ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

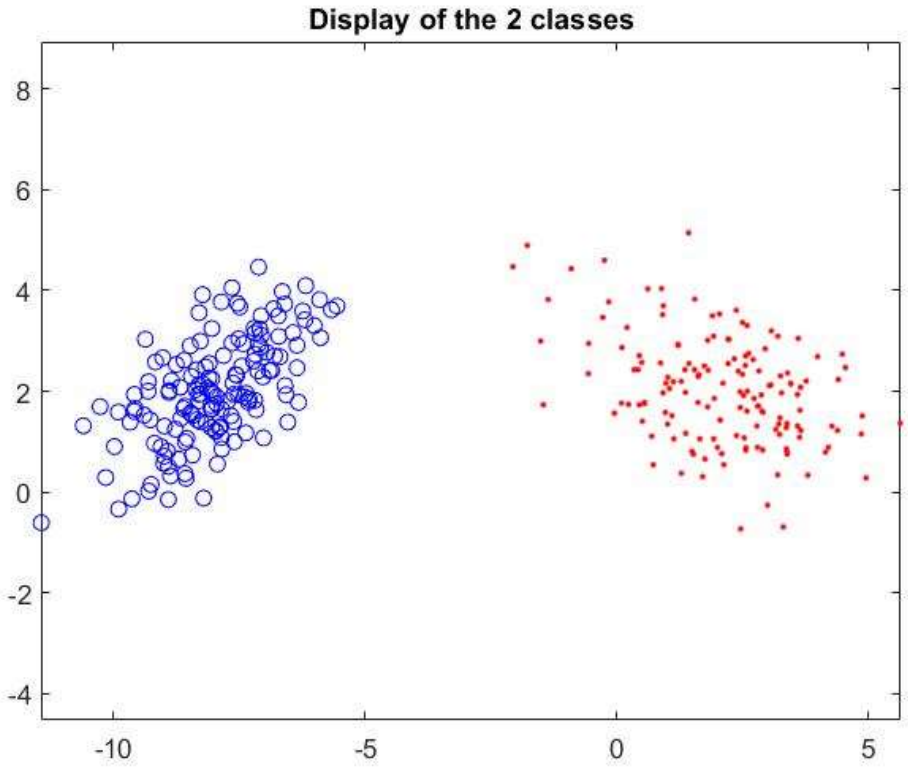
Φαναρίδου Κυριακούλα

9ο εξάμηνο

ΑΕΜ: 57830

ΑΣΚΗΣΗ 1

A. Ζητείται να παραχθούν 150 τυχαία δείγματα από καθεμία από τις κατηγορίες ω_1 και ω_2.
Για να γίνει αυτό χρησιμοποιήθηκε η εντολή mvnrnd(m,S,N) του MATLAB, η οποία δέχεται ως ορίσματα την μέση τιμή (m),τον πίνακα συνδιασποράς (S) και τον αριθμό των δειγμάτων (N=150) για κάθε μία από τις 2 κλάσεις. Σημειώνεται πως χρησιμοποιήθηκε seed και randn για να δημιουργηθούν διαφορετικά τυχαία δείγματα για τις 2 κλάσεις διαστάσεων 150x2.
Στη συνέχεια, έγινε plot των δεδομένων με διαχωρισμό 1 και -1 αναλόγως σε ποια κλάση ανήκουν.
Το διάγραμμα που παράχθηκε είναι το εξής:



B.

Στη συνέχεια ζητείται να υλοποιηθεί ο αλγόριθμος Batch Perceptron και να υπολογιστεί ένας γραμμικός ταξινομητής για τις κλάσεις αυτές.
Ο αλγόριθμος Batch Perceptron πρακτικά αρχικοποιεί ένα τυχαίο διάνυσμα βαρών (w_ini) και το διάνυσμα διόρθωσης w (δηλαδή το διάνυσμα βαρών που υπολογίζει ο αλγόριθμος). Δουλεύει στον επαυξημένο χώρο l+1 διάστατο χώρο και συγκλίνει σε λύση μετά από πεπερασμένο αριθμό βημάτων επανάληψης. Ο αλγόριθμος

διορθώνει το διάνυσμα βαρών προς την κατεύθυνση του x. Με αυτό τον τρόπο, στρέφει το υπερεπίπεδο με σκοπό το x να ταξινομείται σωστά στη σωστή κλάση.

