Neural Network with R

This lab manual is to demonstrate how to build a Neural Network in R. The package that we will be using is H2O Scalable Machine Learning Platform.

Building the neural network

We will be using the consultation dataset. The dataset contains 6 columns as follows.

- Qualification: Qualification and degrees held by the physician
- Experience: Experience of the physician in number of years
- Rating: Rating given by patients
- Place: Area and the city where the physician is located
- Profile: Type of the physician
- Fees: Consultation fee

Download the consultation dataset. Load the package as follows.

```
> install.packages("h2o")
> library(h2o)
> install.packages("Metrics")
> library(Metrics)
```

Before we can build the model, we need to start H2O.

```
> h2o.init(
```

Load the data into R. H2O. H2O implement its own data.frame which is similar to R's data.frame.

```
> consultation.frame <- h2o.importFile("consultation_fee.csv")
```

or

```
> consultation <- read.csv("consultation_fee.csv")
> consultation.frame <- as.h2o(consultation)
```

We can examine the columns in the data frame.

```
> str(consultation.frame)
```

Column "Experience", "Rating", and "Fees" need to be normalized since they are in different range or scales. First, we define a function to perform the normalization.

```
> normalize <- function(x) {
    return <- ((x - min(x)) / (max(x) - min(x)))
  }

> consultation.frame$Experience.norm <-
  normalize(consultation.frame$Experience)

> consultation.frame$Rating.norm <-
  normalize(consultation.frame$Rating)
```

```
> consultation.frame$Fees.norm <- normalize(consultation.frame$Fees)
```

Next, we split the data into training dan test sets, in a ratio of 70:30. The training set is used for training and creating the model. The test set is to evaluate the accuracy of the model.

```
> split <- h2o.splitFrame(consultation.frame, ratios=0.7)
> train <- split[[1]]
> test <- split[[2]]
```

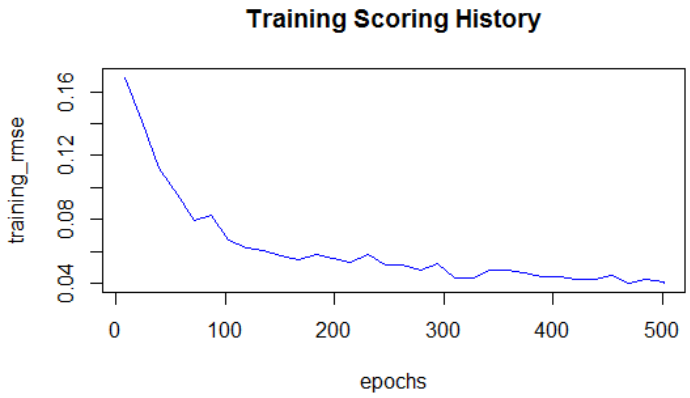
We would like to predict the consultation fees given the attributes. Let's build a neural network model with mostly default parameters.

x	Defines the column number of the attributes.
y	Defines the column number of the target.
training_frame	id of the training frame.
epochs	The number of times the dataset will be iterated.
mini_batch_size	Size of the mini-batch
hidden	Hidden layer sizes, default to c(200,200)
seed	Seed for random numbers

```
> nn <- h2o.deeplearning(x=c(1,4,5,7,8), y=9, training_frame=train,
  epochs=500, mini_batch_size=32, hidden=c(20,20), seed=1)
```

We can plot the scoring history as follows.

```
> plot(nn)
```



Let's perform the prediction on the test data and evaluate its performance.

```
> pred <- h2o.predict(nn, test)
> rmse(test$Fees.norm, pred)
```

As we can see, the root mean squared error of the prediction is 0.2247. You might get a different value.

Regularization

Early Stopping

Let's implement early stopping. We add three (3) more parameters which as follows.

stopping_round	Early stopping based on convergence of stopping_metric. Stop if the stopping_metric does not improve for k scoring events (0 to disable) Defaults to 5.
stopping_metric	Metric to use for early stopping. We can also use mean squared error (mse).
stopping_tolerance	Relative tolerance for metric-based stopping criterion (stop if relative improve-ment is not at least this much) Defaults to 0.
score_interval	Specify the shortest time interval (in seconds) to wait between model scoring.

```
> nn <- h2o.deeplearning(x=c(1,4,5,7,8), y=9, training_frame=train,
epochs=500, mini_batch_size=32, hidden=c(20,20), seed=1,
stopping_metric="rmse", stopping_rounds=3, stopping_tolerance=0.05,
score_interval=1)

> pred <- h2o.predict(nn, test)

> rmse(test$Fees.norm, pred)
```

Here, we stop the training when the score (RMSE) does not improve >=0.05 for 3 times. We can see that the error has been reduced to 0.2167 (you might get a different value).

Dropout Regularization

Let's implement dropout regularization. There are three parameters which can be set to implement the regularization.

activation	Activation function. Must be one of: "Tanh", "TanhWithDropout", "Rectifier","RectifierWithDropout", "Maxout", "MaxoutWithDropout". Defaults to Recti-fier. To implement dropout, choose either "TanhWithDropout", " RectifierWithDropout" or "MaxoutWithDropout".
input_dropout_ratio	Input layer dropout ratio (can improve generalization, try 0.1 or 0.2). Defaultsto 0.
hidden_dropout_ratio	Hidden layer dropout ratios (can improve generalization), specify one value perhidden layer, defaults to 0.5.

```
> nn <- h2o.deeplearning(x=c(1,4,5,7,8), y=9, training_frame=train,
  epochs=500, mini_batch_size=32, hidden=c(20,20), seed=1,
  stopping_metric="rmse", stopping_rounds=3, stopping_tolerance=0.05,
  score_interval=1, activation="RectifierWithDropout",
  hidden_dropout_ratio=c(0.5,0.5))

> pred <- h2o.predict(nn, test)

> rmse(test$Fees.norm, pred)
```

The error has been significantly reduced to 0.1894 (you might get a different value). We could further improve the error by dropping the input neurons.

```
> nn <- h2o.deeplearning(x=c(1,4,5,7,8), y=9, training_frame=train,
  epochs=500, mini_batch_size=32, hidden=c(20,20), seed=1,
  stopping_metric="rmse", stopping_rounds=3, stopping_tolerance=0.05,
  score_interval=1, activation="RectifierWithDropout",
  hidden_dropout_ratio=c(0.5,0.5), input_dropout_ratio=0.1)

> pred <- h2o.predict(nn, test)

> rmse(test$Fees.norm, pred)
```

The error has been further reduced to 0.1864 (you might get a different value).

References

https://cran.r-project.org/web/packages/h2o/h2o.pdf

http://docs.h2o.ai/h2o-tutorials/latest-stable/tutorials/deeplearning/index.html