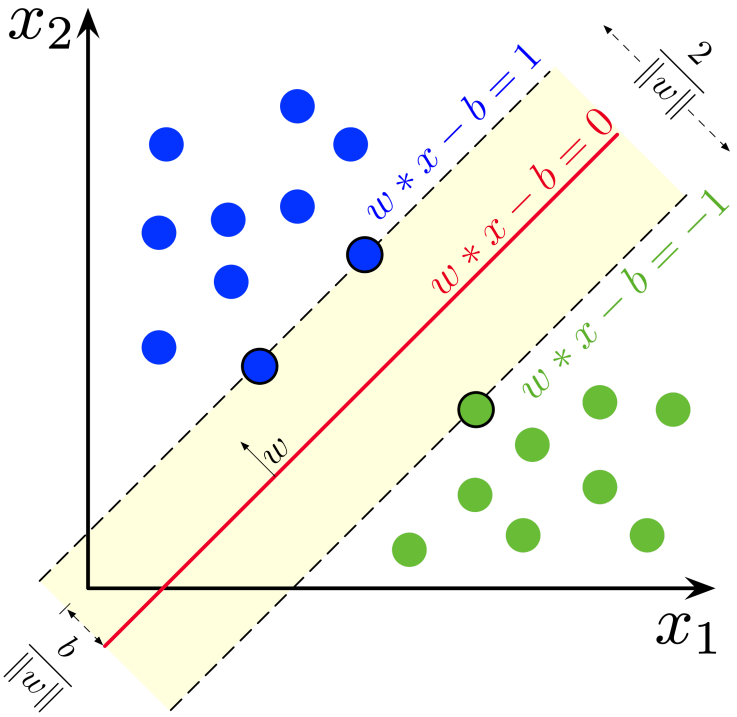
Αναγνώριση Προτύπων

2021-2022

Εργασία 6η

SUPPORT VECTOR MACHINES



Καρυπίδης Ευστάθιος 57556

20/12/21, Ξάνθη

Για την υλοποίηση των ερωτημάτων της εργασίας έγινε χρήση γλώσσας python και των βιβλιοθηκών numpy,scipy και matplotlib.

Άσκηση 1

Δίνεται το πρόβλημα της XOR δηλαδή:

$$\omega_1 = [(0,0),(1,1)] \text{ με } t_1 = -1$$
$$\omega_2 = [(0,1),(1,0)] \text{ με } t_1 = 1$$

Ζητείται να χωριστούν οι 2 κατηγορίες χρησιμοποιώντας ένα Non-Linear SVM με συνάρτηση Kernel:

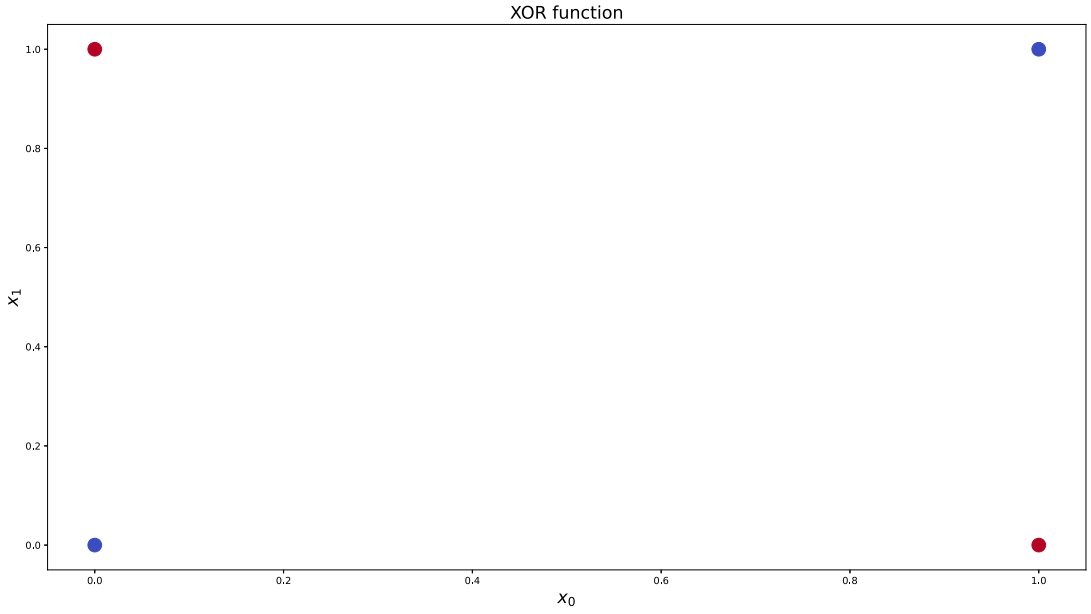
$$\Phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T \text{ και}$$
$$K(x,y) = \Phi^T(x) \cdot \Phi(y) = (x_1y_1 + x_2y_2)^2 = (x^T \cdot y)^2$$

όπου:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Αρχικά, ορίζουμε τα δεδομένα του προβλήματος (στο αρχείο python) υπάρχει και ο κώδικας για το plot:

```
# Define XOR data
X = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
Y = np.array([-1, 1, 1, -1]).reshape(-1, 1)
```



Προφανώς τα δεδομένα είναι μη γραμμικά διαχωρίσιμα. Στα πλαίσια της συγκεκριμένης άσκησης για την υλοποίηση των SVM θα χρησιμοποιηθεί η βιβλιοθήκη sklearn. Ειδικότερα υπάρχουν τα παρακάτω kernels:

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$, where r is specified by `coef0`.

Όπως βλέπουμε παρακάτω η κλαση SVC υλοποιεί τα SVM όπως ακριβώς είδαμε στη θεωρία και βασίζονται στο λογισμικό LIBSVM -- A Library for Support Vector Machines.

1.4.7.1. SVC

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, in two classes, and a vector $y \in \{1, -1\}^n$, our goal is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by $\text{sign}(w^T\phi(x) + b)$ is correct for most samples.

SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

Intuitively, we're trying to maximize the margin (by minimizing $||w||^2 = w^T w$), while incurring a penalty when a sample is misclassified or within the margin boundary. Ideally, the value $y_i(w^T\phi(x_i) + b)$ would be ≥ 1 for all samples, which indicates a perfect prediction. But problems are usually not always perfectly separable with a hyperplane, so we allow some samples to be at a distance ζ_i from their correct margin boundary. The penalty term `C` controls the strength of this penalty, and as a result, acts as an inverse regularization parameter (see note below).

The dual problem to the primal is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to } y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

where e is the vector of all ones, and Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. The terms α_i are called the dual coefficients, and they are upper-bounded by C. This dual representation highlights the fact that training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ: see kernel trick.

Once the optimization problem is solved, the output of decision_function for a given sample x becomes:

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b,$$

and the predicted class correspond to its sign. We only need to sum over the support vectors (i.e. the samples that lie within the margin) because the dual coefficients α_i are zero for the other samples.

These parameters can be accessed through the attributes `dual_coef_` which holds the product $y_i \alpha_i$, `support_vectors_` which holds the support vectors, and `intercept_` which holds the independent term b

Αξίζει να σημειωθεί εδώ ότι, όπως αναφέρεται στη τελευταία πρόταση το attribute `dual_coef_` επιστρέφει κατευθείαν το γινόμενο $y_i a_i$ δηλαδή τα πρόσημα που θα εμφανιστούν είναι σωστά. Ακόμη, όπως αναφέρεται το bias θα βρισκεται στο attribute `intercept_`. Τέλος θέλει προσοχή στο `support_vectors` καθως βρίσκονται τα σημεία που θα θεωρούνται support support vectors αλλά με συγκεκριμένη σειρά. Για να έχουμε τη σωστή σειρά χρησιμοποιούμε το attribute `support_`.

support_ : *ndarray of shape (n_SV)*
Indices of support vectors.