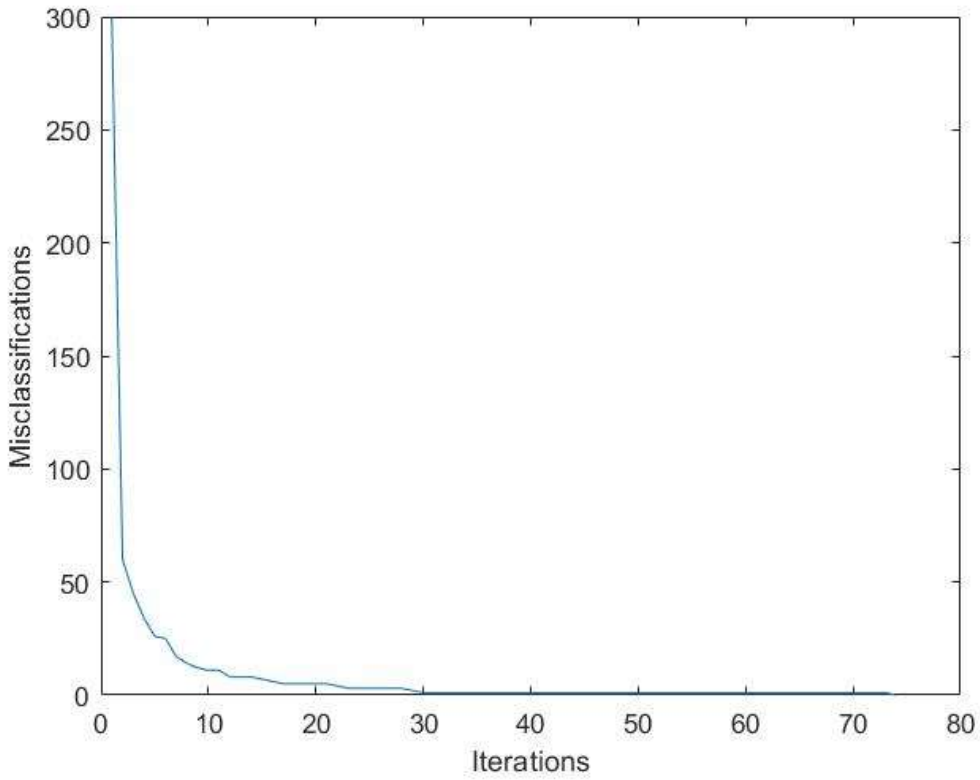
Τα αντίστοιχα αρχεία που χρησιμοποιήθηκαν είναι askisi_1.m και perce_with_plot.m.

Το διάνυσμα βαρών όταν το σφάλμα μηδενίζεται είναι:

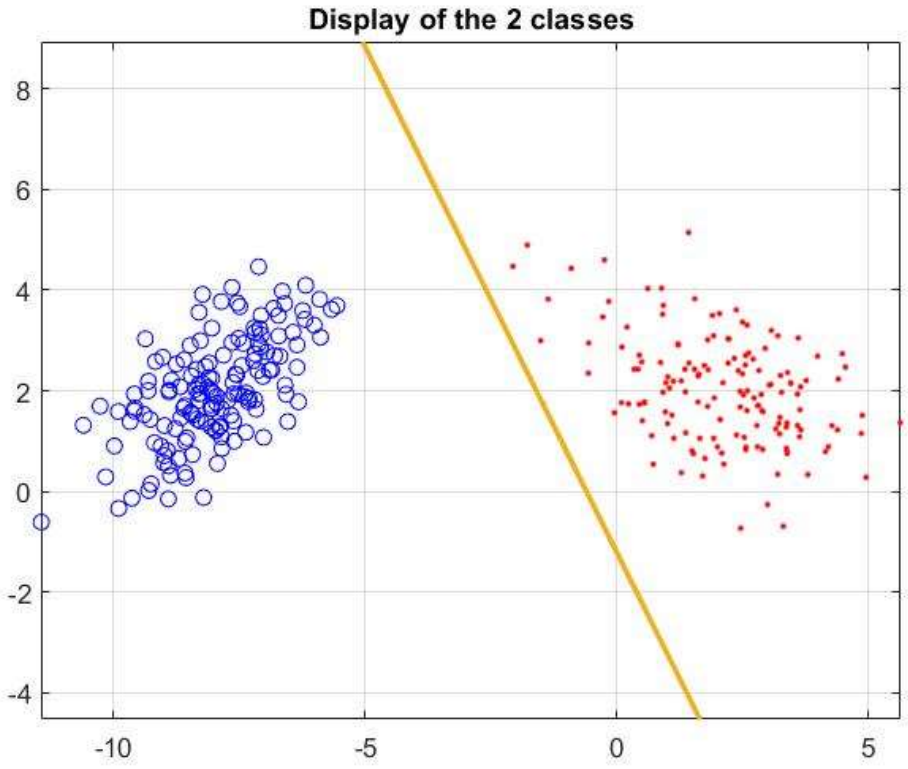
W initial	W final
6	**-7.2245**
8	**-3.5966**
-0.5	**-4.35**

Παρατίθεται και το διάνυσμα αρχικών βαρών για σύγκριση.

