P12

edgardjfc

May 2022

1 Introduction

The objective of this assignment is to make a neural network and train it to recognize basic text characters of number digits. With the goal of studying its behavioral score to see its performance given the variation of pixels of color white, grey and black.

2 Code

The base for this code was written based on the work of our teacher E. Schaeffer [3] and classmate J. "FeroxDeitas" [2]. With the entirety of this code being found in the github repository of E. "edgardjfc" [1].

The first part of the code describes the two functions used to transform binary numbers to decimal ones, and decimal numbers into binary.

```
binary <- 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)
}
```

The following data frame describes the probability for the different colors of each cell for each character.

```
df = data.frame()
probn = c(0.98,0.65,0.35) #black
probg = c(0.95,0.75,0.55) #grey
probb = c(0.45,0.10,0.005) #white
```

In the following code begin the iterations of the lists of probabilities, and 12 repetition for each iteration is made, by taking the values from a text file organized as a grid, recreating a bit map similar to the fig. 2.

```
for (ne in probn){
  for (g in probg){
    for (b in probb){
      for (replica in 1:12){
        modelos <- read.csv("digits.txt", sep="␣",
            header=FALSE, stringsAsFactors=F)
        modelos[modelos=='n'] <- ne
        modelos[modelos=='g'] <- g
        modelos[modelos=='b'] <- b

        r <- 5
        c <- 3
        dim <- r * c

        tasa <- 0.15
        tranqui <- 0.99

        tope <- 9
        digitos <- 0:tope
        k <- length(digitos)
        contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
        rownames(contadores) <- 0:tope
        colnames(contadores) <- c(0:tope, NA)
```

The following code establishes the learning rate of the neural network. It consists in taking the values from 0 to 9 and comparing the results with the previous part of the code.

```
n <- floor(log(k-1, 2)) + 1
        neuronas <- matrix(runif(n * dim), nrow=n, ncol=dim)

        for (t in 1:5000) { # 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
              }
            }
          }
```

This part of the code is in charge of applying the trained receptors in order to make decisions directly on the bit map. This will determine the value of each digit. Then these values are scored on their precision.

```
for (t in 1:300) { # prueba
          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
        }
        #Calculo puntaje F-score
        precision = diag(contadores) / colSums(contadores[,1:10])
        recall = diag(contadores) / rowSums(contadores)
        fscore = (2 * precision * recall) / (precision + recall)
        result = c(ne, g, b, replica, fscore)
        df = rbind(df, result)
      }
    }
  }
}
```

3 Results

On figure 1 we can observe the distribution of F-score on the different variables of probability with each having 12 repetitions. We can see how the images generated with more black pixels and less grey pixels are more accurate, with a higher f-score. This is to be expected since by having less grey pixels there is less ambiguity and more concrete shapes.

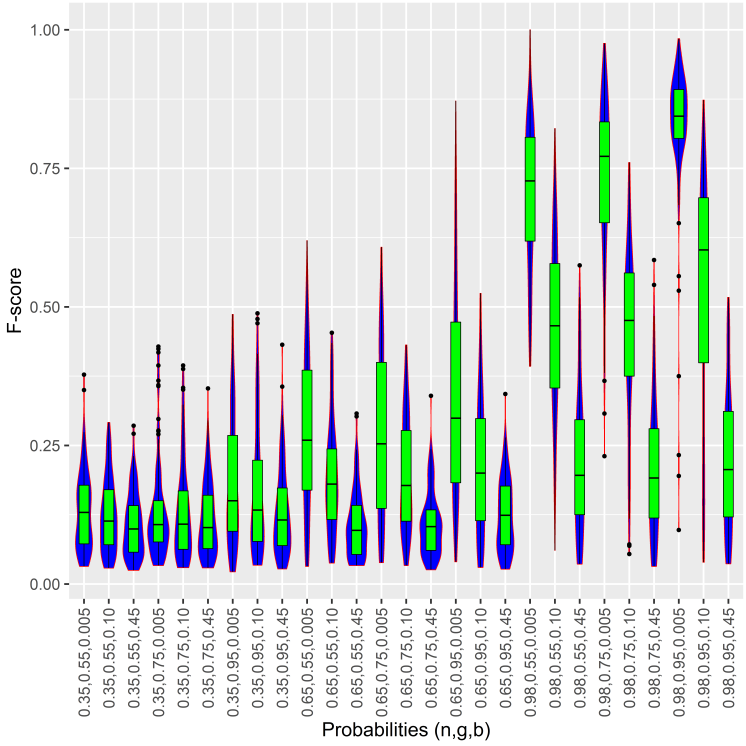

Figure 1: Results against their F-score

4 Analysis

Statistical analysis were needed to obtain a conclusive understanding on the results. A Shapiro-Wilk test was done to observe the distribution of the results of the F-scores. Then a Levene test to observe the variance between the groups. The results in figure 2 show a value p is smaller than 0.05 meaning they do not have a normal distribution.

