Tarea 5 - 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 quinta 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

En este ejercicio demostramos la convergencia del algoritmo de perceptron que vimos en la clase para el caso cuando los datos son linealmente separables y donde incluimos en las x_i's ya un componente igual a 1 para poder limitarnos a hiperplanos que pasan por el origen (es decir $\alpha = 0$ y ya no aparece en el algoritmo).
La clase de x_i denotamos con y_i, donde y_i vive en $\{-1, 1\}$.
Para simplificar la notación y los cálculos, definimos

$$z_i = x_i * y_i$$

así buscamos β tal que

$$\forall i : \beta^t z_i > 0$$

.

Siempre podemos rescalar las observaciones tal que la norma de los z_i's es menor que 1.
Usando la notación de lo anterior y eligiendo $\eta = 1$, en cada iteración calculamos

$$\beta^{t+1} = \beta^t + z_i I(\beta^t z_i \leq 0) \tag{1}$$

(a) Explica que, si los datos son linealmente separables, existe una β_{opt} tal que

$$\beta_{opt} z_i \geq 1$$

Solución : Como que sabe que el conjunto de datos x_i es linealmente separable entonces deben existir una w y una k tal que:

$$\begin{aligned} w^T x_i \geq k \ si \ y_i = 1 \\ w^T x_i < -k \ si \ y_i = -1 \end{aligned} \tag{2}$$

Es decir, podemos encontrar un hiperplano tal que este corte un conjunto de puntos x_i dentro de un espacio n-dimensional en dos conjuntos de puntos separados. Ahora, sin pérdida de generalidad suponiendo que $k > 0$ podemos dividir ambos lados de la ecuación (2) por k.

Primero trabajamos el caso donde $y_i = 1$.

$$\begin{aligned} \frac{w^T x_i}{k} &\geq \frac{k}{k} \\ \frac{w^T x_i}{k} &\geq 1 \end{aligned} \tag{3}$$

Ahora multiplicamos ambos lados de la inecuación por y_i.

$$\begin{aligned} (y_i) * \frac{w^T x_i}{k} &\geq 1 * (y_i) \\ \frac{w^T}{k} z_i &\geq 1 \end{aligned} \tag{4}$$

Ahora hacemos lo mismo para el caso donde $y_i = -1$.

$$\begin{aligned} \frac{w^T x_i}{k} &< -\frac{k}{k} \\ \frac{w^T x_i}{k} &< -1 \\ (y_i) * \frac{w^T x_i}{k} &< -1 * (y_i) \\ \frac{w^T}{k} z_i &> 1 \end{aligned} \tag{5}$$

De esta forma tenemos que para todo i se cumple que $\frac{w^T}{k} z_i \geq 1$ y de esta forma queda demostrado que existe un $\beta_{opt} = \frac{w}{k}$ que cumple lo anterior mencionado.

(b) Usando lo anterior, verifica que si obtenemos β^{t+1} usando (1) para una z_i mal clasificada:

$$0 \leq ||\beta^{t+1} - \beta_{opt}||^2 \leq ||\beta^t - \beta_{opt}||^2 - 1 \tag{6}$$

Solución : Empezamos trabajando con la primer parte de la desigualdad

$$\begin{aligned} 0 &\leq ||\beta^{t+1} - \beta_{opt}||^2 \\ 0 &\leq (\beta^{t+1} - \beta_{opt})^T (\beta^{t+1} - \beta_{opt}) \\ 0 &\leq ((\beta^{t+1})^T - (\beta_{opt})^T)(\beta^{t+1} - \beta_{opt}) \\ 0 &\leq (\beta^{t+1})^T \beta^{t+1} - (\beta^{t+1})^T \beta_{opt} - (\beta_{opt})^T \beta^{t+1} + (\beta_{opt})^T \beta_{opt} \\ 0 &\leq (\beta^{t+1})^T \beta^{t+1} - 2(\beta^{t+1})^T \beta_{opt} + (\beta_{opt})^T \beta_{opt} \end{aligned} \tag{7}$$

Ahora de la ecuación anterior se sustituye β^{t+1} por $\beta^t + z_i$, esto debido a que por hipótesis z_i es un dato mal clasificado.

