Classification Artificial Neural Network

A picture containing diagram

Description automatically generatedClassification ANNs seek to classify an observation as belonging to some discrete class as a function of the inputs. The input features (independent variables) can be categorical or numeric types, however, we require a categorical feature as the dependent variable.

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

Replication Requirements

The following packages are required for classification ANN analysis.

library(tidyverse)

library(neuralnet)

library(GGally)

Data Preparation

Our classification ANN will use [Haberman’s Survival](http://archive.ics.uci.edu/ml/machine-learning-databases/haberman/haberman.data) data set from UCI’s Machine Learning Repository. Haberman’s data set was provided by Tjen-Sien Lim [email](mailto:stat.wisc.edu), and contains cases from a 1958 and 1970 study conducted at the University of Chicago’s Billings Hospital on the survival of 306 patients who had undergone surgery for breast cancer. We will use this data set to predict a patient’s 5-year survival as a function of their age at date of operation, year of the operation, and the number of positive axillary nodes detected.

We first download the data from UCI. When this data is imported, the Survival feature is imported as an integer, this needs to be a categorical logical value so we will modify this feature using themutate() function in the dplyr package. A value of 1 in the Survival feature indicates that the patient survived for at least 5 years after the operation, and a value of 0 indites that the patient died within 5 years.

url <- 'http://archive.ics.uci.edu/ml/machine-learning-databases//haberman/haberman.data'

Hab\_Data <- read\_csv(file = url,

col\_names = c('Age', 'Operation\_Year',

'Number\_Pos\_Nodes','Survival')) %>%

na.omit() %>%

mutate(Survival = ifelse(Survival == 2, 0, 1),

Survival = factor(Survival))

A brief examination of the data set…

ggpairs(Hab\_Data, title = "Scatterplot Matrix of the Features of the Haberman's Survival Data Set")

Diagram

Description automatically generated

shows that many more patients survived at least 5 years after the operation. Of the patients that survived (bottom-subplots of the Survival row in the Scatterplot Matrix), we see many of the patients have few numbers of positive axillary nodes detected. Examination of the Age feature shows a few of the most elderly patients died within 5 years, and of the youngest patients we see increased 5-year survivability. We forego any more detailed visual inspection in favor of learning the relationships between the features using our classification ANN.

As in the regression ANN, we must scale our features to fall on the closed $[0,1]$ interval. For classification ANNs using the neuralnet package we will not use a training and test set for model evaluation, instead we will use the **Akaike Information Criterion** ([AIC](https://en.wikipedia.org/wiki/Akaike_information_criterion)) and **Bayesian Information Criterion** ([BIC](https://en.wikipedia.org/wiki/Bayesian_information_criterion)) for final model selection. These metrics balance the error as a function of the total number of model parameters, in the ANN case the model parameters correspond to the total number of hidden nodes.

scale01 <- function(x){

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

}

Hab\_Data <- Hab\_Data %>%

mutate(Age = scale01(Age),

Operation\_Year = scale01(Operation\_Year),

Number\_Pos\_Nodes = scale01(Number\_Pos\_Nodes),

Survival = as.numeric(Survival)-1)

Classification ANNs in the neuralnet package require that the response feature, in this case Survival, be inputted as a Boolean feature. We modify the feature then run the initial classification ANN.

Hab\_Data <- Hab\_Data %>%

mutate(Survival = as.integer(Survival) - 1,

Survival = ifelse(Survival == 1, TRUE, FALSE))

1st Classification ANN

We construct a 1-hidden layer ANN with 1 neuron. The neuralnet package defaults to random initial weight values, for reproducibility we set a seed and construct the network. We have added three additional arguments for the classification ANN using the neuralnet package, linear.output, err.fct, and likelihood. Setting the linear.output to FALSE and the err.fct to “ce” indicates that we are performing a classification, and forces the model to output what we may interpret as a probability of each observation belonging to Survival class 1. For classification ANN the cross-entropy error metric is more appropriate than the SSE used in regression ANNs. The likelihood argument set to TRUE indicates to neuralnet that we would like to see the AIC and BIC metrics.

set.seed(123)

Hab\_NN1 <- neuralnet(Survival ~ Age + Operation\_Year + Number\_Pos\_Nodes,

data = Hab\_Data,

linear.output = FALSE,

err.fct = 'ce',

likelihood = TRUE)

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

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

A picture containing diagram

Description automatically generated

The error displayed in this plot is the cross-entropy error, which is a measure of the differences between the predicted and observed output for each of the observations in the Hab\_Data data set. To view the Hab\_NN1 AIC, BIC, and error metrics run the following.

