Práctica 12: red neuronal

31 de octubre de 2017

1. Introducción

En esta práctica se trabaja el tema de redes neuronales, para simplificar el tema pensemos que tenemos una maquina a la cual le debemos de enseñar como distinguir una cosa de otra. En este caso tendremos los dígitos del cero al nueve representados en 15 pixeles cada uno, el objetivo de la tarea en será paralelizar el código existente y observar las variaciones de los tiempos respecto a la cantidad de dígitos generados, en el objetivo general aún no se pide que se observe la acertividad de la red, sin embargo es un punto que se tratará en los retos uno y dos.

2. Parámetros de trabajo

La experiemtnación se realizó en un HP Z230 Tower Workstation con procesador Intel(R) Xenon(R) CPU E3-1240 v3 y 3.40 GHz de memoria ram 16 GB y un sistema operativo de 64 bits con Windows 7 Home Premium.

Se fijó una cantidad de veinte repeticiones para cada una de las combinaciones de las variables, para la cantidad de imágenes o dígitos que se generaron fue $n \in \{300, 500, 700, 1000\}$, se mantuvo la cantidad de rondas de entrenamiento en 5000.

3. Modificaciones del código

Lo primero que se incluyó en el código fue la librería `Parallel`, así como la función `fprueba`. Así como el comando necesario para la creación de los `clusters`.

```
library(parallel)
fprueba <- function(i){
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
if(correcto[i] == salida[i]){
return(c(d,1))
}else{return(c(decimal(salida, n),1))}
} }
cluster <- makeCluster(detectCores() - 1)
clusterExport(cluster, "fprueba")
clusterExport(cluster, "modelos")
clusterExport(cluster, "binario")
...
```

Las se enviaron las variables necesarias para la paralelización, de igual forma se incluyeron los contadores de tiempo, creando los `data.frame` que iban a guardad a dichas variables. en total se

utilizaron para esta parte dos ciclos `for`, uno que corriera para la cantidad de réplicas y otro que iba variando la cantidad de cifras generadas.

4. Resultados y conclusiones

En la figura **??** podemos observar de forma clara la diferencia de tiempos de ejecución. Es muy notoria la diferencia cuando la cantidad de dígitos generada crece.

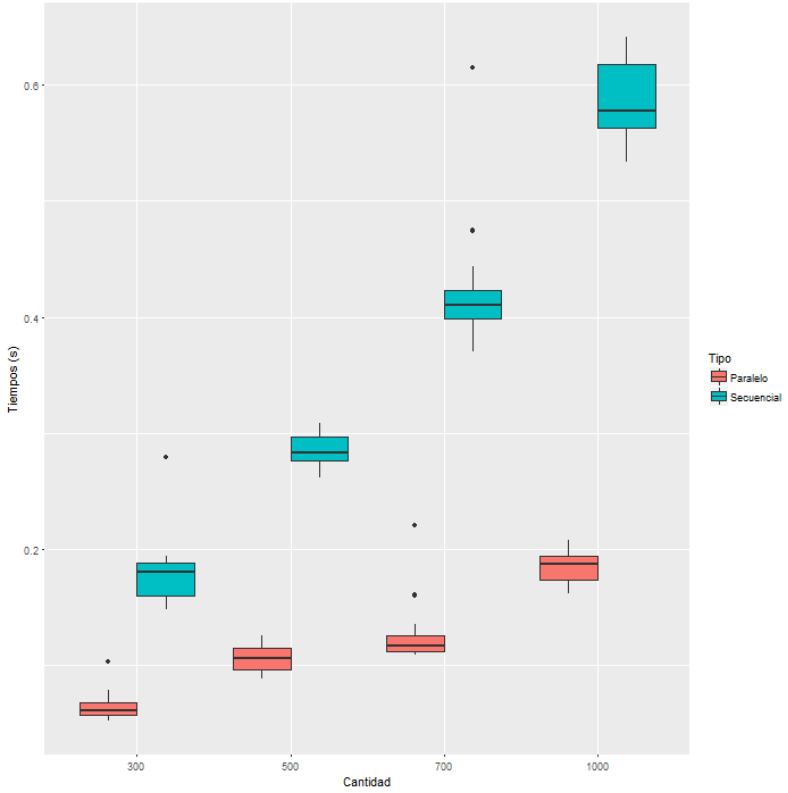

Figura 1: Tiempos de la ejecución con respecto a cantidad de dígitos generados.

