Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

9ο Εξάμηνο
Φθινόπωρο 2019

Classification στην R: kNN και SVM

1 Εισαγωγή

1.1 Εισαγωγή στους αλγορίθμους kNN και SVM

Ο k-Nearest Neighbors (kNN) είναι ένας αλγόριθμος μηχανικής μάθησης για τον οποίο η πρόβλεψη για ένα νέο δείγμα βασίζεται στα k κοντινότερα training δείγματα στο συγκεκριμένο δείγμα. Έτσι, η τιμή κλάσης για ένα νέο δείγμα αποφασίζεται κατά πλειοψηφία (majority vote) με βάση τις τιμές κλάσης των k κοντινότερων δειγμάτων του.

Τα SVM είναι ένας αλγόριθμος που αφορά την εύρεση ενός γραμμικού υπερεπίπεδου το οποίο διαχωρίζει τα δεδομένα. Το πρόβλημα έγκειται στη μεγιστοποίηση του περιθωρίου (margin) μεταξύ του υπερεπιπέδου και των δεδομένων:

$$\text{Μεγιστοποίηση } Margin = 2/\|w^2\| \text{ με δεδομένο ότι } f(x) = \begin{cases} 1 & if\ w \cdot x + b \geq 1 \\ -1 & if\ w \cdot x + b \leq 1 \end{cases}$$

Σε περίπτωση που το πρόβλημα δεν επιλύεται γραμμικά, είναι δυνατή η αντιστοίχιση του σετ εκπαίδευσης σε χώρο ανώτερης διάστασης με τη χρήση kernels, ώστε πλέον το σετ εκπαίδευσης να είναι γραμμικά διαχωρίσιμο.

1.2 kNN στην R

Για τον αλγόριθμο kNN στην R, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη class:

```
> library(class)
```

Για να ταξινομήσουμε μια νέα τιμή τρέχουμε την εντολή knn:

```
> knn(X_train, X_test, Y_train, k = 1, prob = TRUE)
```

όπου δηλώνουμε τα δεδομένα εκπαίδευσης, το νέο δείγμα και την τιμή του k, ενώ προσθέτοντας την παράμετρο prob = TRUE δίνονται επιπλέον οι πιθανότητες για την κλάση.

1.3 SVM στην R

Για την κατασκευή ενός SVM στην R, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη e1071:

```
> library(e1071)
```

Για να κατασκευάσουμε ένα μοντέλο τρέχουμε την εντολή svm:

```
> model = svm(y ~ ., data, kernel = "radial", type = "C-classification")
```

όπου με το kernel επιλέγουμε τον πυρήνα του SVM (linear, polynomial, radial) ενώ μπορούμε να ορίσουμε επιπλέον παραμέτρους, όπως τα degree, gamma, coef0. Μπορούμε να εκτελέσουμε την εντολή `?svm` για να δούμε όλες τις παραμέτρους.

Έχοντας ένα μοντέλο, για να προβλέψουμε νέες τιμές τρέχουμε την εντολή

```
> predict(model, test)
```

Αν θέλουμε επιπλέον να επιστρέψουμε τις πιθανότητες για κάθε κλάση θα πρέπει να θέσουμε την παράμετρο probability = TRUE και κατά το training (στην εντολή svm) και κατά το testing (στην εντολή predict).

2 Κατασκευή Μοντέλου kNN και Κατάταξη Τιμών

Για την κατασκευή ενός μοντέλου kNN θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα δυαδικής ταξινόμησης.

X1	**X2**	**Y**
0.7	0.7	A
0.7	0.8	A
0.6	0.6	A
0.5	0.5	A
0.5	0.6	A
0.5	0.7	A
0.5	0.8	A
0.7	0.5	B
0.8	0.7	B
0.8	0.5	B
0.8	0.6	B
1.0	0.3	B
1.0	0.5	B
1.0	0.6	B

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα με διαφορετικά χρώματα/σύμβολα για κάθε κλάση.

β) Χρησιμοποιώντας τον kNN με k = 1, σε ποια κλάση θα κατατάσσατε μία νέα παρατήρηση με τιμές (X1, X2) = (0.7, 0.4);

γ) Επαναλάβετε το ερώτημα (β) χρησιμοποιώντας k = 5.

