Práctica 12: red neuronal

1445183

7 de mayo de 2019

1. Objetivo

Estudiar de manera sistemática el desempeño de la red neuronal para diez dígitos en función de las tres probabilidades asignadas a la generación de dígitos (`ngb`) en el código proporcionado por esta práctica [1] variando a las tres en un experimento factorial adecuado.

2. Descripción

El código se paraleliza desde el principio, usando tres núcleos de los cuatro posibles.

```
cluster <- makeCluster(detectCores() - 1)


binario <- function(d, l) {
  b <-  rep(FALSE, l)
  while (l > 0 | d > 0) {
    b[l] <- (d %%2 == 1)
    l <- l - 1
    d <- bitwShiftR(d, 1)
  }
  return(b)
}
decimal <- function(bits, l) {
  valor <- 0
  for (pos in 1:l) {
    valor <- valor + 2^(l - pos) * bits[pos]
  }
  return(valor)
}

.
.
.

 clusterExport(cluster, c("neuronas", "binario", "decimal", "modelos", "tope" ,"k", "dim", "n"))


        #PRUEBA
        contadores <-parSapply(cluster, 1:prueba , function(x){
          d <- sample(0:tope, 1)
          pixeles <- runif(dim) < modelos[d + 1,] # fila 1 contiene el cero, etc.
          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)
          return(r==d)})
        datos<-rbind(datos, data.frame(Replica= replicas, Negro=pn, Gris=pg, Blanco=pb, Porcentaje=(sum
            (contadores)/prueba)*100))

      }
    }
  }
}
```

Se genera un archivo *CSV* con los datos proporcinados por la práctica para indicar la ubicación de los pixeles y se vincula al código en la rutina de generación de pixeles (`ngb`), después se varían las probabilidades para `ngb` como se muestra en el siguiente código haciendo uso de `for` con `20` repeticiones cada una y obteniendo los respectivos porcentajes para cada combinación.

```
#repeticiones
replica<-20

tmax<-5000
entrenamiento <- ceiling(0.7 * tmax)
prueba <- tmax - entrenamiento
datos<- data.frame( Replica= integer(), Negras=integer(), Grises=integer(), Blancas=integer(),
    Porcentaje=integer())


#archivo csv
modelos <- read.csv("digitos.modelo.csv", sep=" ", header=FALSE, stringsAsFactors=F)

#variar probabilidades
for (PN in c(0.995,0.92,0.002)) {
  for(PG in c(0.92,0.002,0.995)){
    for(PB in c(0.002,0.995,0.92)){

      modelos[modelos=='n'] <- pn # pixeles negros en plantillas
      modelos[modelos=='g'] <- pg # pixeles grises en plantillas
      modelos[modelos=='b'] <- pb # pixeles blancos en plantillas
      for(replicas in 1:replica){
        print(replicas)

        tasa <- 0.15
        contadores <-vector()
        n <- floor(log(k-1, 2)) + 1
        neuronas <- matrix(runif(n * dim), nrow=n, ncol=dim) # perceptrones


         #ENTRENAMIENTO
        for (t in 1:entrenamiento) { # entrenamiento
          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
```

