A Look at Artificial Neural Networks

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

December 18, 2014

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 perceptron combinations 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 n fold cross validation on the data set provided to a neural network instance. This framework allows us to explore the performance and accuracy trade offs of varying the neural network parameter space. For this study, we utilize the Wisconsin Breast Cancer data set [6] to motivate the necessity of comprehensive parameter space exploration.

1 Introduction

Artificial neural networks were originally inspired by the complex web of neurons that create the human brain [5]. 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 backpropagation. Because of this, a neural network with hidden layers can represent arbitrary functions [2].

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 [3], performing image recognition tasks [1], and performing autonomous tasks such as flight control [4].

In this paper we focus on neural networks with up to two hidden layers. Attempting to utilize more than two hidden layers trained using standard backpropagation does not improve accuracy over the data set 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 demonstrate predictive accuracy under different network configurations. We show that there are indeed significant trade offs that can be made with respect to predictive accuracy and computational cost; even on a simple data set.

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 set we use for experimentation. and the experiments we ran. Section 4 explains the two sets of experiments we ran. Section 5 reports the results from the experiments we ran on our framework. Section 6 concludes.

2 Design and Implementation

We implemented our artificial neural network framework in Python. The framework 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 hidden 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 the user specifies the use of cross validation, the user has control over the number of folds to create from the supplied data set.

When an instance of our neural network class is invoked, all weight matrices are instantiated with random weight values in the range $[-0.1, 0.1]$. 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 $i+1$. We chose to represent the network activations and errors at each layer i as single row matrices for computational convenience. As a result, all computations are fast and efficient. For the sigmoid function we use $f(x) = \frac{1}{1+e^{-x}}$. Note, we plan to extend our framework to include weight initialization and choice of the sigmoid function as initialization parameters.

The 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 network weights, errors, and activations. The functionality exported by *(v)* is merely there for comprehensibility and to ease debugging from the command line.

3 Data Set

For the parameter space analysis we utilize the Wisconsin Breast Cancer data set from the UCI Machine Learning Repository. The WBC data set contains 699 instances and each instance contains 9 attributes that can take on values from the set $1, ..., 10$. Detailed information about each attribute can be found on the UCI web page [1] The class label an instance can take on is either *benign* or *malignant*.

To utilize the WBC data set we did a small amount of pre-processing to format the data in a manner our framework understands. First, we converted the comma separated list of attributes for each instance into a single row matrix. The class label is converted into a separate single row matrix as well. Note, although the WBC data set class label is binary, the above step allows our framework to generalize to multi-class data sets easily. Second, we converted the class label representation by assigning the value 0 to *benign* and 1 to *malignant*.

4 Methodology

The goal of our study is to investigate the performance and cost trade offs of a neural network. To this end we designed two sets of experiments to motivate and capture these trade offs. We want to stress the generality of the approach below (*i.e.,* the parameter space exploration described below could be implemented into our neural network framework). We motivate the approach by using the WBC data set, but a similar parameter space exploration can, and should, be done when training any neural network.

[1] `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)`

4.1 Parameter Space Exploration

Our first set of experiments consists of an iterative search through the parameter space. We ran a search over all combinations $(i_1, ..., i_h)$ of hidden units present in a network with h hidden layers. All i_k's range from 0 to n_k where $1 < k \leq h$ and i_1 ranges from 1 to n_1. In our experiment $h = 2$ and $n = 15$ for all h. We record the predictive accuracy for each $(i_1, ..., i_h)$ combination. Note, each combination utilized 10-fold cross validation and each fold was trained for 10 epochs. We then utilize the best $i_1, ..., i_h$ combination to train a network for e epochs recording the average predictive accuracy of the ten folds at each iteration. We set $e = 400$ in this experiment. The best $i_1, ..., i_h$ combination and e value were used to train and record predictive accuracy of a single neural network.

4.2 Computational Costs

We explore the computational costs of neural network training (*e.g.,* parameter space exploration) in our second set of experiments. To do this, we ran a similar experimental set to the set enumerated above while recording run times of various portions. For the first experiment we ran the first experiment from the set described above and recorded the computational time it took to generate the predictive accuracy for i_1 between 1 and n_1 while holding i_2 constant. Note, this is the equivalent of generating a row from Figure 1. This was done for all values of i_2 from

0 to n_2. For the second experiment, we ran the second experiment from the set described above and recorded the computational time for each epoch from 1 to e.