δ) Σε ποια κλάση θα κατέτασσε ο kNN με k = 5 μία παρατήρηση με τιμές (X1, X2) = (0.7, 0.6)

2.1 Εισαγωγή Δεδομένων και Βιβλιοθηκών

Αρχικά διαβάζουμε τα δεδομένα και φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
> knndata = read.csv("knndata.txt")
> library(class)
```

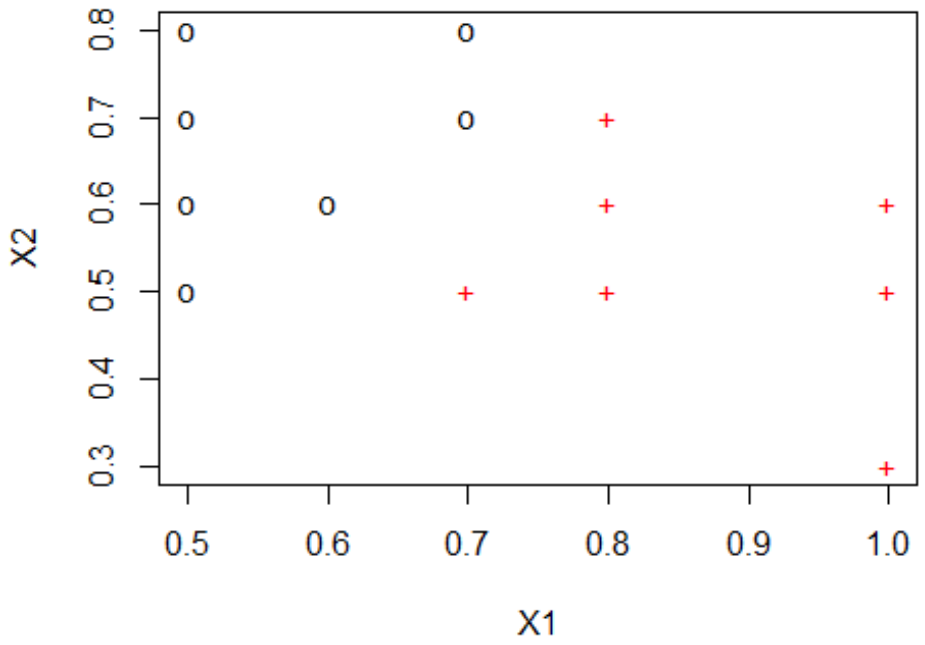
2.2 Εφαρμογή Αλγορίθμου kNN

Χωρίζουμε αρχικά το dataset όπως παρακάτω:

```
> X_train = knndata[,c("X1","X2")]
> Y_train = knndata$Y
```

Για το ερώτημα (α) εκτελούμε :

```
> plot(X_train, col = Y_train, pch = c("o","+")[Y_train])
```



Για k = 1 (ερώτημα (β)), αρκεί να βρούμε την κοντινότερη παρατήρηση (ευκλείδεια απόσταση) στην (0.7, 0.4), η οποία είναι η (0.7, 0.5) που ανήκει στην κλάση B. Οπότε εκτελώντας την εντολή:

```
> knn(X_train, c(0.7, 0.4), Y_train, k = 1, prob = TRUE)
```

προκύπτει ότι η νέα παρατήρηση ανήκει στην κλάση B με πιθανότητα 1.

Για k = 5 (ερώτημα (γ)), θα βρούμε τις 5 πιο κοντινές παρατηρήσεις, που είναι οι (0.7, 0.5), (0.8, 0.5), (0.6, 0.6), (0.8, 0.6), (0.7, 0.7) που ανήκουν στις κλάσεις B, B, A, B, A αντίστοιχα. Άρα αφού τα 3/5 των παρατηρήσεων αυτών ανήκουν στην κλάση B εκτελώντας την εντολή:

```
> knn(X_train, c(0.7, 0.4), Y_train, k = 5, prob = TRUE)
```

προκύπτει ότι η νέα παρατήρηση ανήκει στην κλάση B με πιθανότητα 0.6.

Για το ερώτημα (δ), θα θέλαμε να βρούμε τις 5 πιο κοντινές παρατηρήσεις στο σημείο (0.7, 0.6). Παρατηρούμε όμως ότι οι παρατηρήσεις (0.8, 0.7) και (0.8, 0.5) (που ανήκουν στην κλάση B) ισαπέχουν από το συγκεκριμένο σημείο. Έτσι (και επειδή η παράμετρος use.all του kNN είναι TRUE), ο αλγόριθμος θα χρησιμοποιήσει όλες αυτές τις παρατηρήσεις. Άρα αφού τα 4/6 των παρατηρήσεων ανήκουν στην κλάση B εκτελώντας την εντολή:

```
> knn(X_train, c(0.7, 0.6), Y_train, k = 5, prob = TRUE)
```

προκύπτει ότι η νέα παρατήρηση ανήκει στην κλάση B με πιθανότητα 0.66.

3 Κατασκευή Μοντέλου SVM και Εφαρμογή με Cross-validation

Για την κατασκευή ενός μοντέλου SVM θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του αρχείου που δίνεται (alldata.txt) για ένα πρόβλημα δυαδικής ταξινόμησης.

Ένα summary των δεδομένων με την R είναι το παρακάτω:

```
       X1                    X2                    y
 Min.   :-3.25322007   Min.   :-4.664161   Min.   :1.0
 1st Qu.:-0.70900886   1st Qu.: 0.625361   1st Qu.:1.0
 Median :-0.01162501   Median : 1.641370   Median :1.5
 Mean   : 0.03493570   Mean   : 1.939284   Mean   :1.5
 3rd Qu.: 0.73337179   3rd Qu.: 3.115350   3rd Qu.:2.0
 Max.   : 3.63957363   Max.   : 9.784076   Max.   :2.0
```

Αρχικά, εισάγουμε τα δεδομένα και τα διαχωρίζουμε σε training set και test set:

```
alldata = read.csv("alldata.txt")
trainingdata = alldata[1:600, ]
testdata = alldata[601:800, ]
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα training data με διαφορετικά χρώματα/σύμβολα για κάθε κλάση.

β) Εφαρμόστε τον SVM με RBF kernel και την παράμετρο gamma ίση με 1 και σχεδιάστε το υπερεπίπεδο στο σχήμα του ερωτήματος (α).

γ) Επαναλάβετε το ερώτημα (β) για τιμές του gamma ίσες με 0.01 και με 100. Τι παρατηρείτε;

Χρησιμοποιώντας για την παράμετρο gamma τις τιμές

```
gammavalues = c(0.01, 0.1, 1, 10, 100, 1000, 10000, 100000, 1000000)
```

θα απαντήσουμε επιπλέον στα παρακάτω ερωτήματα:

δ) Υπολογίστε το σφάλμα εκπαίδευσης (training error) και το σφάλμα ελέγχου (testing error) στο training set και στο test set αντίστοιχα.

ε) Σχεδιάστε τα δύο σφάλματα στο ίδιο διάγραμμα. Τι παρατηρείτε;

στ) Εφαρμόστε 10-fold cross validation για να βρείτε την καλύτερη τιμή για το gamma.

3.1 Εισαγωγή Βιβλιοθηκών

Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
library(MLmetrics)
library(e1071)
```

3.2 Εφαρμογή Αλγορίθμου SVM και Σχεδίαση Υπερεπίπεδου

Για το ερώτημα (α) εκτελούμε:

```
> plot(trainingdata[, c(1:2)], col = trainingdata$y,
                               pch = c("o","+")[trainingdata$y])
```

Για τα ερωτήματα (β), (γ), κατασκευάζουμε αρχικά ένα grid πάνω στο οποίο θα σχεδιάσουμε τα υπερεπίπεδα:

```
> X1 = seq(min(trainingdata[, 1]), max(trainingdata[, 1]), by = 0.1)
> X2 = seq(min(trainingdata[, 2]), max(trainingdata[, 2]), by = 0.1)
> mygrid = expand.grid(X1, X2)
> colnames(mygrid) = colnames(trainingdata)[1:2]
```

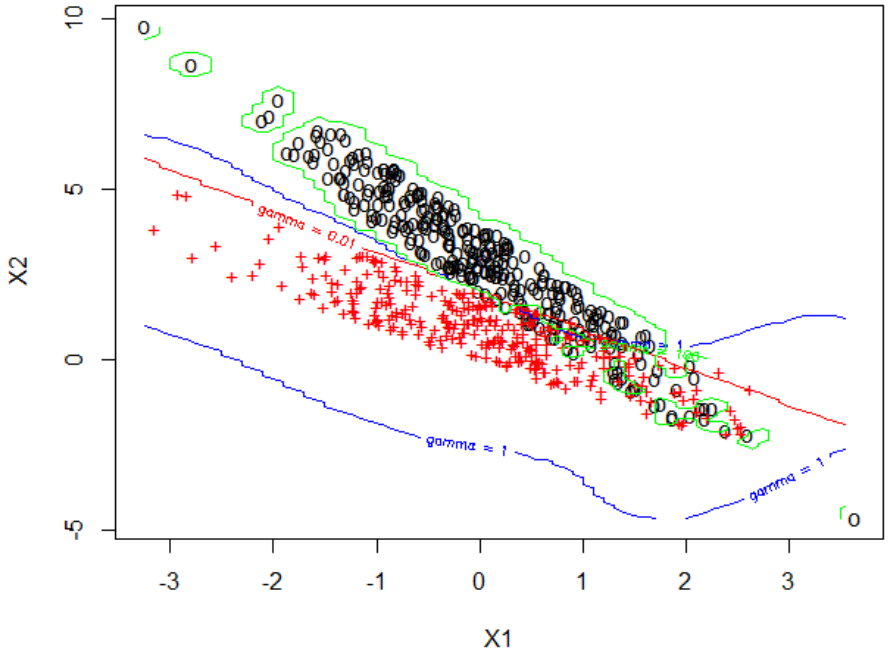
Για το (β) κάνουμε train το svm για gamma = 1 και το εφαρμόζουμε στα σημεία του grid:

```
> svm_model = svm(y ~ ., kernel="radial", type="C-classification", data = trainingdata, gamma = 1)

> pred = predict(svm_model, mygrid)

> Y = matrix(pred, length(X1), length(X2))

> contour(X1, X2, Y, add = TRUE, levels = 1.5, labels = "gamma = 1", col = "blue")
```

Όμοια για gamma ίσο με 0.01 και για gamma ίσο με 100 (ερώτημα (γ)), προκύπτει:

```
> # Plot a contour for the SVM with gamma = 0.01

> svm_model = svm(y ~ ., kernel="radial", type="C-classification", data = trainingdata, gamma = 0.01)

> pred = predict(svm_model, mygrid)

> Y = matrix(pred, length(X1), length(X2))

> contour(X1, X2, Y, add = TRUE, levels = 1.5, labels = "gamma = 0.01", col = "red")

# Plot a contour for the SVM with gamma = 100

> svm_model = svm(y ~ ., kernel="radial", type="C-classification", data = trainingdata, gamma = 100)

> pred = predict(svm_model, mygrid)

> Y = matrix(pred, length(X1), length(X2))

> contour(X1, X2, Y, add = TRUE, levels = 1.5, labels = "gamma = 100", col = "green")
```

Έτσι, τελικά προκύπτει το παρακάτω διάγραμμα:



3.3 Σχεδίαση Καμπυλών Training και Testing Error

Για το ερώτημα (δ) εκτελούμε μια for για τις τιμές του gamma και υπολογίζουμε το σφάλμα εκπαίδευσης για κάθε τιμή:

```
> training_error = c()
> for (gamma in gammavalues) {
> svm_model = svm(y ~ ., kernel="radial", type="C-classification", data = trainingdata, gamma = gamma)
> pred = predict(svm_model, trainingdata[, c(1:2)])
> training_error = c(training_error, 1 - Accuracy(trainingdata$y, pred))
> }
```

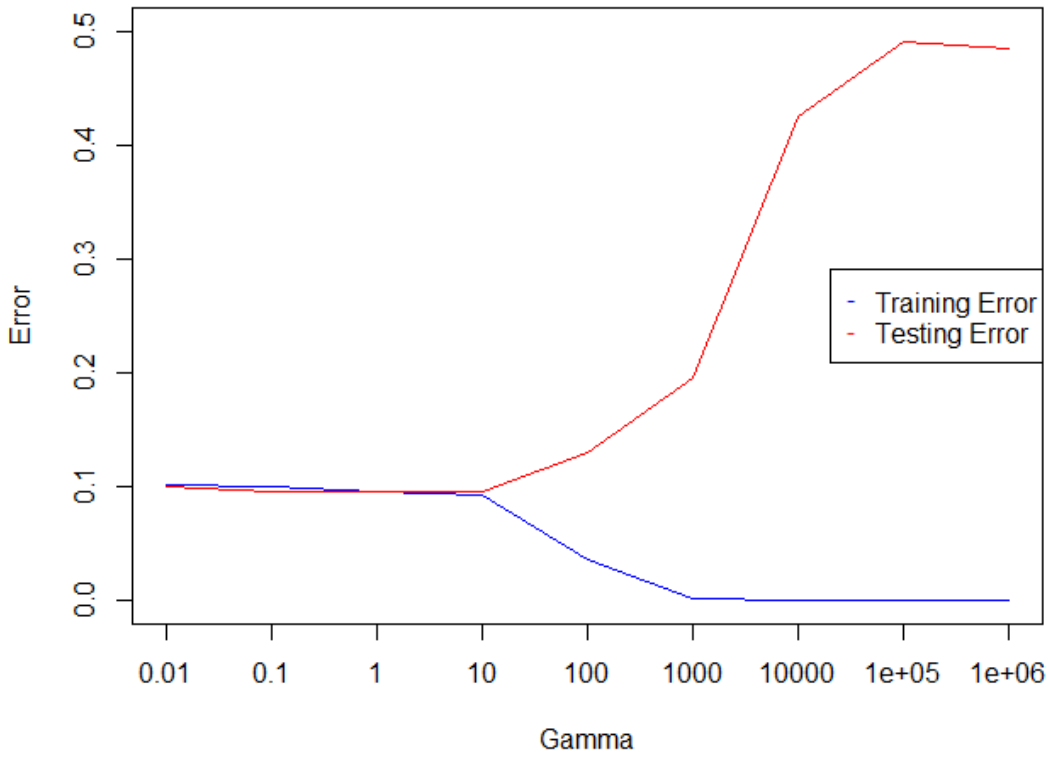
Όμοια, μπορούμε να υπολογίσουμε το testing error:

```
> testing_error = c()
> for (gamma in gammavalues) {
> svm_model = svm(y ~ ., kernel="radial", type="C-classification", data = trainingdata, gamma = gamma)
> pred = predict(svm_model, testdata[, c(1:2)])
> testing_error = c(testing_error, 1 - Accuracy(testdata$y, pred))
> }
```

Σχεδιάζουμε τα δύο σφάλματα στο ίδιο διάγραμμα με τις παρακάτω εντολές:

```
> plot(training_error, type = "l", col="blue", ylim = c(0, 0.5), xlab = "Gamma", ylab = "Error", xaxt = "n")

> axis(1, at = 1:length(gammavalues), labels = gammavalues)

> lines(testing_error, col="red")

> legend("right", c("Training Error", "Testing Error"), pch = c("-","-"), col = c("blue", "red"))
```

Οπότε προκύπτει το παρακάτω διάγραμμα:



3.4 Εφαρμογή k-fold Cross Validation

Θα εφαρμόσουμε k-fold cross validation για να υπολογίσουμε τη βέλτιστη τιμή για την παράμετρο gamma (ερώτημα (στ)). Αρχικά κατασκευάζουμε k folds:

```
> k = 10

> dsize = nrow(trainingdata)

> folds = split(sample(1:dsize), ceiling(seq(dsize) * k / dsize))
```

Στη συνέχεια, δημιουργούμε ένα βρόχο επανάληψης για το gamma, και εντός του βρόχου δημιουργούμε ένα νέο (εμφωλευμένο) βρόχο για τα folds.

```
> accuracies <- c()
> for (gamma in gammavalues) {
>   predictions <- data.frame()
>   testsets <- data.frame()
>   for(i in 1:k){
>     # Select 9 out of 10 folds for training and 1 for validation
>     trainingset <- trainingdata[unlist(folds[-i]),]
>     validationset <- trainingdata[unlist(folds[i]),]
>     # Train and apply the model
>     svm_model = svm(y ~ ., kernel="radial", type="C-classification",
      data = trainingset, gamma = gamma)
>     pred = predict(svm_model, validationset[, c(1:2)])
>     # Save predictions and testsets
>     predictions <- rbind(predictions, as.data.frame(pred))
>     testsets <- rbind(testsets, as.data.frame(validationset[,3]))
>   }
>   # Calculate the new accuracy and add it to the previous ones
>   accuracies = c(accuracies, Accuracy(predictions, testsets))
> }
```

Για κάθε fold υπολογίζουμε το accuracy και τελικά επιλέγουμε το gamma με το μέγιστο accuracy:

```
> print(accuracies)
> bestgamma = gammavalues[which.max(accuracies)]
```

4 Πρόβλημα για Εξάσκηση

Δίνονται τα παρακάτω δεδομένα όπου Y είναι το διάνυσμα κλάσης:

X1	X2	Y
2	2	1
2	-2	1
-2	-2	1
-2	2	1
1	1	2
1	-1	2
-1	-1	2
-1	1	2

Μπορείτε να κατασκευάσετε τα δεδομένα στην R με τις παρακάτω εντολές:

```
> X1 = c(2, 2, -2, -2, 1, 1, -1, -1)
> X2 = c(2, -2, -2, 2, 1, -1, -1, 1)
> Y = c(1, 1, 1, 1, 2, 2, 2, 2)
> alldata = data.frame(X1, X2, Y)
```

Απαντήστε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα με διαφορετικά χρώματα/σύμβολα για κάθε κλάση.

β) Να βρεθεί το υπερεπίπεδο διαχωρισμού χρησιμοποιώντας SVM με πυρήνα RBF και να σχεδιάσετε το υπερεπίπεδο στο σχήμα του ερωτήματος (α).

γ) Που θα κατατάξει ο αλγόριθμος το σημείο (4, 5) μετά την εκπαίδευση;