$$\begin{aligned}
0 &\leq (\beta^t + z_i)^T(\beta^t + z_i) - 2(\beta^t + z_i)^T\beta_{opt} + (\beta_{opt})^T\beta_{opt} \\
0 &\leq ((\beta^t)^T + z_i^T)(\beta^t + z_i) - 2((\beta^t)^T + z_i^T)\beta_{opt} + (\beta_{opt})^T\beta_{opt} \\
0 &\leq (\beta^t)^T\beta^t + (\beta^t)^T z_i + z_i^T\beta^t + z_i^T z_i - 2(\beta^t)^T\beta_{opt} - 2z_i^T\beta_{opt} + (\beta_{opt})^T\beta_{opt} \\
0 &\leq (\beta^t)^T\beta^t - 2(\beta^t)^T\beta_{opt} + (\beta_{opt})^T\beta_{opt} + (\beta^t)^T z_i + z_i^T\beta^t + z_i^T z_i - 2z_i^T\beta_{opt} \\
0 &\leq ||\beta^t - \beta_{opt}||^2 + 2(\beta^t)^T z_i + ||z_i||^2 - 2z_i^T\beta_{opt}
\end{aligned} \tag{8}$$

Por otra parte tenemos que

$$\begin{aligned}
\beta_{opt} z_i &\geq 1 \\
-2\beta_{opt} z_i &\leq -2
\end{aligned} \tag{9}$$

Teniendo lo anterior podemos ver que la ecuación (6) la podemos acotar superiormente, primero porque sabemos que por hipótesis z_i es un dato mal clasificado, por lo tanto $(\beta^t)^T z_i \leq 0$, ahora en la descripción del problema nos dice que se pueden reescalar las observaciones de tal forma que $||z_i|| < 1 \rightarrow ||z_i||^2 < 1$ y por último usamos la cota descrita en la inecuación (7) para el último término de la inecuación 6.

$$\begin{aligned}
0 \leq ||\beta^t - \beta_{opt}||^2 + 2(\beta^t)^T z_i + ||z_i||^2 - 2z_i^T\beta_{opt} &\leq ||\beta^t - \beta_{opt}||^2 + 0 + 1 - 2 \\
0 \leq ||\beta^{t+1} - \beta_{opt}||^2 &\leq ||\beta^t - \beta_{opt}||^2 - 1
\end{aligned} \tag{10}$$

Llegando a la desigualdad que se quería demostrar.

(c) Explica que lo anterior significa que en un tiempo finito β^t debe converger.

Solución : Primeramente podemos interpretar las normas de la sucesión anterior como la distancia del Beta en la iteración t o $t+1$ hacia el Beta óptimo, asumiento datos linealmente separables, claro.
Lo que se puede apreciar a primera vista de la sucesión anterior es que es una sucesión decreciente, acotada y monótona, lo cual nos asegura que tiene una convergencia en un tiempo finito aunque para motivar de una mejor manera esto que se acabó de decir, podemos imaginar que en el peor de los casos la desigualdad anterior sugiere que la norma del nuevo Beta es simplemente la norma del Beta anterior menos uno, esto pasa en el caso donde tenemos la igualdad, es decir:

$$0 \leq ||\beta^{t+1} - \beta_{opt}||^2 = ||\beta^t - \beta_{opt}||^2 - 1$$

.

Esto sugiere que el nuevo Beta sólamente avanzó una unidad al Beta óptimo en comparación que el Beta anterior.
Ahora, sea un β_0 el primer Beta que calculó el algoritmo, se sabe que si hay separabilidad linear en los datos, entonces debe existir un valor m tal que:

$$||\beta_0 - \beta_{opt}||^2 = m$$

el cual es un valor finito. Y ahora como se mencionaba anteriormente, si tuvieramos el peor de los casos en cada iteración del algoritmo tendríamos una convergencia en a lo más m iteraciones, lo cual demuestra que el perceptrón converge en tiempo finito cuando hay separabilidad linear en los datos.

2. Problema 2

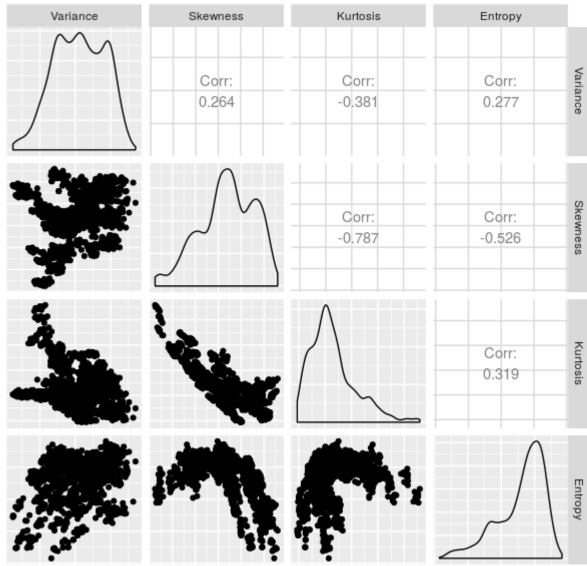
Este ejercicio es sobre el uso de métodos de clasificación para detectar billetes falsos:
En el paper que se anexa a la tarea se resume cada billete con 4 características (varianza, skewness, curtosis y entropía) extraidas de la forma del histograma de los coeficientes de la transformación de Wavelet.

