Radial Basis Function and Multilayer Perceptron Neural Networks Convergence Rates and Performance

Clint Cooper CLINTGCOOPER@GMAIL.COM
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Emily Rohrbough EMILYROHRBOUGH@YAHOO.COM
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Leah Thompson LEAH.THOMPSON@MSU.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Abstract

Multilayer perceptron and radial basis function neural networks are adaptive algorithms that use input-output information to learn, or approximate, unknown functions. This project required the development of both a multilayer perceptron and a radial basis function network to be trained on sets of function values for the Rosenbrock equation for five dimensions. Once these networks are trained, inputs within the training range were tested to determine the correctness of the function approximation. Overall, the radial basis network was expected to outperform the multilayer perceptron network when provided with the same inputs and expected outputs. It was predicted that the multilayer perceptron neural network containing more hidden layers would approximate better than if it contained fewer hidden layers, and that the radial basis neural network would approximate better when more radial basis nodes were used.

Keywords: Neural networks, multilayer perceptron, feedforward neural network, radial basis function network, function approximation

1. Introduction

Artificial neural networks are adaptive, biologically inspired algorithms which can attempt to learn, or approximate, unknown functions by the passing of input information through weighted neurons (Sheppard, 2015a). This paper analyzes two separate neural network implementations: multilayer perceptron (MLP) neural network and radial basis function (RBF) neural network. Like other artificial neural networks, both of these networks are able to approximate a function, given a set of inputs, to a degree of accuracy within a specified threshold. (Kurt Hornik and White, 1989), (Park and Sandberg, 1991). MLP networks apply activation functions to approximate the unknown function, while RBF networks use a radial basis function (Sheppard, 2015a). To determine the quality of each network, five variations of the Rosenbrock equation were approximated using sets of function values from two to six dimensions. The experiments used the five Rosenbrock variations on different versions of the networks to determine the convergence rate and performance of each algorithm under various conditions.

2. Problem Statement and Hypothesis

Function approximation is used to obtain an approximation of $f(x)$ for a set of function values (X, F) given $f_1, f_2, ..., f_m$ for m nodes of $x_1, x_2, ..., x_m$ (Den Haan, 2014). Learning a neural network model to

implement the function $f : X \to R$ is the goal of function approximation (Sheppard, 2015a). For this project, the five versions of the Rosenbrock function were used because converging to a minimum for this function is known to be difficult. Thus, this made an interesting analysis of the MLP and RBF neural networks implemented (Sujanovic and Bingham, 2013).

Given the behavior of the layers in each network, we expect the RBF neural network will outperform the MLP network when provided with the same set of function values. We propose the RBF neural network will approximate functions more accurately when an increased number of radial basis nodes are used, although there is a cost associated with the increased computational complexity(Engelbrecht, 2007). It is predicted that an MLP network containing more hidden layers will approximate better than a network that contains fewer hidden layers. Additionally, an MLP network that uses only sigmoid activation functions is thought to outperform and converge quicker than an MLP network that uses only linear activation functions. Finally, it is thought that the MLP network will converge quicker, but with less accuracy when approximating function values for lower dimensions, where as the radial basis function network will likely converge slower, but at a higher convergence rate when using more radial basis nodes.

3. Algorithms Implemented

3.1 Multilayer Feedforward Neural Networks

A multilayer feedforward neural network that uses the backpropogation learning function is a biologically inspired algorithm that consists of a directed graph organized as input, hidden, and output layers (Mitchell, 1997)(Kurt Hornik and White, 1989). The number of hidden layers and nodes are a design parameter that is set at run-time. Each node is connected to all nodes in the next layer with an associated weight, which is used to evaluate the input with either a linear step or sigmoid activation function (Sheppard, 2015a). During the training session, once the final layer is reached, the squared error of the output is determined against the expected output. If the error is not within a specified threshold, the neural network will use gradient decent backpropagation to determine the change of each weight for every node, starting in the output layer.

3.1.1 Feedforward Perceptron

Multilayer feedforward neural networks use functions, referred to as activation functions, to evaluate each node's input in the hidden and output layers given input weights and a bias weight(Mitchell, 1997). The linear and sigmoid functions are the two types activation functions used in the network developed for this project. Note: the activation functions for each node are determined at run-time and do not have to be consistent throughout the network. The Heaviside step function was the linear activation function used:(Sheppard, 2015a):

$$f(x) = \begin{cases} 1, x > 0 \\ 0, x < 0 \end{cases}$$

Multilayer feedforward networks are also capable of representing nonlinear functions when using the sigmoid activation function (Mitchell, 1997). The sigmoid function produces a nonlinear output that is a multiplicity of a smoothed, differentiable threshold function, making the inputs computable (Mitchell, 1997). The two common sigmoid activation functions are the hyperbolic tangent function, which ranges from -1 to 1, and the logistic function, which ranges from 0 to 1(Sheppard, 2015a). Both functions are applied on the linear combination of inputs to scale the output between their function's perspective ranges which produces a similar shape when graphed(Sheppard, 2015a). For this project, the logistic function was used(Sheppard, 2015a):

