Deep Learning

H2O platform was used to perform deep learning in order to predict departure delay greater than 90 minutes. H2O follows the multi-layer, feedforward neural network model of deep learning predictions [@DeepFeedForward]. In the feedforward neural network model, the inputs are weighted, combined and transmitted as output signal by the connected neuron. Function f shown in Figure 1 shows a nonlinear activation function where the bias accounts for the activation threshold. A nonlinear activation function ensures that the linearly input hidden layers experience variation. Otherwise the output will simply be a linear combination of the hidden layers making the use of hidden layers irrelevant. Examples of activation functions include sigmoid and rectified linear unit (ReLU). The multi-layer platform consists of layers of interconnected neurons that is initiated with the input layer followed by layers of nonlinearity culminating in a regression or classification layer [@deepH2Odoc]. An example of a multi-layer neural network is shown in Figure 2.

Figure 1: Neural Network Model

Figure 2: Hidden Layers

Overall H2O's deep learning functionalities include specification of regularization options, learning rate, annealing, hyperparameter optimization and model selection through grid and random search. Additionally H2O facilitates automatic processing for categorical and numerical data along with automatic imputation of missing values and ensures fast convergence.

H2O was used with the flights dataset containing over million observations filtered by departure delay greater than 90 minutes. The response variable was a continuous predictor capturing the extent of the departure delay in minutes. This dataset differed from the dataset used to perform logistic regression, which contained a binary predictor to predict whether or not delay greater than 30 minutes occurred. In the deep learning scenario, the explanatory variables of interest included year, month, arrival delay, carrier, distance, hour, week, weekend and season.

Data Partitioning

Before data preparation and model building, connection was established to the YARN client. Following connection, data sample of 200000 was obtained given that this datasize was easily transferable to the Spark environment. Since hyperparameter optimization was performed in the deep learning algorithm, a validation data set was used along with the test set for additional verification. Data split was split with 60/20/20 split consisting of 60% training, 20% validation and 20% test set.

```r
options(rsparkling.sparklingwater.version = "1.6.8")
sc <- spark_connect(master = "yarn-client")

set.seed(12)
thousand <- FullDat[sample(nrow(FullDat), 200000, replace = FALSE,
                           prob = NULL),]
mtcars_tbl <- copy_to(sc, thousand, "mtcars", overwrite = TRUE)

partitions <- mtcars_tbl %>%
  sdf_partition(training = 0.6, validation = 0.20, test = 0.20,
                seed = 1099)
```

Deep Learning Model Building

Following data partitioning, h2o.deeplearning function was used to predict departure delay over 90 minutes as a function of year, month, arrival delay, carrier, distance, hour, week, weekend and season. The h2o.deeplearning function includes specification of the explanatory (x) and the response predictors (y). In addition, h2o.deeplearning paramaters include activation function specification (Tanh, TanhWithDropout, Rectifier, RectifierWithDropout, Maxout, MaxoutWithDropout, see figure Figure 3), training and validation_frame delineation along with fine-tuning parameters like maximum model iterations, regularization parameters like l1 and l2, non-negative shrinkage parameter lambda and cross validation parameter (nfolds) specifying number of folds and model iterations. A random seed can also be specified though this is only reproducible with algorithms running on a single thread. Additionally h2o.deeplearning model provides the ability to stop the model learning early if no apparent changes in the loss function are observed.

Figure 3: Activation Function

A simple model shown below was constructed to predict departure delay over 90 minutes as a function of the explantory variables. Epoch of one was used indicating a single data iteration. In addition, 5 folds cross validation was used. Tanh activation layer was used since it is more adept at exponentially rising functions and consequently better at regulating regularization.

```r
myX = setdiff(colnames(training), ("dep_delay"))
deepmod <- h2o.deeplearning(
  y="dep_delay",
  x=myX,
  activation="Tanh",
  training_frame=training,
  validation_frame=validation,
  epochs=1,
  variable_importances=T,
  nfolds = 5,
  keep_cross_validation_predictions=T
)
```

A variable importance plot can be used to view the most important predictors produced by deepmod. In this case, a plot of the top 20 predictors was produced. As Figure 4 shows, arrival delay appears to be most important at predicting departure delay greater than 90 minutes. Following arrival delay, day status (weekday or weekend) appears to be next important. In regards to particular carriers, Continental Airlines (CO), Comair, Inc. (OH) and 9 Air Co Ltd appear to be most important in predicting departure delay greater than 90 minutes. Additionally hour 14 (2 pm) appears to be important in predicting departure delay greater than 90 minutes.

```r
h2o.varimp_plot(deepmod, num_of_features = 20)
```

Figure 4: Variable Importance Deep Learning

Model Assessment

The performance of the deep learning model can be assessed. Since the response predictor is continuous, mean squared error (MSE) and root mean squared error (RMSE) were used as error metrics.

