Práctica 12: red neuronal

En esta práctica se simula el proceso de aprendizaje utilizando una red neuronal, cuyo elemento básico es llamado perceptrón. Idealmente, un perceptrón es la frontera entre una entrada verdadera y otra falsa, sin embargo, para lograr este grado de exactitud es necesario llevar a cabo una serie de ajustes al perceptrón inicial durante un lapso, conocido como fase de entrenamiento. Después de esta fase, se prosigue a la fase de prueba, en el cual se evalúa la eficiencia de dicho perceptrón, es decir, cual es el porcentaje de aciertos obtenidos.

Meta

Paralelizar el código secuencial utilizado como ejemplo en clase [1] y estudiar el efecto de la paralelización en el tiempo de ejecución.

Desarrollo del código

Con el fin de eliminar la elección al azar del número correcto `d`, se procedió a crear el vector `combinaciones` el cual es una secuencia del vector `digitos` ordenada al azar, cuyos valores se repiten con la misma frecuencia.

```
> test<-t.pruebas/length(digitos)
> combinaciones<-rep(digitos,test)
> combinaciones<-sample(combinaciones)
```

Con lo que respecta al objetivo mencionado previamente, no se recomienda paralelizar la fase de aprendizaje o entrenamiento, ya que en ella se realizan cambios a los perceptrones si se obtiene una respuesta incorrecta., por lo que un paso depende del valor anterior, mientras que en la fase de pruebas se evalúa de manera independiente si la respuesta es correcta o no. Por este motivo sólo se paralelizó la fase de prueba utilizando la función `prueba` de la cual se obtiene el vector lógico `aciertos` que contabiliza las identificaciones correctas durante este periodo. Cabe destacar que la paralelización se realizó utilizando el paquete `doParallel`.

```
> prueba<-function(t){
+   d <- combinaciones[t]
+   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)
+    if (r==d){return(TRUE)}
+ }
> acierto<-foreach(t=1:t.pruebas,.combine=c)%dopar%prueba(t)
> acierto<-sum(acierto)/t.pruebas*100
```

Con el objetivo de observar el efecto de la paralelización en el tiempo de ejecución de ambas versiones de la red neuronal, se procedió a utilizar un nuevo código en el cual se varía el número de pruebas y se toma el tiempo de ejecución de diez repeticiones de estas versiones. Estos resultados serán presentados gráficamente utilizando el paquete `ggplot2`.

```
> for (replica in 1:10){
+   for (t.pruebas in seq(300,900,300)){
+
+   source('~/GitHub/Simulacion/Simulacion/P12/Codigo/P12.R')
+   secuencial<-cbind(replica,"Original",t.pruebas,tiempo,acierto)
+
+   source('~/GitHub/Simulacion/Simulacion/P12/Codigo/P12_T12.R')
+   paralelo<-cbind(replica,"Paralelo",t.pruebas,tiempo,acierto)
+
+   resultados<-rbind(resultados,secuencial,paralelo)
+    }
+ }
```

Resultados

En la figura 1 muestra un gráfico tipo caja-bigote en el que se muestran los tiempos de ejecución del código de red neuronal en su versión secuencial y paralelizada. En este gráfico se aprecia una leve mejoría en el tiempo de ejecución cuando se elige trabajar con la versión paralelizada, además se observa que la diferencia entre ambos tiempos de ejecución aumenta cuando el número de pruebas así lo hace.

