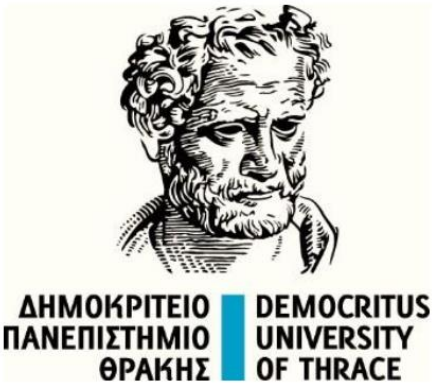


6Η ΕΡΓΑΣΙΑ

Στο μάθημα:
«Αναγνώριση Προτύπων»
Καθηγητής: Νικόλαος Μητιανούδης

Στυλιανός Μούσλεχ
ΑΜ:57382
13/12/2020 ,Ξάνθη

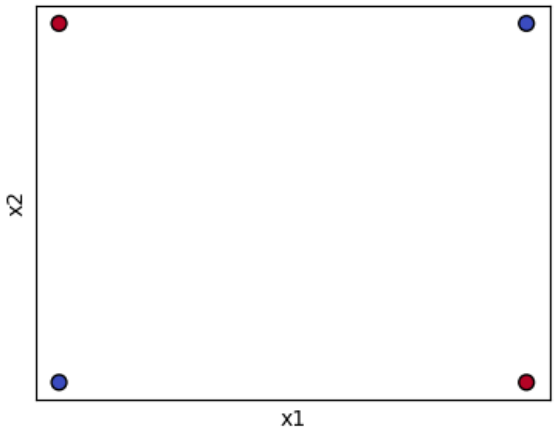
ΕΙΣΑΓΩΓΗ

Η υλοποίηση γίνεται στην κάνοντας χρήση της βιβλιοθήκης scikit-learn.

ΆΣΚΗΣΗ 6.1: Πρόβλημα XOR

Ο κώδικας του ερωτήματος γίνεται στο αρχείο “Ex6_1.py”.

Το πρόβλημα XOR για 2 εισόδους x1,x2 είναι μια περίπτωση μη γραμμικά διαχωρίσιμων δεδομένων, δηλαδή δεν μπορούμε να σχηματίσουμε γραμμή που να ξεχωρίζει τις 2 κλάσεις



Αρχικά θα δημιουργήσουμε τα 4 σημεία μας καθώς και την κατηγοριοποίηση του καθενός:

```
X = np.array([[0,0],[0,1],[1,0],[1,1]])
Y=np.logical_xor(X[:,0],X[:,1])
```

Στην συνέχεια θα φτιάξουμε το μοντέλο μας με βάση το Kernel της εκφώνησης:

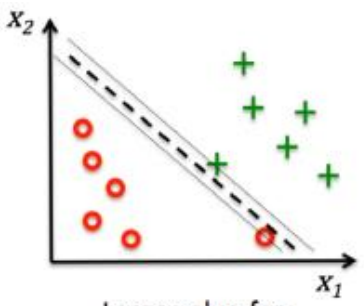
$$\Phi(\mathbf{x}) = (x_1^2 + \sqrt{2}x_1x_2 + x_2^2),$$

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x}).\Phi(\mathbf{y}) = (x_1y_1 + x_2y_2)^2 = (\mathbf{x}.\mathbf{y})^2$$

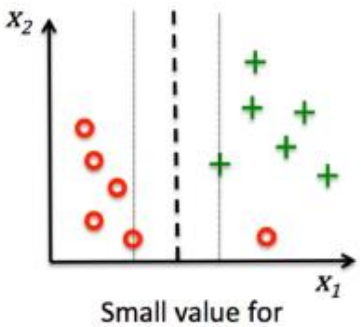
Εδώ το Kernel είναι πολυωνυμικό 2ου βαθμού και με βάση το documentation της βιβλιοθήκης:

