Reporte práctica 12
Red neuronal

1. INTRODUCCIÓN

En la práctica se demuestra el aprendizaje a máquina: se reconocen dígitos de imágenes pequeñas en blanco y negro con una red neuronal. El elemento básico de una red neuronal es un perceptrón que esencialmente es un híperplano (una línea si nos limitamos a dos dimensiones) que busca colocarse en la frontera que separa las entradas verdaderas y las entradas falsas. La dimensión *d* del perceptrón es el largo del vector *x* (*pixeles*) que toma como entrada, y su estado interno se representa con otro vector *w* que contiene sus pesos. Si el producto de w · x es un número positivo la respuesta del perceptrón será *TRUE* de lo contrario será *FALSE*. [1]

Cada error causa un movimiento del perceptrón, y va disminuyendo la tasa de aprendizaje.

2. SIMULACIÓN

Para iniciar la simulación, se crea una plantilla que contiene los dígitos en código de colores, y se asigna una probabilidad para cada color. Se inicia con una *tasa* de aprendizaje que irá disminuyendo en cada paso al ser multiplicada por el factor *tranqui*.

En la fase de entrenamiento se obtiene una matriz al azar mediante un *runif* que "dibuja" un número y cada neurona lanza un resultado, el cual, es comparado (mediante un *if*) con el número que se le dio al perceptrón. Si el resultado es incorrecto se hará un ajuste y disminuirá la tasa de aprendizaje.

Después entra la fase de prueba, en donde se repite el procedimiento de la fase de entrenamiento exceptuando el ajuste y cada error es contabilizado en *contadores*, en dónde se guarda cuantas veces se equivocó el resultado y con cuál número fue confundido.

En la tarea se buscó paralelizar el cálculo donde se creyó conveniente; se graficaron los tiempos de ejecución, así como los porcentajes de error de ambos códigos.

Se decidió hacer en paralelo la fase de prueba del código; mediante el uso del *foreach*, así como se muestra a continuación:

```
f2<-function () {
  d <- sample (0: tope, 1)
  pixeles <- runif(dim) < modelos [d + 1,]
  correcto <- binario (d, n)
```

```
  salida <- rep (FALSE, n)
  for (i in 1: n) {
    w <- neuronas[i,]
    deseada <- correcto[i]
    resultado <- sum (w * pixeles) >= 0
    salida[i] <- resultado
  }
  r <- min (decimal (salida, n), k) # todos los no-existentes van al
final
  if(r==d) {
    return(TRUE)
  }
}

cuenta<-foreach (t=1:pr, .combine=c) %dopar% f2()
bueno=sum(cuenta)
porcentaje=round((bueno/pr) *100,2)
```

Se guardó el porcentaje de aciertos y se eliminó la sección de contadores.

Para determinar la diferencia en ejecución de cada código (original y paralelo) se utilizó la herramienta *source* y con ayuda de un *for* se determinaron cinco corridas y utilizando otro *for* se varió la cantidad *pruebas* (600, 800, 1000 y 1200).

```
suppressMessages(library(doParallel))
registerDoParallel(makeCluster(detectCores () - 1))
Tiempos<-data.frame()
Tnp<-numeric ()
Tp<-numeric()
For (pr in seq (600, 1200, 200)) {
  For (corrida in 1:5) {

   source('~/GitHub/SimulacionComputacional/P12/originalP12.R')
    Tnp <- cbind ("original", tiempo, pr,porcentaje,  corrida)

  source('~/GitHub/SimulacionComputacional/P12/paraleloP12.R')
    Tp <- cbind ("paralelo",tiempo, pr, porcentaje, corrida)

  Tiempos<-  rbind (Tiempos,Tnp, Tp)
    }
  }
stopImplicitCluster ()
```

