CSCI447 - Assignment 3

Chris Major CHRISMICHELMAJOR@HOTMAIL.COM
Farshina Nazrul-Shimim FARSHINA287LEO@GMAIL.COM
Tysen Radovich RADOVICH.TYSEN@GMAIL.COM
Allen Simpson ALLEN.SIMPSON@MOTIONENCODING.COM

Editor: (none)

Abstract

This project involves the implementation of two neural network training algorithms, namely 'multi-layer feedforward network with backpropagation' and 'radial basis function (RBF) neural network' to perform two specific tasks - classification and regression on some particular set of examples. The data sets used in the experiment were provided by the UCI Machine Learning Repository and divided into training and testing subsets through the 10-Fold cross-validation technique. In case of the feedforward neural network the number of inputs, hidden layers, hidden nodes in each layer, and outputs were chosen depending on the structure of each data set. From the results of the edited nearest neighbor, condensed nearest neighbor, k-means clustering, and k-medoids clustering from the previous experiment, four different versions of the RBF network was tested. Though the RBF neural network failed to provide meaningful results, the functionality of the feedforward net was demonstrated.

1. Introduction

Neural networks (NN) are a series of computational units, known as "nodes," that are modeled to recognize patterns, as described by Hertz et al. (1991). In theory, these networks resemble the biological ability of the brain to learn and perform specific tasks through examples and patterns. If configured correctly, they can serve as a means of classification or regression and learn from incorrectly performed tasks.

This paper describes a series of experiments conducted on several data sets, in the interests of classification and regression with neural networks. The first is a feedforward network with backpropogation; the second is a radial basis function network. Both networks were designed, tuned, and run on six selected datasets from the UCI Machine Learning repository, per Dua and Graff (2017). The results are listed and analyzed in this document.

2. Problem Statement

Neural networks are also known as "universal approximators" as they have the ability to learn and model non-linear and complex functions. Therefore, they can aid greatly in solving classification and regression problems. In the design of a neural network, many factors play important roles in determining the success of operation: the dataset size, pre-processing, and proper selection of kernel functions.

For proper training, a NN requires an optimal range of data and size of dataset, with the goal of convergence under the gradient descent algorithm. A data set that is too small can restrict performance and result in poor representation due to noise, unwanted effects from outliers, and misrepresentation of the set's true distribution, amongst other negative outcomes. On the other hand, large data sets are resource-expensive, can affect run-time performance of the algorithm, and can result in possible over-fitting of data.

Parameter specification and tuning are components of the experiment that greatly affect the accuracy of the model to a great extent. Use of Kernel functions in data-projection onto the higher dimensions usually results in sparsity and hence the same problems are faced as the large data sets. This is commonly refered to as the "curse of dimensionality" and is not a problem unique to NN algorithms.

Thus, we hypothesize that the feedforward neural network performs better with the classification as in case of classification data, the number of nodes of the output layer is more than that of the regression data, which subsequently improves the performance of backpropagation algorithm of the neural network. Another hypothesis involves with the linear separability of the data. The feedforward neural network performs poorly both in case of linearly inseparable and overlapped data type.

2.1 Data Sets

`abalone`, `car`, and `segmentation` are classification data sets, with each observation linked to a specific class. The regression data sets include `machine`, `forestfires`, `winequality-red`, and `winequality-white`.

All data set values are floating-point values, save for `foresfire`'s integer and date values as well as `car` and `machine`'s categorical vendor and model values. There are no missing values in any of these data sets. Both `winequality-red`, and `winequality-white` data sets were combined for the sake of convenient reference.

3. Experiments

The data sets were divided into training and testing sets and evaluated through 10-Fold Cross Validation, as described by Dietterich (1998). Each individual data set is divided into 10 equally-portioned subsets, which alternate between training the algorithms and testing the algorithms. This method provides an acceptably thorough analysis of each algorithm's performance, of which the results are discussed in the Results and Analysis section.

3.1 Feedforward Neural Network

A NN consists of an input layer, $\mathcal{X}$, an arbitrary amount of hidden layers $\mathcal{H}$ and an output layer $\mathcal{Y}$. Between each layer is a set of weights and biases $\mathcal{W}$ and $\mathcal{B}$.

For each of the datasets, the input layer represents the multi-dimensional data. The number of hidden layers varies depending on the trial of the experiment, per Kulkarni and Harman (2011). The activation value for any node in the hidden and output layer can be expressed as:

$$a_i^l = \sigma\left(\sum_k w_{ik}^l a_k^{l-1} + b_i^l\right)$$

Which shows the relation of the activation a_i^l of the i^{th} neuron in the l^{th} layer to the activations in the $(l-1)$th layer.

An activation function α for each node in the hidden layer is used to determine the "firing", or activation, of that node. For this experiment, a logistic function - a special type of Sigmoidal activation function - has been used. This function and its derivative are described as follows:

$$y = \frac{1}{1+e^{-x}}$$

$$y' = y(1-y) = \left(\frac{1}{1+e^{-x}}\right) - \left(\frac{1}{1+e^{-x}}\right)^2$$

The output layer is the predicted class of that data in case of classification data set, and the value of the regression function in case of the regression data set. The output layer has only one node which contains the value n.

3.2 Radial Basis Function Neural Network

Radial Basis Function (RBF) neural networks overcome the limitations of linear inseparability of the input dataset by using a single hidden layer to project the input into a higher dimensional space with the help of a Kernel function, per Orr (1996). In this experiment, a Gaussian basis function has been used as the activation function for the hidden layer nodes. The Gaussian basis function is expressed as:

$$K_g(x_i, x_j) = e^{-\frac{(x_i - x_j)^2}{2\sigma^2}}$$

Here, selection of standard deviation σ and receptor x_j determines the shape of the function. A significantly small σ can result in over-fitting and vice-versa. The output layer is obtained using a linear activation function with the hidden layer nodes, per the description of Broomhead and Lowe (1988).

