digit processing

Shinjini

12 December 2017

INTRODUCTION

In this competition, your goal is to correctly identify digits from a dataset of tens of thousands of handwritten images. We've curated a set of tutorial-style kernels which cover everything from regression to neural networks. We encourage you to experiment with different algorithms to learn first-hand what works well and how techniques compare.

Data Description

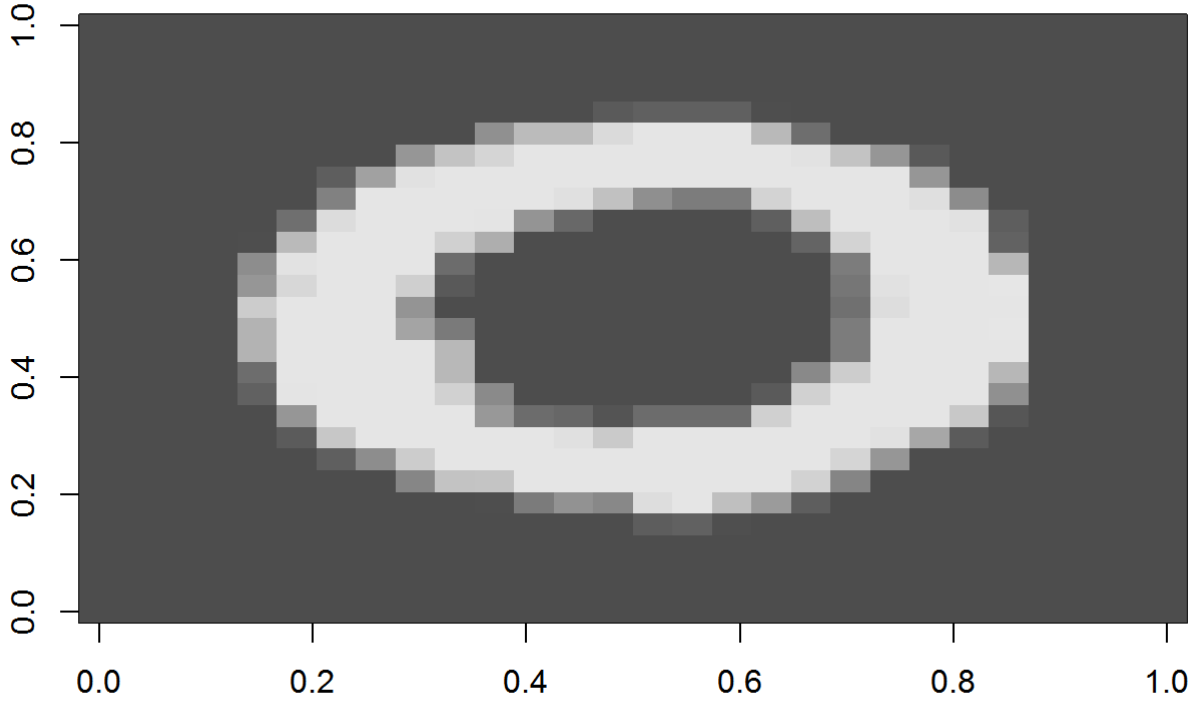
- The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine.
- Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.
- The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.
- Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i * 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero)

