ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

Μελέτη κατηγοριοποίησης δεδομένων με Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) και υλοποίηση εφαρμογής

Πτυχιακή Εργασία

Κωνσταντίνου Γιώργος (1310)

Επιβλέπων: Τσιμπίρης Αλκιβιάδης, Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ, ΜΑΙΟΣ 2012

Σκοπός της Πτυχιακής εργασίας

Σκοπός της πτυχιακής εργασίας είναι η μελέτη του αλγορίθμου των Μηχανών Διανυσμάτων Υποστήριξης (SVM) καθώς και η εφαρμογή του σε γνωστές και άγνωστες ομάδες δεδομένων.

Θα αναπτυχθεί μια εφαρμογή σε matlab και με κατάλληλο GUI θα επικοινωνεί με βάσεις δεδομένων μέσω ODBC για λήψη, αποθήκευση και διαχείριση των δεδομένων. Στην εφαρμογή αυτή θα δοκιμαστεί ο αλγόριθμος SVM και θα διερευνηθούν όλες του οι παράμετροι χρησιμοποιώντας κυρίως το Spider matlab toolbox.

Τέλος θα γίνει σύγκριση των αποτελεσμάτων και με άλλες διαθέσιμες μηχανές όπως LibSVM κά.

ΠΕΡΙΛΗΨΗ

Η ταξινόμηση δεδομένων σε κατηγορίες καθώς και η αναζήτηση σχέσεων ομοιοτήτων - διαφορών μεταξύ δεδομένων, αποτελεί αντικείμενο έρευνας σε όλους σχεδόν τους επιστημονικούς τομείς. Οι Μηχανές Διανυσμάτων Υποστήριξης ΜΔΥ (Support Vector Machines, SVM) είναι μία οικογένεια αλγορίθμων επιβλεπόμενης μάθησης που αναπτύχθηκαν από τον Vladimir Vapnik και χρησιμοποιούνται ευρύτατα σε προβλήματα κατάταξης. Ο αλγόριθμος SVM έχει το πλεονέκτημα να χειρίζεται πολύ καλά μεγάλο πλήθος χαρακτηριστικών και παρουσιάζει υψηλή απόδοση κατά την κατηγοριοποίηση αντικειμένων(αντικείμενο ορίζουμε μία γραμμή πίνακα (ΔΙΑΝΥΣΜΑ) που έχει ένα συγκεκριμένο πλήθος χαρακτηριστικών, χαρακτηριστικά ενός αντικειμένου είναι π.χ. πλάτος, ύψος και το βάρος ενός τραπεζιού) μεταξύ δύο (2) κατηγοριών. Ο αλγόριθμος SVM είναι ικανός στο να κατασκευάζει μοντέλα αρκετά πολύπλοκα για να επιλύει δύσκολα προβλήματα του πραγματικού κόσμου. Τα SVM μοντέλα που δημιουργούνται κατόπιν εκπαιδεύσεως του 80% περίπου των δεδομένων έχουν μια απλή λειτουργική μορφή και είναι λογικευμένα σε θεωρητικές αναλύσεις. Οι ΜΔΥ περιλαμβάνουν, ως ειδικές περιπτώσεις, ένα μεγάλο εύρος από νευρωνικά δίκτυα, ακτινικών συναρτήσεων (radial basis functions) και πολυωνυμικούς ταξινομητές.

Σε προβλήματα κατηγοριοποίησης οι ΜΔΥ μπορούν να χειριστούν στόχους με συνεχόμενο εύρος. Το SVM μοντέλο μαθαίνει την γραμμική αναδρομική συνάρτηση από τα εισαχθέντα δεδομένα προς εκπαίδευση και ακολούθως γίνεται η αυτόματη κατηγοριοποίηση των νέων «άγνωστων» δεδομένων. Ο SVM αλγόριθμος είναι ένα σύστημα που αναγνωρίζει κατηγορίες από πρότυπα και λέγεται ταξινομητής (classifier), ο ταξινομητής πρώτα εκπαιδεύεται (train) από το 70% - 80% των δεδομένων και στην συνέχεια ταξινομεί αυτόματα τα εναπομείναντα δεδομένα (test). Ο διαχωρισμός των θετικών παραδειγμάτων (κατηγορία 1) και των αρνητικών παραδειγμάτων (κατηγορία -1) επιτυγχάνεται μέσω του καταλληλότερου διαχωριστικού υπερεπιπέδου που επιλέγεται από τον αλγόριθμο SVM καθώς υπάρχουν πάρα πολλά διαχωριστικά υπερεπίπεδα μη κατάλληλα στο εκάστοτε πείραμα που πραγματοποιείται.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ	ΤΙΤΛΟΣ ΚΕΦΑΛΑΙΟΥ/ΥΠΟΚΕΦΑΛΑΙΟΥ	ΣΕΛΙΔΑ
1	**Εισαγωγή**	**7**
2	**Μηχανική Μάθηση(Machine Learning)**	**9**
	2.1. Εισαγωγή στην Μηχανική Μάθηση	9
	2.2. Μηχανική Μάθηση και Τεχνητή Νοημοσύνη	10
	2.3. Εξόρυξη γνώσης (Data Mining)	10
	Κατηγοριοποίηση (Classification)	12
	Συσταδοποίηση (Clustering)	12
	Εκτίμηση και πρόβλεψη (Estimation & Prediction)	12
	Παλινδρόμηση (Regression)	13
3	**Κατηγοριοποίηση (Classification)**	**14**
	3.1. Γραμμική Ταξινόμηση	14
	3.2. Μη γραμμική κατηγοριοποίηση	16
4	**Αναγνώριση Προτύπων - Νευρωνικά Δίκτυα (Neural Networks)**	**19**
5	**Support Vector Machines (Μηχανές Διανυσμάτων Υποστήριξης)**	**23**
	5.1 Στόχος και επεξήγηση του SVM αλγόριθμου	23
	5.2 $1^η$ $2^η$ και $3^η$ διάσταση	27
	5.3 Περιγραφή Προτύπων – Χαρακτηριστικών	28
	5.4 Εσωτερικό γινόμενο διανυσμάτων	28
	5.5 Πίνακας Εσωτερικού Γινόμενου (Kernel Inner Product Matrix)	29
	5.6 Ταξινόμηση	30
	5.7 Πυρήνες Kernel	30
	5.8 VC Διάσταση	31
	5.9 Υπερεπίπεδο	32
6	**Βάσεις Δεδομένων**	**38**
	6.1 SQL (Structured Query Language)	39
	6.2 Τι είναι τα ερωτήματα (views) SQL;	39
	6.3 ODBC (Open DataBase Connectivity) πρότυπο	40
	6.4 Στόχος του προτύπου ODBC	40
7	**Matlab**	**42**
	7.1 Ιστορική Αναδρομή	42
	7.2 Χρήση του Matlab	42
	7.3 Χρήση του Matlab στα δικά μας πειράματα	43
	7.4 Matlab και Βάσεις Δεδομένων	43
8	**ΜΕΡΟΣ Α. Υλοποίηση εφαρμογής και επεξήγηση κώδικα χρησιμοποιώντας την εργαλειοθήκη SPIDER Matlab Toolbox**	**45**
	8.1 Αρχικά Βήματα	45
	8.2 Αρχικοποίηση μεταβλητών κατά την εκκίνηση της εφαρμογής	47

	8.3 UiPanel με τίτλο DataBase Panel	48
	8.4 Listbox με static text Table Columns(Callback)	56
	8.5 Pushbutton Load Table Data(Callback)	57
	8.6 Listbox με static text Data From Columns(Callback)	60
	8.7 UiPanel με τίτλο SPIDER Control Panel	60
	ΜΕΡΟΣ Β. Υλοποίηση εφαρμογής και επεξήγηση κώδικα χρησιμοποιώντας την LIBSVM βιβλιοθήκη	**73**
	8.8.1 Το κουμπί «LIBSVM TRAIN» (Callback)	74
	8.8.2 Το κουμπί «LIBSVM TEST» (Callback)	78
9	**Πειράματα Κατηγοριοποίησης**	**80**
	9.1 Σετ δεδομένων(Iris, Wine, Glass, Breast Canser Wisconsis (Original) και Παρέλαση Τελειοφοίτων)	80
	9.2 Πίνακας αποτελεσμάτων κάνοντας χρήση το πρόγραμμα WEKA	86
	9.3 Πίνακας αποτελεσμάτων κάνοντας χρήση τον SPIDER SVM	87
	9.4 Πίνακας αποτελεσμάτων κάνοντας χρήση την βιβλιοθήκη LIBSVM	88
10	**Προβλήματα τα οποία προέκυψαν κατά την επίλυση της εφαρμογής**	**90**
11	**Συμπεράσματα**	**91**
12	**ΠΑΡΑΡΤΗΜΑ**	**93**
	Α) Διαδικασία εγκατάστασης των απαραίτητων εκτελέσιμων εφαρμογών (Εκπαιδευτικό Πρόγραμμα WEKA)	93
	Β) Διαδικασία Ρύθμισης και Εκτέλεσης της Εφαρμογής SPIDER SVM	94
	Γ) Σετ Δεδομένων	94
	Δ) Δημιουργία ODBC Data Source στο matlab για σύνδεση με SQL Server	96
	Ε) Τρόποι Εισαγωγής Δεδομένων για υλοποίηση πειραμάτων	97
	Ζ) Εισαγωγή υπάρχουσας ΒΔ στο Enterprise Manager του Microsoft SQL Server 2000	98
	Η) Εισαγωγή πίνακα στο SQL Server από ένα ήδη υπάρχον αρχείο excel (*.xls) ή από υπάρχον αρχείο κειμένου (*.txt)	99
	Θ) Τρόπος λειτουργίας της εφαρμογής SVM_Classification.fig	100
	Ι) Προσαρμογή και εγκατάσταση της βιβλιοθήκης LIBSVM- 3.11	102
	Κ) Όλα όσα πρέπει να γνωρίζουμε κάνοντας χρήση το εκπαιδευτικό πρόγραμμα WEKA	102
13	**Βιβλιογραφία**	104

Την παρούσα πτυχιακή την αφιερώνω στους γονείς μου…

Κεφάλαιο 1

Εισαγωγή

Οι μηχανές διανυσμάτων υποστήριξης (SVM) μπορούν να χειρίζονται δεδομένα ταξινομώντας τα αυτόματα σε κατηγορίες αποτελούμενες από αντικείμενα-πρότυπα(το κάθε πρότυπο αποτελείτε από ένα συγκεκριμένο αριθμό χαρακτηριστικών, απαραίτητη προϋπόθεση είναι να έχουν όλα τα πρότυπα σε κάθε πείραμα που πραγματοποιούμε ίδιο πλήθος χαρακτηριστικών αλλιώς θα προκύψει σφάλμα στο πείραμα), μέσω μιας διαδικασίας η οποία αποτελείτε από δύο (2) στάδια. Αρχικά γίνεται η εκπαίδευση ανά δύο (2) κατηγορίες κάθε φορά (γίνεται εκπαίδευση δεδομένων τόσες φορές όσες είναι και οι δυνατοί συνδυασμοί που προκύπτουν μεταξύ των διαθέσιμων κατηγοριών που κατέχουμε) όπου ο αλγόριθμος μας "μαθαίνει" από τα δεδομένα(training set {80%}) που του εισήχθησαν ούτως ώστε να μπορεί να γίνει εφικτή στο μετέπειτα στάδιο η αυτόματη κατηγοριοποίηση του εναπομείναντα μέρους των δεδομένων(testing set {20%}). Να "μάθει" ο αλγόριθμος τι χαρακτηρίζει την κάθε κατηγορία, σε τι διαφέρει από τις άλλες κατηγορίες και πότε κάποιο αντικείμενο-πρότυπο θα ανήκει σε αυτή **[10]**.

Ένα πρόβλημα που παρουσιάζεται είναι ότι ο αλγόριθμος κατατάσσει ένα δεδομένο σε μία(1) από δύο(2) διαφορετικές κατηγορίες, είτε σε αυτή που συμβολίζουμε με -1 ή σε εκείνη που συμβολίζουμε με 1, πράγμα που σημαίνει ότι σε κάθε ταξινόμηση που θα πραγματοποιείτε θα πρέπει να γίνεται πάντα μόνο ανά δύο(2) κατηγορίες. Αυτός ο τρόπος ταξινόμησης λέγεται <u>δυαδική ταξινόμηση</u> και με αυτό τον τρόπο δουλεύουμε εμείς στα πειράματα μας. Η διαδικασία αυτή γίνεται ανά δύο κατηγορίες κάθε φορά και θα γίνει τόσες φορές όσες είναι και οι μεταξύ των κατηγοριών δυνατοί συνδυασμοί (π.χ. αν έχουμε 4 κατηγορίες έχουμε τους συνδυασμούς: K1-K2, K1-K3, K1-K4, K2-K3, K2-K4, K3-K4). Ο διαχωρισμός των δεδομένων σε κατηγορίες γίνεται με βάση κάποιες ξεχωριστές ιδιαιτερότητες(χαρακτηριστικά) που έχει η κάθε κατηγορία από τις άλλες(π.χ. έχουμε συνολικά εκατό (100) παραδείγματα όπου αυτά ανήκουν σε τέσσερις (4)

διαφορετικές κατηγορίες και η κάθε κατηγορία αποτελείται από πέντε (5) κοινά χαρακτηριστικά, χρησιμοποιούμε διανύσματα επειδή το κάθε παράδειγμα αναπαριστάται από ένα 5-διάστατο διάνυσμα, δηλαδή αντιμετωπίζουμε κάθε παράδειγμα ως μια ακολουθία πέντε αριθμών. Άρα τα δεδομένα μας μαζί με όλες τις πληροφορίες που μεταφέρουν δεν είναι για μας παρά ένας πίνακας 100x5(ένα διάνυσμα ανά γραμμή).

Η επιτυχία της SVM τεχνικής αποδίδεται κυρίως στις δυνατές θεωρητικές βάσεις βασισμένες στην Vapnik-Chervonenkis (VC) θεωρία. Υπάρχουν μερικοί περιορισμοί στον κυρίως (standard) SVM κώδικα όπου μειώνουν την πρακτική χρησιμότητα του αλγορίθμου:

- Μη-γραμμικά μοντέλα μπορούν να μεγαλώσουν κατά πολύ σε μέγεθος, κάνοντας την διαδικασία ολοκλήρωσης αργή και μη-πρακτική.
- Ο αλγόριθμος εκτελεί μια συγκεκριμένη λειτουργία, μόνο ανά δύο κατηγορίες μπορεί να χειριστεί. Στον SVM αλγόριθμο ο ορισμός των παραμέτρων και η προετοιμασία των δεδομένων συνήθως απαιτούνται.

Το oracle’s svm είναι πρόγραμμα με ποιότητα στην εφαρμογή του όπου είναι διαθέσιμο μέσα σε πρόγραμμα βάσεων δεδομένων(oracle database). Ως μια εναλλακτική λύση κάποιος μπορεί να λάβει υπόψη του αυτόνομες(stand-alone) ακαδημαϊκές εφαρμογές SVM (π.χ. SPIDER Matlab Toolbox, LibSVM, WEKA, TinySVM, SVMLight, SVMTorch, HeroSVM) **[10]**.

Ο αλγόριθμος SVM kernel linear είναι το αντικείμενο με το οποίο θα ασχοληθούμε περαιτέρω στα επόμενα κεφάλαια και θα γίνει ανάλυση πειραμάτων αυτόματης κατηγοριοποίησης δεδομένων.

Κεφάλαιο 2

Μηχανική μάθηση (Machine Learning)

2.1 Εισαγωγή στη μηχανική μάθηση

Η μηχανική μάθηση μας βοηθά στο να βρίσκουμε λύσεις σε πολλά προβλήματα σε θέματα αναγνώρισης φωνής, σε θέματα ρομποτικής, σε θέματα αναγνώρισης προσώπων δηλαδή αναγνώρισης προτύπων και άλλα πολλά. Είναι μια νέα τεχνολογία εξόρυξης γνώσης από δεδομένα, μία τεχνολογία όπου πολλοί άνθρωποι αρχίζουν να παίρνουν στα σοβαρά. Ας δούμε το παράδειγμα αναγνώρισης προσώπων, καθημερινά αναγνωρίζουμε συγγενικά πρόσωπα και φίλους κοιτώντας τα πρόσωπά τους ή κοιτώντας φωτογραφίες τους ασχέτως με τον φωτισμό τον οποίο έχει η εκάστοτε φωτογραφία και το στιλ των μαλλιών των ατόμων αυτών. Απλά εμείς τους αναγνωρίζουμε ασυνείδητα χωρίς να μπορούμε να εξηγήσουμε το πώς το κάνουμε. Επειδή δεν μπορούμε να εξηγήσουμε την τεχνογνωσία μας δεν μπορούμε να γράψουμε την τεχνογνωσία αυτή σε πρόγραμμα με κάποια γλώσσα προγραμματισμού **[15]**.

Την ίδια στιγμή γνωρίζουμε ότι σε μία εικόνα που έχει ένα πρόσωπο δεν είναι απλά μία τυχαία συλλογή από εικονοστοιχεία (pixels), το πρόσωπο έχει μία δομή. Είναι συμμετρικό. Υπάρχει η μύτη, τα μάτια, το στόμα τοποθετημένα σε συγκεκριμένα σημεία επάνω στο πρόσωπο. Κάθε πρόσωπο οποιουδήποτε ατόμου είναι ένα πρότυπο αποτελούμενο από ένα ξεχωριστό – ιδιαίτερο συνδυασμό από χαρακτηριστικά. Το πρόγραμμα όπου αναγνωρίζει πρόσωπα από φωτογραφίες πρέπει πρώτα να του εισαχθούν εικόνες π.χ. του ατόμου Α δημιουργώντας έτσι ένα πρότυπο για το άτομο Α και μετά γίνεται η εισαγωγή εικόνας του ατόμου Α όπου το πρόγραμμα κάνει σύγκριση μεταξύ του πρότυπου και της εικόνας που εισάχθηκε και η έξοδος που μας βγάζει το πρόγραμμα είναι η ταυτοποίηση του ατόμου Α. Αυτό είναι ένα παράδειγμα αναγνώρισης προτύπων.

Με τα πλεονεκτήματα στην τεχνολογία των Η/Υ πλέον έχουμε την δυνατότητα για αποθήκευση και επεξεργασία τεράστιων όγκων δεδομένων. Ας πάρουμε για παράδειγμα μια αλυσίδα υπεραγορών που αποτελείται από εκατοντάδες υποκαταστήματα σε μια χώρα πουλώντας χιλιάδες προϊόντα σε εκατομμύρια

πελάτες. Οι Η/Υ των υπεραγορών καταγράφουν λεπτομέρειες από την κάθε συναλλαγή η οποία γίνεται καταγράφοντας: ημερομηνία, τον κωδικό του πελάτη, τα προϊόντα που αγόρασε, πόσα χρήματα πλήρωσε ο πελάτης κτλ. Ο όγκος πληροφοριών μιας ημέρας μπορεί να ανέρθει σε gigabytes. Αυτές οι πληροφορίες που αποθηκεύονται μπορούν να γίνουν χρήσιμες μόνο κατόπιν ανάλυσης και μετατραπούν σε πληροφορία όπου θα μας είναι χρήσιμη, για παράδειγμα το να κάνουμε προβλέψεις. Υποθέτοντας ότι μελλοντικά σε 2-3 μήνες δεν θα διαφέρουμε και πολύ από το παρελθόν όταν τα δεδομένα ανά πελάτη συλλέχθηκαν και άρα οι μελλοντικές προβλέψεις μπορούν να θεωρηθούν ως σωστές.

2.2 Μηχανική μάθηση και τεχνητή νοημοσύνη

Η μηχανική μάθηση δεν έχει να κάνει μόνο με βάσεις δεδομένων που έχουν αποθηκευμένες πληροφορίες, είναι και ένα κομμάτι της τεχνητής νοημοσύνης. Για να είναι "έξυπνο" ένα σύστημα, πρέπει σε ένα περιβάλλον που συνεχώς αλλάζει να έχει την ικανότητα να μαθαίνει. Εάν το σύστημα μαθαίνει και αναπροσαρμόζεται σε αυτές τις αλλαγές τότε ο προγραμματιστής του συστήματος δεν χρειάζεται πλέον να προβεί σε αλλαγές και να παρέχει λύσεις για όλες τις πιθανές καταστάσεις οι οποίες μπορεί να προκύψουν, όταν ο προγραμματιστής μεταφέρει τέτοιες γνώσεις (δηλ. την νοημοσύνη του) στο σύστημα και το σύστημα αναπροσαρμόζεται από μόνο του τότε αυτό το σύστημα έχει τεχνητή νοημοσύνη.

Η αναλογία είναι ότι από το τεράστιο όγκο της γης και διάφορων άλλων υλικών που εξορίσετε από ένα ορυχείο όπου κατόπιν επεξεργασίας καταλήγει ένα μικρό μέρος πολύτιμων υλικών(χρυσός, διαμάντια), παρομοίως στην εξόρυξη γνώσης ένας μεγάλος όγκος δεδομένων επεξεργάζεται για να κατασκευαστεί ένα απλό μοντέλο μεγάλης σημασίας έχοντας μεγάλης ακρίβειας ικανότητα πρόβλεψης.

2.3 Εξόρυξη Γνώσης (data mining)

Οι 10 καλύτεροι αναγνωρισμένοι αλγόριθμοι από το IEEE International Conference (ICDM) στην εξόρυξη γνώσης είναι: C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naive Bayes και CART **[9]**.

Η εξόρυξη γνώσης (data mining) είναι μια τεχνολογία ανάλυσης μεγάλης σημασίας καθώς μεγάλοι όγκοι δεδομένων συλλέγονται σε βάσεις δεδομένων διάφορων κυβερνήσεων και διάφορων βιομηχανιών. Καθώς στις βάσεις δεδομένων ο κύριος σκοπός είναι η ανάκτηση δεδομένων, το data mining θέτει νέες προκλήσεις(ανάκτηση και επεξεργασία δεδομένων). Οι επιτυχημένες εφαρμογές της τεχνολογίας εξόρυξης γνώσεων συνήθως απαιτούν σύνθετες μεθοδολογίες και την συμμετοχή αναλυτών εξόρυξης γνώσεων. Η εξόρυξη γνώσης τυπικά είναι μία συνεχής υπολογιστική δραστηριότητα όπου απαιτεί μεγάλη υπολογιστική ισχύ.

Οι ΜΔΥ είναι ένα τυπικό παράδειγμα αλγόριθμου εξόρυξης γνώσης όπου μπορεί να παράγει καλής ποιότητας αποτελέσματα όταν χρησιμοποιείται από κάποιον ειδικό κατέχοντας επαρκής πόρους συστήματος. Ο SVM αλγόριθμος δημιουργήθηκε ως ένας αλγόριθμος επιλογής για προκλήσεις μοντελοποίησης δεδομένων πολλών-διαστάσεων εκεί όπου μερικές τεχνικές υπολειτουργούν. Συγκρίνοντας με διάφορες μεθόδους, η απόδοση του SVM αλγόριθμου φάνηκε να είναι ανώτερη από decision trees, νευρωνικά δίκτυα (neural networks) και Bayesian προσεγγίσεις.

Ο αλγόριθμος SVM ανήκει στον κλάδο εξόρυξης γνώσης όπου με την σειρά του ο κλάδος εξόρυξης γνώσης βασίζεται σε χρησιμοποιημένες και δοκιμασμένες τεχνικές από κλάδους όπως της μηχανικής μάθησης(machine learning), αναγνώρισης προτύπων(pattern recognition), στατιστικής και άλλων και αφορούν: κατηγοριοποίηση δεδομένων, συσταδοποίηση δεδομένων, εκτίμηση και πρόβλεψη μελλοντικών τάσεων, παλινδρόμηση κ.ά.

Η εξόρυξη γνώσης είναι μια τεχνολογία ανάλυσης στην οποία περιλαμβάνετε :

- Κατηγοριοποίηση (Classification)
- Συσταδοποίηση (Clustering)
- Εκτίμηση και πρόβλεψη (Estimation & Prediction)
- Παλινδρόμηση (Regression)

Classification (κατηγοριοποίηση)

Ονομάζεται η διαδικασία ταξινόμησης δεδομένων σε κλάσεις οι οποίες είναι γνωστές από πριν, είναι δηλαδή προκαθορισμένες όπου είναι και το αντικείμενο που μας απασχολεί στην παρούσα πτυχιακή (δυαδική ταξινόμηση). Αφού έχουμε ένα σετ δεδομένων που όλα έχουν το ίδιο πλήθος χαρακτηριστικών γίνεται πρώτα μία διαδικασία εκπαίδευσης(training) όπου ορίζει ο χρήστης μερικές ετικέτες(labels 1 ή -1) στα δεδομένα και γίνεται η εκπαίδευση του αλγόριθμου. Όταν έρθει εις πέρας η εκπαίδευση, δημιουργείται ένα «μοντέλο πρόβλεψης» όπου με βάση αυτό θα γίνει μετά η δοκιμή των νέων «άγνωστων» δεδομένων τα οποία είναι τα εναπομείναντα δεδομένα και γίνεται από τον αλγόριθμο η αυτόματη κατηγοριοποίηση.

Στόχος: Ο ορισμός ενός πολύ καλού γενικευμένου μοντέλου όπου θα μπορεί να ταξινομεί σωστά τα νέα «άγνωστα» δεδομένα.

Clustering (συσταδοποίηση)

Τα πρότυπα συγκεντρώνονται σε ίδια ομάδα(συγκέντρωση(cluster)) όταν οι αντίστοιχες τιμές των χαρακτηριστικών τους έχουν κοντινότερες τιμές σε σχέση με τα υπόλοιπα πρότυπα από το εκάστοτε σύνολο δεδομένων. Με βάση κάποιο μαθηματικό κριτήριο ομοιότητας δύο(2) προτύπων (π.χ. City Block, Ευκλείδια απόσταση, Chebychev, Mahalanobis) βρίσκεται η απόσταση μεταξύ δύο(2) προτύπων. Το clustering δεν βασίζεται σε προκαθορισμένες κλάσεις.

Estimation & prediction (εκτίμηση και πρόβλεψη)

Είναι τεχνικές εκτίμησης και πρόβλεψης τιμών ή τάσεων οι οποίες μπορεί να προκύψουν στο μέλλον με στόχο την κατασκευή μοντέλου όπου θα είναι ικανό να προβλέπει την τιμή μιας συγκεκριμένης μεταβλητής από μερικές γνωστές τιμές άλλων μεταβλητών.

Regression (παλινδρόμηση)

Με την τεχνική αυτή δημιουργούνται μοντέλα πρόβλεψης αριθμητικών τιμών π.χ. πρόβλεψη ισοτιμίας νομισμάτων καθώς και για πρόβλεψη τιμής μετοχής.

Κεφάλαιο 3

Κατηγοριοποίηση(classification)

Σε κάθε περίπτωση που πραγματοποιούμε κατηγοριοποίηση δεν είναι πάντα εφικτό να φτιαχτεί ένα αλάνθαστο σύστημα, άρα εμείς περιμένουμε σχεδόν πάντα ένα ποσοστό από λανθασμένες κατηγοριοποιήσεις. Αυτό μπορεί να συμβεί:

- Είτε γιατί τα δεδομένα εισόδου εκπαίδευσης που χρησιμοποιήθηκαν για την κατασκευή του μοντέλου πρόβλεψης είναι ανεπαρκή ή ακατάλληλα.
- Η γιατί υπάρχει επικάλυψη μεταξύ των διαφορετικών κατηγοριών.

Συμπέρασμα: Στα πειράματα που πραγματοποιούνται πρέπει να περιμένουμε από τον ταξινομητή (Classifier) να πετυχαίνει "πολύ καλά" ή αποδεκτά αποτελέσματα.

3.1 Γραμμική ταξινόμηση

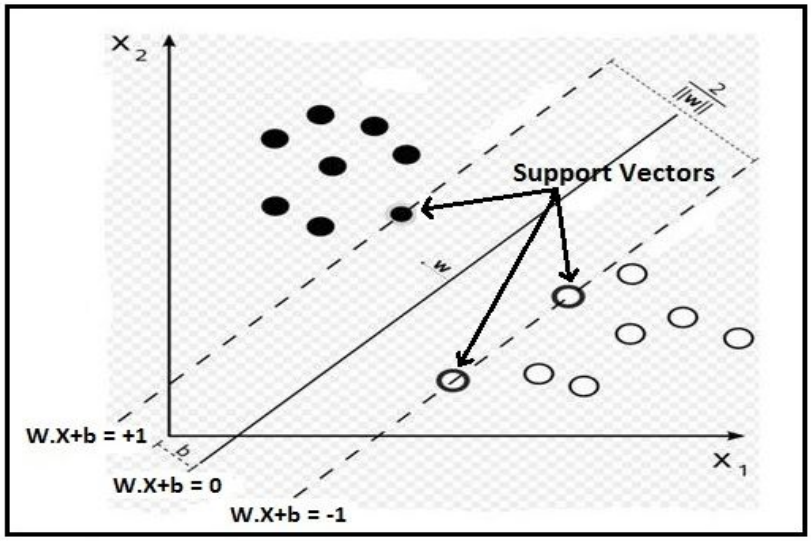
Η ταξινόμηση είναι το αντικείμενο που μας απασχολεί στην πτυχιακή αυτή χρησιμοποιώντας τον αλγόριθμο γραμμικής ταξινόμησης SVM, αφού χρησιμοποιούμε αλγόριθμο γραμμικής ταξινόμησης σημαίνει ότι το σετ δεδομένων που θα εισάγεται πρέπει να έχουν όλα τα δεδομένα ίδιο μέγεθος(ίσο πλήθος χαρακτηριστικών) και αφού ορίσουμε εμείς δικούς μας κανόνες που θα ταξινομηθούν τα δεδομένα θα πρέπει να έχουν ένα (1) τουλάχιστον χαρακτηριστικό το οποίο να είναι διαχωρίσιμο από το εκάστοτε ολόκληρο σετ δεδομένων.