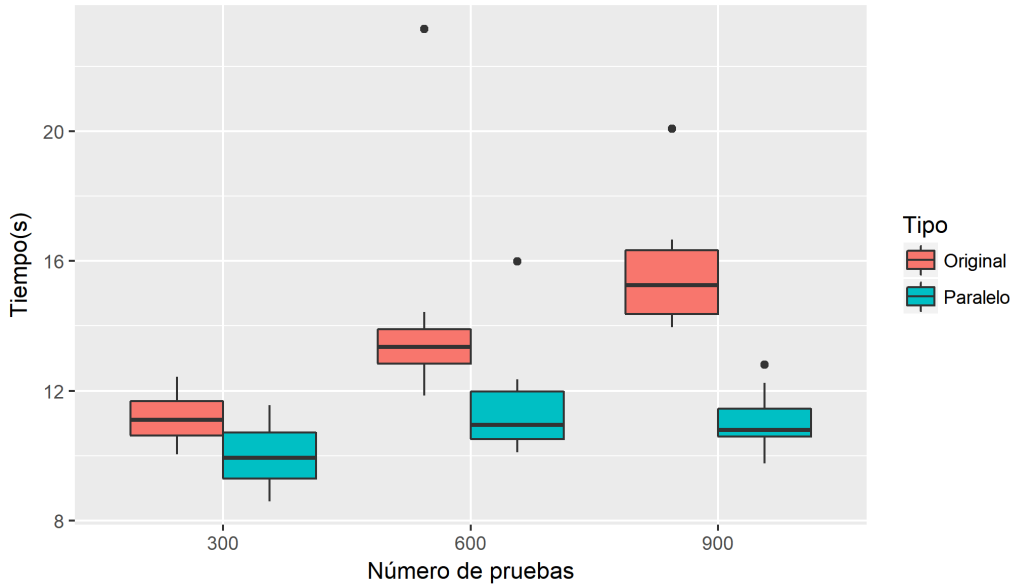


Figura 1 Variación en el tiempo para el código secuencial y el código paralelo en función al número de pruebas

Si bien existe una diferencia apreciable entre los tiempos de ejecución es necesario confirmar la naturaleza de esta diferencia con base en pruebas estadísticas como la prueba `t.test`. Para ello se procedió a formular la siguiente hipótesis nula: "El número de pruebas no interfiere en el aumento del tiempo de ejecución del programa secuencial y paralelizado". Los resultados se presentan en el Cuadro 1, de la cual se concluye que la diferencia entre ambos tiempos de ejecución es significativa, rechazando esta hipótesis.

Cuadro 1 Influencia del número de pruebas en el tiempo de ejecución

Numero de pruebas	p-valor	Hipótesis nula	Diferencia
300	0.008833	Nula	Significativa
600	0.03679	Nula	Significativa
900	9.793×10^{-6}	Nula	Significativa

En la figura 2 se muestra el porcentaje de aciertos en ambas versiones de la red neuronal en función al número de pruebas. Se puede observar que el código paralelizado marca una mejoría con respecto al código secuencial.

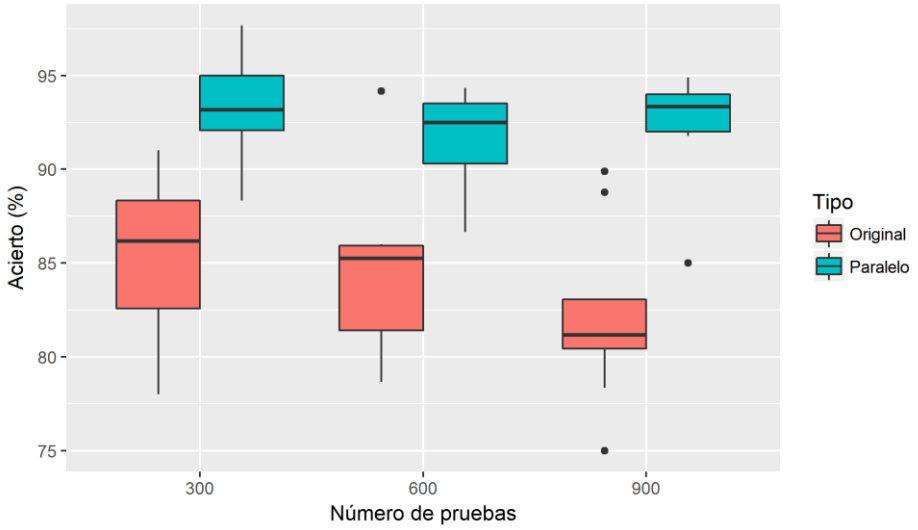


Figura 2 Porcentaje de aciertos de la versión de código secuencial y paralelizado en función al número de pruebas

Reto 1

Meta

Medir el desempeño de la red neuronal en función a las probabilidades asignadas a la generación de dígitos.