Import Data

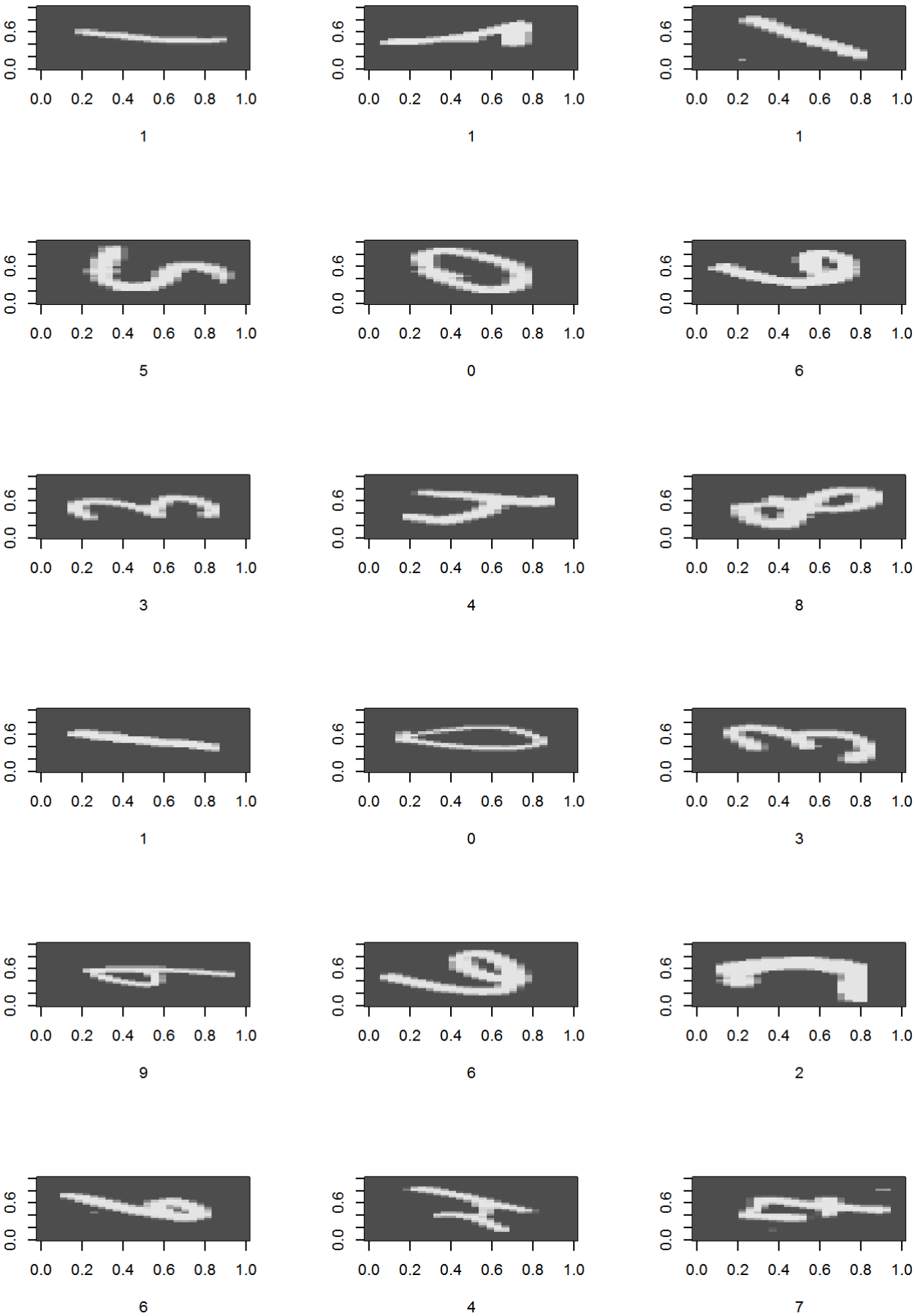
```
digit.test=read.csv("C://Users//Administrator//Desktop//KAGGLE//DIGIT PROCESSING//test.csv")
digit.train=read.csv("C://Users//Administrator//Desktop//KAGGLE//DIGIT PROCESSING//train.csv"
)
head.train= head(digit.train)
```

View the data

```
# Create a 28*28 matrix with pixel color values
m = matrix(unlist(digit.train[2,-1]), nrow = 28, byrow = TRUE)
# Plot that matrix
image(m,col=grey.colors(255))
```