Υπάρχουν δύο (2) τρόποι εκτέλεσης γραμμικής κατηγοριοποίησης:

1. Hard Margin SVM όπου δεν επιτρέπεται καμία λάθος ταξινόμηση κατά την δημιουργία του μοντέλου πρόβλεψης(Εκπαίδευση (training))
2. Soft Margin SVM όπου επιτρέπονται μερικές λάθος ταξινομήσεις κατά την δημιουργία του μοντέλου πρόβλεψης(Εκπαίδευση (training)) και το μοντέλο πρόβλεψης θεωρείται ως σωστό

Στην παρούσα πτυχιακή εργασία χρησιμοποιούμε τον Hard Margin SVM όπου δεν επιτρέπεται καμία λάθος ταξινόμηση κατά την δημιουργία του μοντέλου πρόβλεψης, το μοντέλο πρόβλεψης είναι το σημαντικότερο σημείο γιατί με βάση

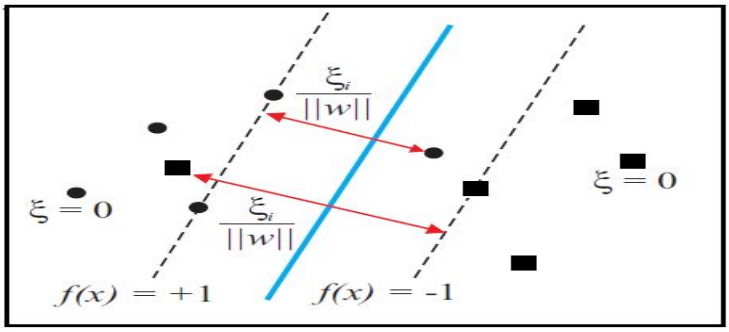
αυτό γίνεται η μετέπειτα αυτόματη ταξινόμηση των νέων «άγνωστων» παραδειγμάτων στο στάδιο της δοκιμής (test).



Εικόνα 3.1.1: Linear kernel SVM (Hard Margin SVM)

Ανάλογα με τον πυρήνα kernel που εκάστοτε χρησιμοποιείται δημιουργείτε και το ανάλογο υπερεπίπεδο, με ένα ίδιο σετ δεδομένων διαφορετικό υπερεπίπεδο θα κατασκευαστεί με τον linear kernel svm και διαφορετικό υπερεπίπεδο όταν χρησιμοποιηθεί π.χ. ο πολυωνυμικός kernel.

Εμείς χρησιμοποιούμε τον Hard Margin SVM όπου δεν επιτρέπεται καμία λάθος ταξινόμηση κατά την διάρκεια την εκπαιδεύσεως, στον Soft Margin SVM μπορεί να οριστεί κάνοντας χρήση τις μεταβλητές **ξ** (slack variables), ένα ποσοστό λανθασμένων ταξινομήσεων κατά την εκπαίδευση και να θεωρηθεί η ταξινόμηση ως σωστή. Ένα παράδειγμα με soft margin svm φαίνεται στην παρακάτω εικόνα 3.1.2.



Εικόνα 3.1.2: Linear kernel SVM (Soft Margin SVM)

Οι μεταβλητές ξ_i λέγονται μεταβλητές χαλαρότητας (slack variables) που σκοπό έχουν να μετρούν την απόσταση των προτύπων από μία θέση ιδανική που πραγματοποιείται γραμμικός διαχωρισμός.

Όταν ισχύει ο πιο κάτω περιορισμός:

$$0 < \xi_i \leq 1$$

[3.1.1]

Τότε τα πρότυπα βρίσκονται μέσα στο χώρο της διαχωριστικής περιοχής και στην σωστή πλευρά του υπερεπιπέδου. Όταν $\xi_i > 1$ ή $\xi_i < 0$ το παράδειγμα δεν βρίσκεται στην σωστή πλευρά του υπερεπιπέδου.

Τα διανύσματα που ικανοποιούν την παρακάτω εξίσωση ακριβώς είναι τα διανύσματα υποστήριξης:

$$d_i(w^T.x_i + b) \geq 1 - \xi_i, \qquad i = 1,2,\dots,N$$

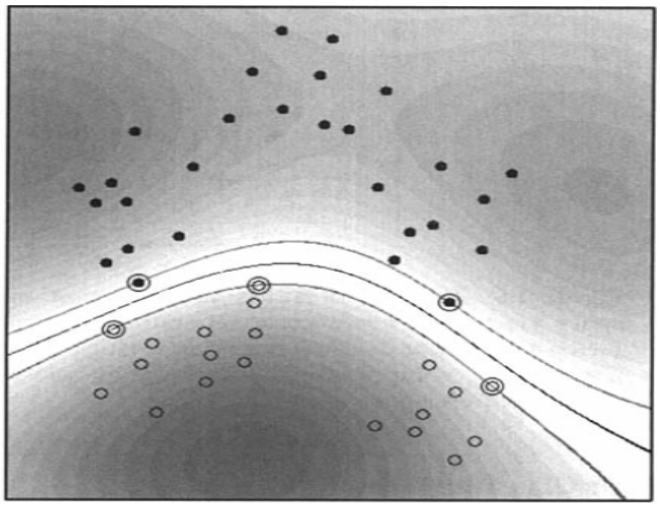
[3.1.2]

Επίσης μπορεί να υπάρχουν διανύσματα υποστήριξης που ικανοποιούν την συνθήκη $\xi_i = 0$. Υπάρχει η πιθανότητα ένα πρότυπο με $\xi_i > 0$ να αποκλειστεί από το σετ εκπαίδευσης, εάν αυτό γίνει τότε η επιφάνεια απόφασης θα αλλάξει. Συμπεραίνουμε ότι τα διανύσματα υποστήριξης σε soft και hard margin ορίζονται ακριβώς με τον ίδιο τρόπο.

Στον soft margin αλγόριθμο στη ταξινόμηση δεν είναι απαραίτητο από όλα τα παραδείγματα προς εκπαίδευση να υπάρχει κάποιο χαρακτηριστικό το οποίο να είναι διαχωρίσιμο ώστε να μπορεί να πραγματοποιηθεί η ταξινόμηση, ο αλγόριθμος μπορεί να δουλέψει ακόμα και αν μερικά δεδομένα της κατηγορίας Α εμπίπτουν στην κατηγορία Β και αντίστροφα. Στην περίπτωση γραμμικά διαχωρίσιμων κατηγοριών, το υπερεπίπεδο που κατασκευάζεται (**w**,b) είναι η μέγιστη <u>απόσταση(**ρ**)</u> από τα κοντινότερα παραδείγματα μεταξύ των δύο(2) κατηγοριών σε αυτό τον πολυδιάστατο χώρο των χαρακτηριστικών. Για να γίνει πιο κατανοητό αυτό γίνεται περισσότερη επεξήγηση στο Κεφάλαιο 5.

3.2 Μη γραμμική ταξινόμηση

Δημιουργία ενός μη γραμμικού ταξινομητή με τη χρήση διανυσμάτων υποστήριξης ενός δισδιάστατου διανύσματος χαρακτηριστικών φαίνεται στην παρακάτω εικόνα 3.2.1:



Εικόνα 3.2.1: Παράδειγμα ενός ταξινομητή διανυσμάτων υποστήριξης **[14]** χρησιμοποιώντας συνάρτηση kernel radial basis. Οι διπλοί κύκλοι είναι τα διανύσματα υποστήριξης. Η μεσαία γραμμή είναι η επιφάνεια αποφάσεων.
Όπως φαίνεται από την παραπάνω εικόνα παρατηρούμε ότι ο διαχωρισμός των δύο κατηγοριών δεν γίνεται με μια ευθεία γραμμή όπως γίνεται στον γραμμικό ταξινομητή.

Συνάρτηση απεικόνισης φ

X → Είναι ο χώρος εισόδου (input space)

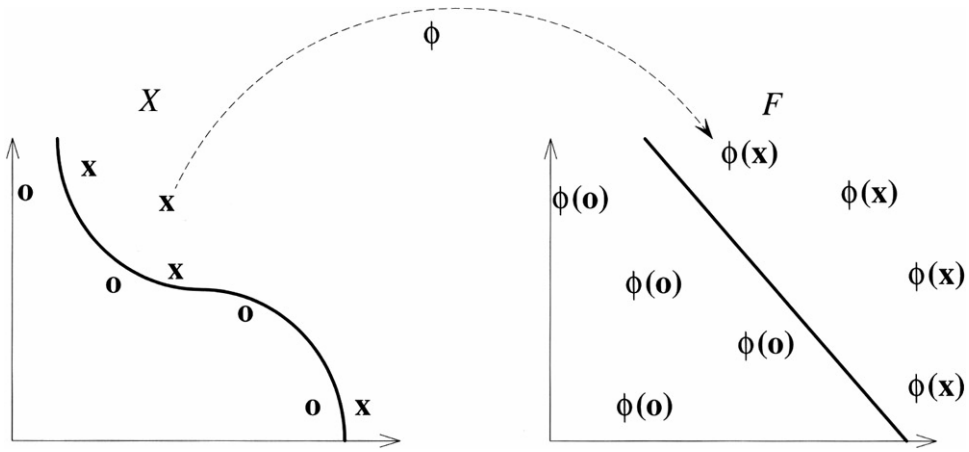
F → Χώρος χαρακτηριστικών (feature space)

Η τοποθέτηση των παραδειγμάτων εισόδου (σετ εκπαίδευσης x) από τον χώρο εισόδου X σε ένα νέο χώρο F(feature space)γίνεται μέσω της συνάρτησης απεικόνισης φ.

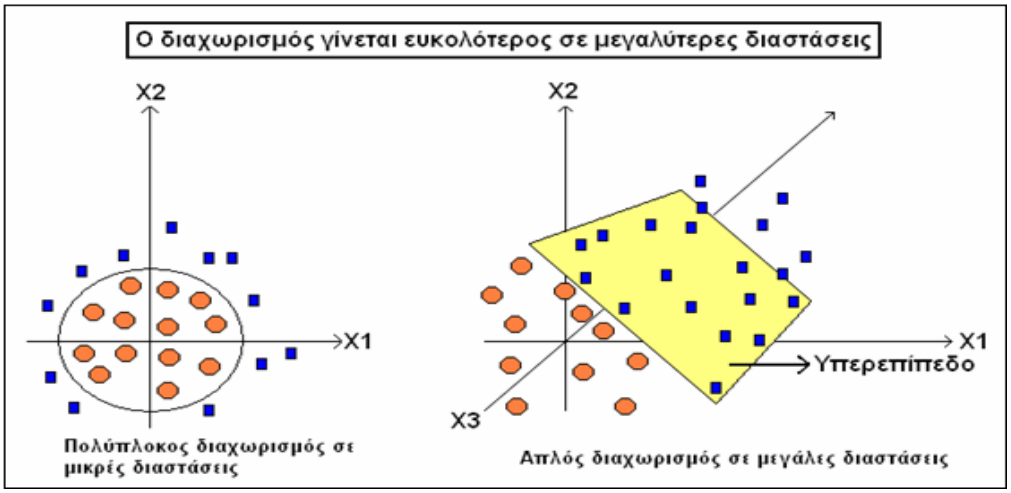
$\phi: x \rightarrow \phi(x)$

Η συνάρτηση απεικόνισης φ από τον χώρο εισόδου X μεταφέρει τα δεδομένα στον χώρο των χαρακτηριστικών(feature space) F μετασχηματίζοντας τις συνιστώσες τους(Εικόνα 3.2.2).

$\phi: X \rightarrow F$



Εικόνα 3.2.2: Η τοποθέτηση των χαρακτηριστικών σε ένα νέο χώρο μπορεί να απλοποιήσει το πρόβλημα της κατηγοριοποίησης.



Εικόνα 3.2.3: Υπερεπίπεδο(επίπεδο σε ένα χώρο με περισσότερες των 3 διαστάσεων) το οποίο διαχωρίζει τα θετικά από τα αρνητικά παραδείγματα εκπαίδευσης. Μη γραμμικά διαχωρίσιμα σε 2D, γραμμικά διαχωρίσιμα σε 3D **[13]**.

Τώρα πλέον τα μετασχηματισμένα με τη συνάρτηση απεικόνισης ϕ δεδομένα είναι γραμμικός διαχωρίσιμα, γίνεται μετασχηματισμός δεδομένων σε ένα άλλο χώρο. Συνηθέστεροι πυρήνες για αυτή την περίπτωση είναι ο Gaussian radial basis συνάρτηση kernel και ο πολυωνυμικός kernel. Πλέον το σύστημα έχει ευκολότερο έργο, την ανίχνευση γραμμικών προτύπων.

Σημαντική ιδιότητα των συναρτήσεων kernel είναι ότι υπολογίζουν τα εσωτερικά γινόμενα χωρίς να χρειάζεται να γνωρίζουμε την συνάρτηση απεικόνισης ϕ.

Κεφάλαιο 4

Αναγνώριση Προτύπων - Νευρωνικά Δίκτυα (Neural Networks)

Ο άνθρωπος ξεχωρίζει από τα διάφορα άλλα όντα(ζώα) στην νοημοσύνη χάρις στον εγκέφαλο που διαθέτει ο οποίος αποτελείτε από διάφορα τμήματα, ο εγκέφαλος έχει πάρα πολλούς νευρώνες όπου αναπτύσσονται - δημιουργούνται κατά πολύ στην παιδική ηλικία και λιγότερο στο υπόλοιπο της ζωής του. Ο εγκέφαλος ενός ανθρώπου είναι ένα σύνθετο σε πολυπλοκότητα, μη γραμμικό και με παράλληλη δομή σύστημα επεξεργασίας πληροφοριών. Αρκετοί επιστήμονες προσπαθούσαν να αντιγράψουν τον ανθρώπινο εγκέφαλο δημιουργώντας τεχνητά νευρωνικά δίκτυα (ΤΝΔ), πράγμα το οποίο είναι αδύνατο να πραγματοποιηθεί. Συγκρίνοντας την δομή και την πολυπλοκότητα του ανθρώπινου εγκεφάλου οι νευρώνες που χρησιμοποιούμε για κατασκευή νευρωνικού δικτύου είναι "πρωτόγονοι" **[4]**. Ο ανθρώπινος εγκέφαλος έχει περίπου 100 δισεκατομμύρια νευρώνες και ο κάθε νευρώνας περίπου έχει 1000 συνάψεις πράγμα που μας δίνει να καταλάβουμε το πόσο δύσκολο είναι να αντιγραφεί ο εγκέφαλος σε μορφή ΤΝΔ. Γίνεται ταυτόχρονος παραλληλισμός της επεξεργασίας από τους νευρώνες και ακολούθως γίνεται η κατανομή της πληροφορίας στα διάφορα μέρη του εγκεφάλου.

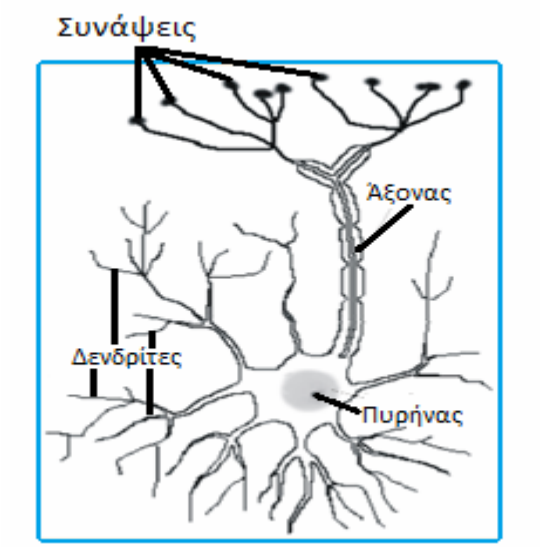
<u>Νοημοσύνη</u>

- Η αναγνώριση εικόνων
- Η αναγνώριση φωνής
- Η λήψη αποφάσεων
- Η κατάστρωση στρατηγικής
- Η λογική, η ανάπτυξη επιχειρημάτων
- Η μάθηση και η αυτοπροσαρμογή
- Η μνήμη
- Η αυτόματη πλοήγηση στο χώρο

Η αναγνώριση προτύπων από μηχανές μπορούν να αντικαταστήσουν ποικίλες επίπονες ανθρώπινες εργασίες παρέχοντας ταχύτατα αποτελέσματα (π.χ. αναγνωρίζοντας πρόσωπα από φωτογραφίες οι οποίες είναι αποθηκευμένες σε ένα Η/Υ για ταυτοποίηση προσώπων). Ένα άλλο παράδειγμα αναγνώρισης προτύπων από μηχανές είναι αναγνωρίζοντας φωνές ατόμων από τηλεφωνικές συνδιαλέξεις για ταυτοποίηση φωνής. Επίσης σε μία πανεπιστημιακή βιβλιοθήκη μπορούν να ψηφιοποιηθούν όλα τα βιβλία της σε ηλεκτρονική μορφή με κωδικοποίηση κατά ASCII από μία μηχανή που σαρώνει(scanning) τα βιβλία αυτά υπο μορφή εικονοστοιχείων (pixels) αποθηκεύοντας τα βιβλία στον σκληρό δίσκο ενός Η/Υ(FTP Server) χρησιμοποιώντας και ένα πρόγραμμα OCR(Optical Character Recognition).

Τα βιβλία με αυτόν τον τρόπο μετατρέπονται σε ηλεκτρονική μορφή και παραμένουν αναλλοίωτα από την πάροδο του χρόνου από τυχόν φθορές που μπορούν να συμβούν στα βιβλία(σκισίματα, μουντζούρες). Τέλος όταν τα βιβλία ψηφιοποιηθούν σε μορφή .pdf θα μπορεί να γίνεται ταχύτατη αναζήτηση σε αυτά μέσω του εύχρηστου κουμπιού αναζήτησης(search) ενός pdf προγράμματος πετυχαίνοντας έτσι γρήγορη πρόσβαση σε συγκεκριμένα σημεία ενός βιβλίου που ενδιαφέρουν τον φοιτητή με επαγγελματισμό και χωρίς να χρονοτριβή.

Νευρώνας



Οι δενδρίτες είναι είσοδοι, δέχονται ηλεκτρικά σήματα από άλλους νευρώνες. Οι συνάψεις είναι σημεία ένωσης μεταξύ διακλαδώσεων του άξονα ενός νευρώνα και των δενδριτών από άλλους νευρώνες. Ο άξονας είναι έξοδος και έχει μήκος από 1mm έως > 1m, στέλνει ηλεκτρικούς παλμούς σταθερού πλάτους αλλά μεταβλητής συχνότητας.

Σε δύο βασικές κατηγορίες χωρίζονται τα νευρωνικά δίκτυα:

1. Σε δίκτυα εμπρόσθιας διάδοσης(feedforward networks) και
2. Ανατροφοδοτούμενα δίκτυα(recurrent networks)

Στην αναγνώριση προτύπων γίνεται κατηγοριοποίηση προτύπων στις σωστές κατηγορίες αναπαριστώντας τα πρότυπα σε μορφή διανυσμάτων.

Ανάλογα με τον τρόπο όπου τα δεδομένα θα εκπαιδευτούν υπάρχουν δύο(2) βασικές κατηγορίες εκπαίδευσης **[3]**:

1. Αλγόριθμοι εκπαίδευσης με επίβλεψη (supervised learning) και
2. Αλγόριθμοι εκπαίδευσης χωρίς επίβλεψη (unsupervised learning)

Από τα διάφορα δίκτυα εμπρόσθιας διάδοσης στην αναγνώριση προτύπων το πολυεπίπεδο δίκτυο perceptron(multilayer perceptron (MLP)) και το δίκτυο ακτινικών συναρτήσεων βάσης (Radial Basis Function network (RBF)) είναι τα πιο διαδεδομένα.

<u>Αποστάσεις</u>

Η σχέση d($\vec{x},\vec{y}$) θεωρείται ως απόσταση εάν για οποιοδήποτε άνυσμα x,y,z ισχύει:

d($\vec{x},\vec{y}$)d($\vec{y},\vec{x}$)

d($\vec{x},\vec{y}$)$\leq$ d($\vec{x},\vec{z}$)d($\vec{z},\vec{y}$)

d($\vec{x},\vec{y}$)≥ 0

Αν d($\vec{x},\vec{y}$)=0 άρα $\vec{x}=\vec{y}$

Απόσταση μεταξύ δύο(2) σημείων σε διανυσματικούς χώρους

Μία από τις μεθόδους μέτρησης της απόστασης μεταξύ δύο σημείων στο καρτεσιανό σύστημα συντεταγμένων είναι η ευκλείδεια απόσταση.

Ευκλείδεια απόσταση

Ειδική περίπτωση της Minkowski για s=2

$$d_E(\vec{x},\vec{y}) = \left[\sum_{v}^{N} [(x_v - y_v)^2]^{\frac{1}{2}} = [(x-y)^T.(x-y)]^{\frac{1}{2}}\right]$$

Κεφάλαιο 5

Support Vector Machines (Μηχανές Διανυσμάτων Υποστήριξης)

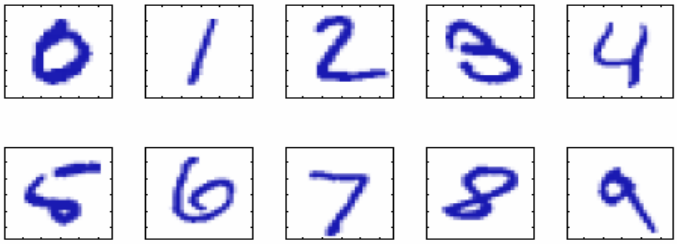
5.1 Στόχος και επεξήγηση του SVM αλγόριθμου

Στόχος του γραμμικού ταξινομητή SVM όπου βασίζεται σε διανύσματα υποστήριξης είναι να "κατασκευάσει" ένα βέλτιστο διαχωριστικό υπερεπίπεδο(feature space). Οι ΜΔΥ είναι συστήματα τα οποία "μαθαίνουν" χρησιμοποιώντας μια αλγοριθμική προσέγγιση που βασίζεται στη θεωρία βελτιστοποίησης.

Ο αλγόριθμος SVM είναι επιβλεπόμενης μάθησης αναπτυσσόμενος την τελευταία δεκαετία από τον Vlatimir Vapnik και άλλους (Vlatimir Vapnik , Statistical Learning Theory 1998) **[12]**. Ο αλγόριθμος είναι επικεντρωμένος στο γενικό πρόβλημα της μάθησης όπου ξεχωρίζονται τα θετικά και αρνητικά μέλη(πρότυπα) από ένα σύνολο δεδομένων με ν-διάστατα διανύσματα. Εάν έχουμε μία συλλογή από ψηφιοποιημένες εικόνες με αριθμητικά ψηφία γραμμένα με το χέρι, ο SVM αλγόριθμος μπορεί να μας πει για παράδειγμα ότι στην εικόνα που εισάγαμε αναγράφεται ο αριθμός π.χ. 8.

Κάθε αριθμός στην παρακάτω εικόνα αντιστοιχεί ως μία εικόνα μεγέθους 28 X 28 εικονοστοιχεία(pixels) όπου αυτή η εικόνα μπορεί να αναπαρασταθεί με ένα διάνυσμα X περιλαμβάνοντας 784 πραγματικούς αριθμούς **[11]**. Ο στόχος είναι να φτιαχτεί μια τέτοια μηχανή όπου θα παίρνει ως είσοδο ένα τέτοιο διάνυσμα X και θα βγάζει ως έξοδο την ταυτότητα του γραμμένου με το χέρι αριθμού. Υπάρχουν αμέτρητοι τρόποι όπου μπορούν να γραφτούν αριθμοί αφού ο κάθε άνθρωπος έχει και ένα δικό του τρόπο γραφής(μοναδικό). Η μηχανή θα μπορούσε να αναγνωρίζει αριθμούς γραμμένους με το χέρι εάν όλοι χρησιμοποιούσαν τυποποιημένους τρόπους γραφής των αριθμών αυτών, στην πράξη αυτό είναι αδύνατο να γίνει καθώς οι κανόνες αυτοί δεν θα τηρούταν από όλους και τα αποτελέσματα θα είχαν

μικρό ποσοστό επιτυχίας. Για καλύτερα αποτελέσματα θα μπορούσαμε να εισάγουμε ως δεδομένα εκπαίδευσης ένα τεράστιο σετ από Ν ψηφία γραμμένα με το χέρι από διαφορετικούς ανθρώπους {Χ1,…..,Χn} δημιουργώντας έτσι ένα καλό μοντέλο κατηγοριοποίησης. Ένα παρόμοιο παράδειγμα είναι η ταυτοποίηση γραφικού χαρακτήρα ή και ταυτοποίηση υπογραφής μέσω Η/Υ, απαραίτητη προϋπόθεση να υπάρχει δείγμα υπογραφής του εκάστοτε ατόμου που θα γίνεται η ταυτοποίηση.



Εικόνα 5.1.1: Παραδείγματα αριθμών γραμμένων με το χέρι από αμερικάνικους ταχυδρομικούς κώδικες.

Γενικά ο αλγόριθμος SVM λειτουργεί ως εξής: χαρτογραφεί τα δεδομένα εκπαίδευσης σε ένα "πιθανό" πολύ-διάστατο χώρο χαρακτηριστικών και προσπαθεί να εντοπίσει σε αυτό το χώρο ένα επίπεδο όπου διαχωρίζει τα θετικά από τα αρνητικά παραδείγματα. Αφού ο αλγόριθμος βρει το καταλληλότερο επίπεδο διαχωρισμού, τότε μπορεί να προβλέψει την κατηγοριοποίηση ενός χωρίς ετικέτα παραδείγματος τοποθετώντας το στον χώρο των χαρακτηριστικών και ρωτώντας σε ποια πλευρά του διαχωριστικού επιπέδου ανήκει τοποθετεί αναλόγως το νέο αυτό παράδειγμα.

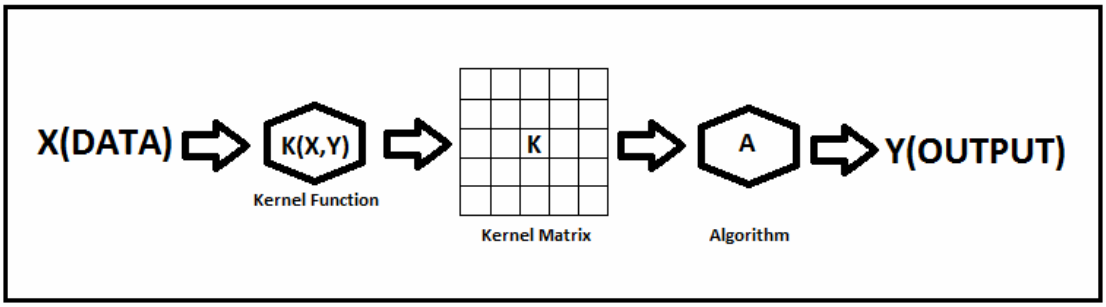
Ένα μεγάλο μέρος της δύναμης του SVM αλγόριθμου αντλείτε από το κριτήριο επιλογής ενός(1) διαχωριστικού επιπέδου όταν υπάρχουν πολλά υποψήφια διαχωριστικά επίπεδα: Ο SVM αλγόριθμος επιλέγει το διαχωριστικό επίπεδο που έχει το μεγαλύτερο περιθώριο (απόσταση **ρ**) από όλα τα σημεία του σετ εκπαίδευσης. Δηλαδή τα κοντινότερα σημεία μεταξύ των διαφορετικών κατηγοριών και με βάση αυτών ορίζεται το διαχωριστικό επίπεδο. Η θεωρία στατιστικής μάθησης (statistical learning theory) προτείνει ότι εάν έχουμε καλά διαχωρίσιμα δεδομένα σε κατηγορίες χωρίς να εμπίπτουν τα δεδομένα της

κατηγορίας 1 στην κατηγορία -1, η επιλογή του υπερεπίπεδου με το μέγιστο περιθώριο θα οδηγήσει στην μέγιστη γενικοποίηση (πολύ καλό μοντέλο διαχωρισμού) όταν θα γίνεται η δοκιμή (test) νέων άγνωστων δεδομένων για αυτόματη κατηγοριοποίηση.

Στην εκάστοτε κατηγορία που θέτουμε ως κατηγορία 1 θέτουμε κάποια διανύσματα σε αυτήν, για το κάθε διάνυσμα ξεχωριστά π.χ. (1.85, 70) γίνεται μια μαθηματική πράξη μεταξύ των αριθμών που αντιπροσωπεύουν το διάνυσμα και λόγω του ότι ορίσαμε να ανήκει στην κατηγορία 1 το αποτέλεσμα που προκύπτει είναι ένας «θετικός» πραγματικός αριθμός (λόγω του αριθμού της κατηγορίας που είναι ο αριθμός +1) και τοποθετείτε στην θετική πλευρά του διαχωριστικού επίπεδου σαν μία τελεία (.) ή σαν (o,x,*,+). Στην παρούσα πτυχιακή ο SPIDER **[7]** αλγόριθμος εμφανίζει τα σύμβολα o(μπλε κύκλους) και x(κόκκινα x) στις δύο(2) ξεχωριστές κατηγορίες. Αντίστοιχα στην κατηγορία -1 λόγω του πρόσημου της κατηγορίας όπου είναι το – (μείον), για ένα οποιοδήποτε διάνυσμα που θα θέσουμε σε αυτήν π.χ. (1.65, 67) ο αριθμός που θα προκύψει κατόπιν μαθηματικής πράξης θα είναι αρνητικός και άρα θα τοποθετηθεί στην αρνητική πλευρά του διαχωριστικού επιπέδου.

Ας υποθέσουμε ότι έχουμε ένα σετ εκπαίδευσης όπου έχει m παραδείγματα όπου το κάθε παράδειγμα είναι ένα διάνυσμα με n αριθμούς. Αυτά τα διανύσματα μπορούμε να τα σκεφτούμε ως σημεία(dot product) σε ένα n-διάστατο χώρο. Dot products είναι μεγάλοι μπλε κύκλοι (o) και κόκκινα (x) στο γράφημα (plot) της εφαρμογής μας. Επειδή ο αλγόριθμος βρίσκει ένα διαχωριστικό υπερεπίπεδο στον χώρο των χαρακτηριστικών(feature space) κάνοντας αποκλειστικά χρήση διανυσμάτων στον χώρο εισαγωγής(train) αναπαριστώντας τα διανύσματα ως τελείες ((.) dot products) στον χώρο των χαρακτηριστικών, μια ΜΔΥ μπορεί να εντοπίσει ένα υπερεπίπεδο απλά ορίζοντας μία συνάρτηση ονομαζόμενη ως kernel συνάρτηση όπου παίζει τον ρόλο της τελείας ((.) dot product) στον χώρο των χαρακτηριστικών(feature space). Αυτή η τεχνική αποφεύγει τον υπολογιστικό φόρτο που αναπαρίστανται τα διανύσματα στον χώρο των χαρακτηριστικών(feature space).