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

Θα βάλουμε degree=2, coef0=0 (το coef0 έχει default τιμή 0)
Αναφορικά με την Slack variable C ουσιαστικά ελέγχει το πλάτος του margin,
Μεγάλες τιμές C => μεγάλη ποινή για misclassification
Μικρές τιμές C => μικρή ποινή για misclassification



Large value for parameter *C*

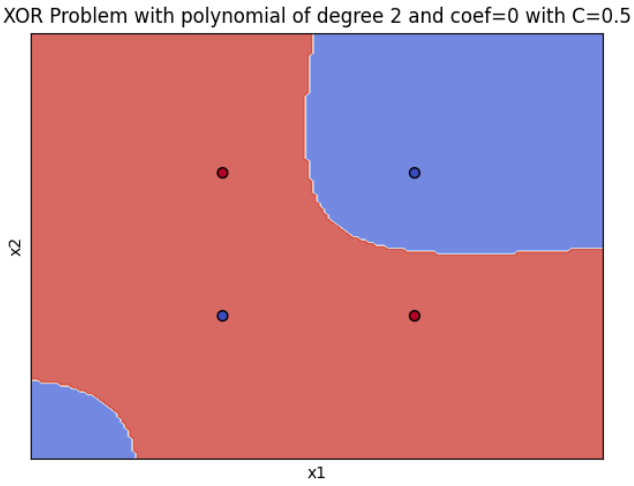


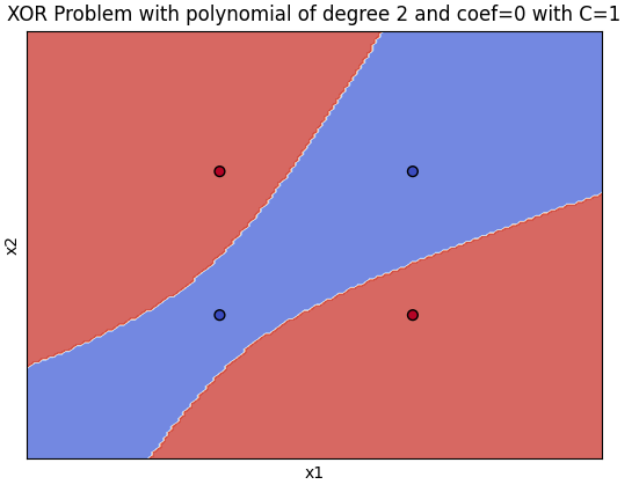
Small value for parameter *C*

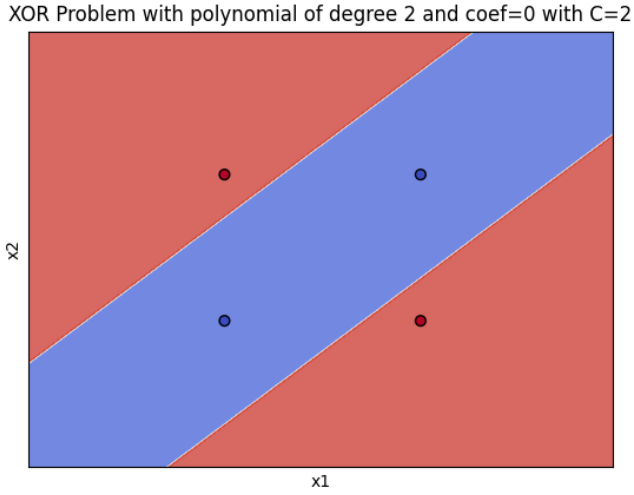
Οπότε εκπαιδεύουμε το SVM:

