A Look at Artificial Neural Networks

Aaron Cahn
University of Wisconsin-Madison
cahn@cs.wisc.edu

Abstract

In this study we implement a general artificial neural network framework. Our framework allows us to construct neural networks with arbitrary input, hidden, and output units as well as an arbitrary number of hidden layers. The user is also able to specify the learn rate, number of epochs to train the neural network, and whether to do cross validated training with n folds. This framework allows us to explore the performance and accuracy trade offs of varying the neural network parameters. For this study, we utilize the Wisconsin Breast Cancer data set [3] to motivate the necessity of parameter space exploration. We also compare the performance and accuracy of training a single neural network to that of training an ensemble of neural networks. We show <list some results/generalizations>.

1 Introduction

Artificial neural networks were originally inspired by the complex web of neurons that create the human brain. The core of a neural network is the perceptron, which is a simple representation of a neuron consisting of an activation function. The activation function is responsible for producing the output of a perceptron. Perceptrons are linked together in layers to form what is generally known as a multi-layer neural network. Historically, neural networks consist of an input layer, a single hidden layer, and an output layer. All layers are completely connected to the next layer and edge weights from layer i to layer $i+1$ are trained by an algorithm known as back propagation. Because of this, a neural network with hidden layers can represent arbitrary functions.

When a neural network contains more than a single layer, it is often referred to as a deep network. However, there is debate over whether this criteria alone constitutes a deep network. Generally, training a deep network through backpropagation alone is insufficient because of the large number of local minimum that exist over the complex error surface. Therefore, different techniques, which incrementally train one layer of weights at a time, are needed for deep networks.

Neural networks, and more specifically the edge weights from layer to layer, can easily be represented by matrices. This representation

lends itself to fast, parallel computations. As a result, neural networks are extremely popular for tasks that require a complex representation space. Many groups have used neural networks for modeling complex systems **CITE**, performing image recognition tasks [1], and performing autonomous tasks such as flight control [2].

In this paper we focus on neural networks with one and two hidden layers. Attempting to utilize more than two hidden layers trained using standard backpropagation does not improve accuracy over the data sets we use. Extending the neural network framework to include a deep training function is left to future work as we were unable to implement it before the deadline.

Our paper attempts to shed light on the trade offs a researcher can make between performance and computation cost. We motivate this by exploring the parameter space of our neural network framework and demonstrating the accuracy under different network configurations. We also explore building an ensemble of neural networks on top of our framework. We make the following contributions: <list contributions>

The rest of this paper is organized as follows. Section 2 explains the design and implementation choices of our framework. Section 3 explains the data sets we use for experimentation and the experiments we ran. Section 4 reports the results from the experiments we ran on our framework. Section 5 concludes.

2 Design and Implementation

Our artificial neural network framework was implemented in Python. It consists of a single class which takes arguments for *(i)* the number of input units *(ii)* a list H, where the ith entry of H represents the number of units to create at layer i *(iii)* the number of output units *(iv)* the learn rate to use during training and *(v)* the number of epochs to train the neural network for. If cross validation is being used the user also has control over the number of folds to create.

A neural network's edge weights from layer i to layer $i+1$ are represented by a $m-by-n$ matrix, where m is the number of units in layer i and n is the number of units in layer $i+1$. Additionally, we represent network activations and errors as each layer i are represented as single row matrices for computational convenience. As a result, all computations are fast and efficient. Note, this design choice makes the neural network implementation very comprehensible.

The neural network framework exports an API which allows the developer to *(i)* feed a single instance through the network *(ii)* run backpropagation *(iii)* train on a set of data *(iv)* test on a set of data and *(v)* print weights, errors, and activations. The functionality exported by *(v)* is for ease of debugging from the command line.

IMPLEMENTATION CHECKLIST:
1. Randomized initial values for weights
2. Allowing this to be a parameter
3. Ensemble implementation (how voting works)
4. Sigmoid function (parameterize this as well)

3 Methodology

METHODOLOGY CHECKLIST:
1. Explain the data set and data set scrubbing
2. Explain experimental setup and experiments

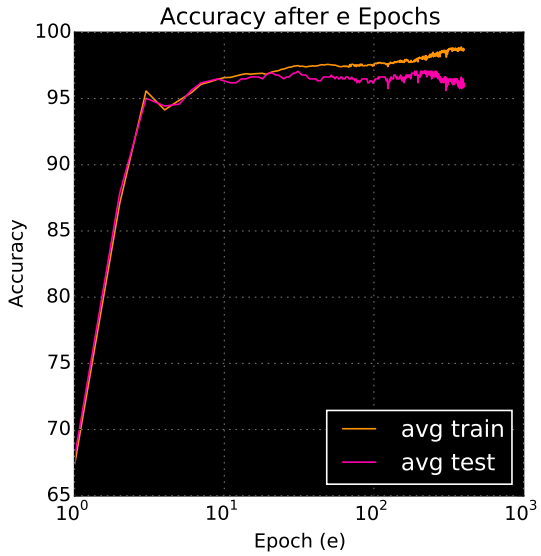

Figure 2: The figure shows the average accuracy on the training and test sets at every epoch from 1 to 400 for a neural network with 8 units at the first hidden layer and 8 units at the second. TODO: add fold talk

run
3. Explain the hypothesis and general experiment framework (how this works for any data set)

4 Results

RESULT CHECKLIST:
1. Add the ensemble plot
2. Discuss each plot in turn (interesting point from each)
3. Synthesize general result message (prediction accuracy vs. cost)

