The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project 8

Modeling the strength of concrete with Artificial Neural Networks

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Introduction

**Artificial Neural Network (ANN)**

It is a machine learning method which is based on the functioning and problem-solving capacities of the human brain. The model uses a number of nodes which are called neurons and each neuron solves a complex problem. Similar to the human neurons, the nodes receive an input signal which it solves and give an output signal which is then transmitted to a different neuron.

An artificial neural network learns the patterns between different factor affecting an outcome by adjusting the strength of connections between the neurons in an artificial neural network.

**Backpropagation Algorithm**

It is one of the algorithms which can be used to train an artificial neural network. As the name suggest, the algorithm uses a strategy called back- propagation error which involves a forward and a backward pass through the neural network.

The output is calculated during a forward pass and during the backward pass, the error between the actual and predicted output is calculated which is transmitted through the network. The partial derivative of the error calculated during the backward pass is used to update the weights using algorithm such as gradient descent. Updated weight is the used for next forward pass. The process will repeat itself till the error is minimized.

This is a key algorithm and is widely used in training artificial neural networks which resulted in the use of multi-layered feedforward networks which are widely used in the field of datamining today.

Problem Description

The world today is rapidly growing, especially in terms of its architecture and the sheer scale of the modern-day buildings. The one component which is being used in this process for a long time is concrete, which could be used as a building agent or a cementing agent to bind different components together. It is the concrete which determines the strength of a building which is directly proportional to the strength of the concrete.

Different compositions of concrete exist and is used for different types of building projects. Since concrete is a mixture of several components, it was found that concrete performance varies greatly due to different types of ingredients that interact with each other.

Hence with the use of a machine learning model such as artificial neural network, we can find or predict the concrete strength of a final concrete mixture based on listing of compositions of input material. This would lead to the production of optimum quality of concrete which would lead to safer construction practices and safer result of construction projects.

Objectives:

The primary goal of this report is to:

* To comprehensively introduce Backpropagation Algorithm, explaining its foundations and applications in identifying a concrete strength given its input materials.
* Outline the method involved in building and training the model.
* Steps to reduce errors and to improve the model.
* Propose recommendations for future research and development in this field.

Method

This project will be using the dataset “concrete.csv.” The primary source of literature used is “Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)”.

Listed below are the steps taken in the report:

* Step 1 – Collecting data.
* Step 2 – Exploring and preparing the data.
* Step 3 – Training a model on the data.
* Step 4 – Evaluating model performance.
* Step 5 – Improving model performance.

Steps taken:

**Step 1 – Collecting data.**

The data used in this project is a real-world data of concrete compressive strength which was donated to the UCI machine learning Data repository. The data has 1030 examples of concrete samples with a total of eight features.

A picture containing text, screenshot, font

Description automatically generated

**Step 2 – Exploring and preparing the data.**

We will begin by loading the data using the read.csv() function and will name the data “con”. We will use the str() command to check the data structure. We see that there are a total of nine variables out of which eight are features and one is the outcome we expected, which in this case would be the strength of concrete.

We realize through looking at the data structure that the data ranges anywhere from zero to thousands. In case of a neural network, it functions best or give the best results when the input data is in a narrow range.

To solve this issue, we will rescale the data. We would do this in the R using the normalize() function to normalize the data to a 1 – 0.

A picture containing text, font, screenshot

Description automatically generated

We would then apply it to every column of the data frame using lapply() command. We can then confirm the process of normalization by using the summary() command to see the maximum and minimum strength before and after the normalization. We see that the strength is now between 0 to 1 when compared to the original values which were 2.33 and 82.60.

A picture containing text, font, screenshot, line

Description automatically generated

The CSV file is already in a random order. Hence, we can go ahead and split the data into training and testing sets. In this case, we used a ratio of 80 to 20 for the split. That is 824(80%) of data for the training set and testing set is 206(20 %) of the data.

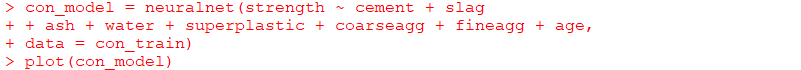


**Step 3 – Training a model on the data.**

We will be using a multilayered feedforward neural network to establish a relationship between the strength of concrete and its components. We will install and load the neuralnet package to implement this network.