5. Reto 1

El primer reto consiste en estudiar el desempeño de la red, es decir que tan asertiva es con respecto a la variación de las probabilidades asignadas por color. Aquí es necesario aclarar, para la generación de la imagen, es decir el número de forma gráfica, a cada uno de los bits de la secuencia se le asignó un color y una probabilidad de pintarse de ese color, el color negro como gris tienen una alta probabilidad de pintarse, y el blanco por contrario posee una muy baja, observaremos que pasa si las probabilidades se invierten o varían.

6. Modificación del código y parámetros de R1

Para este reto fue necesario realizar algunos cambios al código paralelizado, se retiraron los comandos de medición de tiempo, se mantuvo el ciclo de variación de población pero con una ligera modificación, siendo los valores utilizados en las probabilidades los siguientes: $pn \in \{.995, .9, .995, .9, .002, .0003\}$, `pn` representa la probabilidad del negro, de manera similar para en gris y el blanco,$pg \in \{.92, .825, .995, .9, .002, .0002\}$, $pb \in \{.002, .093, .995, .9, .002, .0003\}$. De igual forma para variar las probabilidades , fue necesario crear un ciclo `for`, este con el fin de automatizar las réplicas.

```
for(vg in 1:length(pn)){
modelos[modelos=='n'] <- mn[vg]
```

```
modelos[modelos=='g'] <- mg[vg]
modelos[modelos=='b'] <- mb[vg]
...
```

7. Resultados y conclusiones de R1

Para fines ilustrativos las primeras dos combinaciones de las probabilidades son muy parecidas a los originales, es decir podríamos esperar que estas representaran fielmente la experimentación original. La combinación tres y cuatro poseen valores muy grandes tanto para el color negro como para el blanco, es decir de este se podría suponer un error un poco mayor ya que mi imagen no es muy clara para la selección, de modo contrario la combinación cinco y seis poseen valores muy pequeños para los tres colores.

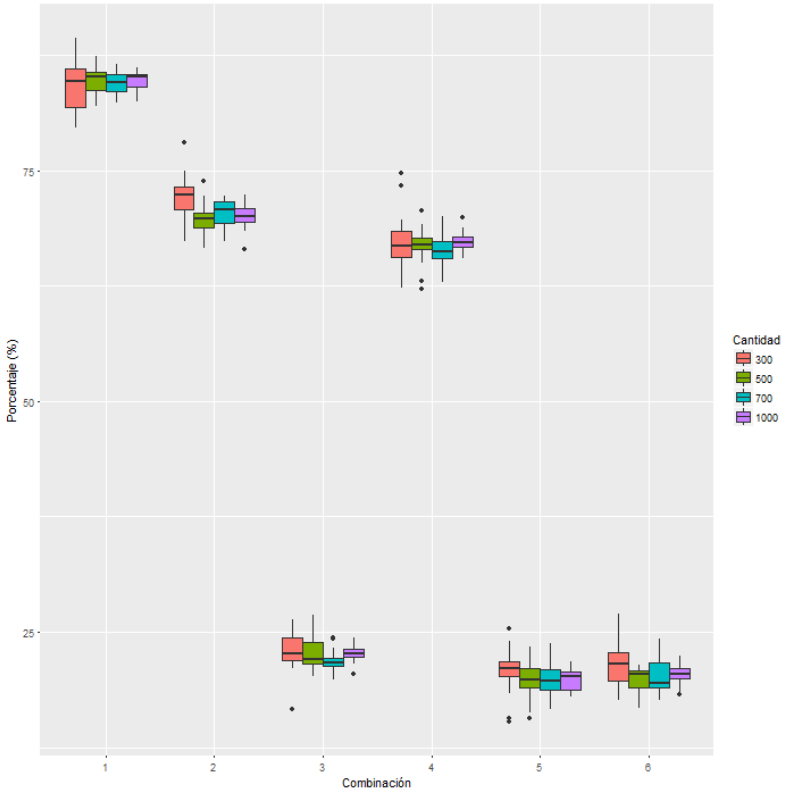

Figura 2: Porcentaje de aciertos con respecto a las variaciones ya establecidas.

Como podemos observar los que arrojan mayor error de manera general son las combinaciones donde tanto como los colores obscuros así como el blanco poseen probabilidades muy pequeñas, esto sugiere que bien se pudo haber seleccionado de forma similar los colores, sin prioridad, de igual forma la combinación tres en la cual las probabilidades de los tres colores era muy alta y de igual valor.