Se anexa el conjunto de datos. La última columna indica si el billete es falso o no (sin hacer distincción entre falso de alta o baja calidad). Resume, visualiza y analiza los datos. Construye algunos clasificadores interesantes basado en k-NN y redes neuronales.
Estima su poder predictivo (divide muchas veces los datos en conjunto de prueba y de entrenamiento).

Solución : Lo primero que se hizo fue leer los datos en R usando la función read.table, esto devolvió un dataframe de los mismos datos el cual posteriormente se segmentaría como datos de prueba y datos de entrenamiento.
Posteriormente se hizo un pequeño análisis de los datos, así como una visualización, los cuales se muestran a continuación.

```
> summary(data)
    Variance          Skewness          Kurtosis          Entropy            Label
 Min.   :-7.0421   Min.   :-13.773   Min.   :-5.2861   Min.   :-8.5482   Min.   :0.0000
 1st Qu.:-1.7730   1st Qu.: -1.708   1st Qu.:-1.5750   1st Qu.:-2.4135   1st Qu.:0.0000
 Median : 0.4962   Median :  2.320   Median : 0.6166   Median :-0.5867   Median :0.0000
 Mean   : 0.4337   Mean   :  1.922   Mean   : 1.3976   Mean   :-1.1917   Mean   :0.4446
 3rd Qu.: 2.8215   3rd Qu.:  6.815   3rd Qu.: 3.1793   3rd Qu.: 0.3948   3rd Qu.:1.0000
 Max.   : 6.8248   Max.   : 12.952   Max.   :17.9274   Max.   : 2.4495   Max.   :1.0000
```

(a) Figura 1. Resumen de los datos.

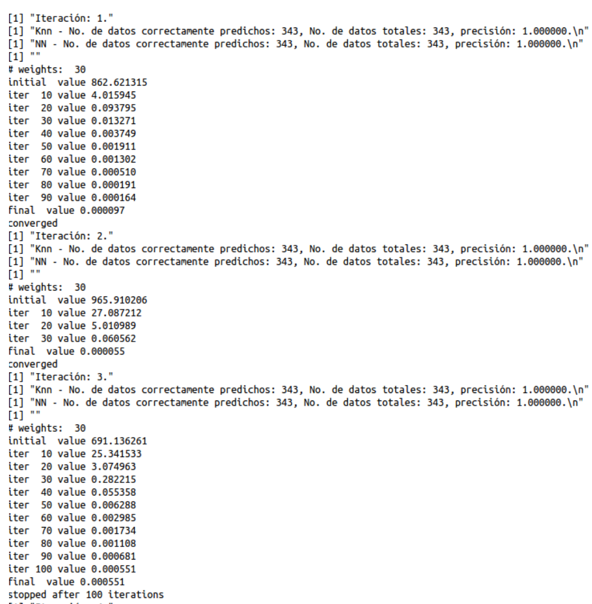


(b) Figura 2. Pairs plot de los datos.

Lo primero que notamos en la Figura 1 es que en algunas columnas de los datos hay valores que no corresponden con el tipo de dato, como por ejemplo, la varianza y la entropía continene valores negativos, esto lo notamos fácilmente En la fila con el nombre *Min* la cual corresponde al dato más pequeño de dicha categoría. Más adelante se verá que esto no afecta en el proceso de clasificación. En la Figura 2 se presenta un pairs plot de los datos quitando la columna de label. En dicha Figura podemos apreciar que existen niveles de correlación considerables en los datos, esto muestra que hay una cierta dependencia de los mismos.

Para la clasificación, la mecánica fue repetir 30 veces el mismo proceso, el cual consistía en permutar todos los datos de forma aleatoria y posteriormente separarlos en dos conjuntos, uno de prueba y uno de entrenamiento, esta separación también se hace de manera aleatoria con la función *sample*, se decidió que el 75 % de los datos serían de prueba y el otro 25 % de entrenamiento.
Posteriormente se usaron los métodos K-NN y un perceptrón simple (ambos ya contenidos en R) para poder hacer la predicción de la calidad de los billetes en base a las 4 características antes mencionadas.
Para cada una de las 30 iteraciones se contabilizó el número de datos predichos correctamente y se obtuvo la métrica de presición de dicha iteración la cual se define como $prec = \frac{TP}{TP+FP}$ donde TP son los verdaderos positivos y FP son los falsos positivos, la cual serviría para hacer un promedio de las 30 iteraciones.
En general se obtuvieron muy buenos resultados por parte de ambos clasificadores. Se pudo lograr a obtener un ratio de éxito de 99 % en ambos clasificadores al terminar las 30 ejecuciones. En el caso de la red neuronal para obtener dicho