Φαίνεται, πως ο αλγόριθμος συγκλίνει μετά από 74 επαναλήψεις και δεν υπάρχουν λάθος ταξινομημένα σημεία.



C. Η επιφάνεια απόφασης που προκύπτει είναι η εξής:



D. Ζητείται η χρήση γραμμικού SVM για υπολογισμού γραμμικού ταξινομητή για τα ίδια δεδομένα.

Περιληπτικά, οι Μηχανές Διανυσματικής Στήριξης (SVM) βασίζονται στην έννοια του περιθωρίου (margin) . Έστω ένας γραμμικός ταξινομητής :

$$\mathbf{w^T x + w_0 = 0}$$

Το περιθώριο είναι η περιοχή μεταξύ δύο παράλληλων υπερεπιπέδων

$$\mathbf{w^T x + w_0 = 1 \text{ και } w^T x + w_0 = -1}$$

Η λύση δίνεται από ένα ζυγισμένο μέσο των σημείων εκπαίδευσης,

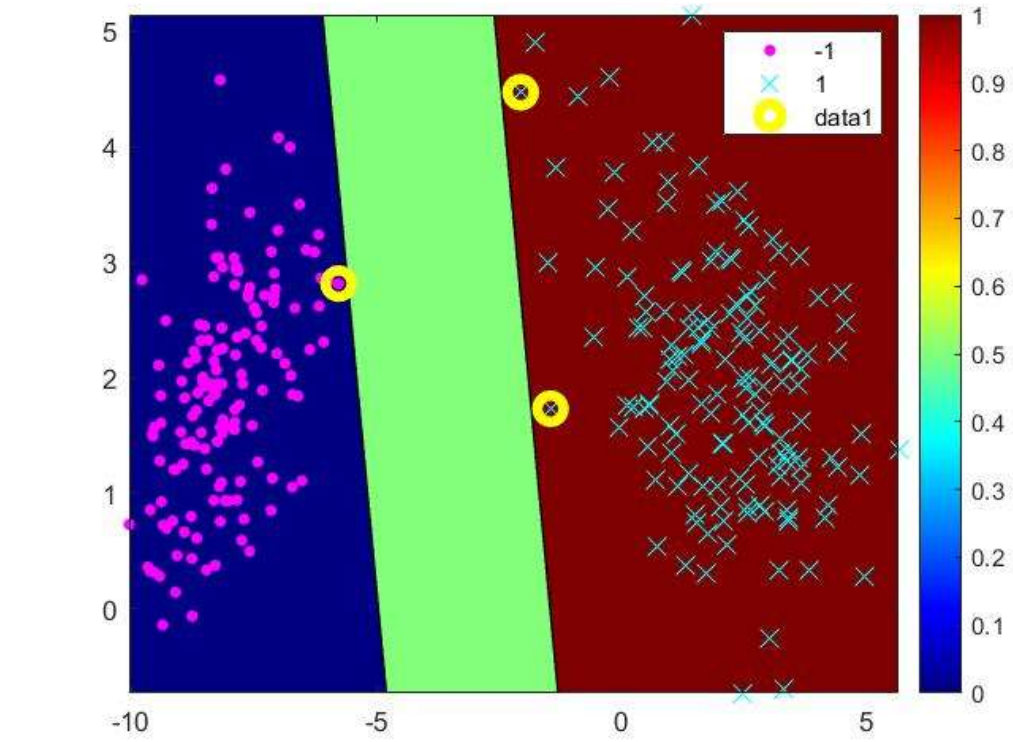
$$\mathbf{W} = \sum_{i=1}^{N} \boldsymbol{\lambda y x}$$

Οι συντελεστές λ , ή αλλιώς Lagrange πολλαπλασιαστές της βελτιστοποίησης είναι ίσες με μηδέν για όλα τα σημεία εκτός του περιθωρίου που ταυτόχρονα βρίσκονται στη σωστή πλευρά του ταξινομητή. Άρα, τα σημεία αυτά δεν συνεισφέρουν στον καθορισμό της διεύθυνσης του ταξινομητή. Τα σημεία με μη μηδενικά λ που συνεισφέρουν στη κατασκευή w καλούνται διανύσματα υποστήριξης (support vectors) .

Για την υλοποίηση του γραμμικού SVM χρησιμοποιήθηκε η fitcsvm με παραμέτρους τα διανύσματα X,Y και kernel = 'linear' , δηλαδή γραμμικό πυρήνα.

