Neural Network Machine Learning Analysis for Noisy Data: R Programming

Yagyanath Rimal*
Department of Science and Technology, Pokhara University, Lekhnath, Nepal

Abstract
This review paper clearly discusses the compression between Neural Network Machine Learning Analysis for Noisy Data: R Programming. Although there is large gap between data analysis to analyze overfitting and multicollinearity problems in data sets. Its primary purpose is to explain the machine learning procedures using neural network whose data structure were cross validation using R software whose outputs were sufficiently explain with various intermediate output and graphical interpretation to reach the conclusion. Therefore, this paper presents easiest way of machine learning analysis when data sets with multicollinearity and its strengths for data analysis using R programming.

Keyword: Hidden Neuron, Neural Network, over fitted Data, Rectified Linear Unit, Multi-Layer Perceptron

*Author for Correspondence E-mail: rimal.yagya@gmail.com

INTRODUCTION
The neural network was a versatile application in the modern world, whose design was highly applicable in automatic automobiles, in the processing of medical image recognition and in artificial intelligence that is based on the database algorithm at a time called neural network artificial [1, 2]. The neural network is the science of neuronal study similar to the neural system of the human body, in which every neuron is connected to dendrites that transmit signals to another neuron in the form of electrical impulse. The body refers to those signals and decides which special actions should be taken. for such cases. The artificial neuron is the heart of the neural network. The activation node takes input from the dendrites and then the activation neuron performs a competitive probability action [3]. Where the multiple input process produces and produces a single output where all input signals are analyzed progressively until the appropriate decision is made. The various terms of input as variables whose magnitude is considered as a certain weight denoted as (w) are passed to the summary function which already has some fixed distortion, the bias always evaluates and produces both true and false judgment. After obtaining zero or an activation function, the cap is calculated, which is further tested with the real prediction with the value y. This process is continuously tested with binary; categorical or numerical values of the search variables, this process is known as single-level perception using a single node [4]. Basically, there are three parts of the neural network: input levels, hidden levels and output levels. Input levels always take the input of variables and output levels always output. The middle layer has all the powers to analyze data that takes all the input and produces output for output levels. The terms of the input level as a subscript x (1, 2, 3, 4 ... m) and hidden layer h (1, 2, 3 ... n) could easily be determined during the design of the model formulation. If a network has multiple hidden levels, it is known as deep learning. Deep learning is a machine learning technique that teaches the computer to do what is natural for human learning through examples taken from previous recordings [5]. Therefore, a neural network is a hardware and/or software system modeled after the operation. There are several activation functions, the most popular is the threshold or pass function that passes 1 if the activation with bios is greater than zero, otherwise it goes to zero. The sigmoid or logistic (1/1 + ex) function is a widely used...
activation function that is best for predicting probability. Another is the hyperbolic tangent which is based on the sigmoid but which goes beyond -1 to 1, is similar to the regression values in which the gradient is deeper and \(\frac{(ex-e-x)}{(ex-e-x)}\). The rectified linear unit (RELU) is the best version of the neural network model; it also produces 1 if \(x\) is + ve; otherwise, it produces 0, which is popular due to a less expensive architecture and a faster approach, but requires more iteration of experiments. The desired output activity is achieved only when its errors from the previous y-values return to the neural function, so that errors are minimal in each pass [6]. The next extension is the input error feedback process so that its weight is greatly reduced for subsequent propagation, so the network could easily have adjusted its weight at each iteration so as to obtain minimal errors (errors = errors - 1). \(J (\theta) = 1/m \sum m_i (z_m - y_i) \), this requires more computing power, use the descent of the systematic gradient which finally finds the minimum learning rate. The machine learning process begins with data observations, direct experiences or instructions to predict models in the data and make better decisions in the future. The main goal is to enable computers to learn automatically without human intervention or assistance and to adjust automatically actions [7].