As shown by Figure 5, MSE on the training data was 459.14 while the MSE on the validation data was 472.58. In comparison, MSE on the test data was 491.59. Since MSE for the validation and test data is higher than the MSE on the training data, this is an indication of a good fit. Overfitting does not appear to be extremely visible since the difference between the MSE of the training, validation and test set does not appear to be too visible.

```r
deepmod@parameters
h2o.performance(deepmod, train = TRUE)
h2o.performance(deepmod, valid = TRUE)
h2o.performance(deepmod, newdata = test)
```

Figure 5: Deep Learning Model Performance

Saving Model

Once the model is constructed, it can be saved using the h2o.saveModel command as shown below with the path specification.

```
DeepModel <- h2o.saveModel(m1, path = "/home/ajavaid17", force = FALSE)
```

Model can be loaded with the h2o.loadModel command with the specified path as an argument.

```
ld <- h2o.loadModel(path ="/home/ajavaid17/
                     DeepLearning_model_R_1487567612904_2")
```

Grid Search Model

H2O provides grid search functionality which allows the user to experiment with different hyperparameter combinations. All possible combinations of the hyperparameters are tested. In the model below, 2 different activation functions, 2 hidden layers, 2 input_dropout_ratio and 3 rate parameters were tested resulting in 24 models. Tanh and TanhWithDropout parameters were used since they better regularize for exponential functions. The hidden variable specifies the hidden layer sizes. The rate parameter specifies the learning rate where a higher rate produces less model stability and a lower rate produces slower model convergence. The rate_annealing parameter attempts to adjust learning rate.

```r
hyper_params <- list(
  activation=c("Tanh", "TanhWithDropout"),
  hidden=list(c(20,20),c(40,40)),
  input_dropout_ratio=c(0,0.05),
  rate=c(0.01,0.02,0.03)
)
```

After setting up the hyperparameters, h2o.grid functionality was used to iterate over models. In order to expedite the model building process, stopping metrics were specified so the h2o.grid functionality stops when the MSE does not improve by greater than or equal to 2% (stopping_tolerance) for 2 events (stopping_rounds). In addition to the hyperparameters specified, epoch of 10 was chosen. Momentum was specified to reduce the chance of the algorithm halting at a local minima. Theoretically, momentum specification reduces terrrain irregularities thus preventing algorithm to stop at the minima. The l1 and l2 regularization parameters attempt to prevent overfitting while the max_w2 sets the constraint for squared sum of incoming weights per unit.

```r
grid <- h2o.grid(
  algorithm="deeplearning",
  grid_id="gridDeep",
  training_frame=training,
  validation_frame=validation,
  y="dep_delay",
  x=myX,
  epochs=10,
  stopping_metric="MSE",
  stopping_tolerance=2e-2,
  stopping_rounds=2,
  score_duty_cycle=0.025,
  adaptive_rate=T,
  momentum_start=0.5,
  momentum_stable=0.9,
  momentum_ramp=1e7,
  variable_importances=T,
  l1=1e-5,
  l2=1e-5,
  max_w2=10,
  hyper_params=hyper_params
)
```

For loop can be used to iterate over all 24 models to output the MSE. Direct indexing in the grid object can be used to retrieve the optimal model along with the associated parameters. The gridDeep grid search resulted in an optimal model with parameters including the Tanh activation layer, hidden layer of (40, 40), input_dropout_ratio of 0 and rate of 0.01 as depicted by Figure 6.

```r
for (model_id in grid@model_ids) {
  model <- h2o.getModel(model_id)
  mse <- h2o.mse(model, valid = TRUE)
  sprintf("Validation set MSE: %f", mse)
}

#Retrieve optimal model by MSE
grid@summary_table[1,]
optimal <- h2o.getModel(grid@model_ids[[1]])

optimal@allparameters #print all parameters of best model
h2o.performance(optimal, train = TRUE) #retrieve training MSE
h2o.performance(optimal, valid = TRUE) #retrieve validation MSE
h2o.performance(optimal, newdata = test) #retrieve test MSE
```

Figure 6: Deep Learning Grid Model Parameters

As shown by Figure 7, the MSE on the optimal model for the training data was 318.74 whereas MSE for the validation data was 372.26 and 390.94 for the test data. The validation and test errors were higher than training signaling towards a good model fit, with some reservations for overfitting. This compares with deepmod where the training MSE was 374.39, validation MSE of 430.06 and test MSE of 423.41.

