# Tarea 6 - Reconocimiento estadístico de patrones

Erick Salvador Alvarez Valencia

CIMAT A.C.,
erick.alvarez@cimat.mx

**Resumen** En el presente reporte se hablará sobre la resolución de los ejercicios de la sexta tarea de reconocimiento de patrones, de la misma manera se presentará el código de los mismos y los resultados que fueron obtenidos en los problemas que lo requieren.

## 1. Problema 1

Consideramos un problema de clasificación binaria (con codificación 1 y -1). Se quiere construir un clasificador de la forma

$$y = sign(f(x))$$

donde $f$ minimiza la función de costo $E(1-Yf(X))^2$. Verifica que para la $f$ que minimiza esta función de costo, $y(x) = sign(f(x))$ corresponde al Clasificador Bayesiano óptimo.

**Solución** : Empezamos desarrollando la esperanza:

$$\begin{aligned} E(1-Yf(x))^2 &= E(1-2Yf(x)+(Yf(x))^2) \\ &= E_X E_{Y|X=x}(1-2Yf(x)+(Yf(x))^2) \end{aligned} \tag{1}$$

De (1) basta que para todo $X$ se minimice

$$\min_{f(x)} E_{Y|X=x}(1-2Yf(x)+(Yf(x))^2) \tag{2}$$

Aplicamos la definición de esperanza para $Y \in \{-1,1\}$

$$\begin{aligned} E_{Y|X=x}(1-2Yf(x)+(Yf(x))^2) = (1-2f(x)+f(x)^2)P(Y=1|X=x)+ \\ (1+2f(x)+f(x)^2)P(Y=-1|X=x) \\ = P(Y=1|X=x)-2f(x)P(Y=1|X=x)+f(x)^2P(Y=1|X=x)+ \\ P(Y=-1|X=x)+2f(x)P(Y=-1|X=x)+f(x)^2P(Y=-1|X=x) \\ = 1+2f(x)(P(Y=-1|X=x)-P(Y=1|X=x))+f(x)^2(P(Y=1|X=x)+ \\ P(Y=-1|X=x)) \\ E_{Y|X=x} = 1+2f(x)(1-2P(Y=1|X=x))+f(x)^2 \end{aligned} \tag{3}$$

Ahora, para encontrar el mínimo, derivamos lo anterior con respecto a $f(x)$, lo que nos queda:

$$\frac{dE_{Y|X=x}}{df(x)} = 2(1 - 2P(Y = 1|X = x)) + 2f(x) \tag{4}$$

Igualamos a cero y despejamos $f(x)$

$$\begin{aligned} 2(1 - 2P(Y = 1|X = x)) + 2f(x) = 0 \\ f(x) = 2P(Y = 1|X = x) - 1 \end{aligned} \tag{5}$$

Para demostrar que lo anterior es un mínimo local hacemos el criterio de la segunda derivada. Por lo que volvemos a derivar (4).

$$\frac{d^2E_{Y|X=x}}{df(x)^2} = 2 \tag{6}$$

Vemos que la derivada es positiva, por lo que tenemos un mínimo local. Ahora desarrollamos por casos:

1. Si $P(Y = 1|X = x) > \frac{1}{2} \rightarrow f(x) > 0$ y por lo tanto $\hat{y}(x) = 1$.
2. Si $P(Y = 1|X = x) < \frac{1}{2} \rightarrow f(x) < 0$ y por lo tanto $\hat{y}(x) = -1$.

A lo cual tenemos que lo anterior es el Clasificador Bayesiano óptimo y lo podemos escribir de la siguiente forma:

$$\hat{y}(x) = sign(\log(\frac{P(Y = 1|X = x)}{P(Y = -1|X = x)})) \tag{7}$$

## 2. Problema 2

En este ejercicio usamos un SVM para clasificar dos tipos (clases) de textos. Cada uno de ustedes pueden elegir cuales dos clases de textos tomar pero debe ser sobre un tema de actualidad (no los libros de Oz, ni el Don Quijote ...).