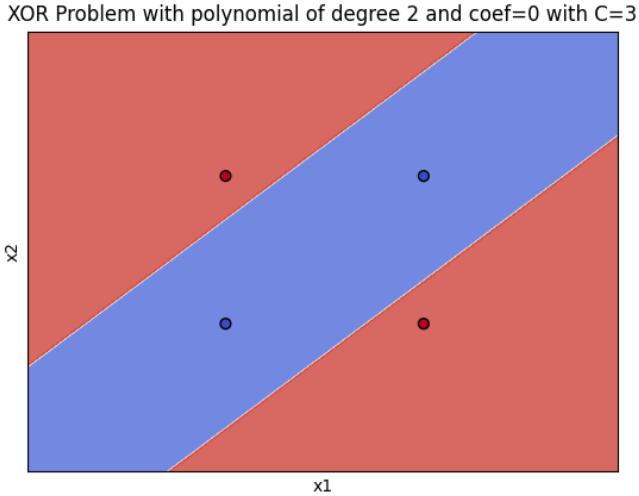
```
model = svm.SVC(kernel='poly',degree=2,coef0=0,C=2)
clf = model.fit(X, Y)
```

και το εκπαίδευσα για διαφορετικές τιμές C για να δούμε πως διαφοροποιείται το αποτέλεσμα:









Παρατηρούμε ότι μπορούμε να διαχωρίσουμε σωστά τις 2 κατηγορίες με πολύ καλό τρόπο. Κάτι που δεν γινόταν με έναν μοναδικό γραμμικό ταξινομητή. Βέβαια αν διαλέξουμε πολύ

μικρό C (εδώ στο 0.5) δεν διαχωρίζουμε σωστά καθώς όπως είπαμε με μικρό C δεν «τιμωρούμε» αρκετά την λάθος ταξινόμηση.

Αναλυτική λύση

Έχουμε το kernel που μας δόθηκε, πρέπει αρχικά να υπολογίσουμε τα α.

Για καλύτερες πράξεις μετατρέπουμε το πόβλημα γύρω από την αρχή των αξόνων οπότε:

Διάνυσμα εισόδου	Έξοδος
(-1,-1)	-1
(-1,+1)	+1
(+1,-1)	+1
(+1,+1)	-1

Και από $\bar{L}(a) = \sum_{i=1}^{4} a_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4} a_i a_j y_i y_j K(x_i, x_j)$

Βγάζουμε πρώτα τις τιμές του Kernel για κάθε περίπτωση ζευγαριών inputs:

$K(x_i, x_j) = (x_i^1 x_j^1 + x_i^2 x_j^2)$	j=1: (-1,-1)	j=2: (-1,+1)	j=3: (+1,-1)	j=4: (+1,+1)
i=1: (-1,-1)	4	0	0	0
i=2: (-1,+1)	0	4	0	0
i=3: (+1,-1)	0	0	4	0
i=4: (+1,+1)	0	0	0	4

Οπότε θα έχουμε από τον παραπάνω τύπο αντικαθιστώντας:

$$\bar{L}(a) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\left(4\alpha_1^2 + 4\alpha_2^2 + 4\alpha_3^2 + 4\alpha_4^2\right)$$

Και από τις μερικές παραγώγους υπολογίζω:

$$\frac{\partial \bar{L}}{\partial \alpha_1} = 0 \implies 1 - 4\alpha_1 = 0 \implies \alpha_1 = \frac{1}{4}$$
$$\frac{\partial \bar{L}}{\partial \alpha_2} = 0 \implies 1 - 4\alpha_2 = 0 \implies \alpha_2 = \frac{1}{4}$$
$$\frac{\partial \bar{L}}{\partial \alpha_3} = 0 \implies 1 - 4\alpha_3 = 0 \implies \alpha_3 = \frac{1}{4}$$
$$\frac{\partial \bar{L}}{\partial \alpha_4} = 0 \implies 1 - 4\alpha_4 = 0 \implies \alpha_4 = \frac{1}{4}$$