Figure 7: Deep Learning Grid Model

Variable importance plot can be used to view the most important predictors. As Figure 8 shows, arrival delay appears to be most important in predicting departure delay greater than 90 minutes. Following arrival delay, distance and air time appear to be next important. JetSuite Air (XE) appears to be the most important carrier in predicting departure delay greater than 90 minutes. Season summer and month August additionally appear to be important in predicting departure delay greater than 90 minutes.

```r
h2o.varimp_plot(deepmod, num_of_features = 20)
```

Figure 8: Variable Importance Deep Learning

Random Grid Search Model

In comparison to grid search which iterated over combinations exhaustively and sequentially, random grid search model can be used to accelerate the process of hyperparameter selection. This is particularly useful in situations where the user wants to consider a large number of hyperparameters. Random grid search proceeds to randomly search the user specified space using an established search criteria.

Since random grid search model was used, additional hyperparameters were tested for analysis. As shown below, Tanh and TanhWithDropout functions were tested. The hidden layer additionally included (30, 30, 30), (50,50) and (70,70). Rate 0.03 was also tested along with 0.0 and 0.02. In addition, different combinations of regularization parameters (l1 and l2) were tested.

```
hyper_params <- list(
  activation=c("Tanh","TanhWithDropout"),
  hidden=list(c(20,20),c(30,30,30),c(40,40,40),c(50,50),c(70,70)),
  input_dropout_ratio=c(0,0.05),
  rate=c(0.01,0.02,0.03),
  l1=seq(0,1e-4,1e-6),
  l2=seq(0,1e-4,1e-6)
)
```

With the hyperparameters specified, next step included defining the search criteria. As the criteria below shows, the algorithm was defined to stop when the top 5 models were within 2% of each other. Max model running time was 600 seconds (10 minutes). In addition to the max running time, number of max_models can also be specified.

```
search_criteria = list(strategy = "RandomDiscrete",
                       max_runtime_secs = 600, max_models = 100,
                       seed=22, stopping_rounds=5,
                       stopping_tolerance=2e-2)
```

Following delineation of the search criteria, the h2o.grid function can be used to specify additional fixed parameters. These include definition of epochs of 10, max_w2 of 10, score_validation_samples of 10000 and score_duty_cycles of 0.025. The score_validation_samples specify the number of validation set samples for scoring while the score_duty_cycles specifies the maximum duty cycle fraction for scoring. The same stopping parameters as the grid search model were used. The algorithm was thus indicated to stop when the MSE does not improve by at least 2% for 2 scoring events.

```
random_grid <- h2o.grid(
  algorithm="deeplearning",
  grid_id = "Gridrandom",
  training_frame=training,
  validation_frame=validation,
  x=myX,
  y="dep_delay",
  epochs=10,
  stopping_metric="MSE",
  stopping_tolerance=2e-2,
  stopping_rounds=2,
  score_validation_samples=10000,
  score_duty_cycle=0.025,
  max_w2=10,
  hyper_params = hyper_params,
  search_criteria = search_criteria
)
```

The validation set MSE for all the random search models can be printed with a for loop. Additionally, the best model and its associated parameters can be viewed. As shown by Figure 9, optimal model generated by random grid search has an activation function of Tanh, hidden layer of (30,30,30), input_dropout_ratio of 0, rate of 0.02, l1 of 2.3e-05 and l2 of 7.6e-05.

```r
for (model_id in grid@model_ids) {
  model <- h2o.getModel(model_id)
  mse <- h2o.mse(model, valid = TRUE)
  sprintf("Validation set MSE: %f", mse)
}

#Retrieve optimal model by MSE
grid@summary_table[1,]
optimal <- h2o.getModel(grid@model_ids[[1]])

optimal@allparameters #print all parameters of best model
h2o.performance(optimal, train = TRUE) #retrieve training MSE
h2o.performance(optimal, valid = TRUE) #retrieve validation MSE
h2o.performance(optimal, newdata = test) #retrieve test MSE
```

