Laboratorio 3

Introducción

En este laboratorio utilizaremos el reconocimiento de caracteres manuscritos para poder visualizar las imágenes. Se iniciará realizando un análisis exploratorio para entender de mejor forma como se encuentran los datos y poder analizar qué algoritmos son la mejor opción a desarrollar.

Luego trabajaremos en un modelo de redes neuronales simples, se realizará un modelo de deep learning y un algoritmo el cual se considere sea la mejor opción para poder reconocer la imagen. Al final se determinará la efectividad de cada uno.

```
#descarga de los archivos
test <-read.csv("./test.csv", stringsAsFactors = TRUE)
train <- read.csv("./train.csv")

#visualizacion de variables
str(test)
str(train)
```

Análisis exploratorio

```
#Nombrando la base
labels <- train$label
train$label <- as.factor(train$label)

#plotting
colors = rainbow(length(unique(train$label)))
names(colors) = unique(train$label)

#ejecución del algoritmo
Rtsne(train[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
exeTimeTsne<- system.time(Rtsne(train[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500))

plot(tsne_out$Y,col=train$label)
```

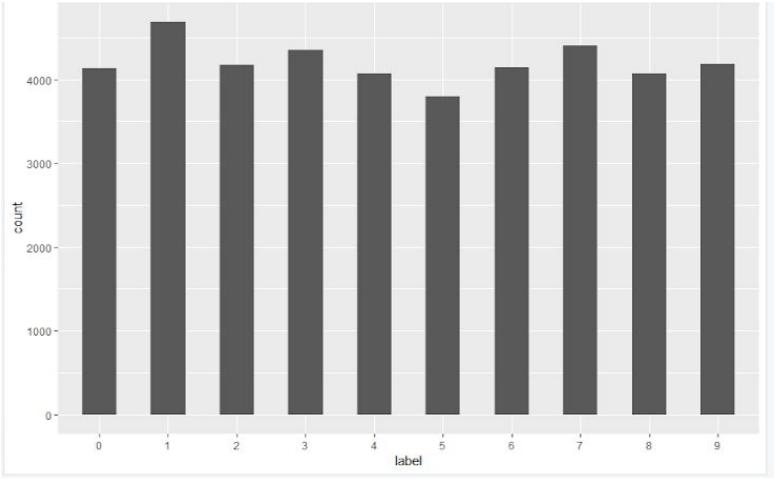
```
```{r gráfica de barras}
ggplot(train, aes(x=label)) + geom_bar(width = 0.5)
```
```

```
Read the 42000 x 50 data matrix successfully!
Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
Computing input similarities...
Normalizing input...
Building tree...
 - point 0 of 42000
 - point 10000 of 42000
 - point 20000 of 42000
 - point 30000 of 42000
 - point 40000 of 42000
Done in 700.82 seconds (sparsity = 0.002967)!
Learning embedding...
Iteration 50: error is 114.644309 (50 iterations in 93.16 seconds)
Iteration 100: error is 114.644280 (50 iterations in 89.81 seconds)
Iteration 150: error is 109.148620 (50 iterations in 92.36 seconds)
Iteration 200: error is 99.642178 (50 iterations in 81.97 seconds)
Iteration 250: error is 97.136314 (50 iterations in 86.31 seconds)
Iteration 300: error is 4.429755 (50 iterations in 81.65 seconds)
Iteration 350: error is 4.051957 (50 iterations in 75.98 seconds)
Iteration 400: error is 3.828852 (50 iterations in 77.44 seconds)
Iteration 450: error is 3.670195 (50 iterations in 77.11 seconds)
Iteration 500: error is 3.547836 (50 iterations in 76.60 seconds)
Fitting performed in 832.39 seconds.
$`theta`
[1] 0.5

$perplexity
[1] 30
```

Como primer paso se normaliza la matriz por medio de iteraciones, la normalización consta de tomar como 0 negro y 255 blanco en la matriz. Con este procedimiento se hace reducir la matriz según distancias euclidianas.

