Regression Artificial Neural Network

Diagram

Description automatically generatedRegression ANNs predict an output variable as a function of the inputs. The input features (independent variables) can be categorical or numeric types, however, for regression ANNs, we require a numeric dependent variable. If the output variable is a categorical variable (or binary) the ANN will function as a classifier (see [next tutorial](http://uc-r.github.io/ann_classification)).

tl;dr

In this tutorial we introduce a neural network used for numeric predictions and cover:

1. [Replication requirements](http://uc-r.github.io/ann_regression#req): What you’ll need to reproduce the analysis in this tutorial.
2. [Data Preparation](http://uc-r.github.io/ann_regression#prep): Preparing our data.
3. [1st Regression ANN](http://uc-r.github.io/ann_regression#one-layer): Constructing a 1-hidden layer ANN with 1 neuron.
4. [Regression Hyperparameters](http://uc-r.github.io/ann_regression#hyper): Tuning the model.
5. [Wrapping Up](http://uc-r.github.io/ann_regression#summary): Final comments and some exercises to test your skills.

Replication Requirements

We require the following packages for the analysis.

library(tidyverse)

library(neuralnet)

library(GGally)

Data Preparation

Our regression ANN will use the [Yacht Hydrodynamics](http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics) data set from UCI’s Machine Learning Repository. The yacht data was provided by Dr. Roberto Lopez [email](mailto:roberto-lopez@users.sourceforge.net). This data set contains data contains results from 308 full-scale experiments performed at the Delft Ship Hydromechanics Laboratory where they test 22 different hull forms. Their experiment tested the effect of variations in the hull geometry and the ship’s [Froude number](https://en.wikipedia.org/wiki/Froude_number) on the craft’s residuary resistance per unit weight of displacement.

To begin we download the data from UCI.

url <- 'http://archive.ics.uci.edu/ml/machine-learning-databases/00243/yacht\_hydrodynamics.data'

Yacht\_Data <- read\_table(file = url,

col\_names = c('LongPos\_COB', 'Prismatic\_Coeff',

'Len\_Disp\_Ratio', 'Beam\_Draut\_Ratio',

'Length\_Beam\_Ratio','Froude\_Num',

'Residuary\_Resist')) %>%

na.omit()

Prior to any data analysis lets take a look at the data set.

ggpairs(Yacht\_Data, title = "Scatterplot Matrix of the Features of the Yacht Data Set")

Diagram

Description automatically generated with medium confidence

Here we see an excellent summary of the variation of each feature in our data set. Draw your attention to the bottom-most strip of scatter-plots. This shows the residuary resistance as a function of the other data set features (independent experimental values). The greatest variation appears with the Froude Number feature. It will be interesting to see how this pattern appears in the subsequent regression ANNs.

Prior to regression ANN construction we first must split the Yacht data set into test and training data sets. Before we split, first scale each feature to fall in the [0,1][0,1] interval.

# Scale the Data

scale01 <- function(x){

(x - min(x)) / (max(x) - min(x))

}

Yacht\_Data <- Yacht\_Data %>%

mutate\_all(scale01)

# Split into test and train sets

set.seed(12345)

Yacht\_Data\_Train <- sample\_frac(tbl = Yacht\_Data, replace = FALSE, size = 0.80)

Yacht\_Data\_Test <- anti\_join(Yacht\_Data, Yacht\_Data\_Train)

The scale01() function maps each data observation onto the [0,1][0,1] interval as called in the dplyr mutate\_all() function. We then provided a seed for reproducible results and randomly extracted (without replacement) 80% of the observations to build the Yacht\_Data\_Train data set. Using dplyr’s anti\_join() function we extracted all the observations not within the Yacht\_Data\_Train data set as our test data set in Yacht\_Data\_Test.

