Práctica 12: red neuronal

Ricardo Rosas Macías

21 de mayo de 2019

1. Introducción

La red neuronal es un modelo que funciona de una forma similar y simplificada a la de un cerebro humano, en un ordenador. Este tipo de inteligencia artificial permite obtener una solución con la ayuda de criterios de utilidad, a través de la combinación de parámetros y ponderaciones de entrenamiento de selección multi-objetivo; asistidos de perceptrónes, que favorecen la identificación de la posición, de modo que favorecen discernir la predicción del híperplano.

2. Objetivo

Para llevar acabo el objetivo de la práctica, se hizo un proceso de aprendizaje de patrones, con los cuales la red neuronal fuera capaz de crear ponderaciones sinápticas para la elección correcta del resultado. Asimismo, se paralelizó el experimento para obtener un mejor desempeño en los tiempos de ejecución del código.

2.1. Descripción

La finalidad del experimento es [3]:

> "Paralelizar un segmento del código y estudiar de manera sistemática el desempeño de la red neuronal para los diez dígitos en función de las tres probabilidades asignadas a la generación de los dígitos (ngb), variando a las tres en un experimento factorial adecuado.
>
> Para el primer reto, se extiende y entrena la red neuronal para que reconozca además por al menos doce símbolos ASCII adicionales, aumentando la resolución de las imágenes a 5×7 de lo original de 3×5 (modificando las plantillas de los dígitos acorde a este cambio)."

3. Resultados y conclusiones

De acuerdo a la literatura anteriormente reportada [1], se hicieron líneas de código que permitieran una ejecución paralela. De igual manera, se cuantificó el tiempo que toma en realizar la ejecución del código y se comparo con la ejecución normal, como se muestra en la parte inferior.

```
repetitions <- 30
trainNP <- microbenchmark(
  for (t in 1:5000) {
    d <- sample(0:tope, 1)
    pixeles <- runif(dim) < modelos[d + 1,]
    correcto <- binario(d, n)
    for (i in 1:n) {
      w <- neuronas[i,]
      deseada <- correcto[i]
      resultado <- sum(w * pixeles) >= 0
      if (deseada != resultado) {
        ajuste <- tasa * (deseada - resultado)
        tasa <- tranqui * tasa
        neuronas[i,] <- w + ajuste * pixeles
      }
    }
  }
  , times = repetitions, unit = "s")

testNP <- microbenchmark({
  contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
  rownames(contadores) <- 0:tope
  colnames(contadores) <- c(0:tope, NA)

  for (t in 1:300) {
    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)
    contadores[d+1, r+1] <- contadores[d+1, r+1] + 1
  }
}
, times = repetitions, unit = "s")

newrons <- function(i, neuronas, pixeles)
newr <- function(){
  d <- sample(0:tope, 1)
  pixeles <- runif(dim) < modelos[d + 1,]

  salida <- sapply(1:n, neuronas[i,])
  r <- min(decimal(salida, n), k)
  return(c(d+1, r+1))
}
suppressMessages(library(doParallel))
clust <- makeCluster(detectCores()-1)
registerDoParallel(clust)

```

```
testP <- microbenchmark({
  contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
  rownames(contadores) <- 0:tope
  colnames(contadores) <- c(0:tope, NA)
  conta <- foreach(t = 1:300, .combine = "rbind", .export = c("tope", "dim", "modelos",
      "neuronas", "pixeles")) %dopar% newr()
  for (i in 1:300){
    contadores[conta[i, 1], conta[i, 2]] <- contadores[conta[i, 1], conta[i, 2]] + 1
  }
}
, times = repetitions, unit = "s")

stopCluster(clust)
trains <- cbind("Normal" = trainNP$time)
tests <- cbind("Paralelizada" = testP$time, "Normal" =testNP$time)
```

Se realizó una visualización gráfica con ayuda de la paquetería *ggplot2*, como se muestra en la figura 1, en la cual se observa claramente un aumento de tiempos al realizar la ejecución normal, de modo que permite definir que la ejecución paralela es mejor para el experimento.