Άρα έχουμε: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{4}$

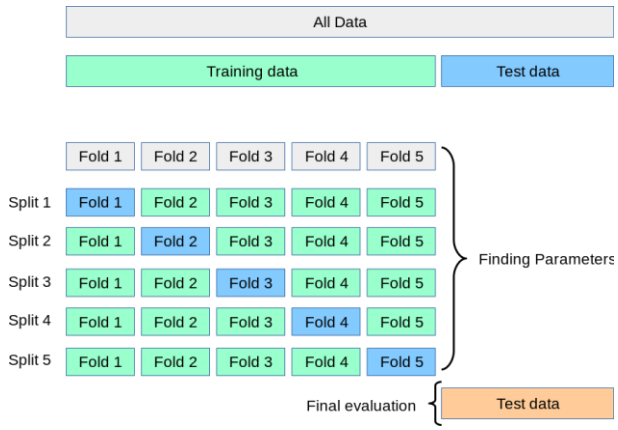
Αντικαθιστώντας: $\overline{L_o}(a) = 1 - \frac{1}{2}1 = \frac{1}{2}$

και $\frac{1}{2}||w_o||^2 = \frac{1}{2} \Longrightarrow ||w_o|| = 1$

ΆΣΚΗΣΗ 6.2: IRIS Dataset

Στην εκφώνηση χωρίζουμε το dataset σε train/validation/testing. 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 και το μοντέλο εκπαιδεύεται στα k-1 groups και τεστάρεται σε 1 διαφορετικό validation set. Έτσι μπορούμε να δούμε ποιο μοντέλο γενικοποιεί καλύτερα με βάση το average accuracy και το standard deviation αυτών. Η παρακάτω εικόνα οπτικοποεί καλά την διαδικασία για K=5



Σημαντική υποσημείωση αποτελεί ότι όταν κάνουμε predict τα test data έχουμε κάνει train σε όλο το train dataset.

Στην εκφώνηση έχουμε 10 ανά κάθε κατηγορία άρα 30 συνολικό validation test άρα 120/30=4 άρα K=4

Πρόβλημα 2 κατηγοριών

Χρησιμοποιώντας χαρακτηριστικά 1,2,4

Ο κώδικας στο αρχείο «Ex6_2class3d»

Αρχικά ενώνουμε τις κλάσεις ω1 και ω3 και παίρνουμε τα χαρακτηριστικά 1,2,4 (0,1,3 στον πίνακα)

```
iris=load_iris()
X=iris.data
Y=iris.target

#get all w1 and w3 into one class so 0 and 2 should be 0
Ynew=np.where(Y==2,0,Y)
print("USING ONLY 1,3,4 dims")
#use only 1st,2nd,4th characteristics
Xnew=X[:,[0,1,3]]
```

Στη συνέχεια δημιουργούμε το training και test dataset προσέχοντας να έχουμε τον ίδιο αριθμό δεδομένων από κάθε κλάση με την χρήση της παραμέτρου stratify

```
X_train, X_test, y_train, y_test = train_test_split(Xnew,Ynew, test_size=0.2, random_state=0,stratify=Ynew)
```

Στη συνέχεια τρέξαμε τον κώδικα για διαφορετικά kernel και παραμέτρους. Πιο συγκεκριμένα έγινε χρήση των Kernels= Linear,sigmoid,rbf,polynomial για διαφορετικές τιμές c (και στην περίπτωση του polynomial και διαφορετικά degrees)

Η παράμετρος gamma κρατήθηκε σταθερή στην αυτόματη τιμή της που φάνηκε να λειτουργεί καλύτερα και είναι «1 / (n_features * X.var())».

Έχουμε τα παρακάτω αποτελέσματα:

LINEAR SVM

```
                        USING LINEAR SVM
=============================================================================================================
USING C= 0.5
Validation Accuracy  : 0.68 (+/- 0.03)
Test accuracy: 0.7333333333333333
=============================================================================================================
USING C= 1
Validation Accuracy  : 0.70 (+/- 0.08)
Test accuracy: 0.7333333333333333
=============================================================================================================
USING C= 10
Validation Accuracy  : 0.72 (+/- 0.07)
Test accuracy: 0.7666666666666667
=============================================================================================================
USING C= 100
Validation Accuracy  : 0.73 (+/- 0.12)
Test accuracy: 0.7666666666666667
=============================================================================================================
USING C= 1000
Validation Accuracy  : 0.72 (+/- 0.09)
Test accuracy: 0.7666666666666667
=============================================================================================================
```