porcentaje de éxito se tuvieron que usar 4 neuronas en la capa oculta. A continuación se muestran dos capturas de pantalla tomadas a R studio y con los datos obtenidos de las 30 ejecuciones.



```
[1] "Iteración: 1."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 862.621315
iter  10 value 4.015945
iter  20 value 0.093795
iter  30 value 0.013271
iter  40 value 0.003749
iter  50 value 0.001911
iter  60 value 0.001302
iter  70 value 0.000510
iter  80 value 0.000191
iter  90 value 0.000164
final  value 0.000097
converged
[1] "Iteración: 2."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 965.910206
iter  10 value 27.087212
iter  20 value 5.010989
iter  30 value 0.060562
final  value 0.000055
converged
[1] "Iteración: 3."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 691.136261
iter  10 value 25.341533
iter  20 value 3.074963
iter  30 value 0.282215
iter  40 value 0.055358
iter  50 value 0.006288
iter  60 value 0.002985
iter  70 value 0.001734
iter  80 value 0.001108
iter  90 value 0.000681
iter 100 value 0.000551
final  value 0.000551
stopped after 100 iterations
```

(c) Figura 3. Métodos K-NN y NN clasificando los datos.

```
# weights:  30
initial  value 677.081845
iter  10 value 13.146039
iter  20 value 0.832223
iter  30 value 0.099050
iter  40 value 0.019139
iter  50 value 0.000175
final  value 0.000084
converged
[1] "Iteración: 27."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 714.310909
iter  10 value 20.421598
iter  20 value 0.247962
iter  30 value 0.002186
final  value 0.000044
converged
[1] "Iteración: 28."
[1] "Knn - No. de datos correctamente predichos: 342, No. de datos totales: 343, precisión: 0.997085.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 698.054857
iter  10 value 31.659150
iter  20 value 0.913008
iter  30 value 0.095527
iter  40 value 0.000227
final  value 0.000057
converged
[1] "Iteración: 29."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
# weights:  30
initial  value 744.346428
iter  10 value 37.980682
iter  20 value 0.964803
iter  30 value 0.018925
iter  40 value 0.000486
final  value 0.000060
converged
[1] "Iteración: 30."
[1] "Knn - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] "NN - No. de datos correctamente predichos: 343, No. de datos totales: 343, precisión: 1.000000.\n"
[1] ""
>
> avg_acc_knn <- avg_acc_knn / no_tests
> avg_acc_nn <- avg_acc_nn / no_tests
> print(sprintf("Tasa de éxito promedio del Knn: %f", avg_acc_knn))
[1] "Tasa de éxito promedio del Knn: 0.999223"
> print(sprintf("Tasa de éxito promedio de la red neuronal: %f", avg_acc_nn))
[1] "Tasa de éxito promedio de la red neuronal: 0.999320"
>
```

(d) Figura 4. Resultado de los métodos K-NN y NN.

