UNIVERSIDAD DEL VALLE DE GUATEMALA
Data Science 1
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Laboratorio 3

Rodrigo Chávez - 14429
Daniel García - 14152
Anaís Castañeda- 17291

Agosto 2018

Descripción de los Datos

Los 784 datos son proporcionados por una imágen que tiene 255 píxeles; estos indican el color que hay en cada espacio. Se tienen como objetivo que el programa reconozca la imágen y esta sea asimilada con los valores que se encuentran en "label".

Para lograr dicho objetivo se realizaron diferentes pasos como deep learning, redes neuronales

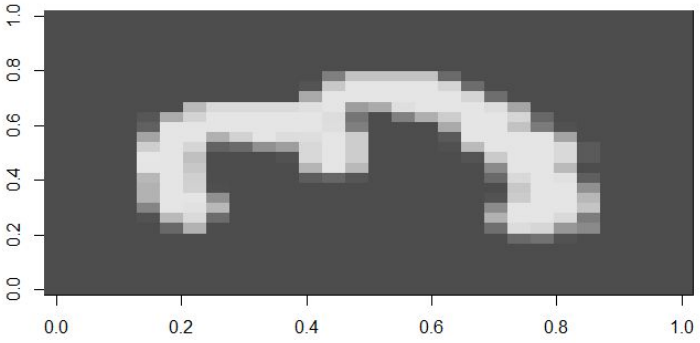
Análisis Exploratorio

Para comenzar en este laboratorio no se hizo una limpieza de los datos debido a que no existían "N/A" y limpios.

Para que los datos fueran entendidos y visualizados se utilizó el siguiente código:

```
#--------------------------- Graficación de Digitos --------------------------------#
# Se crea una matriz de 28*28 con los valores de los colores de los pixeles
m = matrix(unlist(train[10,-1]),nrow = 28,byrow = T)
# Luego se grafica la matriz, para verificar si el número aparece correctamente
image(m,col=grey.colors(255))
```

Ya que se tenía la imágen del número se notó que este necesitaba una rotación para que se pudiera ver la imágen en un aposición legible.



Para que estos tuvieran una mejor presentación se rotaron con el siguiente código

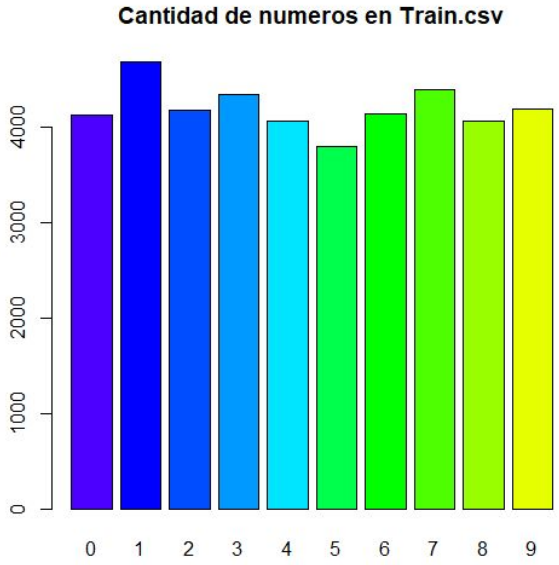
```
#Como la imagen aparece rotada, debemos de utilizar una función para que aparezca correctamente
rotar <- function(x) t(apply(x, 2, rev)) #rotación de la matriz
```

Luego, se tomó una muestra para verificar si los números están bien ploteados. Para eso se utilizó el siguiente código.

```r
#Se grafican varias imagenes y se rotan con la función anterior, para verificar si aparecen correctamente
par(mfrow=c(2,3))
lapply(1:10,
       function(x) image(
         rotar(matrix(unlist(train[x,-1]),nrow = 28,byrow = T)),
         col=grey.colors(255),
         xlab=train[x,1]
       )
)
#Luego, se regresan las opciones de ploteo a default
par(mfrow=c(1,1))
```

Con ello, se obtienen los siguientes resultados:

Se realizó una gráfica de barra con el propósito de visualizar la cantidad de números contenidos en el dataset.



Ya que los números aparecen correctamente, se procede a realizar los modelos.

Modelo de redes neuronales

El modelo de redes neuronales utilizado se implementó por medio de la librería "neuralnet" de R. En este caso la red neuronal recibe todos los píxeles como parámetros y se ingresan en 5 neuronas. El algoritmo se encarga de repetir los cálculos hasta que se llega a una precisión de 0.1 decimales, en este caso no se necesita más puesto que se desea encontrar valores enteros.

```
#------------------------------- Modelo Redes Neuronales ---------------------------------#
#Se crean sub dataframes con menos datos puesto que las 784 variables tardan bastante tiempo en calcularse
ind <- sample((1:28000), 1000)
sampleTrain <- train[ind,]

#se toman todos los nombres de las columnas y se ingresan como funcion en la redneuronal
nombres<-colnames(sampleTrain[2:785])
nombresEnFormula<-paste(nombres,collapse = "+")

#se calcula la red neuronal con 5 neuronas
form=as.formula(paste("label~",nombresEnFormula,collapse="+"))
RedNeuronal <- neuralnet(formula =  form,
                         data = sampleTrain, hidden = 5)

#se parte el dataset de test de la misma forma que el train, asi se puede comparar resultados obtenidos
sampleTest <- test[ind,]

#se realiza la prediccion
RedResultados <- compute(RedNeuronal,sampleTest)

#se crea una tabla para visualizar los resultados.
TablaResultados <- data.frame(actual = sampleTrain$label, prediccion = round(RedResultados$net.result))
```