**Solución** : Para este problema se consiguió un conjunto de datos el cual se compone de correos electrónicos que hablan de distintos temas (conjunto usado en machine learning), y entre ellos se eligieron dos en particular: Çarreras de autosz "Motocicletas", esto porque son parecidos en ciertos aspectos y por ende los datos no son separables tan fácilmente. Lo siguiente fue crear el clasficiador y para ello se usó como base el código proporcionado en la descripción de la tarea, esto para importar el conjunto de textos, realizarles un filtrado sobre ciertos caracteres, y posteriormente hacer el conteo de las palabras que más se repetían en ambos conjuntos, con esto se generaba un data frame con las palabras

más características de ambas categorías, en donde las filas correspondían a los archivos y las columnas a las palabras. A continuación se añadió la etiqueta correspondiente a cada fila del data frame indicando a que clase pertenecía dicho archivo y de ahí se partió el data frame en dos conjuntos, uno de prueba y otro de entrenamiento, con el conjunto de prueba conteniendo el 80 % de las filas del data frame original.
Lo siguiente fue usar el método SVM para realizar la clasificación, para dicho algoritmo se usó la librería **caret** con el método *train*. La elección de dicha librería es porque provee un modo de entrenamiento en el cual se puede indicar el uso del modo cross validation, esto para evitar el sobre-ajuste hacia el conjunto de entrenamiento. Además de que permite como entrada un vector de números los cuales corresponden a los valores de penalización para la función de costo y por lo tanto se puede ver el nivel de ajuste que se tuvo con cada uno de estos modelos.
A continuación se muestran los resultados obtenidos al aplicar el método SVM al conjunto de datos en donde se usaron los siguientes parámetros:

1. **Número de ocurrencias mínimas en las palabras seleccionadas de ambos conjuntos** : 7.
2. **Número de archivos en el conjunto 1 (Carreras de autos)** : 990.
3. **Número de archivos en el conjunto 2 (Motocicletas)** : 994.
4. **Número de pruebas en la validación cruzada** : 10.
5. **Valores de penalización usados en las pruebas** : $\{0,0{,}01, 0{,}1, 0{,}5, 1{,}5, 2{,}5, 5, 10\}$.
6. **Kernel usado en el método** : Lineal.

Los resultados obtenidos fueron los siguientes:

```
+     avg_acc = avg_acc + 1;
+   }
+ }
> avg_acc = avg_acc / length(test_labels);
> print(sprintf("Porcentaje de aciertos con el conjunto de prueba: %f\n", avg_acc));
[1] "Porcentaje de aciertos con el conjunto de prueba: 0.974811\n"
```

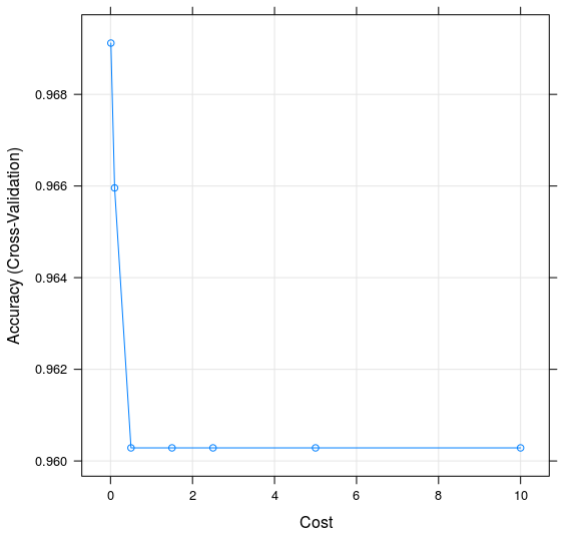
(a) Figura 1. Precisión del modelo con los datos de prueba.