Como se pueden apreciar en las imágenes anteriores los algoritmos mostraban los resultados de la clasificación, en donde previamente se pasó por el proceso de entrenamiento con el 75 % de los datos, y posteriormente se evaluaron el otro 25 % datos (343 datos) lo cual se imprimió como resultado de esa ejecución.
Finalmente y para las 30 ejecuciones se muestra la tasa de éxito obtenida por cada clasificador, vemos que ambos logran el 99 % de éxito con una minúscula diferencia a favor por parte del K-NN.
A continuación se muestra el código empleado en el presente ejercicio, de la misma manera este se adjuntará en los archivos que se enviarán.

```
setwd("Directorio")
library("class")
library("nnet")

data <- as.data.frame(read.table("data.txt", sep = ",", header = FALSE))
colnames(data) <- c("Variance", "Skewness", "Kurtosis", "Entropy", "Label")

no_tests <- 30
train_data_per <- 0.75
avg_acc_knn <- 0.0
avg_acc_nn <- 0.0
```

```r
for(t in 1:no_tests) {
  shuffled_data <- data[sample(nrow(data)),]
  smp_size <- floor(train_data_per * nrow(data))
  train_ind <- sample(seq_len(nrow(data)), size = smp_size)

  train <- shuffled_data[train_ind, ]
  test <- shuffled_data[-train_ind, ]

  train_set <- train[,1:4]
  train_labels <- train[,5]
  train_labels_nn <- class.ind(train_labels)

  test_set <- test[, 1:4]
  test_labels <- test[,5]

  #Hacer un K-NN
  knn_pred <- knn(train = train_set, test = test_set, cl = train_labels, k = 2)
  #Hacer una NN
  n_net <- nnet(train_labels_nn~Variance+Skewness+Kurtosis+Entropy,
  size = 3, softmax = T, data = train_set)
  nn_pred <- predict(n_net, test_set)

  same_vals_knn <- 0
  same_vals_nn <- 0
  for(i in 1 : length(test_labels)) {
    nn_res_i <- 0
    if(nn_pred[i, 1] < nn_pred[i, 2]) {
      nn_res_i <- 1
    }
    if(test_labels[i] == nn_res_i) {
      same_vals_nn <- same_vals_nn + 1
    }
    if(test_labels[i] == knn_pred[i]) {
      same_vals_knn <- same_vals_knn + 1
    }
  }
  prec_knn <- same_vals_knn / (same_vals_knn + length(test_labels) -
  same_vals_knn)
  prec_nn <- same_vals_nn / (same_vals_nn + length(test_labels) -
  same_vals_nn)
  avg_acc_knn <- avg_acc_knn + prec_knn
  avg_acc_nn <- avg_acc_nn + prec_nn

  print(sprintf("Iteracion: %d.", t))
```

```
  print(sprintf("Knn - No. de datos correctamente predichos: %d,
  No. de datos totales: %d.\n", same_vals_knn, length(test_labels)))
  print(sprintf("NN - No. de datos correctamente predichos: %d,
  No. de datos totales: %d.\n", same_vals_nn, length(test_labels)))
  print("")
}

avg_acc_knn <- avg_acc_knn / no_tests
avg_acc_nn <- avg_acc_nn / no_tests
print(sprintf("Tasa de exito promedio del Knn: %f", avg_acc_knn))
print(sprintf("Tasa de exito promedio de la red neuronal: %f", avg_acc_nn))
```

3. Problema 3

Considera la siguiente función que surge de una red de base radial:

$$f_{\sigma,\beta,\mu}(in) = \sum_{j=1}^{p} \beta_j \exp(-(in - \mu_j)^2/\sigma_j) \tag{11}$$

donde $\sigma = (\sigma_1, ..., \sigma_p)$, $\beta = (\beta_1, ..., \beta_p)$, $\mu = (\mu_1, ..., \mu_p)$. Para un conjunto de datos $\{(in^d, out^d)\}$, define la función de costo:

$$E(\sigma, \beta, \mu) = \sum_d (out^d - f_{\sigma,\beta,\mu}(in^d))^2. \tag{12}$$

(a) Calcula el gradiente de $E()$ con respeto a todos los parámetros (puedes re-parametrizar para facilitar los cálculos).

Solución : Primero derivamos la función de costo con respecto a β_i.

$$\frac{\partial E}{\partial \beta_i} = -2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d)) \exp(\frac{-(in^d - \mu_i)^2}{\sigma_i})] \tag{13}$$

Por lo tanto el gradiente de la función con respecto a β sería:

$$\nabla E_\beta(\sigma, \beta, \mu) = \begin{pmatrix} -2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d)) \exp(\frac{-(in^d-\mu_1)^2}{\sigma_1})] \\ -2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d)) \exp(\frac{-(in^d-\mu_2)^2}{\sigma_2})] \\ \vdots \\ -2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d)) \exp(\frac{-(in^d-\mu_n)^2}{\sigma_n})] \end{pmatrix} \tag{14}$$

Ahora sacamos la derivada parcial con respecto a σ_i.

$$\frac{\partial E}{\partial \sigma_i} = 2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))(-\beta_i)\exp(\frac{-(in^d-\mu_i)^2}{\sigma_i}) * (in^d-\mu_i)^2\sigma^{-1}]$$
$$\frac{\partial E}{\partial \sigma_i} = \frac{-2\beta_i}{\sigma_i^2}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_i)^2}{\sigma_i}) * (in^d-\mu_i)^2] \tag{15}$$

Por lo tanto el gradiente de la función con respecto a σ sería:

$$\nabla_\sigma E(\sigma,\beta,\mu) = \begin{pmatrix} \frac{-2\beta_1}{\sigma_1^2}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_1)^2}{\sigma_1}) * (in^d-\mu_1)^2] \\ \frac{-2\beta_2}{\sigma_2^2}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_2)^2}{\sigma_2}) * (in^d-\mu_2)^2] \\ \vdots \\ \frac{-2\beta_n}{\sigma_n^2}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_n)^2}{\sigma_n}) * (in^d-\mu_n)^2] \end{pmatrix} \tag{16}$$

Ahora sacamos la derivada parcial con respecto a μ_i.