$$f(x) = \frac{1}{1 - e^{-x}}$$

3.1.2 Backpropagation

The backpropagation learning function is also referred to as stochastic gradient descent learning (Engelbrecht, 2007). It takes the output layer values obtained during the feedforward pass through the network, determines the squared error of the output, and performs backward propagation on the hidden layer if the error exceeds a specific threshold (Sheppard, 2015a). The squared error on the output layer is the summation of the delta errors of each output node, calculated by comparing the outputs to the expected outputs(Kurt Hornik and White, 1989):

$$Err(X) = \frac{1}{2}\sum_{x}^{X}(d_x - o_x)^2.$$

If the error threshold is exceeded, backward propagation is necessary and gradient decent is applied to the delta errors to update the weights of each node in the network(Sheppard, 2015a). Through backpropagation, the calculated errors from the output layer are sent towards the input layer. The hidden layer's weights are simultaneously updated based off of the propagated error signal (Kurt Hornik and White, 1989). The weights of the output layers are updated as follows (Sheppard, 2015a):

$$w_{ji} \leftarrow w_{ji} + \eta\Delta w_{ji}$$

where

$$\Delta w_{ji} = x_{ji}\delta_j$$

and δ_j depends on the derivative of the activation function used:

$$\delta_j = \begin{cases} (d_j - o_j) & LinearActivationFunction \\ (d_j - o_j)o_j(1 - o_j) & LogisticActivationFunction \end{cases}$$

The hidden layers are updated as:

$$w_{kj} \leftarrow w_{kj} + \eta\Delta w_{kj}$$

where

$$\Delta w_{kj} = x_{kj}o_j(1 - o_j)\sum_{k \in ds(k)} w_{kj}\delta_k.$$

The method for this paper incorporated momentum into the update weight calculation. This is a tunable parameter between 0 and 1 that impacts the new weight based on the previous weight, w', to avoid local minima (Sheppard, 2015a). Thus, the equations used for updating the weights in the implemented networks for this project are as follows:

$$w_{ji} \leftarrow w_{ji} + (1 - M)\eta\Delta w_{ji} + M(w_{ji} - w'_{ji})$$

and

$$w_{kj} \leftarrow w_{kj} + (1 - M)\eta\Delta w_{kj} + M(w_{kj} - w'_{kj})$$

3.2 Radial Basis Neural Network

A radial basis function (RBF) neural network is a feedforward network that can approximate a function based on a set of function values using three, and only three, layers. (Sheppard, 2015a). The calculations of the hidden layer nodes apply radial basis functions. A given radial basis function, ϕ, depends on the input values, as well as centroids and width parameters, both which act as weights between the input and hidden layer. At the output layer, a linear summation is applied on the hidden layer's outputs and the respective weights. After the final outputs are computed, the delta error is calculated to determine if the error is within a specified threshold (Engelbrecht, 2007). The RBF network implemented, uses a two-phase method for training to improve the convergence time of the network: (1) unsupervised learning for determining the centroids, and (2) gradient descent for adjusting the weights (Engelbrecht, 2007).

3.2.1 Phase One

The first step in training an RBF network is determining the centroids, μ, using an unsupervised learning method. The centroids are found within the input vector and are used to approximate the function the network is attempting to learn. There are several unsupervised techniques that can be used to determine the centroids: static centers, randomized centers, clustering based approaches, i.e. k-means, and orthogonal least squares.

For this project, the centroids were random sub-samples of the input vector. These centroids are used to calculate a width, σ, around the centroid. This is determined as follows:

$$\sigma = \frac{d_{max}}{\sqrt{2K}}$$

where d_{max} is the maximum Euclidean Distance between all the centers and K is the total number of centers. These two values are used in the radial basis function, along with the input value, to determine the output of the hidden nodes.

Radial basis functions (RBFs), also referred to as kernel functions, are strictly positive, radially symmetric functions that utilize the centroids and widths (Engelbrecht, 2007). The output of an RBF indicates the closeness of the input vector, X, to the center μ, where σ specifies the width of the input space for a hidden unit(Sheppard, 2015a). There are several basic functions that can be implemented within an RBF network, and it has been shown when more are implemented, the function approximation improves (Engelbrecht, 2007). However, for this project, only the Gaussian radial basis function was used. The Gaussian function is as follows:

$$\phi(x, \mu, \sigma) = \exp(\frac{-||x - \mu||^2}{2\sigma^2})$$

3.3 Phase 2

The output of the final RBF network layer is calculated as a linear combination of the hidden layer outputs and the respective weights, where the weights are strictly between the hidden and output layers. The initial weights can be set randomly or with matrix inversion. In the RBF network implemented for this project, the weights were randomly sets. The output of the final layer is as follows:

$$f(x) = \sum(w_i \phi(x_i))$$

After output is calculated, the squared error is calculated and compared against the threshold. If the error is too large, gradient descent is used to update the centers. This is the final learning phase in the network and weights will continue to be adjusted until the error is within the specified threshold. Thus, the weights are updated as follows:

$$w_i \leftarrow w_i + \eta \Delta w_i$$

where

$$\Delta w_{ij} = \frac{\partial Error}{\partial w_{ij}} = (d_j - o_j)\phi(x_i)$$