5 Results

Figure 1 shows the results from our search through the parameter space of hidden unit combinations. An entry (i, j) in the table is the predictive accuracy of a neural network with j perceptrons in the 1st hidden layer and i perceptrons in the 2nd hidden layer. We overlay a heat map on the table to better quantify how the choices for (i, j) combinations effects predictive accuracy. For example, it is clear from the table a neural network with one hidden layer (*e.g.*, $j = 0$) performs well (above 80% predictive accuracy) if it consists of 9 or more perceptrons. Additionally, for low values of j (less than 5), increasing the number of perceptrons in the 2nd hidden layer does not improve predictive accuracy substantially. We posit this is because the representational power of the 1st hidden layer is too low and the feature space is being compressed to heavily that no amount of additional complexity at the 2nd hidden layer can compensate.

Although we do not see this is Figure 1, ideally we want to see a concentrated area of red surrounded by blue. We speculate this did not occur because the target concept for the WBC data set was very easy to learn and therefore, adding more perceptrons at each hidden layer does not lead to significant overfitting of the data. Normally, it is advantageous to pick an (i, j) combination that falls on the horizon.

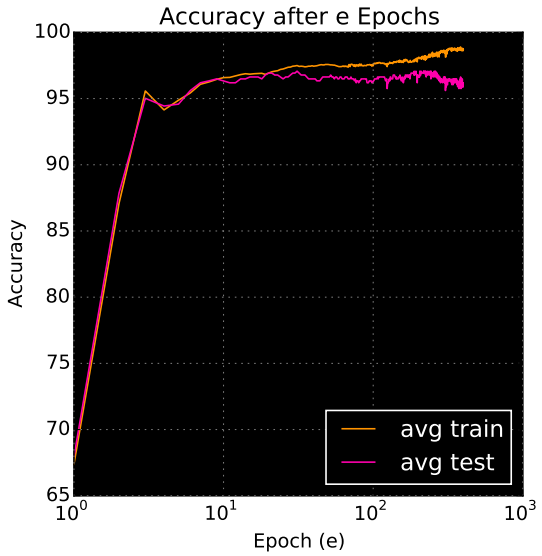

Figure 2: Average accuracy on the training and test sets at every epoch from 1 to 400 for a neural network with 14 units at the 1st hidden layer and 13 units at the 2nd hidden layer. 10-fold cross validation used during each epoch.

However, we chose to proceed with the best (i, j) combination from the table. Note, for the space we searched this is $(13, 14)$.

Figure 2 shows the results from training a neural network with 13 perceptrons at the 1st hidden layer and 14 perceptrons at the 2nd hidden layer. The x-axis is a log scale of e the number of epochs the network was trained for. There is no true point whereafter it is clear the network is overfitting the data set. However, the average predictive accuracy over the training and test sets does begin to deviate after about 60 epochs. This result is also, perhaps more easily, seen in Figure 5