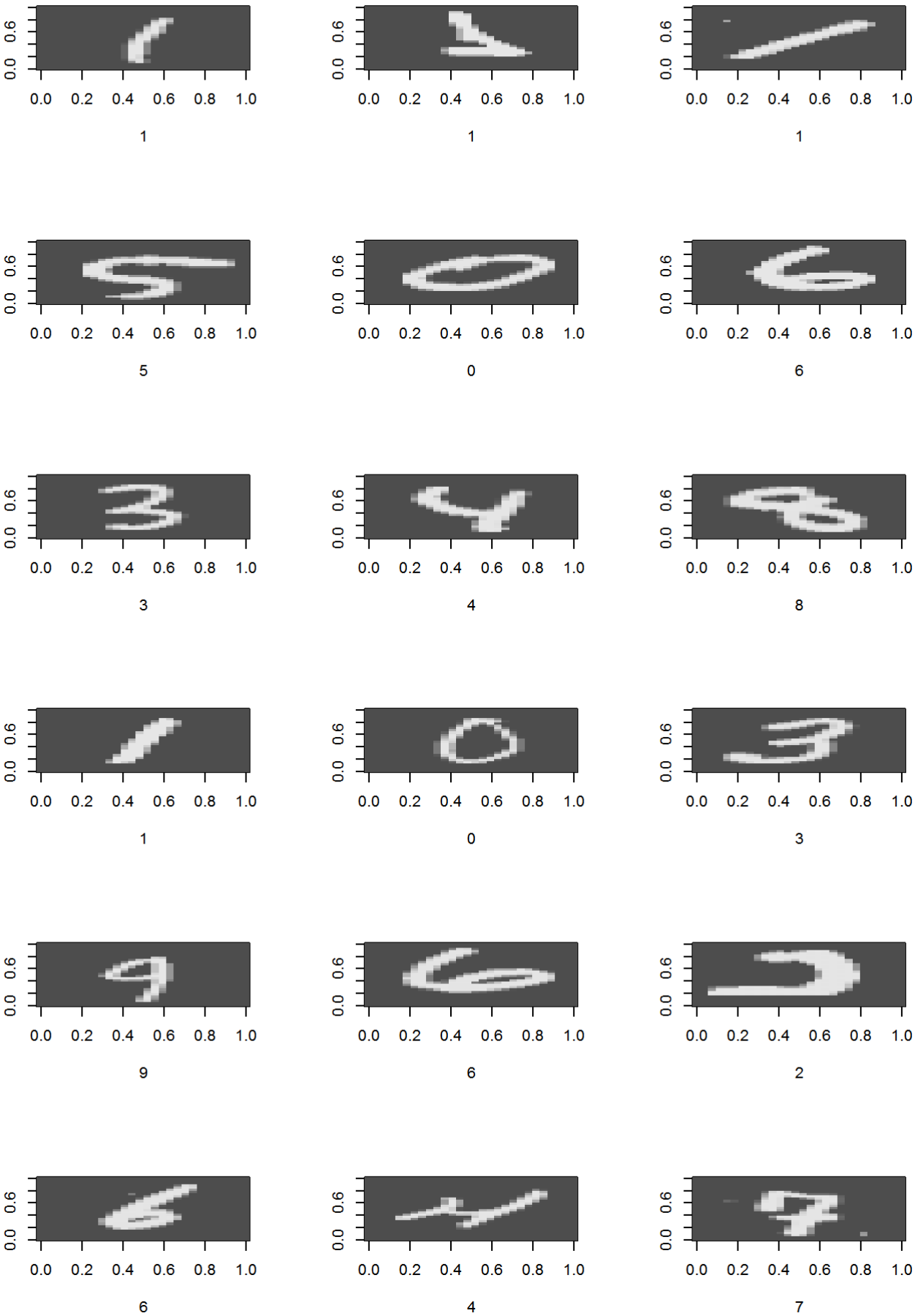

```
# Plot some of images
par(mfrow=c(3,3))
lapply(60:77,
       function(x) image(
         (matrix(unlist(digit.train[x,-1]),nrow = 28, byrow = TRUE)),
         col=grey.colors(255),
         xlab=digit.train[x,1]
       )
)
```


1
1
1
5
0
6
3
4
8
1
0
3
9
6
2
6
4
7

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
```

Notice that we need to rotate the images

