin[j] = temp[0][j] ;
          Xmax[j] = temp[0][j] ;
       }
       (*Yis)[0] = Form1->ADOQuery1->Fields->Fields[FieldCount-1]->Value ;
       //move to next line
       Form1->ADOQuery1->Next();

  for (i=1; i<Rows; i++) {
        for (j=0; j<Columns; j++) {
            temp[i][j] = Form1->ADOQuery1->Fields->Fields[j+1]->Value;
            if ( temp[i][j] < Xmin[j] ) Xmin[j] = temp[i][j] ;
            if ( temp[i][j] > Xmax[j] ) Xmax[j] = temp[i][j] ;
        }
        (*Yis)[i] = Form1->ADOQuery1->Fields->Fields[FieldCount-1]->Value;
        //move to next line
```

```
        Form1->ADOQuery1->Next();
    }

   ShowMessage("load data file OK");
         Form1->ADOQuery1->Close();

 if (normalizedYesNo) {
     for (i=0; i<Rows; i++)
        for (j=0; j<Columns; j++)
          temp[i][j] = (temp[i][j] - Xmin[j])/(Xmax[j]-Xmin[j]);
    }

  if (Xmin != NULL) free (Xmin);
  if (Xmax != NULL) free (Xmax);

   *dataColumns = Columns ;
   *dataRows = Rows ;

    return temp;

}

void __fastcall TForm1::Button5Click(TObject *Sender)
{
    long int i;
  int c , k , current;

   /*real loads input data file   */
  allDataRegression = loadGRNNDataForRegression(&dataColumns, &dataSize,
                &regressAllY, Form1->ComboTables->Text, 1);

Memo1->Clear();
//---------------------------------------------------------------------------

 trainPercentage = Edit6->Text.ToDouble() ;

 InitializeRegression (dataColumns, dataSize, allDataRegression, regressAllY,
 &trainX, &testX, &regressTrainY, &regressTestY,&trainSize, &testSize, trainPercentage);


//----------------apply GPNN -------------------------------------------
 summationNeurons = (double*)malloc(2*sizeof(double));
sigma = Edit1->Text.ToDouble();

  sigmastart = Edit2->Text.ToDouble();
  sigmaend = Edit3->Text.ToDouble();
```

```
  sigmastep = Edit4->Text.ToDouble();
   Memo1->Lines->Add("trainSize =" + AnsiString(trainSize) + ", testSize =" +AnsiString(testSize) );
  Memo1->Lines->Add("sigma,\t trainError,\t testError,\t sumOfSquareTrainErrors,\t sumOfSquareTestErrors");

  sigmaCounter = 0;
  while (sigmastart>=sigmaend) {
     sigmastart = sigmastart - sigmastep ;
     sigmaCounter++ ;
  }
  sigmasErrors = allocate2DArray(sigmaCounter, 5) ;

  //reset sigmastart
  sigmastart = Edit2->Text.ToDouble();
  current = 0;

 while (sigmastart>=sigmaend) {

   sigma = sigmastart ;
   testError = 0.0;
   sumOfSquareTestErrors = 0.0 ;

 for (i=0 ; i<testSize; i++) {

     summationNeurons[0] = summationNeurons[1] = 0.0 ;
     grnnSummationsRegression(dataColumns, trainSize, trainX, regressTrainY, sigma , testX[i], summationNeurons);

     outputY =  (summationNeurons[0]/ summationNeurons[1]) ;

     sumOfSquareTestErrors +=  ( (double)regressTestY[i] - outputY ) *
                        ( (double)regressTestY[i] - outputY ) ;

    if ((int)(floor(outputY+0.5)/1) != regressTestY[i]) testError++ ;
     sumOfSquareTestErrors = sumOfSquareTestErrors/ dataSize;

 }//end for testsize

  trainError = 0.0 ;
  sumOfSquareTrainErrors = 0.0 ;
        // with holdout
 for (i=0 ; i<trainSize; i++) {

     summationNeurons[0] = summationNeurons[1] = 0.0 ;
```

```
      grnnSummationsRegression(dataColumns, trainSize, trainX, regressTrainY, sigma ,
trainX[i],  summationNeurons);

      outputY =  (summationNeurons[0]/ summationNeurons[1]) ;
      sumOfSquareTrainErrors += ( (double)regressTrainY[i] - outputY ) *
                        ( (double)regressTrainY[i] - outputY ) ;

     if ((int)(floor(outputY+0.5)/1) !=regressTrainY[i]) trainError++ ;
          sumOfSquareTrainErrors = sumOfSquareTrainErrors/ dataSize;
  }//end for testsize


  trainError = trainError/(double)trainSize ;
  testError = testError/(double)testSize ;