Εδώ παρατηρούμε ότι τα αποτελέσματα του γραμμικού svm για 2 κλάσεις που δεν είναι γραμμικά διαχωρίσιμες είναι αρκετά κοντά στον γραμμικό ταξινομητή της 5^{ης} άσκησης ,κάτι αναμενόμενο καθώς ουσιαστικά πάλι έχουμε μια ευθεία διαχωριστική γραμμή για μη γραμμικώς διαχωρίσιμα δεδομένα 2 κλάσεων.

NON LINEAR SVM

```
================================================================================================
                        USING NON LINEAR SVM WITH RBF KERNEL
================================================================================================
USING C= 0.5
Validation Accuracy  : 0.67 (+/- 0.00)
Test accuracy: 0.6666666666666666
================================================================================================
USING C= 1
Validation Accuracy  : 0.77 (+/- 0.05)
Test accuracy: 0.9
================================================================================================
USING C= 10
Validation Accuracy  : 0.94 (+/- 0.07)
Test accuracy: 0.9666666666666667
================================================================================================
USING C= 100
Validation Accuracy  : 0.97 (+/- 0.07)
Test accuracy: 0.9666666666666667
================================================================================================
USING C= 1000
Validation Accuracy  : 0.93 (+/- 0.14)
Test accuracy: 0.9666666666666667
================================================================================================
```

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. Εδώ τα αποτελέσματα είναι πολύ καλύτερα σε σχέση με τον γραμμικό ταξινομητή της προηγούμενης άσκησης

Αντιθέτως Χρησιμοποιώντας sigmoid kernel έχουμε χειρότερα αποτελέσματα:

```
================================================================================================
                        USING NON LINEAR SVM WITH SIGMOID KERNEL
================================================================================================
USING C= 0.5
Validation Accuracy  : 0.67 (+/- 0.00)
Test accuracy: 0.6666666666666666
================================================================================================
USING C= 1
Validation Accuracy  : 0.67 (+/- 0.00)
Test accuracy: 0.6666666666666666
================================================================================================
USING C= 10
Validation Accuracy  : 0.67 (+/- 0.05)
Test accuracy: 0.7
================================================================================================
USING C= 100
Validation Accuracy  : 0.55 (+/- 0.15)
Test accuracy: 0.6666666666666666
================================================================================================
USING C= 1000
Validation Accuracy  : 0.52 (+/- 0.15)
Test accuracy: 0.7
================================================================================================
```

Δοκιμάζοντας και πολυωνυμικό kernel με διάφορους βαθμούς και c :

```
============================================================
                        USING NON LINEAR SVM WITH POLY KERNEL
============================================================
USING DEGREE= 2 C= 0.5
Validation Accuracy  : 0.79 (+/- 0.11)
Test accuracy: 0.8666666666666667
============================================================
USING DEGREE= 2 C= 1
Validation Accuracy  : 0.85 (+/- 0.06)
Test accuracy: 0.9
============================================================
USING DEGREE= 2 C= 10
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 100
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9666666666666667
============================================================
USING DEGREE= 2 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9666666666666667
============================================================
```

```
============================================
USING DEGREE= 4 C= 0.5
Validation Accuracy  : 0.79 (+/- 0.11)
Test accuracy: 0.8666666666666667
============================================
USING DEGREE= 4 C= 1
Validation Accuracy  : 0.85 (+/- 0.06)
Test accuracy: 0.9
============================================
USING DEGREE= 4 C= 10
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9333333333333333
============================================
USING DEGREE= 4 C= 100
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9666666666666667
============================================
USING DEGREE= 4 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9666666666666667
============================================
```

```
============================================
USING DEGREE= 6 C= 0.5
Validation Accuracy  : 0.79 (+/- 0.11)
Test accuracy: 0.8666666666666667
============================================
USING DEGREE= 6 C= 1
Validation Accuracy  : 0.85 (+/- 0.06)
Test accuracy: 0.9
============================================
USING DEGREE= 6 C= 10
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9333333333333333
============================================
USING DEGREE= 6 C= 100
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9666666666666667
============================================
USING DEGREE= 6 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9666666666666667
============================================
```