```
# reverses (rotates the matrix)
par(mfrow=c(3,3))
rotate <- function(x) t(apply(x, 2, rev))
lapply(60:77,
       function(x) image(
         rotate(matrix(unlist(digit.train[x,-1]),nrow = 28, byrow = TRUE)),
         col=grey.colors(255),
         xlab=digit.train[x,1]
       )
)
```

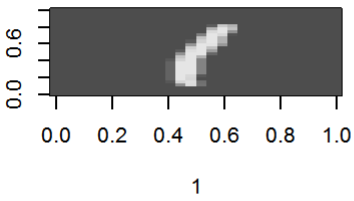

1
1
1
5
0
6
3
4
8
1
0
3
9
6
2
6
4
7

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
```

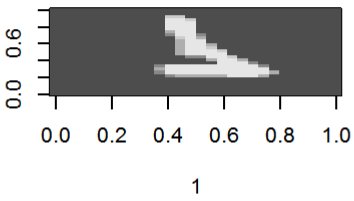
Step1:Normalization

The pixel values are gray scale between 0 and 255. It is almost always a good idea to perform some scaling of input values when using neural network models. Because the scale is well known and well behaved, we can very quickly normalize the pixel values to the range 0 and 1 by dividing each value by the maximum of 255.

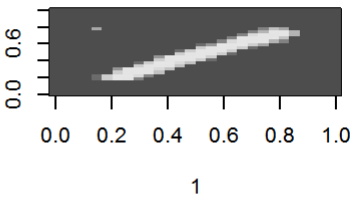
```
train.normal=digit.train
train.normal$label=as.factor(train.normal$label)
i=1
for(i in 2:30)
{
train.normal[,i]=(train.normal[,i]/255)
}

par(mfrow=c(3,3))
rotate <- function(x) t(apply(x, 2, rev))
lapply(60:77,
       function(x) image(
         rotate(matrix(unlist(train.normal[x,-1]),nrow = 28, byrow = TRUE)),
         col=grey.colors(255),
         xlab=train.normal[x,1]
       )
)
```