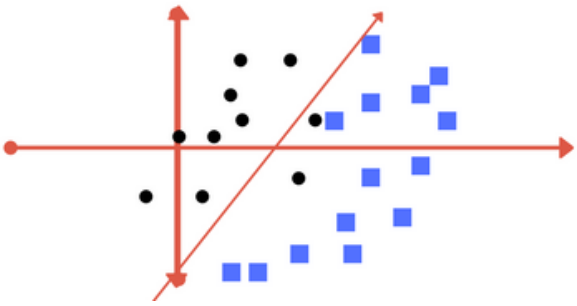
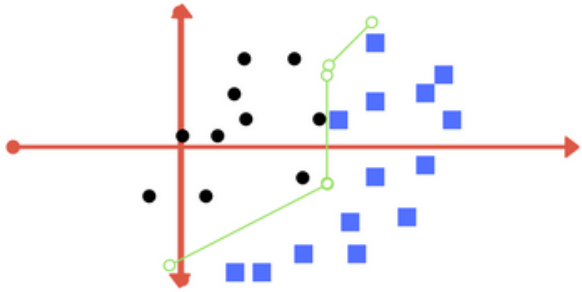
support_vectors_ : *ndarray of shape (n_SV, n_features)*
Support vectors.

Τέλος, αξίζει να δωθεί σημασία στις παραμέτρους γ και C καθώς επηρεάζουν συμαντικά το μοντέλο. Όσον αφορά την παράμετρο γ θα τεθεί 1 στις δοκιμές αλλά παρόλα αυτά το default στη συγκεκριμένη κλάση δεν είναι 1. Παρακάτω παρουσιάζεται η επηρροή αυτών των 2 παραμέτρων.

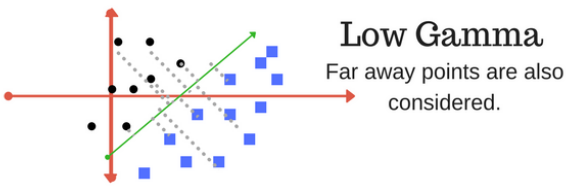
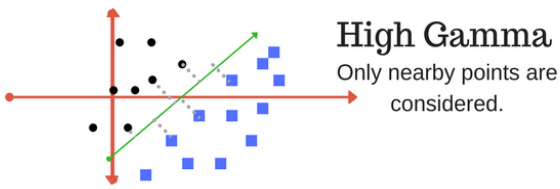
Παράμετρος C: Η παράμετρος C είναι μία παράμετρος για regularization(κανονικοποίησης). Ειδικότερα ο όρος σφάλματος ταξινόμησης η αλλιώς

$$C\sum_{i=1}^{n}\zeta_i$$

μπορεί να μεγαλώσει η να μειωθεί ανάλογα με τη τιμή της παραμέτρου C. Με άλλα λόγια η παράμετρος C μπορεί να ορίσει το κατα πόσο είναι ανεκτό το σφάλμα ταξινόμησης. Συνεπώς η επιλογή της παραμέτρου C δημιουργεί ενα trade-off μεταξύ σωστής ταξινόμησης ενάντια στη μεγιστοποίηση του περιθορίου της επιφάνειας απόφασης(margin), καθώς μικρότερες τιμές του C θα οδηγήσουν σε μεγαλύτερα περιθόρια και πιο απλή επιφάνεια/(συνάρτηση) ταξινόμησης αλλά με κόστος την πιθανή αύξηση του λάθους ταξινόμησης ενώ αντίθετα μεγάλες τιμές του C θα οδηγήσουν σε καλύτερο ποσοστό ταξινόμησης(μικρότερο σφάλμα) αλλά ταυτόχρονα μικρότερο περιθόριο. Παραστατικά στα παρακάτω 2 σχήματα, αριστερά χρησιμοποιείται χαμηλότερη τιμή του C ενώ στα δεξιά μεγαλύτερη τιμή του C. Συνεπώς, πολύ μεγάλη τιμή του C μπορεί να οδηγήσει σε overfitting ενώ πολύ μικρή τιμή του C σε underfitting.

Μικρή τιμή C	Μεγάλη τιμή C
	

Παράμετρος γ: Η παράμετρος γ είναι άλλη μία παράμετρος που επηρρεάζει πολύ τη παράμετρο του μοντέλου. Ειδικότερα η παράμετρος αυτή καθορίζει το πόσο μακριά φτάνει η επιρροή ενός μεμονωμένου δείγματος του συνόλου εκπαίδευσης, με τις χαμηλές τιμές να σημαίνουν «μακριά» και τις υψηλές που σημαίνει «κοντά». Με άλλα λόγια, μια χαμηλή τιμή στο γ θα έχει ως αποτέλεσμα σημεία πολύ μακριά από την επιφάνεια απόφασης να λαμβάνονται υπ΄οψην ενώ μία μεγάλη τιμή θα έχει ως αποτέλεσμα να λαμβάνονται υπ όψην σημεία κοντινότερα στην επιφάνεια απόφασης.

Μικρή τιμή γ	Μεγάλη τιμή γ
	

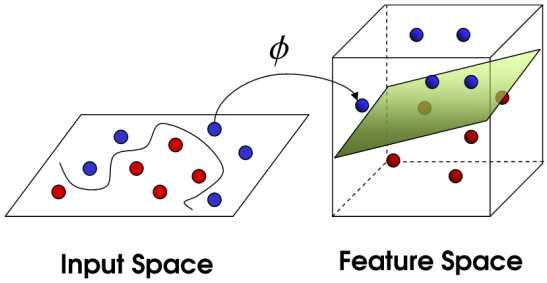
Όπως αναφέρθηκε προηγουμένως στη συγκεκριμένει εργασία θα τεθεί $\gamma = 1$ στον παρακάτω τύπου και δε θα δοκιμαστεί και αξιολογηθεί η επίδραση της στο μοντέλο.

polynomial: $(\gamma\langle x, x'\rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.

Τέλος, πριν προχωρήσουμε στη προγραμματιστική υλοποίηση αξίζει να γίνει μία συντομη αναφορά στη χρήση το kernel trick.

Kernel Trick

Πολλές φορές, όπως και στο πρόβλημα της XOR τα δεδομένα του προβλήματος δεν ειναι γραμμικά διαχωρίσιμα στον υπάρχον/τρέχον χώρο των χαρακτηριστικων. Όπως γνωρίζουμε ένας τρόπος ώστε μη γραμμικά διαχωρίσιμα δεδομένα να γίνουν γραμμικά διαχωρίσιμα είναι η προσθήκη χαρακτηριστικών μέσω της άυξησης των διαστάσεων.

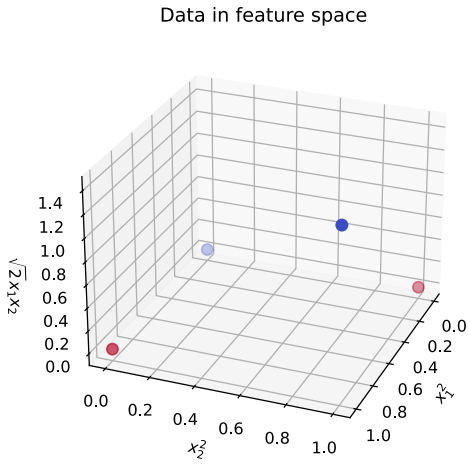


Έτσι λοιπόν έχουμε για παράδειγμα το polynomial Kernel που είναι της μορφής $K(x,y) = \Phi^T(x) \cdot \Phi(y) = (x_1y_1 + x_2y_2)^2 = (x^T \cdot y)^2$. Έχουμε δηλαδή ότι μία συνάρτηση πυρήνα (kernel) για 2 διανύσματα του αρχικού χώρου είναι ισοδύναμη με το εσωτερικό γινόμενο των διανυσμάτων στο μετασχηματισμένο χώρο. Έτσι, εκτός του οτι δε χρειάζεται να οριστεί ο μετασχηματισμός $\Phi(x)$ των δεδομένων σε νέο χώρο, ο υπολογισμός των εσωτερικών γινομένων γίνεται στον αρχικό χώρο και δεδομένου ότι αυτός είναι συνήθως λιγότερων διαστάσεων, απαιτούνται λιγότεροι υπολογισμοί. Όπως φαίνεται λοιπόν από την μορφή της συνάρτησης διάκρισης ο καθορισμός του $K(x, x_n)$ και των συντελεστών/πολλαπλασιαστών Lagrange είναι αρκετός και δεν απαιτείται η εύρεση του διανύσματος βαρών