En la Figura 1. se muestra el porcentaje de datos correctamente clasificados por el algoritmo usando el conjunto de prueba, el cual corresponde al 97 % de los datos, un gran accuracy obtenido.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187   8
         1   2 200

               Accuracy : 0.9748
                 95% CI : (0.9542, 0.9879)
    No Information Rate : 0.5239
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9496
 Mcnemar's Test P-Value : 0.1138

            Sensitivity : 0.9894
            Specificity : 0.9615
         Pos Pred Value : 0.9590
         Neg Pred Value : 0.9901
             Prevalence : 0.4761
         Detection Rate : 0.4710
   Detection Prevalence : 0.4912
      Balanced Accuracy : 0.9755

       'Positive' Class : 0
```

(b) Figura 2. Matriz de confusión del modelo.

En la Figura 2. se muestra una matriz de confusión creada en base a las predicciones hechas con el conjunto de prueba. En dicha imagen se muestra primeramente una matriz que contiene los verdaderos positivos (arriba izquierda), falsos positivos (arriba derecha), falsos negativos (abajo izquierda), verdaderos negativos (abajo derecha). Se puede ver que en general fueron más los falsos positivos que el algoritmo clasificó que los falsos negativos.
Posteriormente se muestran varias métricas tomadas de las predicciones hechas, la primera de ellas es la precisión del algoritmo, la cual fue mostrada previamente, luego tenemos algunas otras métricas como sensitividad y especificidad las cuales también cuentan con un alto porcentaje.




(c) Figura 3. Gráfica del entrenamiento en relación a los parámetros de penalización.

Finalmente tenemos la Figura 3. la cual muestra una gráfica hecha por el modelo de entrenamiento. Dicha gráfica indica la precisión del modelo (SVM) en base al parámetro de penalización usado.
Como se mencionó anteriormente, se usaron diferentes parámetros de penalización para ver la eficiencia de los mismos en el modelo. En la gráfica se aprecia que los parámetros que mejor ajuste dieron fueron 0 y 0,01 con lo cual se obtuvo un 96.6 % y un poco más de precisión, y con los demás parámetros de obtuvo un 96 % de precisión, por lo que se concluye que no fue un descenso tan grande en los ajustes.

A continuación se adjunta el código usado en este ejercicio, de igual manera dicho código se enviará en los archivos de la tarea.

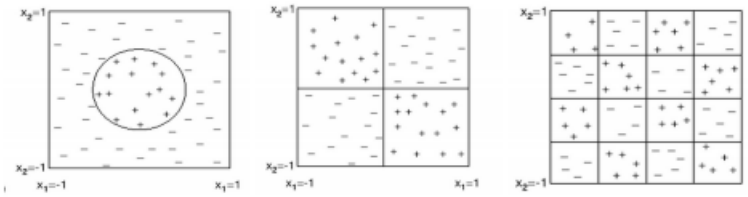
```
library("tm")
library("SnowballC")
library("e1071")
library("caret")

#Directorio donde estan los dos conjuntos de datos
data_dir <- "Directorio de los datos"

#Leer los datos y crear la matriz de ocurrencias de palabras
text <- Corpus(DirSource(data_dir))
matrix_d <- DocumentTermMatrix(text, control = list(minWordLength = 2,
```

```
minDocFreq = 5, stemming = T, removePunctuation = T, removeNumbers = T))
d <- DocumentTermMatrix(text, list(dictionary = findFreqTerms(matrix_d, 7)))

#Numero de archivos por cada clase
c1_n <- 990
c2_n <- 994
train_data_per <- 0.8
#Anadir las etiquetas de las clases a la matriz anterior
classes_v <- factor(c(rep(0, c1_n), rep(1, c2_n)))
a1 <- data.frame(as.matrix(d), classes_v)
colnames(a1)[ncol(a1)] <- 'y'
#Crear aleatoriamente los conjuntos de prueba y entrenamiento
shuffled_data <- a1[sample(nrow(a1)), ]
smp_size <- floor(train_data_per * nrow(a1))
train_ind <- sample(seq_len(nrow(a1)), size = smp_size)

train_data <- shuffled_data[train_ind, ]
test_data <- shuffled_data[-train_ind, ]