1st Regression ANN

To begin we construct a 1-hidden layer ANN with 1 neuron, the simplest of all neural networks.

set.seed(12321)

Yacht\_NN1 <- neuralnet(Residuary\_Resist ~ LongPos\_COB + Prismatic\_Coeff +

Len\_Disp\_Ratio + Beam\_Draut\_Ratio + Length\_Beam\_Ratio +

Froude\_Num, data = Yacht\_Data\_Train)

The Yacht\_NN1 is a list containing all parameters of the regression ANN as well as the results of the neural network on the test data set. To view a diagram of the Yacht\_NN1 use the plot() function.

plot(Yacht\_NN1, rep = 'best')

Diagram

Description automatically generated

This plot shows the weights learned by the Yacht\_NN1 neural network, and displays the number of iterations before convergence, as well as the SSE of the training data set. To manually compute the SSE you can use the following:

NN1\_Train\_SSE <- sum((Yacht\_NN1$net.result - Yacht\_Data\_Train[, 7])^2)/2

paste("SSE: ", round(NN1\_Train\_SSE, 4))

## [1] "SSE: 0.0361"

This SSE is the error associated with the training data set. A superior metric for estimating the generalization capability of the ANN would be the SSE of the test data set. Recall, the test data set contains observations **not** used to train the Yacht\_NN1 ANN. To calculate the test error, we first must run our test observations through the Yacht\_NN1 ANN. This is accomplished with the neuralnet package compute() function, which takes as its first input the desired neural network object created by the neuralnet() function, and the second argument the test data set feature (independent variable(s)) values.

Test\_NN1\_Output <- compute(Yacht\_NN1, Yacht\_Data\_Test[, 1:6])$net.result

NN1\_Test\_SSE <- sum((Test\_NN1\_Output - Yacht\_Data\_Test[, 7])^2)/2

NN1\_Test\_SSE

## [1] 0.008417631461

The compute() function outputs the response variable, in our case the Residuary\_Resist, as estimated by the neural network. Once we have the ANN estimated response we can compute the test SSE. Comparing the test error of 0.0084 to the training error of 0.0361 we see that in our case our test error is smaller than our training error.

Regression Hyperparameters

We have constructed the most basic of regression ANNs without modifying any of the default hyperparameters associated with the neuralnet() function. We should try and improve the network by modifying its basic structure and hyperparameter modification. To begin we will add depth to the hidden layer of the network, then we will change the activation function from the *logistic* to the *tangent hyperbolicus* (tanh) to determine if these modifications can improve the test data set SSE. When using the tanh activation function, we first must rescale the data from [0,1][0,1] to [−1,1][−1,1] using the rescale package. For the purposes of this exercise we will use the same random seed for reproducible results, generally this is not a best practice.

# 2-Hidden Layers, Layer-1 4-neurons, Layer-2, 1-neuron, logistic activation

# function

set.seed(12321)

Yacht\_NN2 <- neuralnet(Residuary\_Resist ~ LongPos\_COB + Prismatic\_Coeff + Len\_Disp\_Ratio + Beam\_Draut\_Ratio + Length\_Beam\_Ratio + Froude\_Num,

data = Yacht\_Data\_Train,

hidden = c(4, 1),

act.fct = "logistic")

## Training Error

NN2\_Train\_SSE <- sum((Yacht\_NN2$net.result - Yacht\_Data\_Train[, 7])^2)/2

## Test Error

Test\_NN2\_Output <- compute(Yacht\_NN2, Yacht\_Data\_Test[, 1:6])$net.result

NN2\_Test\_SSE <- sum((Test\_NN2\_Output - Yacht\_Data\_Test[, 7])^2)/2

# Rescale for tanh activation function

scale11 <- function(x) {

(2 \* ((x - min(x))/(max(x) - min(x)))) - 1

}

Yacht\_Data\_Train <- Yacht\_Data\_Train %>% mutate\_all(scale11)

Yacht\_Data\_Test <- Yacht\_Data\_Test %>% mutate\_all(scale11)

# 2-Hidden Layers, Layer-1 4-neurons, Layer-2, 1-neuron, tanh activation

# function

set.seed(12321)

Yacht\_NN3 <- neuralnet(Residuary\_Resist ~ LongPos\_COB + Prismatic\_Coeff + Len\_Disp\_Ratio + Beam\_Draut\_Ratio + Length\_Beam\_Ratio + Froude\_Num,

data = Yacht\_Data\_Train,

hidden = c(4, 1),

act.fct = "tanh")