Figure 9: Deep Learning Grid Model Parameters

As Figure 10 indicates, the MSE on the optimal random model for the training data was 306.25 whereas MSE for the validation set was 309.64 and 412.57 for the test set. The training and validation error are lower than those yielded by the grid search model (318.74 for training and 372.26 for validation) while the test error for the random search model is higher than the test error for the grid search model (390.94 for test for grid search model). Since the random search test error is much higher than the training MSE, there may be some occurrence of overfitting. The errors for the random search model are lower than the initial deep learning model deepmod (training MSE of 374.39, validation MSE of 430.06 and test MSE of 423.41).

Figure 10: Deep Learning Random Grid Performance

Variable importance plot can be used to view the most important predictors. In this case, a plot of the top 20 predictors was produced. As Figure 11 shows, arrival delay appears to be the most important predictor in predicting departure delay greater than 90 minutes. Following arrival delay, air time and distance appear to be the next important predictors. Additionally hour 4 (4 am), hour 5 (5 am), hour 0 (12 am) and weekend appear to be the next important predictors. Spirit appears to be the most important carrier in predicting departure delay greater than 90 minutes.

```r
h2o.varimp_plot(optimal, num_of_features = 20)
```

Figure 11: Variable Importance Random Deep Learning

Checkpoint Model

Checkpoint functionality can be used in H2O to continue iterations from a previously built model. Checkpoint option allows specification of a previously built model key. The new model is then built as a continuation of the old model. If the model key is not supplied, then a new model is built instead. In the checkpoint model, the value of the parameters must be greater than their value set in the previous model. Parameters like activation function, max_categorical_features, momentum_ramp, momentum_stable, momentum_start and nfolds cannot be modified. A full list of all the parameters that cannot be modified can be found at http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/checkpoint.html.

Since the test MSE (412.57) produced by the random search model was much higher than the errors on the training (306.25) and validation set (309.64), higher l1 and l2 parameters were used to produce better performance on the test set. Additionally epochs of 20 were used. These additional parameters were specified from the initial basis of the optimal random grid search model, specified below in the checkpoint specification as Gridrandom6_model_6. Same activation (Tanh), hidden layer (30,30,30) and rate (0.02) were used since these parameters can't be altered in the checkpoint model.

```
max_epochs <- 20
checkpoint <- h2o.deeplearning(
  model_id="GridModRandom_continued2",
  activation="Tanh",
  checkpoint="Gridrandom6_model_6",
  training_frame=training,
  validation_frame=validation,
  y="dep_delay",
  x=myX,
  hidden=c(30,30,30),
  epochs=max_epochs,
  stopping_metric="MSE",
  stopping_tolerance=2e-2,
  stopping_rounds=2,
  score_duty_cycle=0.025,
  adaptive_rate=T,
  l1=1e-4,
  l2=1e-4,
  max_w2=10,
  rate = 0.02,
  variable_importances=T
)
```

As shown by Figure 12, MSE on the training data was 356.75, 388.46 for validation and 399.96 for the test data. The test MSE is much closer to the validation set whereas MSE previously on the random grid search test data was 412.57. Since the test MSE is closer to the training and validation set's MSE, this reduces chance of overfitting.

Figure 12: Deep Learning Checkpoint Performance

According to the variable importance chart shown by Figure 13, arrival delay appears to be most important in predicting departure delay greater than 90 minutes. Hour 4 (4 am), air time, hour 0 (12 am) and hour 5 (5 am) appear to be next important. Additionally Spirit appears to be an important carrier in predicting departure delay greater than 90 minutes.

Figure 13: Variable Importance Checkpoint

Conclusions

This chapter discussed deep learning models including concepts like setting up hyperparameters for deep learning models through the grid search and random search methods. In addition, the checkpoint model functionality was discussed.

Overall, airline delay appears to be most important in predicting departure delay greater than 90 minutes which indicates that airlines experiencing airline delay appear to have a higher likelihood of experiencing departure delays. Additional important predictors include amount of distance covered as well as the amount of air time. Early morning hours between 12-5 am also appear to be important predictors of departure delay greater than 90 minutes.

In regards to model performance, the random search model produced similar and in some regards better results than the grid search model (lower MSE on training and validation sets). The checkpoint model improved upon the random search model by reducing the test set MSE, thus lowering incidence of overfitting.