Desarrollo del código

Buscando cumplir con el objetivo de este reto, se agregaron tres iteraciones de la clase `for` en las cuales se varían las probabilidades del color negro, gris y blanco.

```
> t.pruebas<-200
> var.n<-seq(0.02,0.99,0.19)
> var.g<-seq(0.02,0.99,0.19)
> var.b<-seq(0.02,0.99,0.19)
> resultados<-data.frame()
>
> for (negro in var.n){
+   for (gris in var.g){
+     for (blanco in var.b){
+
+       source('~/GitHub/Simulacion/Simulacion/P12/Codigo/P12_T12.R')

+ proba<-cbind(negro,gris,blanco,acierto)
+ resultados<-rbind(resultados,proba)
+     }
+   }
+   }
```

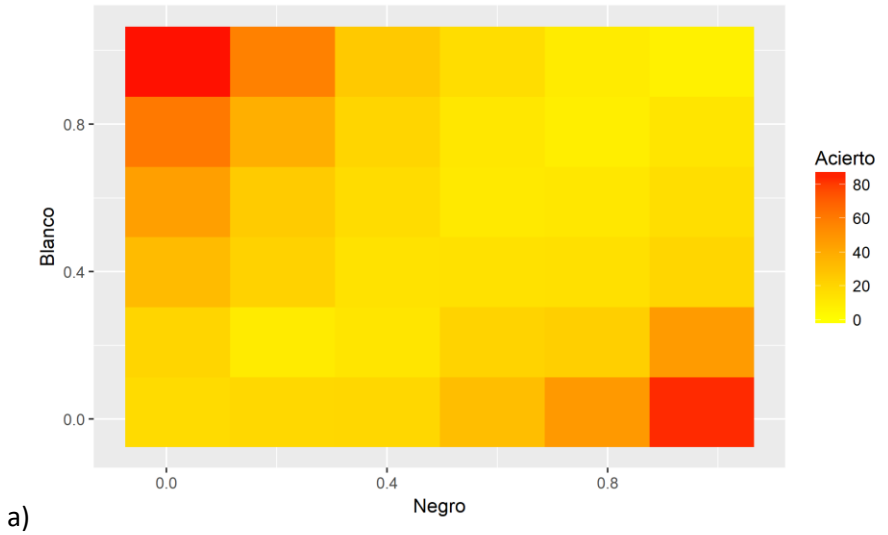
Para observar claramente la relación entre los distintos valores de las probabilidades de tres subconjuntos de valores (gris-blanco, gris-negro, blanco-negro) se decidió representarlos gráficamente con un mapa de calor.

```
> colnames(resultados)<-c("Negro","Gris","Blanco","Acierto")
>
> library(ggplot2)
> ggplot(resultados, aes(Blanco, Gris)) +
+   geom_raster(aes(fill=Acierto)) +
+   scale_fill_gradient(low="yellow", high="red")
> ggsave("Heat_GB.png")
Saving 6.83 x 3.96 in image
>
> ggplot(resultados, aes(Negro, Gris)) +
+   geom_raster(aes(fill=Acierto)) +
+   scale_fill_gradient(low="yellow", high="red")
> ggsave("Heat_GN.png")
Saving 6.83 x 3.96 in image
>
> ggplot(resultados, aes(Negro, Blanco)) +
+   geom_raster(aes(fill=Acierto)) +
+   scale_fill_gradient(low="yellow", high="red")
> ggsave("Heat_BN.png")
Saving 6.83 x 3.96 in image
```