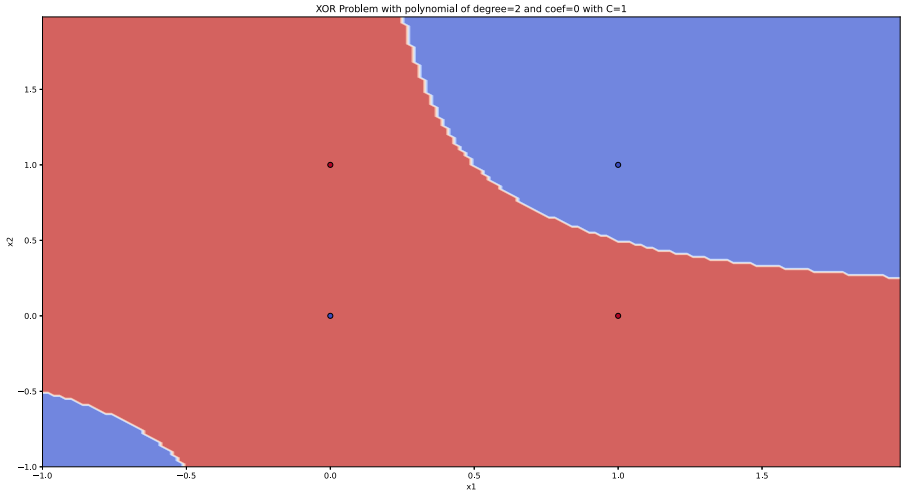
$$y(x) = \sum_{n=i}^{N} = a_n y_n K(x, x_n) + b$$

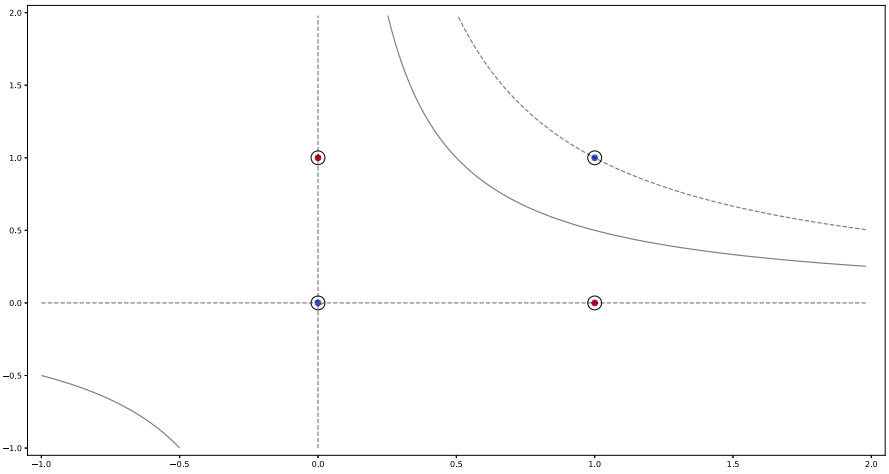
```
# SVM classifier
c = 1.0  # SVM regularization parameter
model = svm.SVC(kernel='poly', degree=2, coef0=0, C=c, gamma=1)
clf = model.fit(X, Y.ravel())
ay_values = clf.dual_coef_
bias = clf.intercept_
sv_order = clf.support_
```

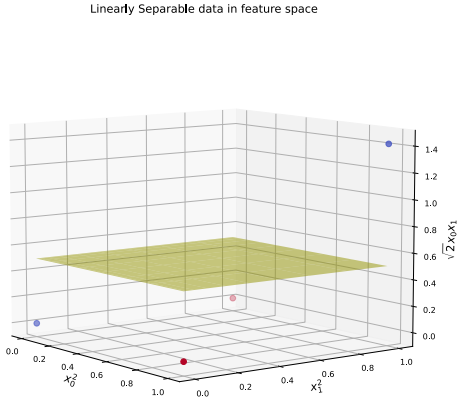
Πριν προχωρήσουμε όμως στα αποτέσματα αξίζει να δούμε πως μοιάζει ο χώρος των χαρακτηριστικών εαν χρησιμοποιηθεί στα δεδομένα η συνάρτηση μετασχηματισμού $\Phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$.



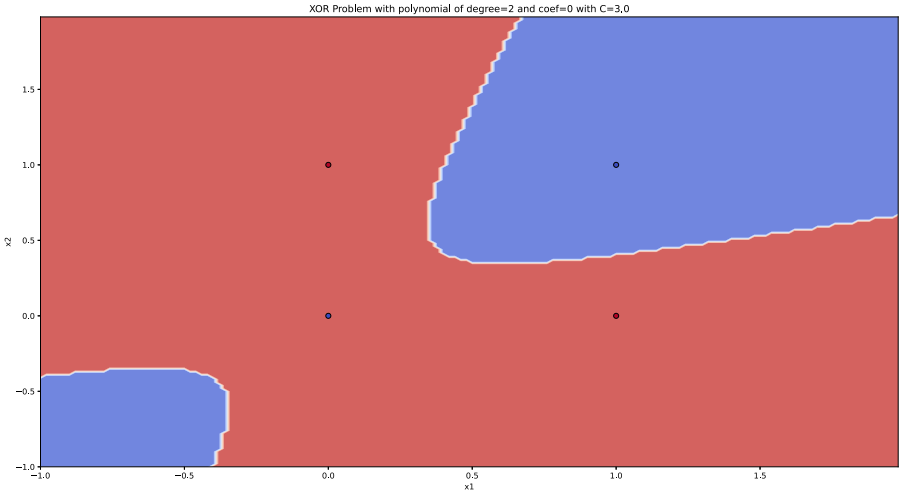
Για C = 1 έχουμε τα παρακάτω αποτελέσματα