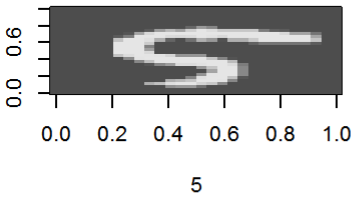
1



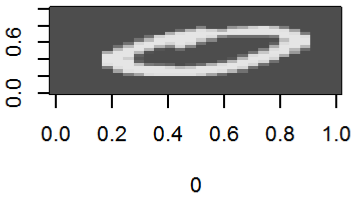
1



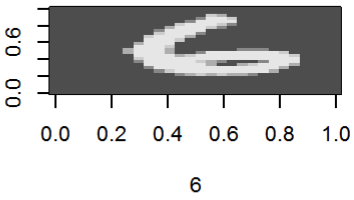
1



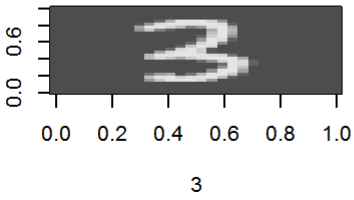
5



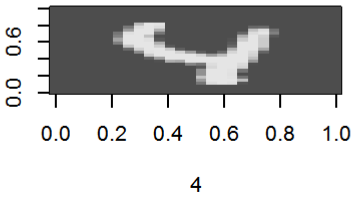
0



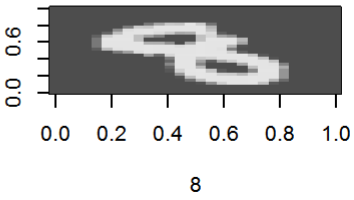
6



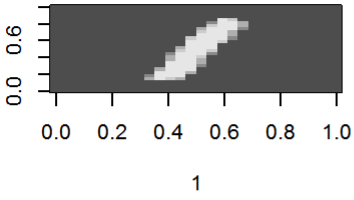
3



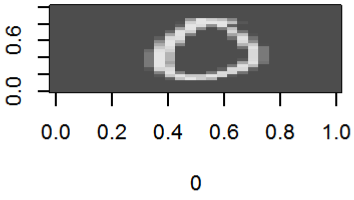
4



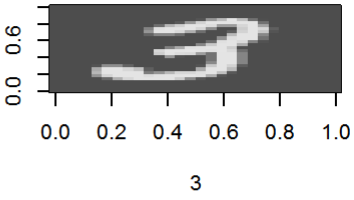
8



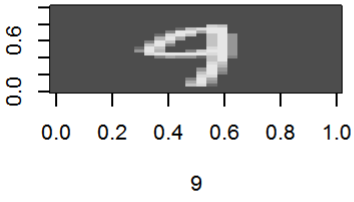
1



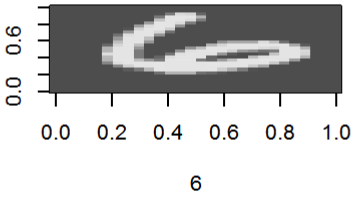
0



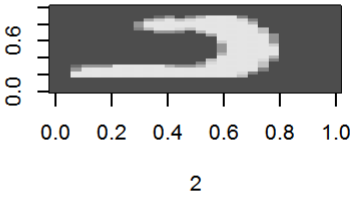
3



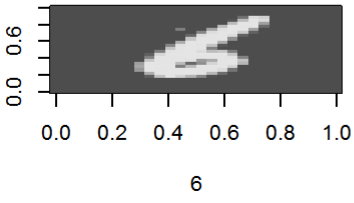
9



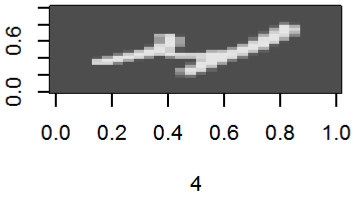
6



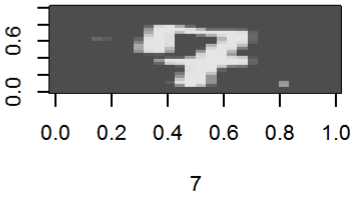
2



6



4



7

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
```

Step 2: Separate the dataset to 80% for training and 20% for testing

```
indx=sample.int(nrow(train.normal),size = (0.8*nrow(train.normal)))
train= train.normal[indx,]
test=train.normal[-indx,]
```

```
library(h2o)
```

```
## Warning: package 'h2o' was built under R version 3.4.3
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
#start a local h2o cluster
local.h2o <- h2o.init(ip = "localhost", port = 54321, startH2O = TRUE, nthreads=-1)
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\ADMINI~1\AppData\Local\Temp\RtmpcjlqmH/h2o_Administrator_started_from_r.out
##     C:\Users\ADMINI~1\AppData\Local\Temp\RtmpcjlqmH/h2o_Administrator_started_from_r.err
##
##
## Starting H2O JVM and connecting: . Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 seconds 399 milliseconds
##     H2O cluster version:        3.16.0.2
##     H2O cluster version age:    12 days
##     H2O cluster name:           H2O_started_from_R_Administrator_ntx209
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.76 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:         Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.4.2 (2017-09-28)
```

```
# pass dataframe from inside of the R environment to the H2O instance
trData<-as.h2o(train)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
tsData<-as.h2o(test)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
res.dl <- h2o.deeplearning(x = 2:785, y = 1, trData, activation = "Tanh", hidden=rep(160,5),e
pochs = 20)
```