En este gráfico podemos observar la frecuencia con la que se obtienen los píxeles, es posible observar que no existe una diferencia significativa sin embargo el segundo pixel es el que más aparece en las imágenes a encontrar.

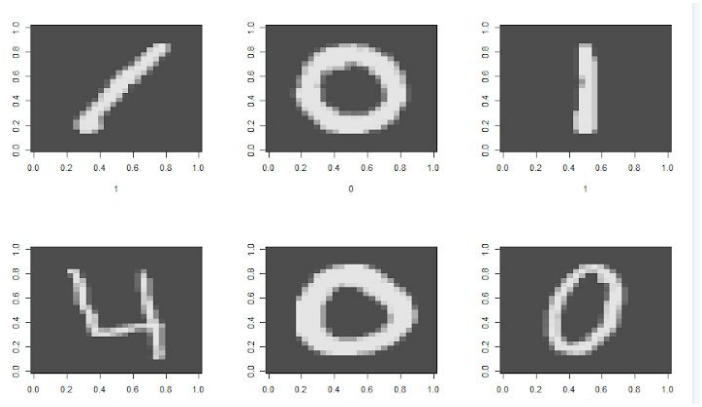


Discusión de Modelos.

Los el código de los modelos realizados está en el repositorio de github

Redes neuronales:

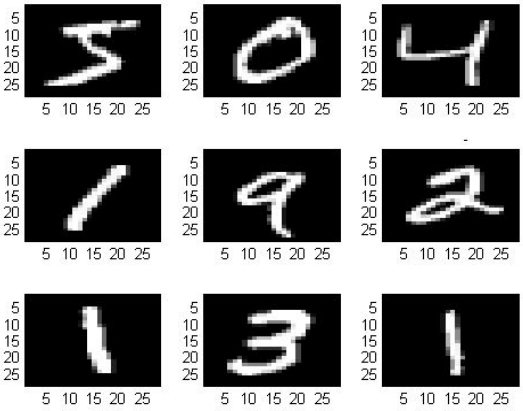
En los resultados de redes neuronales podemos obtener imágenes donde los caracteres se pueden reconocer fácilmente, aunque un poco pixelados y con baja resolución, el desempeño de este modelo fue lo calificamos como medio debido a que es fácil identificar los caracteres de las imágenes que muestra



user	system	elapsed
-0.09	-0.03	-0.11

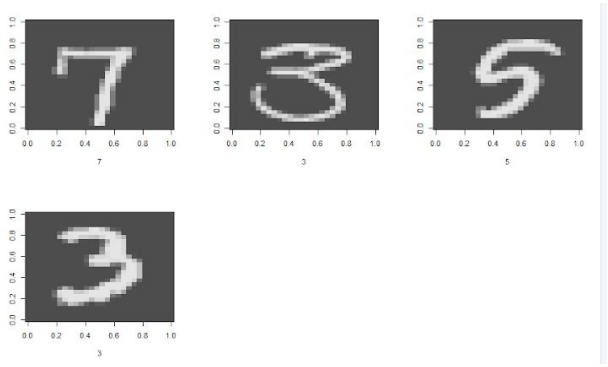
Deep learning:

En el resultado de deep learning obtuvimos con el modelo de deep learning, fueron imágenes con una mejor resolución aun un poco pixeleada y los caracteres más definidos, el desempeño de este modelo lo calificamos como alto debido a que los caracteres en cada imagen están más concretos



Convolutional neural network:

En este tercer modelo podemos obtener un resultado con una resolución un poco más baja pero con mejor definición de caracteres, mostrando un desempeño bastante alto ya que los caracteres en la imagen mucho más fáciles de reconocer.



Concluimos que nuestro último modelo mostró un mejor desempeño aun así sobre los demás igual los tres mostraban los caracteres de forma clara y dan resultado bastante claros en cada imagen.