Tarea V

Para entregar el 25 de abril antes de la clase

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

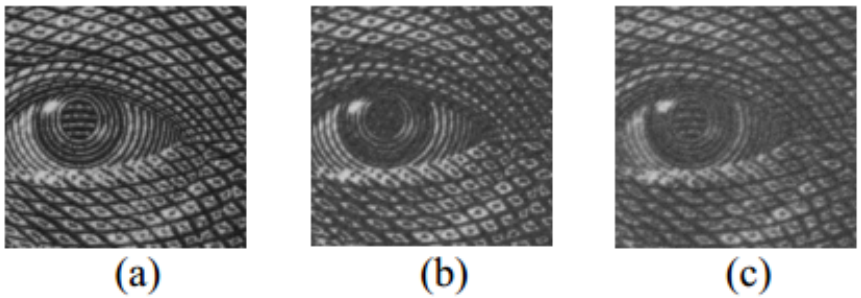
   $$\beta_{opt}^t z_i \geq 1.$$

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

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

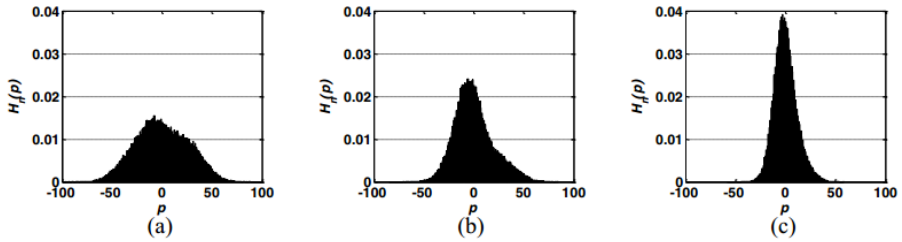
   (c) Explica que lo anterior significa que en un tiempo finito β^t debe converger.

2. Este ejercicio es sobre el uso de métodos de clasificación para detectar billetes falsos:



(a) (parte de) un billete de verdad; (b) billete falso (de alta calidad); (c) billete falso (de baja calidad)

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. Los histogramas a continuación muestran como cambia la forma cuando el billete ya no es auténtico.



(a) histograma de los coeficientes de un billete de verdad; (b) billete falso (de alta calidad); (c) billete falso (de baja calidad)

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)

Info complementario:
Como ilustración de como usar `nnet()` para clasificación:

```
#generar los dos grupos
library(nnet)
x1<-rnorm(100,0,1)
y1<-rnorm(100,0,1)
x2<-rnorm(100,4,1)
y2<-rnorm(100,4,1)
clase<-c(rep(0,100),rep(1,100))
d<-data.frame(c(x1,x2),c(y1,y2))
names(d)<-c("X","Y")

#generar las etiquetas de la clase como vectores binarios
clase<-class.ind(clase)

#ajustar una red (2-1-2) con una capa oculta; la opcion {\tt softmax} indica
#que tenemos un problema de clasificacion y no de regresion
n<-nnet(clase~X+Y,size=1,softmax=T,data=d)

#para calcular las predicciones de esta red con una conjunta (nueva):
predict(n,d) #  d debe ser un dataframe donde "X" y "Y" son definidas !!!
```

Para entender mejor la forma exacta de la red como R la maneja; la funcion `predict` para el caso anterior es algo de la forma (los valores de los pesos fueron obtenidos con `summary(n)` o `n$wts` y para cada corrida cambian en general bastante):

```
logistic<-function(x) {return(1/(1+exp(-x)))}

predecir<-function(x)
{x1<-x[1];x2<-x[2];
 h1<- 10.835625-x1*3.172014 -x2*2.172675
 o1<- -7.185283 + 15.584699*logistic(h1)
 o2<-  7.465928  -16.707176*logistic(h1)
 return(c(exp(o1)/(exp(o1)+exp(o2)),exp(o2)/(exp(o1)+exp(o2)) ))}
```

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

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

$$\sigma = (\sigma_1, \cdots, \sigma_p), \beta = (\beta_1, \cdots, \beta_p), \mu = (\mu_1, \cdots, \mu_p).$$

Para un conjunto de datos $\{(in^d, out^d)\}$, define la función de costo:

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

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

(b) Implementa un algoritmo que ajusta $f_{\sigma,\beta,\mu}(\cdot)$ 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.