Simulación de Sistemas
Práctica 12

Red neuronal

1455175: Ángel Moreno

30 de octubre de 2018

1. Introducción

La ultima práctica trata de reconocer dígitos de imágenes en blanco y negro con una red neuronal. El El elemento básico de una red neuronal es un **perceptrón** que busca colocarse en la frontera que separa las entradas verdaderas y las entradas falsas.

En la tarea se usan quince pixeles por dígito, tres de ancho y cinco de altura. Para que la red neuronal identifique los diez dígitos del sistema decimal se usa la **representación en base de dos** del los diez dígitos esto implica utilizar una cadena de cuatro **bits**, es decir, cuatro percentrones.

Además se usa una platilla para crear números de una manera probabilista en donde los pixeles que son negros en la plantilla serán puestas casi siempre, mientras los grises ocasionalmente, y los blancos solamente como ruido aleatorio de poca frecuencia.

1.1. Tarea

Paraleliza lo que se pueda sobre la red neuronal (no todo se podrá durante entrenamiento, ya que los pesos pueden cambiar en cada paso, pero una vez entrenado, ya se facilita más). Estudia el efecto de esto en su tiempo de ejecución con las pruebas estadísticas y las visualizaciones pertinentes.

2. Simulación

Se hizo en paralelo la prueba de la red neuronal utilizando la función `parRapply` en `R`, la cual trabaja en paralelo sobre una matriz como iterador fila por fila. Se usa el siguiente código:

```
numeros <- table(sample(0:tope, prueba, replace = TRUE))
numeros <- cbind(0:tope, as.numeric(numeros))

contadores <- parRapply(nucleos, numeros, function(l){
        count <- rep(0, k+1)
        num <- l[1]
        cuantos <- l[2]
        modelo <- modelos[num+1,]
        for (i in 1:cuantos) {
          pixeles <- runif(dim) < modelo
          salida <- rep(FALSE, n)
          for (j in 1:n) {
            w <- neuronas[j,]
            resultado <- sum(w * pixeles) >= 0
            salida[j] <- resultado
          }
          digito <- min(decimal(salida, n), k)
          count[digito + 1] <- count[digito + 1] + 1
        }
        return(count)
        })
```

Se ejecuta una simulación donde se varia cantidad de iteraciones máximas en 1000, 2500, 5000, 10000 y 20000, el resto de los parámetros están fijos como las probabilidades de los pixeles negros, grises y blancos con se utiliza en el ejemplo para la tarea.

2.1. Resultados

La figura 1 muestra los resultados.

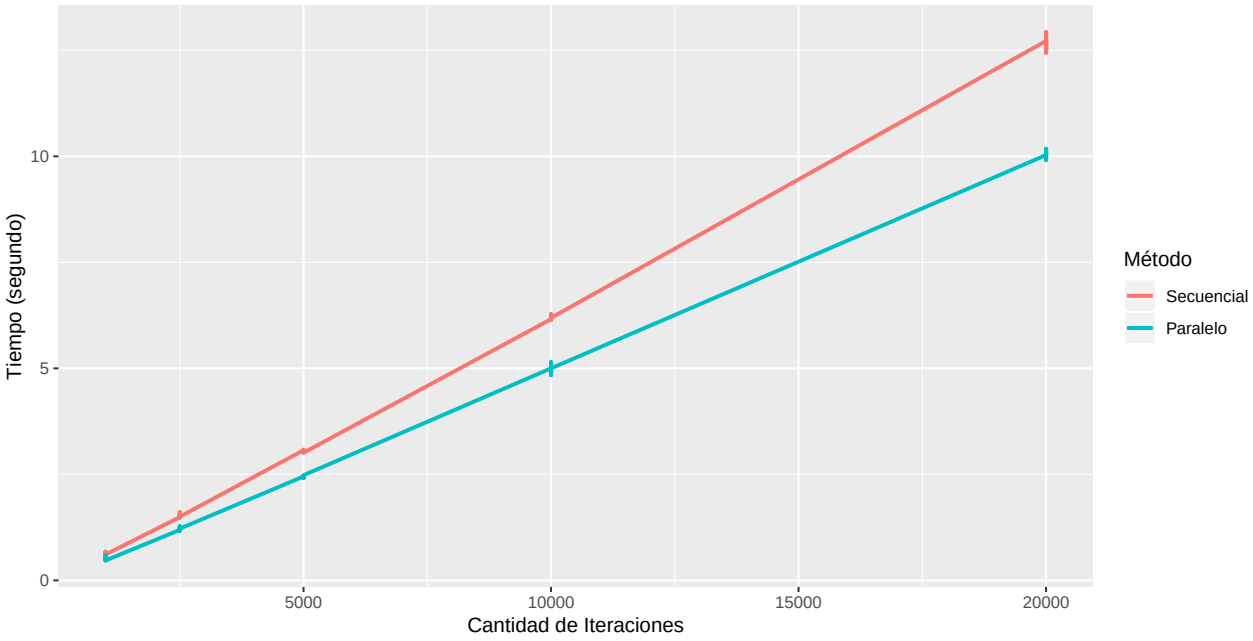

Figura 1: Resultados de tiempos de ejecución contra la cantidad de iteraciones de 15 réplicas.

