Project 2
Classification of MNIST data

Using Neural Network in H2O cluster

Abbas Navsariwala
Deenadayalamuthu
Rohan Thorat

Table of Contents

Introduction:

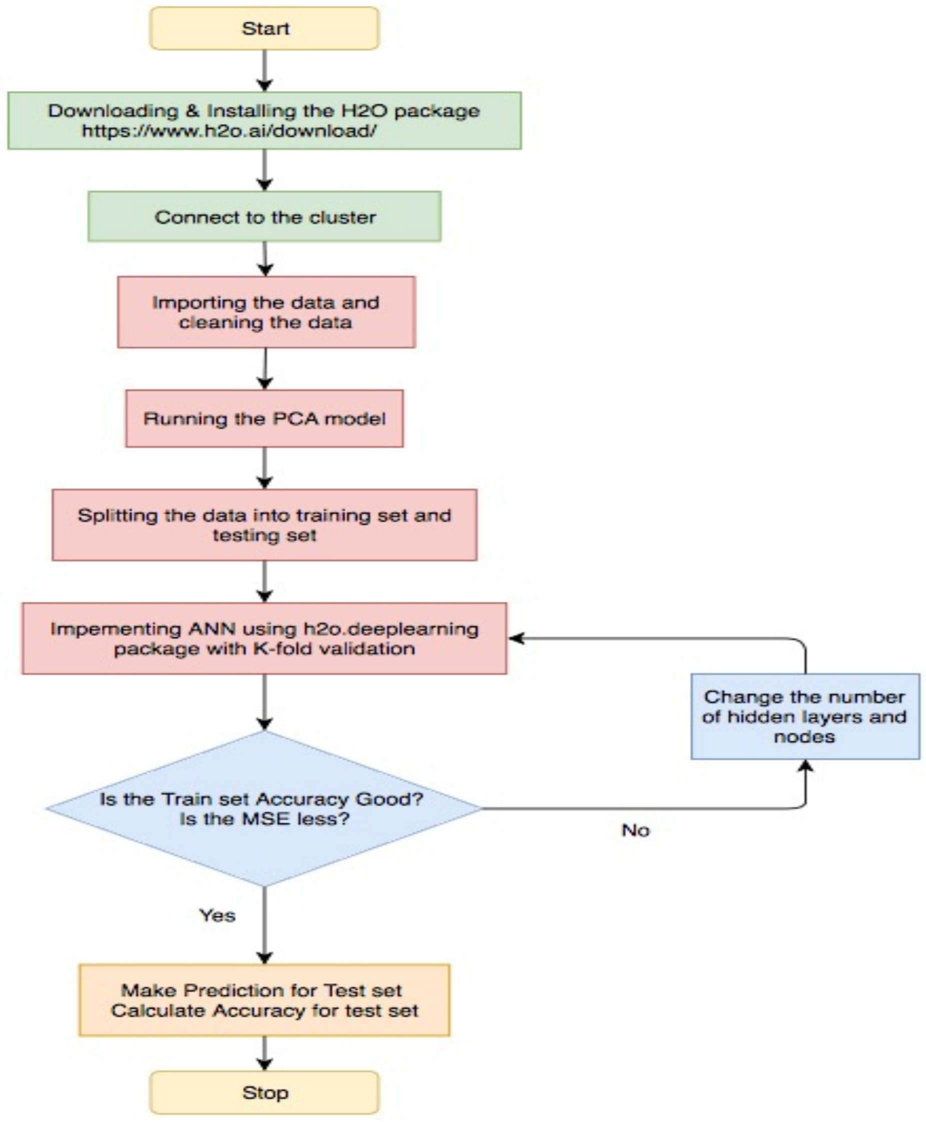
In this project our task was to classify handwritten digits in the mnist dataset using Artificial Neural Network (ANN). The 'mnist' dataset contains 60000 observations and 785 features. The first column contains actual hand written digit and the rest of the features contain greyscale index of the digits in the range of 0 to 255.

For the execution of Artificial Neural Network (ANN) we used H2O deep learning cluster for quicker execution of large dataset.

Summary of the Project:

After the successful execution of Neural Network model using K-fold cross validation in the H2O cluster, predictions based upon the model built achieved 94.84858 % for trained set and 93.48329 % for test set.