$$\frac{\partial E}{\partial \mu_i} = 2\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))(-\beta_i)\exp(\frac{-(in^d-\mu_i)^2}{\sigma_i}) * \frac{2}{\sigma_i}(in^d-\mu_i)]$$
$$\frac{\partial E}{\partial \mu_i} = \frac{-4\beta_i}{\sigma_i}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_i)^2}{\sigma_i}) * (in^d-\mu_i)] \tag{17}$$

Por lo tanto el gradiente de la función con respecto a μ sería:

$$\nabla_\mu E(\sigma,\beta,\mu) = \begin{pmatrix} \frac{-4\beta_1}{\sigma_1}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_i)^2}{\sigma_1}) * (in^d-\mu_1)] \\ \frac{-4\beta_2}{\sigma_2}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_2)^2}{\sigma_2}) * (in^d-\mu_2)] \\ \vdots \\ \frac{-4\beta_n}{\sigma_n}\sum_d [(out^d - f_{\sigma,\beta,\mu}(in^d))\exp(\frac{-(in^d-\mu_n)^2}{\sigma_n}) * (in^d-\mu_n)] \end{pmatrix} \tag{18}$$

(b) Implementa un algoritmo que ajusta $f_{\sigma,\beta,\mu}(\Delta)$ a un conjunto de datos $\{(in^d, out^d)\}$ dados, usando descenso de gradiente para encontrar los parámetros óptimos. Usalo para una aplicación que eliges.

Una variante consiste en elegir / estimar primero (σ,μ) y después minimizar E sobre β, fijando $(\hat{\sigma},\hat{\mu})$. La estimación de (σ,μ) se puede hacer a través de algún método de clústering como k-medias y tomando como μ los centroides correspondientes. Compara tus resultados anteriores con esta versión.

Solución : Para este ejercicio se decidió ajustar una mezcla de gaussianas para un problema de clasificación sencillo, el cual consistió en generar dos conjuntos de datos, ambos normales pero con medias distintas y optimizar la función de costo anteriormente descrita para realizar la predicción a cuál clase pertenecía cada dato.

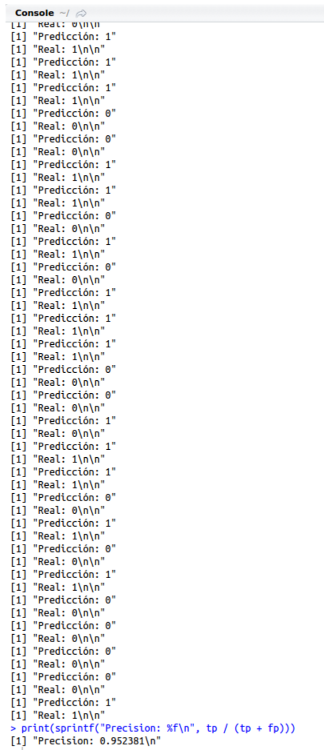
El proceso consistió en la generación de los datos, en los cuales se generaron 2 conjuntos, uno de entrenamiento y otro de prueba, a su vez el conjunto de entrenamiento se dividió en dos subconjuntos, los cuales consistían en los datos asociados a cada una de las clases. Esto anterior porque la idea es buscar dos conjuntos de parámetros (μ, $sigma$, β) que pudieran clasificar a los datos en sus respectivas clases.
Posteriormente se continuó con el proceso de optimización, en el cual se implementó el método de descenso por gradiente, y para este método se usaron los gradientes descritos en el punto anterior. La idea con este método para esta primer parte del ejercicio fue ir optimizando cada parámetro por separado, fijando los otros dos. Esto se realizó iterativamente hasta que la norma de los gradientes de los tres parámetros fuera menor a cierta tolerancia. Como comentarios secundarios de este método, se utilizó un tamaño de paso fijo en cada iteración, además se pusieron como criterio de paro el número de iteraciones y una tolerancia para la norma del gradiente la cual fue de 0.001.
Una vez que obtuvieron los parámetros óptimos con el método del gradiente se procedió a realizar el proceso de clasificación y para ello se utilizó un conjunto de prueba, el cual consistía de datos de ambas clases permutados aleatoriamente. Para el proceso de clasificación se utilizaron las siguientes funciones para determinar a qué clase pertenecía cada elemento del conjunto de prueba:

$$\begin{aligned} f(c;\beta^j,\sigma^j,\mu^j) &= \sum_{i=1}^{n} \beta_i^j exp(\frac{-(c-\mu_i^j)}{\sigma_i^j}) \\ F(c;\beta^1,\sigma^1,\mu^1) &= \frac{f(c;\beta^1,\sigma^1,\mu^1)+\epsilon}{f(c;\beta^1,\sigma^1,\mu^1)+f(c;\beta^2,\sigma^2,\mu^2)+2\epsilon} \\ F(c;\beta^2,\sigma^2,\mu^2) &= \frac{f(c;\beta^2,\sigma^2,\mu^2)+\epsilon}{f(c;\beta^1,\sigma^1,\mu^1)+f(c;\beta^2,\sigma^2,\mu^2)+2\epsilon} \end{aligned} \tag{19}$$