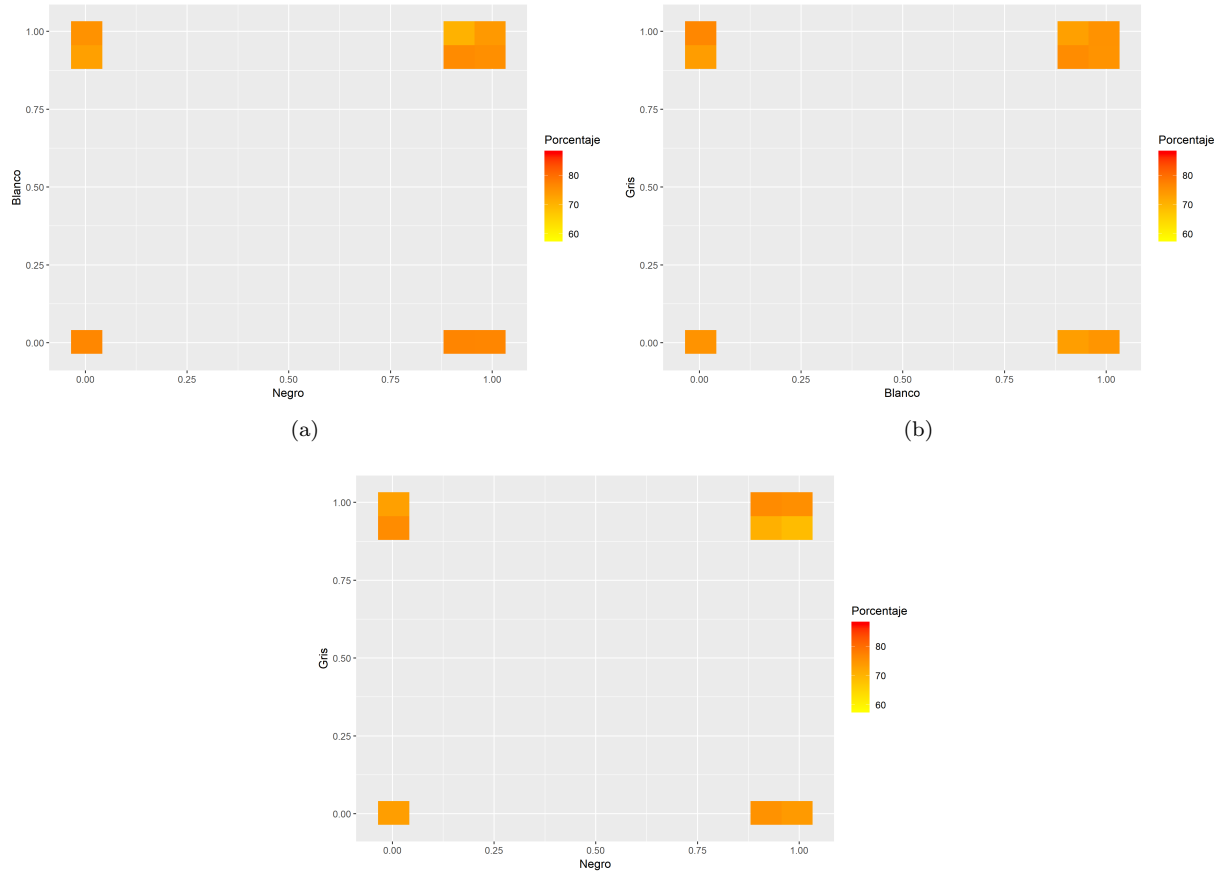
```
43                  }
44              }
45          }
```

3. Resultados

En la figura 1*a* se puede observar que existe mayor porcentaje cuando la probabilidad de *Negro* es mayor (crecano al 1).

En la figura 1*b* el mayor porcentaje es cuando la probabilidad es mayor en *Gris*.

En la figura 1*c* el mayor porcentaje se encuentra a cualquier probabilidad de *Negro* (preferentemente cercano al 0)y mayor probabilidad de *Gris*.



(a) (b)

(c)

Figura 1:

4. Conclusión

Los pixeles que dominan la formación de caracteres son los pixeles *Negro* y *Gris* dado a que los porcentajes de que se reconociera correctamente los dígitos fueron mayores para estos pixeles sin importar las variaciones y combinaciones de probabilidades.