4. Experimental Approach

The MLP and RBF neural networks developed for this project were trained and tested on the same inputs and outputs generated by the Rosenbrock function for five dimensions (Rosenbrock, 1960). The training and testing sets were partitioned using the Pareto Principle of 80/20 respectively (Newman, 2005). The sets of function values were used on the MLP algorithm using either a linear or sigmoid function where the number of hidden layers were 0, 1, and 2, and the RBF algorithm using a Gaussian function of 3, 5 and 7 nodes. This resulted in a total of 45 tests.

4.1 Rosenbrock Function

Rosenbrock, also referred to as the Valley or Banana function, is unimodal such that the global minimum lies in a narrow, parabolic valley making convergence in an arbitrarily restricted space difficult (Sujanovic and Bingham, 2013). The Rosenbrock equation is as follows (Sheppard, 2015b):

$$f(X) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2]$$

where x_1 through x_n are the inputs, running the equation with n dimensions.

For the purposes of this project, the input values were randomly selected over a range between zero and four, and the number of dimensions inputted into the Rosenbrock equation varied between two to six. Thus, five sets of function values were used on each network, where each set correlates with a specific dimensionality. The sets of function values were of size twenty (sixteen sets for training and four different sets for testing) to accommodate the time needed for full test coverage.

4.2 Tuning Process

The MLP and RBF neural networks were designed to handle flexible parameters, taking in the inputs, the outputs, a minimum learning rate, an error threshold, a momentum rate and the hidden layer topology. The hidden layer topology was assumed differently for each network. The MLP network allowed specification in terms of the number of hidden layers used and which activation function each hidden node used. The RBF network took in the number of hidden layer nodes, but always used the Gaussian radial basis function on each hidden node.

Additionally, to assist with the speed of convergence for the MLP network, a dynamic learning rate was implemented using the inputted learning rate, LR, at a minimum to ensure learning always occurred. The dynamic learning rate was calculated as follows:

$$\eta = max(LR, 1 - \frac{1}{MaxIterations - CurrentIteration + 1})$$

For each test, the following parameters were used: a 0.3 minimum learning rate, a 5 percent error threshold, and a 0.5 momentum rate. The thirty tests ran on the MLP network assumed the same activation functions throughout the network, either all linear or all sigmoid, for 0, 1, and 2 hidden layers. This was to easily interpret the results. The fifteen tests ran on the RBF network used 3, 5, and 7 nodes for the hidden layer using the Gaussian radial basis function. For clear measurement of results, a maximum number of iterations was set to 1,000,000 for the MLP network and 500,000 for the RBF network. The MLP network allowed for a larger number of maximum iterations to accommodate the dynamic learning rate.

5. Results

45 total tests were completed. The neural network was expected to approximate the Rosenbrock function given sixteen function sets corresponding to a specific dimension. Once the network was trained, the accuracy of the function approximation was tested with four function value sets. The resulting accuracy can be viewed in the three tables below where:

- Table 1 corresponds to the MLP network when using only linear activation functions.
- Table 2 corresponds to the MLP network when using only sigmoid activation functions.
- Table 3 corresponds to the RBF network.

Input Dimension	Number of Layers	Test Error 1	Test Error 2	Test Error 3	Test Error 4
2	0	624.8	>1000.0	>1000.0	333.1
2	1	624.8	203.7	624.8	435.5
2	2	>1000.0	846.5	595.6	51.5
3	0	252.5	153.5	330.0	>1000.0
3	1	>1000.0	758.1	575.7	>1000.0
3	2	37.2	66.3	335.3	130.4
4	0	951.7	131.0	135.4	548.9
4	1	33.7	111.7	131.7	40.9
4	2	128.6	139.1	257.9	247.0
5	0	182.1	116.0	84.4	124.9
5	1	125.5	306.5	150.8	131.4
5	2	234.0	907.1	158.5	>1000.0
6	0	148.5	420.4	119.4	105.9
6	1	>1000.0	105.4	120.1	114.8
6	2	113.7	178.0	116.1	108.6

Table 1: Error percentage for each test given an input dimension and number of layers of the MLP network using the linear activation function.