Utilizando la herramienta *ggplot2* se graficó la ejecución de los dos códigos y los porcentajes de aciertos:

```
library(ggplot2)
png("p11R1_plot2.png")
ggplot (data=Tiempos, aes (x=Prueba, y=Tiempo, color=Tipo)) +
guides (color=guide_legend (title = NULL)) +
  geom_boxplot () +
  theme (plot.title = element_text(hjust = 0.5))+
  ggtitle ("Tiempos comparados con diferente tamaño de prueba")
graphics.off ()

png("p12R1_por.png")
Tiempos$porcentaje=as.numeric(levels(Tiempos$porcentaje))
[Tiempos$porcentaje]
ggplot (data=Tiempos, aes (x=Prueba, y=porcentaje, color=Tipo)) +
  guides (color=guide_legend (title = NULL)) +
  geom_boxplot () +
  theme (plot.title = element_text(hjust = 0.5))+
  ggtitle ("Porcentaje de error")
graphics.off ()
```

3. RESULTADOS

Al correr los códigos no se observó una diferencia entre el tiempo de ejecución del programa secuencial y el paralelo para 300 pruebas, por lo que se decidió aumentar el tamaño de pruebas, se varió secuencialmente de 600 a 1200 con saltos de 200 (figura 1). Resultando que el tiempo de ejecución del programa paralelizado disminuye a partir de mil pruebas.