Donde $\epsilon = 0{,}01$. Si $F(c;\beta^1,\sigma^1,\mu^1) < F(c;\beta^2,\sigma^2,\mu^2)$ entonces para el dato c se asigna a la clase 0, en caso contrario se asigna a la clase 1.
Para cuantificar el poder de clasificación del método se utilizó la métrica de presición misma que se usó en el ejercicio anterior. A continuación se muestran los resultados obtenidos al ejecutar el algoritmo.

```
Console ~/
[1] "Iteración: 426\n"
[1] "Norma del gradiente: 0.012804\n"
[1] "Iteración: 427\n"
[1] "Norma del gradiente: 0.012671\n"
[1] "Iteración: 428\n"
[1] "Norma del gradiente: 0.012539\n"
[1] "Iteración: 429\n"
[1] "Norma del gradiente: 0.012409\n"
[1] "Iteración: 430\n"
[1] "Norma del gradiente: 0.012280\n"
[1] "Iteración: 431\n"
[1] "Norma del gradiente: 0.012152\n"
[1] "Iteración: 432\n"
[1] "Norma del gradiente: 0.012025\n"
[1] "Iteración: 433\n"
[1] "Norma del gradiente: 0.011900\n"
[1] "Iteración: 434\n"
[1] "Norma del gradiente: 0.011776\n"
[1] "Iteración: 435\n"
[1] "Norma del gradiente: 0.011654\n"
[1] "Iteración: 436\n"
[1] "Norma del gradiente: 0.011532\n"
[1] "Iteración: 437\n"
[1] "Norma del gradiente: 0.011412\n"
[1] "Iteración: 438\n"
[1] "Norma del gradiente: 0.011294\n"
[1] "Iteración: 439\n"
[1] "Norma del gradiente: 0.011176\n"
[1] "Iteración: 440\n"
[1] "Norma del gradiente: 0.011060\n"
[1] "Iteración: 441\n"
[1] "Norma del gradiente: 0.010945\n"
[1] "Iteración: 442\n"
[1] "Norma del gradiente: 0.010831\n"
[1] "Iteración: 443\n"
[1] "Norma del gradiente: 0.010718\n"
[1] "Iteración: 444\n"
[1] "Norma del gradiente: 0.010606\n"
[1] "Iteración: 445\n"
[1] "Norma del gradiente: 0.010496\n"
[1] "Iteración: 446\n"
[1] "Norma del gradiente: 0.010387\n"
[1] "Iteración: 447\n"
[1] "Norma del gradiente: 0.010279\n"
[1] "Iteración: 448\n"
[1] "Norma del gradiente: 0.010172\n"
[1] "Iteración: 449\n"
[1] "Norma del gradiente: 0.010066\n"
[1] "Iteración: 450\n"
[1] "Norma del gradiente: 0.009961\n"
[1] "Iteración: 1\n"
[1] "Norma del gradiente: 0.000069\n"
[1] "Iteración: 1\n"
[1] "Norma del gradiente: 0.000209\n"
```

(e) Figura 5. Proceso de optimización de los parámetros β, σ, μ.

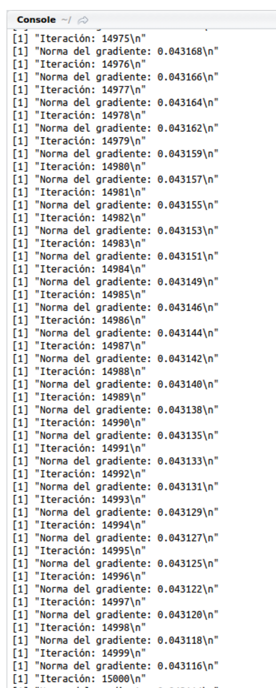


(f) Figura 6. Resultado de la clasificación.

En la Figura 5 podemos ver el proceso de optimización del parámetro β, en dicha imagen se muestra el resultado de las iteraciones en donde se imprime la norma del gradiente de dicho parámetro, podemos ver que este va decreciendo poco a poco en el conjunto de iteraciones.
En la Figura 6 se muestra el proceso de validación del método en donde se utilizó un conjunto de 100 elementos de prueba mezclados para las dos clases como se mencionó anteriormente, en dicha imagen podemos ver algunos de los resultados de la clasificación realizada en donde se muestra el valor de la clase predicha por el método y el valor real. Finalmente vemos que se logró un 95 % en la métrica de presición, lo cual es buen en general. Este resultado se obtuvo con una mezcla de 7 gaussianas.