Παρατηρούμε ότι και το πολυωνυμικό kernel έχει πολύ καλά αποτελέσματα. Επίσης παρατηρούμε ότι για το συγκεκριμένο πρόβλημα, η αύξηση του βαθμού του πολυωνύμου δεν αλλάζει πολύ τα ήδη καλά αποτελέσματα

Χρησιμοποιώντας χαρακτηριστικά 1,2,3,4

Ο κώδικας βρίσκεται στο αρχείο «Ex6_2class4d»

Κάνοντας την ταξινόμηση για 4 χαρακτηριστικά παρατηρούμε τα ίδια περίπου αποτελέσματα και με τα 3 χαρακτηριστικά και καταλήγουμε στα ίδια συμπεράσματα με τα παραπάνω

Γραμμικό και rbf:

```
USING ALL CHARACTERISTICS
                        USING LINEAR SVM
================================================
USING C= 0.5
Validation Accuracy  : 0.68 (+/- 0.03)
Test accuracy: 0.7333333333333333
================================================
USING C= 1
Validation Accuracy  : 0.72 (+/- 0.10)
Test accuracy: 0.7333333333333333
================================================
USING C= 10
Validation Accuracy  : 0.76 (+/- 0.10)
Test accuracy: 0.7666666666666667
================================================
USING C= 100
Validation Accuracy  : 0.77 (+/- 0.10)
Test accuracy: 0.7666666666666667
================================================
USING C= 1000
Validation Accuracy  : 0.78 (+/- 0.11)
Test accuracy: 0.7666666666666667
================================================
```

```
============================================================
                        USING NON LINEAR SVM WITH RBF KERNEL
============================================================
USING C= 0.5
Validation Accuracy  : 0.92 (+/- 0.10)
Test accuracy: 0.9
============================================================
USING C= 1
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9333333333333333
============================================================
USING C= 10
Validation Accuracy  : 0.98 (+/- 0.03)
Test accuracy: 0.9333333333333333
============================================================
USING C= 100
Validation Accuracy  : 0.95 (+/- 0.11)
Test accuracy: 0.9666666666666667
============================================================
USING C= 1000
Validation Accuracy  : 0.95 (+/- 0.11)
Test accuracy: 0.9666666666666667
============================================================
```

Sigmoid

```
============================================================================
                        USING NON LINEAR SVM WITH SIGMOID KERNEL
============================================================================
USING C= 0.5
Validation Accuracy  : 0.67 (+/- 0.00)
Test accuracy: 0.6666666666666666
============================================================================
USING C= 1
Validation Accuracy  : 0.67 (+/- 0.00)
Test accuracy: 0.6666666666666666
============================================================================
USING C= 10
Validation Accuracy  : 0.53 (+/- 0.17)
Test accuracy: 0.5666666666666667
============================================================================
USING C= 100
Validation Accuracy  : 0.45 (+/- 0.14)
Test accuracy: 0.5
============================================================================
USING C= 1000
Validation Accuracy  : 0.45 (+/- 0.14)
Test accuracy: 0.5
============================================================================
```