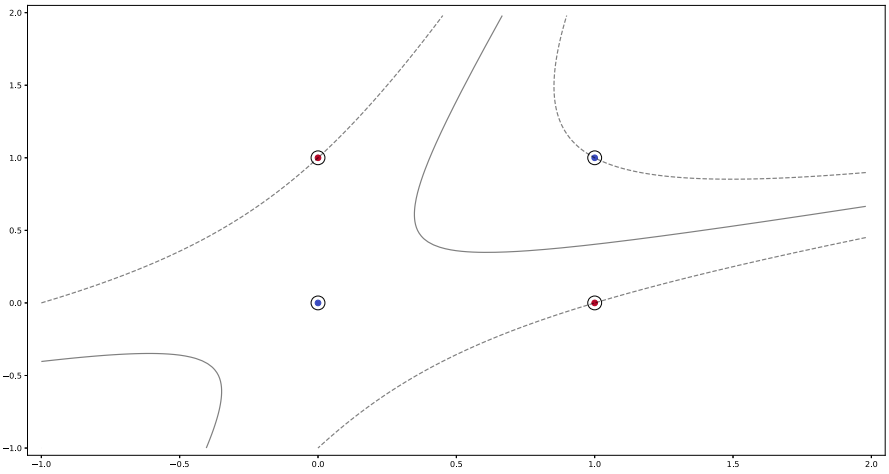




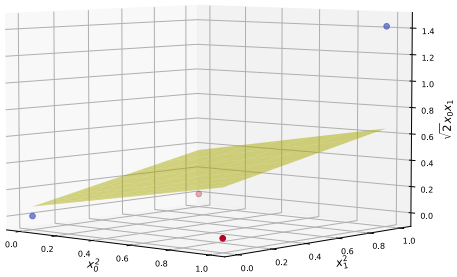


Για C = 3 έχουμε τα παρακάτω αποτελέσματα

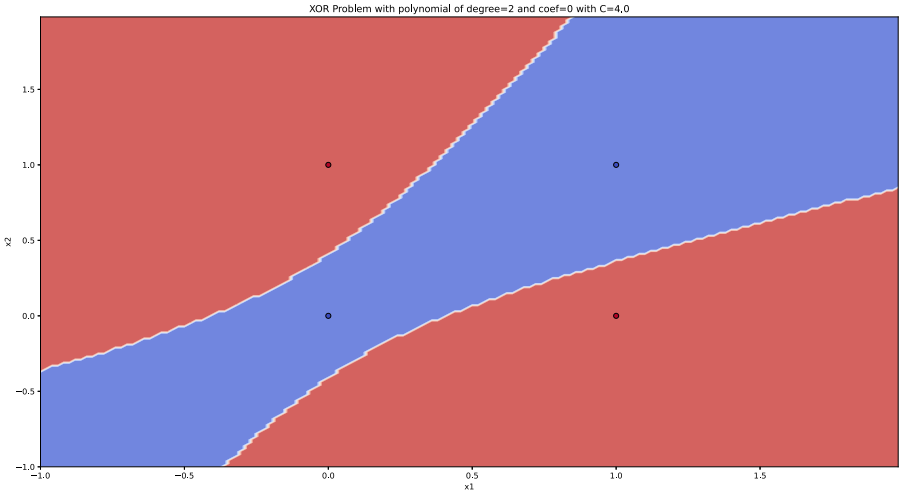


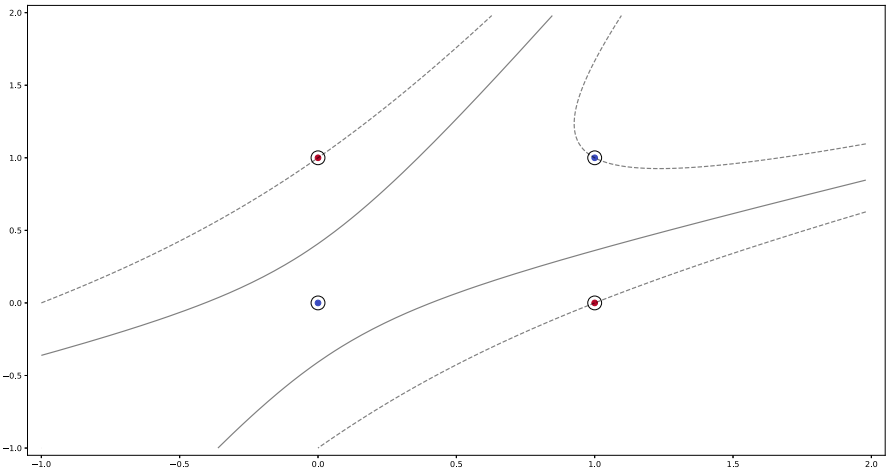


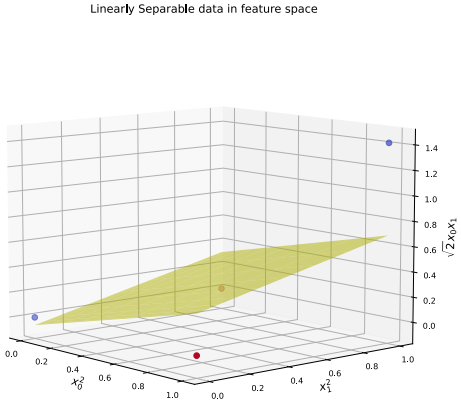
Linearly Separable data in feature space



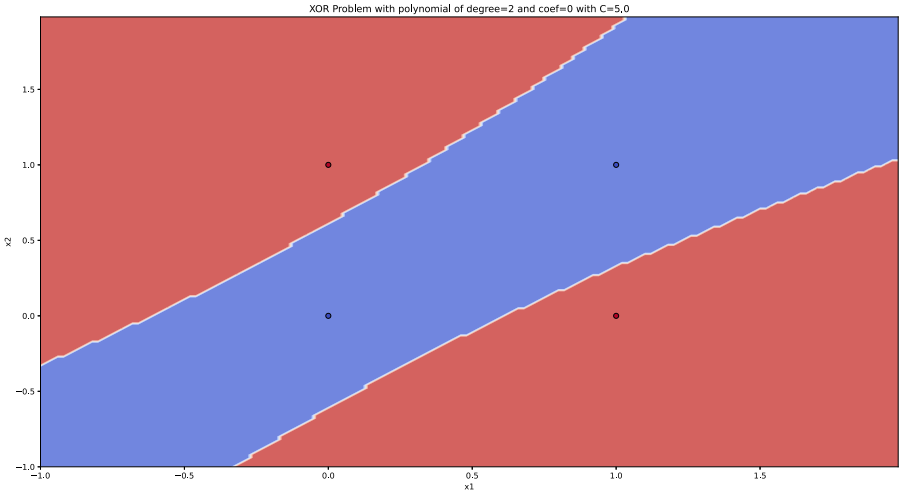
Για C = 4 έχουμε τα παρακάτω αποτελέσματα

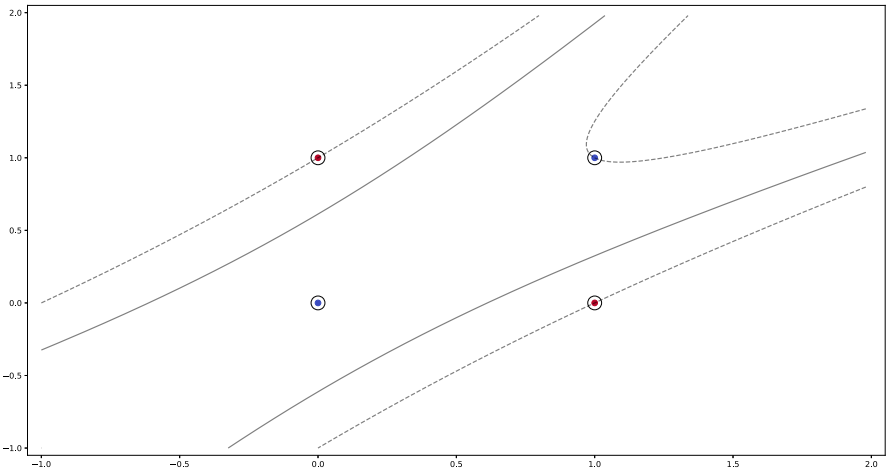


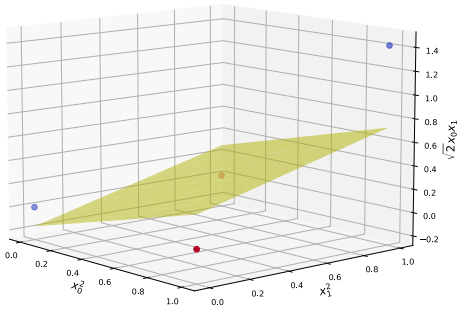




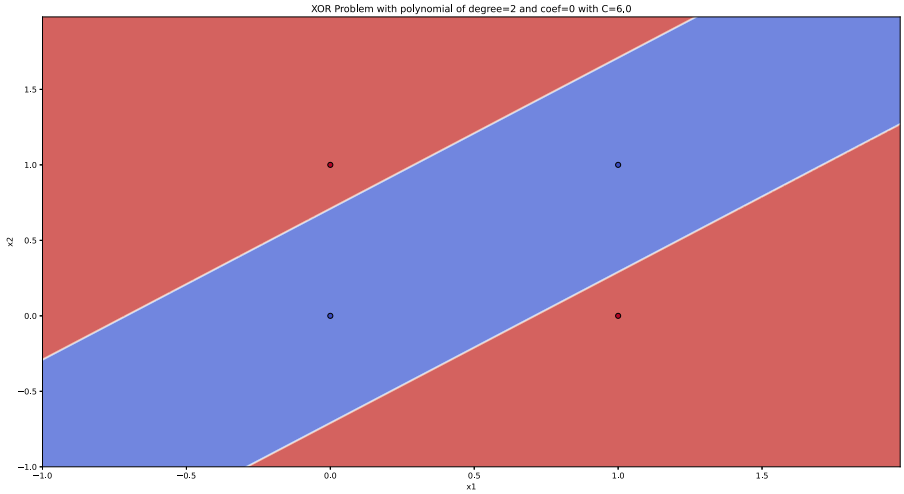
Για C = 5 έχουμε τα παρακάτω αποτελέσματα