Para la segunda parte del ejercicio se usó la modificación sugerida en la descripción de la tarea, la cual consiste en hacer una estimación inicial de dos parámetros, los cuales son (σ, μ) y sólamente optimizar β. La estimación del parámetro μ se hace a través del algoritmo de K medias en donde buscamos tantos centroides como gaussianas tenga el modelo. Y para el parámetro σ este simplemente se estimó aleatoriamente y se dejó fijo, aquí también se

pudo haber tenido la alternativa de escoger arbitrariamente un parámetro para σ pero se decidió calcularlo aleatorio para ambos conjuntos.
A continuación se muestran los resultados obtenidos para esta variante del algoritmo.



```
Console ~/
[1] "Iteración: 14975\n"
[1] "Norma del gradiente: 0.043168\n"
[1] "Iteración: 14976\n"
[1] "Norma del gradiente: 0.043166\n"
[1] "Iteración: 14977\n"
[1] "Norma del gradiente: 0.043164\n"
[1] "Iteración: 14978\n"
[1] "Norma del gradiente: 0.043162\n"
[1] "Iteración: 14979\n"
[1] "Norma del gradiente: 0.043159\n"
[1] "Iteración: 14980\n"
[1] "Norma del gradiente: 0.043157\n"
[1] "Iteración: 14981\n"
[1] "Norma del gradiente: 0.043155\n"
[1] "Iteración: 14982\n"
[1] "Norma del gradiente: 0.043153\n"
[1] "Iteración: 14983\n"
[1] "Norma del gradiente: 0.043151\n"
[1] "Iteración: 14984\n"
[1] "Norma del gradiente: 0.043149\n"
[1] "Iteración: 14985\n"
[1] "Norma del gradiente: 0.043146\n"
[1] "Iteración: 14986\n"
[1] "Norma del gradiente: 0.043144\n"
[1] "Iteración: 14987\n"
[1] "Norma del gradiente: 0.043142\n"
[1] "Iteración: 14988\n"
[1] "Norma del gradiente: 0.043140\n"
[1] "Iteración: 14989\n"
[1] "Norma del gradiente: 0.043138\n"
[1] "Iteración: 14990\n"
[1] "Norma del gradiente: 0.043135\n"
[1] "Iteración: 14991\n"
[1] "Norma del gradiente: 0.043133\n"
[1] "Iteración: 14992\n"
[1] "Norma del gradiente: 0.043131\n"
[1] "Iteración: 14993\n"
[1] "Norma del gradiente: 0.043129\n"
[1] "Iteración: 14994\n"
[1] "Norma del gradiente: 0.043127\n"
[1] "Iteración: 14995\n"
[1] "Norma del gradiente: 0.043125\n"
[1] "Iteración: 14996\n"
[1] "Norma del gradiente: 0.043122\n"
[1] "Iteración: 14997\n"
[1] "Norma del gradiente: 0.043120\n"
[1] "Iteración: 14998\n"
[1] "Norma del gradiente: 0.043118\n"
[1] "Iteración: 14999\n"
[1] "Norma del gradiente: 0.043116\n"
[1] "Iteración: 15000\n"
[1] "Norma del gradiente: 0.043114\n"
```

(g) Figura 7. Proceso de optimización del parámetro β.

```
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 0\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 0\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
[1] "Predicción: 0"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 0\n\n"
[1] "Predicción: 1"
[1] "Real: 1\n\n"
> print(sprintf("Precision: %f\n", tp / (tp + fp)))
[1] "Precision: 0.769231\n"
```

(h) Figura 8. Resultado de la clasificación.

En la Figura 7 al igual que la 5 se muestra un poco sobre la parte de la optimización donde en este caso el único parámetro a optimizar es β, se puede apreciar que la norma del gradiente va descendiendo de igual manera que en la versión anterior del algoritmo. Como comentario, esta vez se tuvo que utilizar un tamaño de paso un poco más pequeño ya de de otra manera no minimizaba.
En la Figura 8 se muestra el resultado de la clasificación, en donde se aprecia para algunas iteraciones el valor de las clase predicha por el método y el valor real, además de el valor del parámetro precisión, aquí se puede apreciar que en esta variante no se obtuvo resultados tan buenos como en la anterior, ya que la precisión sse obtuvo de 79 %.

El código de estos dos ejercicios no se anexará en el reporte ya que resultó ser un poco largo, pero de igual manera se mandará en los archivos de la tarea.