Για κάποια σετ δεδομένων, ο SVM αλγόριθμος είναι πιθανόν να μην μπορέσει να βρει διαχωριστικό υπερεπίπεδο στον χώρο των χαρακτηριστικών, είτε επειδή η συνάρτηση kernel είναι ακατάλληλη για τα δεδομένα εκπαίδευσης ή γιατί στα δεδομένα περιέχονται παραδείγματα με λανθασμένες ετικέτες(labels).



Εικόνα 5.1.2: Στην πιο πάνω εικόνα απεικονίζεται σχηματικά ολόκληρη η διαδικασία που πραγματοποιείται από τον χρήστη όταν καλείται να εκτελέσει πείραμα γραμμικής ταξινόμησης χρησιμοποιώντας τον αλγόριθμο SVM kernel linear.

Ανάλογα με την φύση του προβλήματος έχουν αναπτυχθεί πολλές τεχνικές μηχανικής μάθησης όπου κατατάσσονται σε ένα από τα παρακάτω είδη:

A) μάθηση με επίβλεψη(supervised learning)

B) μάθηση χωρίς επίβλεψη(unsupervised learning)

Μερικοί αλγόριθμοι που χρησιμοποιούν ταξινόμηση είναι svm kernel linear, knn ή αλλιώς ibk(WEKA) και c4.5 ή αλλιώς J48(WEKA). Οι ΜΔΥ προσπαθούν να βρουν ένα υπερεπίπεδο (hyperplane) που διαχωρίζει τα αρνητικά από τα θετικά παραδείγματα των δύο(2) κατηγοριών. Αυτό το υπερεπίπεδο επιλέγεται από τον αλγόριθμο, ώστε να απέχει όσο το δυνατό περισσότερο από τα κοντινότερα(μεταξύ δύο(2) κατηγοριών) θετικά και αρνητικά παραδείγματα(Maximum Margin Hyperplane, MMH).

Όταν η μάθηση γίνεται με επίβλεψη το σύστημα καλείται να μάθει μία συνάρτηση(συνάρτηση στόχος) από ένα σύνολο με δεδομένα εκπαίδευσης και με βάση αυτά δημιουργείτε ένα μοντέλο, το μοντέλο πρόβλεψης(prediction model). Η

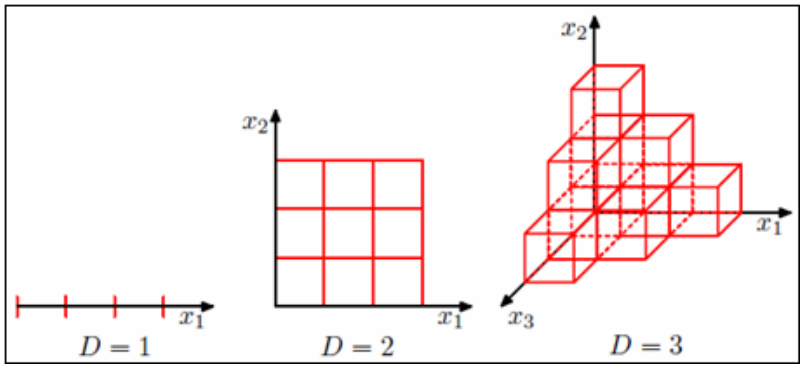
συνάρτηση στόχος (target function) αποτελεί έκφραση του μοντέλου πρόβλεψης που περιγράφει τα δεδομένα που εκπαιδεύτηκαν.

Στη μάθηση χωρίς επίβλεψη από μόνο του το σύστημα πρέπει να ανακαλύψει συσχετίσεις ή ομάδες από όλα τα δεδομένα.

5.2 1η, 2η και 3η διάσταση

Ο ανθρώπινος νους μπορεί να κατανοήσει μέσω των οπτικών σημάτων που λαμβάνει από τα μάτια μόνο τον χώρο των τριών διαστάσεων και όχι περισσοτέρων διαστάσεων, στην πτυχιακή έγινε πείραμα με δεδομένα Iris στο οποίο τα χαρακτηριστικά των παραδειγμάτων έχουν πλήθος τέσσερα(4), δηλαδή έχουμε να κάνουμε με 4 – διάστατα ανύσματα. Δεν μπορούμε σε αυτήν την περίπτωση του Iris πειράματος να δούμε διαγραμματικά(δηλαδή να δούμε σε plot) τα διανύσματα υποστήριξης που προκύπτουν κατά την εκπαίδευση λόγω του ότι το πλήθος των διανυσμάτων που μπορεί να αναπαραστήσει ο αλγόριθμος στο γράφημα(plot) της εφαρμογής μας είναι μόνο για 2 – διάστατα ανύσματα. Αυτό όμως δεν μας απασχολεί ιδιαίτερα καθώς εμείς επιθυμούμε να δουλεύει σωστά ο αλγόριθμος παράγοντας σωστά και αξιόπιστα αποτελέσματα στην στήλη “Labels” της εφαρμογής μας όταν έρθει εις πέρας η δοκιμή (test) των νέων άγνωστων δεδομένων, υποδεικνύοντας μας με τους αριθμούς 1 ή -1 σε ποια κατηγορία ανήκει το αντίστοιχο άνυσμα της στήλης “Labeled Characteristics” της εφαρμογής μας.

Στην εφαρμογή μας γίνονται δύο (2) πειράματα, στο πρώτο πείραμα έχουμε δεδομένα Iris(4 – διάστατα διανύσματα) και στο δεύτερο πείραμα έχουμε δεδομένα τελειόφοιτων μαθητών για συμμετοχή σε παρέλαση(2 – διάστατα διανύσματα). Στο δεύτερο πείραμα χρησιμοποιείται το γράφημα (plot) για να μπορεί να δει ο χρήστης και οπτικά την εργασία του αλγόριθμου SVM βλέποντας τα διανύσματα υποστήριξης που προκύπτουν κατά την εκπαίδευση κατανοώντας έτσι πλήρως το πείραμα και διαγραμματικά.



Εικόνα 5.2.1: Φαίνονται η 1η διάσταση, η 2η διάσταση και η 3η διάσταση. Για λόγους σαφήνειας μόνο ένα υποσύνολο κύβων εμφανίζονται για την 3η διάσταση.

5.3 Περιγραφή Προτύπων – Χαρακτηριστικών

Ένα παράδειγμα διανύσματος είναι [2, 0.70, 0.55, 1.52] όπου αντιπροσωπεύει τα χαρακτηριστικά ενός τραπεζιού. Το 2 είναι το χρώμα του τραπεζιού(π.χ. μαύρο) το οποίο επειδή είναι ποιοτικό χαρακτηριστικό το συμβολίζουμε με αριθμό, το 0.70 είναι το ύψος, το 0.55 είναι το πλάτος και το 1.52 είναι το μήκος του.

Διανυσματική περιγραφή: Το κάθε πρότυπο για το κάθε πείραμα που θα πραγματοποιήσουμε έχει ένα πεπερασμένο πλήθος χαρακτηριστικών (χρώμα, πλάτος, ύψος, βάρος).

Κάθε παράδειγμα του ίδιου συνόλου δεδομένων θεωρείτε διαφορετικό ακόμα και αν όλες οι αντίστοιχες τιμές των χαρακτηριστικών τους είναι ίδιες[3]. Από ένα σύνολο δεδομένων που αποτελείται από M πλήθος παραδειγμάτων Πμ, μ=1,2,3,4,…,M και N το πλήθος των χαρακτηριστικών, τότε το παράδειγμα Πμ περιγράφεται από ένα διάνυσμα – πίνακα στήλης Χμ (π.χ. Χ1=[4,8,2]) σε ένα χώρο N διαστάσεων. Εάν το διάνυσμα Χμ έχει τα χαρακτηριστικά π.χ. [4, 8, 2] τότε μπορεί να περιγραφεί σε ένα χώρο 3 διαστάσεων, άρα όσο είναι το πλήθος των χαρακτηριστικών ενός προτύπου τόσο θα είναι και το πλήθος των διαστάσεων που θα περιγράφει το πρότυπο.

5.4 Εσωτερικό γινόμενο διανυσμάτων

Όλου του τύπου δεδομένα που επεξεργάζεται το matlab τα χειρίζεται ως πίνακες, έτσι και τα διανύσματα $\vec{x}$ και $\vec{z}$ περιγράφονται από πίνακες στο matlab,

x=[$x_1, x_2, \dots, x_N$] και z=[$z_1, z_2, \dots, z_N$]

[5.4.1]

το εσωτερικό γινόμενο διανυσμάτων περιγράφεται:

$$\sum_{V=1}^{N} x_V . z_V = x^T . z$$

[5.4.2]

Πιο αναλυτικά το εσωτερικό γινόμενο γράφεται ως:

$$\vec{x}.\vec{z} \quad x_1 . z_1 + x_2 . z_2 + \cdots + x_N . z_N \qquad [5.4.3]$$

Ουσιαστικά το εσωτερικό γινόμενο δύο διανυσμάτων θα είναι πάντα ένας αριθμός και όχι ένα νέο διάνυσμα όπως στην πρόσθεση και την αφαίρεση διανυσμάτων.

5.5 Πίνακας Εσωτερικού Γινόμενου(Kernel Inner Product Matrix)

Ο πίνακας εσωτερικού γινόμενου προκύπτει κατόπιν μαθηματικής πράξης εσωτερικού γινομένου διανυσμάτων **[13]**. Εάν έχουμε για παράδειγμα ένα σετ δεδομένων με συνολικό αριθμό 80 με 2 – διάστατα ανύσματα (από 1° παράδειγμα έως το 40° παράδειγμα είναι η κατηγορία 1 και τα υπόλοιπα της κατηγορίας -1) τότε τα πρώτα 32 παραδείγματα όπου αντιπροσωπεύουν το 80% της κατηγορίας 1 και από το 41° έως το 72° αντιπροσωπεύουν επίσης το 80% της κατηγορίας -1, αυτά τα 64 παραδείγματα θα εκπαιδευτούν. Αυτά τα 64 παραδείγματα για εμάς δεν είναι τίποτε άλλο από ένα πίνακα με 64 γραμμές και δύο (2) στήλες (δεδομένα εκπαίδευσης). Μετά γνωρίζοντας όλες τις πληροφορίες των δεδομένων εκπαίδευσης, δημιουργείτε ο πίνακας εσωτερικού γινομένου με όλους τους δυνατούς συνδυασμούς των παραδειγμάτων μεταξύ τους, δηλ. 64 X 64 πίνακας.

Πίνακας Εσωτερικού Γινομένου (Kernel Inner Product Matrix)

K	*1*	*2*	*3*	*...*	*M*
1	$K(X_1, X_1)$	$K(X_1, X_2)$	$K(X_1, X_3)$	...	$K(X_1, X_m)$
2	$K(X_2, X_1)$	$K(X_2, X_2)$	$K(X_2, X_3)$	...	$K(X_2, X_m)$

3	$K(X_3, X_1)$	$K(X_3, X_2)$	$K(X_3, X_3)$	…	$K(X_3, X_m)$
4	$K(X_4, X_1)$	$K(X_4, X_2)$	$K(X_4, X_3)$	…	$K(X_4, X_m)$
.	…	…	…	…	…
.	…	…	…	…	…
m	$K(X_m, X_1)$	$K(X_m, X_2)$	$K(X_m, X_3)$	…	$K(X_m, X_m)$

Είναι όλοι οι δυνατοί συνδυασμοί των διανυσμάτων που εκάστοτε ο χρήστης επιλέγει για την εκπαίδευση του αλγόριθμου SVM. Υπολογίζονται τα εσωτερικά γινόμενα τους και δημιουργείται ένας πίνακας που εμείς ονομάσαμε ως «Kernel Inner Product Matrix». Αφού έρθουν εις πέρας όλοι οι υπολογισμοί, τα αποτελέσματα αυτά χρησιμοποιούνται από τον αλγόριθμο για να δημιουργήσει το μοντέλο διαχωρισμού. Από τον κάθε συνδυασμό προκύπτει ένας αριθμός(εσωτερικό γινόμενο) όπου αυτή η πληροφορία είναι αρκετή για τον αλγόριθμο για να λειτουργήσει.

5.6 Ταξινόμηση

Η ταξινόμηση ανήκει στην επιβλεπόμενη μάθηση και αφορά στη δημιουργία μοντέλων πρόβλεψης διακριτών τάξεων(κατηγοριών). Οι ΜΔΥ προσπαθούν να εντοπίσουν μία υπερ-επιφάνεια (hypersurface) που να διαχωρίζει στο χώρο των παραδειγμάτων τα θετικά από τα αρνητικά παραδείγματα. Ως καταλληλότερη υπερεπιφάνεια επιλέγεται αυτή που απέχει όσο το δυνατό περισσότερο από τα κοντινότερα θετικά και αρνητικά παραδείγματα (maximum margin hypersurface, MMH), η υπερεπιφάνεια ουσιαστικά είναι μία ευθεία γραμμή.

Στόχος του γραμμικού ταξινομητή SVM καθώς και οποιουδήποτε άλλου αλγόριθμου οι οποίοι βασίζονται σε διανύσματα υποστήριξης είναι να “φτιάξει” ένα βέλτιστο διαχωριστικό υπερεπίπεδο(hyperplane) σε ένα πολυδιάστατο χώρο χαρακτηριστικών(feature space). “Βέλτιστο” εννοούμε ότι στην περίπτωση μας που έχουμε να κάνουμε με κατηγορίες γραμμικά διαχωρίσιμες, το υπερεπίπεδο που θα κατασκευαστεί (w,b) θα είναι εκείνο με την μεγαλύτερη απόσταση(περιθώριο **ρ**) από τα πιο κοντινά παραδείγματα στον πολυδιάστατο χώρο των χαρακτηριστικών.

5.7 Πυρήνες Kernel

Οι πυρήνες kernel έχουν ένα σημαντικό πλεονέκτημα του ότι δεν χρειάζεται να γνωρίζουν την συνάρτηση απεικόνισης φ αφού μπορούν και υπολογίζουν τα εσωτερικά γινόμενα των διανυσμάτων, πληροφορία η οποία είναι αρκετή για τους πυρήνες kernel. Η έννοια των πυρήνων παίζει πάρα πολύ σημαντικό ρόλο στην κατασκευή των μηχανών διανυσμάτων υποστήριξης που αναφέρονται στην βιβλιογραφία και ως μέθοδοι πυρήνων (kernel methods) **[13]**.

Γραμμικός kernel (Euclidean Inner Product)

K(x,z) = → K(x,z) = [5.7.1]

Η συνάρτηση Kernel γράφεται ως K, όπου για κάθε διάνυσμα x,z $\in$ X όπου η συνάρτηση απεικόνισης φ μεταφέρει από τον χώρο X στον χώρο F (εσωτερικό γινόμενο). Τα πειράματα που πραγματοποιήθηκαν στην πτυχιακή έγιναν με τον γραμμικό kernel.

SVM Linear Kernel → Μεταφέρει ολόκληρο το σετ δεδομένων εκπαίδευσης σε ένα νέο χώρο χωρίς όμως να αλλάζει τη σχετική θέση των δειγμάτων μεταξύ τους.

Gaussian RBF kernel

K(x_i, x_j) = exp) [5.7.2]

Το σ^2 καθορίζεται από τον χρήστη

Πολυωνυμικός kernel

K(x_i, x_j) = $((x_i, x_j))^d$

[5.7.3] Το d καθορίζεται από τον χρήστη

5.8 VC Διάσταση

Η VC θεωρία αναπτύχθηκε από το 1960 – 1990 από τον Vlatimir Vapnik και τον Alexey Chervonenkis. Αυτή η θεωρία περιλαμβάνει την θεωρία υπολογιστικής μάθησης(computational learning theory) η οποία προσπαθεί να εξηγήσει την διαδικασία μάθησης από στατιστικής άποψης.

Η VC(Vapnik-Chervonenkis) διάσταση μεταξύ κλάσεων αυξάνει κατά πολύ την ικανότητα γενικοποίησης ενός μοντέλου ώστε να μπορεί το μοντέλο αυτό να παράγει σε μεγάλο βαθμό καλής ποιότητας αποτελέσματα. Αυτό επιτυγχάνεται δημιουργώντας το μέγιστο περιθώριο(maximum marginal hyperplane, MMH) μεταξύ των κατηγοριών.

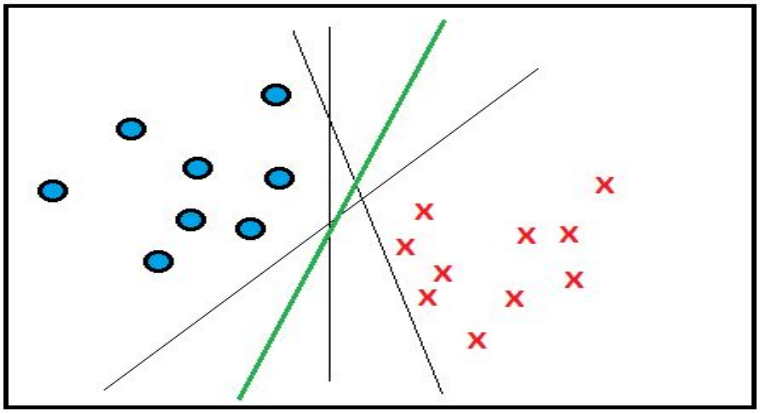
Για αλγόριθμους γραμμικής ταξινόμησης σε ένα N-διάστατο χώρο:

- Ένα υπερεπίπεδο στον χώρο διαστάσεων R^n διασκορπίζει (shatters)οποιοδήποτε σετ h=n+1 σε γραμμικά ανεξάρτητα σημεία(points)
- Κατά συνέπεια, η VC-διάσταση αλγόριθμων που χρησιμοποιούν γραμμική στρατηγική είναι h=n+1.

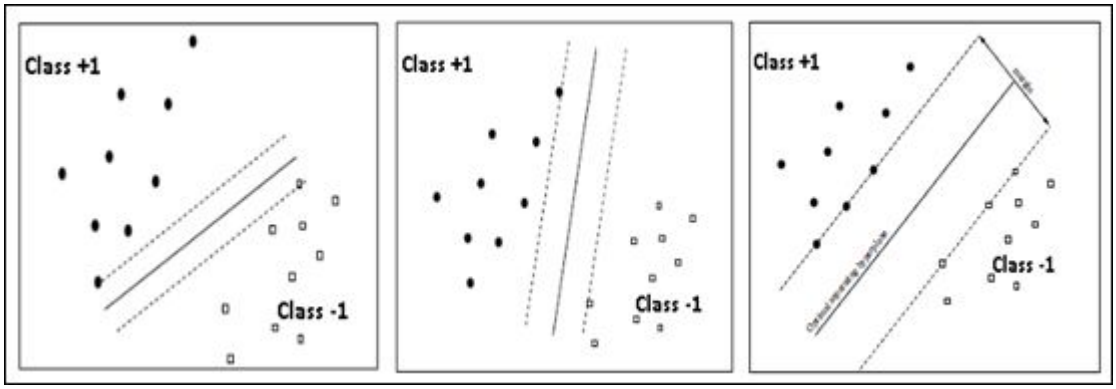
5.9 Υπερεπίπεδο

Ο στόχος του ταξινομητή είναι να δημιουργήσει ένα μοντέλο όπου να δουλεύει "σωστά" όταν θα του εισάγονται «άγνωστα»(νέα) δεδομένα, δηλαδή να γίνει ένα καλά γενικευμένο μοντέλο με το μέγιστο περιθώριο ρ **[13]**. Λαμβάνοντας υπ' όψιν την παρακάτω εικόνα 5.9.1, παρατηρούμε ότι υπάρχουν πάρα πολλές ευθείες όπου μπορούν να διαχωρίσουν τα δεδομένα, αλλά μόνο μία από αυτές μεγιστοποιεί το περιθώριο(μεγιστοποιεί την απόσταση μεταξύ των δύο(2) κλάσεων, δηλαδή βρίσκεται ακριβώς στην μέση των δύο(2) κλάσεων και απέχει όσο το δυνατό περισσότερο από τα κοντινότερα θετικά και αρνητικά παραδείγματα). Άρα λοιπόν η εύρεση του βέλτιστου υπερεπιπέδου είναι και το **"κλειδί"** για δημιουργία ενός καλά γενικευμένου μοντέλου, καθώς ελαχιστοποιεί το σφάλμα κατηγοριοποίησης άγνωστων(νέων) δεδομένων προς αυτόματη ταξινόμηση.

Με τη μέθοδο των ΜΔΥ αναζητείτε το υπερεπίπεδο με μέγιστο περιθώριο ρ (maximum marginal hyperplane MMH), στην πραγματικότητα υπάρχουν άπειρες ευθείες διαχωρισμού, μερικές ευθείες γραμμές διαχωρισμού φαίνονται στην παρακάτω εικόνα 5.9.1. Στην ουσία το υπερεπίπεδο αναπαριστάται ως μια ευθεία γραμμή διαχωρισμού μεταξύ κατηγοριών.



Εικόνα 5.9.1: Βέλτιστο(ελαχιστοποιημένο, δηλαδή ελαχιστοποιείτε το ποσοστό λανθασμένων ταξινομήσεων) διαχωριστικό υπερεπίπεδο.



Εικόνα 5.9.2: Από αριστερά προς δεξιά στην πρώτη εικόνα το περιθώριο ρ()είναι μικρότερο από το περιθώριο της μεσαίας εικόνας, το οποίο περιθώριο ρ της μεσαίας εικόνας είναι μικρότερο από το βέλτιστο διαχωριστικό υπερεπίπεδο το οποίο βλέπουμε στην τρίτη εικόνα.

<u>Μαθηματική Επεξήγηση:</u>

Ο οποιοσδήποτε αλγόριθμος ο οποίος λειτουργεί με ΜΔΥ έχει ως ρόλο την επίτευξη του εσωτερικού γινομένου πυρήνα (Inner Product Kernel) μεταξύ του διανύσματος υποστήριξης x_s και του διανύσματος x από το σετ δεδομένων εισόδου, δηλαδή των εναπομενόντων δεδομένων δοκιμής. <u>Συμπεραίνουμε</u> ότι τα διανύσματα υποστήριξης αποτελούν ένα μικρό υποσύνολο του σετ εκπαιδεύσεως.

<u>Παράδειγμα</u>

Θεωρούμε ένα σύνολο εκπαίδευσης $\{x_i, d_i\}_{i=1}^{N}$ όπου x_i είναι το τελευταίο άνυσμα του σετ δεδομένων εκπαίδευσης και d_i είναι η ετικέτα (1 ή -1) του. Επίσης θεωρούμε ότι η κατηγορία των ανυσμάτων της οποίας ισχύει

$d_i = +1$ και η κατηγορία διανυσμάτων της οποίας ισχύει $d_i = -1$ είναι γραμμικά διαχωρίσιμες.

Η εξίσωση της υπερεπιφάνειας που περιγράφει μια επιφάνεια απόφασης (decision surface), η οποία καθορίζει τον διαχωρισμό των κατηγοριών είναι:

$$w^T.x + b = 0$$

[5.9.1]

όπου $w = \{w_1, w_2, w_3, \dots, w_n\}$ το διάνυσμα βαρών, το x είναι ένα διάνυσμα εισόδου και b ένας αριθμός πόλωσης (bias).

Σύμφωνα με την σχέση [5.9.1] μπορούμε να γράψουμε:

[5.9.2]

Από τις σχέσεις [5.9.1] και [5.9.2] παρατηρούμε την κεντρική ιδέα πίσω από της μηχανές διανυσμάτων υποστήριξης κατά έναν απλό τρόπο. <u>Η υπόθεση αυτή αναιρείτε παρακάτω</u>:

Θεωρούμε ότι w_0 και b_0 οι βέλτιστες τιμές του διανύσματος βαρών και πόλωσης (bias). Άρα το βέλτιστο διαχωριστικό υπερεπίπεδο θα δίνεται από την σχέση:

$$w_0^T.x + b_0 = 0$$

[5.9.3]

Όπου η σχέση [5.9.3] είναι επαναδιατύπωση της εξίσωσης [5.9.1]. Η διακρίνουσα συνάρτηση

g(x) $w_0^T.x + b_0 = 0$ [5.9.4]

μας δίνει ένα αλγεβρικό μέτρο της απόστασης του διανύσματος x από το βέλτιστο υπερεπίπεδο.

Αλγεβρικά για να δούμε πόσο απέχει ένα διάνυσμα από την βέλτιστη διαχωριστική επιφάνεια υπολογίζεται η απόσταση r. Το r είναι θετικό εάν το διάνυσμα βρίσκεται στη θετική πλευρά της βέλτιστης διαχωριστικής επιφάνειας και το r είναι αρνητικό εάν το διάνυσμα x βρίσκεται στην αρνητική πλευρά της βέλτιστης διαχωριστικής επιφάνειας και έχουμε:

r [5.9.5]

Η απόσταση από την αρχή των αξόνων (x=0) της βέλτιστης διαχωριστικής επιφάνειας δίνεται από τον τύπο $\frac{b_0}{\|w_0\|}$. Εάν $b_0 > 0$, η αρχή των αξόνων βρίσκεται από την θετική πλευρά του βέλτιστου υπερεπιπέδου, εάν $b_0 < 0$ στην αρνητική πλευρά και τέλος εάν $b_0 = 0$ το βέλτιστο υπερεπίπεδο περνά από την αρχή των αξόνων.

Προηγουμένως αναφέραμε ότι δεδομένου ενός συνόλου εκπαίδευσης $\{x_i, d_i\}_{i=1}^{N}$ σκοπός είναι η εύρεση των παραμέτρων w_0 και b_0 των οποίων το διαχωριστικό υπερεπίπεδο είναι βέλτιστο. Το ζεύγος w_0, b_0 πρέπει να ικανοποιεί τους περιορισμούς:

$$w_0^T + b_0 \geq +1 \quad \text{για } d_i = +1$$

$$w_0^T + b_0 \leq -1 \quad \text{για } d_i = -1$$

[5.9.6]

Στα ανύσματα (x_i,d_i) για τα οποία ισχύει η πρώτη ή η δεύτερη γραμμή της εξίσωσης [5.9.6] <u>γίνει ισότητα</u>, τότε αυτά τα ανύσματα είναι τα διανύσματα υποστήριξης (support vectors). Όταν ολοκληρωθεί η διαδικασία εκπαίδευσης και βρεθούν όλα τα διανύσματα υποστήριξης, όλα τα εναπομείναντα διανύσματα εισόδου δεν μας ενδιαφέρουν. Τα διανύσματα υποστήριξης αφού βρίσκονται κοντύτερα στο βέλτιστο υπερεπίπεδο είναι και τα πιο δύσκολα διανύσματα να ταξινομηθούν.

Θεωρούμε τώρα ένα διάνυσμα υποστήριξης $x^{(s)}$για το οποίο $d^{(s)} = +1$ τότε έχουμε:

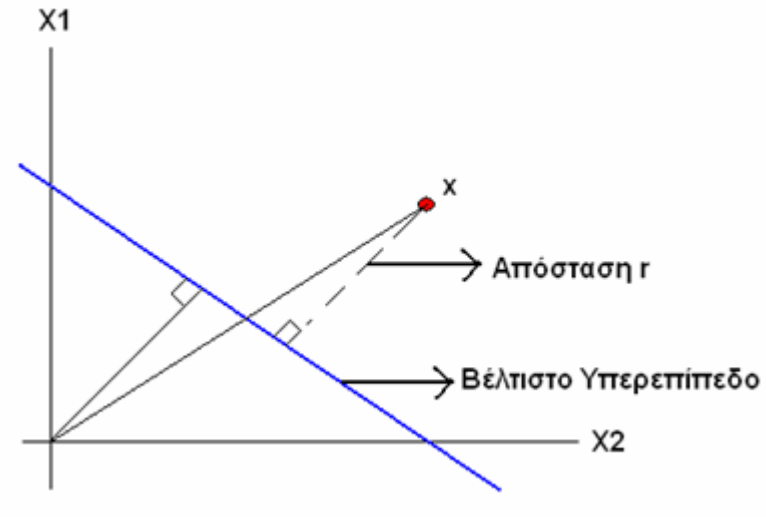
$$= w_0^T$$

g($x^{(s)}$) $.x^{(s)} + b_0 = \pm 1$ για $d^{(s)} = \pm 1$

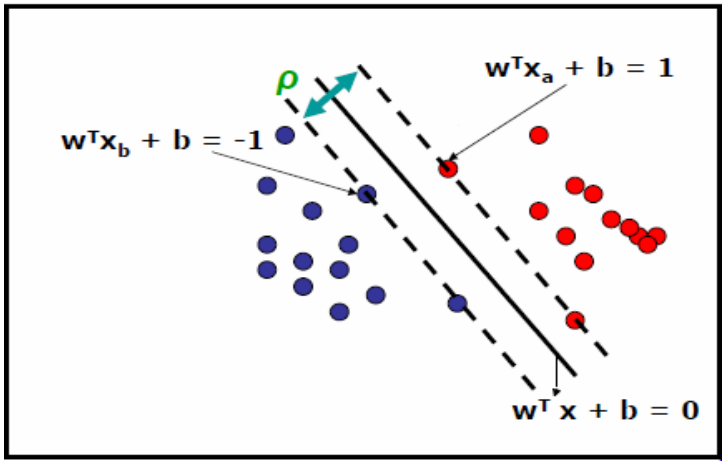
[5.9.7]

Από την εξίσωση [5.9.5] παίρνουμε την αλγεβρική απόσταση του διανύσματος υποστήριξης $x^{(s)}$ από το βέλτιστο υπερεπίπεδο, η οποία είναι:

r [5.9.8]



Εικόνα 5.9.3: Γραφική αναπαράσταση των αλγεβρικών αποστάσεων σημείων από το βέλτιστο υπερεπίπεδο.



Εικόνα 5.9.4: Διαχωρίζον υπερεπίπεδο $W^T.x + b = 0$

Τα παραδείγματα εκπαίδευσης που πέφτουν στα υπερεπίπεδα και

(πλευρές που καθορίζουν τα όρια) είναι τα διανύσματα

υποστήριξης(support vectors). Όλα τα άλλα παραδείγματα που δεν είναι support vectors έχουν τιμή μηδέν(0), ενώ τα διανύσματα υποστήριξης έχουν τιμή α>0.