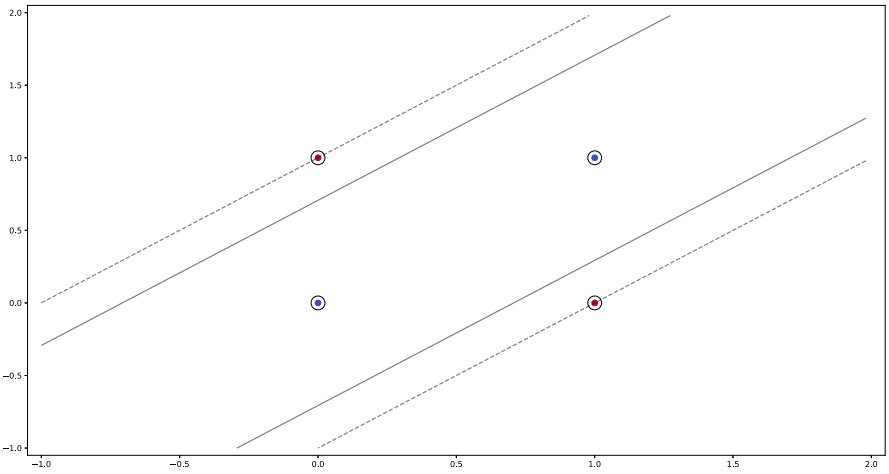


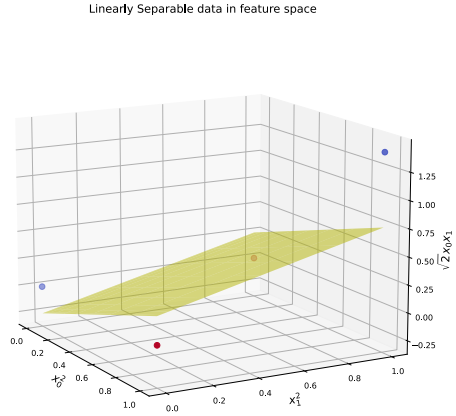




Για C = 6 έχουμε τα παρακάτω αποτελέσματα







Όπως παρατηρούμε για τιμές του C μικρότερες του 4 έχουμε σφάλμα ταξινόμησης πράγμα λογικό καθώς έτσι δεν "τιμωρείται"(δίνεται το απαραίτητο βάρος) στο παράγοντα του σφάλματος ταξινόμησης. Από τιμές 4 και πάνω το σφάλμα ταξινόμησης είναι μηδενικό ενώ από τι τιμή $C = 6$ και πάνω η μορφή της συνάρτησης διάκρισης και τα αποτελέσματα για τους συντελεστές δε παρουσιάζουν κάποια αλλαγή. Αξίζει να σημειωθεί και πάλι ότι στις συγκεκριμένες δοκιμές η τιμή του γ παραμένει σταθερή και ίση με 1. Στη πραγματικότητα μπορούμε να δοκιμάσουμε και τιμές μεγαλύτερες του 1 για τη παράμετρο γ και να δούμε όπως αναμένεται το μηδενικό σφάλμα ταξινόμησης να έρχεται για μικρότερη τιμή του C. Παρακάτω παρουσιάζονται τα αποτελέσματα από το κώδικα.

```
a*y values =  [-5.99878067 -1.99959356  3.99959356  3.99878067]
bias [-0.99979678]
support vector order [0 3 1 2]
weights [ 1.99918712  2.         -2.82785233]
```

Έχουμε δηλαδή ότι:

$$a_1 = 6$$
$$a_2 = 4$$
$$a_3 = 4$$
$$a_4 = 2$$
$$w = [2, 2, -2.82785233]^T, \;\; b = -1$$

Η συνάρτηση διάκρισης θα είναι της μορφής:

$$f(y) = -6(0y_1 + 0y_2)^2 + 4(0y_1 + 1y_2)^2 + 4(1y_1 + 0y_2)^2 - 2(1y_1 + 1y_2)^2 - 1 \Rightarrow$$
$$f(y) = 4y_2^2 + 4y_1^2 - 2y_1^2 - 4y_1y_2 - 2y_2^2 - 1 \Rightarrow$$
$$f(y) = 2y_1^2 + 2y_2^2 - 4y_1y_2 - 1 \Rightarrow$$
$$f(y) = 2y_1^2 + 2y_2^2 - 2\sqrt{2}\sqrt{2}y_1y_2 - 1$$

Καταλήγουμε λοιπόν και στη συνηθισμένη μορφή καθώς και επιβεβαιώνονται τα αποτελέσματα που προκύπτουν από το κώδικα:

$$f(y) = \begin{bmatrix} 2, & 2, & -2\sqrt{2}, & -1 \end{bmatrix} \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1y_2 \\ 1 \end{bmatrix}$$

Αξίζει να σημειωθεί ότι βλέπουμε αυτές τις τιμες(πχ -5.99878067 αντί για -6) καθώς το σύστημα στη πραγματικότητα δε λύνεται και συνεπώς γίνονται αυτές οι μικρές αλλαγές προκειμένου να μπορεί να λυθεί από τη βιβλιοθήκη. Αν κάνουμε τις πράξεις του $K(x, y) = (x_1y_1 + x_2y_2)^2$ στο χέρι έχουμε:

(x0,x1)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	0	0	0	0
(0,1)	0	1	0	1
(1,0)	0	0	1	1
(1,1)	0	1	1	2

Θα έχουμε ότι:

$$L(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_i - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j y_k y_j K(x_k, x_j) \Rightarrow$$

$$L(\boldsymbol{\alpha}) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\left(\alpha_2^2 + \alpha_3^2 - 2\alpha_2\alpha_4 - 2\alpha_3\alpha_4 + 2\alpha_4^2\right)$$

Παίρνοντας τις μερικές παραγώγους έχουμε:

$$\frac{\partial L}{\partial \alpha_1} = 0 \Longrightarrow 1 - 0 = 0\,?$$
$$\frac{\partial L}{\partial \alpha_2} = 0 \Longrightarrow 1 - \alpha_2 + \alpha_4 = 0$$
$$\frac{\partial L}{\partial \bar{\alpha}_3} = 0 \Longrightarrow 1 - \alpha_3 + \alpha_4 = 0$$
$$\frac{\partial L}{\partial \alpha_4} = 0 \Longrightarrow 1 + \alpha_2 + \alpha_3 - 2\alpha_4 = 0$$

Εφόσον για το a_1 δεν έχουμε κάποια εξίσωση το θέτουμε αυθαίρετα σύμφωνα με το όριο $a_i \leq C$ ότι $a_1 = 6$. Για τα υπόλοιπα έχουμε ότι:

$$1 - \alpha_2 + \alpha_4 = 0 \Rightarrow \alpha_2 - 1 = \alpha_4$$
$$1 - \alpha_3 + \alpha_4 = 0 \Rightarrow \alpha_3 - 1 = \alpha_4$$

Άρα έχουμε ότι $\alpha_2 = \alpha_3$. Έστω ότι επιλεγουμε $a2 = a3 = 4$. Στη συνέχεια παρατηρούμε ότι και η 4η εξίσωση που έχουμε για το $\alpha 4$ οδηγεί και αυτή σε μία μη αληθή σχέση ότι $1 = 0$?. Τέλος, πρέπει να ισχυεί ότι $\sum_{k=1}^{n} \alpha_i y_i = 0$ άρα έχουμε ότι

$$\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3 + \alpha_4 y_4 = 0 \Rightarrow$$
$$-6 + 4 + 4 - \alpha_4 = 0 \Rightarrow$$
$$\alpha_4 = 2$$

Άρα επιβεβαιώνονται τα αποτελέσματα του κώδικα και $a_1 = 6, a_2 = 4, a_3 = 4, a_4 = 2$. Τελικά, το κόστος είναι:

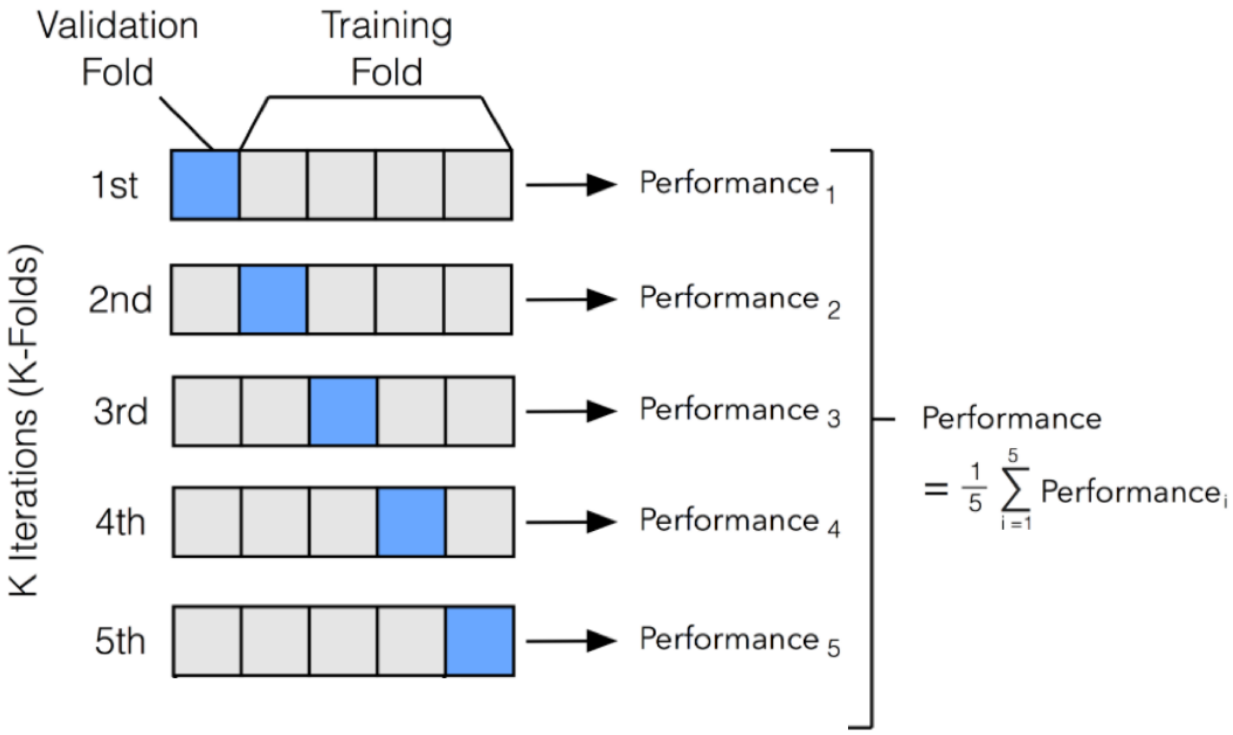
$$L(\boldsymbol{\alpha}) = 16 - \frac{1}{2}(16 + 16 - 16 - 16 + 8) = 16 - 4 = 12$$

Τέλος για το $||\, w\,||$ και το margin θα έχουμε:]

$$||\, w\,|| = \sqrt{2^2 + 2^2 + (-2\sqrt{2})^2} = \sqrt{4 + 4 + 8} = 4$$
$$d = \frac{1}{||\, w\,||} = 0.25$$

Άσκηση 2

Με βάση εκφώνηση ζητείται να χωρίζουμε το dataset σε train/validation/testing σε αναλογία 60%/20%/20%. Δηλαδή να εκπαιδεύσουμε το μοντέλο στο 60% των διαθέσιμων δεδομένων, να κάνουμε επιλογή υπερπαραμέτρων με βάση την δοκιμή στο validation set(20%) και τέλος να δοκιμάσουμε το μοντέλο στο testing δηλαδή το υπόλοιπο (20%). Θεώρησα καλύτερη γενίκευση να χωρίσουμε το dataset σε train και test (με ίδιο μέγεθος το test) και στο training dataset να κάνουμε K-Folds Cross Validation και να βλέπουμε το average του validation accuracy για την κάθε υλοποίηση έτσι ώστε να βλέπουμε ποιο μοντέλο θα έκανε καλύτερο Generalization. Το K-folds Cross validation ουσιαστικά χωρίζει το train dataset σε K groups(folds) και το μοντέλο εκπαιδεύεται στα k-1 groups και τεστάρεται σε 1 διαφορετικό validation set. Έτσι μπορούμε να δούμε ποιο μοντέλο εμφανίζει την καλύτερη γενίκευση καλύτερα με βάση το average accuracy και το standard deviation αυτών. Για παράδειγμα εάν χρησιμοποιούσαμε K=5 τότε θα είχαμε κάτι τέτοιο.



Μια σημαντική διαφορά είναι ότι στη περίπτωση του K-fold εδώ θα κάνουμε εκπαίδευση στο 80% τελικά των δεδομένων και συνεπώς θα έχουμε καλύτερα αποτελέσματα στα predictions των test data. Απομένει, λοιπόν να επιλέξουμε τη κατάλληλη τιμή του K έτσι ώστε να έχουμε το ποσοστό 20% για validation που ζητείται. Στην εκφώνηση έχουμε 10 δείγματα ανά κάθε κατηγορία άρα 30 συνολικό validation test. Εφόσον το συνόλικό training είναι $0.8 \cdot 150 = 120$ άρα θα έχουμε 120/30=4. Συνεπώς πρέπει να επιλεχθεί τιμή K = 4.

Ερώτημα A,B - Ταξινόμηση 2 κατηγοριών (Class 2 vs Class13)

Με χρήση 3 χαρακτηριστικών(1,2,4)

Αρχικά ζητείται να ενώσουμε τις κλάσεις $\omega 1$ και $\omega 3$ σε μία και να επιλύσουμε το πρόβλημα ταξινόμησης 2 κατηγοριών. Στη πρώτη δοκιμή θα χρησιμοποιηθούν τα 3 μόνο χαρακτηριστικά. Ο κώδικας για την συγκεκριμένη άσκηση βρίσκεται στο αρχείο **HW6_Exercise2_AB_3features.py**. Πρώτο βήμα είναι να φορτώσουμε και να επιλέξουμε τα δεδομένα (χαρακτηριστικά) που θέλουμε καθώς και να ρυθμισουμε κατάλληλα τα Labels. Στη συνέχεια δημιουργούμε το training και test dataset προσέχοντας να έχουμε τον ίδιο αριθμό δεδομένων από κάθε κλάση με την χρήση της παραμέτρου stratify. Τέλος τρέχουμε το τον κώδικα για διαφορετικά kernel και παραμέτρους. Πιο συγκεκριμένα έγινε χρήση των Kernels= Linear, Sigmoid,RBF, Polynomial για διαφορετικές τιμές του c (και στην περίπτωση του polynomial και διαφορετικά degrees). Η παράμετρος gamma κρατήθηκε σταθερή στην default τιμή της που φάνηκε να λειτουργεί καλύτερα και είναι $g = \frac{1}{n\_features \cdot X.var()}$. Τα αποτελέσματα είναι τα εξής:

Classification of class 2 with classes 1 and 3 using features 1,2,4

USING LINEAR SVM

SVM	Kernel Type	C	Validation Accuracy	Test Accuracy
Linear SVM	Linear	0.5	0.683 ± 0.016	0.73333
Linear SVM	Linear	1	0.7 ± 0.040	0.73333
Linear SVM	Linear	10	0.716 ± 0.037	0.76667
Linear SVM	Linear	100	0.725 ± 0.059	0.76667
Linear SVM	Linear	1000	0.725 ± 0.043	0.76667