The large data contain the high social, economic and scientific value that has existed throughout the academic and industrial world. Big data in real value, but there are four attributes specific to big data, known as volume, variety, speed and value. Extend the value space of big data. Meanwhile, they become big challenges for big data analysis. There are three main problems: Big data from multiple and heterogeneous sources, storage of huge unstructured data and big data of great value. The three central scientific problems in the analysis of big data are the representation, archiving and forecast of big data. Traditional methods cannot handle big data. New methods for big data are indispensable. Human brains are, of course, excellent big data processors. Neural networks are computational replicas of human data. The history of research in neural networks has ups and downs. Nowadays, with the support of the development of computational power, the analysis of big data using neural networks has achieved great success, especially in big data applications, for example, big data analysis, big data analysis, large medical analysis data. Neural networks are conducting investigations on artificial intelligence. On the one hand, neural networks are able to extract summaries. They can combine multiple sources of information, process heterogeneous data and acquire dynamic changes. They are the bridge for the implementation of the transformation of the value of big data. Extraordinary training samples that allow the training of neural networks with a large number of parameters. However, there are still some problems in the neural network model. In the aspect of neuronal network research, the structure is further research and development; The scale of the network lacks the theoretical model; and the learning algorithm is having some inherent problem [8]. In the big data aspect, there are three central scientific problems to ensure coherence in the high-dimensional dispersed space; implement storage space; and to represent the temporal correlation and implementation of the big data forecast. Many other investigations are urgent in this area, including those of theoretical and practical aspects. In particular, interdomain surveys are important. Research in the area can act in coordination with that to understand the processing of large data in the human brain. Neurosciences and neurosciences are necessary to solve the fundamental scientific problems of neural networks and the investigation of large volumes of data, in which analysis of analysis of large amounts of data could be improved [2]. The neural network is a hardware and/or software system modeled on the functioning of neurons in the human brain. An artificial neural network models synapses and biological neurons can also be used to make predictions for complex neural networks also called artificial intelligence networks or IA datasets [9]. Neural networks and their associated algorithms are the most interesting of all machine-learning techniques. The output layer of the neural network collects and transmits the information for which it was designed. The model presented by the output level can be traced directly to the input level. The number of neurons in the output layer must be directly
related to the type of work performed by the neural network. To determine the number of neurons in the output layer, first consider the intended use of the neural network [10]. In a direct power supply network, input moves from input nodes, through hidden nodes (if any), and to output nodes. There are no loops or loops in the network. The number of neurons in the input layer depends on the training, given the number of hidden layers and the number of neurons in each layer, which are generally determined by a cross-validation methodology [1]. The most common neural network model is the perceptron multilayer (MLP). This type of neural network is known as a supervised network because it requires the desired output for learning. The goal of this type of network is to create a model that correctly assigns input to output using historical data, so that the model can be used to produce output when the desired output is unknown. Neural networks allow a fast and efficient way to model phenomena at different levels of complexity [11]. They can be used in both well-structured and poor problematic processes, whose knowledge is limited. The advantages of neural networks have been transferred to the possibilities of defining probability distributions of individual activities in cyclical construction processes. The additional hidden levels through which errors must propagate backwards make the gradient more unstable and the number of false minimums increases. In addition, excessive adjustments may occur when the training sets are small in relation to the number of hidden neurons; the size of the training set and the dimensions of the hidden layer are merged. Although dynamic networks can be trained using the same gradient-based algorithms used for static networks, the performance of algorithms in dynamic networks can be very different and the gradient must be calculated in a more complex way [12].

**Neural Network Using R Programming**

```r
> getwd()
[1] "C:/Users/rimal/OneDrive/Documents"
> getwd()
[1] "C:/Users/rimal/OneDrive/Documents"
> data = read.csv("binray.csv", header = TRUE)
> data = personal data
> str(data)
'data.frame': 400 obs. of 4 variables:
$ admit: int 0 1 1 0 1 0 1 0 1 1 ...
$ gre: int 380 660 800 640 520 760 560 400 540
$ gpa: num 3.61 3.67 4 3.19 2.93 3 2.98 3.08
Range of $: int 3 3 1 4 4 2 1 2 3 2 ...
```

Here, I am using binary data frame having 400 records with 4 variables. The admit variable having properties 0 or 1 whether student will have admitted or not admitted whose decision is based on other three variables gre, gpa and rank of student score where those former school were ranked from 1 to 4 as Likert scale. While doing neural network the prediction variable should be in categorical and all other values should be lies in-between 0 to 1 this could achieve by deducting minimum and dividing max by minimum values (Figures 1 and 2).