Όταν γίνεται η δοκιμή(test), τα παραδείγματα δοκιμής συγκρίνονται μόνο με τα διανύσματα υποστήριξης όπου είναι τα σημαντικότερα διανύσματα και αποφασίζεται από τον αλγόριθμο σε ποια κατηγορία(μεριά)θα τοποθετηθούν.

Τα υπερεπίπεδα που καθορίζουν τις πλευρές του ορίου είναι το υπερεπίπεδο

για την κλάση +1 και για την κλάση -1 το υπερεπίπεδο .

$$\mathbf{H_{+1}(w,b) = \{x \in X:\ W^T.x + b = +1, \text{για } Y = +1\}}$$

$$\mathbf{H_{-1}(w,b) = \{x \in X:\ W^T.x + b = -1, \text{για } Y = -1\}}$$

Τα διανύσματα υποστήριξης απέχουν <u>απόσταση **γ**</u> τους από το βέλτιστο διαχωριστικό υπερεπίπεδο ($w_0^T.x + b_0 = 0$) η οποία απόσταση γ είναι: [σχέση 5.9.9] , ισχύει για τα διανύσματα υποστήριξης και των δύο (2) κατηγοριών +1 και -1 με μόνη διαφορά ότι για την κατηγορία -1 έχουμε απόσταση $\gamma = -\frac{1}{\|w\|}$.

$$\boldsymbol{\gamma} = \frac{w^T + b}{\|w\|} = \frac{1}{\|w\|}$$

[5.9.9]

Το περιθώριο(margin) ***ρ*** μεταξύ των δύο διαφορετικών κατηγοριών(και

) έχει απόσταση: και αποδεικνύεται παρακάτω από την σχέση [5.9.10].

$$\rho = \min_{x_i,\, y_i=+1}\left\{\frac{|w^T.x_i + b|}{\|w\|}\right\} + \min_{x_i,\, y_i=-1}\left\{\frac{|w^T.x_i + b|}{\|w\|}\right\} = \frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|} \qquad [5.9.10]$$

ή εναλλακτικά το περιθώριο ***ρ*** μπορεί να υπολογιστεί χρησιμοποιώντας την απόσταση μεταξύ των δύο υπερεπιπέδων (H_{+1} και H_{-1})

$$\rho = d(H_{+1}(w,b), H_{-1}(w,b)) = \frac{|1-(-1)|}{\|w\|} = \frac{2}{\|w\|}$$

[5.9.11]

ή ακόμα πιο απλά το περιθώριο $\boldsymbol{\rho}$ υπολογίζεται χρησιμοποιώντας την απόσταση r:

$$\rho = 2r = \frac{2}{\|w\|}$$

[5.9.12]

Από την εξίσωση [5.9.10] καταλαβαίνουμε ότι το πρόβλημα μεγιστοποίησης της βέλτιστης διαχωριστικής επιφάνειας μεταξύ δύο (2) κλάσεων είναι ισοδύναμο με το πρόβλημα ελαχιστοποίησης της Ευκλείδειας νόρμας του διανύσματος βαρών w.

Για παράδειγμα ας υποθέσουμε ότι έχουμε στη διάθεση μας μια συλλογή δεδομένων εκπαίδευσης ως διανύσματα ενός διανυσματικού χώρου μεγάλου πλήθους διαστάσεων και προσπαθούν να εντοπίσουν σε αυτό το χώρο ένα υπερεπίπεδο το οποίο διαχωρίζει τα θετικά από τα αρνητικά παραδείγματα εκπαίδευσης. Το πλήθος των διαστάσεων που θα δημιουργηθούν καθορίζεται από το πλήθος των χαρακτηριστικών(διανύσματα) που έχουν τα εκάστοτε αντικείμενα-πρότυπα προς εκπαίδευση. Αν από μία συλλογή δεδομένων εκπαίδευσης η οποία έχει K αντικείμενα-πρότυπα και το καθένα είναι ένα διάνυσμα με πλήθος N, τότε αυτά μπορούν να θεωρηθούν ως σημεία(π.χ. συμβολισμένα με: x,+,*,o) σε ένα N-διάστατο χώρο κατόπιν μαθηματικής πράξης που πραγματοποιείτε από τον αλγόριθμο, στην περίπτωση μας η μαθηματική πράξη γίνεται από τον γραμμικό ταξινομητή (Classifier) SVM.

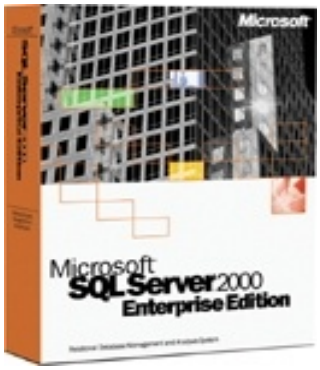
Ο αλγόριθμος SVM του spider matlab toolbox **[7]** είναι ένας ταξινομητής γραμμικά διαχωρίσιμων αντικειμένων-προτύπων, δηλαδή τα αντικείμενα εκτός του ότι πρέπει να έχουν όλα(σε train και σε test) ίδιο πλήθος χαρακτηριστικών πρέπει επίσης <u>να είναι και γραμμικά διαχωρίσιμα μεταξύ τους με τουλάχιστον ένα (1) χαρακτηριστικό τους.</u>

Κεφάλαιο 6

Βάσεις Δεδομένων (DataBases)

Μία βάση δεδομένων είναι μια δομημένη συλλογή καταχωρημένων δεδομένων αποθηκευμένη σε σκληρό δίσκο ενός Η/Υ. Με λίγα λόγια μια βάση δεδομένων είναι ένα είδος ηλεκτρονικής αρχειοθήκης, ένας χώρος όπου αποθηκεύονται τα δεδομένα σε ηλεκτρονική μορφή προσφέροντας μας ευελιξία, ταχύτητα και μεταφερσιμότητα. Τα πλεονεκτήματα των δεδομένων όπου είναι αποθηκευμένα σε ηλεκτρονική μορφή είναι πάρα πολλά, μερικά πλεονεκτήματα είναι **[1]**:

- Σε βάσεις δεδομένων που χρησιμοποιούν το πρότυπο ODBC δίνεται η δυνατότητα στο χρήση να «συνδέει» τις εφαρμογές του(Matlab, C++ Builder) ανεξάρτητα με ποιο πρόγραμμα DBMS χρησιμοποιεί
- Ταχύτατη αναζήτηση δεδομένων – καταχωρήσεων μέσω sql ερωτημάτων(Views)
- Σε όλους τους πίνακες μιας βάσης δεδομένων δημιουργούμε μία στήλη σε κάθε πίνακα ως πρωτεύων κλειδί, με αυτόν τον τρόπο εξασφαλίζουμε ότι η κάθε καταχώρηση που γίνεται στη βάση δεδομένων να είναι μοναδική(αποφυγή διπλότυπων εγγραφών)
- Κάθε βάση δεδομένων έχει δύο αρχεία που την αντιπροσωπεύουν, το $1^{ο}$ είναι το αρχείο .MDF και το $2^{ο}$ το αρχείο .LDF. Καταλαβαίνουμε ότι έχουμε δυνατότητα μεταφερσιμότητας σε οποιονδήποτε Η/Υ.



Το πρόγραμμα SQL Server 2000 χρησιμοποιείται στην παρούσα πτυχιακή για εισαγωγή δεδομένων από βάσεις δεδομένων για πραγματοποίηση των πειραμάτων μας. Μία βάση δεδομένων αποτελείται κυρίως από πίνακες(tables), από διαγράμματα(diagrams) και από ερωτήματα(views). Το πρόγραμμα Microsoft SQL Server 2000 υποστηρίζει την γλώσσα προγραμματισμού SQL όπου είναι και η πιο διαδεδομένη γλώσσα προγραμματισμού βάσεων δεδομένων. Σε ένα οποιοδήποτε πίνακα έχουμε γραμμές(rows) και στήλες(columns), οι στήλες του πίνακα έχουν διαφορετική ονομασία μεταξύ τους και μπορούν να διαφέρουν ως προς τον τύπο δεδομένων τους(π.χ. varchar, int, float, char).

Ένα απλό παράδειγμα όπου προσπελαύνουμε δεδομένα από βάσεις δεδομένων είναι όταν κάνουμε αναζήτηση στο διαδίκτυο από μία μηχανή αναζήτησης π.χ. google, πατώντας το κουμπί αναζήτησης(search) και στην συνέχεια εμφανίζονται τα

αποτελέσματα στην οθόνη μας(ιστοσελίδες, εικόνες, αρχεία κειμένου). Πλέον με τα σημερινά δεδομένα υπάρχουν σχεδόν παντού εγκατεστημένα προγράμματα βάσεων δεδομένων για αποθήκευση πληροφοριών, οργάνωση και γρήγορη προσπέλαση δεδομένων διαφόρων Α.Τ.Ε.Ι., βιομηχανιών, κυβερνήσεων, αποθηκών, υπεραγορών, εταιρειών κ.ά..

6.1 SQL (Structured Query Language)

Η γλώσσα προγραμματισμού SQL είναι μία δομημένη γλώσσα αναζήτησης που σχεδιάστηκε για τη διαχείριση δεδομένων σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων RDBMS(Relational DataBase Management System) και αρχικά βασίστηκε στη σχεσιακή άλγεβρα.

Τα βασικά τμήματα της SQL είναι:

- Γλώσσα Ορισμού Δεδομένων(DDL, Data Definition Language)
- Γλώσσα Χειρισμού Δεδομένων(DML, Data Manipulation Language)
- Περιέχει εντολές για τον ορισμό και την επεξεργασία συναλλαγών(transactions) και εντολές για την ασφάλεια(authentication)

6.2 Τι είναι τα ερωτήματα (views) SQL;

Στα ερωτήματα sql μπορούμε να γράψουμε sql κώδικα ανάλογα πάντα με το τι ψάχνουμε να βρούμε από την τρέχουσα βάση δεδομένων. Εκτός από αναζήτηση μπορούμε επίσης να διαγράψουμε δεδομένα και να κάνουμε εισαγωγή(update) νέων δεδομένων. Ένα παράδειγμα ερωτήματος είναι να βρίσκουμε καθημερινά σε μία υπεραγορά όταν κλείνουν τα ταμεία της πόσοι πελάτες έκαναν αγορές πάνω από εκατό ευρώ, ποιοι είναι ονομαστικά και ποια προϊόντα προτιμούνται από αυτούς περισσότερο. Το αποτέλεσμα μας εμφανίζεται σε ένα νέο πίνακα με τα επιθυμητά αποτελέσματα. Με την έκδοση Maltab R2008a μπορούμε μέσα σε M-files να γράψουμε ερωτήματα sql, πράγμα που μας είναι απαραίτητο για την υλοποίηση της εφαρμογής μας.

6.3 ODBC (Open DataBase Connectivity) πρότυπο

Η εταιρεία Microsoft το έτος 1991 παρουσίασε μία νέα μέθοδο για πρόσβαση σε δεδομένα η οποία ονομάστηκε ODBC δηλαδή ανοικτή συνδεσιμότητα βάσεων δεδομένων, αυτή η μέθοδος είναι η πλέον τυποποιημένη διεπαφή και έγινε αποδεκτή από όλη την βιομηχανία που φτιάχνει προγράμματα βάσεων δεδομένων. Διάφορες εταιρείες πληροφορικής όπως είναι η IBM, Novell, Oracle, Sybase, Digital και Lotus καθώς και με κοινοπραξία των οργανισμών ANSI, ISO, x/Open SQL Access Group έκαναν πραγματικότητα την τυποποίηση της διεπαφής ODBC. Μερικοί κατασκευαστές όπως η IBM, Informix και Watcom σχεδίασαν δικές τους βάσεις δεδομένων βασιζόμενοι στο ODBC πρότυπο **[2]**.

6.4 Στόχος του προτύπου ODBC

Στόχος του προτύπου ODBC είναι να μπορεί ο χρήστης από οποιανδήποτε εφαρμογή π.χ. Matlab ανεξάρτητα με ποιο DBMS(DataBase Management System, Σύστημα Διαχείρισης Βάσεων Δεδομένων) έχει να μπορεί να προσπελαύνει δεδομένα μιας βάσης δεδομένων γλιτώνοντας και χρόνο και κόπο. Προϋπόθεση ότι η εφαρμογή(Matlab) και το DBMS να είναι συμβατά με το πρότυπο ODBC όταν μια εφαρμογή εκτελεί εντολές ODBC το DBMS να δίνει σωστά αποτελέσματα. Το database driver είναι ένα ενδιάμεσο επίπεδο του DBMS όπου μέσω αυτού επιτυγχάνεται η σύνδεση DBMS και εφαρμογής (π.χ. Matlab, C++ Builder).

Το πρότυπο ODBC (Open Database Connectivity, Ανοικτή Συνδεσιμότητα Βάσεων Δεδομένων) το χρησιμοποιούμε για την σύνδεση της βάσης δεδομένων του προγράμματος Microsoft SQL Server 2000 με το matlab. Στην πτυχιακή εργασία χρησιμοποιήθηκε το σύστημα διαχείρισης βάσης δεδομένων Microsoft SQL Server 2000 με το οποίο πραγματοποιείται η σύνδεση του SQL Server με το Matlab. Η σύνδεση μεταξύ Matlab και SQL Server γίνεται μέσω του πρότυπου ODBC που υπάρχει μέσα στο πρόγραμμα Microsoft SQL Server 2000.

Κεφάλαιο 7

Matlab

7.1 Ιστορική Αναδρομή

Το πρόγραμμα matlab (Matrix Laboratory) δημιουργήθηκε από τον Cleve Moler σε γλώσσα προγραμματισμού Fortran την δεκαετία του 1970. Το 1984 η εταιρεία Mathworks ιδρύεται αναλαμβάνοντας το έργο που δημιούργησε ο Cleve Moler και η χρήση του matlab από αποκλειστικά εκπαιδευτική γίνεται τώρα και εμπορική η χρήση του, σήμερα το πρόγραμμα matlab είναι γραμμένο σε γλώσσα προγραμματισμού C και Java **[5]**.

Μερικές τυπικές χρήσεις του matlab:

- Αποδοτική υλοποίηση αλγορίθμων
- Ανάπτυξη λογισμικού (GUI)
- Επεξεργασία σημάτων
- Αριθμητική ανάλυση, γραμμική άλγεβρα
- Εμπορικές εφαρμογές (Motorola, Toyota, Boeing, Philips)

7.2 Χρήση του matlab

Το matlab πλέον χρησιμοποιείται ευρύτατα σε όλους σχεδόν τους επιστημονικούς τομείς για υπολογισμό τεράστιου όγκου υπολογιστικών πράξεων απαλλάσσοντας τους καθηγητές, επιστήμονες και μαθητές σε χρόνο και κόπο παράγοντας αξιόπιστα αποτελέσματα σε πολύ λίγο χρονικό διάστημα.

Το matlab μπορεί να χρησιμοποιηθεί ως μια εύχρηστη αριθμομηχανή και ως μια γλώσσα προγραμματισμού (C) όπου μπορούν να δημιουργηθούν απλά ή σύνθετα προγράμματα(M-Files). Επίσης το matlab είναι αποκλειστικά σχεδιασμένο για επιστημονικούς τομείς πράγμα που μπορεί να θεωρηθεί ως μειονέκτημα γιατί δεν μπορεί προς το παρόν να χρησιμοποιηθεί από το ευρύ κοινό για άλλους σκοπούς.

Το πρόγραμμα matlab είναι διαθέσιμο για λειτουργικά συστήματα όπως είναι τα Windows, Macintosh και Unix πράγμα που καθιστά τον κώδικα που είναι γραμμένος σε αρχεία M-Files να έχει πλήρη μεταφερσιμότητα – συμβατότητα σε οποιοδήποτε Η/Υ με οποιοδήποτε λειτουργικό σύστημα.

Εικόνες που δείχνουν την εμπορική πλέον χρήση του προγράμματος matlab.

7.3 Χρήση του Matlab στα δικά μας πειράματα

Η έκδοση (version) του matlab που χρησιμοποιείται για τα πειράματα μας είναι η 7.6.0.324(R2008a) όπου μας δίνει την δυνατότητα να δημιουργούμε GUI καθώς και να εισάγουμε πίνακες με δεδομένα από το πρόγραμμα SQL Server 2000 μέσω ενός ODBC driver ενώ π.χ. η έκδοση 6.5 του matlab δεν μας παρέχει αυτές τις δυνατότητες.

Στα πειράματα γραμμικής ταξινόμησης που πραγματοποιούμε επιλέχθηκε ως κατάλληλο εργαλείο για τους υπολογισμούς που απαιτούνται το πρόγραμμα matlab καθώς και του ότι ο αλγόριθμος SVM από το SPIDER matlab toolbox **[7]** είναι γραμμένος σε M-Files αρχεία(συμβατότητα).

7.4 Matlab και Βάσεις Δεδομένων

Ο ταυτόχρονος συντονισμός μεταξύ του προγράμματος matlab και του προγράμματος SQL Server 2000 απαιτείται για τα πειράματα που πραγματοποιούμε στην πτυχιακή αυτή καθώς ο χρήστης όταν η εφαρμογή βρίσκεται σε λειτουργία πρέπει να εισάγει τα εκάστοτε δεδομένα από την εκάστοτε βάση δεδομένων που βρίσκεται μέσα στο πρόγραμμα SQL Server 2000.

Μια τελευταία ρύθμιση είναι να δημιουργήσουμε ένα νέο πρότυπο ODBC το οποίο δημιουργείται μέσω του SQL Server, στο πρότυπο αυτό δίνεται η ονομασία “sql_odbc_data_source” το οποίο χρησιμοποιείται μέσα στον κώδικα της

εφαρμογής μας στο matlab. Επίσης με την ίδια ακριβώς ονομασία εμφανίζεται στο GUI μας, άρα επιλέγουμε για data source την επιλογή “sql_odbc_data_source”.

Κεφάλαιο 8

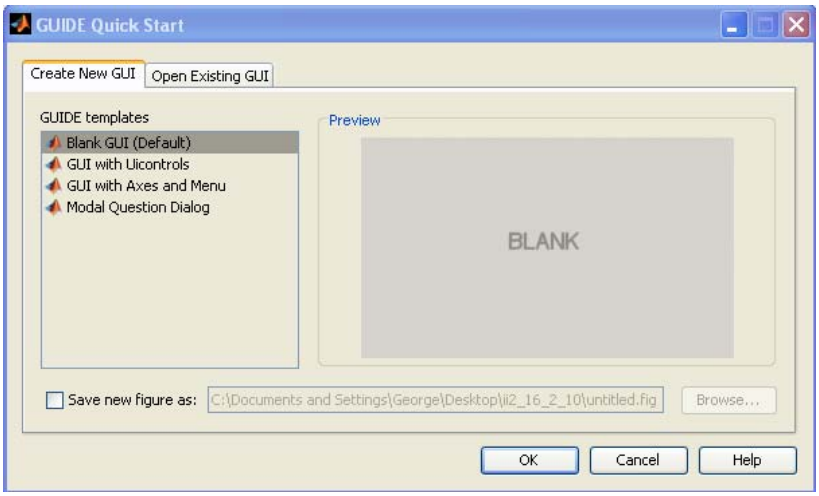
ΜΕΡΟΣ Α

Υλοποίηση εφαρμογής και επεξήγηση κώδικα χρησιμοποιώντας την εργαλειοθήκη SPIDER Matlab Toolbox

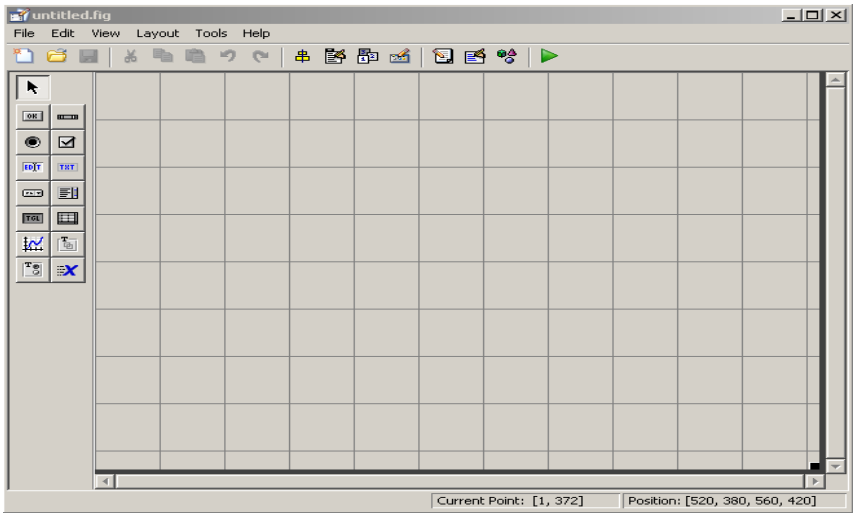
Σε αυτό το κεφάλαιο επεξηγείται με πλήρη σαφήνεια ο τρόπος με τον οποίο η εφαρμογή SVM_Classification.fig ήρθε εις πέρας. Επεξηγείται πλήρως ο κώδικας που χρησιμοποιήθηκε στην εφαρμογή για να μπορέσει ο οποιοσδήποτε αναγνώστης της πτυχιακής να καταλάβει πολύ απλά το πώς και γιατί χρησιμοποιούνται διάφορες εντολές, συναρτήσεις και μεταβλητές.

8.1 Αρχικά Βήματα

Το γραφικό περιβάλλον(figure) της εφαρμογής μας έχει ονομασία SVM_Classification.fig, γράφοντας την εντολή guide στο command prompt του Matlab εμφανίζεται το παρακάτω παράθυρο:

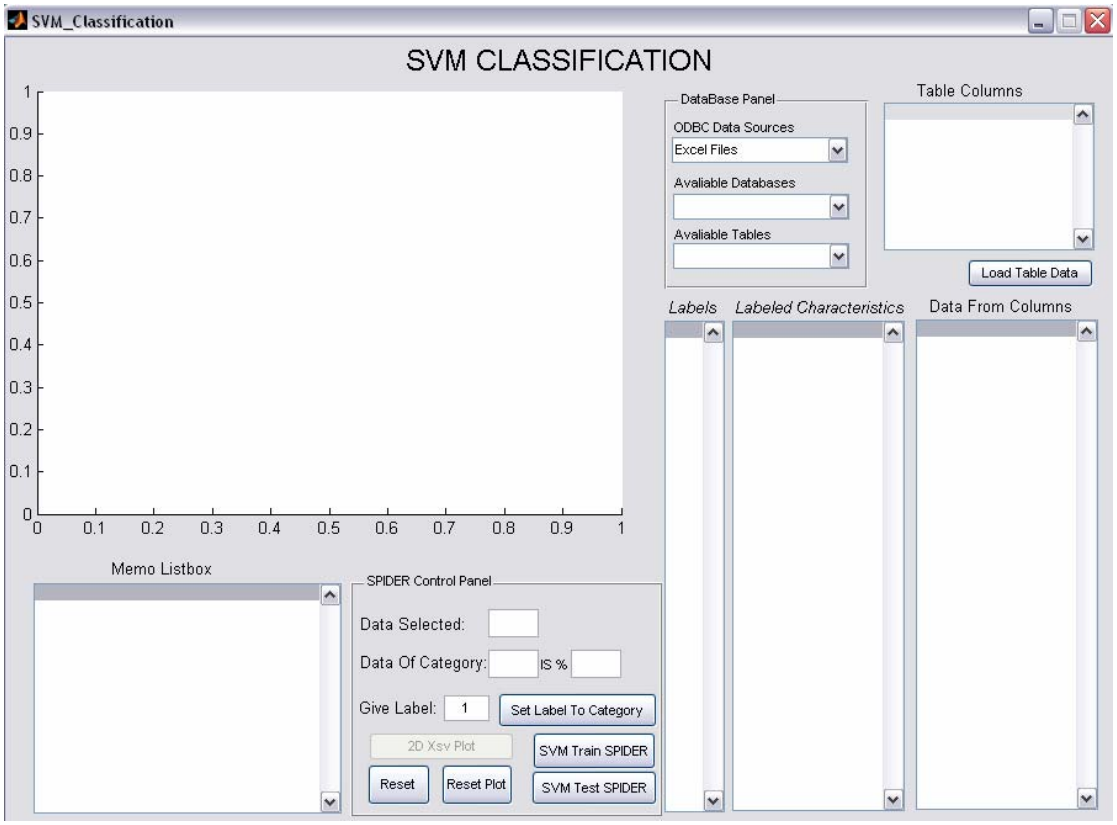


Παρατηρούμε λοιπόν ότι έχουμε τις εξής επιλογές, ή να δημιουργήσουμε ένα νέο GUI(Grafical User Interface) γραφικό περιβάλλον ή να τρέξουμε ένα GUI που ήδη υπάρχει. Αν θέλουμε να δημιουργήσουμε ένα καινούργιο γραφικό περιβάλλον, από την καρτέλα Create New GUI επιλέγουμε την επιλογή Blank GUI (Default) και πατώντας το κουμπί OK μας εμφανίζεται ένα παράθυρο που έχει την εξής μορφή:



Στην αριστερή πλευρά του GUI βλέπουμε ότι υπάρχουν διάφορα components όπως “Push Button”, “Edit Text”, “Listbox”, “Panel” κ.ά. όπου μπορούμε να τοποθετήσουμε επάνω στο GUI για υλοποίηση εφαρμογής.

Το GUI της πτυχιακής εργασίας έχει την παρακάτω μορφή:



Εκτελώντας την εντολή guide στο command prompt, πηγαίνουμε στην καρτέλα Open Existing GUI και βρίσκουμε μέσω του κουμπιού αναζήτησης την εφαρμογή

που θέλουμε να εκτελέσουμε. Η εφαρμογή μας λέγεται SVM_Classification.fig και ανοίγοντας την εμφανίζεται το παραπάνω παράθυρο. Στο GUI(Grafical User Interface) υπάρχουν τα εξής components: static text, popupmenu, listbox, axes, pushbutton και edit text. Τα components αποτελούνται από συναρτήσεις (functions), μερικές από αυτές τις συναρτήσεις είναι η συνάρτηση CreateFnc και η συνάρτηση Callback. Το κάθε component έχει και μια ετικέτα(tag) δηλαδή έχει μια ονομασία όπου το Matlab το αναγνωρίζει μέσα στον κώδικα του GUI μας, την ετικέτα(tag) του κάθε component μπορούμε να την δούμε στο Property Inspector κάνοντας δεξί κλικ επάνω στο component επιλέγοντας την επιλογή Property Inspector.

8.2 Αρχικοποίηση μεταβλητών κατά την εκκίνηση της εφαρμογής

Καταρχάς πρέπει να αρχικοποιήσουμε στην συνάρτηση(function SVM_Classification_OpeningFcn) όπου μόλις πατήσουμε το κουμπί Run Figure και πριν γίνει σε εμάς ορατό το GUI αρχικοποιούνται δύο(2) πίνακες της εφαρμογή μας, ο πίνακας handles.Y_axis και ο πίνακας handles.oldrowselection όπως βλέπουμε στον παρακάτω κώδικα:

```
%adiazo ton pinaka me tis times twn labels
        handles.Y_axis=[];
%adiazo ton pinaka oldrowselection
        handles.oldrowselection=[];
```

Όταν έρθει εις πέρας η εκπαίδευση(train) των δεδομένων μας αμέσως μετά ακολουθεί η δοκιμή των εναπομίναντων δεδομένων(test). Αδειάζουμε ξανά τους πίνακες αυτούς όταν θα κάνουμε την δοκιμή(test) των νέων δεδομένων απλά πατώντας το κουμπί “Reset”. Αδειάζετε το listbox με static text “Labels” και το listbox με static text “Labeled Characteristics” όπου μπαίνουν τα νέα δεδομένα προς δοκιμή(test).

8.3 UiPanel με τίτλο DataBase Panel

Στο DataBase panel υπάρχουν τα εξής components:

1. Το popupmenu με static text ODBC Data Source
2. Το popupmenu με static text Avaliable Databases
3. Το popupmenu με static text Avaliable Tables

8.3.1.1 Popupmenu με static text ODBC Data Source (CreateFnc)

Εμφανίζονται στο popupmenu με static text "ODBC Data Sources" τα διαθέσιμα data sources.

Σημαντική επισήμανση

{{Η CreateFnc συνάρτηση όλων των components εκτελείται μόλις πατάμε το κουμπί Run Figure του GUI.}}

{{Στην συνάρτηση set το hObject υποδουλώνει το ίδιο το component. }}

Η CreateFnc έχει ετικέτα(tag) "popupmenu_odbc_data_source" όπου εκτελείται ο παρακάτω κώδικας:

```
setdbprefs('TempDirForRegistryOutput','c:\temporary');
myODBCdir = getenv('WINDIR');
dt_sources= getdatasources;
set(hObject,'String',dt_sources);
```

Επεξήγηση Κώδικα

- setdbprefs('TempDirForRegistryOutput','c:\temporary');
- myODBCdir = getenv('WINDIR');
- dt_sources= getdatasources;

Σύνταξη εντολής setdbprefs→ setdbprefs(Property, Allowable Values).

setdbprefs→ ορισμός προτιμήσεων για ανάκτηση δεδομένων.

TempDirForRegistryOutput→ Χρησιμοποιείται όταν προσθέτουμε data sources και δεν έχουμε πρόσβαση εγγραφής στον τρέχον κατάλογο του Matlab.

Κατ' αρχάς με την εντολή setdbprefs και με το πεδίο TempDirForRegistry ορίζουμε την προσωρινή διαδρομή που θα γραφτούν τα δεδομένα της σύνδεσης ODBC με την

εκτέλεση της εντολής getdatasources, στην διαδρομή c:\temporary δημιουργείτε ένα αρχείο με όνομα temporaryODBC.ini όπου έχει τα εξής περιεχόμενα:

```
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\SOFTWARE\ODBC\ODBC.INI\ODBC Data Sources]
"dBASE Files"="Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)"
"Excel Files"="Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
"MS Access Database"="Microsoft Access Driver (*.mdb, *.accdb)"
"sql_odbc_data_source"="SQL Server"
```

Η 2η εντολή που εκτελείται είναι η εντολή myODBCdir = getenv('WINDIR'); όπου θέτει την διαδρομή ως c:\Windows. Επιστρέφει τα έγκυρα ODBC και JDBC data sources που υπάρχουν στο σύστημα στην μεταβλητή dt_sources σε μορφή αλφαριθμητικών κελιών όπως φαίνεται παρακάτω:

'Excel Files' 'LocalServer' 'MS Access Database' 'dBASE Files' 'sql_odbc_data_source'

Η 3η εντολή getdatasources παίρνει τα ονόματα των ODBC data sources από το αρχείο ODBC.INI το οποίο βρίσκεται στον κατάλογο c:\Windows.