Polynomial

```
============================================================
                    USING NON LINEAR SVM WITH POLY KERNEL
============================================================
USING DEGREE= 2 C= 0.5
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 1
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9666666666666667
============================================================
USING DEGREE= 2 C= 10
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9666666666666667
============================================================
USING DEGREE= 2 C= 100
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 1000
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9666666666666667
============================================================
```

```
========================================
USING DEGREE= 4 C= 0.5
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 1
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9666666666666667
========================================
USING DEGREE= 4 C= 10
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9666666666666667
========================================
USING DEGREE= 4 C= 100
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 1000
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9666666666666667
========================================
```

```
========================================
USING DEGREE= 6 C= 0.5
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 1
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9666666666666667
========================================
USING DEGREE= 6 C= 10
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9666666666666667
========================================
USING DEGREE= 6 C= 100
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 1000
Validation Accuracy  : 0.96 (+/- 0.07)
Test accuracy: 0.9666666666666667
========================================
```

Πρόβλημα 3 κατηγοριών

Χρησιμοποιώντας όλα τα χαρακτηριστικά

Ο κώδικας βρίσκεται στο αρχείο «Ex6_3class4d»

Εδώ πλέον έχουμε 3 κλάσεις και θα χρησιμοποιήσουμε την τακτική one vs all (ή one vs rest) ουσιαστικά κάνουμε train έναν classifier για κάθε κλάση, που διαχωρίζει αυτή τη κλάση με μια άλλη κλάση που είναι όλες οι υπόλοιπες μαζί

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.

Δοκιμάζοντας για όμοιες παραμέτρους με πριν :

Linear SVM

```
                        USING LINEAR SVM
=============================================================================================
USING C= 0.5
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 1.0
=============================================================================================
USING C= 1
Validation Accuracy  : 0.96 (+/- 0.03)
Test accuracy: 1.0
=============================================================================================
USING C= 10
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 1.0
=============================================================================================
USING C= 100
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9666666666666667
=============================================================================================
USING C= 1000
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9666666666666667
=============================================================================================
```

Sigmoid Kernel

```
=============================================================================
                        USING NON LINEAR SVM WITH SIGMOID KERNEL
=============================================================================
USING C= 0.5
Validation Accuracy  : 0.07 (+/- 0.05)
Test accuracy: 0.03333333333333333
=============================================================================
USING C= 1
Validation Accuracy  : 0.07 (+/- 0.05)
Test accuracy: 0.03333333333333333
=============================================================================
USING C= 10
Validation Accuracy  : 0.07 (+/- 0.05)
Test accuracy: 0.0
=============================================================================
USING C= 100
Validation Accuracy  : 0.03 (+/- 0.06)
Test accuracy: 0.06666666666666667
=============================================================================
USING C= 1000
Validation Accuracy  : 0.03 (+/- 0.07)
Test accuracy: 0.06666666666666667
=============================================================================
```

Rbf kernel

```
================================================================
                    USING NON LINEAR SVM WITH RBF KERNEL
================================================================
USING C= 0.5
Validation Accuracy  : 0.92 (+/- 0.09)
Test accuracy: 1.0
================================================================
USING C= 1
Validation Accuracy  : 0.94 (+/- 0.09)
Test accuracy: 1.0
================================================================
USING C= 10
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================================
USING C= 100
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 1.0
================================================================
USING C= 1000
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9666666666666667
```

Polynomial Kernel

```
================================================================
                    USING NON LINEAR SVM WITH POLY KERNEL
================================================================
USING DEGREE= 2 C= 0.5
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================================
USING DEGREE= 2 C= 1
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================================
USING DEGREE= 2 C= 10
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9666666666666667
================================================================
USING DEGREE= 2 C= 100
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================================
USING DEGREE= 2 C= 1000
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================================
```

```
================================================
USING DEGREE= 4 C= 0.5
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================
USING DEGREE= 4 C= 1
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================
USING DEGREE= 4 C= 10
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9666666666666667
================================================
USING DEGREE= 4 C= 100
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================
USING DEGREE= 4 C= 1000
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================
```

```
================================================
USING DEGREE= 6 C= 0.5
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================
USING DEGREE= 6 C= 1
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================
USING DEGREE= 6 C= 10
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9666666666666667
================================================
USING DEGREE= 6 C= 100
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================
USING DEGREE= 6 C= 1000
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 0.9333333333333333
================================================
```

