Introduction to Machine Learning, Programming Assignment 6

Feedforward Neural Networks with Backpropagation

Julie P. Garcia

Johns Hopkins University

Abstract

Artificial Neural Networks are models used in machine learning that mimic biological neural networks like the human brain. Specifically feedforward neural networks are artificial neural networks in which information only flows forward through the network. We evaluated the performance of one method for training a feedforward network termed the Backpropagation Algorithm. Backpropagation is a supervised learning technique for training data in a multilayer feedforward network. [1, 3] In this paper, we demonstrate that adding hidden layers to a feedforward neural network trained with backpropagation increases accuracy to a point, before performance begins to decline.

Hypothesis

In this paper, we demonstrate that using a model of a feedforward neural network trained with backpropagation will show better performance the more hidden layers it is trained on up until a threshold in which performance begins to decline.

Algorithms Implemented

A multilayer perceptron was implemented to construct the neural network. In a multilayer neural network, a number of inputs are fed into the input layer along with the bias value. The network is built in a feed-forward manner. The Backpropagation algorithm was used to train the neural network and adjust the weights. This method can be used for both classification and regression problems and is used to better process the effect of weights on the accuracy of the model by analyzing the output layer first, adjusting and then looking how that affects the input layer. The network was trained using stochastic gradient descent. Training was done using an online learning technique, which means inputs are fed into the network one at a time and error function is computed on each individual instance. Parameters are adjust at each iteration in order to minimize error.[1]

Backpropagation

Backpropagation was discovered by three teams of computer scientists separately, but is most noted to be discovered by Rummelhart, Hinton, and Williams in 1986. Backpropagation can be used to train a multiplayer neural network. In this algorithm, between each layer, the error calculated for a set of inputs between the expected output and the actual output is used to update the state of the network. The algorithm was implemented as follows:

1. Initialize the network with random values in the range (-0.01, 0.01).
    a. The input layer is initialized with random weights for each feature in the dataset.
    b. The output layer is initialized with random weights, one for each class value.
2. Next forward propagation on the network is performed by looping through the rows of the dataset and feeding them into the network one-by-one. For each row in the dataset:
    a. Activation is calculated by summing the weighted inputs and adding the bias.
    b. Neuron transfer is then performed to calculate the output. A sigmoid function was used for this.
3. After running forward propagation, backpropagation is calculated. This is done by calculating the derivative and then comparing the expected value with the output value.
4. The weighted values are then updated using the error calculated from step 3.