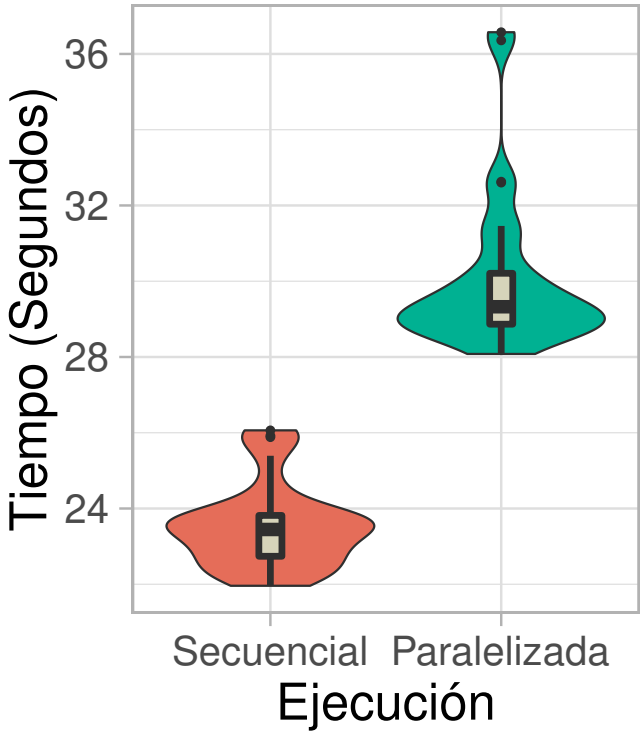

Figura 1: Tiempo de tarda la ejecución del código

Acorde al precedente [1], se hicieron líneas de código que proporcionen el desempeño de la red neuronal para los diez dígitos. Por ende, en el código inferior, se determinó una variación de las probabilidades de color los pixeles; de manera que permita observar el desempeño de la red neuronal a estos cambios.

```
newrons <- function(i, neuronas, pixeles){
  w <- neuronas[i,]
  return(sum(w * pixeles) >= 0)
}
newr <- function(){
  d <- sample(0:tope, 1)
  pixeles <- runif(dim) < modelos[d + 1,]
  correcto <- binario(d, n)
  salida <- sapply(1:n, newrons, neuronas, pixeles)
  r <- min(decimal(salida, n), k)
  return(c(d+1, r+1))
}

Negros <- c(0.90, 0.95, 0.995)
Grises <- c(0.80, 0.90, 0.992)
Blancos <- c(0.002, 0.02, 0.1)

data <- data.frame("Porcen"=numeric(), "N" = numeric(), "G" = numeric(), "B"= numeric())
for (negro in Negros){
  for (gris in Grises) {
    for (blanco in Blancos){

 modelos <- read.csv("digitos.modelo.csv", sep=" ", header=FALSE, stringsAsFactors=F)
      modelos[modelos=='n'] <- negro
      modelos[modelos=='g'] <- gris
      modelos[modelos=='b'] <- blanco

      suppressMessages(library(doParallel))
      clust <- makeCluster(detectCores()-1)
      registerDoParallel(clust)

      contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
      rownames(contadores) <- 0:tope
      colnames(contadores) <- c(0:tope, NA)
      conta <- foreach(t = 1:300, .combine = "rbind", .export = c("tope", "dim", "
          modelos", "neuronas", "pixeles")) %dopar% newr()
      for (i in 1:300){
        contadores[conta[i, 1], conta[i, 2]] <- contadores[conta[i, 1], conta[i, 2]] + 1
      }
      stopCluster(clust)

      atino <- numeric()
      noatino <- numeric()
      for (i in 1:(dim(contadores)[1])){
        noatino[i] <- 0
        for (j in 1:(dim(contadores)[2])){
          if(i == j){
            atino[i] <- contadores[i,j]
          } else {
            noatino[i] <- sum(noatino[i], contadores[i,j])
          }
        }
      }
      porcen <- numeric()
      for (i in 1:length(atino)){
```

```
            porcen[i] <- noatino[i]/(atino[i]+noatino[i])
            por <- c(porcen[i], negro, gris, blanco)
            data <- rbind(data, por)
        }
    }
  }
}
```

Por consiguiente, se obtuvieron los resultados del código anterior gráficamente, en la figura 2, en donde se puede observar como un gran porcentaje de error de cada dígito se encuentra muy cerca de la normalidad. Por lo cual, se procedió a realizar un análisis de varianza; para obtener información más detallada, cómo se muestra en el cuadro 1, en el cual se aprecia un mayor efecto en las probabilidades de los píxeles negros y blancos, en atención a lo cual los pixeles blancos cuentan con mayor relevancia, debido a la mayor influencia de clasificación.