8. Reto 2

El reto dos consiste en modificar la cantidad de valores o caracteres a identificar, es decir, ya no vamos a poseer solamente los valores del cero al nueve,

9. Modificación de código y parámetros R2

10. Resultados y conclusiones de R2

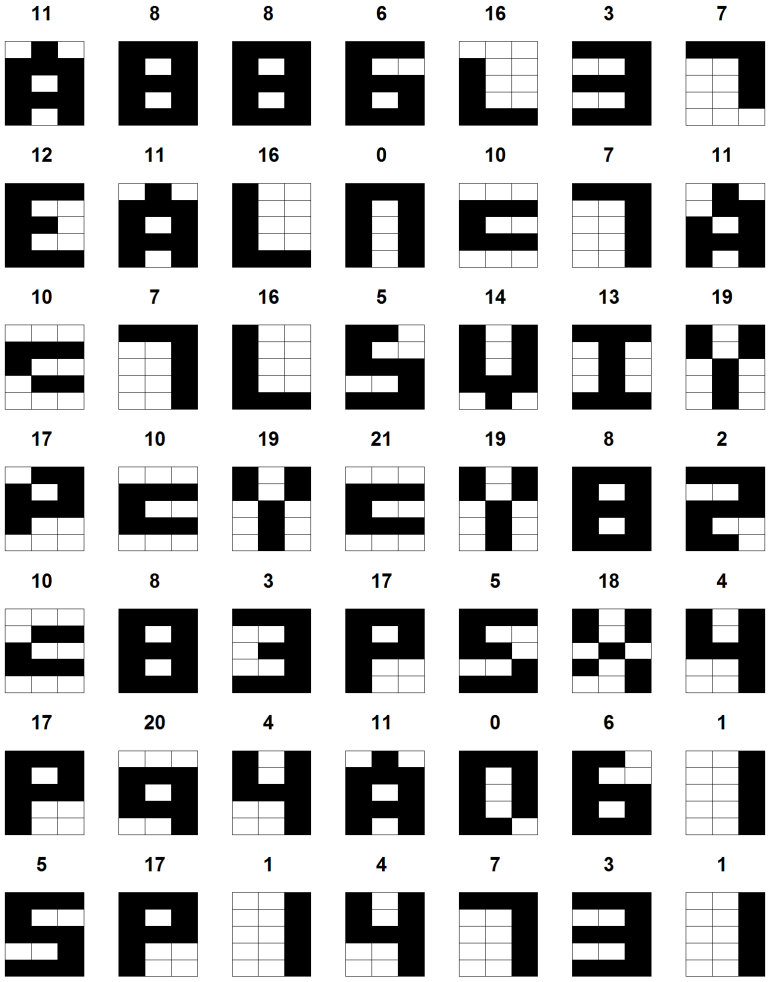

Figura 3:

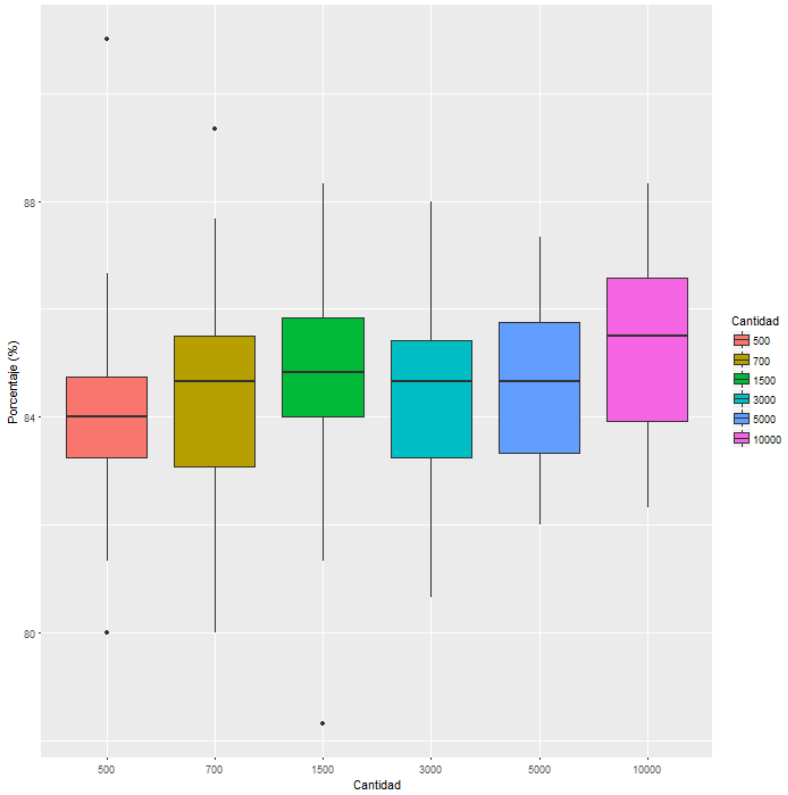

Figura 4: Porcentaje de acierto respecto a variación en entrenamiento