Evaluación ML I

Ejercicio 2 (SVM)

Inmaculada Perea Fernández

junio 2017

EJERCICIO 2

1. Construya un clasificador SVM con kernel radial para la base de datos ***Glass*** del paquete ***mlbench***, usando validación cruzada con 10 pliegues y parámetros $C = 0.8$ y $\gamma = 0.25$.
2. Determine la tasa de clasificación correcta (*accuracy*) global y en cada pliegue.
3. Realice el ajuste de parámetros para obtener los valores más adecuados de C y γ dentro de los siguientes conjuntos de valores:

$$C \in \{2^{-5}, 2^{-3}, \ldots, 2^{5}\} \qquad \gamma \in \{2^{-7}, 2^{-5}, \ldots, 2^{1}, 2^{3}\}$$

4. Utilice el mejor modelo obtenido del ajuste de parámetros para clasificar las siguientes instancias:

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
1.49	13.45	4.05	1.21	73.18	0.37	7.98	0.2	0
1.52	13.74	3.87	1.29	71.97	0.25	8.02	0	0.13

5. Repita el apartado 3 utilizando validación cruzada con 15 pliegues dentro del procedimiento de ajuste de parámetros (**Indicación**: utilice la función *tune.control*).

Figure 1: Enunciado Ejercicio 2 (Evaluación MLI)

Carga de las librerías necesarias

```
if (!require('mlbench')) install.packages('mlbench'); library('mlbench')
if (!require('e1071')) install.packages('e1071'); library('e1071')
if (!require('caret')) install.packages('caret'); library('caret')
```

Establecimiento de la semilla

```
set.seed(123456789)
```

Carga, inspección y preparación de los datos

El conjunto de datos *Glass* consta de 214 observaciones y 10 variables:

- [,1] **RI**: refractive index
- [,2] **Na**: Sodium

- [,3] **Mg**: Magnesium
- [,4] **Al**: Aluminum
- [,5] **Si**: Silicon
- [,6] **K**: Potassium
- [,7] **Ca**: Calcium
- [,8] **Ba**: Barium
- [,9] **Fe**: Iron
- [,10] **Type**: Type of glass (class attribute)

```
data(Glass)
dim(Glass)
```

```
## [1] 214  10
```

```
summary(Glass)
```

```
##        RI              Na              Mg              Al        
##  Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290  
##  1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190  
##  Median :1.518   Median :13.30   Median :3.480   Median :1.360  
##  Mean   :1.518   Mean   :13.41   Mean   :2.685   Mean   :1.445  
##  3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630  
##  Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500  
##        Si              K               Ca              Ba        
##  Min.   :69.81   Min.   :0.0000   Min.   : 5.430   Min.   :0.000  
##  1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000  
##  Median :72.79   Median :0.5550   Median : 8.600   Median :0.000  
##  Mean   :72.65   Mean   :0.4971   Mean   : 8.957   Mean   :0.175  
##  3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000  
##  Max.   :75.41   Max.   :6.2100   Max.   :16.190   Max.   :3.150  
##        Fe          Type      
##  Min.   :0.00000   1:70  
##  1st Qu.:0.00000   2:76  
##  Median :0.00000   3:17  
##  Mean   :0.05701   5:13  
##  3rd Qu.:0.10000   6: 9  
##  Max.   :0.51000   7:29
```

```
str(Glass)
```

```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
head(Glass)
```

```
##        RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
```

```
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
```

```
table(is.na(Glass))
```

```
##
## FALSE
##  2140
```

La variable objetivo es *Type* y consta de 6 clases (1, 2, 3, 5, 6 y 7). Todos los datos están completos, no hay valores perdidos.

1)

Construya un clasificador SVM con kernel radial para la base de datos *Glass* del paquete *mlbench*, usando validación cruzada con 10 pliegues y parámetros *C=0.8* y *gamma=0.25*

```
(model_1=svm(Type ~ .,           # Fórmula
             data=Glass,         # Dataframe de datos
             kernel="radial",    # Kernel radial
             scale=TRUE,         # Tipifica variables
             gamma=0.25,         # Gamma
             cost=0.8,           # Parámetro de regularización C
             cross=10))          # Validación cruzada con 10 folds
```

```
##
## Call:
## svm(formula = Type ~ ., data = Glass, kernel = "radial", gamma = 0.25,
##     cost = 0.8, cross = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.8
##       gamma:  0.25
##
## Number of Support Vectors:  179
```

2)