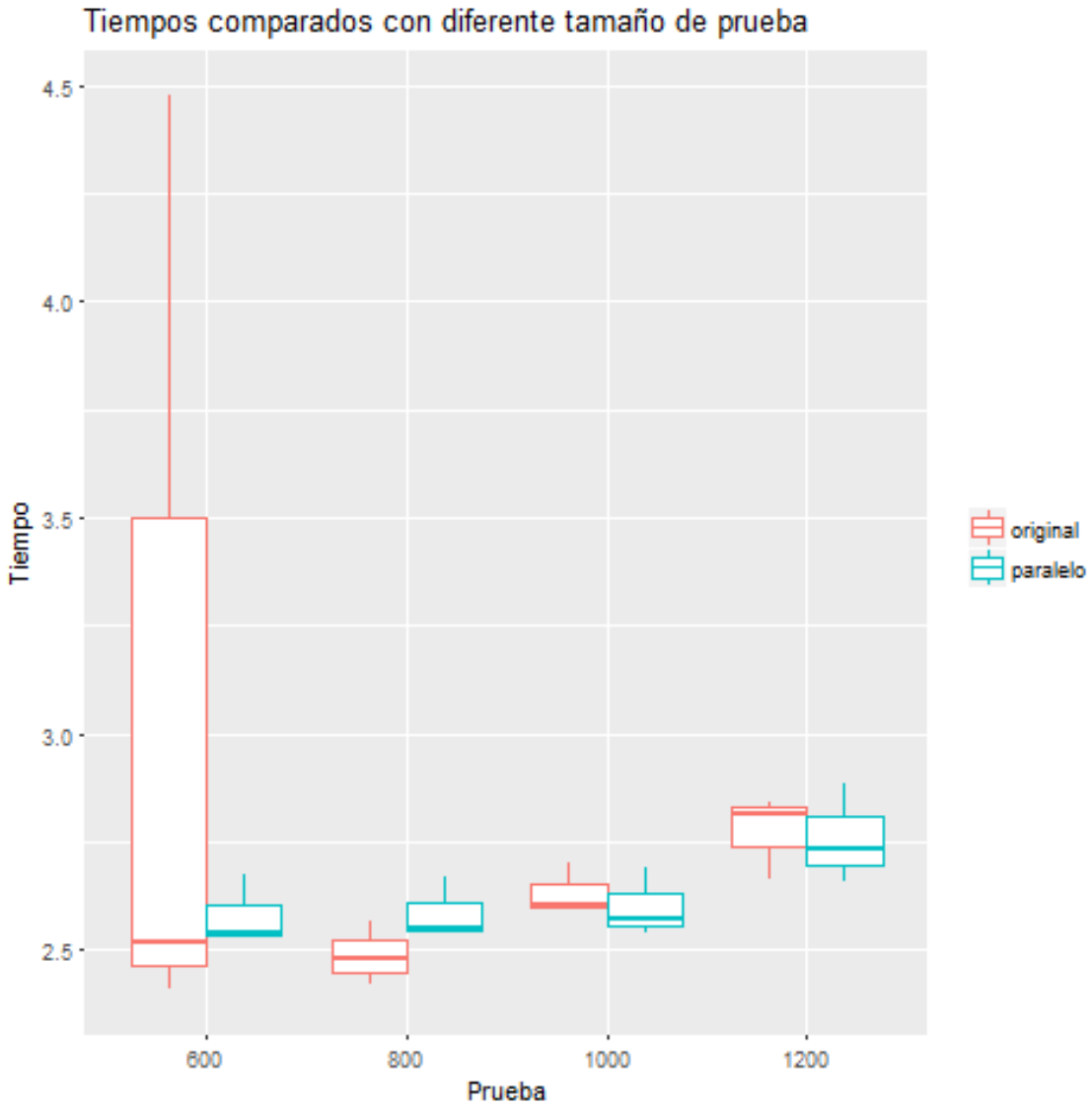

Figura 1. Gráfica caja-bigote de los tiempos de ejecución de los programas original (color salmón) y paralelizado (color celeste) variando tamaño de prueba (eje horizontal).

Se verificó la tasa de aciertos de cada código, para corroborar que al paralelizar no se modificó. No se obtuvo disminución de los porcentajes de aciertos para el paralelo como se muestra en la figura 2 e inclusive se mejora el porcentaje para 1000 y 1200 pruebas en este código, resultando entre 85% y 90%.

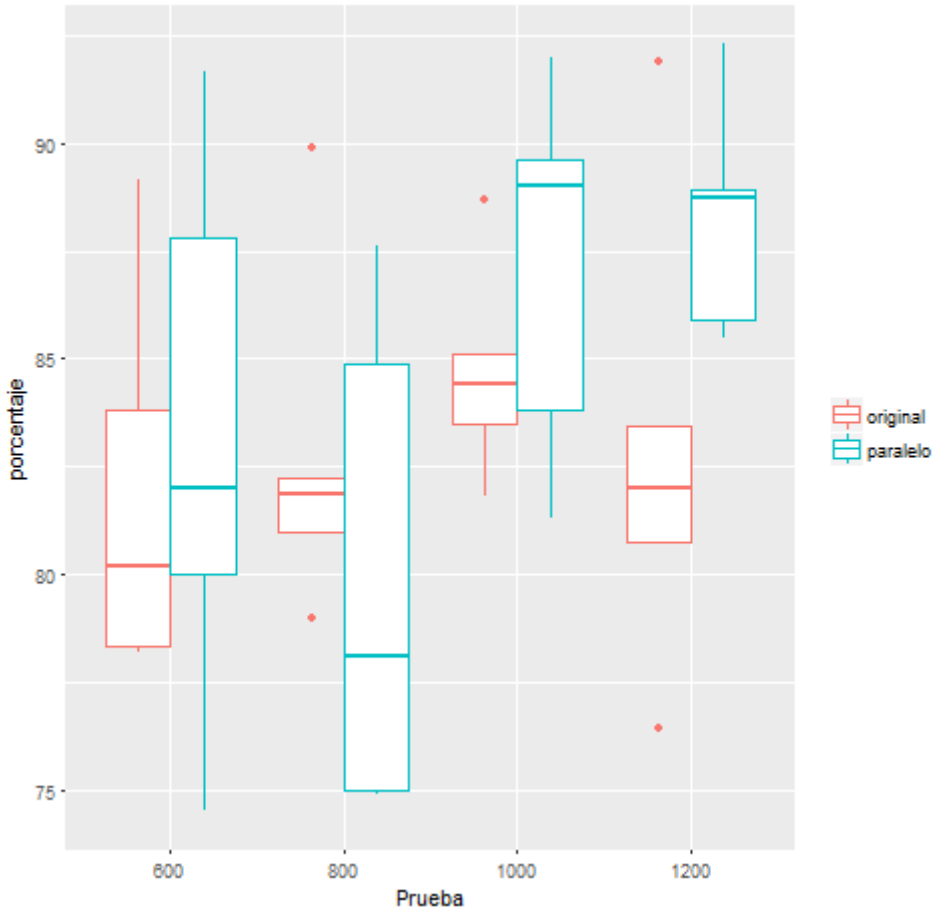

Figura 2. Gráfica caja-bigote de los porcentajes de aciertos para cada programa, original (color salmón) y paralelizado (color celeste), variando tamaño de prueba (eje horizontal).

