Exercício 5 - Aplicação SVM

Rodrigo Machado Fonseca - 2017002253

November 23, 2021

1 Introdução

Neste exercício utilizaremos o *Support Vector Machine* (SVM) para resolver o problema de classificação de tipos de vidros do banco de dados Glass (base nativa da Linguagem) a partir de suas características químicas.

2 Metodologia

A priori, será carregada a base de dados Glass do R. A seguir pode-se ver um pequena amostra da base de dados:

```
       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    1
2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0  0    1
3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0  0    1
4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0  0    1
5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0  0    1
```

Em sequência, iremos utilizar a normalização para que todas variáveis possam ter o mesmo peso no modelo, com a seguinte equação:

$$x_i' = \frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

Neste problema as classes estão desbalanceadas como pode ser visto a seguir, onde cada coluna representa o *label* e a linha representa a quantidade de dados de cada classe:

```
y
 1  2  3  5  6  7
70 76 17 13  9 29
```

Por fim iremos separar 70% do conjunto para treino e 30% para teste. Como neste exercício as classes estão desbalanceadas iremos forçar que todas tenham a mesma proporção do conjunto original. Para isso construímos a seguinte função:

```
> sample_proportion <- function(x, y){
+   labels <- names(table(y))
+   x_train <- matrix(ncol = ncol(x))
+   y_train <- c()
+   x_test <- matrix(ncol = ncol(x))
+   y_test <- c()
+   for(i in labels){
+     aux <- strtoi(i)
+     y_aux <- y[y==aux]
+     x_aux <- x[y==aux, ]
+     index <- sample(1:nrow(x_aux), length(1:nrow(x_aux)))
+     x_aux <- x_aux[index,1:ncol(x_aux)]
+     y_aux <- y_aux[index]
+     training_sample_number = round(nrow(x_aux)*0.7)+1
+     x_aux_train <- x_aux[1:training_sample_number,]
+     y_aux_train <- y_aux[1:training_sample_number]
+     x_aux_test <- x_aux[(training_sample_number+1):nrow(x_aux), ]
+     y_aux_test <- y_aux[(training_sample_number+1):nrow(x_aux)]
+     y_train <- c(y_train, y_aux_train)
+     x_train <- rbind(x_train, x_aux_train)
+     y_test <- c(y_test, y_aux_test)
+     x_test <- rbind(x_test, x_aux_test)
+   }
+   x_train <- x_train[2:nrow(x_train), 1:ncol(x_train)]
+   index <- sample(1:nrow(x_train), length(1:nrow(x_train)))
+   x_train <- x_train[index,1:ncol(x_train)]
+   y_train <- y_train[index]
+   x_test <- x_test[2:nrow(x_test), 1:ncol(x_test)]
+   index <- sample(1:nrow(x_test), length(1:nrow(x_test)))
+   x_test <- x_test[index,1:ncol(x_test)]
```

```
+   y_test <- y_test[index]
+   return(list(x_train,
+               y_train,
+               x_test,
+               y_test))
+ }
```

3 Resultados

Neste experimento iremos para cada conjunto de parâmetros "kpar" e "C" faremos ele 10 vezes e por fim analisaremos a média e o desvio padrão.

```
> set.seed(1)
> accuracy_final <- matrix(nrow=10, ncol=9)
> for(i in 1:10){
+   samples <- sample_proportion(x, y)
+   x_train <- samples[[1]]
+   y_train <- samples[[2]]
+   x_test <- samples[[3]]
+   y_test <- samples[[4]]
+   r <- c(0.025, 0.025, 0.025, 0.015, 0.015, 0.015, 0.01, 0.01, 0.01)
+   C_par <- c(700, 500, 900, 800, 900, 930, 500, 600, 800)
+   results <- training_svm(x_train, y_train, x_test, y_test, r, C_par)
+   svm <- results[[1]]
+   accuracy_list <- results[[2]]
+   accuracy_final[i,] <- as.matrix(accuracy_list, nrow=1)
+ }
```

Nas tabelas a seguir estão representadas as médias e os desvios padrão para cada experimento.

```
  colMeans.accuracy_final.
1                0.7474576
2                0.7440678
3                0.7440678
4                0.7406780
5                0.7423729
6                0.7406780
```

```
7                0.7440678
8                0.7406780
9                0.7406780

          sd
1 0.04889536
2 0.05018400
3 0.04689606
4 0.06242874
5 0.06381931
6 0.06493489
7 0.05841382
8 0.05819483
9 0.06394422
```

4 Discussão

Neste exercício os parâmetros "kpar" e "C" foram escolhidos novamente de forma aleatória, e foram ajustados conforme a acurácia dos resultados. Isso mostrou-se extremamente ineficiente. Para melhorar o desempenho do algoritmo sugiro duas hipóteses: (1) definir os parâmetros, pelo menos os primeiros, com alguma expressão envolvendo a dimensionalidade do problema; (2) rodar o algoritmo em paralelo com um conjunto de valores, e para cada conjunto escolher o melhor e criar um novo conjunto de acordo com uma distribuição de probabilidade. No final, comparar entre cada resultado obtido e escolher o melhor. O único problema do passo 2 pode ser o custo computacional para construí-lo.

O melhor resultado obtido foi para "kpar" igual a 0.025 e "C" igual a 700. Nota-se que este valor ($\approx 75\%$) é muito menor do que o obtido pelo problema anterior (100 %). Esse problema possui 9 dimensões e 6 classes, o que aumenta consideravelmente a complexidade do problema.

Logo, com o experimento foi possível compreender melhor o comportamento do algoritmo SVM, além de que conseguimos construir e validar um classificador SVM. É possível afirmar que a prática foi bem executada e o classificador construído apresentou um bom razoável.