```
## Warning in .h2o.startModelJob(algo, params, h2oRestApiVersion): Dropping bad and constant
columns: [pixel729, pixel644, pixel645, pixel448, pixel727, pixel728, pixel560, pixel52, pixe
l364, pixel760, pixel10, pixel54, pixel167, pixel53, pixel168, pixel56, pixel11, pixel55, pix
el57, pixel16, pixel18, pixel17, pixel19, pixel754, pixel755, pixel756, pixel757, pixel758, p
ixel759, pixel83, pixel196, pixel82, pixel85, pixel671, pixel84, pixel111, pixel672, pixel11
2, pixel673, pixel113, pixel476, pixel392, pixel700, pixel701, pixel141, pixel780, pixel30, p
ixel781, pixel782, pixel420, pixel783, pixel31, pixel421, pixel224, pixel588, pixel140, pixel
699, pixel139, pixel8, pixel9, pixel616, pixel6, pixel617, pixel7, pixel4, pixel5, pixel2, pi
xel3, pixel0, pixel21, pixel1, pixel20, pixel23, pixel532, pixel730, pixel22, pixel731, pixel
25, pixel24, pixel27, pixel26, pixel29, pixel28].
```

```
##
  |
  |                                                                  |   0%
  |
  |=                                                                 |   1%
  |
  |=                                                                 |   2%
  |
  |==                                                                |   3%
  |
  |===                                                               |   4%
  |
  |===                                                               |   5%
  |
  |====                                                              |   6%
  |
  |====                                                              |   7%
  |
  |=====                                                             |   7%
  |
  |=====                                                             |   8%
  |
  |======                                                            |   9%
  |
  |=======                                                           |  10%
  |
  |=======                                                           |  11%
  |
  |========                                                          |  12%
  |
  |========                                                          |  13%
  |
  |=========                                                         |  13%
  |
  |=========                                                         |  14%
  |
  |==========                                                        |  15%
  |
  |==========                                                        |  16%
  |
  |===========                                                       |  16%
  |
  |===========                                                       |  17%
  |
  |============                                                      |  18%
  |
  |============                                                      |  19%
  |
  |=============                                                     |  19%
  |
  |=============                                                     |  20%
  |
  |==============                                                    |  21%
  |
  |==============                                                    |  22%
  |
  |===============                                                   |  22%
```

```
|
|================                                                 |  23%
|
|=================                                                |  24%
|
|=================                                                |  25%
|
|==================                                               |  25%
|
|==================                                               |  26%
|
|==================                                               |  27%
|
|===================                                              |  27%
|
|===================                                              |  28%
|
|====================                                             |  29%
|
|====================                                             |  30%
|
|====================                                             |  31%
|
|=====================                                            |  32%
|
|=====================                                            |  33%
|
|======================                                           |  33%
|
|======================                                           |  34%
|
|======================                                           |  35%
|
|=======================                                          |  35%
|
|=======================                                          |  36%
|
|=======================                                          |  36%
|
|========================                                         |  37%
|
|=========================                                        |  38%
|
|==========================                                       |  39%
|
|==========================                                       |  40%
|
|==========================                                       |  41%
|
|==========================                                       |  41%
|
|==========================                                       |  42%
|
|===========================                                      |  43%
|
|============================                                     |  44%
|
|============================                                     |  45%
```

```
  |
  |==============================                                   |  46%
  |
  |==============================                                   |  47%
  |
  |==============================                                   |  47%
  |
  |==============================                                   |  48%
  |
  |==============================                                   |  49%
  |
  |==============================                                   |  50%
  |
  |=================================================================| 100%
```

```
pred.dl<-h2o.predict(object=res.dl, newdata=tsData[,-1])
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
pred.dl.df<-as.data.frame(pred.dl)
test_labels<-test[,1]

#calculate number of correct prediction
sum(diag(table(test_labels,pred.dl.df[,1])))
```

```
## [1] 8065
```

```
# read test.csv


test_h2o<-as.h2o(digit.test)
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
# convert H2O format into data frame and save as csv
pred.dl.test<-h2o.predict(object=res.dl, newdata=test_h2o[,-1])
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

```
df.test <- as.data.frame(pred.dl.test)
df.test <- data.frame(ImageId = seq(1,length(df.test$predict)), Label = df.test$predict)
write.csv(df.test, file = "submission.csv", row.names=FALSE)

# shut down virtual H2O cluster
h2o.shutdown(prompt = FALSE)
```

```
## [1] TRUE
```

```
```