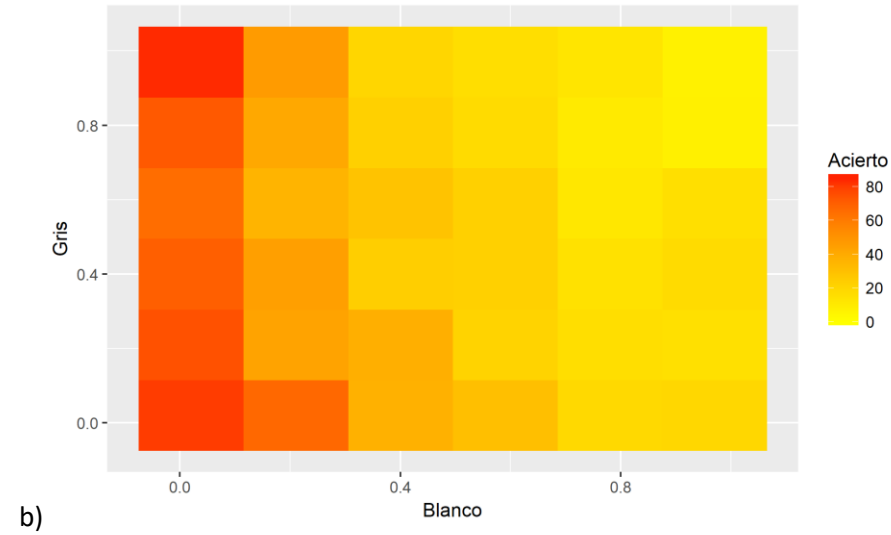
Resultados

La Figura 3 está compuesta por tres gráficos; en el primer gráfico se muestra el porcentaje de aciertos en función a las probabilidades del color negro y del color blanco, en el segundo lo hace en función de las probabilidades del color gris y blanco; por último, el tercer gráfico muestra dicho valor en función a la probabilidad de los colores negro y gris. Siguiendo la leyenda los cuadros rojos son aquellos que tuvieron un mayor porcentaje de aciertos, mientras que los cuadros amarillos obtuvieron un menor porcentaje de aciertos, esto permite formular algunas condiciones para maximizar el porcentaje de aciertos en la red neuronal.

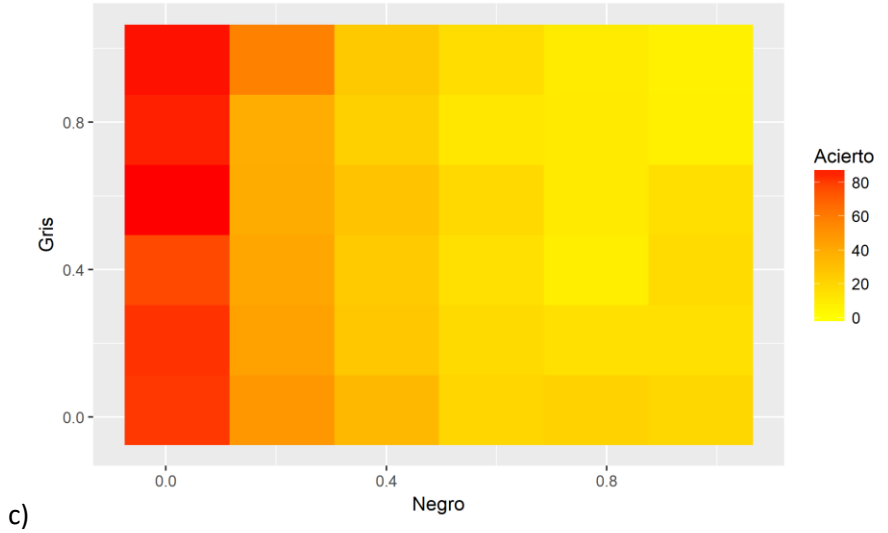
1. El valor de la probabilidad de los colores contrastantes (blanco y negro) debe ser contraria, es decir, si la probabilidad del color negro es alta, la probabilidad color blanco debe ser pequeña.
2. El valor de la probabilidad del color gris predomina ante los otros colores, es decir si se tienen combinaciones de color gris-blanco o gris-negro el porcentaje de aciertos tiene una mayor influencia de la probabilidad del color gris que el otro color.



a)



b)



c)

Figura 3 Mapas de calor de la variación de la probabilidad de colores (ngb)

Referencias

[1] Elisa.dyndns-web.com. (2017). P12 — R paralelo — Schaeffer. [online] Available at: http://elisa.dyndns-web.com/teaching/comp/par/p12.html [Accessed 31 Oct. 2017].