Το αρχείο ODBC.INI έχει τα εξής δεδομένα:

```
[ODBC 32 bit Data Sources]
sql_odbc_data_source=SQL Server (32 bit)
[sql_odbc_data_source]
Driver32=C:\WINDOWS\system32\SQLSRV32.dll
```

Το data source που συνδέει τον SQL Server με το Matlab έχει την ονομασία sql_odbc_data_source.

- set(hObject,'String',dt_sources);

Στην συνάρτηση set το hObject υποδουλώνει το ίδιο το component δηλαδή το popupmenu_odbc_data_source και στο 'String' του popupmenu θέτονται τα περιεχόμενα της μεταβλητής dt_sources.

8.3.1.2 Popupmenu με static text ODBC Data Source (Callback)

Επιλέγεται από τον χρήστη το data source «sql_odbc_data_source», μόλις επιλεχθεί το data source εμφανίζονται οι διαθέσιμες βάσεις δεδομένων που υπάρχουν στον SQL Server στο popupmenu με static text “Avaliable Databases”.

Σημαντικές Επισημάνσεις

{{Σε όλα τα components που έχουν Callback συνάρτηση εκτελείται ο κώδικας που υπάρχει σε αυτά κάθε φορά που πατάμε στο component είτε αυτό είναι popupmenu, button, listbox ή σε οποιοδήποτε άλλο component.}}

{{Στο Matlab υπάρχει μια δομή με όνομα handles, γράφοντας π.χ. handles.var1 η μεταβλητή var1 μπορεί να γίνεται προσπελάσιμη από όλες τις συναρτήσεις γράφοντας απλά handles.var1.}}

{{ guidata(hObject,handles)→ Βάζουμε αυτή την γραμμή κώδικα στο τέλος κάθε συνάρτησης Callback για να γίνει ανανέωση(update) των περιεχομένων των μεταβλητών με δομή handles, ως αποτέλεσμα να παίρνουμε τα πιο πρόσφατα δεδομένα από τις μεταβλητές μας που έχουν δομή handles.}}

{{ Η εντολή disp(var1) εμφανίζει στο command prompt του Matlab τα περιεχόμενα(τα πεδία) της μεταβλητής var1 για να βλέπουμε τα περιεχόμενα μεταβλητών που θέλουμε να εξετάσουμε βγάζοντας συμπεράσματα και κατανοώντας καλύτερα την εφαρμογή μας. Δεν εμφανίζεται κατά την διάρκεια εκτέλεσης της εφαρμογής καμία disp αλλά υπάρχει υπο μορφή σχολίων μέσα στον κώδικα μας για καλύτερη κατανόηση της εφαρμογής. Επίσης χρησιμοποιείται για προβολή πληροφοριών και για ανίχνευση τυχόν λαθών καθώς και για σκοπούς ελέγχου και διάφορων τροποποιήσεων που έγιναν. }}

Στο Callback του popupmenu με static text “ODBC Data Sources”, εκτελείται ο παρακάτω κώδικας:

```
dt_sources_list=get(handles.popupmenu_odbc_data_source,'String');
selected_dt_source=get(handles.popupmenu_odbc_data_source,'Value');
handles.selected_dts=dt_sources_list(selected_dt_source);
handles.conn = database(handles.selected_dts,'','');
```

```
handles.metadata= dmd(handles.conn);
sources=get(handles.metadata);
set(handles.popupmenu_avaliable_databases,'String',sources.Catalogs);
guidata(hObject,handles);
```

Επεξήγηση Κώδικα

- dt_sources_list=get(handles.popupmenu_odbc_data_source,'String');

Παίρνουμε όλα τα περιεχόμενα του πεδίου 'String' του popupmenu με ετικέτα(tag) «popupmenu_odbc_data_source» και τα αποθηκεύουμε στην μεταβλητή dt_sources_list χρησιμοποιώντας την συνάρτηση get.

- selected_dt_source=get(handles.popupmenu_odbc_data_source,'Value');

Πάλι με την βοήθεια της συνάρτησης get και του πεδίου 'Value' του popupmenu παίρνουμε <u>το επιλεγμένο data source του χρήστη</u> και το αποθηκεύουμε στην μεταβλητή selected_dt_source.

- handles.selected_dts=dt_sources_list(selected_dt_source);

<u>Παίρνουμε την πιο πρόσφατη επιλογή χρήστη</u> και την αποθηκεύουμε στην μεταβλητή selected_dts σε handles δομή για να μπορεί η μεταβλητή να γίνεται προσπελάσιμη και από άλλες συναρτήσεις.

handles.selected_dts→ {sql_odbc_data_source} Στο ODBC Data Source όπου γίνεται η σύνδεση του προγράμματος SQL Server 2000 με το πρόγραμμα maltab δόθηκε η ονομασία sql_odbc_data_source και αυτό επιλέγουμε.

- handles.conn = database(handles.selected_dts,'','');

Σύνταξη συνάρτησης database→ database(odbc datasource name, username, password).

Το odbc datasource βρίσκεται αποθηκευμένο στην μεταβλητή handles.selected_dts, ως username δεν ορίσαμε τίποτα και γι'αυτό βάζουμε '' (δύο μονά εισαγωγικά χωρίς κενό μεταξύ τους) καθώς επίσης και στο password ''.

<u>handles.conn→ είναι ένα αντικείμενο βάσης δεδομένων{database object: 1-by-1}, όπου σ' αυτό υπάρχει το data source μας «sql_odbc_data_source».</u>

- handles.metadata= dmd(handles.conn);

Σύνταξη εντολής dmd→ D= dmd(c)

dmd→κατασκευάζει ένα αντικείμενο βάσεων δεδομένων metadata(Database MetaData object).

c→Είναι ένα αντικείμενο σύνδεσης βάσεων δεδομένων (δηλαδή το handles.conn).

disp(handles.metadata)→{dmd object: 1-by-1}, από αυτό το αντικείμενο βάσεων δεδομένων χρειαζόμαστε το πεδίο «Catalogs» που περιέχει τα διαθέσιμα ονόματα βάσεων δεδομένων που υπάρχουν στον SQL Server.

- sources=get(handles.metadata);

disp(sources)

```
%                    AllProceduresAreCallable: 1
%                    AllTablesAreSelectable: 1
%                    DataDefinitionCausesTransactionCommit: 1
%                    DataDefinitionIgnoredInTransactions: 0
%                    DoesMaxRowSizeIncludeBlobs: 0
%                    Catalogs: {6x1 cell}
```

Το πεδίο Catalogs περιέχει σε μορφή κελιών τα ονόματα των διαθέσιμων βάσεων δεδομένων που υπάρχουν, επειδή χρειαζόμαστε τα ονόματα των βάσεων δεδομένων να εμφανιστούν στο popupmenu με static text “Avaliable Databases ” χρησιμοποιούμε το πεδίο Catalogs όπως φαίνεται στην παρακάτω γραμμή κώδικα:

- set(handles.popupmenu_avaliable_databases,'String',sources.Catalogs);

Για να έχουμε πρόσβαση στο πεδίο Catalogs της μεταβλητής sources βάζουμε μια τελεία(.) ενδιάμεσα της μεταβλητής και του πεδίου(sources.Catalogs). Με την set θέτουμε να γραφτούν όλα τα ονόματα των βάσεων δεδομένων που υπάρχουν στο πεδίο Catalogs στο popupmenu με static text “Avaliable Databases”.

- guidata(hObject,handles);

Γίνεται ανανέωση(update) της δομής handles.

8.3.2 Popupmenu με static text Avaliable Databases (Callback)

Επιλέγεται η βάση δεδομένων από τον χρήστη και εμφανίζονται οι πίνακες της επιλεγμένης βάσης στο popupmenu με static text “Available Tables”.

```
sel_list_db=get(handles.popupmenu_avaliable_databases,'String');
sel_index_db=get(handles.popupmenu_avaliable_databases,'Value');
handles.sel_db=sel_list_db(sel_index_db);
sources= tables(handles.metadata,handles.sel_db,'dbo');
handles.sourcesize= size(sources);
x=0;
for i=1:handles.sourcesize(1,1)
  if(regexp(sources{i,2},'^TABLE$'));
    x=x+1;
    handles.string{x}=sprintf('%s',sources{i});
  end
end
set(handles.popupmenu_avaliable_tables,'String',handles.string);
guidata(hObject,handles);
```

Επεξήγηση κώδικα

- sel_list_db=get(handles.popupmenu_avaliable_databases,'String');
- sel_index_db=get(handles.popupmenu_avaliable_databases,'Value');
- handles.sel_db=sel_list_db(sel_index_db);

handles.sel_db→ Παίρνουμε την βάση δεδομένων που επέλεξε ο χρήστης και την αποθηκεύουμε στην μεταβλητή handles.sel_db.

- sources= tables(handles.metadata,handles.sel_db,'dbo');

Σύνταξη συνάρτησης tables→ T = tables(D,C,S)

tables→ Επιστέφει τα ονόματα των πινάκων και των ερωτημάτων(views) της βάσης δεδομένων που επιλέχτηκε από τον χρήστη.

D→ Είναι το αντικείμενο metadata της βάσης δεδομένων, προέκυψε από ένα αντικείμενο βάσης δεδομένων {handles.metadata=dmd(handles.conn)}.

C→ Βάζουμε την μεταβλητή που περιέχει το όνομα της βάσης δεδομένων που επιλέχτηκε από τον χρήστη (handles.sel_db).

S→ Κάθε αντικείμενο που δημιουργείτε πρέπει να έχει και ένα ιδιοκτήτη. Είναι ο ιδιοκτήτης της βάσης γνωστό ως 'dbo'.

Μερικά από τα περιεχόμενα της μεταβλητής sources φαίνονται παρακάτω:

```
disp(sources)
%   'syscolumns'        'SYSTEM TABLE'
%   'syscomments'       'SYSTEM TABLE'
%   'dtproperties'      'TABLE'
%   'IRIS'              'TABLE'
%   'sysconstraints'    'VIEW'
%   'syssegments'       'VIEW'
```

Εμείς θέλουμε μόνο τις γραμμές που έχουν 'TABLE' και γι' αυτό χρησιμοποιούμε μία δομή επανάληψης και να πάρουμε μόνο αυτές τις γραμμές.

```
handles.sourcesize= size(sources);
```

size→ Η εντολή size μας επιστρέφει τις διαστάσεις ενός πίνακα, ο πρώτος αριθμός είναι ο συνολικός αριθμός γραμμών και μετά είναι ο συνολικός αριθμός των στηλών. Οι διαστάσεις αποθηκεύονται στην μεταβλητή handles.sourcesize, αυτή την πληροφορία την χρησιμοποιούμε στον βρόγχο(loop) for.

```
x=0;
for i=1:handles.sourcesize(1,1)
   if(regexp(sources{i,2},'^TABLE$'));
      x=x+1;
      handles.string{x}=sprintf('%s',sources{i});
   end
End
```

x=0;→ Αρχικοποιούμε την μεταβλητή x με το μηδέν.

Στην συνέχεια χρησιμοποιούμε μια επαναληπτική δομή for όπου εκτελείται τόσες φορές όσες είναι και ο αριθμός των γραμμών που έχει η μεταβλητή handles.sourcesize παίρνοντας το πρώτο πεδίο(1,1) που είναι αποθηκευμένο σε αυτήν όπου είναι ο συνολικός αριθμός γραμμών.

Με την if γίνεται έλεγχος σε κάθε επανάληψη εάν το πεδίο που προσπελαύνεται είναι πίνακας('TABLE').

Σύνταξη εντολής regexp→ regexp(string,expression) ταιριάζει την κανονική έκφραση(expression) με το αλφαριθμητικό(string). Στις κανονικές εκφράσεις χρησιμοποιούνται μεταχαρακτήρες όπου ο καθένας έχει την δική του σημασία, δηλαδή το ^ (^TABLE) σημαίνει ότι το αλφαριθμητικό που συγκρίνεται(sources {i,2}) πρέπει να ξεκινάει με το γράμμα T και ο μεταχαρακτήρας $ να τελειώνει με το γράμμα E (TABLE$). Εάν ισχύει η συνθήκη if:

- x=x+1;
- handles.string{x}=sprintf('%s',sources{i});

δημιουργούνται τόσα πεδία στην handles.string όσα είναι και τα διαθέσιμα ονόματα πινάκων. Αυτό γίνεται για να διαχωρίσουμε τα μόνο τα ονόματα των πινάκων από όλες τις άλλες τις πληροφορίες που υπάρχουν στην μεταβλητή sources.

- set(handles.popupmenu_avaliable_tables,'String',handles.string);

Τέλος θέτουμε στο String του popupmenu με static text "Avaliable Tables" τα ονόματα τον πινάκων για να επιλέξει ο χρήστης όποιο πίνακα επιθυμεί.

8.3.3 Popupmenu με static text Avaliable Tables (Callback)

Ο χρήστης επιλέγει τον πίνακα που επιθυμεί, ακολούθως εμφανίζονται οι στήλες του επιλεγμένου πίνακα στο listbox με static text «Table Columns».

```
sel_list_table=get(handles.popupmenu_avaliable_tables,'String');
sel_table_no=get(handles.popupmenu_avaliable_tables,'Value');
handles.sel_table=sel_list_table(sel_table_no);
handles.columns=columns(handles.metadata,handles.sel_db,'dbo',handles.sel_table
);
set(handles.listbox_table_columns,'String',handles.columns);
guidata(hObject,handles);
```

Επεξήγηση κώδικα

- sel_list_table=get(handles.popupmenu_avaliable_tables,'String');
- sel_table_no=get(handles.popupmenu_avaliable_tables,'Value');
- handles.sel_table=sel_list_table(sel_table_no);

handles.sel_table→ Σε αυτή την μεταβλητή αποθηκεύεται το όνομα του πίνακα που επέλεξε ο χρήστης.

- handles.columns=columns(handles.metadata,handles.sel_db,'dbo',handles.sel_table);

columns συνάρτηση→ Επιστρέφει μια λίστα όπου είναι τα ονόματα των στηλών του πίνακα της βάσης δεδομένων στην μεταβλητή handles.columns.

Σύνταξη της εντολής columns→ i=columns(dbmeta,'cata','sch','tab')

'tab'→ Είναι ο πίνακας που επιλέχτηκε.

΄sch΄→ Είναι ο ιδιοκτήτης βάσης δεδομένων, δηλαδή dbo.

'cata'→ Είναι το όνομα της επιλεγμένης βάσης δεδομένων.

dbmeta→ Είναι το αντικείμενο metadata της βάσης δεδομένων.

- set(handles.listbox_table_columns,'String',handles.columns);
- guidata(hObject,handles);

Με την συνάρτηση set θέτουμε στο πεδίο String του listbox με static text "Table Columns" τα ονόματα των στηλών που βρέθηκαν.

8.4 Listbox με static text Table Columns (Callback)

Ο χρήστης έχει την δυνατότητα να επιλέγει δυναμικά όποιες στήλες επιθυμεί για την πραγματοποίηση των πειραμάτων του.

handles.columns_selected=get(hObject,'Value');
handles.only_selected_columns=handles.columns(handles.columns_selected);

Αποθηκεύονται στην μεταβλητή handles.only_selected_columns μόνο οι επιλεγμένες από τον χρήστη στήλες. Η μεταβλητή handles.only_selected_columns χρησιμοποιείται στην συνάρτηση Callback του κουμπιού “Load Table Data” για να εμφανιστούν μόνο οι στήλες που επιλέχτηκαν από τον χρήστη στο listbox με static text “Data From Columns”.

8.5 Pushbutton Load Table Data (Callback)

Εμφάνιση δεδομένων των επιλεγμένων στηλών στο listbox με static text «Data From Columns».

```
DB= handles.sel_db{1};    To {1} χρειάζεται αλλιώς βγαίνει error
Table= handles.sel_table{1};  To {1} χρειάζεται αλλιώς βγαίνει error
exec(handles.conn,['USE [' DB ']']);
sel_from_user_cols=handles.only_selected_columns;
var='';
   for i=1:size(sel_from_user_cols,2)
      var{i}=str2mat(sprintf('%s,',sel_from_user_cols{1,i}));
   end
if size(sel_from_user_cols,2) == 2
      set(handles.pushbutton_2D_Xsv_Plot,'Enable','on')
else
      set(handles.pushbutton_2D_Xsv_Plot,'Enable','off')
end
var{size(sel_from_user_cols,2)}=str2mat(sprintf('%s',sel_from_user_cols{1,size(sel_fr
om_user_cols,2)}));
var=[var{:}];
setdbprefs('DataReturnFormat','numeric');
handles.LBDataN= fetch(handles.conn,['select ' var ' from [' Table ']']);
%disp(handles.LBDataN)---> 1,86 80
handles.LBDataN= handles.LBDataN(:,:);
set(handles.listbox_data_from_columns,'String',num2str(handles.LBDataN));
guidata(hObject,handles);
```

Επεξήγηση κώδικα

- DB= handles.sel_db{1};
- Table= handles.sel_table{1};

Μεταφέρουμε τα περιεχόμενα της μεταβλητής handles.sel_db{1} στην τοπική μεταβλητή DB καθώς επίσης και της μεταβλητής handles.sel_table{1} στην τοπική μεταβλητή Table.

- exec(handles.conn,['USE [' DB ']']);

exec συνάρτηση→ Εκτελεί ερώτημα μορφής SQL (EXECute SQL statement).

Σύνταξη εντολής exec→ exec(K,COMMAND) εκτελείται η εντολή(COMMAND).

Κάνουμε χρήση sql ερωτήματος USE DB το οποίο αλλάζει την προεπιλεγμένη βάση δεδομένων σε αυτή που θέσαμε.

handles.conn→ είναι το αντικείμενο βάσης δεδομένων {database object: 1-by-1}, όπου σ'αυτό υπάρχει η ονομασία του συνδετικού κρίκου μεταξύ του προγράμματος SQL Server 2000 και του προγράμματος matlab «sql_odbc_data_source».

[ ]→ Βάζουμε αριστερά και δεξιά της μεταβλητής DB γιατί εάν το όνομα της βάσης δεδομένων έχει κενά τότε το sql ερώτημα δεν θα εκτελεστεί σωστά με αποτέλεσμα να μην γίνει εφικτή η σύνδεση στην αιτούμενη από τον χρήστη βάση.

''→ Βάζουμε αριστερά και δεξιά της μεταβλητής DB ώστε το πρόγραμμα matlab να την αναγνωρίσει ως μεταβλητή και όχι ως μια λέξη που ανήκει στο sql ερώτημα.

- sel_from_user_cols=handles.only_selected_columns;
- var='';

Θέτουμε τα ονόματα των στηλών που επέλεξε ο χρήστης στην τοπική μεταβλητή sel_from_user_cols και αρχικοποιούμε την τοπική μεταβλητή var.

disp(sel_from_user_cols)→ π.χ. 'Sepal_Length' 'Petal_Width'

- for i=1:size(sel_from_user_cols,2)
- var{i}=str2mat(sprintf('%s,',sel_from_user_cols{1,i}));
- end

Χρησιμοποιώντας την δομή επανάληψης for γεμίζουμε την μεταβλητή «var» με τα ονόματα των εκάστοτε επιλεγμένων στηλών βάζοντας ένα κόμμα (,) στο τέλος τους

γιατί στο ερώτημα sql στο select τα ονόματα των στηλών χωρίζονται με ένα κόμμα (,) μεταξύ τους.

disp(var)→ 'Sepal_Length,' 'Petal_Width,'

```
if size(sel_from_user_cols,2) == 2
    set(handles.pushbutton_2D_Xsv_Plot,'Enable','on')
else
    set(handles.pushbutton_2D_Xsv_Plot,'Enable','off')
end
```

Το κουμπί «2D Xsv Plot» ενεργοποιείται μόνο εάν ο αριθμός των στηλών που επιλέχτηκαν από τον χρήστη είναι ίσος με 2, σε κάθε άλλη περίπτωση παραμένει απενεργοποιημένο.

```
var{size(sel_from_user_cols,2)}=str2mat(sprintf('%s',sel_from_user_cols{1,size(sel_from_user_cols,2)}));
var=[var{:}];
```

Αποθηκεύουμε στον πίνακα var τα ονόματα των επιλεγμένων στηλών.

```
setdbprefs('DataReturnFormat','numeric');
handles.LBDataN= fetch(handles.conn,['select ' var ' from [' Table ']']);
disp(handles.LBDataN)---> 1,86 80
handles.LBDataN= handles.LBDataN(:,:);
set(handles.listbox_data_from_columns,'String',num2str(handles.LBDataN));
```

setdbprefs→ Θέτουμε τα δεδομένα να επιστρέφονται σε αριθμητική μορφή.

fetch συνάρτηση→ X=FETCH(CONN,SQLSTRING,ROWING). Το πεδίο ROWING δεν το συμπεριλαμβάνουμε, είναι προαιρετικό. Αποθηκεύονται όλα τα δεδομένα των στηλών μας στην μεταβλητή handles.LBDataN και εμφανίζουμε το αποτέλεσμα στο listobox με static text «Data From Columns».

8.6 Listbox με static text Data From Columns (Callback)

Δυναμική επιλογή γραμμών από το listbox με static text «Data From Columns».

handles.lines_selected=get(hObject,'Value');

Επεξήγηση κώδικα

- handles.lines_selected=get(hObject,'Value');

Στο property inspector αυτού του listbox έγινε αλλαγή στα πεδία Min και Max, ως ελάχιστη επιλογή γραμμών ορίστηκε μία(1) και ως μέγιστος αριθμός επιλεγμένων γραμμών χίλιες(1000) για να μπορεί ο χρήστης να επιλέγει πάρα πολλές γραμμές από το listbox. Κάθε φορά που επιλέγει ο χρήστης γραμμές από το listbox αποθηκεύονται στην μεταβλητή handles.lines_selected. Η μεταβλητή αυτή χρησιμοποιείται στην συνάρτηση Callback του κουμπιού «Set Label To Category».

8.7 UiPanel με τίτλο SPIDER Control Panel

Στο SPIDER Control Panel υπάρχουν:

1. Ένα EditText με static text «Give Label: »
2. Ένα EditText με static text «Data Of Category:»
3. Ένα EditText με static text «Data Selected»
4. Το κουμπί %
5. Το κουμπί «Set Label To Category»
6. Το κουμπί «SVM Train SPIDER»
7. Το κουμπί «Reset»
8. Το κουμπί «SVM Test SPIDER»
9. Το κουμπί «2D Xsv Plot»
10. Το κουμπί «Reset Plot»

Για να μπορούμε να προσπελαύνουμε τον κώδικα της εργαλειοθήκης Spider **[7]**, από τις επιλογές της καρτέλας File του Matlab επιλέγουμε Set Path…, μετά πατάμε στο κουμπί Add with Subfolders…, βρίσκουμε τον φάκελο spider ο οποίος είναι

καταχωρημένος στην διαδρομή c:\spider και τέλος πατάμε το κουμπί save. Τώρα μπορούμε να έχουμε πρόσβαση σε όλους τους φακέλους αρχείων του φακέλου spider.

8.7.1. EditText με static text «Give Label:»

Ο χρήστης εισάγει τον αριθμό κατηγορίας(label) που επιθυμεί. Εάν δεν δοθεί η προβλεπόμενη τιμή(1 ή -1) τότε εμφανίζεται ένα προειδοποιητικό μήνυμα στην οθόνη(`Give a valid value for Y. Only [1 or -1] values are acceptable!`), ένα παράθυρο όπου υπενθυμίζει στον χρήστη τις επιθυμητές τιμές για να γίνει εφικτή η λειτουργία του αλγόριθμου SVM.

8.7.2 Edit Text με static text «Data Of Category:»

Ο χρήστης για να επιτύχει υψηλά ποσοστά επιτυχίας κατά την αυτόματη κατηγοριοποίηση στα πειράματα του πρέπει πρώτα απ' όλα να γνωρίζει:

- Πόσα συνολικά παραδείγματα απαρτίζουν το εκάστοτε πείραμα
- Πόσες κατηγορίες υπάρχουν
- Πόσα συνολικά παραδείγματα έχει η κάθε κατηγορία

Γνωρίζοντας τα πιο πάνω, θέτουμε στο Edit Text «Data From Category» τον συνολικό αριθμό παραδειγμάτων της εκάστοτε κατηγορίας ξεχωριστά. Όταν επιλέγουμε δεδομένα (listbox με static text «Data From Columns»)για να εκπαιδευτούν, ο συνολικός αριθμός των επιλεγμένων δεδομένων εμφανίζεται στο Edit Text «Data Selected:». Πατώντας το κουμπί % ο χρήστης βλέπει σε πόσα τοις εκατό (%) αντιστοιχούν τα δεδομένα που επέλεξε στο Edit Text «IS %». Ο χρήστης επιλέγει δεδομένα πατώντας το κουμπί % μέχρις ότου να εμφανιστεί στο Edit Text «IS %» 75% - 80%. Το 80% περίπου από τα παραδείγματα αυτά πρέπει να εκπαιδευτούν για να έχουμε στην δοκιμή(test) πολύ καλά αποτελέσματα αυτόματης κατηγοριοποίησης.

8.7.3 Edit Text με static text «Data Selected:»

Αυτό το Edit Text δημιουργήθηκε για ευκολία του χρήστη όπου αναγράφεται σε αυτό κάθε φορά ο αριθμός παραδειγμάτων που επέλεξε ο χρήστης κάθε φορά από το listbox «Data From Columns» αλλά και για να μπορούμε αυτόματα με το πάτημα του κουμπιού % να βλέπουμε την εκάστοτε φορά(ανάλογα με τις τιμές που έχουν τα Edit Text «Data Selected» και «Data Of Category») στο Edit Text που βρίσκεται αριστερά του κουμπιού % το ποσοστό παραδειγμάτων που επιλέχθηκαν για εκπαίδευση(train). Το επιθυμητό ποσοστό παραδειγμάτων προς εκπαίδευση είναι 75% - 80%.

8.7.4 Το κουμπί % (Callback)

```
data_selected=get(handles.edit_data_selected,'String');
data_selected_num=str2num(data_selected);
data_of_category=get(handles.edit_data_of_category,'String');
data_of_category_num=str2num(data_of_category);
percent=(data_selected_num*100)/data_of_category_num;
set(handles.edit_is_percent,'String',percent);
guidata(hObject,handles);
```

Επεξήγηση Κώδικα

- data_selected=get(handles.edit_data_selected,'String');
- data_selected_num=str2num(data_selected);

Παίρνουμε τον αριθμό παραδειγμάτων που επέλεξε ο χρήστης και τα αποθηκεύουμε στην μεταβλητή data_selected_num υπό μορφή αριθμού.

- data_of_category=get(handles.edit_data_of_category,'String');
- data_of_category_num=str2num(data_of_category);

Ο χρήστης αναγράφει τον συνολικό αριθμό της εκάστοτε κατηγορίας και ο αριθμός αυτός αποθηκεύεται στην μεταβλητή data_of_category_num υπό μορφή αριθμού.

- percent=(data_selected_num*100)/data_of_category_num;

- set(handles.edit_is_percent,'String',percent);

Αυτοματοποιούμε την μέθοδο των τριών και το αποτέλεσμα αποθηκεύεται στην μεταβλητή percent, ακολούθως εμφανίζεται το ποσοστό των παραδειγμάτων προς εκπαίδευση σε σύγκριση με τον αριθμό παραδειγμάτων της εκάστοτε κατηγορίας που κατέγραψε ο χρήστης στο Edit Text «Data Of Category».

8.7.5 Το κουμπί «Set Label To Category» (Callback)

```
c=get(handles.edit_label,'String');
c2=str2num(c);
if (c2==1 || c2==-1)
   n1=size(handles.LBDataN(handles.lines_selected,:));
   handles.oldrowselection=[handles.oldrowselection handles.lines_selected];
      for i=1:n1
          handles.Y_axis=[handles.Y_axis c2];
      end
   bb=handles.LBDataN(handles.oldrowselection,:);
   col=size(bb,2);
   handles.Y=handles.Y_axis';
   handles.X=bb;
   bb(:,col+1)=handles.Y_axis';
   set(handles.listbox_charasteristics_with_labels,'String',num2str(bb));
else msgbox('Give a valid value for Y. Only [-1 or 1] values are
acceptable','WARNING','warn');
end
guidata(hObject,handles)
```

Επεξήγηση κώδικα

- c=get(handles.edit_label,'String');
- c2=str2num(c);

Στην τοπική μεταβλητή c αποθηκεύουμε το περιεχόμενο που υπάρχει στο πεδίο του Edit Text με static text “Give Label”, επειδή όμως το περιεχόμενο αυτό είναι αλφαριθμητικό το μετατρέπουμε σε αριθμό με την εντολή str2num και αποθηκεύουμε τον αριθμό στην τοπική μεταβλητή c2 για να μπορούμε να κάνουμε σύγκριση των αριθμών που θα δίνουμε στο Edit Text χρησιμοποιώντας μία if. Οι μοναδικές τιμές που γίνονται αποδεκτές για να λειτουργήσει ο αλγόριθμος

γραμμικής κατηγοριοποίησης SVM είναι 1 και -1, εάν ο χρήστης δώσει οποιουσδήποτε άλλους αριθμούς εκτός του 1 και -1 τότε εμφανίζεται ένα message box που προτρέπει τον χρήστη να δώσει τιμές για labels 1 και -1.

- if (c2==1 || c2==-1)

Στην Callback συνάρτηση του κουμπιού Set Label To Category για να δοθεί τιμή στο label πρέπει να ισχύει η παραπάνω συνθήκη της if, δηλαδή η τιμή του label πρέπει να είναι 1 ή -1, εάν είναι τότε:

- n1=size(handles.LBDataN(handles.lines_selected,:));
- handles.oldrowselection=[handles.oldrowselection handles.lines_selected];

Η μεταβλητή handles.LBDataN περιέχει όλα τα στοιχεία(γραμμές και στήλες) του πίνακα που επιλέχτηκε από τον χρήστη. Παίρνουμε το μέγεθος των γραμμών που επέλεξε ο χρήστης και το αποθηκεύουμε στην τοπική μεταβλητή n1. Ο πίνακας handles.oldrowselection αρχικοποιείται με το που τρέχουμε την εφαρμογή μας ως ένας κενός πίνακας(function SVM_Classification_OpeningFcn) και αποθηκεύονται ως δεδομένα αυτά που έκανε επιλογή ο χρήστης από το listbox με static text “Data From Columns”.