Determine la tasa de clasificación correcta (accuracy) global y en cada pliegue

```
cat("Accuracy global (modelo 1): ", round(model_1$tot.accuracy, 3), "\n")
```

```
## Accuracy global (modelo 1):  70.093
```

```
cat("Accuracy para cada pliegue: \n", round(model_1$accuracies, 3) )
```

```
## Accuracy para cada pliegue:
##  76.19 57.143 59.091 71.429 59.091 85.714 76.19 86.364 71.429 59.091
```

3)

Realice el ajuste de parámetros para obtener los valores más adecuados de *C* y *gamma* dentro de los siguientes conjuntos de valores:

$$C \in \{2^{-5}, 2^{-3}, \ldots, 2^{5}\} \qquad \gamma \in \{2^{-7}, 2^{-5}, \ldots, 2^{1}, 2^{3}\}$$

Figure 2: valores C y gamma

Ajuste de los parámetros del kernel (*gamma*) y de regularización (*C*) con la función *tune.svm*

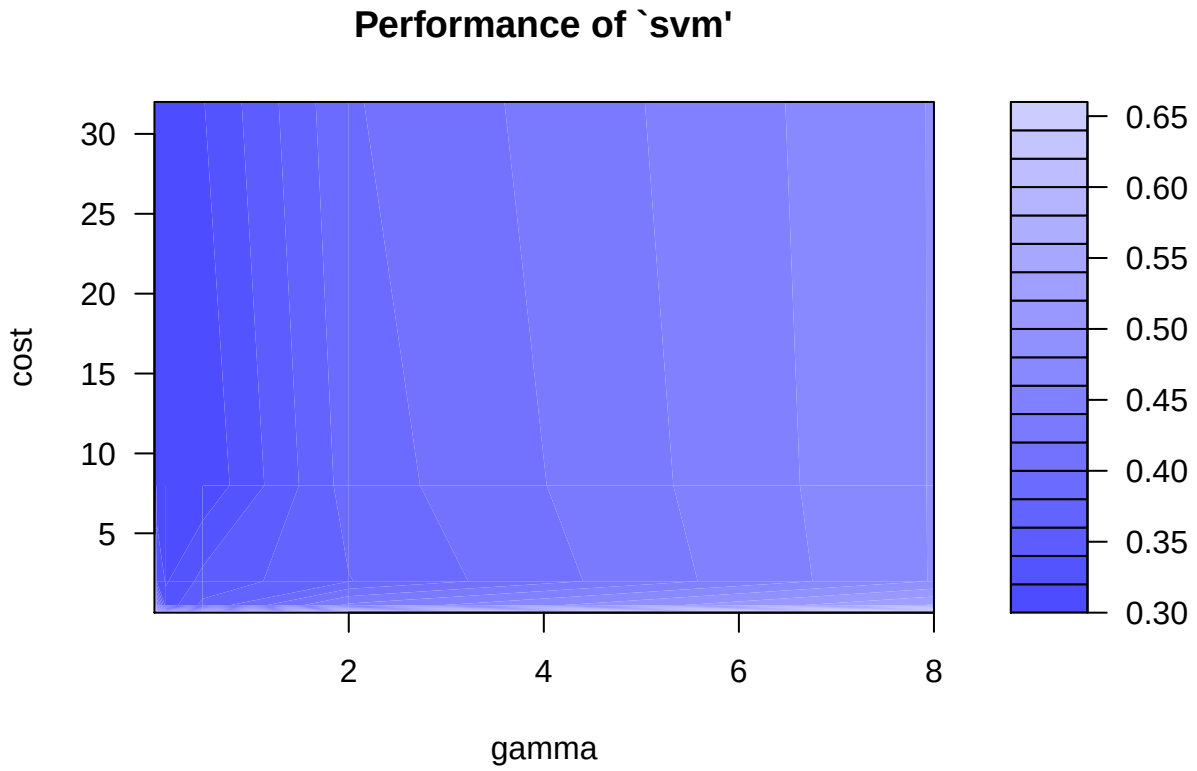
```
t1=tune.svm(Type ~ .,
            data=Glass,
            gamma=2^seq(from=-7,to=3, by=2),
            cost=2^seq(from=-5,to=5,by=2))

summary(t1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.5    8
##
## - best performance: 0.304329
##
## - Detailed performance results:
##        gamma     cost     error dispersion
## 1  0.0078125  0.03125 0.6445887 0.08226880
## 2  0.0312500  0.03125 0.6445887 0.08226880
## 3  0.1250000  0.03125 0.6445887 0.08226880
## 4  0.5000000  0.03125 0.6445887 0.08226880
## 5  2.0000000  0.03125 0.6445887 0.08226880
## 6  8.0000000  0.03125 0.6445887 0.08226880
## 7  0.0078125  0.12500 0.6445887 0.08226880
## 8  0.0312500  0.12500 0.5333333 0.09701542
## 9  0.1250000  0.12500 0.4679654 0.07995843
## 10 0.5000000  0.12500 0.4904762 0.10811057
## 11 2.0000000  0.12500 0.5515152 0.09154704
## 12 8.0000000  0.12500 0.6445887 0.08226880
## 13 0.0078125  0.50000 0.5283550 0.08587041
## 14 0.0312500  0.50000 0.4493506 0.08986783
## 15 0.1250000  0.50000 0.3274892 0.07843513
## 16 0.5000000  0.50000 0.3649351 0.10795155
## 17 2.0000000  0.50000 0.4445887 0.10792262
## 18 8.0000000  0.50000 0.5419913 0.08575970
## 19 0.0078125  2.00000 0.4484848 0.06902164
## 20 0.0312500  2.00000 0.3367965 0.09926445
## 21 0.1250000  2.00000 0.3179654 0.10881250
## 22 0.5000000  2.00000 0.3463203 0.08321378
```

```
## 23 2.0000000  2.00000 0.3792208 0.12150295
## 24 8.0000000  2.00000 0.4811688 0.07059692
## 25 0.0078125  8.00000 0.3452381 0.11036485
## 26 0.0312500  8.00000 0.3086580 0.10270785
## 27 0.1250000  8.00000 0.3181818 0.10495294
## 28 0.5000000  8.00000 0.3043290 0.09714948
## 29 2.0000000  8.00000 0.3887446 0.14278497
## 30 8.0000000  8.00000 0.4811688 0.07059692
## 31 0.0078125 32.00000 0.3268398 0.12197417
## 32 0.0312500 32.00000 0.3177489 0.10776139
## 33 0.1250000 32.00000 0.3043290 0.12750451
## 34 0.5000000 32.00000 0.3188312 0.10257523
## 35 2.0000000 32.00000 0.3978355 0.13051308
## 36 8.0000000 32.00000 0.4811688 0.07059692
```

