Neural Networks

Shantam Gupta

May 13, 2018

Contents

1 Installing the Package **1**

2 Load the Data **1**

3 Preprocess the data: Normalize the data **1**

4 Building Neural Network **2**

1 Installing the Package

2 Load the Data

3 Preprocess the data: Normalize the data

```
new_data <- rbind(S0,Data)
maxs <- apply(new_data %>% select(-c(idfile,RESPONSE)), 2, max)
mins <- apply(new_data %>% select(-c(idfile,RESPONSE)), 2, min)

scaled_data <- as.data.frame(scale(new_data %>% select(-c(idfile,RESPONSE)), center = mins, scale = maxs
#scaled_data$RESPONSE <- ifelse(new_data$RESPONSE =="GO",1,0)
scaled_data$RESPONSE <- as.factor(new_data$RESPONSE)
#scaled_data$RESPONSE <- as.factor(scaled_data$RESPONSE)
scaled_data$idfile <- new_data$idfile

#select random ind for train and test
set.seed(123)

## 75% of the sample size
smp_size <- floor(0.75 * nrow(scaled_data))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(scaled_data)), size = smp_size)


train <- scaled_data[train_ind,]
test <- scaled_data[-train_ind,]
```

4 Building Neural Network

```r
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## 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
```

```r
#generate same set of random numbers (for reproducibility)
set.seed(121)

#launch h2o cluster
localH2O <- h2o.init(nthreads = -1)


#import r objects to h2o cloud
train_h2o <- as.h2o(train)
test_h2o <- as.h2o(test)
```

```r
#build the mlp(multi layer perceptron) deep learning model using h2o
set.seed(100)

dl_model <- h2o.deeplearning(
  model_id="dl_model_first",
  training_frame=train_h2o,
  validation_frame = test_h2o,
  x= colnames(train_h2o[,1:48]),
  y= "RESPONSE",
  activation="Rectifier",
  hidden=c(5,4),
  stopping_metric="mean_per_class_error",
  stopping_tolerance=0.01,
```

```r
  epochs=100
)
```

```r
summary(dl_model)
```

```
## Model Details:
## ==============
##
## H2OBinomialModel: deeplearning
## Model Key:  dl_model_first
## Status of Neuron Layers: predicting RESPONSE, 2-class classification, bernoulli distribution, CrossEn
batch size 1
##   layer units      type dropout       l1       l2 mean_rate rate_rms
## 1     1    48     Input  0.00 %
## 2     2     5 Rectifier  0.00 % 0.000000 0.000000  0.005227 0.004851
## 3     3     4 Rectifier  0.00 % 0.000000 0.000000  0.004046 0.005787
## 4     4     2   Softmax         0.000000 0.000000  0.001208 0.000238
##   momentum mean_weight weight_rms mean_bias bias_rms
## 1
## 2 0.000000   -0.012467   0.246006  0.608905 0.121557
## 3 0.000000    0.132601   0.830154  1.048259 0.200680
## 4 0.000000   -0.559657   2.181219  0.005555 0.047890
##
## H2OBinomialMetrics: deeplearning
## ** Reported on training data. **
## ** Metrics reported on full training frame **
##
## MSE:  0.007385807
## RMSE:  0.08594072
## LogLoss:  0.03213156
## Mean Per-Class Error:  0.02012779
## AUC:  0.9974616
## Gini:  0.9949232
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##        GO NOGO    Error     Rate
## GO     80    3 0.036145    =3/83
## NOGO    4  969 0.004111   =4/973
## Totals 84  972 0.006629  =7/1056
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.543928 0.996401 315
## 2                       max f2  0.446713 0.996302 319
## 3                 max f0point5  0.711292 0.997519 309
## 4                 max accuracy  0.543928 0.993371 315
## 5                max precision  1.000000 1.000000   0
## 6                   max recall  0.002332 1.000000 399
## 7              max specificity  1.000000 1.000000   0
## 8             max absolute_mcc  0.543928 0.954505 315
## 9   max min_per_class_accuracy  0.790724 0.987952 306
## 10 max mean_per_class_accuracy  0.711292 0.989865 309
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F
```

```
## H2OBinomialMetrics: deeplearning
## ** Reported on validation data. **
## ** Metrics reported on full validation frame **
##
## MSE:  0.008362441
## RMSE:  0.09144638
## LogLoss:  0.02741105
## Mean Per-Class Error:  0.02083333
## AUC:  0.9997467
## Gini:  0.9994934
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##        GO NOGO    Error    Rate