Due to not having a normal distribution, a Kruskall-Wallis test was done to determine if there was a significant difference of F depending on the innitial

probabilities. Based on the results we can observe in figure 3 we can see that the initial values of probability do affect the F-score in a significant way.

Figure 2: Resultads of the normality and variance tests.

Outliers	47
Normality by group	0.35/0.55/0.005: $p = 1.72 \times 10^{-6}$ 0.35/0.55/0.10: $p = 1.81 \times 10^{-4}$ 0.35/0.55/0.45: $p = 1.06 \times 10^{-4}$ 0.35/0.75/0.005: $p = 1.62 \times 10^{-7}$ 0.35/0.75/0.10: $p = 2.02 \times 10^{-6}$ 0.35/0.75/0.45: $p = 2.37 \times 10^{-4}$ 0.35/0.95/0.005: $p = 2.29 \times 10^{-6}$ 0.35/0.95/0.10: $p = 4.52 \times 10^{-6}$ 0.35/0.95/0.45: $p = 2.00 \times 10^{-4}$ 0.65/0.55/0.005: $p = 5.26 \times 10^{-5}$ 0.65/0.55/0.10: $p = 2.39 \times 10^{-2}$ 0.65/0.55/0.45: $p = 2.34 \times 10^{-2}$ 0.65/0.75/0.005: $p = 3.41 \times 10^{-3}$ 0.65/0.75/0.10: $p = 3.91 \times 10^{-4}$ 0.65/0.75/0.45: $p = 1.21 \times 10^{-5}$ 0.65/0.95/0.005: $p = 6.12 \times 10^{-3}$ 0.65/0.95/0.10: $p = 8.42 \times 10^{-4}$ 0.65/0.95/0.45: $p = 7.76 \times 10^{-5}$ 0.98/0.55/0.005: $p = 6.71 \times 10^{-4}$ 0.98/0.55/0.10: $p = 2.52 \times 10^{-1}$ 0.98/0.55/0.45: $p = 4.42 \times 10^{-4}$ 0.98/0.75/0.005: $p = 8.88 \times 10^{-6}$ 0.98/0.75/0.10: $p = 1.21 \times 10^{-1}$ 0.98/0.75/0.45: $p = 3.02 \times 10^{-3}$ 0.98/0.95/0.005: $p = 8.81 \times 10^{-15}$ 0.98/0.95/0.10: $p = 2.41 \times 10^{-6}$ 0.98/0.95/0.45: $p = 1.31 \times 10^{-3}$
Homogeneity by variance	$p = 3.39 \times 10^{-22}$

5 Conclusions

By the results we observed in the figure 2 and 3 we can conclude that the probability for each of the pixels to be a different color affects directly the ability for the neural network to process and make accurate readings of the images it is given. An important detail we can infer is that the grey pixels can lead to a poor quality image with vague and inaccurate readings, thus we can

Figure 3: Results of Kruskall-Wallis tests

Chi squared	Value of p
1660.2	2.2×10^{-16}

see a better F-score on the samples that have a lower probability of grey pixels and a higher probability of black pixels.

6 Challenge 1

The goal of this challenge is to extend the neural network to understand a larger array of symbols in the same format as the base assignment, with the key difference that these will have a resolution of 5 by 7, as opposed to the original 3 by 5.