![Histogram of data$gre](image)

**Fig. 1: Graph of Neural Network Prediction.**
Fig. 2: Values Arrived by Deducting Minimum and Dividing max by Minimum Values.

```r
> hist(data$gre)
> data$gre=(data$gre-min(data$gre))/(max(data$gre)-min(data$gre))
> data$gpa=(data$gpa-min(data$gpa))/(max(data$gpa)-min(data$gpa))
> data$rank=(data$rank-min(data$rank))/(max(data$rank)-min(data$rank))
> hist(data$gre)
> set.seed(222)
> ind=sample(2,nrow(data),replace=TRUE,prob=c(.7,.3))
> training=data[ind==1,]
> testing=data[ind==2,]
```

The data partitioning creates training data set separate 281 observation of 4 variables similarly testing creates 119 observation of 4 variables. Which is fundamental idea of machine learning where we design model from training data sets and tested with testing data sets for reliability of prediction in neural network.

```r
> library(neuralnet)
> set.seed(333)
> n <- neuralnet(admit~gre+gpa+rank, + data = training, + hidden = 1, + err.fct = "ce", + linear.output = FALSE)
```

Here we are creating neural network model with admit dependent variable of other three independent variable on training data sets where hidden neuron is 1 and error term is ce and linear output is false.

```r
> plot(n)
```

From the Figure 3, three gre, gpa and rank are the predictor variable input layer neuron are completely connected with 1 and two hidden layers with its weight and bias terms and admit is predicted variables whose values were conversed in 11197 for single hidden layer and 5811 in two neurons in first hidden layers.

```r
> output=compute(n,training[,1])# first column admit is excluded
> head(output)
$'neurons'
$'neurons' [[1]]
   l gre  gpa  rank
1 1 0.2758620690 0.77586206897 0.6666666667
2 5 0.5172413793 0.38505747126 1.0000000000
3 6 0.9310344828 0.42528735632 0.3333333333
4 1 0.4482758621 0.20689655172 0.3333333333
5 385 1 0.4482758621 0.20689655172 0.3333333333
6 387 1 0.8965517241 0.91954022989 0.3333333333
7 389 1 0.7241379310 0.52298850575 0.3333333333
```

Here we get so many output the neural network output predict that the second items of data sets has 337 percent of chance of being selected.
admission but normal data head of training data categorized as completely admitted is misclassification of model.

**Output Calculation**

From the above neural network figure, we can calculate output of each neuron. The hidden neuron input is calculated with sum of bias term and the product of each input variable weight and its input values of three input variables (Figure 4).

\[
> \text{in}4 <- 0.0455 + (0.82344 \times 0.7586206897) + (1.35186 \times 0.8103448276) + (-0.87435 \times 0.6666666667)
\]

\[
> \text{in}4 \quad [1] 1.182751379
\]

This is the input to hidden layer whose output is calculated with sigmoid function which is calculated with \(1/(1+\text{exponent of previous input})\). This process is repeated until final neuron value is calculated. The back prorogation again calculated in reversed manner with adjusting its weight from each final convergence output.

\[
> \text{out}4 <- 1/(1+\exp(-\text{in}4))
\]

\[
> \text{out}4 \quad [1] 0.7654421484
\]

\[
> \text{in}5 <- -7.06125 +(8.5741*\text{out}4)
\]

\[
> \text{in}5 \quad [1] -0.4982724753
\]

\[
> \text{out}5 <- 1/(1+\exp(-\text{in}5))
\]

\[
> \text{out}5 \quad [1] 0.3779467293
\]

This value is similar to output of Net. Neuron as above. Confusion matrix and miss classification errors.

\[
> \text{output=compute(n,training[,-1])}
\]

\[
> \text{p1=output$net.result}
\]

\[
> \text{pred1=ifelse(p1>=0.5,1,0)}
\]

\[
> \text{tab1=table(pred1,training$admit)}
\]

\[
> \text{tab1}
\]

\[
\begin{array}{ccc}
0 & 181 & 57 \\
1 & 8 & 35
\end{array}
\]

```r
fig3 <- read.graph("graphs\fig3\firstpredictablevalueofgre_gpa_rank.png")
plot(fig3)
```

**Fig. 3:** First Predictable Value of gre, gpa, Rank.
From the above output, values is calculated with excluding admit variable and result is stored in another p1 variable (Figure 4). The prediction is calculated on the basic of if the compute output is greater than 0.5 then his/her result is admitted (1) otherwise non-admitted. From the output of confusion matrix 181 persons were not admitted expressed by both model and data interoperation where 57 and 8 data items were miss classified.