#Crear el modelo usando una SVM con kernel lineal
train_control <- trainControl(method = "cv", number = 10)
costs <- expand.grid(C = c(0, 0.01, 0.1, 0.5, 1.5, 2.5, 5, 10))
model <- train(y ~ ., data = train_data, trControl = train_control,
method = "svmLinear", tuneGrid = costs, tuneLength = 5)

#Predecir con datos de prueba
test_labels <- test_data[, ncol(test_data)]
test_data_f <- test_data[, -ncol(test_data)]
test_pred <- predict(model, test_data_f)
#Obtener el porcentaje de aciertos obtenidos en las predicciones
avg_acc <- 0.0
r1 <- as.vector(test_labels, mode = "integer")
r2 <- as.vector(test_pred, mode = "integer")
for(i in 1 : length(r1)) {
  if(r1[i] == r2[i]) {
    avg_acc = avg_acc + 1;
  }
}
avg_acc = avg_acc / length(test_labels);
print(sprintf("Porcentaje de aciertos con el conjunto de prueba: %f\n",
avg_acc));

#Matriz de confusion
confusionMatrix(test_pred, test_labels)
#Plot del accuracy generado en la CV con los diferentes parametros de penalizaci
```

**plot** (**model**)

## 3. Problema 3

Supongamos que tenemos los datos bidimensionales de clasificación binaria (+, -) de la Figura 1(a). Como el área de decisión es $x_1^2 + x_2^2 - 1 = 0$ es suficiente hacer una transformación polinomial de grado 2, i.e. es suficiente trabajar con un kernel polinomial de grado 2.



(a) (b) (c)

Figura 1

(d)

¿De qué grado mínimo debe ser el kernel polinomial para los datos de la Figura 1(b) y de la Figura 1(c)?

**Solución** : Para la Figura 1(b) podemos notar fácilmente que los datos no se pueden separar con un polinomio lineal, por lo que el grado no puede ser 1. Lo que si se puede notar es que estos son separados por los ejes $X$ y $Y$ del plano $R^2$, es decir, con el polinomio $xy = 0$ tenemos el área de decisión para clasificar los datos a como se tiene en la Figura. Esto ya que si $xy > 0$ se asigna la clase 0 (signos menos), y por el contrario si $xy < 0$ se asigna la clase 1 (signos positivos). Por lo que se concluye que con el polinomio de grado 2 se clasifican los datos.

Para la Figura 1(c) es un poco más complejo, lo primero que podemos ver es que el polinomio que debe servir de kernel tiene que valer cero en las rectas $x = \{0, -0{,}5, 0{,}5\}$ y $y = \{0, -0{,}5, 0{,}5\}$.
Ahora, como sabemos que $f(x, y)$ se anula en $x = 0$ lo podemos factorizar como

$$f(x, y) = xg(x, y)$$

donde $g(x, y)$ es otro polinomio con coeficientes reales. Ahora de la misma forma vemos que g(x, y) se debe anular en $x = 0{,}5$ por lo que $g(x, y)$ podemos expresarlo como $g(x, y) = (x - 0{,}5)h(x, y)$, y de esta forma aplicamos el mismo análisis para los demás valores donde se anula el polinomio y finalmente nos queda:

$$f(x,y) = x(x-0{,}5)(x+0{,}5)y(y-0{,}5)(y+0{,}5) = 0 \tag{8}$$

Este polinomio describe el área de decisión para la clasificación de los datos, y si lo desarrollamos podemos ver que es de grado 6, ahora si se desarrollaran todos los casos con los datos se vería que este polinomio clasifica correctamente a los mismos. Finalmente se concluye que el grado mínimo para el polinomio en este caso es 6.