Flow chart:


Start
Downloading & Installing the H2O package https://www.h2o.ai/download/
Connect to the cluster
Importing the data and cleaning the data
Running the PCA model
Splitting the data into training set and testing set
Impementing ANN using h2o.deeplearning package with K-fold validation
Change the number of hidden layers and nodes
Is the Train set Accuracy Good? Is the MSE less?
No
Yes
Make Prediction for Test set Calculate Accuracy for test set
Stop

Code:

```
library(data.table)
## Load H2O library
library(h2o)

## Connect to H2O cluster
h2o.init(nthreads = -1)

## Define data paths
base_path = normalizePath("/Users/rohan/Desktop/MLP2")
mnist_path = paste0(base_path,"/mnist.csv")

## Ingest data
mnist.hex = h2o.importFile(path = mnist_path,destination_frame = "mnist.hex",header=NA)
dim(mnist.hex)

## Converting h2o hex file to a data frame
mnist.data<-as.data.frame(mnist.hex)
dim(mnist.data)

## Removing first column from the data frame
mnist.data.modified<-mnist.data[,-1]
dim(mnist.data.modified)

## Running the PCA model
pca.model<-prcomp(mnist.data.modified)

## Value of the rotated data is taken
mnist_pca_x<-as.data.frame(pca.model$x[,1:261])
dim(mnist_pca_x)
## View(head(mnist_pca_x))

## Combining the data
mnist.data.complete <- as.data.frame(cbind(mnist.data[,1],mnist_pca_x))
##View(head(mnist.data.complete))

## Naming the first column
colnames(mnist.data.complete)[1] <- "C1"

## converting the data frame back to h2o hex file
mnist.hex<-as.h2o(mnist.data.complete,destination_frame = "mnist.hex")

## Splitting the data into two sets 1st set = 90% and second set = 10%
mnist.data.split <- h2o.splitFrame(data=mnist.hex, ratios=0.90)
```

```r
## Checking the dimensions of the split data
print(dim(mnist.data.split[[1]]))
print(dim(mnist.data.split[[2]]))

## Splitting the data and giving it names
train <- mnist.data.split[[1]]
## train
test <- mnist.data.split[[2]]
## test

## Specify the response and predictor columns
y<-"C1"
## View(y)
## Removing the response variable from
x<-setdiff(names(train),y)
## View(x)

## Encode the response column as categorical for multinomial classification
train[,y] <- as.factor(train[,y])
test[,y] <- as.factor(test[,y])

## Perform 10-fold cross-validation on training data frame using K-fold
system.time(
train_model <- h2o.deeplearning(
  x = x,
  y = y,
  training_frame = train,
  distribution = "multinomial",
  activation = "RectifierWithDropout",
  hidden = c(100,100,100),
  input_dropout_ratio = 0.2,
  sparse = TRUE,
  l1 = 1e-5,
  epochs = 10,
  nfolds = 10)
)

## View specified parameters of the deep learning model
train_model@parameters

## Examine the performance of the trained model
## display all performance metrics
h2o.performance(train_model) # training metrics

## Cross validated Mean Square Error
h2o.mse(train_model, xval = TRUE)

## Obtaining the variable importance
head(as.data.table(h2o.varimp(train_model)))
```

```
## validation accuracy
h2o.hit_ratio_table(train_model)[1, 2]

## Classify the test set (predict class labels)
## This also returns the probability for each class
pred <- h2o.predict(train_model, newdata = test)
list(Predicted_model_Accuracy_Test = mean(pred$predict == test$C1))
```

Establishing Connection to H2O cluster:

```
> library(data.table)
> ## Load H2O library
> library(h2o)
> ## Connect to H2O cluster
> h2o.init(nthreads = -1)
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:          24 minutes 15 seconds
    H2O cluster timezone:        America/New_York
    H2O data parsing timezone:   UTC
    H2O cluster version:         3.19.0.4278
    H2O cluster version age:     2 days
    H2O cluster name:            H2O_started_from_R_rohan_zfv829
    H2O cluster total nodes:     1
    H2O cluster total memory:    0.54 GB
    H2O cluster total cores:     4
    H2O cluster allowed cores:   4
    H2O cluster healthy:         TRUE
    H2O Connection ip:           localhost
    H2O Connection port:         54321
    H2O Connection proxy:        NA
    H2O Internal Security:       FALSE
    H2O API Extensions:          XGBoost, Algos, AutoML, Core V3, Core V4
    R Version:                   R version 3.4.1 (2017-06-30)

> ## Define data paths
> base_path = normalizePath("/Users/rohan/Desktop/MLP2")
> mnist_path = paste0(base_path,"/mnist.csv")
> ## Ingest data
> mnist.hex = h2o.importFile(path = mnist_path,destination_frame = "mnist.hex",header=NA)
  |=============================================| 100%
```