## GO     23    1 0.041667   =1/24
## NOGO    0  329 0.000000  =0/329
## Totals 23  330 0.002833  =1/353
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.868842 0.998483 316
## 2                       max f2  0.868842 0.999392 316
## 3                 max f0point5  0.919001 0.998778 313
## 4                 max accuracy  0.868842 0.997167 316
## 5                max precision  1.000000 1.000000   0
## 6                   max recall  0.868842 1.000000 316
## 7              max specificity  1.000000 1.000000   0
## 8             max absolute_mcc  0.868842 0.977461 316
## 9   max min_per_class_accuracy  0.919001 0.993921 313
## 10 max mean_per_class_accuracy  0.919001 0.996960 313
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
##
##
## Scoring History:
##             timestamp   duration training_speed    epochs iterations
## 1 2018-05-17 20:30:21  0.000 sec                  0.00000          0
## 2 2018-05-17 20:30:21  0.101 sec 224680 obs/sec  10.00000          1
## 3 2018-05-17 20:30:21  0.399 sec 324923 obs/sec 100.00000         10
##         samples training_rmse training_logloss training_auc training_lift
## 1      0.000000
## 2  10560.000000       0.18376          0.11958      0.96312       1.08530
## 3 105600.000000       0.08594          0.03213      0.99746       1.08530
##   training_classification_error validation_rmse validation_logloss
## 1
## 2                       0.03883         0.16542            0.09604
## 3                       0.00663         0.09145            0.02741
##   validation_auc validation_lift validation_classification_error
## 1
## 2        0.97480         1.07295                         0.02550
## 3        0.99975         1.07295                         0.00283
##
## Variable Importances: (Extract with `h2o.varimp`)
## =================================================
##
```

```
## Variable Importances:
##                variable relative_importance scaled_importance percentage
## 1  TotalArea.LVNELTEFAK            1.000000          1.000000   0.044828
## 2  MassAccu.HLVDEPQNLIK            0.950939          0.950939   0.042629
## 3 MassAccu.SLHTLFGDELCK            0.936479          0.936479   0.041980
## 4      FWHM.HLVDEPQNLIK            0.790054          0.790054   0.035417
## 5 Charge.VPQVSTPTLVEVSR            0.713035          0.713035   0.031964
##
## ---
##                variable relative_importance scaled_importance percentage
## 43 MassAccu.LVNELTEFAK            0.300878          0.300878   0.013488
## 44   MassAccu.NECFLSHK            0.275860          0.275860   0.012366
## 45   MZ.ECCHGDLLECADDR            0.268748          0.268748   0.012047
## 46   MZ.VPQVSTPTLVEVSR            0.238041          0.238041   0.010671
## 47   Charge.EACFAVEGPK            0.233442          0.233442   0.010465
## 48   FWHM.YICDNQDTISSK            0.225689          0.225689   0.010117
```

The accuracy is 98.57% . The net could be optmizied further to improve the accuracy

4.0.1 Tuning the ANN

The simplest hyperparameter search method is a brute-force scan of the full Cartesian product of all combinations specified by a grid search. There are a lot of paramters to tune and due to limited computational capabilities we shall try to tune only some of them.

```
#hyperparamters to tune
hyper_params <- list(
  hidden=list(c(32,32,32),c(50,200,50)),  # different architectures of hidden layer
  input_dropout_ratio=c(0,0.05),      # values for drop out
  rate=c(0.01,0.02),                  # the learning rae
  activation = c("Rectifier")   # activation functions
)

#grid search

grid <- h2o.grid(
  algorithm="deeplearning",
  grid_id="dl_grid",
  model_id="dl_model_first",
  training_frame=train_h2o,
  x= colnames(train_h2o[,1:12]),
  y= "label",
  stopping_metric="mean_per_class_error",
  hyper_params = hyper_params,
  epochs=1000,
  stopping_tolerance=0.01,
  variable_importances=T
)
```