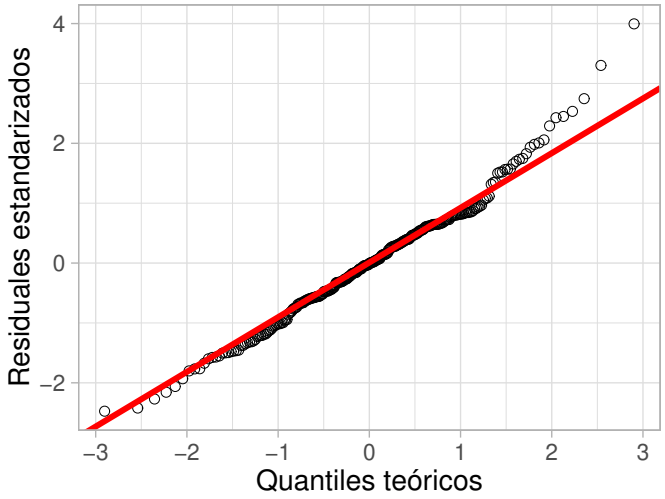

Figura 2: Evaluación de normalidad

Por otro lado, en el cuadro 1 se observa una diferencia estadísticamente significativa en el "Valor F" de 19.408 y 172.176 los píxeles Negro y blanco respectivamente; en comparación al gris, de manera que cuentan con mayores grados de libertad como se muestra en la columna "Df", así como con un un valor "Pr" por debajo de 0.05, que indica una menor desviación estándar. De modo que se puede concluir que hay un incremento en la probabilidad de presencia en los píxeles Blancos.

Cuadro 1: Resultados de análisis de varianza

	Df	Sum Sq	Media Sq	valor F	Pr(>F)
Negro	2	1.435	0.718	19.408	1.52e-08
Gris	2	0.502	0.251	6.792	0.00135
Blanco	2	12.732	6.366	172.176	<2e-16
Negro:Gris	4	0.020	0.005	0.138	0.96800
Negro:Blanco	4	0.245	0.061	1.657	0.16067
Gris:Blanco	4	0.678	0.170	4.586	0.00138
Negro:Gris:Blanco	8	0.183	0.023	0.617	0.76287
Residuals	243	8.984	0.037		