Hab\_NN1\_Train\_Error <- Hab\_NN1$result.matrix[1,1]

paste("CE Error: ", round(Hab\_NN1\_Train\_Error, 3))

## [1] "CE Error: 0.009"

Hab\_NN1\_AIC <- Hab\_NN1$result.matrix[4,1]

paste("AIC: ", round(Hab\_NN1\_AIC,3))

## [1] "AIC: 12.017"

Hab\_NN2\_BIC <- Hab\_NN1$result.matrix[5,1]

paste("BIC: ", round(Hab\_NN2\_BIC, 3))

## [1] "BIC: 34.359"

Classification Hyperparameters

Classification ANNs within the neuralnet package require the use of the ce error. This forces us into using the default act.fun hyperparameter value. As a result we will only change the structure of the classification ANNs using the hidden function setting.

set.seed(123)

# 2-Hidden Layers, Layer-1 2-neurons, Layer-2, 1-neuron

Hab\_NN2 <- neuralnet(Survival ~ Age + Operation\_Year + Number\_Pos\_Nodes,

data = Hab\_Data,

linear.output = FALSE,

err.fct = 'ce',

likelihood =

TRUE, hidden = c(2,1))

# 2-Hidden Layers, Layer-1 2-neurons, Layer-2, 2-neurons

set.seed(123)

Hab\_NN3 <- Hab\_NN2 <- neuralnet(Survival ~ Age + Operation\_Year + Number\_Pos\_Nodes,

data = Hab\_Data,

linear.output = FALSE,

err.fct = 'ce',

likelihood = TRUE,

hidden = c(2,2))

# 2-Hidden Layers, Layer-1 1-neuron, Layer-2, 2-neuron

set.seed(123)

Hab\_NN4 <- Hab\_NN2 <- neuralnet(Survival ~ Age + Operation\_Year + Number\_Pos\_Nodes,

data = Hab\_Data,

linear.output = FALSE,

err.fct = 'ce',

likelihood = TRUE,

hidden = c(1,2))

# Bar plot of results

Class\_NN\_ICs <- tibble('Network' = rep(c("NN1", "NN2", "NN3", "NN4"), each = 3),

'Metric' = rep(c('AIC', 'BIC', 'ce Error \* 100'), length.out = 12),

'Value' = c(Hab\_NN1$result.matrix[4,1], Hab\_NN1$result.matrix[5,1],

100\*Hab\_NN1$result.matrix[1,1], Hab\_NN2$result.matrix[4,1],

Hab\_NN2$result.matrix[5,1], 100\*Hab\_NN2$result.matrix[1,1],

Hab\_NN3$result.matrix[4,1], Hab\_NN3$result.matrix[5,1],

100\*Hab\_NN3$result.matrix[1,1], Hab\_NN4$result.matrix[4,1],

Hab\_NN4$result.matrix[5,1], 100\*Hab\_NN4$result.matrix[1,1]))

Class\_NN\_ICs %>%

ggplot(aes(Network, Value, fill = Metric)) +

geom\_col(position = 'dodge') +

ggtitle("AIC, BIC, and Cross-Entropy Error of the Classification ANNs", "Note: ce Error displayed is 100 times its true value")

Chart, bar chart

Description automatically generated

The plot indicates that as we add hidden layers and nodes within those layers, our AIC and cross-entropy error grows. The BIC appears to remain relatively constant across the designs. Here we have a case where Occam’s razor clearly applies, the ‘best’ classification ANN is the simplest.

Wrapping Up

We have briefly covered classification ANNs in this tutorial. In future tutorials we will cover more advanced applications of ANNs. Also, keep in mind that the neuralnet package used in this tutorial is only one of many tools available for ANN implementation in R. Others include:

* nnet
* autoencoder
* caret
* RSNNS
* h2o

Before you move on, test your new knowledge on the exercises that follow.

1. Build your own classification ANN using the Hab\_Data data set.
2. The iris data set contains 4 numeric features describing 3 plant species. Think about how we would need to modify the iris data set to prepare it for a classification ANN. Hint, the data set for classification will have 7 total features.
3. R package nnet has the capacity to build classification ANNs. Install and take a look at the documentation of nnet to see how it compares with neuralnet.