Output:

```
> ## View specified parameters of the deep learning model
> train_model@parameters
$model_id
[1] "DeepLearning_model_R_1525184243715_7"

$training_frame
[1] "RTMP_sid_b0d3_21"

$nfolds
[1] 10

$overwrite_with_best_model
[1] FALSE

$activation
[1] "RectifierWithDropout"

$hidden
[1] 100 100 100

$epochs
[1] 10.17908

$seed
[1] -8.021453e+18

$input_dropout_ratio
[1] 0.2

$l1
[1] 1e-05

$distribution
[1] "multinomial"

$stopping_rounds
[1] 0

$sparse
[1] TRUE
```

```
$x
  [1] "PC1"   "PC2"   "PC3"   "PC4"   "PC5"   "PC6"  
  [7] "PC7"   "PC8"   "PC9"   "PC10"  "PC11"  "PC12" 
 [13] "PC13"  "PC14"  "PC15"  "PC16"  "PC17"  "PC18" 
 [19] "PC19"  "PC20"  "PC21"  "PC22"  "PC23"  "PC24" 
 [25] "PC25"  "PC26"  "PC27"  "PC28"  "PC29"  "PC30" 
 [31] "PC31"  "PC32"  "PC33"  "PC34"  "PC35"  "PC36" 
 [37] "PC37"  "PC38"  "PC39"  "PC40"  "PC41"  "PC42" 
 [43] "PC43"  "PC44"  "PC45"  "PC46"  "PC47"  "PC48" 
 [49] "PC49"  "PC50"  "PC51"  "PC52"  "PC53"  "PC54" 
 [55] "PC55"  "PC56"  "PC57"  "PC58"  "PC59"  "PC60" 
 [61] "PC61"  "PC62"  "PC63"  "PC64"  "PC65"  "PC66" 
 [67] "PC67"  "PC68"  "PC69"  "PC70"  "PC71"  "PC72" 
 [73] "PC73"  "PC74"  "PC75"  "PC76"  "PC77"  "PC78" 
 [79] "PC79"  "PC80"  "PC81"  "PC82"  "PC83"  "PC84" 
 [85] "PC85"  "PC86"  "PC87"  "PC88"  "PC89"  "PC90" 
 [91] "PC91"  "PC92"  "PC93"  "PC94"  "PC95"  "PC96" 
 [97] "PC97"  "PC98"  "PC99"  "PC100" "PC101" "PC102"
[103] "PC103" "PC104" "PC105" "PC106" "PC107" "PC108"
[109] "PC109" "PC110" "PC111" "PC112" "PC113" "PC114"
[115] "PC115" "PC116" "PC117" "PC118" "PC119" "PC120"
[121] "PC121" "PC122" "PC123" "PC124" "PC125" "PC126"
[127] "PC127" "PC128" "PC129" "PC130" "PC131" "PC132"
[133] "PC133" "PC134" "PC135" "PC136" "PC137" "PC138"
[139] "PC139" "PC140" "PC141" "PC142" "PC143" "PC144"
[145] "PC145" "PC146" "PC147" "PC148" "PC149" "PC150"
[151] "PC151" "PC152" "PC153" "PC154" "PC155" "PC156"
[157] "PC157" "PC158" "PC159" "PC160" "PC161" "PC162"
[163] "PC163" "PC164" "PC165" "PC166" "PC167" "PC168"
[169] "PC169" "PC170" "PC171" "PC172" "PC173" "PC174"
[175] "PC175" "PC176" "PC177" "PC178" "PC179" "PC180"
[181] "PC181" "PC182" "PC183" "PC184" "PC185" "PC186"
[187] "PC187" "PC188" "PC189" "PC190" "PC191" "PC192"
[193] "PC193" "PC194" "PC195" "PC196" "PC197" "PC198"
[199] "PC199" "PC200" "PC201" "PC202" "PC203" "PC204"
[205] "PC205" "PC206" "PC207" "PC208" "PC209" "PC210"
[211] "PC211" "PC212" "PC213" "PC214" "PC215" "PC216"
[217] "PC217" "PC218" "PC219" "PC220" "PC221" "PC222"
[223] "PC223" "PC224" "PC225" "PC226" "PC227" "PC228"
[229] "PC229" "PC230" "PC231" "PC232" "PC233" "PC234"
[235] "PC235" "PC236" "PC237" "PC238" "PC239" "PC240"
[241] "PC241" "PC242" "PC243" "PC244" "PC245" "PC246"
[247] "PC247" "PC248" "PC249" "PC250" "PC251" "PC252"
[253] "PC253" "PC254" "PC255" "PC256" "PC257" "PC258"
[259] "PC259" "PC260" "PC261"

$y
[1] "C1"
```

```
> ## Examine the performance of the trained model model
  # display all performance metrics
> h2o.performance(train_model) # training metrics
H2OMultinomialMetrics: deeplearning
** Reported on training data. **
** Metrics reported on temporary training frame with 9939 samples **
```