Ο κώδικας βρίσκεται στο αρχείο SVM_1.m

Τα αποτελέσματα που πήραμε είναι τα εξής:



Στο plot βλέπουμε το περιθώριο που έχει δημιουργηθεί (πράσινη περιοχή) και τα support vectors (yellow circles).

Ε. Παρατηρούμε πως ο αλγόριθμος perceptron σταματάει μόλις τα δεδομένα ταξινομηθούν σωστά ενώ ο SVM σταματάει όταν βρει το καλύτερο επίπεδο που έχει το μέγιστο περιθώριο. Από τα plot βλέπουμε πως ο SVM δίνει μία καλύτερη ακρίβεια για την σωστή ταξινόμηση μελλοντικών δειγμάτων κάτι που οφείλεται στο περιθώριο.
Επίσης, ο αλγόριθμος perceptron δουλεύει μόνο σε διαχωρίσιμες κλάσεις ενώ ο SVM μπορεί να δουλέψει και σε μη γραμμικώς διαχωρίσιμες κλάσεις και βρίσκει το ελάχιστο κόστος της διαχώρισης (soft margin).

ΑΣΚΗΣΗ 2

Η εργασία 2 πραγματοποιήθηκε σε jupyter notebook.

A. Ζητείται να χωριστεί το σύνολο δεδομένων σε σύνολο εκπαίδευσης, επικύρωσης και δοκιμής με αναλογία 40:15:15, με τυχαία επιλογή δεδομένων και ίδια αναλογία μεταξύ κλάσεων σε κάθε σύνολο.
Για το σκοπό αυτό, αρχικά φορτώθηκε το seeds dataset με την εντολή pd.read_csv() και στη συνέχεια εισάχθηκαν τίτλοι για κάθε στήλη όπου:

- 'A' - area of wheat kernel
- 'P' – perimeter of wheat kernel
- 'C' – compactness
- 'L' – length of kernel
- 'W'- width of kernel
- 'AC' – asymmetry coefficient
- 'LK' – length of kernel groove
- 'Classes' – classes 1,2,3 (Kama, Rosa, Canadian)

Στη συνέχεια, διαχωρίστηκε ο πίνακας σε X (μόνο δεδομένα) και Y (μόνο κλάσεις) και με την εντολή train_test_split διασπάστηκαν με αναλογία 40:15:15 με shuffle και παράγοντα stratify (παίρνει ίδια αναλογία random data από κάθε κλάση).

B. Ζητείται η χρήση γραμμικού SVM για εκπαίδευση ταξινομητή που διαχωρίζει την κλάση Kama από τη Canadian και στη συνέχεια η χρήση του validation test για την ρύθμιση του box constraint (C) και για την καλύτερη τιμή του, εφαρμογή στον ταξινομητή και εκπαίδευση με test set.
Ομοίως με το ερώτημα Α ορίζονται ξανά τα X,Y απλά αυτή τη φορά χωρίς να περιέχουν την κλάση 2. Αυτό πραγματοποιείται με την εντολή:
data_13 = data[data['Classes'] != 2]
Στη συνέχεια γίνεται πάλι διαχωρισμός με 40:15:15 αναλογίες και με τις εντολές
grid = GridSearchCV(SVC(),param_grid,verbose=100)
grid.fit(Xtrain,Ytrain) πρακτικά ψάχνουμε από ένα πίνακα με διάφορες τιμές C
param_grid = {'C':[0.1,1,10,100,1000,10000]} και μας επιστρέφει το καλύτερο box constraint , στη δική μας περίπτωση C=100. Τέλος, μέσω της εντολής predict(Xval) παίρνουμε τις προβλέψεις για τις κλάσεις όταν χρησιμοποιούμε το σετ validate και με την εντολή accuracy_score για τα y_grid_predictions και Yval (πραγματικές τιμές σετ validate για κλάσεις) παίρνουμε 96.67% ακρίβεια.
Για C=100 κάνουμε fit το μοντέλο svm2 για Xtrain,Ytrain και predict στα Xtest τώρα. Με την χρήση της εντολής accuracy_score για Ytest και y_predictions έχουμε error_rate=1-accuracy_test και ίσο με 0.1.