Se observa resultados con comportamientos lineales y "pareceüna ligera variación significativa. Se verifica utilizando una prueba estadística no paramétrica Kruskal-Wallis donde comprueba que que si hay diferencias estadísticas sobre el método utilizado en la simulación.

```
  Kruskal-Wallis rank sum test

data:  Metodo by Tiempo
Kruskal-Wallis chi-squared = 149, df = 96, p-value = 0.0004306
```

3. Reto 1

El primer reto consiste en 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.

Se implementa la simulación con una cantidad de iteraciones fija de 10000 con 10 réplicas, variando las probabilidades de los pixeles para los negros en 0.9, 0.95 y 0.995, los grises en 0.75, 0.85 y 0.92, los blancos en 0.001, 0.01 y 0.1, y se tiene como variable respuesta el porcentaje de error en el acierto del número. La figura 2 muestra el diseño factorial elaborado.

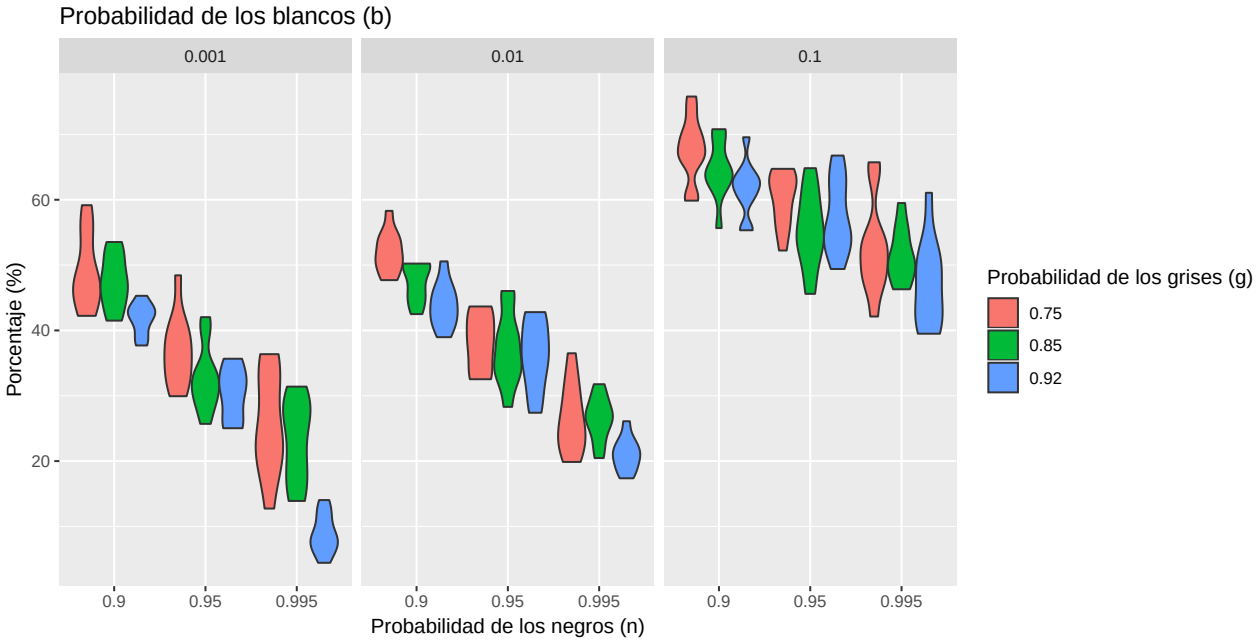

Figura 2: Resultado del porcentaje de error del diseño factorial .

Se observa la eficiencia de la combinación elegida inicial, y como los porcentajes varían conforme varían los factores. Como la combinación de usar las probabilidades para los pixeles negros, grises y blancos de 0.9, 0.75 y 0.1 respectivamente, se tienen peores resultados.

Referencias

[1] Schaeffer E. *R paralelo: simulación y análisis de datos, 2018.* https://elisa.dyndns-web.com/teaching/comp/par/

[2] Valdes E. *Repository of Github, 2017.* https://github.com/eduardovaldesga/SimulacionSistemas

[3] Saus L. *Repository of Github, 2018.* https://github.com/pejli/simulacion