3.1. Reto 1

Con ayuda de la literatura [2] se realizó un documento CSV que se muestra en la parte inferior, en las cuales con la ayuda de la codificación de color para cada pixel, permitió realizar una resolución de imagen de 5×7 y reconocimiento de símbolos ASCII.

```
n g n g n g b b b g n b b b n g b b b g n b b b n g b b b g n g n g n
n n g b b b b n b b b b n b b b b g b b b b n b b b b n b b b n n g n n
n g g n g b b b b g b b b b n g g n g g n b b b b g b b b b b g n g g n
g n g n g b b b b n b b b b g g n g n n b b b b g b b b b n g n g n g
g g b g g n n b n n g g b g g n n g n n b b b g g b b b n n b b b g g
g n n g g g b b b b n b b b b n g g n n b b b b g b b b b b g n g g n n
g n g n g n b b b b g b b b b b n g n g n g b b b g n b b b n g n g n g
n g n g n b b b b g b b b b b n b b g n g b b b b n b b b b g b b b b n
n g n g n g b b b g n b b b b n g n g n g n b b b n g b b b g n g n g n
g n g n g n b b b n g b b b b g n g n g n g b b b g b b b b n b b b b g
n n b b b g g b b b n n b b b g g b b b n n b b b g g g g g g n n n n n
n g n g n g n g n g b b n b b b b g b b b b n b b g n g n g n g n g n
n g n g n g n g n g n g b b b g n b b b n g b b b g n g n g n g n g n
g g b g g n n b n n g g b g g n n n n n g g b g g n n b n n g g b g g
g n g n g n g n g n b b g b b b b n b b b b b g b b b b n b b b b b g b b
n g n g n g n g n g n g b b b g n g n b n g b b b g n g n g n g n g n
n n n n n n n n n n n n b b b n n n b b n n b b b n n b b b n n b b b
n n b n n g g b g g n n b n n g g b g g n n b n n g g g g g g n n n n n
g n g n g n b b b n g b b b g n g n g n g b b b b n b b b b g b b b b b
n g n g n n b b b n g b b b g n n g n n g b b b g n b b b n n b b b n
n b b b n g b b b g n b g b n g b n b g g b n b g n b g b n g g n g g
g n g n g n b n b n g b g b g n b n b n g b g b g n b b b n g b b b g
```

```
modelos <- read.csv("digitosext.modelo.csv", sep=" ", header=FALSE, stringsAsFactors=F)

r <- 7
c <- 5
dim <- r * c

n <- 49
w <- ceiling(sqrt(n))
h <- ceiling(n / w)
```

```
10
11 SASCIIAD <- c(0:9, "U", "L", "I", "C", "H", "M", "T", "P", "A", "W", "E", "F",)
12
13 png("plantilla.png", width=1600, height=2000)
14 par(mfrow=c(w, h), mar = c(0,0,7,0))
15 suppressMessages(library("sna"))
16
17 for (j in 1:n) {
18   d <- sample(0:21, 1)
19   pixeles <- runif(dim) < modelos[d + 1,]
20   imagen <- matrix(pixeles, nrow=r, ncol=c, byrow=TRUE)
21   plot.sociomatrix(imagen, drawlab=FALSE, diaglab=FALSE,
22                    main=paste(SASCIIAD[d+1], ""), cex.main=5)
23 }
24 graphics.off()
```

En la figura 3 se observa la plantilla con los números 0–9 y con los símbolos ASCII, generada con la ejecución del código.

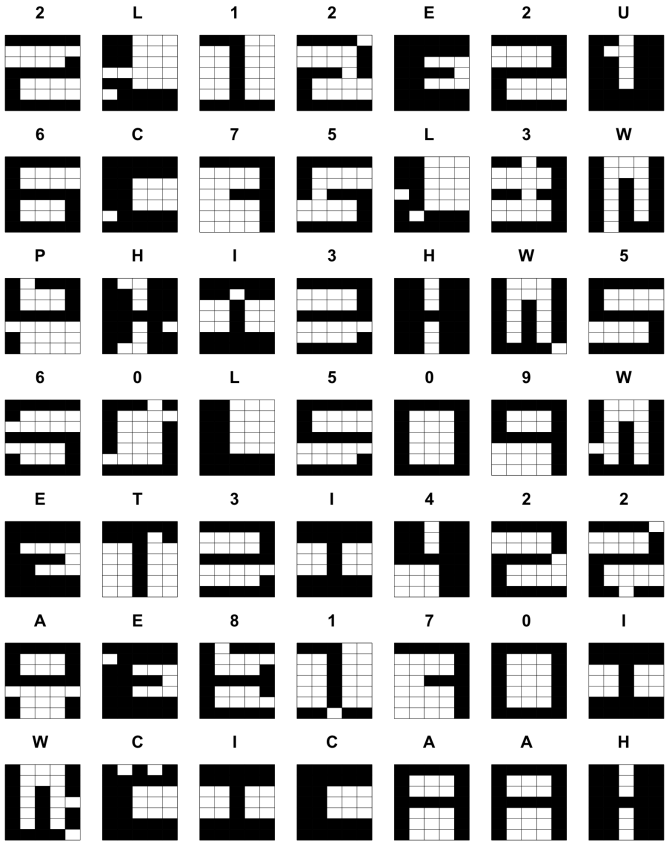

Figura 3: Plantilla con símbolos ASCII adicionales de 5×7

Referencias

[1] Astrid González. Red neuronal, 2018. URL https://sourceforge.net/p/gla-sim/exercises/HEAD/tree/Exercise_12/.

[2] Liliana Saus. Red neuronal, 2018. URL https://github.com/pejli/simulacion/tree/master/P12.

[3] Elisa Schaeffer. Práctica 12: red neuronal, 2019. URL https://elisa.dyndns-web.com/teaching/comp/par/p12.html.