3.3 Backpropagation

Backpropagation is an algorithm for the neural network to train its weight and bias for each activation function with the help of gradient descent. Given a set of input values, the network gives an output using the its activation functions. The predicted output for each data is compared with the actual output using an error function. The objective of the training is to minimize this error function after each run.

Backpropagation identifies the change of the output function with respect to the weight and bias of the combination of the input and the hidden layers, as described by Rumelhart et al. (1985). This makes the function highly nonlinear.

An important concept that affects the convergence of this algorithm is momentum, as described by Sarigül and Avci (2017). Momentum is the ability for the neural network to push the process away from local minima by referring to previously calculated weight values. The equation for a momentum-affected weight Δw_{ji} is as follows:

$$\Delta w_{ji}^{t} = -\eta \left(\frac{\delta Err}{\delta w_{ji}} \right) + \alpha \Delta w_{ji}^{t-1}$$

Here, t represents the current set of weights being calculated and $t-1$ refers to the previous set of weights previously calculated. This feature is implemented within the feed-forward neural network's function call, allowing the user to turn momentum on or off for a given experiment.

3.4 Loss Functions

Two loss functions are considered for these experiments - 0/1 Loss Robert (2007) and Mean-Square Error (MSE) Sai et al. (2009). In the cases of classification, 0/1 Loss assigns a 0 value for incorrectly classified objects and a 1 value for correctly classified objects, providing a measure of accuracy for the classification algorithms.

In the cases of regression, the mean-square error is used to calculate the error in the regression analysis. The MSE can be expressed as:

$$MSE = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y_j})^2$$

Here, $\hat{y_j}$ is the predicted value of output y_j. In this experiment, the output is the value of the regression function of the input data.

3.5 Tuning

The tuning parameter that required the most attention is the number of nodes in each hidden layer, as this value most significantly affected the algorithms' performance. For this experiment, the number of nodes in the hidden layer is less than that of the input layer. Two parameters has been tuned for the feedforward neural network. The first is η, a value to reduce the error of the new weight computed by the momentum equation. And the second is α, a penalty value assigned to the previous weight of the optional momentum component. The standard deviation from the receptor, σ values has been tuned for used in Gaussian basis function in RBF neural network.

4. Results and Analysis

The experiment was conducted for the six datasets - three classification and three regression. For the feedforward neural network the experiment has been conducted for 0, 1 and 2 hidden layers respectively. The hidden layers for RBF neural networks has been performed on the Edited Nearest Neighbor, K-Means, and K-Medoids outputs from the previous assignment.

4.1 Classification

The following tables show the 0/1 loss of the classification experiments and the time elapsed per execution of each algorithm:

Dataset	Feedforward Neural Network			
	0	1	2	3
`abalone`	0.2500	No convergence	0.1117	0.1115
`car`	0.0547	0.01375	0.0143	0.0201
`segmentation`	9.000	0.4056	0.1960	0.1915

Table 1: 0/1 Loss for the Classification Data Set Experiments

4.2 Regression

The following tables show the Mean-Square Error of the regression experiments and the time elapsed per execution of each algorithm:

Dataset	Feedforward Neural Network			
	0	1	2	3
`forestfire`	1675750656	1675893376	1672190464	302379072
`machine`	8353931776	8655540224	7046219776	8705173504
`wine-white`	0.0074	0.0053	0.0035	0.0020
`wine-red`	170.957	173.220	167.9259	168.083

Table 2: MSE Loss for the Regression Data Set Experiments

4.3 Discussion on the Results

As seen above, the feedforward neural network performed decently on the classification datasets, but poorly on the majority of regression datasets. Consistently large MSE values manifested in the regression sets, suggesting potential errors through overflow, or, more likely, an inadequate number of hidden layers and hidden nodes.

Due to complications with generalizing the classification and regression functions for the Gaussian Kernel, we were unable to record usable data from the radial basis function neural network. Our implementation in the code remains; however, concerns regarding the Gaussian kernel - especially in relation to the regression data sets - have severely limited our analysis.

Regardless, the feedforward neural network seems to perfom best at 2 hidden layers, thus providing partial support to our aforementioned hypothesis. Save abalone's massive data set that does not converge on low numbers of layers, and the inefficiency of 0 hidden layers for an MLP network, the network's results are promising.

5. Conclusion

The experiments conducted on the assigned data sets demonstrated the capabilities of two neural networks. The feedforward network used backpropogation to exhibit the ability to learn from incorrect outputs and correct for future errors. The Radial Basis Function network overcomes the issue of linear separability between datapoints. In the end, however, the performance of the Feedforward Neural Network, with backpropogation, has been demonstrated and the basic functionality of the Radial Basis Function neural network has been implemented.

References

D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, Oct 1998. doi: 10.1162/089976698300017197.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991. ISBN 0-201-50395-6.

Sanjeev Kulkarni and Gilbert Harman. An elementary introduction to statistical learning theory. 04 2011. doi: 10.1002/9781118023471.

Mark J. L. Orr. Introduction to radial basis function networks, 1996.

Christian Robert. The bayesian choice: From decision theoretic foundations to computational implementation. 01 2007.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Y. Sai, R. Jinxia, and L. Zhongxia. Learning of neural networks based on weighted mean squares error function. In *2009 Second International Symposium on Computational Intelligence and Design*, volume 1, pages 241–244, Dec 2009. doi: 10.1109/ISCID.2009.67.

M. Sarigül and M. Avci. Performance comparision of different momentum techniques on deep reinforcement learning. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 302–306, July 2017. doi: 10.1109/INISTA.2017.8001175.