Similarly, there is 78+6 is accurate classification but 29+6 were the miss classification of testing data

> 1-sum(diag(tab2))/sum(tab2)
[1] 0.2941176471 is miss classification errors.

Let’s put five hidden neuron in the model

> n <- neuralnet(admit~gre+gpa+rank, +                data = training, +                hidden = 5, +                err.fct = "ce", +                linear.output = FALSE)
> plot(n)

This is completely connected network where each layer nods were connected with other layers but not connected within each node whose output weight were shown. This network converse in 12565 reputations whose errors is 142.09

> pred1 0 1
  0 78 29
  1 6 6

[1] 0.2491103203
The miss classification and errors have decreased than previous model.

Two hidden neurons

```r
> n <- neuralnet(admit~gre+gpa+rank,
+ data = training,
+ hidden = c(2,3),
+ err.fct = "ce",
+ linear.output = FALSE)
```

Which produces error message could not converge due to divergence of model (Figure 5–8).

```r
> n <- neuralnet(admit~gre+gpa+rank,
+ data = training,
+ hidden = c(2,1),
+ err.fct = "ce",
+ linear.output = FALSE)
```

Plot(n)

pred1 0 1

1000 min thresh: 0.1286597328
2000 min thresh: 0.1286597328
3000 min thresh: 0.08483519612
17000 min thresh: 0.0107039752
18000 min thresh: 0.0107039752
error: 132.5507 time: 4.81 secs
hidden: 5 thresh: 0.01 rep: 2/5 steps:
Could not converge
hidden: 5 thresh: 0.01 rep: 3/5 steps:
1000 min thresh: 0.09224424645
2000 min thresh: 0.09224424645
13000 min thresh: 0.01087917161
error: 139.80883 time: 3.4 secs
hidden: 5 thresh: 0.01 rep: 4/5 steps:
1000 min thresh: 0.1472573813
2000 min thresh: 0.1472573813
3000 min thresh: 0.09138904351
13000 min thresh: 0.01091560314
14000 min thresh: 0.01091560314
error: 145.41304 time: 3.9 secs
hidden: 5 thresh: 0.01 rep: 5/5 steps:
1000 min thresh: 0.1131700352
2000 min thresh: 0.102054191
9000 min thresh: 0.01150909692
9000 min thresh: 0.01150909692
10000 min thresh: 0.01012525328
error: 145.64415 time: 2.6
```

From the above output except 2nd reputation all network was converged with step with different error and time taken (Figure 6), this could conclude that the first neuron is best suited because of error: 132.5507 time: 4.81 secs least.

```r
> output=compute(n,training[,-1],rep=1)
> pred1=ifelse(p1>=0.5,1,0)
> tab1=table(pred1,training$admit)
> tab1
```

| pred1 | 0 | 1 |
|-------|---|---|
| 184   | 25 |
| 67    | 5 |

```r
> 1 - sum(diag(tab1))/sum(tab1)
[1] 0.256227758
```

For test data

```r
> output=compute(n,testing[,-1],rep=1)
> pred2=ifelse(p2>=0.5,1,0)
> tab2=table(pred2,testing$admit)
> tab2
```

| pred2 | 0 | 1 |
|-------|---|---|
| 79    | 23 |
| 25    | 15 |

```r
> 1 - sum(diag(tab2))/sum(tab2)
[1] 0.2352941176
```

The confusion matrix and errors of misclassification shows more than above models.

With five neurons

```r
> n <- neuralnet(admit~gre+gpa+rank,
+ data = training,
+ hidden = 5,
+ err.fct = "ce",
+ linear.output = FALSE,
+ lifesign='full',
+ rep=5,
+ algorithm = "rprop+",
+ stepmax = 100000)
```

Output hidden: 5 thresh: 0.01 rep: 1/5 steps:

Could not converge
hidden: 5 thresh: 0.01 rep: 2/5 steps:
hidden: 5 thresh: 0.01 rep: 3/5 steps:
hidden: 5 thresh: 0.01 rep: 4/5 steps:
hidden: 5 thresh: 0.01 rep: 5/5 steps:
```r
From the above output except 2nd reputation all network was converged with step with different error and time taken (Figure 6), this could conclude that the first neuron is best suited because of error: 132.5507 time: 4.81 secs least.

```r
> p2=output$net.result
```