Experimental Approach

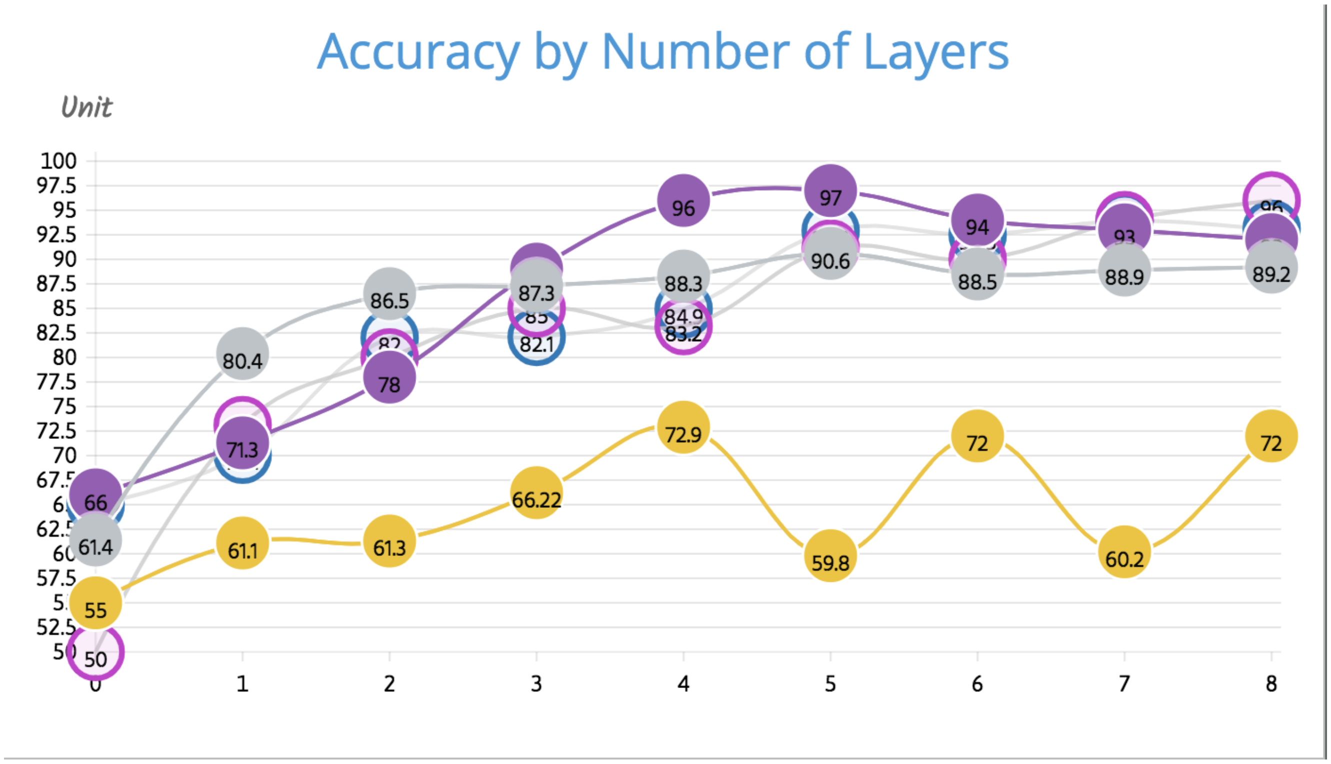
The algorithm was run on five classification datasets downloaded from the UCI Machine Learning Repository: Breast Cancer, Glass, Iris, Soybean and Votes.[2] For this a stratified 5-fold cross-validation method was used. Each dataset was divided into five randomly-selected equal datasets. The datasets are stratified to ensure equal representation of classes across each set. Five experiments were run, each time selecting a different partition (20%) as the test set and the remaining 80% as the training set. This allows for each of the 5 sets to rotate as the test set. Averages were calculated over the five experiments.

Classification error was used as a loss function to measure performance at each layer. We chose a learning factor of 0.01, a smaller value ensured that updates do not rely mainly on recent instances and have a short memory.[1] The logistic sigmoid function was used to transfer the activation. The algorithms were run with 0, 1, 2, 3, 4, 5, 6, 7, and 8 hidden layers in order to compare how the model performed as hidden layers were added.

Results

When the algorithm was run on the Breast dataset (683 examples, 9 features) the percent accuracy gradually increased from 66% to 97% as the number of hidden layers increased from 0 to 5, and then stabilized and even decreased from 6-8 hidden layers. On the Soybean dataset (47 examples, 35 features), again, the percent accuracy increased gradually as hidden layers were added up to 5 and then accuracy began to drop. For the Votes dataset (435 examples, 16 features) the same pattern was observed with gradual increase as layers were added up to 5 (95%) and then slightly declined above 5 hidden layers. The Iris dataset, with the least number of classes and examples (150 examples, 4 classes) shown in yellow in the chart below gradual increase as hidden layers were added up to 4 layers and then fluctuated after that with no real increase in performance above 4 layers. Finally, the Glass dataset (214 examples, 9 classes) showed a similar pattern with gradual increase in accuracy up to 94 and then plateaued above 5 hidden layers.

Below is a chart showing all five datasets plotted with number of hidden layers (X axis) against percent accuracy (Y axis). As you can see, in general as the number of hidden layers in the network increased, the percent accuracy increased up to about 4-5 layers. After that, adding more layers showed diminishing returns or a decrease in performance.



Summary

In summary, training a feedforward neural network showed good performance especially as the number of hidden layers approached 5 layers. However, there is a limit to the percent accuracy, and above 5 hidden layers the performance stabilized. Given that the time complexity greatly increases as layers are added, it does not gain us anything to keep adding layers to increase performance.

References

1. Alpaydin, E. (2014). *Introduction to machine learning*. Cambridge, MA: The MIT Press.
2. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
3. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning Internal Representations by Error Propagation.