4. RETOS

- Reto 1

Para el reto uno se realizó un estudio sistemático en función de la variación de la probabilidad de cada color de pixel (*blanco, negro* y *gris*) para la generación de dígitos.

Se utilizó el código ya paralelizado, se asignó un valor en secuencia para cada probabilidad que va de 0.09 a 0.99 con una variación de 0.1.

Se varió mediante tres *for* esta probabilidad en cada color (*negro*, *blanco* y *gris*), así como se muestra en el código:

```
suppressMessages(library(doParallel))
registerDoParallel(makeCluster(detectCores() - 1))
Tiempos<-data.frame()

Tp<-numeric()

npr<-seq(0.09,0.99, 0.1)
gpr<-seq(0.09,0.99, 0.1)
bpr<-seq(0.09,0.99, 0.1)
```

```
for(negro in npr){
  for(gris in gpr) {
    for(blanco in bpr){



    source('~/GitHub/SimulacionComputacional/P12/codigoP12R1.R',
encoding = 'UTF-8')

    Tp <- cbind(blanco, negro, gris, porcentaje)

    Tiempos<-  rbind(Tiempos,Tp)
  }
    }
}
stopImplicitCluster()
```

Se graficaron, mediante la librería *ggplot2*, mapas de calor comparando *blanco* con *negro*, *negro* con *gris* y *gris* con *blanco*.

```
library(ggplot2)
png("p12R1_BN1.png")
ggplot (data=Tiempos, aes (x=negro, y=blanco)) +
  scale_fill_gradient (low="white", high="blue") +
  geom_raster(aes(fill=porcentaje))
  graphics.off ()

  png("p12R1_NG1.png")
  ggplot(data=Tiempos, aes(x=negro, y=gris)) +
    scale_fill_gradient (low="white", high="blue") +
    geom_raster(aes(fill=porcentaje))
  graphics.off ()

  png("p12R1_BG1.png")
  ggplot (data=Tiempos, aes (x=blanco, y=gris)) +
    scale_fill_gradient (low="white", high="blue") +
    geom_raster(aes(fill=porcentaje))
  graphics.off()
```

Al comparar los porcentajes de aciertos de dos colores variando su probabilidad se observa que al disminuir la probabilidad del color negro conviene aumentar la probabilidad del color *blanco* y *gris* (figura 3a y 3b).

En la figura 3a se observa que el porcentaje de aciertos es inversamente proporcional a la probabilidad del color *negro* y directamente proporcional al aumento de la probabilidad del color *blanco*.

Cuando se compara el color *gris* con cualquiera de los otros colores (*blanco* o *negro*) las probabilidades de éstos tienen que ser menores para mejorar los porcentajes de aciertos mientras que la del color *gris* no varía en gran medida sus porcentajes de aciertos con la probabilidad. Como se muestra en la figura 3b y 3c en dónde los porcentajes más altos de aciertos se localizan más cercanos al eje vertical izquierdo.