Σαν γενικότερα συμπεράσματα βλέπουμε ότι έχουμε σε πολλές περιπτώσεις μέχρι και 100% accuracy. Το γραμμικό,rbf και πολυωνιμικό είχαν όλα πολύ καλά αποτελέσματα ακόμα και αν σκεφτούμε ότι η 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

Χρησιμοποιώντας τα χαρακτηριστικά 1,2,4

Ο κώδικας βρίσκεται στο αρχείο «Ex6_3class3d»

Και εδώ έχουμε όμοια αποτελέσματα με το να χρησιμοποιήσουμε 4 χαρακτηριστικά με μόνη διαφορά λίγο πιο βελτιωμένη sigmoid αλλά και λίγο χειρότερες τις υπόλοιπες.

Linear SVM

```
                          USING LINEAR SVM
================================================
USING C= 0.5
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9666666666666667
================================================
USING C= 1
Validation Accuracy  : 0.97 (+/- 0.08)
Test accuracy: 0.9333333333333333
================================================
USING C= 10
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9333333333333333
================================================
USING C= 100
Validation Accuracy  : 0.95 (+/- 0.06)
Test accuracy: 0.9333333333333333
================================================
USING C= 1000
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9333333333333333
================================================
```

Sigmoid Kernel

```
================================================================
                        USING NON LINEAR SVM WITH SIGMOID KERNEL
================================================================
USING C= 0.5
Validation Accuracy  : 0.20 (+/- 0.12)
Test accuracy: 0.23333333333333334
================================================================
USING C= 1
Validation Accuracy  : 0.20 (+/- 0.12)
Test accuracy: 0.23333333333333334
================================================================
USING C= 10
Validation Accuracy  : 0.20 (+/- 0.12)
Test accuracy: 0.23333333333333334
================================================================
USING C= 100
Validation Accuracy  : 0.09 (+/- 0.03)
Test accuracy: 0.13333333333333333
================================================================
USING C= 1000
Validation Accuracy  : 0.24 (+/- 0.17)
Test accuracy: 0.26666666666666666
```

Rbf kernel

```
================================================================
                        USING NON LINEAR SVM WITH RBF KERNEL
================================================================
USING C= 0.5
Validation Accuracy  : 0.92 (+/- 0.10)
Test accuracy: 0.8
================================================================
USING C= 1
Validation Accuracy  : 0.93 (+/- 0.09)
Test accuracy: 0.9666666666666667
================================================================
USING C= 10
Validation Accuracy  : 0.97 (+/- 0.09)
Test accuracy: 0.9666666666666667
================================================================
USING C= 100
Validation Accuracy  : 0.95 (+/- 0.07)
Test accuracy: 0.9333333333333333
================================================================
USING C= 1000
Validation Accuracy  : 0.95 (+/- 0.06)
Test accuracy: 0.9333333333333333
================================================================
```

Polynomial Kernel

```
============================================================
                    USING NON LINEAR SVM WITH POLY KERNEL
============================================================
USING DEGREE= 2 C= 0.5
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 1
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 10
Validation Accuracy  : 0.94 (+/- 0.07)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 100
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9333333333333333
============================================================
USING DEGREE= 2 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9333333333333333
============================================================
```

```
========================================
USING DEGREE= 4 C= 0.5
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 1
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 10
Validation Accuracy  : 0.94 (+/- 0.07)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 100
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 4 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9333333333333333
========================================
```

```
========================================
USING DEGREE= 6 C= 0.5
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 1
Validation Accuracy  : 0.96 (+/- 0.09)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 10
Validation Accuracy  : 0.94 (+/- 0.07)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 100
Validation Accuracy  : 0.96 (+/- 0.06)
Test accuracy: 0.9333333333333333
========================================
USING DEGREE= 6 C= 1000
Validation Accuracy  : 0.94 (+/- 0.10)
Test accuracy: 0.9333333333333333
========================================
```