Los resultados se imprimen en una tabla donde se tiene el valor verdadero del número y la predicción obtenida de la red neuronal.

	actual	prediccion
26605	2	5
21832	9	2
20913	5	2
2258	4	2
1287	2	5
2768	1	5
13389	9	0
9513	6	2
3741	0	5
4420	5	5
15545	2	0
12971	7	7
13950	5	5
22983	7	5
22050	8	7
4440	1	2
2829	2	2
14554	4	0
19625	7	7
18298	9	7

Deep learning

Se quiere probar si la máquina será buena reconociendo los números con ayuda del código. Se utilizó la librería "h2o" para realizar el modelo de Deep Learning y el conjunto se entrenó con los siguientes parámetros:

```
#Se entrena el modelo, ya que tiene muchos argumentos, se separan en líneas
model =
h2o.deeplearning(x = 2:785,  #Número de columnas para predecir
y = 1,   #Número de columna para el label
training_frame = train_h2o, #Los datos en formato h2o
activation = "RectifierWithDropout", #Algoritmo a utilizar
input_dropout_ratio = 0.2,
hidden_dropout_ratios = c(0.5,0.5),
balance_classes = TRUE,
hidden = c(100,100), #Dos layers de 100 nodos
momentum_stable = 0.99,
nesterov_accelerated_gradient = T, #Se utiliza para agilizar el algoritmo
epochs = 15)
```

matriz de confusión

```
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
           0    1    2   3   4   5    6    7   8   9  Error            Rate
0       1041    0    5   0   0   2    9    1   2   1 0.0189 =   20 / 1,061
1          0  997    7   3   1   1    2    4   4   0 0.0216 =   22 / 1,019
2          2    0  941   4   3   2    9   11   2   1 0.0349 =      34 / 975
3          2    2   27 936   0  29    2   15   4   7 0.0859 =   88 / 1,024
4          0    1    7   0 961   0   14    3   4  17 0.0457 =   46 / 1,007
5          3    0    9   8   4 910   10    2   2   2 0.0421 =      40 / 950
6          4    1   14   0   1   7  975    0   0   0 0.0269 =   27 / 1,002
7          3    5   18   2   2   0    0 1021   1   3 0.0322 =   34 / 1,055
8          2    6   12   7   0   4    6    5 949   3 0.0453 =      45 / 994
9          2    2   11   7   9   4    0   29   3 937 0.0667 =   67 / 1,004
Totals  1059 1014 1051 967 981 959 1027 1091 971 971 0.0419 = 423 / 10,091
```

Ya que se obtuvo la matriz de confusión (en donde se puede observar que tantas veces hay un error entre un factor y otro) se obtuvo un porcentaje de error, en este caso fue bastante bajo 4.20%.

También se obtuvo la cantidad total de segundos que se tardó el código en realizar deep learning. Obteniendo una medida de eficacia para luego poder ser comparada con el modelo de redes neuronales.

```
 user  system elapsed
-1.02   -0.03 -136.68
```

El resultado final fue un csv con los labels de la imágen.

Muestra del csv

ImageId	Label
1	2
2	0
3	9
4	9
5	3
6	7
7	0
8	3
9	0
10	3
11	5
12	7
13	4
14	0

Otro algoritmo

El segundo algoritmo utilizado fue una red neuronal la cual pasó por un pre proceso de acondicionamiento de datos y luego se utilizó otra librería para la creación del modelo ("EXtreme Gradient Boosting Training").

Este algoritmo toma los datos y realiza operaciones de álgebra lineal para reducir las imagenes a vectores simples los cuales ocupan un tamaño mucho menor al original. De esta forma se reducen las imágenes de 784 variables a 253.

Al obtener menos variables en el dataframe, se pudieron utilizar 28000 datos para entrenar al nuevo modelo. Este modelo tuvo un tiempo de generación de 2 minutos y 15 segundos. Utilizando 50 iteraciones en 10 neuronas.

La precisión de este algoritmo es de 97%.

Comparación de algoritmos y conclusión

Modelo de redes neuronales	**Deep learning**	**Otro algoritmo**
La red neuronal implementada resultó tener un bajo desempeño en cuanto a la predicción, muy pocos resultados fueron acertados. El tiempo de respuesta fue muy lento, puesto que sobrepasó los 3 minutos con 20 segundos tomando solo 1000 datos de toda la muestra.	Se obtuvo un porcentaje de error del 4.20%. Cantidad total de segundos que se tardó el código en realizar deep learning 136.68seg.	El porcentaje de error es del 3% aproximadamente. El tiempo de respuesta del algoritmo fue de 2 minutos y 15 segundos, realizando 50 iteraciones en 10 neuronas. Requirió de un preprocesamiento de imágenes y utilizó el total de datos disponibles.
Conclusión: El algoritmo de EXtreme Gradient Boosting Training resultó ser más preciso que el algoritmo Deep Learning de H2O por una leve diferencia del 1%. Sin embargo el algoritmo de deep learning es mucho más sencillo de implementar puesto que no necesita de ningún preprocesamiento de los datos. La red neuronal simple utilizando neuralnet debe de ser ignorada puesto que no es nada eficiente, toma pocos datos y el tiempo de procesamiento es sumamente alto. Esta red es funcional para modelos mucho más simples de menos variables.		