After deriving the ideal (within our search

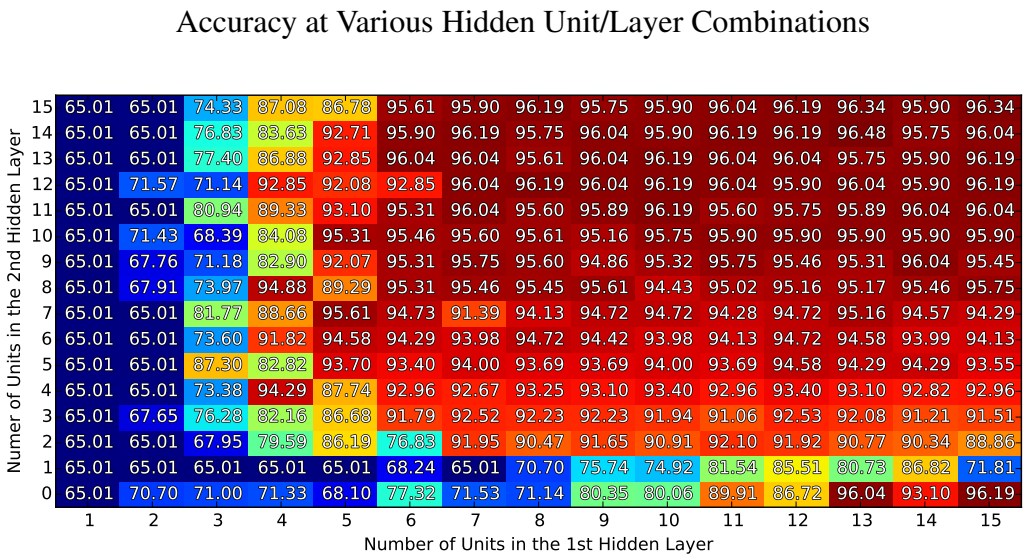

Figure 1: Accuracy for neural networks with one and two hidden layers. Entry (i, j) represents a neural network with j units in the 1st hidden layer and i units in the 2nd hidden layer (*i.e.,* row 0 represents neural networks with only one hidden layer). All combinations were trained for 10 epochs and 10-fold cross validation.

space) neural network parameters for the WBC data set, we trained a final neural network. Figure 3 shows the true positive rate vs. the false positive rate of our model over the WBC data set. Our model does an extremely good job of predicting true positives without incurring many false positives in the process. In the context of the WBC data set this is very important because predicting a true positive means catching malignant breast cancer which could lead to treatment and recovery. The very low false positive rate is important as well because it means that patients will not undergo unnecessary tests and procedures because it was predicted their breast cancer was malignant.

We have shown the trade offs that can be made with respect to choice of parameters and the predictive accuracy achievable. However, these trade offs cannot be considered in isolation. The computational cost of training networks with various parameters needs to be considered as well. This is because the computational time needed to train a network scales with the number of perceptrons and the number of hidden layers. When these parameters get sufficiently large the computational time can be on the order of days.

With the WBC data set we don't reach such long run times for training. Despite this, we are able to demonstrate the effect at a smaller scale. Figure 4 shows the computational time required to vary the number of perceptrons in the 1st hidden layer while holding the number of perceptrons in the 2nd hidden layer constant.

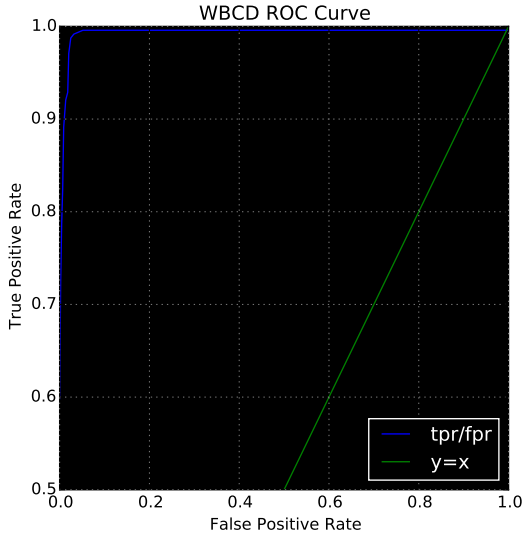

Figure 3: True positive rate as a measure of the false positive rate for a neural network with 14 units at the 1st hidden layer and 13 units at the 2nd hidden layer. Network trained for 60 epochs using 10-fold cross validation within each epoch.

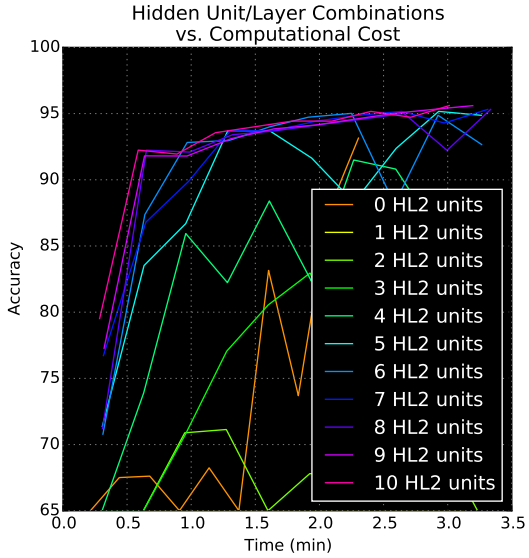

Figure 4: Each line represents the predictive accuracy as a function of computational time required to vary the number of perceptrons in the 1st hidden layer while holding the number of perceptrons in the 2nd hidden layer constant (*i.e.,* this is equal to generating a row of Figure 1