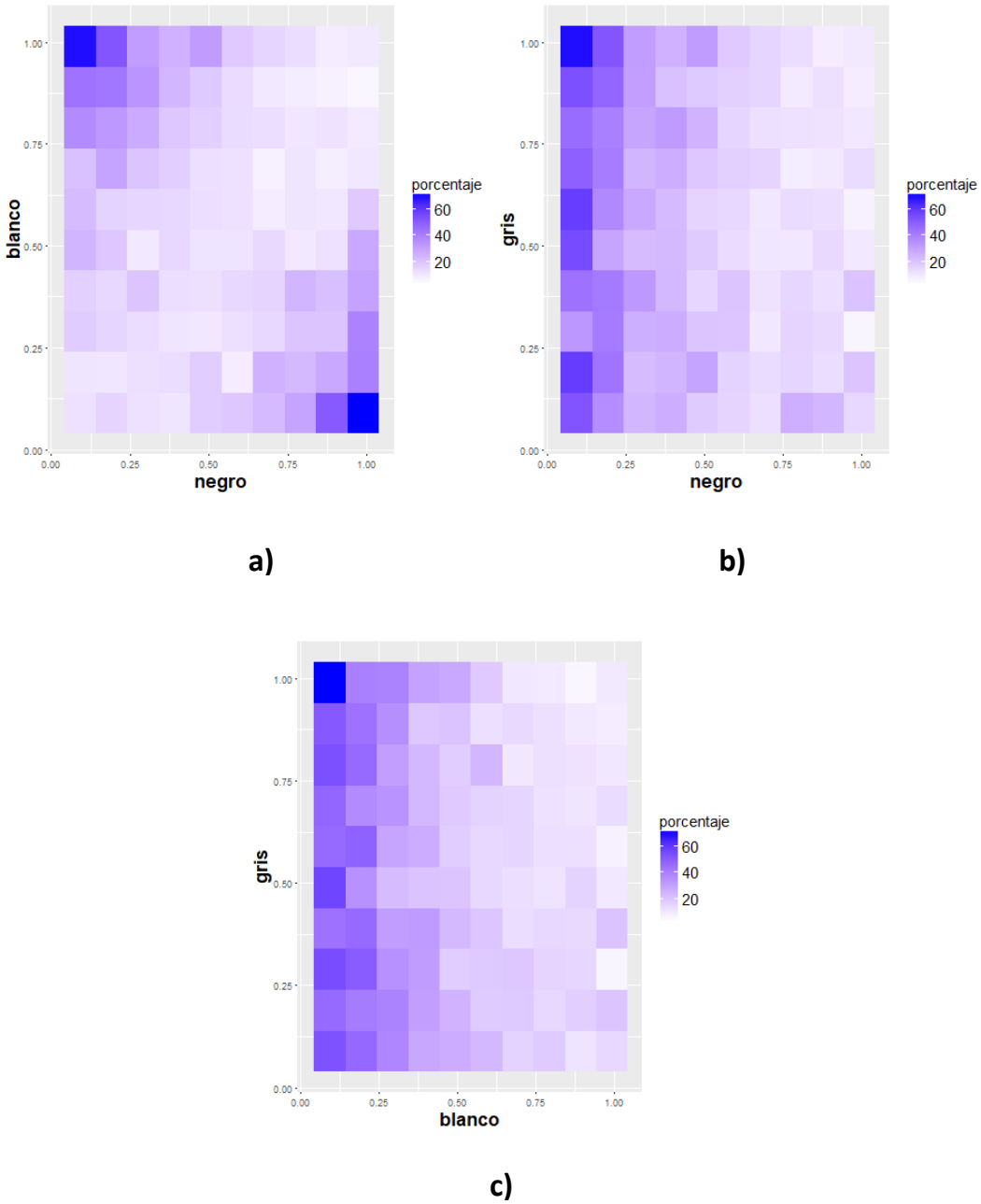

Figura 3. Mapa de calor comparando diferentes probabilidades de dos colores, *blanco* con *negro* (a), *negro* con *gris* (b), y *gris* con *blanco* (c).

5. CONCLUSIONES

Se paralelizó el código dado mediante el uso de la herramienta *doparallel* y el comando *foreach*.

El código se ejecuta en menor tiempo al paralelizar la fase de *prueba* en tamaños mayores o iguales a mil.

Los porcentajes de aciertos entre los códigos son muy similares; incluso, el porcentaje en el código paralelizado aumenta en tamaños de 1000 y 1200.

Se compararon los porcentajes de aciertos en función de las probabilidades de dos colores mediante la herramienta *ggplot2*.

Al aumentar la probabilidad del color negro y disminuir la del color blanco se obtienen porcentajes de aciertos altos, así como al disminuir la probabilidad del color negro y aumentar la del color blanco.

Cuando se compara el color gris con cualquiera de los otros dos colores, la probabilidad de éste tiene mayor efecto en los porcentajes de aciertos, que la probabilidad de los otros.

6. REFERRENCIAS

[1]Elisa.dyndns-web.com. (2017). P12-R paralelo-Schaeffer. [online]Available at: http://elisa.dyndns-web.com/teaching/comp/par/p12.html [Accessed 30 Oct. 2017].