    Memo1->Lines->Add(AnsiString(sigma)+ "   \t" + AnsiString(trainError) + "   \t" +
                AnsiString(testError) + "   \t" + AnsiString(sumOfSquareTrainErrors) +
                "   \t" + AnsiString(sumOfSquareTestErrors) );

   sigmasErrors[current][0] = sigma ;
   sigmasErrors[current][1] = trainError ;
   sigmasErrors[current][2] = testError ;
   sigmasErrors[current][3] = sumOfSquareTrainErrors ;
   sigmasErrors[current][4] = sumOfSquareTestErrors ;

    current++;

   sigmastart -= sigmastep ;
 }//end while

       sizeError=current;

// FREE ALL IN REVERSE ORDER

  if (summationNeurons != NULL) free (summationNeurons);

  if (testX != NULL)  {free(testX[0]); free(testX);} // by allocate2DArray
  if (trainX != NULL) {free(trainX[0]); free(trainX);} // by allocate2DArray

  if (regressAllY != NULL) free (regressAllY);
  if (regressTrainY != NULL) free (regressTrainY);
  if (regressTestY != NULL) free (regressTestY);

   if (allDataRegression != NULL) {free(allDataRegression[0]); free(allDataRegression);} // by
allocate2DArray
```

```
 ShowMessage("OK");
}

//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
Form1->Button3->Visible=true;
Form1->PageControl1->Visible=true;
Form1->Button1->Visible=true;
Form1->Button5->Visible=false;
Form1->Edit6->Visible=true;
Form1->Label8->Visible=true;
Form1->CheckBox2->Checked = false ;
Form1->Button6->Visible = true;
Form1->DBGrid2->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
Form1->Button3->Visible=true;
Form1->PageControl1->Visible=true;
Form1->Button5->Visible=true;
Form1->Button1->Visible=false;
Form1->Edit6->Visible=true;
Form1->Label8->Visible=true;
Form1->CheckBox1->Checked = false ;
Form1->Button6->Visible = true;
Form1->DBGrid1->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
Form1->CheckBox1->Checked = false;
Form1->CheckBox2->Checked = false;
Form1->RadioButton1->Checked = false;
Form1->RadioButton2->Checked = false;
Form1->RadioButton3->Checked = false;
Form1->ComboTables->Clear();
Form1->Memo1->Clear();
Form1->Memo2->Clear();
Chart1->SeriesList->Series[0]->Clear();
Chart1->SeriesList->Series[1]->Clear();
Chart2->SeriesList->Series[0]->Clear();
Chart2->SeriesList->Series[1]->Clear();
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
Label4->Visible=True;
Form1->ComboTables->Visible=true;
Form1->ADOConnection1->Connected=false;

      if (Form1->CheckBox1->Checked)
       {
            Form1->ADOConnection1-
>ConnectionString="Provider=OraOLEDB.Oracle.1;Password=aa;\
            Persist Security Info=True;User ID=Classification;Data Source=localhost";
            Form1->ADOConnection1->Connected=true;
            ADOConnection1->Open() ;
            Form1->ADOConnection1->GetTableNames(Form1->ComboTables->Items, false);
       }

       if (Form1->CheckBox2->Checked)
       {
            Form1->ADOConnection1-
>ConnectionString="Provider=OraOLEDB.Oracle.1;Password=aa;\
            Persist Security Info=True;User ID=Regression;Data Source=localhost";
            Form1->ADOConnection1->Connected=true;
            ADOConnection1->Open() ;
            Form1->ADOConnection1->GetTableNames(Form1->ComboTables->Items, false);
       }
}
//---------------------------------------------------------------------------
```

6 ΒΙΒΛΙΟΓΡΑΦΙΑ

Witten I.H., and Frank E., “Data Mining: Practical Machine Learning Tools and Techniques”, Second Edition, Morgan Kaufmann Publishers, pp. 162-169, 2005.

E. Parzen, “On estimation of a probability density function and mode,” Ann. Math. Statist., vol. 33, pp. 1065–1076, 1962.

Bishop, C. M., “Pattern Recognition and Machine Learning”, Springer 2006.

Specht D., “A General Regression Neural Network”, IEEE Transactions on Neural Networks, vol 2, no 6, pp. 568-576, 1991.

Masters T., “Advanced Algorithms for Neural Networks”, John Wiley & Sons, 1995.