First, we will initiate the training with only one hidden layer node in a multilayred feedforward network which we define as con\_model. We can visualize the network using the plot() command



We see in the plot which shows the eight features as the input nodes with a single hidden and output node predicting the strength of the concrete. The weight of each node is the number mentioned as the bias terms, which are numeric constants which determined the nodes movement. We are also presented with two numerical figures which are the training steps and the Error(sum of squared error) which is the squared aggregate of predicted minus actual values. A lower SSE (Error)indicates a better performance.

A screenshot of a computer

Description automatically generated

**Step 4 – Evaluating model performance.**

The plot or network topology diagram provides only little insight into the black box but no information of how well the data fits. We will use the compute() function to produce predictions on dataset. Compute() would provide us with a list of two components which are $neurons and $net.result which stores neurons for each layer and stored predicted values respectfully.



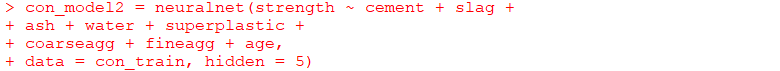
In this case, as the data is numeric in nature, we would use the correlation , cor() function to measure the distance between the concrete strength and true value



The correlation in this case is roughly 0.792. A correlation close to 1 indicates strong linear relationship which is the case here. This suggests the model performed well even with a single hidden node.

**Step 5 – Improving model performance.**

We can make the model more complex by simple increasing the number of hidden nodes which will result in more complex topologies making it capable of learning more difficult concepts.



We use the plot() command to visualize the new topology network which is significantly different than what we had before.



A screenshot of a computer

Description automatically generated

We note that the error is now significantly reduced from 5.29 from the previous model to 1.98. Also, the number of training step also increased drastically because of the model becoming more complex. We would again use the cor() function to check the correlation between the predicted and true value. We see that it has considerably improved from 0.79 to 0.93.

A picture containing text, font, screenshot

Description automatically generated

We can repeat the same process using a different number of hidden nodes. For example, the following shows the same process with a total of six hidden nodes.

A screenshot of a computer

Description automatically generated

A picture containing text, font, screenshot, line

Description automatically generated

The model is now visually more complex and shows six hidden nodes in the network. We see that the number of training step have significantly increased which should come as no surprise as there are a greater number of hidden nodes now compared to the previous model. We also see that the error is now slightly reduced from 1.98 to 1.81.

The major point to consider is that the corelation between the predicted and true value is now less than what we had with five hidden nodes( from 0.93 to 0.92). This proves that there is a limit to the number of hidden nodes to be used for an effective model.

Conclusion

From this report we conclude the following:

* Neural network is a great machine learning model to understand relationship between several features.
* We learn that the neural network needs the data to be normally distributed to give effective results.
* We can change the number of hidden nodes.
* The plot of a neural network provides us with the number of training steps involved along with the sum of squared error.
* The greater the number of hidden nodes is there the more will be the number of training steps with a lowered sum of squared errors because of more complex model with better performance.

Limitations:

Some limitations of the backpropagation algorithm are:

* The resultant model generated through this algorithm is a complex Blackbox model which is difficult to interpret and provide low insight.
* The model is very prone to overfitting the training data.
* This model is known for its computational intensity and slow nature especially when the number of hidden nodes is high.
* The model is very sensitive to the initial values of the weight which can result in varied solutions.

Coding

> con = read.csv("concrete.csv")

> str(con)

> normalize = function(x) {

+ return((x - min(x)) / (max(x) - min(x)))

+ }

> con\_norm = as.data.frame(lapply(con, normalize))

> summary(con\_norm$strength)

> summary(con$strength)

> con\_train = con\_norm[1:824, ]

> con\_test = con\_norm[825:1030, ]

> install.packages("neuralnet")

> library("neuralnet")

> con\_model = neuralnet(strength ~ cement + slag

+ + ash + water + superplastic + coarseagg + fineagg + age,

+ data = con\_train)

> plot(con\_model)

> model\_results = compute(con\_model, con\_test[1:8])

> predicted\_strength = model\_results$net.result

> cor(predicted\_strength, con\_test$strength)

> con\_model2 = neuralnet(strength ~ cement + slag +

+ ash + water + superplastic +

+ coarseagg + fineagg + age,

+ data = con\_train, hidden = 5)

> plot(con\_model2)

> model\_results2 = compute(con\_model2, con\_test[1:8])

> predicted\_strength2 = model\_results2$net.result

> cor(predicted\_strength2, con\_test$strength)

Reference

* Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)