Visualización del resultado del ajuste de parámetros

```
plot(t1)
```



Modelo con los parámetros ajustados

```
(model_2=t1$best.model)
```

```
##
## Call:
## best.svm(x = Type ~ ., data = Glass, gamma = 2^seq(from = -7,
##     to = 3, by = 2), cost = 2^seq(from = -5, to = 5, by = 2))
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  8
##       gamma:  0.5
##
## Number of Support Vectors:  165
```

Medida de rendimiento

```
cat("Tasa de clasificación incorrecta (modelo 2): ", round(t1$best.performance, 3), "\n")
```

```
## Tasa de clasificación incorrecta (modelo 2):  0.304
```

```
cat("Accuracy global (modelo 2): ", round(100*(1-t1$best.performance), 3), "\n")
```

```
## Accuracy global (modelo 2):  69.567
```

Para el modelo 1 obtuvimos un accuracy de 70.093, mayor que el obtenido con el modelo 2, por tanto el nuevo modelo ajustando parámetros C y *gamma* es peor que el construido en el primer apartado.

4)

Utilice el mejor modelo obtenido del ajuste de parámetros para clasificar las siguientes instancias:

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
1.49	13.45	4.05	1.21	73.18	0.37	7.98	0.2	0
1.52	13.74	3.87	1.29	71.97	0.25	8.02	0	0.13

Figure 3: Valores instancia a predecir

Creación de las instancias

```
instancia_1 = data.frame(RI=1.49, Na=13.45, Mg=4.05, Al=1.21, Si=73.18,
                         K=0.37, Ca=7.98, Ba=0.2, Fe=0)


instancia_2 = data.frame(RI=1.52, Na=13.74, Mg=3.87, Al=1.29, Si=71.97,
                         K=0.25, Ca=8.02, Ba=0,   Fe=0.13)
```

Predicción de la instancia 1

```
predict(model_1, instancia_1)
```

```
## 1
## 2
## Levels: 1 2 3 5 6 7
```

Predicción de la instancia 2

```
(predict(model_1, instancia_2))
```

```
## 1
```

```
## 2
## Levels: 1 2 3 5 6 7
```

Ambas instancias son clasificadas como pertenecientes a la clase “2” segun el modelo con parámetros ajustados *modelo_2*.