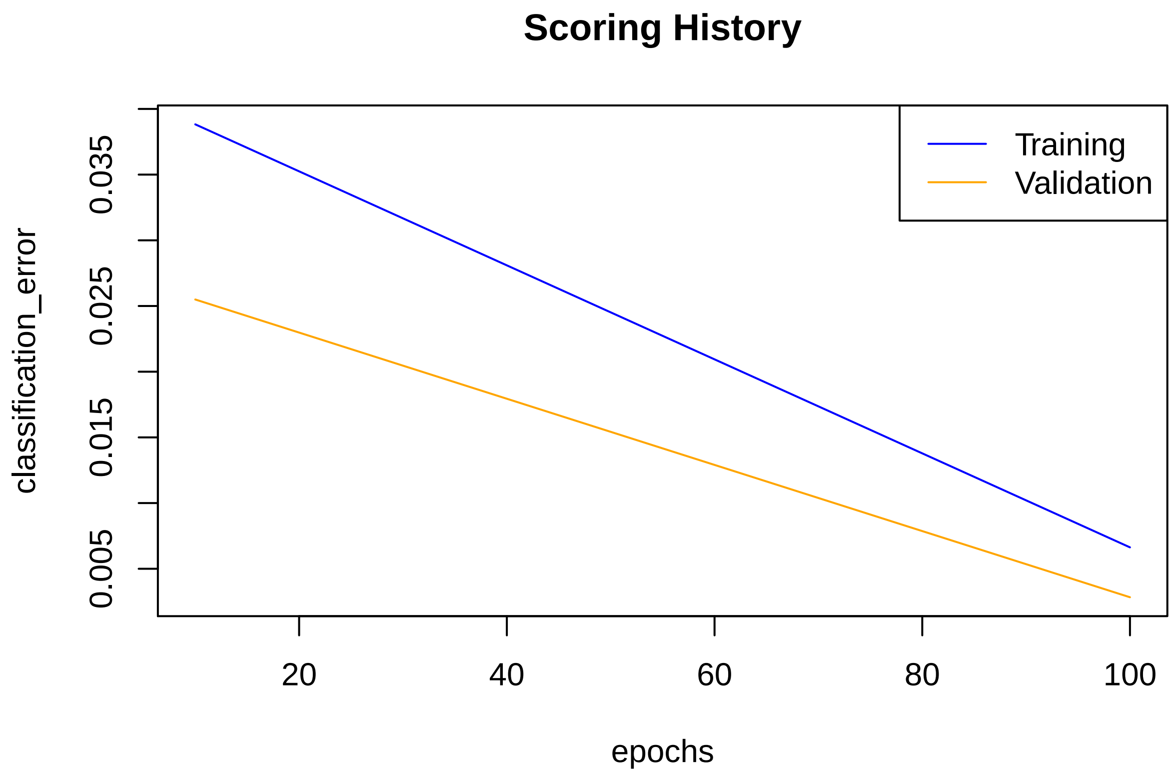
```
# sort the model in the grid in decreasing order of error
grid <- h2o.getGrid("dl_grid", sort_by = "err", decreasing = FALSE)
grid

# best model and its full set of parameters
grid@summary_table[1, ]
```

```r
best_dl_model <- h2o.getModel(grid@model_ids[[1]])
best_dl_model

print(h2o.performance(best_dl_model))

# storing the confusion matrix
best_dl_confusion <- as.data.frame(h2o.confusionMatrix(best_dl_model))
```