Input Dimension	Number of Layers	Test Error 1	Test Error 2	Test Error 3	Test Error 4
2	0	205.7	45.5	349.6	216.8
2	1	275.0	48.6	41.4	39.9
2	2	0.0	36.7	9.7	20.7
3	0	83.5	9.3	12.3	62.6
3	1	>1000.0	2.5	4.9	0.8
3	2	166.4	40.1	148.3	6.1
4	0	20.7	75.7	>1000.0	300.3
4	1	6.4	72.2	68.7	54.9
4	2	228.0	734.0	82.4	10.1
5	0	30.4	228.6	40.6	734.0
5	1	77.5	150.5	14.1	48.9
5	2	133.8	74.9	140.4	124.5
6	0	8.5	83.1	>1000.0	65.6
6	1	264.0	70.2	24.7	128.2
6	2	47.0	28.8	535.2	18.6

Table 2: Error percentage for each test given an input dimension and number of layers of the MLP network using the sigmoid activation function.

Input Dimension	Number of Nodes	Test Error 1	Test Error 2	Test Error 3	Test Error 4
2	3	>1000.0	58.2	10.5	58.2
2	5	>1000.0	>1000.0	>1000.0	>1000.0
2	7	679.5	>1000.0	2.9	13.9
3	3	62.0	940.0	133.9	44.8
3	5	592.5	67.5	>1000.0	287.5
3	7	32.6	19.9	115.2	852.5
4	3	31.3	27.9	226.7	6.4
4	5	527.0	57.3	53.8	63.0
4	7	1.5	115.0	45.1	289.7
5	3	22.5	3.5	52.3	10.2
5	5	36.3	46.0	117.0	18.7
5	7	3.9	19.4	13.8	9.7
6	3	11.6	>1000.0	0.3	394.5
6	5	12.8	16.5	23.2	116.2
6	7	54.0	2.5	27.3	1.2

Table 3: Error percentage for each test given an input dimension and number of nodes of the RBF network using the Gaussian radial basis function.

5.1 Performance

The performance of each network was defined as the accuracy of the function approximation. From this, we concluded:

1. The RBF network did obtain a more accurate Rosenbrock approximation than the MLP network for all of the dimensions. This was concluded by comparing the test errors in Table 1, Table 2, and Table 3 because the same test sets were used for each dimension.

2. Table 3 shows that as the number of Gaussian radial basis nodes increased, the accuracy of the function approximation also increased.

3. The MLP networks trained with the sigmoid activation function were able to approximate with more accuracy than with the MLP network using the linear step activation function.

4. For both MLP networks, Table 1 and Table 2 show that when the number of hidden layers increased from 0 to 1 and then 1 to 2, approximation accuracy did indeed increase.

5.2 Convergence

The convergence rates were dependent on the network parameters, number of hidden layers, and sparseness of the data used for training. Knowing this, we concluded:

1. Comparing the RBF network to both MLP networks, the RBF network always converged in fewer iterations and never reached the set maximum number of iterations.

2. Comparing the convergence rates of the two MLP networks, the MLP network that used the linear activation function was never able to converge within the maximum number of iterations, which is why it always resulted in poor function approximation.

6. Conclusion

The accuracy of our networks was highly dependent on the number of training inputs. The networks would converge and learn, but not well enough for a true function approximation. When we tested our networks with the test function sets, we could get over a 100 percent error. With the concern that the networks were designed incorrectly, we tested our networks with eight sets of input-output values that would train the network to learn an X^2 function. When testing this X^2 function approximation, we would get outputs with far less error (or almost no error) when given dense data points. This indicated our networks were designed correctly, but we had not provided enough training data to learn or approximate the Rosenbrock function. After this complementary test, it could be concluded that our training data set was too small and too sparse for strong, meaningful results. Thus, our results do not depict the networks' ability to accurately approximate the function.

7. Future Works

In the future, the size, range and density of the training sets used will need to be considered thoroughly to correspond to the problem we are attempting to solve. This includes creating a wider range of input data over a denser selection. Additionally, we would like to use training sets corresponding to different functions other than the Rosenbrock function. This would allow us to evaluate the MLP and RBF networks' actual ability to do function approximation.

References

Wouter J Den Haan. Function approximation. 2014.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley and Sons, Ltd, 2007.

Maxwell Stinchcombe Kurt Hornik and Halber White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

T.M. Mitchell. *Machine Learning*. McGraw-Hill Science, Engineering, and Math, 1997.

M.E.J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

John Sheppard. Lecture notes in machine learning: Soft computing, 2015a.

John Sheppard. Csci 447 machine learning: Soft computing project #2 pdf, 2015b.

Sonja Sujanovic and Derek Bingham. Virtua library of simulation experiments, 2013. URL `http://www.sfu.ca/ ssurjano/rosen.html`.