- for i=1:n1
- handles.Y_axis=[handles.Y_axis c2];
- end

Με το που τρέχουμε την εφαρμογή μας αρχικοποιούμε ακόμα ένα πίνακα τον handles.Y_axis[] με κενό(function SVM_Classification_OpeningFcn) και με την επαναληπτική δομή for γεμίζει ο πίνακας αυτός με τόσα 1 ή -1 ανάλογα με το τί τιμή έχει η τοπική μεταβλητή n1(αριθμός γραμμών που επιλέχτηκαν από τον χρήστη).

- bb=handles.LBDataN(handles.oldrowselection,:);
- col=size(bb,2);

Αποθηκεύουμε τις επιλεγμένες γραμμές με τις στήλες τους στην μεταβλητή bb και μετά με την εντολή size βρίσκουμε το συνολικό πλήθος στηλών των επιλεγμένων γραμμών και το αποθηκεύουμε στην μεταβλητή col, αυτό το κάνουμε γιατί θα

προσθέσουμε ακόμα μια στήλη στις επιλεγμένες γραμμές όπου είναι το εκάστοτε label(1 ή -1) που επιλέγει ο χρήστης.

- handles.Y=handles.Y_axis';
- handles.X=bb;
- bb(:,col+1)=handles.Y_axis';

Αντιστρέφουμε τον πίνακα handles.Y_axis που προέκυψε γράφοντας ένα ‘ στο τέλος του πίνακα και τον αποθηκεύουμε στην μεταβλητή handles.Y. Οι γραμμές με τα χαρακτηριστικά(στοιχεία, πεδία των γραμμών) τους αποθηκεύονται στην μεταβλητή handles.X. Ως τελευταία στήλη στη μεταβλητή bb βάζουμε τον ανάστροφο πίνακα handles.Y_axis’.

Τις μεταβλητές handles.Y και handles.X τις χρησιμοποιούμε στον αλγόριθμο κατηγοριοποίησης SVM με το πάτημα του κουμπιού SVM Train SPIDER.

handles.X→ Είναι τα δεδομένα εισόδου (εκπαίδευσης).

handles.Y→ Είναι οι εκάστοτε ετικέτες που όρισε ο χρήστης για το πείραμα.

- set(handles.listbox_charasteristics_with_labels,'String',num2str(bb));

Θέτουμε στο listbox με static text “Labeled Characteristics” τις γραμμές των χαρακτηριστικών μαζί με τα εκάστοτε labels μετατρέποντας τα δεδομένα αυτά σε αλφαριθμητικά αφού στο listbox τα δεδομένα που εισάγονται πρέπει να είναι αλφαριθμητικά χρησιμοποιώντας την εντολή num2str για να γίνουν και τα εκάστοτε labels αλφαριθμητικά.

- else msgbox('Give a valid value for Y. Only [-1 or 1] values are acceptable','WARNING','warn');
- end

Εάν ως label δώσουμε οποιοδήποτε άλλο αριθμό εκτός του 1 ή του -1 μας εμφανίζεται ένα message box με εικονίδιο warn, τίτλο WARNING και ως μήνυμα μας λέει να δώσουμε έγκυρες τιμές[1 ή -1].

8.7.6 Το κουμπί «SVM Train SPIDER» (Callback)

```
handles.d=data(handles.X,handles.Y);
[handles.tr handles.a]=train(svm,handles.d);
memo=get(handles.listbox_charasteristics_with_labels,'String');
set(handles.listbox_memo,'String',memo);
guidata(hObject,handles);
```

Επεξήγηση κώδικα

Πώς γίνεται η εκπαίδευση(train) του αλγόριθμου SVM;

Βήμα 1: Προετοιμάζουμε τα δεδομένα μας(γραμμές χαρακτηριστικών) αποθηκεύοντας τα σε ένα πίνακα(handles.X) και αποθηκεύουμε τα labels σε ένα άλλο πίνακα(handles.Y).

Βήμα 2: Δημιουργούμε ένα αντικείμενο δεδομένων handles.d.

- handles.d=data(handles.X,handles.Y)

Βήμα 3**:** Εκπαιδεύουμε τον αλγόριθμο SVM χρησιμοποιώντας την συνάρτηση train της εργαλειοθήκης SPIDER **[7]**.

- [handles.tr handles.a]=train(svm,handles.d)

Μέσα στην συνάρτηση train βάζουμε δύο(2) ορίζματα τον αλγόριθμο γραμμικής ταξινόμησης SVM και το αντικείμενο δεδομένων που προέκυψε από την συνάρτηση data (handles.d), προκύπτουν ένα αντικείμενο δεδομένων handles.tr όπου είναι οι προβλέψεις (περιέχει τα παραδείγματα που εκπαιδεύτηκαν, τις ετικέτες που όρισε ο χρήστης, το πεδίο findex, το πεδίο index και το πεδίο algorithm) και ένα αντικείμενο svm το handles.a. Handles.a είναι το μοντέλο που εκπαιδεύτηκε, είναι ένα αντικείμενο αλγορίθμου όπου σε αυτή την περίπτωση είναι ένα αντικείμενο svm, με βάση αυτό το μοντέλο που δημιουργήθηκε θα γίνει η μετέπειτα αυτόματη ταξινόμηση(test).

Μπορούμε να έχουμε πρόσβαση στις προβλέψεις γράφοντας handles.tr.X ή handles.tr.Y όπου το X και το Y είναι μέλη του αντικειμένου handles.tr.

struct(object)→ Για να δούμε τα μέλη ενός οποιουδήποτε αντικειμένου χρησιμοποιούμε την εντολή struct(object) π.χ. γράφοντας struct(handles.tr).

Παράδειγμα: Γράφοντας την εντολή struct(handles.tr) βλέπουμε τα πεδία που αποτελείται το αντικείμενο handles.tr. Μετά εμείς ανάλογα με το πιο πεδίο του αντικειμένου θέλουμε να εξετάσουμε γράφουμε απλά: struct(handles.tr.πεδίο).

- handles.d=data(handles.X,handles.Y);

data συνάρτηση→ Υπάρχει στην διαδρομή c:\spider\basic\@data.

Σύνταξη της συνάρτησης data→ d = data(X,Y)

handles.d→ Δημιουργείτε ένα αντικείμενο δεδομένων{data object 1-by-1} όπου περιέχει τα δεδομένα(γραμμές χαρακτηριστικών) των δύο(2) κατηγοριών με τα εκάστοτε labels τους(1 και -1).

Χρησιμοποιούμε την εντολή struct(object) και το αντικείμενο που θέλουμε για να εξετάσουμε.

Παράδειγμα:

Επιλέγοντας τρείς(3) γραμμές για την κατηγορία 1 και δύο(2) γραμμές για την κατηγορία -1 και πατώντας το κουμπί SVM Train SPIDER όπου μέσα στην συνάρτηση Callback του κουμπιού χρησιμοποιήσαμε την γραμμή κώδικα struct(handles.d) και εμφανίζεται:

```
%struct(handles.d)---> X: [5x2 double]
%                      Y: [5x1 double]
%                      index: [1 2 3 4 5]
%                      findex: [1 2]
%                      algorithm: [1x1 algorithm]

        %disp(handles.d.X)--->1.8500   85.0000
        %                      1.8500   82.0000
        %                      1.7800   78.0000
        %                      1.5000   85.0000
        %                      1.5000   82.0000

%disp(handles.d.Y)---> [1 1 1 -1 -1]'
%disp(handles.d.index)---> 1     2     3     4     5
%disp(handles.d.findex)---> 1     2
%disp(handles.d.algorithm) ---> data
```

- [handles.tr handles.a]=train(svm,handles.d);

Παράδειγμα:

Επιλέγοντας τρείς(3) γραμμές για την κατηγορία 1 και δύο γραμμές για την κατηγορία -1 και πατώντας το κουμπί SVM Train SPIDER όπου μέσα στην συνάρτηση Callback του κουμπιού υπάρχει η γραμμή κώδικα struct(handles.tr) εμφανίζεται:

```
%disp(struct(handles.tr))
                    %X: [5x1 double]
                    %Y: [5x1 double]
                    %index: [1 2 3 4 5]
                    %findex: 1
                    %algorithm: [1x1 algorithm]
```

Για να δούμε το περιεχόμενου του μέλους X του αντικείμενου handles.tr: X:[5x1 double] γράφουμε disp(handles.tr.X) και το αποτέλεσμα που εμφανίζεται στο command prompt είναι [1 1 1 -1 -1]'. Για να δούμε το περιεχόμενο του μέλους Y του αντικειμένου handles.tr γράφουμε disp(handles.tr.Y) και το αποτέλεσμα είναι [1 1 1 -1 -1]'.

Το περιεχόμενο του μέλους index είναι [1 2 3 4 5].

Το περιεχόμενο του μέλους findex είναι 1 και τέλος το περιεχόμενο του μέλους algorithm είναι data -> svm kernel linear.

```
Struct(handles.a)→   child: [1x1 kernel]
%                    C: Inf
%                    ridge: 1.0000e-013
%                    balanced_ridge: 0
%                    nu: 0
%                    optimizer: 'default'
%                    alpha_cutoff: -1
%                    alpha: [5x1 double]
%                    b0: -5.2500
%                    Xsv: [1x1 data]
%                    nob: 0
%                    algorithm: [1x1 algorithm]
```

- memo=get(handles.listbox_charasteristics_with_labels,'String');
- set(handles.listbox_memo,'String',memo);

Κάνουμε μια αντιγραφή των δεδομένων προς εκπαίδευση(train) που έβαλε ο χρήστης στο listbox με static text “Memo Listbox” ούτως ώστε ο χρήστης να μην εισάγει ξανά τα ίδια δεδομένα για δοκιμή(test).

8.7.7 Το κουμπί «Reset» (Callback)

```
handles.Y_axis=[];
handles.oldrowselection=[];
set(handles.listbox_charasteristics_with_labels,'String','');
set(handles.listbox_labels,'String','');
```

Επεξήγηση κώδικα

Πρέπει να γίνει σαφές ότι για να γίνει δοκιμή(test) από τον αλγόριθμο SVM πρέπει αφού τελειώσουμε με την εισαγωγή δεδομένων προς εκπαίδευση(train) και αφού πατήσουμε το κουμπί SVM Train SPIDER και γίνει η κατηγοριοποίηση των δεδομένων, όταν θα πάμε να κάνουμε δοκιμή(test) πρέπει να πατήσουμε το κουμπί Reset για να αδειάσει το listbox με static text “Labeled Characteristics”, το listbox με static text “Labels”, να αδειάσει ο πίνακας handles.Y_axis[] καθώς επίσης και για να αδειάσει και ο πίνακας handles.oldrowselection[] γιατί τα νέα δεδομένα για δοκιμή(test) χρησιμοποιούν κοινούς πόρους(listbox με static text “Labeled Characteristics”, τον πίνακα handles.Y_axis[] και τον πίνακα handles.oldrowselection[]).

- handles.Y_axis=[];
- handles.oldrowselection=[];
- set(handles.listbox_charasteristics_with_labels,'String','');
- set(handles.listbox_labels,'String','');

Αρχικοποιούμε τους πίνακες handles.Y_axis[] και handles.oldrowselection[] αδειάζοντάς τους και αδειάζουμε το listbox με static text “Labeled Characteristics” και το listbox με static text “Labels” για να μπορούμε είτε να ξαναεισάγουμε

δεδομένα προς εκπαίδευση από τυχόν λάθος εισαγωγή από τον χρήστη καθώς επίσης για να μπορέσουμε να δώσουμε τα νέα δεδομένα για δοκιμή(test) ή για πραγματοποίηση νέου πειράματος.

8.7.8 Το κουμπί «SVM Test SPIDER»

```
handles.dtest=data(handles.X,handles.Y);
handles.tst=test(handles.a,handles.dtest);
set(handles.listbox8,'String',handles.tst.X)
guidata(hObject,handles)
```

- handles.dtest=data(handles.X,handles.Y);

Χρησιμοποιώντας την έτοιμη συνάρτηση data από την εργαλειοθήκη SPIDER **[7]** δημιουργούμε ένα αντικείμενο δεδομένων handles.dtest{data object 1-by-1}. Στην μεταβλητή handles.X(γραμμές χαρακτηριστικών) τώρα υπάρχουν τα νέα δεδομένα τα οποία θα κληθεί ο αλγόριθμος να κατηγοριοποιήσει αυτόματα και στην μεταβλητή handles.Y είναι τα τυχαία labels που έδωσε ο χρήστης(όλα 1 ή -1).

- handles.tst=test(handles.a,handles.dtest);

Στην συνάρτηση test βάζουμε δύο(2) ορίζματα το αντικείμενο αλγόριθμου svm(handles.a) που προέκυψε από την εκπαίδευση και το αντικείμενο δεδομένων handles.dtest όπου είναι τα νέα δεδομένα στα οποία θα γίνει η δοκιμή(test). Τα αποτελέσματα της συνάρτησης test αποθηκεύονται στην μεταβλητή handles.tst όπου είναι ένα αντικείμενο δεδομένων ενός γραμμικού kernel svm.

handles.tst→ data->svm linear kernel

```
%struct(handles.tst)
%           X: [3x1 double]
%           Y: [3x1 double]
%           index: [1 2 3]
%           findex: 1
%           algorithm: [1x1 algorithm]
handles.tst.Y%---> [1 1 -1]' einai ta tixaia labels pou edwse o xrhsths
handles.tst.X%---> [-1 -1 1]' einai ta labels pou orise o algoritmos ws swsta
```

Το μέλος X του αντικειμένου handles.tst είναι αυτό που μας ενδιαφέρει περισσότερο απ'όλα γιατί σε αυτό αποθηκεύτηκαν τα "σωστά" labels των νέων «άγνωστων» δεδομένων(γραμμές χαρακτηριστικών). Πρέπει να σημειωθεί ότι για να δουλέψει η συνάρτηση test πρέπει ο χρήστης να δώσει labels στα νέα δεδομένα(γραμμές χαρακτηριστικών (σε όλα 1 ή -1)) αλλιώς η συνάρτηση test δεν δουλεύει. Τα σωστά labels(handles.tst.X) εμφανίζονται στο listbox με static text "Labels" όπου μπορούμε να δούμε βάση των κανόνων κατηγοριοποίησης(κανόνες) με τους οποίους όρισε ο χρήστης αν έγινε σωστή η αυτόματη κατηγοριοποίηση των νέων «άγνωστων» δεδομένων στις κατηγορίες 1 και -1.

8.7.9 Το κουμπί «2D Xsv Plot» (Callback)

plot(handles.a);

Μόνο όταν τα παραδείγματα που εκπαιδεύτηκαν είναι 2 – διάστατα μπορούμε να χρησιμοποιήσουμε αυτό το κουμπί. Ο λόγος είναι ότι υπάρχει περιορισμός από τον SPIDER SVM αλγόριθμο **[7]**. Σε διανύσματα με διαστάσεις περισσότερες των 2 διαστάσεων μας εμφανίζεται σφάλμα στο command prompt του matlab αν πατήσουμε αυτό το κουμπί.

Περισσότερο το χρησιμοποιούμε για να δούμε και οπτικά την λειτουργία του αλγόριθμου δείχνοντας μας με έντονους κύκλους τα διανύσματα υποστήριξης όπου με βάσει αυτών θα γίνει η αυτόματη κατηγοριοποίηση των «άγνωστων» εναπομείναντων παραδειγμάτων προς δοκιμή(test).

Αυτό το κουμπί γίνεται ενεργό μόνο όταν το πλήθος των στηλών ισούται ακριβώς με δύο(2), σε οποιαδήποτε άλλη περίπτωση είναι απενεργοποιημένο.

Επεξήγηση Κώδικα

- plot(handles.a);

Εμφανίζονται όλα τα παραδείγματα που εκπαιδεύτηκαν όπου με έντονο κύκλο εμφανίζονται τα διανύσματα υποστήριξης στο component axes1.

8.7.10 Το κουμπί «Reset Plot» (Callback)

```
cla(handles.axes1,'reset');
```

Επεξήγηση Κώδικα

- cla(handles.axes1,'reset');

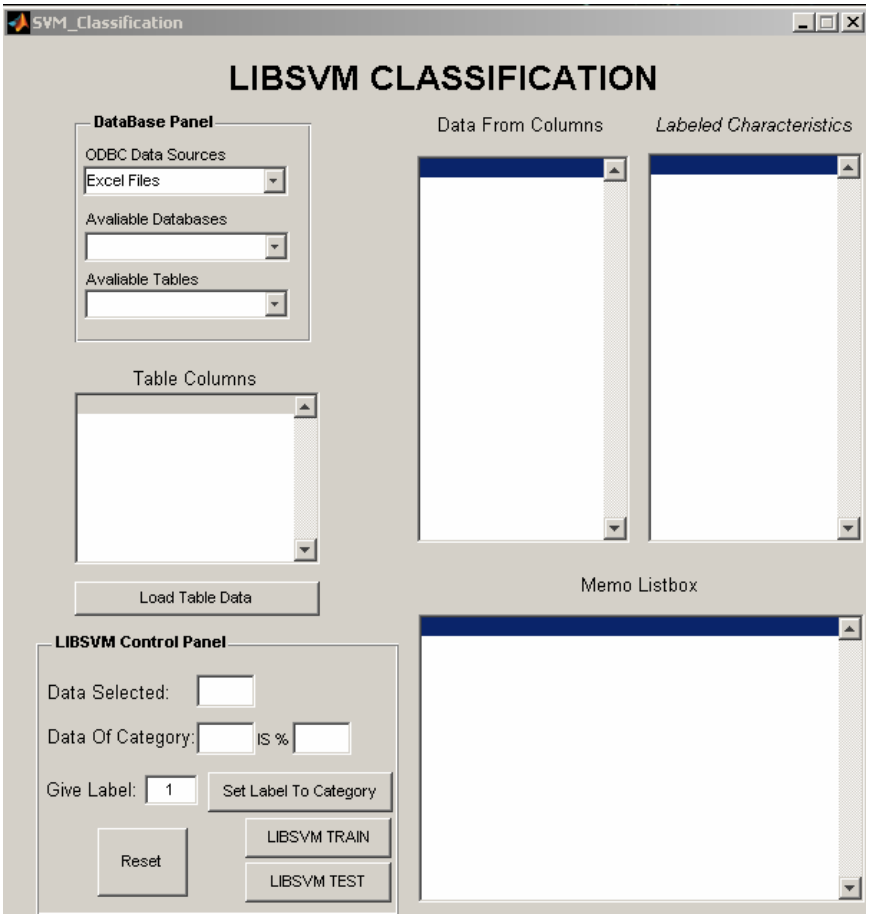
Αδειάζετε το component axes1 για να μπορούμε ξανά να αναπαραστήσουμε γραφικά νέα δεδομένα.

ΜΕΡΟΣ Β

Υλοποίηση εφαρμογής και επεξήγηση κώδικα χρησιμοποιώντας την LIBSVM βιβλιοθήκη

Για ευνόητους λόγους χρησιμοποιήσαμε την ήδη έτοιμη εφαρμογή που δημιουργήσαμε για τον αλγόριθμο SPIDER **[7]** αναπροσαρμόζοντας τα SVM_Classification.fig και SVM_Classification.m αρχεία για την πραγματοποίηση πειραμάτων με την βιβλιοθήκη LIBSVM. Τα σετ δεδομένων που χρησιμοποιούνται στα πειράματα με WEKA, SPIDER και LIBSVM είναι πανομοιότυπα. Σκοπός που δουλεύουμε με αυτό το μήκος κύματος λογικής είναι για να δούμε στο τέλος όταν θα διεξάγουμε συμπεράσματα ποιος αλγόριθμος είναι ο καλύτερος αφού όλοι οι αλγόριθμοι είχαν και στο στάδιο της εκπαίδευσης και στο στάδιο δοκιμής ακριβώς τα ίδια πρότυπα.

Η εφαρμογή μας έχει την εξής μορφή:



Κρατήσαμε όλα τα components της εφαρμογής του SPIDER εκτός από:

- Axes1
- Pushbutton_2D_Xsv_Plot
- Pushbutton_reset_plot
- Listbox_labels
- Pushbutton_SVM_Train_SPIDER
- Pushbutton_SVM_Test_SPIDER

Και δημιουργήθηκαν τα κουμπιά:

- Pushbutton_LIBSVM_TRAIN
- Pushbutton_LIBSVM_TEST

8.8.1 Το κουμπί «LIBSVM TRAIN» (Callback)

```
tic
parameters = ('-s 0 -t 0 ');
handles.model = svmtrain(handles.Y, handles.X, parameters);
fprintf( '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')
fprintf('rho is the bias term in the decision function sgn(w^Tx - rho)\n')
fprintf('nu-svm is a somewhat equivalent form of C-SVM where C is replaced by nu. nu simply shows the corresponding parameter\n')
fprintf('obj is the optimal objective value of the dual SVM problem\n')
fprintf('nSV and nBSV are number of support vectors and bounded support vectors (i.e., alpha_i = C)\n')
fprintf( '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')
% i sintaxi full(handles.model.SVs) mas einai xrisimi gia na topotheti:
%(1,1) (1,2)
 %(2,1) (2,2)
%diladi emfanizei ola ta apotelesmata omiomorfa, ola ta xaraktiristika p.x  tou 1ou support vector se mia grammi klp
sv = full(handles.model.SVs);
memo=get(handles.listbox_charasteristics_with_labels,'String');
set(handles.listbox_memo,'String',memo);

%xrisimes grammes kodika brikame sto FAQ tis istoselidas LIBSVM
w = handles.model.SVs' * handles.model.sv_coef;
b = -handles.model.rho;
if handles.model.Label(1) == -1    %Label(1)-->einai to proto label pou dinetai se dedomena gia train, Label(2)-->to deutero label pou dothike
  w = -w;
```

```
  b = -b;
end

sprintf('The support vectors (SVs) in the current experiment are:\n')
disp(sv)
fprintf( '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')
sprintf('The optimized vector W is:\n')
disp(w')
fprintf( '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')
toc
```

Επεξήγηση κώδικα

Πώς γίνεται η εκπαίδευση (train) της βιβλιοθήκης LIBSVM;

Βήμα 1: Προετοιμάζουμε τα δεδομένα εκπαίδευσης σε ένα πίνακα handles.X και αποθηκεύουμε τις ετικέτες στον πίνακα handles.Y. Ως παραμέτρους, λόγω του ότι η βιβλιοθήκη LIBSVM έχει την ευελιξία να συγκεκριμενοποιήσουμε τις απαιτήσεις μας έχοντας στην διάθεση μας διαφόρων τύπων αλγόριθμων π.χ. C-SVC και διαφόρων τύπων πυρήνων kernel π.χ. γραμμικό.

Βήμα 2: Εκπαιδεύουμε τον γραμμικό αλγόριθμο της βιβλιοθήκης LIBSVM κάνοντας χρήση την συνάρτηση svmtrain.

Βήμα 3: Δημιουργείται το μοντέλο πρόβλεψης handles.model όπου σ' αυτό υπάρχουν τα διανύσματα υποστήριξης του εκάστοτε πειράματος εκπαίδευσης που πραγματοποιήθηκε.

- [tic] [εντολές] [toc]

Η δομή tic toc χρησιμοποιείται για να δούμε πόσος χρόνος χρειάστηκε για να έρθει εις πέρας η διαδικασία εκπαιδεύσης, γι' αυτό η εντολή tic βρίσκεται στην πρώτη γραμμή κώδικα της εκπαίδευσης και στην τελευταία γραμμή κώδικα της εκπαίδευσης γράψαμε την εντολή toc.

- parameters = ('-s 0 -t 0 ');
- handles.model = svmtrain(handles.Y, handles.X, parameters);

Στην μεταβλητή parameters ως πρώτο όρισμα επιλέγουμε τον τύπο αλγορίθμου, βάλαμε –s 0 δηλαδή χρησιμοποιούμε τον αλγόριθμο γραμμικής ταξινόμησης C-SVC.

Ως δεύτερο όρισμα επιλέγουμε τον τύπο του πυρήνα kernel που θέλουμε, εμείς επιλέξαμε τον γραμμικό kernel (u'*v).

Στην συνέχεια το μοντέλο που εκπαιδεύτηκε (handles.model) έχει την εξής δομή:

```
%disp(handles.model)
% Parameters: [5x1 double]
%      nr_class: 2
%       totalSV: 5
%           rho: 349.0000
%         Label: [2x1 double]
%         ProbA: []
%         ProbB: []
%           nSV: [2x1 double]
%       sv_coef: [5x1 double]
%           SVs: [5x2 double]
```

<u>Οι διάφορες τιμές που υπάρχουν στις πιο πάνω σταθερές μεταβλητές ποικίλουν αναλόγως της εκπαίδευσης που τυγχάνουν.</u>

<u>Αναλυτικά:</u>

```
% disp(handles.model.Parameters)
% 0          [είναι το –s δηλαδή επιλέγεται ο τύπος αλγορίθμου {C-SVC}]
% 0          [είναι το –t δηλαδή επιλέγεται ο τύπος πυρήνα kernel {u'*v}]
% 3.0000     [είναι το –d  δηλαδή ο βαθμός της kernel function]
% 0.5000     [είναι το –n δηλαδή nu του nu-SVC,one-class SVM και nu-SVR]
% 0          [είναι  το –b δηλαδή εκτιμήσεις πιθανοτήτων εκπαίδευσης είτε SVC
             είτε του SVR μοντέλου, 0 ή 1]
```

Εξορισμού (by default) από την βιβλιοθήκη LIBSVM δίνονται στο –d η τιμή 3, στο –n η τιμή 0.5 και στο –b η τιμή 0, τιμές οι οποίες δεν επηρεάζουν τα αποτελέσματα των πειραμάτων μας. Οι μόνες μεταβλητές ενδιαφέρουν εμάς είναι η –s και η –t.

Πιο αναλυτικά μπορούμε να δούμε μέσα στον φάκελο "LIBSVM FIGURE" το αρχείο κειμένου "parameter options.txt".

Το δεύτερο πεδίο της δομής handles.model είναι το nr_class το οποίο υποδουλώνει το πλήθος των κλάσεων. Το πεδίο totalSV υποδουλώνει τον συνολικό

αριθμό των διανυσμάτων υποστήριξης. Το πεδίο rho σημαίνει το ίδιο πράγμα με το b (bias) δηλαδή είναι ενας αριθμός πόλωσης. Το πεδίο Label έχει αποθηκευμένες τις ετικέτες που έδωσε ο χρήστης δηλαδή 1 και -1. Τα επόμενα δύο πεδία μένουν κενά εάν –b 0 ή αν έχουμε να κάνουμε με SVM μιας κλάσης, αφού εξορισμού από τον αλγόριθμο δίνεται η τιμή 0 το πεδίο ProbA και ProbB μένουν κενά. Το πεδίο nSV υποδουλώνει τον αριθμό των διανυσμάτων υποστήριξης για κάθε κλάση. Το πεδίο sv_coef είναι οι συντελεστές (coefficients) για SVs σε συναρτήσεις απόφασης.

Τέλος το πεδίο SVs περιέχει όλα τα διανύσματα υποστήριξης.

- sv = full(handles.model.SVs);

Αποθηκεύονται στην τοπική μεταβλητή sv όλα τα διανύσματα υποστήριξης του μοντέλου.

Χρησιμοποιείται η σύνταξη full() για να εμφανίζονται στην οθόνη ομοιόμορφα τα διανύσματα π.χ. όλα τα στοιχεία του 1^{ου} διανύσματος σε μια γραμμή κλπ.

Εάν δεν χρησιμοποιήσουμε την δομή full() έχουμε το παρακάτω αποτέλεσμα στην οθόνη:

```
%   (1,1)    175
%   (2,1)    175
%   (3,1)    175
%   (4,1)    174
%   (5,1)    174
%   (1,2)     70
%   (2,2)     66
%   (3,2)     68
%   (4,2)     70
%   (5,2)     67
```

- memo=get(handles.listbox_charasteristics_with_labels,'String');
- set(handles.listbox_memo,'String',memo);

Αντιγράφουμε στο listbox memo όλα τα πρότυπα που εκπαιδεύτηκαν για να μην χρειάζεται να θυμόμαστε όλα τα πρότυπα που εκπαιδεύτηκαν.

- w = handles.model.SVs' * handles.model.sv_coef;
- b = -handles.model.rho;
- if handles.model.Label(1) == -1   %Label(1)-->einai to proto label pou dinetai se dedomena gia train, Label(2)-->to deutero label pou dothike
- w = -w;
- b = -b;
- end

Η παραπάνω σύνταξη κώδικα υπάρχει στην ιστοσελίδα που βρήκαμε την εργαλειοθήκη LIBSVM στο μέρος FAQ. Το όφελος μας είναι ότι μπορούμε να δούμε το βέλτιστο διάνυσμα w_0 και τον βέλτιστο αριθμό πόλωσης (bias).

- sprintf('The support vectors (SVs) in the current experiment are:\n')
- disp(sv)

Τυπώνουμε στην οθόνη τα διανύσματα υποστήριξης

- fprintf('~~~~~~\n')
- sprintf('The optimized vector W is:\n')
- disp(w')
- fprintf('~~~~~~\n')

Τυπώνουμε στην οθόνη το βέλτιστο διάνυσμα βαρών w_0.

8.8.2 Το κουμπί «LIBSVM TEST» (Callback)

```
[predicted_label, accuracy, decision_values] = svmpredict (handles.Y, handles.X,
handles.model);
%akribia automatis taxinomisis
%disp(accuracy)
%oi times ton ekastote paradeigmaton katopin mathimatikis praxis (arnitikoi
arithmoi gia -1 klasi, thetikoi arithmoi gia +1 klasi)
%disp(decision_values)
%ta labels pou problepse o algorithos os sosta
disp(predicted_label)
```

Επεξήγηση κώδικα:

- [predicted_label, accuracy, decision_values] = svmpredict (handles.Y, handles.X, handles.model);

Αφού δημιουργήθηκε το βέλτιστο μοντέλο ταξινόμησης handles.model ερχόμαστε τώρα να εφαρμόσουμε στην συνάρτηση svmpredict τα νέα «άγνωστα» δεδομένα για αυτόματη κατηγοριοποίηση.