4.0.2 Plotting the model

```r
plot(dl_model,timesteps = "epochs",metric = "classification_error")
```

```
## Warning in plot.window(...): "timesteps" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "timesteps" is not a graphical parameter

## Warning in title(...): "timesteps" is not a graphical parameter

## Warning in plot.window(...): "timesteps" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "timesteps" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "timesteps" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "timesteps" is
## not a graphical parameter

## Warning in box(...): "timesteps" is not a graphical parameter

## Warning in title(...): "timesteps" is not a graphical parameter
```

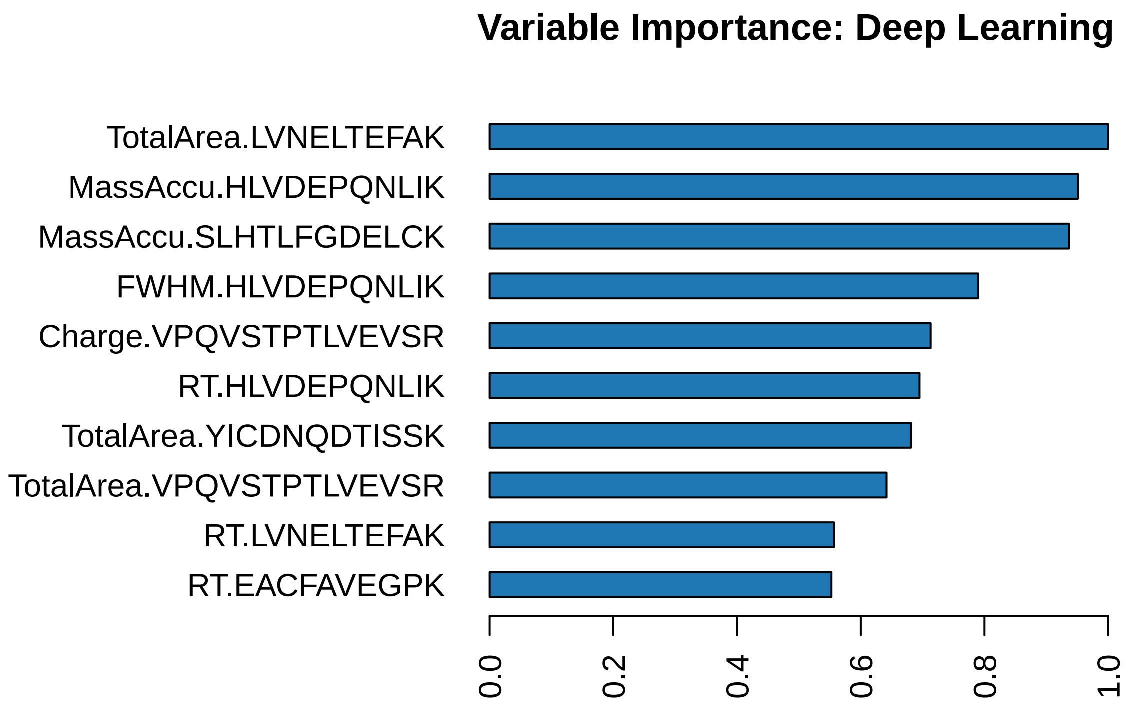


The training accuracy decreases with increase in epochs. However, this might lead to overfitting on training data and poor fit on the test data.

4.0.3 Predictons on test data

```r
dl_predict <- as.data.frame(h2o.predict(dl_model, test_h2o))
```

```r
h2o.varimp_plot(dl_model)
```


Variable Importance: Deep Learning
TotalArea.LVNELTEFAK
MassAccu.HLVDEPQNLIK
MassAccu.SLHTLFGDELCK
FWHM.HLVDEPQNLIK
Charge.VPQVSTPTLVEVSR
RT.HLVDEPQNLIK
TotalArea.YICDNQDTISSK
TotalArea.VPQVSTPTLVEVSR
RT.LVNELTEFAK
RT.EACFAVEGPK
0.0
0.2
0.4
0.6
0.8
1.0