5)

Repita el apartado 3 utilizando validación cruzada con 15 pliegues dentro del procedimiento de ajuste de parámetros (Indicación: utilice la función tune.control)

```
#?tune.control
#?tune.svm

tc=tune.control(sampling="cross",  # validacion cruzada
                cross=15,          # 15 pliegues
                nrepeat=1,
                best.model=TRUE)

t2=tune.svm(Type ~ .,
            data=Glass,
            tunecontrol=tc,
            gamma=2^seq(from=-7,to=3, by=2),
            cost=2^seq(from=-5,to=5,by=2))

summary(t2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 15-fold cross validation
##
## - best parameters:
##  gamma cost
##  0.125   32
##
## - best performance: 0.2803175
##
## - Detailed performance results:
##        gamma     cost     error dispersion
## 1  0.0078125  0.03125 0.6555556 0.12689339
## 2  0.0312500  0.03125 0.6365079 0.15804387
## 3  0.1250000  0.03125 0.6365079 0.15804387
## 4  0.5000000  0.03125 0.6365079 0.15804387
## 5  2.0000000  0.03125 0.6412698 0.14743973
## 6  8.0000000  0.03125 0.6365079 0.15804387
## 7  0.0078125  0.12500 0.6555556 0.12689339
## 8  0.0312500  0.12500 0.5336508 0.15477987
## 9  0.1250000  0.12500 0.4720635 0.15424946
## 10 0.5000000  0.12500 0.4961905 0.13739206
## 11 2.0000000  0.12500 0.5714286 0.13274896
## 12 8.0000000  0.12500 0.6365079 0.15804387
## 13 0.0078125  0.50000 0.5434921 0.11097627
## 14 0.0312500  0.50000 0.4396825 0.12463964
```

```
## 15 0.1250000  0.50000 0.3501587 0.12768486
## 16 0.5000000  0.50000 0.3600000 0.12643475
## 17 2.0000000  0.50000 0.4117460 0.09284173
## 18 8.0000000  0.50000 0.5380952 0.12384876
## 19 0.0078125  2.00000 0.4447619 0.13371907
## 20 0.0312500  2.00000 0.3457143 0.11806122
## 21 0.1250000  2.00000 0.3088889 0.13535717
## 22 0.5000000  2.00000 0.3171429 0.11796515
## 23 2.0000000  2.00000 0.3644444 0.10752188
## 24 8.0000000  2.00000 0.4634921 0.13246396
## 25 0.0078125  8.00000 0.3650794 0.12797375
## 26 0.0312500  8.00000 0.3088889 0.11176162
## 27 0.1250000  8.00000 0.3222222 0.11539522
## 28 0.5000000  8.00000 0.2939683 0.10132691
## 29 2.0000000  8.00000 0.3730159 0.10039873
## 30 8.0000000  8.00000 0.4634921 0.13246396
## 31 0.0078125 32.00000 0.3273016 0.12518165
## 32 0.0312500 32.00000 0.3177778 0.12887928
## 33 0.1250000 32.00000 0.2803175 0.11458106
## 34 0.5000000 32.00000 0.3076190 0.10788181
## 35 2.0000000 32.00000 0.3774603 0.10659295
## 36 8.0000000 32.00000 0.4634921 0.13246396
```

```
model_3=t2$best.model
```

```
cat("Tasa de clasificación incorrecta (modelo 3): ", round(t2$best.performance, 3), "\n")
```

```
## Tasa de clasificación incorrecta (modelo 3):  0.28
```

```
cat("Accuracy global (modelo 3): ", round(100*(1-t2$best.performance), 3), "\n")
```

```
## Accuracy global (modelo 3):  71.968
```

Comparativa

A continuación construiremos una tabla comparativa con el accuracy de los tres modelos obtenidos.

```
table_model1=c(model_1$tot.accuracy, 0.01*(100-model_1$tot.accuracy))
table_model2=c(100*(1-t1$best.performance), t1$best.performance)
table_model3=c(100*(1-t2$best.performance), t2$best.performance)

tabla_resumen = data.frame (round(rbind(table_model1, table_model2, table_model3), 3),
                            row.names=c("modelo 1 (folds=10, C=0.8, gamma=0.25)",
                                        "modelo 2 (folds=10, C=8  , gamma=0.5)",
                                        "modelo 3 (folds=15, C=32 , gamma=0.125)"))

print(knitr::kable(tabla_resumen, format = "pandoc",
                   col.names = c("Accuracy", "Tasa clasificación incorrecta"),
                   align='c'))
```

	Accuracy	Tasa clasificación incorrecta
modelo 1 (folds=10, C=0.8, gamma=0.25)	70.093	0.299
modelo 2 (folds=10, C=8 , gamma=0.5)	69.567	0.304
modelo 3 (folds=15, C=32 , gamma=0.125)	71.968	0.280

El mejor modelo, que presenta mayor accuracy y por tanto menor tasa de clasificación incorrecta, es el último modelo construido con parámetros ajustados usando validación cruzada con 15 pliegues *modelo_3*.