Each line is the computational effort needed to generate one row from Figure 1. Therefore, the time taken to search the entire parameter space from Figure 1 is the sum of these separate lines. We note that once about 6 perceptrons in the 2nd hidden layer are used the accuracy versus time lines stabilize. If the cost and time of computation were a significant concern then it would be practical to early stop the parameter search space. However, we do want to point out that Figure 4 demonstrates that the parameter search can itself be made highly parallel because no line from the figure depends on any other. Each line is training many separate neural networks. These computations can proceed in parallel either at the granularity of a line (from Figure 4), the granularity of individual networks, or even the granularity of matrix operations within a single network.

In similar style to Figure 4, the trade off between the computational cost of training a network for some number of epochs e versus the predictive accuracy is shown in Figure 5. It is clear that there is a significant amount of extra computational effort needed to achieve minimal increases in predictive accuracy after about 60 epochs. Namely, the model has the same predictive accuracy after about 3 minutes of training as

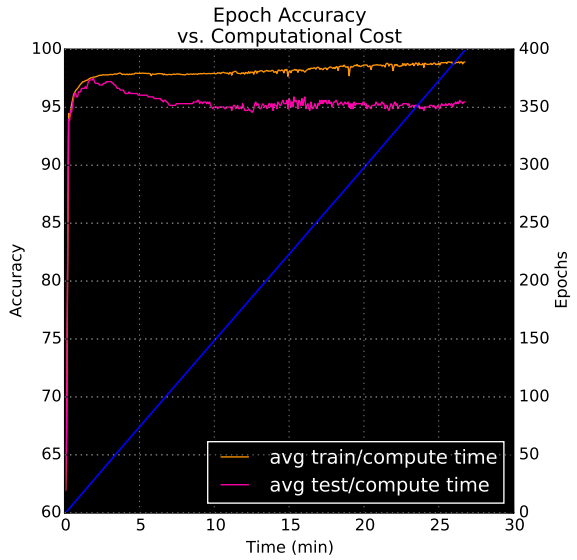

Figure 5: The predictive accuracy as a function of the computational time required to reach that level of accuracy (*i.e.,* the number of epochs trained). The blue line uses the right hand y-axis and indicates the number of epochs e completed at time t.

it does after about 30 minutes of training. Even worse, is that the model begins to over fit the data after the 3 minute mark. Although, overfitting is minimal with the WBC data set, it may be significantly more drastic on a more complicated data set with hundreds of attributes. As with the previous analysis, there is some parallelism that can be exploited during the epoch parameter space search.

6 Summary and Conclusions

In this study we have built a general neural network framework that allows us to construct networks with arbitrary combinations of input/hidden/output perceptrons, any number of hidden layers, and train for any number of epochs. This flexibility allows us to do a parameter space search to both demonstrate the necessity of doing such a search and train a model that has the overall best predictive accuracy on the data set we utilized.

We argue that it is naive for someone wishing to use a neural network for a given learning task to simply hard code these parameters. Due to the potentially large size of the parameter space it is important to exhaust it as best as possible before training a final model. However, we have also demonstrated that this alone does not suffice. As the complexity of the target concept and the instances increases so to will the neural network. This means that very quickly a trade off between the potential predictive accuracy of the model and the computational effort to search the parameter space needs to be made (*i.e.,* what parts of the parameter space should be searched to discover parameters that lead to the desired predictive accuracy). Luckily, much of this work lends itself to models of parallel processing and grid search, in which case the trade offs need not be so stark.

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.

[2] Mark Craven. Neural networks. *CS 760: Machine Learning*, 2014.

[3] A.T.C. Goh. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143 – 151, 1995.

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

[5] Tom M Mitchell. Artificial neural networks. *Machine learning*, pages 81–127, 1997.

[6] 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.