USING NON LINEAR SVM

SVM	Kernel Type	C	Degree	Validation Accuracy	Test Accuracy
Non LINEAR SVM	Sigmoid	0.5	-	0.666 ± 0.0	0.66667
Non LINEAR SVM	Sigmoid	1	-	0.666 ± 0.0	0.66667
Non LINEAR SVM	Sigmoid	10	-	0.666 ± 0.023	0.7
Non LINEAR SVM	Sigmoid	100	-	0.55 ± 0.076	0.66667
Non LINEAR SVM	Sigmoid	1000	-	0.516 ± 0.072	0.7
Non LINEAR SVM	RBF	0.5	-	0.666 ± 0.0	0.66667
Non LINEAR SVM	RBF	1	-	0.766 ± 0.023	0.9
Non LINEAR SVM	RBF	10	-	0.941 ± 0.036	0.96667
Non LINEAR SVM	RBF	100	-	0.966 ± 0.033	0.96667
Non LINEAR SVM	RBF	1000	-	0.933 ± 0.070	0.96667
Non LINEAR SVM	Polynomial	0.5	2	0.791 ± 0.054	0.86667
Non LINEAR SVM	Polynomial	1	2	0.85 ± 0.028	0.9
Non LINEAR SVM	Polynomial	10	2	0.958 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	2	0.95 ± 0.037	0.96667
Non LINEAR SVM	Polynomial	1000	2	0.941 ± 0.049	0.96667
Non LINEAR SVM	Polynomial	0.5	4	0.791 ± 0.054	0.86667
Non LINEAR SVM	Polynomial	1	4	0.85 ± 0.028	0.9
Non LINEAR SVM	Polynomial	10	4	0.958 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	4	0.95 ± 0.037	0.96667
Non LINEAR SVM	Polynomial	1000	4	0.941 ± 0.049	0.96667
Non LINEAR SVM	Polynomial	0.5	6	0.791 ± 0.054	0.86667
Non LINEAR SVM	Polynomial	1	6	0.85 ± 0.028	0.9
Non LINEAR SVM	Polynomial	10	6	0.958 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	6	0.95 ± 0.037	0.96667
Non LINEAR SVM	Polynomial	1000	6	0.941 ± 0.049	0.96667

Συμπεράσματα για Linear SVM: Όπως παρατηρούμε, τα αποτελέσματα του γραμμικού svm για 2 κλάσεις που δεν είναι γραμμικά διαχωρίσιμες είναι αρκετά κοντά στον γραμμικό ταξινομητή της 5ης άσκησης(αν θυμηθούμε ότι κάποιες ταξινομήσεις είχαν σφάλμα 25%). Αυτό βέβαια είναι σχετικά αναμενόμενο καθώς και στη περίπτωση αυτή με το Linear SVM πάλι έχουμε μια ευθεία διαχωριστική γραμμή για μη γραμμικώς διαχωρίσιμα δεδομένα 2 κλάσεων. Ακόμη, επιβεβαιώνεται ότι με την άυξηση του C έχουμε μικρότερο σφάλμα ταξινόμησης καθώς τιμωρείται περισσότερο ο παράγοντας σφάλματος.

Συμπεράσματα για Non Linear SVM:

- **Sigmoid Kernel:** Τα αποτελέσματα δεν είναι πολύ καλά και μάλιστα είναι πολλές φορές χειρότερα από του Linear SVM.
- **RBF Kernel:** Όπως φαίνεται από τα αποτελέσματα, χρησιμοποιώντας μη γραμμικό SVM με **RBF** kernel παρατηρούμε ότι έχουμε πάρα πολύ καλά αποτελέσματα και μάλιστα έχουμε καλή ικανότητα generalization βλέποντας και το average validation accuracy. Όσο αυξάνουμε το C φαίνεται να έχουμε καλύτερα αποτελέσματα εκτός της περίπτωσης με C=100 που παρατηρείται μια μείωση. Εδώ τα αποτελέσματα είναι πολύ καλύτερα σε σχέση με τον γραμμικό ταξινομητή της προηγούμενης εργασίας.
- **Polynomial Kernel:** Όπως βλέπουμε και το πολυωνυμικό kernel έχει πολύ καλά αποτελέσματα. Ειδικότερα το βέλτιστο ποσοστό σωστής ταξινόμησης είναι 96,7%. Επίσης παρατηρούμε ότι για το συγκεκριμένο πρόβλημα, η αύξηση του βαθμού του πολυωνύμου δεν αλλάζει ειδιαίτερα την ήδη πολύ υψηλή ακρίβεια.

Στην συνέχεια εκτελούμε πειράματα για χρήση 4 χαρακτηριστικών. Αναμένουμε τα αποτελέσματα να είναι ελάχιστα καλύτερα.

Με χρήση 4 χαρακτηριστικών

Ο κώδικας για την συγκεκριμένη άσκηση βρίσκεται στο αρχείο **HW6_Exercise2_AB_4features.py**. Τα αποτελέσματα στη περίπτωση αυτή είναι:

```
Classification of class 2 with classes 1 and 3 using features 1,2,3,4
                        USING LINEAR SVM
+------------+-------------+------+----------------------+---------------+
|    SVM     | Kernel Type |  C   | Validation Accuracy | Test Accuracy |
+------------+-------------+------+----------------------+---------------+
| Linear SVM |    Linear   | 0.5  |    0.675 ± 0.028    |    0.73333    |
| Linear SVM |    Linear   |  1   |    0.725 ± 0.098    |    0.73333    |
| Linear SVM |    Linear   |  10  |    0.758 ± 0.098    |    0.76667    |
| Linear SVM |    Linear   | 100  |    0.774 ± 0.098    |    0.76667    |
| Linear SVM |    Linear   | 1000 |    0.783 ± 0.110    |    0.76667    |
+------------+-------------+------+----------------------+---------------+
```

Όπως βλέπουμε με χρήση γραμμικού ταξινομητή δεν παρουσιάζεται διαφορά με τη προσθήκη και του ενός επιπλέον χαρακτηριστικού.

Στην επόμενη σελίδα παρουσιάζονται τα αποτέλεσματα για μη γραμμικά SVM.

USING NON LINEAR SVM

SVM	Kernel Type	C	Degree	Validation Accuracy	Test Accuracy
Non LINEAR SVM	Sigmoid	0.5	-	0.666 ± 0.0	0.66667
Non LINEAR SVM	Sigmoid	1	-	0.666 ± 0.0	0.66667
Non LINEAR SVM	Sigmoid	10	-	0.525 ± 0.086	0.56667
Non LINEAR SVM	Sigmoid	100	-	0.45 ± 0.068	0.5
Non LINEAR SVM	Sigmoid	1000	-	0.45 ± 0.068	0.5
Non LINEAR SVM	RBF	0.5	-	0.924 ± 0.049	0.9
Non LINEAR SVM	RBF	1	-	0.941 ± 0.049	0.93333
Non LINEAR SVM	RBF	10	-	0.983 ± 0.016	0.93333
Non LINEAR SVM	RBF	100	-	0.95 ± 0.055	0.96667
Non LINEAR SVM	RBF	1000	-	0.95 ± 0.055	0.96667
Non LINEAR SVM	Polynomial	0.5	2	0.975 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	2	0.975 ± 0.043	0.96667
Non LINEAR SVM	Polynomial	10	2	0.966 ± 0.040	0.96667
Non LINEAR SVM	Polynomial	100	2	0.966 ± 0.040	0.93333
Non LINEAR SVM	Polynomial	1000	2	0.958 ± 0.036	0.96667
Non LINEAR SVM	Polynomial	0.5	4	0.975 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	4	0.975 ± 0.043	0.96667
Non LINEAR SVM	Polynomial	10	4	0.966 ± 0.040	0.96667
Non LINEAR SVM	Polynomial	100	4	0.966 ± 0.040	0.93333
Non LINEAR SVM	Polynomial	1000	4	0.958 ± 0.036	0.96667
Non LINEAR SVM	Polynomial	0.5	6	0.975 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	6	0.975 ± 0.043	0.96667
Non LINEAR SVM	Polynomial	10	6	0.966 ± 0.040	0.96667
Non LINEAR SVM	Polynomial	100	6	0.966 ± 0.040	0.93333
Non LINEAR SVM	Polynomial	1000	6	0.958 ± 0.036	0.96667

Όπως παρατηρούμε έχουμε μία ελάχιστη βελτίσωη σε κάποιες περιπτώσεις. Ειδικότερα βλέπουμε ότι πλέον για kernels RBF και Polynomial η ακρίβεια αρχίζει από τιμές της τάξης του 93.3% δηλαδή έχουμε πολύ καλή ταξινόμηση.