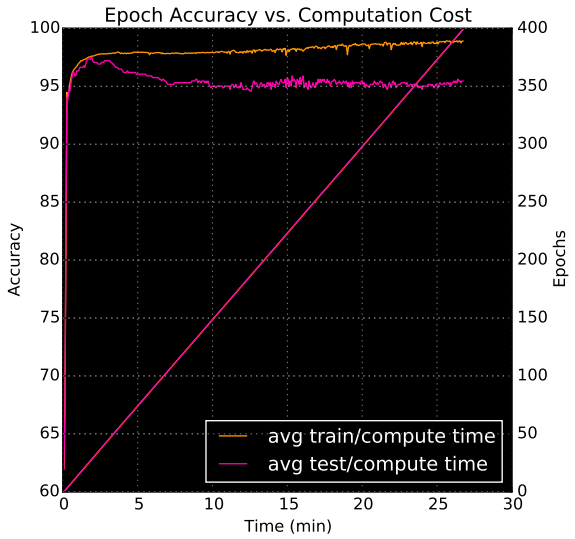

Figure 3: Add caption

5 Summary and Conclusions

SUMMARY CHECKLIST:
1. Take synthesized result and generalize into general rules about the need to search parameter spaces exhaustively
2. Talk about the consideration that need to made for choosing between certain tradoffs 3. Conclude

References

[1] M. Alvira and R. Rifkin. An empirical comparison of snow and svms for face detection. A.I. memo 2001-004, Center for Biological and Computational Learning, MIT, Cambridge, MA, 2001.

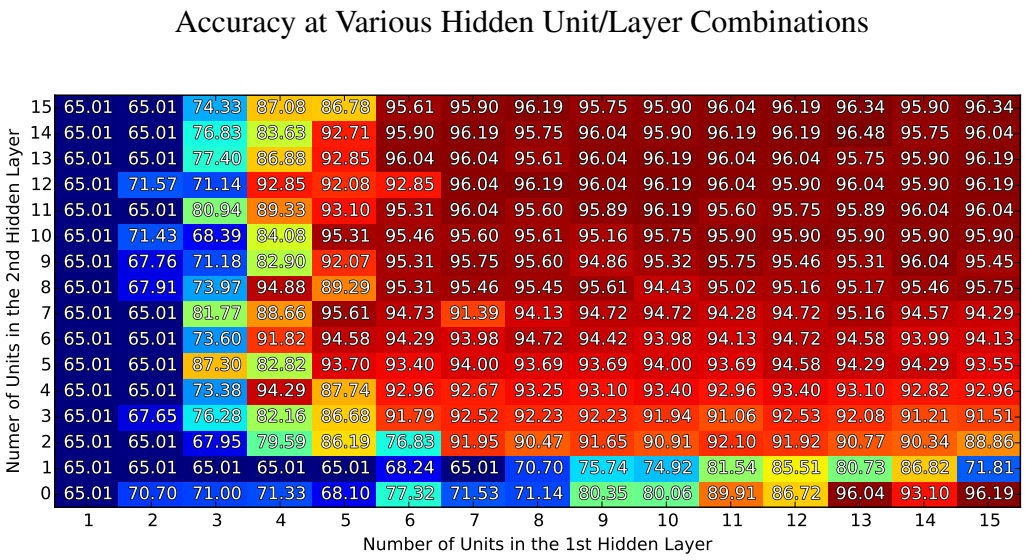

Figure 1: Accuracy for neural networks with one and two hidden layers. Entry i, j represents a neural network with j units in the first hidden layer and i units in the second hidden layer (*i.e.,* row 0 represents neural networks with only one hidden layer). The bold entry represents max accuracy.

[2] Byoung S Kim and Anthony J Calise. Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics*, 20(1):26–33, 1997.

[3] W Wolberg and O Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology,. In *Proceedings of the National Academy of Sciences*, pages 9193–9196, Dec 1990.

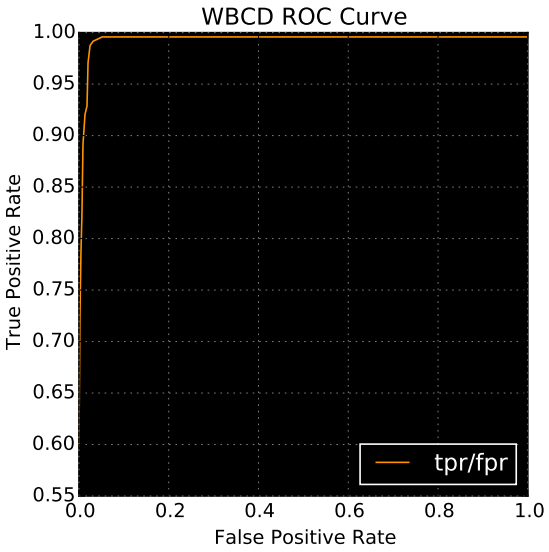

Figure 4: Add caption

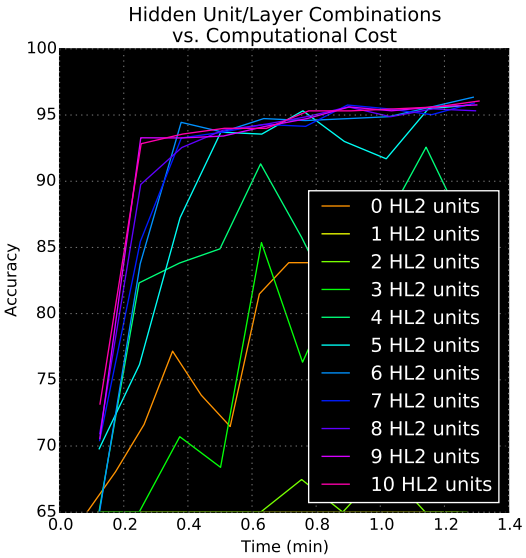

Figure 5: Add caption