Ως ορίσματα η συνάρτηση svmpredict παίρνει τις ετικέτες των δεδομένων δοκιμής, τα δεδομένα σε μορφή διανυσμάτων και το μοντέλο πρόβλεψης που δημιουργήθηκε κατά την εκπαίδευση.

Ως έξοδο η συνάρτηση svmpredict μας βγάζει τις προβλέψεις που έκανε ο αλγόριθμος υπό μορφή ετικετών (1 και -1) στο πεδίο predicted_label, την ποσοστιαία ακρίβεια πρόβλεψης του αλγορίθμου (δεν το χρησιμοποιούμε γιατί η ποσοστιαία ακρίβεια εμφανίζεται στην οθόνη μαζί με τα predicted_label) και στο πεδίο decision_values εμφανίζονται τα διανύσματα δοκιμής ως αριθμοί με ανάλογο πρόσημο βάση της κλάσης που έτυχε.

Να σταθούμε λίγο στο πεδίο decision values, αφού ορίσαμε εμείς γραμμική κατηγοριοποίηση κάνοντας χρήση του γραμμικού τύπου πυρήνα ($u' * v$) ως αποτέλεσμα αυτής της μαθηματικής πράξης προκύπτει ένας αριθμός για το εκάστοτε διάνυσμα δοκιμής, αυτός ο αριθμός με θετικό πρόσημο εάν έτυχε ετικέτας +1 και αρνητικό πρόσημο εάν έτυχε ετικέτας -1.

Κεφάλαιο 9

Πειράματα Κατηγοριοποίησης

Σε αυτό το κεφάλαιο θα δούμε όλα τα σετ δεδομένων που περιλαμβάνονται στα πειράματα μας και τα αποτελέσματα όλων των πειραμάτων που πραγματοποιήθηκαν.

9.1 Σετ Δεδομένων (Iris, Wine, Glass, Breast Cancer Wisconsin (Original) και Παρέλαση Τελειοφοίτων)

Για να δούμε και στην πράξη την χρησιμότητα του γραμμικού αλγόριθμου δυαδικής ταξινόμησης SVM εκτελούμε στο πρόγραμμα SVM Classification.fig πέντε (5) πειράματα τα οποία πειράματα έχουν διαφορετικό πλήθος χαρακτηριστικών (attributes) μεταξύ τους καθώς και διαφορετικό πλήθος παραδειγμάτων (instances). Και στα πέντε (5) πειράματα όλα τα δεδομένα είναι της μορφής πραγματικών αριθμών (integer, float).

Βρήκαμε τα ακόλουθα δεδομένα από την ιστοσελίδα του πανεπιστημίου UCI http://archive.ics.uci.edu/ml/datasets.html

- Wine σετ δεδομένων
- Glass σετ δεδομένων
- Breast Cancer Wisconsin (Original) σετ δεδομένων

Υπάρχουν διαφόρων ειδών δεδομένα στην ιστοσελίδα του πανεπιστημίου UCI εμείς επιλέξαμε αυτά που είναι για μας κατάλληλα για ταξινόμηση και τα δεδομένα να είναι της μορφής πραγματικών αριθμών (integer, float).

Τα δεδομένα Iris τα βρήκαμε από την ελεύθερη εγκυκλοπαίδεια Wikipedia και τέλος δημιουργήσαμε το πέμπτο και τελευταίο σετ δεδομένων το οποίο έχει την ονομασία PARELASI_TELIOFITON που σκοπός του είναι να δούμε μέσο αυτού και σε γραφική παράσταση το πώς τοποθετούνται τα δεδομένα που εκπαιδεύτηκαν και να δούμε επίσης τις ΜΔΥ με έντονους κύκλους στην γραφική παράσταση.

9.1.1 Iris data (150 δεδομένα με 4 - διάστατα ανύσματα)

Το Fisher Iris σετ δεδομένων περιέχει τρία(3) διαφορετικά είδη λουλουδιών, το είδος Iris setosa το είδος Iris virginica και το είδος Iris versicolor. Τα τρία(3) αυτά διαφορετικά είδη λουλουδιών έχουν από πενήντα(50) παραδείγματα το κάθε είδος και όλα τα παραδείγματα περιγράφονται από τέσσερα κοινά χαρακτηριστικά. Τα κοινά χαρακτηριστικά είναι το μήκος και το πλάτος του sepal καθώς επίσης το μήκος και το πλάτος του petal **[6]**. Αυτές οι εκατό πενήντα(150) μετρήσεις έγιναν από τον Ronald A. Fisher(1936) και χρησιμοποιούνται ευρύτατα ως μία τυπική δοκιμή από πολλούς αλγόριθμους κατηγοριοποίησης.

Πίνακας Iris δεδομένων

Είδη λουλουδιών	Παραδείγματα εκπαίδευσης	Παραδείγματα δοκιμής
Iris setosa: 1-50	1-40 [80%]	41-50 [20%]
Iris virginica: 51-100	51-90 [80%]	91-100 [20%]
Iris versicolor: 101-150	101-140 [80%]	141-150 [20%]

Αφού έχουμε τρείς(3) κατηγορίες λουλουδιών, έχουμε τρείς(3) δυνατούς συνδιασμούς (τρεία ξεχωριστά πειράματα) μεταξύ των κατηγοριών(K1-K2, K1-K3 και K2-K3) λόγω του ότι ο αλγόριθμος που χρησιμοποιούμε μόνο με ανα δύο(2) κατηγορίες κάθε φορά μπορεί να δουλέψει. Από την κάθε κατηγορία χρησιμοποιούμε το 80%(δηλαδή τα πρώτα σαράντα(40) παραδείγματα κάθε κατηγορίας) των δεδομένων για εκπαίδευση του αλγόριθμου και το υπόλοιπο 20% (τα υπόλοιπα 10 παραδείγματα κάθε κατηγορίας) για δοκιμή. Η βάση δεδομένων IRIS_DATABASE περιέχει τον πίνακα IRIS_TABLE όπου σε αυτόν τον πίνακα είναι αποθηκευμένα τα δεδομένα ως εξής: Τα πρώτα πενήντα(50) παραδείγματα

ανήκουν στην κατηγορία Iris setosa, από το 51° παράδειγμα εώς το 100° παράδειγμα ανήκουν στην κατηγορία Iris virginica και από το 101° παράδειγμα μέχρι το 150° ανήκουν στην κατηγορία Iris versicolor.

9.1.2 Glass data (214 δεδομένα με 10 – διάστατα ανύσματα)

Το σετ δεδομένων glass συλλέχτηκε από τον B. German και δόθηκαν από την Vida Spiehler (01/09/1987) στην ιστοσελίδα του πανεπιστημίου UCI. Σε αυτό το σετ όλα τα δεδομένα συνολικά είναι διακόσια δέκα τέσσερα (214) και αυτά ανήκουν σε επτά (7) διαφορετικά είδη γυαλιών τα οποία είναι:

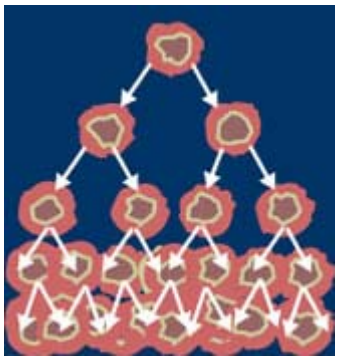
1. Building windows float processed
2. Building windows non float processed
3. Vehicle windows float processed
4. Vehicle windows non float processed (Δεν υπάρχουν σε αυτή τη Βάση Δεδομένων)
5. Containers
6. Tableware
7. Headlamps

Όλες οι παραπάνω κατηγορίες χωρίζονται σε δύο (2) μεγάλες κατηγορίες οι οποίες είναι η window glass (περιέχει τις κατηγορίες 1,2 και 3) και η άλλη είναι η non window glass (περιέχει τις κατηγορίες 5, 6 και 7).

Πίνακας Glass δεδομένων

Είδη γυαλιών	Παραδείγματα εκπαίδευσης	Παραδείγματα δοκιμής
Window glass: 163 (1-163)	1 – 130 (130) [79,7546%]	131 – 163 (33) [20,2454%]
Non-Window glass: 51 (164- 214)	164 – 205 (41) [80,3922%]	205 – 214 (10) [19,6078%]

9.1.3 Breast Cancer Wisconsin (Original) data (699 δεδομένα με 10 - διάστατα ανύσματα)



Το σετ δεδομένων Wisconsin Breast Cancer Database συλλέχτηκε από τον Dr. William H. Wolberg (Φυσικός) στις 08/01/1991 και δόθηκε από την Olvi Mangasarian στο πανεπιστήμιο UCI. Σε αυτό το σετ όλα τα δεδομένα συνολικά είναι εξακόσια ενενήντα εννέα (699) και χωρίζονται σε δύο (2) κατηγορίες. Στην 1^{η} κατηγορία είναι όλα τα δεδομένα όπου η μορφή καρκίνου του μαστού είναι κακοήθης (Maligrant)και στην 2^{η} κατηγορία είναι όλα τα δεδομένα όπου η μορφή του καρκίνου του μαστού είναι καλοήθης (Benign). Από την ιστοσελίδα του πανεπιστημίου UCI η συγκεκριμένη βάση δεδομένων υπάρχει με ελλιπής τιμές σε δεκαέξι (16) παραδείγματα, οι ελλιπής τιμές σημειώνονται με “?” πράγμα που καθιστά τα δεδομένα αυτά μη κατάλληλα για τον αλγόριθμο. Εμείς συμπληρώσαμε με τα σωστά στοιχεία όπου υπήρχαν “?” από το cancer_dataset.mat του matlab που βρίσκεται στην διαδρομή: C:\Program Files \MATLAB\R2008a\toolbox\nnet\ nndemos\nndatasets.

Αυτό το σετ δεδομένων μπορεί να χρησιμοποιηθεί από ένα γιατρό και ένα επιστημονικό συνεργάτη ούτως ώστε να δημιουργηθεί μια τεράστια βάση δεδομένων όπου σε συνδυασμό με μια μηχανή διανυσμάτων υποστήριξης να γίνεται έγκυρη πρόβλεψη των καλοήθης καρκίνων μαστού αποφεύγοντας την επιδείνωση τους σε κακοήθης καρκίνων μαστού.

Πίνακας Breast Cancer Wisconsin (Original)

Είδος καρκίνου του μαστού	Παραδείγματα εκπαίδευσης	Παραδείγματα δοκιμής
Κακοήθης (Maligrant): 241 (1-241)	1 – 192 (192) [79,668%]	193 – 241 (49) [20,332%]
Καλοήθης (Benign): 458 (242-699)	242 – 607 (366) [79,9127%]	608 – 699 (92) [20,0873%]

9.1.4 Wine recognition data (178 δεδομένα με 13 – διάστατα ανύσματα)

Το σετ δεδομένων Wine συλλέχτηκε από την Forina, M. et al, PARVUS τον Ιούλιο του 1991 και δόθηκε στην ιστοσελίδα του πανεπιστημίου UCI από τον Stefan Aeberhard Από αυτό το σετ δεδομένων υπάρχουν τα αποτελέσματα χημικής ανάλυσης που καλλιεργούνται στην ίδια περιοχή στην Ιταλία αλλά προέρχονται από τρείς (3) διαφορετικές ποικιλίες κρασιών. Σ' αυτήν την χημική ανάλυση υπάρχουν δεκατρία (13) κοινά συστατικά (χαρακτηριστικά) και για τις τρείς (3) διαφορετικές κατηγορίες κρασιών.

Λόγω του ότι έχουμε τρείς (3) διαφορετικές ποικιλίες κρασιών και αφού ο αλγόριθμος μας χειρίζεται δυαδικά προβλήματα ταξινόμησης εκτελούμε το πείραμα πρώτα για τις K1-K2 κατηγορίες κρασιού, μετά για τις K1-K3 κατηγορίες και τέλος K2-K3.

Πίνακας Wine Recognition Data

Ποικιλίες Κρασιών	Παραδείγματα Εκπαίδευσης	Παραδείγματα Δοκιμής
Κ1 Ποικιλία: 59 (1-59)	1 – 47 (47) [79,661%]	48 – 59 (12) [20,339%]
Κ2 Ποικιλία: 71 (60-130)	60 – 116 (57) [80,2817%]	117 – 130 (14) [19,7182%]
Κ3 Ποικιλία: 48 (131-178)	131 – 168 (38) [79,1667%]	169 – 178 (10) [20,8333%]

9.1.5 Παρέλαση Τελειοφοίτων (80 δεδομένα με 2 – διάστατα ανύσματα)

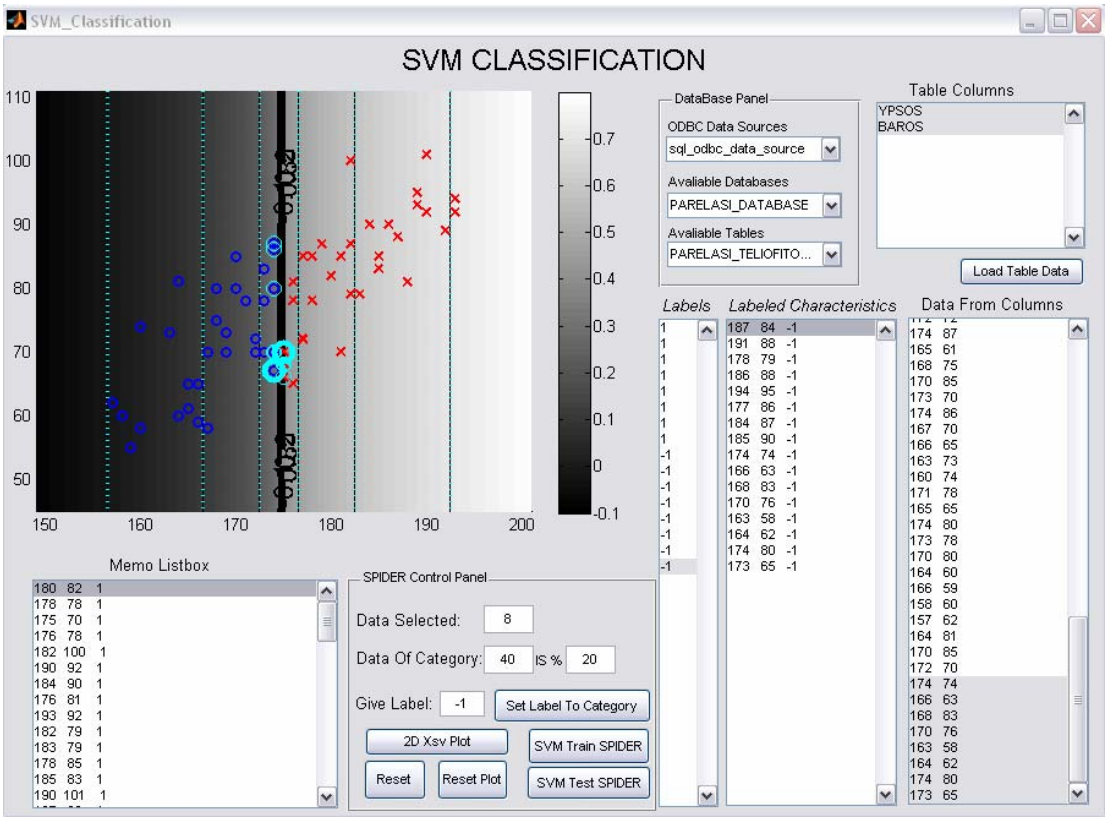
Αυτό το σετ δεδομένων δημιουργήθηκε από εμάς και έγινε περισσότερο **για να δούμε γραφικά** το πώς τοποθετούνται τα δεδομένα σε ένα γραφικό διάγραμμα αφού τα δεδομένα μας είναι 2-διάστατα, βλέποντας επίσης και τα διανύσματα υποστήριξης πατώντας το κουμπί «2D Xsv Plot» της εφαρμογής μας.

Στο εξής πείραμα έχουμε 80 τελειόφοιτους μαθητές όπου θα συμμετέχουν σε παρέλαση όπου τα χαρακτηριστικά τους είναι το ύψος και το βάρος του καθενός, η ταξινόμηση τους γίνεται ανάλογα με το ύψος τους ούτως ώστε να γίνει ομοιόμορφη η κατανομή των τελειοφοίτων σε δύο ομάδες. Στη $1^{η}$ ομάδα μπαίνουν αυτοί που έχουν ύψος 175cm και πάνω και στη $2^{η}$ αυτοί που έχουν 174cm και κάτω.

Άρα έχουμε δυο κατηγορίες, την Κ1(ψηλοί) και την Κ2(κοντοί).

Τα πρώτα 40 άτομα είναι οι ψηλοί και οι υπόλοιποι από τον $41^{ο}$ εως τον $80^{ο}$ είναι κοντοί. Στην αρχή εισάγονται τα πρώτα 32 άτομα(80%) και των δύο κατηγοριών για εκπαίδευση του αλγόριθμου με ετικέτα 1 αυτοί της Κ1 κατηγορίας(Ψηλοί) και με ετικέτα -1 αυτοί της κατηγορίας Κ2(κοντοί). Αφού ολοκληρωθεί η εκπαίδευση του αλγόριθμο δημιουργώντας το μοντέλο βέλτιστης ταξινόμησης το οποίο έχει τα support vectors στην συνέχεια γίνεται η αυτόματη κατηγοριοποίηση των υπόλοιπων μαθητών που απέμειναν.

Στην εικόνα που ακολουθεί φαίνεται διαγραμματικά η απεικόνιση όλων των παραδειγμάτων που χρησιμοποιήθηκαν για την εκπαίδευση του αλγόριθμου γραμμικής ταξινόμησης SVM καθώς επίσης φαίνονται και οι μηχανές διανυσμάτων υποστήριξης (ΜΔΥ) με έντονους κύκλους :



9.2 Πίνακας αποτελεσμάτων κάνοντας χρήση το πρόγραμμα WEKA

Τα πειράματα στο πρόγραμμα WEKA **[8]** γίνονται χωρίς επιβλεπόμενη μάθηση (unsupervised learning) δηλαδή εμείς επιλέγουμε το σετ δεδομένων που θέλουμε για υλοποίηση πειράματος ως είσοδο αλγόριθμου και ως έξοδο μας βγάζει την ποσοστιαία τιμή σωστής ταξινόμησης. Το εκπαιδευτικό πρόγραμμα WEKA ήταν η πρώτη μας επαφή την με μηχανική μάθηση και την εξόρυξη γνώσης.

Με το πρόγραμμα WEKA **[8]**πραγματοποιούμε τα πειράματα μας με τέσσερις (4) διαφορετικούς αλγόριθμους:

- J48 (C 4.5)
- iBK (kNN)
- NaiveBayes
- LibSVM

Οι πιο πάνω αλγόριθμοι επιλέχτηκαν γιατί ανήκουν μέσα στους δέκα (10) καλύτερους αλγόριθμους που επιλέχτηκαν από το IEEE International Conference (ICDM) στην εξόρυξη γνώσης **[9]**. Επιλέξαμε τέσσερις (4) από τους καλύτερους αλγόριθμους εξόρυξης γνώσης και από αυτούς θα βρούμε τον καλύτερο.

Δημιουργήσαμε πέντε (5) αρχεία του τύπου (*.arff, Attribute – Relation File Format) όπου είναι τα δεδομένα εισόδου (data) για να μπορέσουμε να διεξάγουμε τα πειράματα μας στο πρόγραμμα WEKA. Τα αρχεία αυτά περιέχουν τα ίδια σετ δεδομένων που χρησιμοποιήσαμε στον SPIDER SVM.

ΣΕΤ ΔΕΔΟΜΕΝΩΝ	J48	iBK	NaiveBayes	LibSVM
IRIS(150)	96%(144)	95,33%(143)	96%(144)	96,66%(145)
WINE(178)	93,82%(167)	94,94%(169)	96,62%(172)	95,50%(170)
GLASS(214)	66,82%(143)	70,56%(151)	48,59%(104)	62,61%(134)
BREAST CANCER(699)	94,56%(661)	95,13%(665)	95,99%(671)	96,85%(677)
ΠΑΡΕΛΑΣΗ(80)	100%	93,75%(75)	88,75%(71)	98,75%(79)

Με έντονο κόκκινο χρώμα φαίνεται το μεγαλύτερο ποσοστό επιτυχίας για το εκάστοτε πείραμα που διεξάχθηκε. Καλύτερος αλγόριθμος παρατηρούμε ότι είναι ο LibSVM γιατί πέτυχε τις περισσότερες επιτυχίες από τους υπόλοιπους αλγόριθμους.

9.3 Πίνακας αποτελεσμάτων κάνοντας χρήση τον SPIDER SVM

Τα πέντε (5) πειράματα έγιναν με επιβλεπόμενη μάθηση (supervised learning) ορίζοντας εμείς μέσω της εφαρμογής μας στο matlab το ποια δεδομένα θα εκπαιδευτούν και σε ποια θα γίνει η δοκιμή.

Κάναμε συνολικά πέντε (5) πειράματα ταξινόμησης με τον αλγόριθμο γραμμικής κατηγοριοποίησης SPIDER **[7]** SVM του οποίου τα αποτελέσματα φαίνονται ως ακολούθως:

ΣΕΤ ΔΕΔΟΜΕΝΩΝ	K1 – K2	K1 – K3	K2 – K3
IRIS(150)	100%	100%	-
WINE(178)	100%	100%	100%
GLASS(214)	100%		
BREAST CANCER(699)	100%		
ΠΑΡΕΛΑΣΗ(80)	100%		

Ο αλγόριθμος SPIDER **[7]** εκπλήρωσε με απόλυτη επιτυχία 100% την αυτόματη κατηγοριοποίηση στα εναπομείναντα παραδείγματα και των πέντε (5) πειραμάτων πλειν του σετ δεδομένων IRIS μεταξύ των κατηγοριών K2-K3.

Παρατήρηση: Επειδή ο αλγόριθμος SVM της εργαλειοθήκης SPIDER **[7]** είναι του τύπου Hard Margin SVM, δεν επιτρέπει να γίνεται επικάλυψη δεδομένων μεταξύ των κατηγοριών προς εκπαίδευση. Η ταξινόμηση μεταξύ των κατηγοριών K2-K3 του σετ δεδομένων IRIS δεν γίνεται εφικτή γιατί τα δεδομένα της K2 εμπίπτουν στην κατηγορία K3 και αντίστροφα.

9.4 Πίνακας αποτελεσμάτων κάνοντας χρήση την βιβλιοθήκη LIBSVM

Και εδώ όπως και στο πείραμα με τον SPIDER αλγόριθμο έχουμε πέντε (5) πειράματα που έγιναν με επιβλεπόμενη μάθηση (supervised learning) ορίζοντας εμείς μέσω της εφαρμογής μας στο matlab το ποια δεδομένα θα εκπαιδευτούν και σε ποια θα γίνει η δοκιμή.

Κάναμε συνολικά πέντε (5) πειράματα ταξινόμησης με τον αλγόριθμο γραμμικής κατηγοριοποίησης LIBSVM του οποίου τα αποτελέσματα φαίνονται ως ακολούθως:

ΣΕΤ ΔΕΔΟΜΕΝΩΝ	K1 – K2	K1 – K3	K2 – K3
IRIS(150)	100%	100%	100%
WINE(178)	100%	100%	100%
GLASS(214)	100%		
BREAST CANCER(699)	100%		
ΠΑΡΕΛΑΣΗ(80)	100%		

Ο αλγόριθμος LIBSVM εκπλήρωσε με απόλυτη επιτυχία 100% την αυτόματη κατηγοριοποίηση στα εναπομείναντα παραδείγματα και των πέντε (5) πειραμάτων ακόμα και του σετ δεδομένων IRIS μεταξύ των κατηγοριών K2-K3 όπου στις μεταξύ των δύο (2) κατηγοριών υπάρχει επικάλυψη δεδομένων της μίας κατηγορίας μέσα στην άλλη κατηγορία.

Προβλήματα τα οποία προέκυψαν κατά την επίλυση της εφαρμογής

Η ανεύρεση των καταλληλότερων προσομοιωτών για την διεξαγωγή των πειραμάτων μας ήταν ένα από τα κύρια προβλήματα για την εκπόνηση της παρούσας πτυχιακής εργασίας. Από τις διάφορες εφαρμογές που εξηύρα υπήρχαν λάθη στη μετάφραση και στο runtime όπου αποπειράθηκα να επέμβω στον κώδικα των εφαρμογών πράγμα που αποδείχτηκε πάρα πολύ δύσκολο γιατί δεν υπάρχει η κατάλληλη βιβλιογραφία αυτών. Το πρόβλημα αυτό ξεπεράστηκε βρίσκοντας την εργαλειοθήκη SPIDER **[7]** και την βιβλιοθήκη LIBSVM όπου αυτές οι εφαρμογές είναι κατάλληλες στο να λύνουν προβλήματα ταξινόμησης κάνοντας χρήση μηχανών διανυσμάτων υποστήριξης.

Η μετατροπή των σετ δεδομένων στην κατάλληλη μορφή για εισαγωγή τους στους αλγόριθμους WEKA, SPIDER και LIBSVM ήταν ακόμη ένα σημαντικό πρόβλημα το οποίο παρουσιάστηκε. Αυτό το πρόβλημα ξεπεράστηκε ομαλά γιατί υπήρχε η κατάλληλη βιβλιογραφία. Οι προσομοιωτές φαίνεται να είναι αρκετά εκλεκτικοί στα δεδομένα που τους δίνονται για εκπαίδευση και δοκιμή.

Ακόμα ένα πρόβλημα το οποίο προέκυψε κατά την επίλυση του κώδικα στο matlab είναι ότι αφού γινόταν η εκπαίδευση των δεδομένων μεταξύ των δύο(2) κατηγοριών και όταν εισάγονταν τα νέα ¨άγνωστα¨ δεδομένα για δοκιμή χωρίς ετικέτες και πατώντας το κουμπί «SVM Test SPIDER» ο αλγόριθμος παρουσίαζε σφάλμα και δεν μπορούσε να πραγματοποιηθεί η δοκιμή(test). Το σφάλμα διορθώθηκε εισάγοντας μία τυχαία ετικέτα π.χ. ετικέτα -1 για όλα τα δεδομένα στην δοκιμή. Κάτι αντίστοιχο για το πείραμα LIBSVM, όταν εισάγονταν τα νέα ¨άγνωστα¨ δεδομένα για δοκιμή για να βγει σωστό το ποσοστό της πρόβλεψης (Prediction) εάν δώσουμε σε όλα τα δεδομένα δοκιμής π.χ. -1 ή σε όλα 1 το ποσοστό πρόβλεψης βγαίνει λανθασμένο. Ως λύση αυτού του προβλήματος εισάγουμε τις σωστές ετικέτες των κατηγοριών των δεδομένων δοκιμής και τότε παίρνουμε σωστή ποσοστιαία πρόβλεψη έξοδο από τον αλγόριθμο LIBSVM.

Συμπεράσματα

Ως μια εισαγωγή στην μελέτη μας για κατηγοριοποίηση δεδομένων με μηχανές διανυσμάτων υποστήριξης είχαμε μη-επιβλεπόμενη μάθηση χρησιμοποιώντας το εκπαιδευτικό πρόγραμμα WEKA. Βρήκαμε τέσσερις (4) διαφορετικούς αλγόριθμους που χρησιμοποιούν ταξινόμηση και εισάγοντας σε όλους τα σετ δεδομένων μας συμπεράναμε ότι η βιβλιοθήκη LIBSVM πέτυχε τις περισσότερες επιτυχίες από τους υπόλοιπους αλγόριθμους.

Με την μη-επιβλεπόμενη μάθηση έχουμε υψηλό ποσοστό επιτυχίας χρησιμοποιώντας την τεχνική διασταυρωμένων επικυρώσεων (cross-validation) αλλά όχι το απόλυτο 100% που επιτυγχάνεται με την προσεγμένη επιστημονική προσέγγιση με επίβλεψη. Στην μη-επιβλεπόμενη μάθηση ο χρόνος που χρειάζεται για να έρθει εις πέρας κάποιο πείραμα είναι κατά πολύ λιγότερος από τον χρόνο που χρειάζεται για να έρθει εις πέρας κάποιο πείραμα με επίβλεψη γιατί στην πρώτη περίπτωση επιλέγουμε μόνο τον αριθμό διασταυρωμένων επικυρώσεων ενώ στην δεύτερη περίπτωση με επίβλεψη χρειάζεται να γίνει η προετοιμασία των δεδομένων, η επιλογή δεδομένων εκπαίδευσης και δεδομένων δοκιμής, η επιλογή του πυρήνα kernel κ.ά..

Ο κύριος λόγος της ταχύτατης και απόλυτης επιτυχίας από SPIDER αλγόριθμο και από την βιβλιοθήκη LIBSVM οφείλεται στο μοντέλο πρόβλεψης όπου αποθηκεύει

ως πληροφορία μόνα τα διανύσματα υποστήριξης (και της κλάσης $-\frac{1}{\|w\|}$), τον αριθμό των κλάσεων, τον συνολικό αριθμό διανυσμάτων υποστήριξης που εκάστοτε βρέθηκαν, το βέλτιστο διάνυσμα βαρών w και τον βέλτιστο αριθμό πόλωσης b (bias).

Συγκρίνοντας τον αλγόριθμο SPIDER με την βιβλιοθήκη LIBSVM καταλήγουμε στο συμπέρασμα ότι και οι δύο πέτυχαν το απόλυτο επιτυχίας σε όλα τα πειράματα που φέραμε εις πέρας, με την διαφορά ότι η βιβλιοθήκη LIBSVM έφερε εις πέρας και την ταξινόμηση του σετ δεδομένων IRIS μεταξύ των κατηγοριών K2-K3 εκεί όπου ο αλγόριθμος SPIDER δεν εκπλήρωσε αυτή την ταξινόμηση παρουσιάζοντας σφάλμα στο runtime.

Προσωπική επιλογή ως καταλληλότερου εργαλείου κατηγοριοποίησης με μηχανές διανυσμάτων υποστήριξης επιλέγω την βιβλιοθήκη LIBSVM γιατί πέτυχε τα καλύτερα ποσοστά επιτυχίας. Επίσης γιατί έχει περισσότερες επιλογές διάφορων τύπων αλγόριθμων, τύπων πυρήνων kernel πράγμα που αφήνει τροφή για

επόμενους σπουδαστές να συνεχίσουν την έρευνα μελέτης κατηγοριοποίησης δεδομένων.