The code used by this challenge is very similar to the base assignment, with the difference that the number of receptors used will be higher, since we will be using a higher number of symbols in this experiment.

```
modelos <- read.csv("digits2.txt", sep="␣", header=FALSE, stringsAsFactors=F)
modelos[modelos=='n'] <- 0.995
modelos[modelos=='g'] <- 0.92
modelos[modelos=='b'] <- 0.002

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

tasa <- 0.15
tranqui <- 0.99

tope <- 21
digitos <- 0:tope
k <- length(digitos)
contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
rownames(contadores) <- 0:tope
colnames(contadores) <- c(0:tope, NA)

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

The results obtained are in the form of the grid found in figure 4, where the different symbols are organized with their likely hood to be compared to a generated symbol.

We can observe how the results are mainly within the diagonal from top left to bottom right, although there are very predominant outliers, the main bulk of

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	<NA>
0	0	0	0	0	0	0	0	0	10	2	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	10	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	8	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	1	0
5	1	1	0	0	0	11	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	1	10	0	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
10	0	0	0	0	0	0	0	0	4	0	11	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	1	0	0	0	12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	15	0	0	0	0	0	0	0	0	3
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
16	0	0	0	0	0	0	0	0	10	2	0	1	0	0	0	1	0	0	0	0	0	0	0
17	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	2
18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	1	0	0	3
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	1	2
20	0	0	0	0	3	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	2	0	1
21	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	4

Figure 4: Matrix of results

results show that the neural network is able to trend towards accurate results.

References

[1] E. Edgardjfc. *GitHub,P11*. URL: `https://github.com/edgardjfc/Simulacion-Nano-2022/tree/main/P12`.

[2] J. FeroxDeitas. *GitHub,P11*. URL: `https://github.com/FeroxDeitas/Simulacion-Nano/blob/main/Tareas/P_12`.

[3] E. Schaeffer. *GitHub,neuralNetwork*. URL: `https://github.com/satuelisa/Simulation/tree/master/NeuralNetwork`.

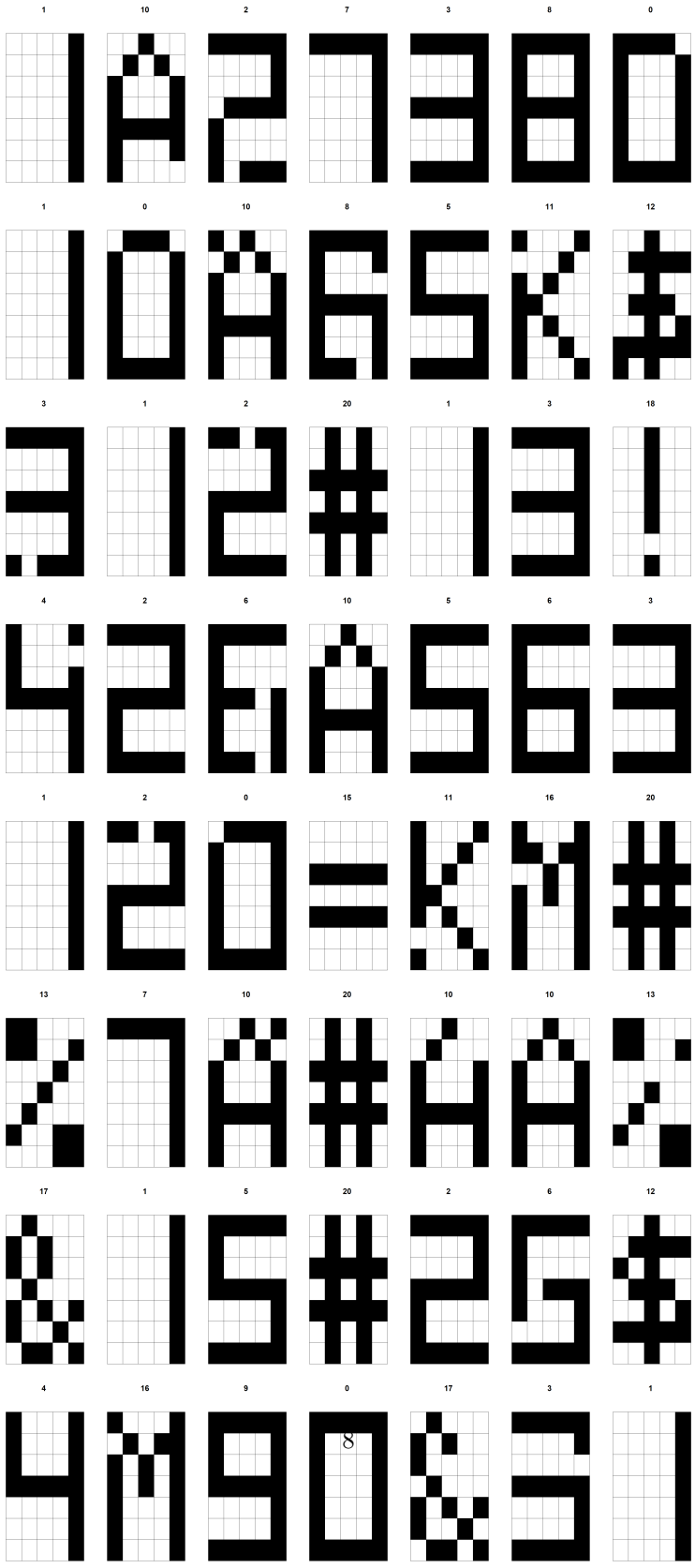

Figure 5: Examples of characters being read.