Ερώτημα C,D - Ταξινόμηση One vs All

Με χρήση 3 χαρακτηριστικών(1,2,4)

Εδώ πλέον έχουμε 3 κλάσεις και θα χρησιμοποιήσουμε την τακτική one vs all ή αλλιώς one vs rest όπου ουσιαστικά κάνουμε train έναν classifier για κάθε κλάση, ο οποίος διαχωρίζει αυτή τη κλάση από όλες τις υπόλοιπες μαζί όπου τις θεωρούμε 1 κλαση. Στον κώδικα η μόνη αλλαγή που χρειάζεται είναι να δώσουμε στο μοντέλο τη παράμετρο `decision_function_shape="ovr"` όπου το ovr σημαίνει one versus all. Ο κώδικας για την συγκεκριμένη άσκηση βρίσκεται στο αρχείο **HW6_Exercise2_CD_3features.py**. Τα αποτελέσματα είναι:

One vs All Classification using features 1,2,4

USING LINEAR SVM

SVM	Kernel Type	C	Validation Accuracy	Test Accuracy
Linear SVM	Linear	0.5	0.95 ± 0.037	0.96667
Linear SVM	Linear	1	0.966 ± 0.040	0.93333
Linear SVM	Linear	10	0.95 ± 0.037	0.93333
Linear SVM	Linear	100	0.95 ± 0.028	0.93333
Linear SVM	Linear	1000	0.958 ± 0.027	0.93333

USING NON LINEAR SVM

SVM	Kernel Type	C	Degree	Validation Accuracy	Test Accuracy
Non LINEAR SVM	Sigmoid	0.5	-	0.199 ± 0.062	0.23333
Non LINEAR SVM	Sigmoid	1	-	0.199 ± 0.062	0.23333
Non LINEAR SVM	Sigmoid	10	-	0.199 ± 0.062	0.23333
Non LINEAR SVM	Sigmoid	100	-	0.091 ± 0.014	0.13333
Non LINEAR SVM	Sigmoid	1000	-	0.241 ± 0.086	0.26667
Non LINEAR SVM	RBF	0.5	-	0.924 ± 0.049	0.8
Non LINEAR SVM	RBF	1	-	0.933 ± 0.047	0.96667
Non LINEAR SVM	RBF	10	-	0.975 ± 0.043	0.96667
Non LINEAR SVM	RBF	100	-	0.95 ± 0.037	0.93333
Non LINEAR SVM	RBF	1000	-	0.95 ± 0.028	0.93333
Non LINEAR SVM	Polynomial	0.5	2	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	2	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	10	2	0.941 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	2	0.958 ± 0.027	0.93333
Non LINEAR SVM	Polynomial	1000	2	0.941 ± 0.049	0.93333
Non LINEAR SVM	Polynomial	0.5	4	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	4	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	10	4	0.941 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	4	0.958 ± 0.027	0.93333
Non LINEAR SVM	Polynomial	1000	4	0.941 ± 0.049	0.93333
Non LINEAR SVM	Polynomial	0.5	6	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	1	6	0.958 ± 0.043	0.93333
Non LINEAR SVM	Polynomial	10	6	0.941 ± 0.036	0.93333
Non LINEAR SVM	Polynomial	100	6	0.958 ± 0.027	0.93333
Non LINEAR SVM	Polynomial	1000	6	0.941 ± 0.049	0.93333

Συμπεράσματα θα αναφερθούν στο τέλος.

Με χρήση 4 χαρακτηριστικών

Ο κώδικας για την συγκεκριμένη άσκηση βρίσκεται στο αρχείο **HW6_Exercise2_CD_4features.py**. Τα αποτελέσματα είναι:

Classification of class 2 with classes 1 and 3 using features 1,2,3,4

USING LINEAR SVM

SVM	Kernel Type	C	Validation Accuracy	Test Accuracy
Linear SVM	Linear	0.5	0.95 ± 0.037	1.0
Linear SVM	Linear	1	0.958 ± 0.014	1.0
Linear SVM	Linear	10	0.958 ± 0.027	1.0
Linear SVM	Linear	100	0.958 ± 0.027	0.96667
Linear SVM	Linear	1000	0.966 ± 0.023	0.96667

USING NON LINEAR SVM

SVM	Kernel Type	C	Degree	Validation Accuracy	Test Accuracy
Non LINEAR SVM	Sigmoid	0.5	-	0.066 ± 0.023	0.03333
Non LINEAR SVM	Sigmoid	1	-	0.066 ± 0.023	0.03333
Non LINEAR SVM	Sigmoid	10	-	0.066 ± 0.023	0.0
Non LINEAR SVM	Sigmoid	100	-	0.025 ± 0.027	0.06667
Non LINEAR SVM	Sigmoid	1000	-	0.033 ± 0.033	0.06667
Non LINEAR SVM	RBF	0.5	-	0.924 ± 0.043	1.0
Non LINEAR SVM	RBF	1	-	0.941 ± 0.043	1.0
Non LINEAR SVM	RBF	10	-	0.95 ± 0.016	1.0
Non LINEAR SVM	RBF	100	-	0.958 ± 0.027	1.0
Non LINEAR SVM	RBF	1000	-	0.958 ± 0.027	0.96667
Non LINEAR SVM	Polynomial	0.5	2	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	1	2	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	10	2	0.958 ± 0.027	0.96667
Non LINEAR SVM	Polynomial	100	2	0.966 ± 0.023	0.93333
Non LINEAR SVM	Polynomial	1000	2	0.966 ± 0.023	0.93333
Non LINEAR SVM	Polynomial	0.5	4	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	1	4	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	10	4	0.958 ± 0.027	0.96667
Non LINEAR SVM	Polynomial	100	4	0.966 ± 0.023	0.93333
Non LINEAR SVM	Polynomial	1000	4	0.966 ± 0.023	0.93333
Non LINEAR SVM	Polynomial	0.5	6	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	1	6	0.95 ± 0.016	1.0
Non LINEAR SVM	Polynomial	10	6	0.958 ± 0.027	0.96667
Non LINEAR SVM	Polynomial	100	6	0.966 ± 0.023	0.93333
Non LINEAR SVM	Polynomial	1000	6	0.966 ± 0.023	0.93333

Σαν γενικότερα συμπεράσματα μπορούμε να πούμε ότι έχουμε πολύ καλή ταξινόμηση. Το γραμμικό,rbf και πολυωνιμικό είχαν όλα πολύ καλά αποτελέσματα παρά το γεγονός ότι κλάσεις ω2 με ω3 δεν είναι γραμμικώς διαχωρίσιμες. Βλέπουμε ότι το svm και η μέθοδος one vs all δουλεύει πάρα πολύ καλά στην περίπτωση που έχουμε multiclass classification. Από την άλλη το sigmoid kernel είχε κακά αποτελέσματα κάτι που είναι αναμενόμενο καθώς αυτό το kernel είναι πιο χρήσιμο σε binary classification προβλήματα. Ειδικότερα στη περίπτωση χρήσης 3 χαρακτηριστικών παρατηρούμε και μία περίεργη συμπεριφορά με χρήση του Polynomial kernel δηλαδή το accuracy παραμένει σταθερό στο 93%. Στη περίπτωση της χρήσης και των 4 χαρακτηριστικών έχουμε τα βέλτιστα αποτελέσματα. Όπως φαίνεται σε πολλές περιπτώσεις ακόμη και με linear kernel έχουμε Test Accuracy 100% δηλαδή μηδενικό σφάλμα. Βέβαια τα συμπεράσματα δεν μπορούν να είναι απόλυτα καθώς ο αριθμός των δεδομένων είναι περιορισμένος.