Μειονέκτημα στις ΜΔΥ: είναι ότι στη μάθηση του αλγορίθμου δίνεται ολόκληρο το σετ δεδομένων εκπαίδευσης, όταν ολοκληρωθεί η εκπαίδευση δεν μπορούν να προστεθούν ακόμη μερικά πρότυπα εκπαίδευσης στο δίκτυο εκπαίδευσης.

Πλεονέκτημα στις ΜΔΥ: είναι ότι μπορούν να χειριστούν διανύσματα μεγάλων διαστάσεων και την μεγάλη ικανότητα γενίκευσης τους.

ΠΑΡΑΡΤΗΜΑ

Α) Διαδικασία εγκατάστασης των απαραίτητων εκτελέσιμων εφαρμογών (Εκπαιδευτικό Πρόγραμμα WEKA)

Πώς γίνεται προσαρμογή του CLASSPATH στα Windows για το εκπαιδευτικό πρόγραμμα WEKA;

1) Κατεβάζουμε από την ιστοσελίδα: http://www.java.com/en/ το πρόγραμμα java και το εγκαθιστούμε στον Η/Υ.
2) Κατεβάζουμε, αποσυμπιέζουμε και κάνουμε επικόλληση στην διαδρομή C:\ τον φάκελο "libsvm-3.1". Ο φάκελος "libsvm-3.1" (Version 3.1, April 2011) είναι η βιβλιοθήκη που χρησιμοποιήσαμε στο πείραμα με το πρόγραμμα WEKA και βρίσκεται στην ιστοσελίδα: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
3) Το εκπαιδευτικό πρόγραμμα WEKA βρίσκεται στην ιστοσελίδα: http://www.cs.waikato.ac.nz/ml/weka/ Κατεβάζουμε εκείνο που είναι για" Windows x86 με Java VM 1,6" και το εγκαθιστούμε στον Η/Υ.

Επίσης για να λειτουργήσει η βιβλιοθήκη "libsvm-3.1" στα WINDOWS πρέπει:

Α) Δεξί κλίκ στο εικονίδιο "My computer" στην επιφάνεια εργασίας και επιλέγουμε "properties"→καρτέλα "Advanced"→πατάμε το κουμπί "Environment Variables".

Β) Στο "system variables" πατάμε το κουμπί "New". Βάζουμε ως Variable name: "CLASSPATH" και ως Variable value: "C:\libsvm-3.1\java\libsvm.jar;C:\Program Files\Weka-3-6\weka.jar;"

Αυτό έγινε για να μπορεί το λειτουργικό σύστημα "Windows XP" να «δει» τις βιβλιοθήκες αυτές [.jar] ("libsvm.jar" και την βιβλιοθήκη "weka.jar") για να μπορέσουμε να πραγματοποιήσουμε τα πειράματα στο εκπαιδευτικό πρόγραμμα WEKA **[8]**.

Ιστοσελίδες που χρησιμοποιήσαμε για τα πειράματα WEKA:

- http://www.cs.waikato.ac.nz/ml/weka/
- http://weka.wikispaces.com/LibSVM
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- http://weka.wikispaces.com/LibSVM

Β) Διαδικασία Ρύθμισης και Εκτέλεσης της Εφαρμογής SPIDER SVM

1) Τα ακόλουθα προγράμματα πρέπει να είναι εγκατεστημένα:

   MS SQL Server 2000

   MATLAB R2008a

2) Κάνουμε επικόλληση τον φάκελο "SPIDER" στην διαδρομή C:\

   Ανατρέχουμε στο ΠΑΡΑΡΤΗΜΑ Ζ) βρίσκουμε τον φάκελο arxeia(arff,excel,txt,mdf kai log).

3) Ανατρέχουμε στο ΠΑΡΑΡΤΗΜΑ Δ) Δημιουργία ODBC Data Source στο matlab για σύνδεση με SQL Server.

4) Ανοίγουμε το matlab και στο command prompt εκτελούμε την εντολή >>guide.

   Ανατρέχουμε στο Κεφάλαιο 8 μέχρι και το Κεφάλαιο 8.1.

Γ) Σετ Δεδομένων

Για κάθε σετ δεδομένων που χρησιμοποιήσαμε στα πειράματα αυτόματης ταξινόμησης στον αλγόριθμο SVM εργαζόμαστε ως εξής:

Αντιγράφουμε (copy) όλα τα δεδομένα από την εκάστοτε ιστοσελίδα που υπάρχει το σετ δεδομένων και τα κάνουμε επικόλληση (paste) σε ένα αρχείο excel. Τα δεδομένα που κάναμε επικόλληση στο αρχείο excel συνήθως δεν είναι τακτοποιημένα όπως εμείς θα θέλαμε, δηλαδή είτε είναι τα παραδείγματα ενωμένα το ένα παράδειγμα με το άλλο χωρισμένα με “,” είτε περιέχουν αχρείαστες πληροφορίες για εμάς π.χ. να υπάρχει στήλη που να έχει τον αριθμό δείγματος

παραδειγμάτων “ID Number” η οποία πληροφορία είναι αχρείαστη για τον αλγόριθμο.

Κάνουμε save as ως excel 97-2003 με το επιθυμητό όνομα που θέλουμε αφήνοντας μόνο μέσα σε αυτό τα δεδομένα μας όπου είναι πραγματικοί αριθμοί (integer και float). Μετά τα δεδομένα τα κάνουμε επικόλληση σε ένα αρχείου .txt Ακολούθως κάνουμε μια νέα βάση δεδομένων στον SQL Server με όνομα π.χ. IRIS_DATABASE και κάνουμε Import Data επιλέγοντας το αρχείο excel IRIS σε ένα πίνακα τον οποίο ονομάζουμε IRIS_TABLE. Τώρα πλέον μέσω του ODBC Driver του SQL Server μπορούμε με την εφαρμογή μας να περιεργαστούμε τα iris δεδομένα στα πειράματα μας.

Από τα πειράματα μας θα είμαστε σε θέση για κάθε νέο δεδομένο που θα εισάγουμε να απαντούμε εάν προήλθε από την πρώτη , την δεύτερη ή την τρίτη κατηγορία λουλουδιών.

Πώς εργαζόμαστε όταν πάμε να πραγματοποιήσουμε ένα πείραμα;

Εργαζόμαστε ως εξής:

Ως απαραίτητη προϋπόθεση πρέπει τα δεδομένα να είναι αποθηκευμένα σε ένα πίνακα στον SQL Server και όλα τα δεδομένα να έχουν ίσο αριθμό χαρακτηριστικών μεταξύ τους.

Καταρχάς τα δεδομένα που θα χρησιμοποιηθούν σε πείραμα πρέπει να είναι γραμμικώς διαχωρίσιμα με τουλάχιστον ένα(1) χαρακτηριστικό τους αφού εμείς χρησιμοποιούμε τον γραμμικό kernel αλγόριθμο. Εάν θέσουμε δικούς μας κανόνες διαχωρισμού π.χ. όσοι είναι 1.80 και πάνω είναι ψηλοί και όσοι είναι κάτω από 1.80 είναι κοντοί, πρέπει αυτοί οι κανόνες να τηρηθούν πιστά από τον χρήστη κατά την διάρκεια της εκπαίδευσης δεδομένων ούτως ώστε να δημιουργηθεί ένα καλά γενικευμένο μοντέλο ταξινόμησης αφού το μοντέλο αυτό να χρησιμοποιηθεί μετέπειτα στην δοκιμή των νέων(άγνωστων) δεδομένων. Στο στάδιο της εκπαίδευσης πρέπει να εισάγεται ένα ικανοποιητικό μέρος των δεδομένων, ποσοστιαία αυτό το ικανοποιητικό μέρος είναι το 70% - 80% από τα συνολικά δεδομένα. Να γίνεται σωστή χρήση ετικετών (labels) από τον χρήστη.

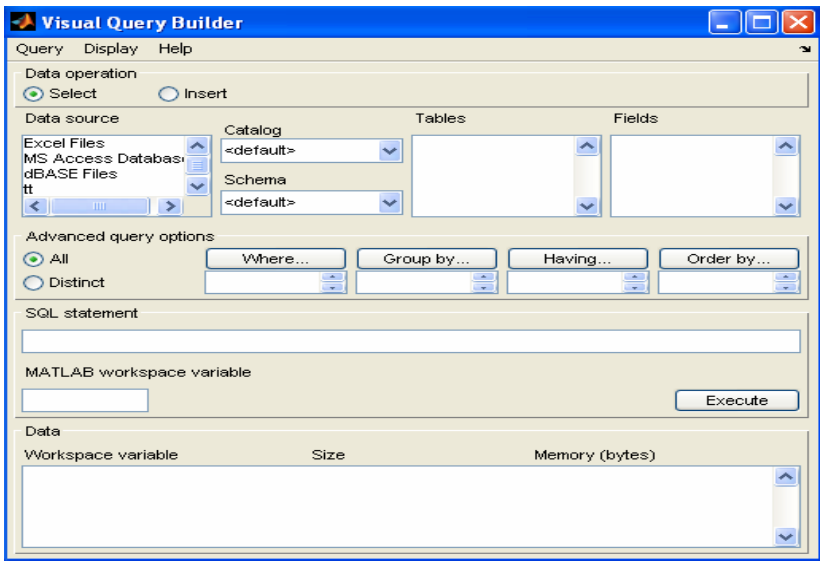
Όταν φτάσει η στιγμή να εισάγουμε στο σύστημα τα δεδομένα δοκιμής(test data) για να δοκιμάσουμε την απόδοση του μοντέλου που δημιουργήθηκε κατά την εκπαίδευση του αλγορίθμου εμείς γνωρίζοντας εκ των προτέρων:

- τον συνολικό αριθμό των δεδομένων προς δοκιμή(test) και
- την σωστή κατηγορία στην οποία ανήκει κάθε παράδειγμα προς δοκιμή(test), τότε θα ήμαστε σε θέση να καταλήγουμε σε ένα ποσοστό επιτυχίας αυτόματης κατηγοριοποίησης.

Ακολούθως από το listbox με static text "Labels"(πίνακας εξόδου) μετράμε πόσες κατηγοριοποιήσεις έγιναν λανθασμένες και μετά καταλήγουμε σε ένα επί τοις εκατό ποσοστό σωστής από τον αλγόριθμο αυτόματης κατηγοριοποίησης.

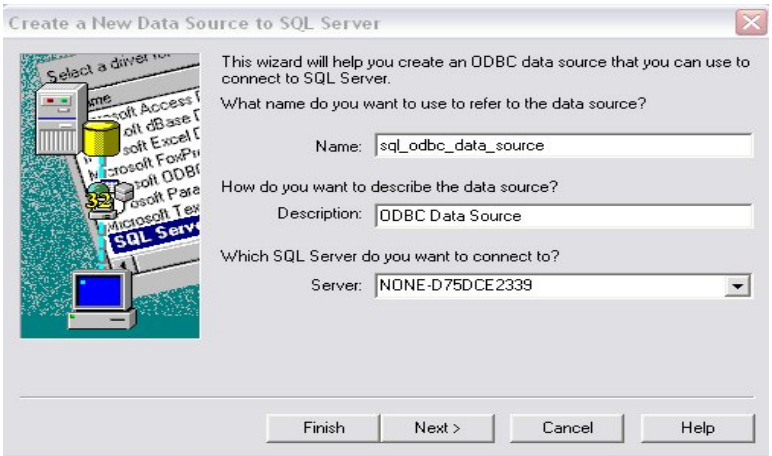
Δ) Δημιουργία ODBC Data Source στο matlab για σύνδεση με SQL Server

Για να δημιουργήσουμε ένα ODBC Data source(sql_odbc_data_source), εκτελούμε την εντολή querybuilder στο command prompt του matlab. Όταν εκτελέσουμε την εντολή querybuilder μας εμφανίζεται το ακόλουθο παράθυρο:



Μετά από το menu Query→Define ODBC Data Source… . Ανοίγει το παράθυρο ODBC Data Source Administrator και από την καρτέλα User DSN πατάμε το κουμπί Add… και ανοίγει το παράθυρο Create New Data Source, επιλέγουμε SQL Server και πατάμε το κουμπί Finish.

Ακολούθως εμφανίζεται το παρακάτω παράθυρο:



Τέλος πατάμε το κουμπί Finish, τώρα έχουμε δημιουργήσει ένα ODBC data source με όνομα “sql_odbc_data_source” όπου είναι το ίδιο όνομα που χρησιμοποιείται στον κώδικα του matlab. Πριν τρέχουμε την εφαρμογή μας πρέπει να κάνουμε αντιγραφή και επικόλληση στον C:\ τον φάκελο SPIDER όπου περιέχεται ο αλγόριθμος μας, αφού γίνει αυτό ανοίγουμε το matlab και από το menu του matlab File→Set Path… ανοίγει το παράθυρο Set Path, πατάμε το κουμπί Add with Subfolders…, βρίσκουμε τον φάκελο spider που υπάρχει στον C:\ και τέλος πατάμε το κουμπί Save.

Τώρα πλέον είμαστε έτοιμοι να τρέξουμε την εφαρμογή μας εκτελώντας απλά από το command prompt του matlab την εντολή guide, ανοίγει το παράθυρο GUIDE Quick Start, επιλέγουμε την καρτέλα Open Existing GUI και κάνοντας αναζήτηση μέσω του κουμπιού Browse βρίσκουμε την εφαρμογή SVM_Classification.fig και τέλος πατάμε το κουμπί Open.

Ε) Τρόποι Εισαγωγής Δεδομένων για υλοποίηση πειραμάτων

Μπορούμε με τρείς(3) τρόπους να εισάγουμε δεδομένα για τα πειράματα μας:

1. Να δημιουργήσουμε βάση δεδομένων στον enterprice manager SQL Server
2. Να πάρουμε μια έτοιμη βάση δεδομένων (.MDF και .LDF αρχεία) και να την χρησιμοποιήσουμε στα πειράματα μας
3. Να εισάγουμε αρχείο excel ή ένα αρχείο .txt υπό μορφή πίνακα στον SQL Server

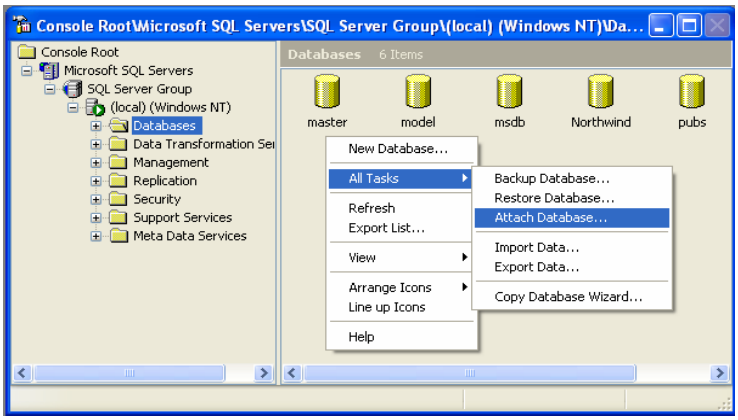
Ζ) Εισαγωγή υπάρχουσας ΒΔ στο Enterprise Manager του Microsoft SQL Server 2000

Διαδικασία:

Βήμα 1: Πρώτα σταματάμε τον Service Manager του SQL Server κάνοντας δεξί κλικ πάνω στο εικονίδιο του και ακολούθως πατάμε την επιλογή stop στον Η/Υ που θέλουμε να πάρουμε μια υπάρχον βάση, πηγαίνουμε στην διαδρομή C:\Program Files\Microsoft SQL Server\MSSQL\Data, σε αυτή την διαδρομή η κάθε βάση δεδομένων αποτελείται από δύο(2) αρχεία(.MDF και .LDF) όπου κάνουμε αντιγραφή αυτά τα δύο αρχεία και τα κάνουμε επικόλληση σε κάποιο μέσο αποθήκευσης(Memory Stick, Compact Disk, External Hard Drive).

Βήμα 2: Στον Η/Υ που θέλουμε να κάνουμε επικόλληση την έτοιμη βάση δεδομένων που βρήκαμε, σταματάμε τον service manager και κάνουμε επικόλληση τα δύο(2) αρχεία της εκάστοτε βάσης δεδομένων στην διαδρομή: C:\Program Files\Microsoft SQL Server\MSSQL\Data.

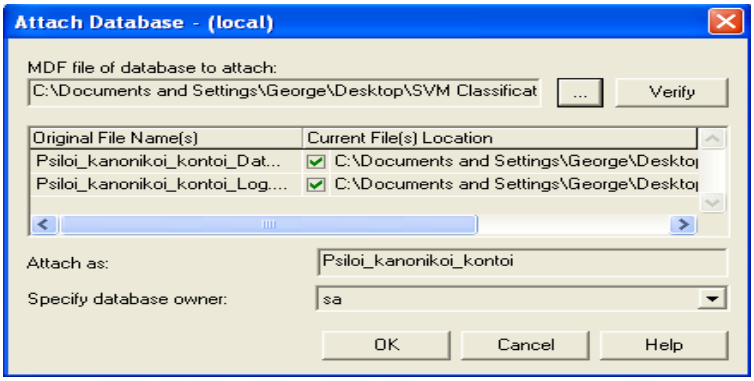
Βήμα 3: Κάνουμε επανεκκίνηση τον service manager πατώντας δεξί κλικ πάνω στο εικονίδιο του και πατάμε την επιλογή start για να ξεκινήσει. Ανοίγουμε τον Enterprise Manager και όπως φαίνεται στην παρακάτω εικόνα:



Πατάμε στον φάκελο Databases→δεξί κλικ στο κενό του παράθυρου που εμφανίζονται τα εικονίδια των βάσεων δεδομένων→All Tasks→Attach Database… .

<u>Βήμα 4:</u>

Από το παρακάτω παράθυρο που εμφανίζεται:



Πηγαίνουμε και βρίσκουμε τον φάκελο όπου αποθηκεύσαμε τα δύο(2) αρχεία της βάσης δεδομένων δηλαδή στην διαδρομή C:\ Program Files\Microsoft SQL Server\MSSQL\Data και επιλέγουμε το .MDF αρχείο, τέλος πατάμε το κουμπί ok. Τώρα πλέον μπορούμε να χρησιμοποιήσουμε τους έτοιμους πίνακες στα πειράματα μας στον αλγόριθμο γραμμικής κατηγοριοποίησης SVM.

H) <u>Εισαγωγή πίνακα στο SQL Server από ένα ήδη υπάρχον αρχείο excel (*.xls) ή από υπάρχον αρχείο κειμένου (*.txt)</u>

Ανοίγουμε τον SQL Server και δημιουργούμε μία νέα βάση δεδομένων π.χ. “IRIS_DATABASE”, μετά πατάμε δεξί κλίκ στην βάση “IRIS_DATABASE” → Properties → Καρτέλα Permissions και τσεκάρουμε το Create Table.

Όταν δημιουργηθεί πίνακας με τα στοιχεία του αρχείου excel καταλαμβάνεται αυτόματα από τον SQL Server η πρώτη γραμμή του αρχείου excel για την ονοματολογία των στηλών π.χ. F1,F2 κλπ. Γι' αυτό τον λόγο προσθέτουμε μία νέα γραμμή στην αρχή για να την αποθήκευση ονοματολογίας στηλών(F1,F2 κτλ) χωρίς να χάνουμε την πρώτη γραμμή από τα πειραματικά μας δεδομένα. Από το μενού Tools→Data Transformation Services→ Import Data… → στο παράθυρο που μας εμφανίζεται πατάμε το κουμπί next. Ως Data source επιλέγουμε →Microsoft Excel 97-2000, μετά βρίσκουμε και επιλέγουμε το αρχείο IRIS.xls, πατάμε next, next και next. Επιλέγουμε το φύλλο του αρχείου excel που θέλουμε να εισάγουμε που στην περίπτωση μας έγινε μετονομασία και από Φύλλο1 μετονομάστηκε σε IRIS και πατάμε next.

Στην συνέχεια αφήνουμε τσεκαρισμένη την επιλογή Run immediately, τσεκάρουμε την επιλογή Save DTS Package και το radio button SQL Server, πατάμε next next και τέλος πατάμε το κουμπί finish. Τέλος κάνουμε δεξί κλικ→ Refresh στο κενό όπου υπάρχουν οι πίνακες και μας εμφανίζεται ο νέος πίνακας μας IRIS.

Αναλόγως εργαζόμαστε και όταν θέλουμε να εισάγουμε ένα αρχείο κειμένου τύπου (*.txt) πηγαίνοντας από το μενού Tools→Data Transformation Services→Import Data…→στο παράθυρο που μας εμφανίζεται πατάμε το κουμπί next και τέλος ως Data Source επιλέγουμε → *.txt και τα υπόλοιπα βήματα για να ολοκληρωθεί η διαδικασία είναι ακριβώς τα ίδια όπως αναγράφονται για αρχείο excel.

Θ) Τρόπος λειτουργίας της εφαρμογής SVM Classification.fig

Εκτέλεση πειράματος π.χ. το πείραμα με το σετ δεδομένων Iris: Λόγω του ότι το σετ δεδομένων Iris απαρτίζεται από τρία (3) διαφορετικά είδη λουλουδιών αυτό σημαίνει ότι έχουμε μέχρι τρείς (3) δυνατούς συνδυασμούς πειραμάτων (K1-K2, K1-K3 και K2-K3) για να ολοκληρωθεί το πείραμα Iris γιατί ο SVM αλγόριθμος είναι

δυαδικής ταξινόμησης, δηλαδή μόνο ανά δύο (2) κατηγορίες δουλεύει την εκάστοτε φορά εκτέλεσής του.

K1-K2 (Iris Setosa – Iris Virginica)

1) Πρώτα επιλέγουμε τα πρώτα 40 παραδείγματα [80%] από την κατηγορία K1 βάζοντας τους στον κενό χώρο δίπλα από το static text «Give Label» την ετικέτα 1 και μετά πατάμε το κουμπί «Set Label To Category » για να γίνει η καταχώρηση στο listbox με static text «Labeled Characteristics», η ίδια διαδικασία γίνεται και για τα πρώτα 40 παραδείγματα της κατηγορίας K2 με μόνη διαφορά ότι για ετικέτα βάζουμε τον αριθμό -1.
2) Αφού τοποθετηθούν όλα τα παραδείγματα προς εκπαίδευση(train) στο listbox με «Labeled Characteristics» πατάμε το κουμπί «Train SVM SPIDER» και όταν εμφανιστεί στο command prompt του Matlab το μήνυμα «training svm kernel linear....» σημαίνει ότι ο αλγόριθμος δημιούργησε επιτυχώς το βέλτιστο μοντέλο ταξινόμησης, τώρα εάν εμφανιστεί μήνυμα λάθους στο command prompt σημαίνει ότι υπάρχει επικάλυψη διανυσμάτων μεταξύ των κατηγοριών και ότι το μοντέλο που δημιουργήθηκε δεν θα λειτουργεί σωστά.
3) Αδειάζουμε το listbox «Labeled Characteristics» πατώντας το κουμπί «Reset» γιατί μέσα σε αυτό θα μπουν τα εναπομείναντα «άγνωστα» δεδομένα δοκιμής. Επιλέγουμε τα εναπομείναντα παραδείγματα για δοκιμή (test) βάζοντας σε όλα ίδια ετικέτα (για λόγους απλότητας) π.χ. -1. Τέλος πατάμε το κουμπί «Test SVM SPIDER».

Τα αποτελέσματα εξόδου του αλγόριθμου εμφανίζονται στο listbox «Labels».

K1-K3 (Iris Setosa – Iris Versicolor)

- Πατάμε το κουμπί «Reset» και αδειάσουν οι μεταβλητές «handles.Y_axis=[];» και «handles.oldrowselection=[];» καθώς επίσης οι στήλες «Labeled Characteristics» και «Labels»
- Εκτελούμε τα προηγούμενα βήματα 1), 2) και 3) που έγιναν για τις κατηγορίες K1-K2 αλλά τώρα για τις κατηγορίες K1-K3

K2-K3 (Iris Virginica – Iris Versicolor)

Δεν είναι δυνατή η πραγματοποίηση πειράματος από τον αλγόριθμο σε αυτήν την περίπτωση γιατί τα διανύσματα της κατηγορία K2 εμπίπτουν στην κατηγορία K3 και αντίστροφα. Ο αλγόριθμος μπορεί να χειριστεί μόνο δυαδικές ταξινομήσεις (ανά δύο (2) κατηγορίες κάθε φορά) όπου τα παραδείγματα μεταξύ των δύο (2) διαφορετικών κατηγοριών πρέπει να είναι διαχωρίσιμα με τουλάχιστον ένα (1) από τα χαρακτηριστικά τους.

Ι) Προσαρμογή και εγκατάσταση της βιβλιοθήκης LIBSVM-3.11

Πρέπει πρώτα να κατεβάσουμε την βιβλιοθήκη LIBSVM στον Η/Υ μας. Η βιβλιοθήκη βρίσκεται στην ιστοσελίδα: http://www.csie.ntu.edu.tw/~cjlin/libsvm/#download

Για να γίνει εφικτή η λειτουργία της βιβλιοθήκης LIBSVM πρέπει να εγκαταστήσουμε στον Η/Υ δύο (2) εκτελέσιμα προγράμματα τα οποία είναι:

- Microsoft Visual C++ 2010 Express
- Microsoft Windows Software Development Kit (SDK) 7.1

Περισσότερες πληροφορίες για την εγκατάσταση των δύο (2) προγραμμάτων βρίσκονται στην παρακάτω ιστοσελίδα: http://www.mathworks.com/support/solutions/en/data/1-ECUGQX/

Κ) Όλα όσα πρέπει να γνωρίζουμε κάνοντας χρήση το εκπαιδευτικό πρόγραμμα WEKA

Ως μέθοδο βελτιστοποίησης του ποσοστού σωστής ταξινόμησης χρησιμοποιούμε την διασταυρωμένη επικύρωση (Cross Validation). Η διασταυρωμένη επικύρωση συνδυασμένη με ένα βρόγχο επανάληψης for μπορεί να επιτύχει την βέλτιστη τιμή της παραμέτρου πολυπλοκότητας C και π.χ. της γ παραμέτρου για παλινδρόμηση (regression). Ένα παράδειγμα προγράμματος σε κώδικα όπου επιτυγχάνονται αυτές

οι βέλτιστες τιμές για C και γ παραμέτρων βρίσκεται στο FAQ της ιστοσελίδας LibSVM.

Η CV τεχνική χωρίζει ολόκληρο το σετ δεδομένων σε ίσα μέρη ανάλογα με το πλήθος των διασταυρωμένων επικυρώσεων που ορίζει ο χρήστης. Έχουμε δηλαδή K-fold CV, αυτό σημαίνει ότι εάν ο χρήστης δώσει το K=4, το πρόγραμμα θα τρέξει τέσσερις (4) φορές και την εκάστοτε φορά θα έχει ένα ξεχωριστό ¼ των δεδομένων για δοκιμή (test).

Έστω ότι N είναι ο συνολικός αριθμός προτύπων και K ο αριθμός που αντιπροσωπεύει το πλήθος των διασταυρωμένων επικυρώσεων που πρόκειται να συμβούν. Άρα έχουμε: $\frac{N}{K}$. Η συνηθέστερη είναι η 10-fold CV και αυτή εφαρμόζεται.

Ο αλγόριθμος J48 (C4.5) ανήκει στα δένδρα απόφασης και ρυθμίζεται από μια κύρια παράμετρο C (confidence factor). Ο αλγόριθμος ibk (kNN) ανήκει στην κατηγορία Lazy και ως κύρια παράμετρο λειτουργίας έχει την kNN δηλαδή το πλήθος γειτόνων που θα έχει καθώς και η παράμετρος cross validate. Χρησιμοποιεί την ευκλείδεια απόσταση για την εύρεση των αποστάσεων των προτύπων μεταξύ τους. Ο αλγόριθμος NaiveBayes είναι η πιο απλή μορφή κατά Bayes και η απόδοση του είναι συγκρίσιμη με decision trees. Πλεονέκτημα του ότι ενημερώνεται συνεχώς από νέα πρότυπα χωρίς εκ νέου εκπαίδευση. Ο αλγόριθμος LibSVM χρησιμοποιεί μηχανές διανυσμάτων υποστήριξης.

Βιβλιογραφία

[1] Επιστημονικός Συνεργάτης Τσιμπίρης Αλκιβιάδης. Βάσεις Δεδομένων ΙΙ

[2] Wikipedia, Microsoft SQL Server. Διαθέσιμο στη διεύθυνση: http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[3] Δρ. Χαράλαμπος Π. Στρουθόπουλος Καθηγητής στο ΤΕΙ ΣΕΡΡΩΝ. Αναγνώριση Προτύπων – Νευρωνικά Δίκτυα

[4] Wikipedia, Νευρωνικά Δίκτυα. Διαθέσιμο στη διεύθυνση: http://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%8E%CE%BD%CE%B1%CF%82

[5] Wikipedia, Matlab εγχειρίδιο. Διαθέσιμο στη διεύθυνση: http://en.wikipedia.org/wiki/MATLAB

[6] Wikipedia, Iris σετ δεδομένων. Διαθέσιμο στη διεύθυνση: http://en.wikipedia.org/wiki/Iris_flower_data_set

[7] SPIDER MATLAB TOOLBOX(Core Package). Διαθέσιμο στη διεύθυνση: http://people.kyb.tuebingen.mpg.de/spider/download_frames.html

[8] WEKA για Windows x86 με JAVA VM 1,6. Διαθέσιμο στη διεύθυνση: http://www.cs.waikato.ac.nz/ml/weka/

[9] X. Wu et al, “Top 10 algorithms in data mining”, Knowledge Information Systems, 14:1-37, 2008

[10] SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines. Διαθέσιμο στη διεύθυνση:

http://www.vldb2005.org/program/paper/wed/p1152-milenova.pdf

[11] Pattern Recognition and Machine Learning (Christopher M. Bishop)

[12] Overview of the SVM algorithm

[13] Principles and Theory for Data Mining and Machine Learning (Bertrand Clarke, Ernest Fokoue, Hao Helen Zhang)

[14] Learning with kernels (Bernhard Scholkopf and Alexander J. Smola)

[15] Introduction to Machine Learning (Ethem Alpaydin)