> pred2=ifelse(p2>=0.5,1,0)
> tab2=table(pred2,testing$admit)
> tab2
```

| pred2 | 0 | 1 |
|-------|---|---|
| 79    | 23 |
| 25    | 15 |

```r
> 1 - sum(diag(tab2))/sum(tab2)
[1] 0.2352941176
```

The miss classification and errors has decreased than previous model.

Two hidden neurons

```r
> n <- neuralnet(admit~gre+gpa+rank,
+ data = training,
+ hidden = c(2,3),
+ err.fct = "ce",
+ linear.output = FALSE)
```
Fig. 5: Error: 142.09957 Steps: 12565.

Fig. 6: Error: 156.179136 Steps: 5987.
Fig. 7: Error: 132.550704 Steps: 18339.

Fig. 8: Output Graph.
Therefore, neural network could analyze the categorical predictive variable on depends on various numeric variables with based on various single or more hidden with rprop+ algorithm, rep life sign and many more assumptions (Figure 7).

CONCLUSION
In this work, I first studied the neural network algorithm for data analysis. Simulation of neural networks in neural networks. Although, the neuronal network has less interpretability than the decision tree. But it is more suitable for noisy datasets for unrecognized patterns. An R regression analysis was performed to measure the correlation between results and objectives. We create a regression graph to validate the network that shows the relationship between network outputs and goals. It is observed that the value of R is closer to 1. Neural networks are widely used in pattern recognition due to their ability to respond to unexpected inputs/patterns. During training, neurons are taught to recognize different specific patterns and to be or not shoot when that model is received.

REFERENCES
1 Bouziane A. (2018). Neural network analysis. Online.
2 Chetan Sharma. (2014). Big data analysis using neural networks. Master Thesis, San Jose State University, San Jose, California.
3 Gurney K. (1997). An introduction to neural network. CRC Press, 1st Edition, 1997, London.
4 Haluk D. (2015). Innovations with smart service systems: Analytics, big data, cognitive assistance, and the internet of everything. Communications of the Association for Information Systems, 37(1), 733-752.
5 Johnson S. (2004). Neural coding strategies and mechanisms of competition. Cognitive systems research. Neural coding strategies and mechanisms of competition, 5(2), 93-117p.
6 Lim S. (2018). Adaptive learning rule for hardware based deep neural networks. Neural and Evolutionary Computing, 31(11), 8101-8116p.
7 McAteer M. (2018). An introduction to probabilistic programming, now available in Tensor Flow probability. [Online] available from: https://blog.tensorflow.org/2018/12/an-introduction-to-probabilistic.html [Online Access] September 2019.
8 Noon H. (2013). Artificial neural network: Beginning of the ai revolution. [Online] Available from: https://hackernoon.com/artificial-neural-network-a843ff870338 [Accessed Dec. 2019].
9 Ch. Mamatha, P. Buddha Reddy, M.A. Ranjit Kumar, Shrawan Kumar. Analysis of Big Data with Neural Network (2017). International Journal of Civil Engineering and Technology, 8(12), 211-215p.
10 Stefan Kojouharov, Downloadable: Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Data Science PDF (2019). [Online] Available from: https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-science-pdf-f22dc900d2d7 [Accessed Dec. 2019].
11 Shwe M. (2018). An introduction to probabilistic programming, now available in Tensor Flow probability. Retrieved from https://medium.com/TensorFlow/an-introduction-to-probabilistic-programming-now-available-in-tensor-flow-probability. [Accessed Dec. 2019].
12 ZHANG Yi, GUO Quan, WANG Jianyong. Big Data Analysis Using Neural Networks[J]. Advanced Engineering Sciences, 2017, 49(1):9-18p.

Cite this Article
Yagyanath Rimal. Neural Network Machine Learning Analysis for Noisy Data: R Programming. Recent Trends in Programming Languages. 2019; 6(3): 1–10p.