C. Ζητείται να γίνουν 5 τυχαίοι διαμερισμοί των δεδομένων. Για να γίνει αυτό χρησιμοποιήθηκε *η StratifiedKFold* με n_splits =5. Στη συνέχεια, μέσω των δεικτών που παράχθηκαν πήραμε τα νέα πλέον Xtrain,Ytrain,Xtest,Ytest. Ομοίως με το ερώτημα Β υπολογίζουμε το error_rate. Για να υπολογίσουμε τη μέση τιμή χρησιμοποιήθηκε η εντολή *np.array(error_rate).mean()* και έφερε ως αποτέλεσμα 0.179 και για να υπολογιστεί η τυπική απόκλιση χρησιμοποιήθηκε η εντολή *np.array(error_rate).std()* και μετρήθηκε 0.

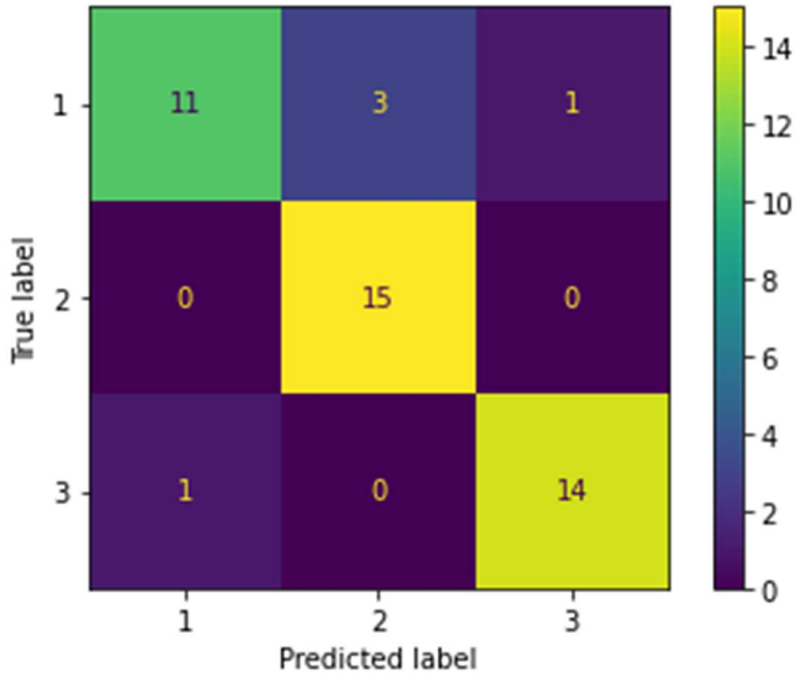
D. Για μη γραμμικό πυρήνα τα σφάλματα που πετύχαμε για τις διάφορες συναρτήσεις πυρήνα ήταν τα εξής:

- Poly kernel – σφάλμα 0.214
- Rbf kernel – σφάλμα 0.25
- Sigmoid kernel – σφάλμα 0.536

Παρατηρούμε πως το καλύτερο σφάλμα το δίνει ο πυρήνας poly. Ένας πιθανός λόγος είναι πως οι συναρτήσεις πυρήνα linear, poly καθώς διαφέρουν ελάχιστα δίνουν το καλύτερο σφάλμα. Καθώς η rbf δημιουργεί καμπύλες γύρω από τα δεδομένα και τις αθροίζει για να φτιάξει την επιφάνεια απόφασης.

E. Χρήση γραμμικού SVM για εκπαίδευση και των 3 κλάσεων με προσέγγιση one vs one και καταμέτρηση ψήφων.
Αρχικά, χωρίστηκαν τα X_initial,Y_initial ανά δύο κλάσεις δηλαδή δημιουργήθηκαν datasets για X_{12},X_{13},X_{23}
Στη συνέχεια έγινε train_test_split κάθε dataset που δημιουργήθηκε και δημιουργήθηκε μοντέλο svm για κάθε συνδυασμό κλάσεων.
Άρα, έχουμε 3 μοντέλα svm_{12},svm_{13},svm_{23}.
Στη συνέχεια, χρησιμοποιήθηκε η εντολή VotingClassifier για την καταμέτρηση ψήφων. Ανάλογα με την συγκέντρωση των ψήφων κάνουμε fit το μοντέλο clf_voting και predict με τα Xtest1 και η ακρίβεια βγήκε 88.88%

Ο πίνακας confusion :



Ο πίνακας σύγχυσης ουσιαστικά δείχνει την απόδοση της ταξινόμησης. Κανονικά, σε μία πολύ καλή ταξινόμηση θα έπρεπε όλα τα στοιχεία να είναι μηδέν εκτός από αυτά της κύριας διαγωνίου.

Παρατηρούμε πως οι τιμές που είναι εκτός της κύριας διαγωνίου και δεν είναι μηδέν μας δείχνουν τις κλάσεις που μοίαζουν περισσότερο και δεν μπορούν να ταξινομηθούν με υψηλή ακρίβεια.