Training Set Metrics:

```
Training Set Metrics:
=====================

MSE: (Extract with `h2o.mse`) 0.04809453
RMSE: (Extract with `h2o.rmse`) 0.2193047
Logloss: (Extract with `h2o.logloss`) 0.1781372
Mean Per-Class Error: 0.05196243
Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train = TRUE)`)
```

Confusion Matrix:

```
=========================================================================
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
       0     1     2     3     4     5     6     7     8     9
0      990   0     2     3     1     8     7     2     2     0
1      3     1133  7     2     3     2     1     1     11    2
2      4     1     932   11    7     0     10    2     18    1
3      3     0     24    897   1     19    1     3     12    4
4      1     2     6     0     913   0     5     0     7     21
5      7     1     5     14    5     810   2     1     6     9
6      4     1     5     0     1     12    956   0     3     0
7      11    10    9     3     14    2     0     960   2     31
8      6     4     7     13    4     16    6     0     915   6
9      6     0     0     8     28    5     1     15    9     921
Totals 1035  1152  997   951   977   874   989   984   985   995
       Error       Rate
0      0.0246      =     25 / 1,015
1      0.0275      =     32 / 1,165
2      0.0548      =     54 / 986
3      0.0695      =     67 / 964
4      0.0440      =     42 / 955
5      0.0581      =     50 / 860
6      0.0265      =     26 / 982
7      0.0787      =     82 / 1,042
8      0.0635      =     62 / 977
9      0.0725      =     72 / 993
Totals 0.0515      =     512 / 9,939
```

```
Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>,train = TRUE)`
=======================================================================
Top-10 Hit Ratios:
         k       hit_ratio
1        1       0.948486
2        2       0.980280
3        3       0.990341
4        4       0.994668
5        5       0.996881
6        6       0.997485
7        7       0.997887
8        8       0.999094
9        9       0.999598
10       10      1.000000

> ## Cross validated Mean Square Error
> h2o.mse(train_model, xval = TRUE)
[1] 0.06354254

> ## Obtaining the variable importance
> head(as.data.table(h2o.varimp(train_model)))
   variable    relative_importance      scaled_importance      percentage
1:      PC2     1.0000000               1.0000000              0.02724315
2:      PC4     0.8118187               0.8118187              0.02211650
3:      PC5     0.7173519               0.7173519              0.01954293
4:      PC7     0.6056777               0.6056777              0.01650057
5:      PC6     0.5865965               0.5865965              0.01598074
6:      PC8     0.5720177               0.5720177              0.01558357
```

Training dataset Accuracy:

```
> ## validation accuracy
> h2o.hit_ratio_table(train_model)[1, 2]
[1] 0.9484858
```

Test dataset Accuracy:

```
> ## Classify the test set (predict class labels)
> ## This also returns the probability for each class
> pred <- h2o.predict(train_model, newdata = test)
  |=========================================| 100%
> list(Predicted_model_Accuracy_Test = mean(pred$predict == test$C1))
$Predicted_model_Accuracy_Test
[1] 0.9348329
```