Reto 1

El primer reto consiste en extender y entrenar la red neuronal para que reconozca además por lo menos doce símbolos ASCII adicionales, aumentando la resolución de las imágenes a 5×7 de lo original de 3×5, estos se obtuvieron tomando de referencia a *Saus* [**?**]

```
modelos <- read.csv("digitos.reto.csv", sep=" ", header=FALSE, stringsAsFactors=F)
modelos[modelos=='n'] <- 0.995 # pixeles negros en plantillas
modelos[modelos=='g'] <- 0.92 # pixeles grises en plantillas
modelos[modelos=='b'] <- 0.002 # pixeles blancos en plantillas

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

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

letras <- c(0:9, "L", "I", "C", "H", "T", "E", "F", "U", "P", "A", "W", "M")

png("plantilla.png", width=1600, height=2000)
par(mfrow=c(w, h), mar = c(0,0,7,0))
suppressMessages(library("sna"))

for (j in 1:n) {
  d <- sample(0:21, 1)
  pixeles <- runif(dim) < modelos[d + 1,] # fila 1 contiene el cero, etc.
  imagen <- matrix(pixeles, nrow=r, ncol=c, byrow=TRUE)
  plot.sociomatrix(imagen, drawlab=FALSE, diaglab=FALSE,
                   main=paste(letras[d+1], ""), cex.main=5)
}
graphics.off()
```

5. Resultados

En la figura 2 se observan los números y letras codificadas donde solo 18 caracteres fueron reconocidos de manera satisfactoria.

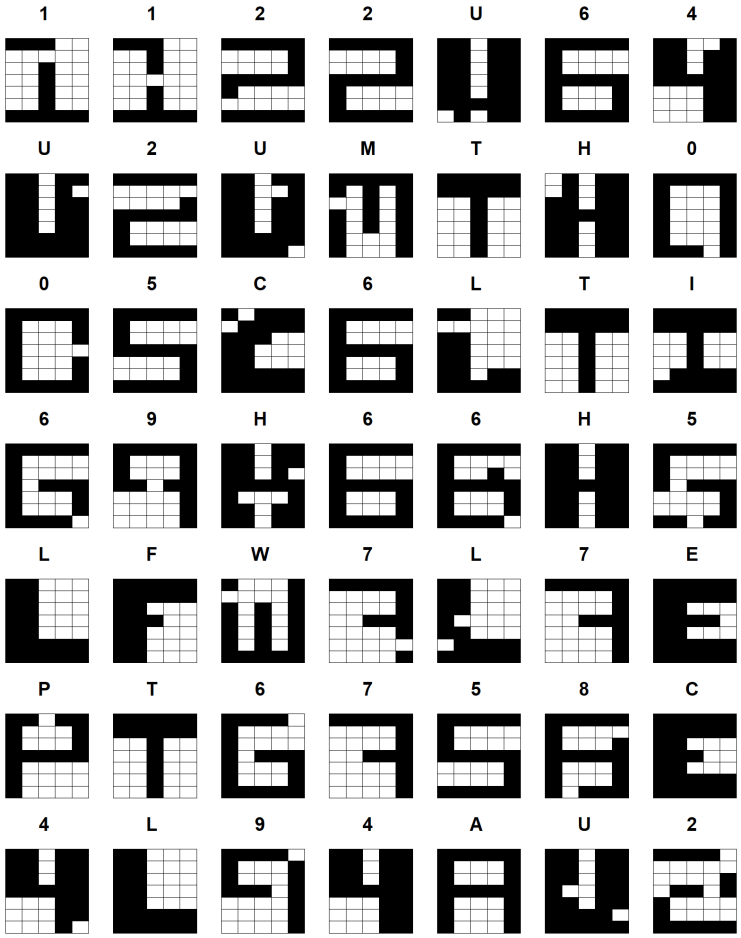


Figura 2: Caracteres de red neuronal extendida

6. Conclusión

En este caso no se pudo entrenar debidamente la red neuronal bajo estas nuevas condiciones, observando la figura 2 se puede deducir que para una red neuronal extendida se necesita mayor tiempo de entrenamiento para que posteriormente pueda hacer un reconocimiento correcto.

Referencias

[1] Elisa Schaeffer. Práctica 11: Frentes de Pareto, 2019. URL https://elisa.dyndns-web.com/teaching/comp/par/p12.html.