## Training Error

NN3\_Train\_SSE <- sum((Yacht\_NN3$net.result - Yacht\_Data\_Train[, 7])^2)/2

## Test Error

Test\_NN3\_Output <- compute(Yacht\_NN3, Yacht\_Data\_Test[, 1:6])$net.result

NN3\_Test\_SSE <- sum((Test\_NN3\_Output - Yacht\_Data\_Test[, 7])^2)/2

# 1-Hidden Layer, 1-neuron, tanh activation function

set.seed(12321)

Yacht\_NN4 <- neuralnet(Residuary\_Resist ~ LongPos\_COB + Prismatic\_Coeff + Len\_Disp\_Ratio + Beam\_Draut\_Ratio + Length\_Beam\_Ratio + Froude\_Num,

data = Yacht\_Data\_Train,

act.fct = "tanh")

## Training Error

NN4\_Train\_SSE <- sum((Yacht\_NN4$net.result - Yacht\_Data\_Train[, 7])^2)/2

## Test Error

Test\_NN4\_Output <- compute(Yacht\_NN4, Yacht\_Data\_Test[, 1:6])$net.result

NN4\_Test\_SSE <- sum((Test\_NN4\_Output - Yacht\_Data\_Test[, 7])^2)/2

# Bar plot of results

Regression\_NN\_Errors <- tibble(Network = rep(c("NN1", "NN2", "NN3", "NN4"), each = 2),

DataSet = rep(c("Train", "Test"), time = 4),

SSE = c(NN1\_Train\_SSE, NN1\_Test\_SSE,

NN2\_Train\_SSE, NN2\_Test\_SSE,

NN3\_Train\_SSE, NN3\_Test\_SSE,

NN4\_Train\_SSE, NN4\_Test\_SSE))

Regression\_NN\_Errors %>%

ggplot(aes(Network, SSE, fill = DataSet)) +

geom\_col(position = "dodge") +

ggtitle("Regression ANN's SSE")

Chart, bar chart

Description automatically generated

As evident from the plot, we see that the best regression ANN we found was Yacht\_NN2 with a training and test SSE of 0.0188 and 0.0057. We make this determination by the value of the training and test SSEs only. Yacht\_NN2’s structure is presented here:

plot(Yacht\_NN2, rep = "best")

Diagram

Description automatically generated

We have looked at one ANN for each of the hyperparameter settings. Generally, researchers look at more than one ANN for a given setting of hyperparameters. This capability is built into the neuralnet package using the rep argument in the neuralnet() function. Using the Yacht\_NN2 hyperparameters we construct 10 different ANNs, and select the best of the 10.

set.seed(12321)

Yacht\_NN2 <- neuralnet(Residuary\_Resist ~ LongPos\_COB + Prismatic\_Coeff + Len\_Disp\_Ratio + Beam\_Draut\_Ratio + Length\_Beam\_Ratio + Froude\_Num,

data = Yacht\_Data\_Train,

hidden = c(4, 1),

act.fct = "logistic",

rep = 10)

plot(Yacht\_NN2, rep = "best")

Diagram

Description automatically generated

By setting the same seed, prior to running the 10 repetitions of ANNs, we force the software to reproduce the exact same Yacht\_NN2 ANN for the first replication. The subsequent 9 generated ANNs, use a different random set of starting weights. Comparing the ‘best’ of the 10 repetitions, to the Yacht\_NN2, we observe a decrease in training set error indicating we have a superior set of weights.

Wrapping Up

We have briefly covered regression ANNs in this tutorial. In the next tutorial we will cover classification ANNs. The neuralnet package used in this tutorial is one of many tools available for ANN implementation in R. Others include:

* nnet
* autoencoder
* caret
* RSNNS
* h2o

Before you move on to the [next tutorial](http://uc-r.github.io/ann_classification), test your new knowledge on the exercises that follow.

1. Why do we split the yacht data into a training and test data sets?
2. Re-load the Yacht Data from the UCI Machine learning repository yacht data **without scaling**. Run any regression ANN. What happens? Why do you think this happens?
3. After completing exercise question 1, re-scale the yacht data. Perform a simple linear regression fitting Residuary\_Resist as a function of all other features. Now run a regression neural network (see 1st Regression ANN section). Plot the regression ANN and compare the weights on the features in the ANN to the p-values for the regressors.
4. Build your own regression ANN using the scaled yacht data modifying one hyperparameter. Use ?neuralnet to see the function options. Plot your ANN.