Comparison of Artificial Neural Networks

Preston Engstrom

***Abstract*—Artificial Neural Networks (ANN) have been used as a workhorse for classification and regression problems, particularly in large covariate spaces. ANN models are built around multiple parameters, which with very slight modification can yield vast changes in model performance. In this paper, we explore several different models and the methods to compare them. We demonstrate these models and methods using a single layer, feed-forward network trained on well studied datasets, adjusting parameters, activation functions and learning algorithms. We apply ANOVA, t-tests, hypothesis tests and confidence intervals to explore what changes in a models parameters are significant.**

***Index Terms*—Neural Networks, Machine Leaning, Feed Forward, Propagation**

I. Introduction

ARTIFICIAL neural networks for classification have been a subject of study for decades. ANN's have proven so useful because of their ability to use arbitrary approximation functions to learn form observed data. This means that for any data set there are a very large number of possible models to choose from, each with their own set of internal parameters to be optimized. This complexity means that an understanding of the theory behind these different models and parameters is important to the selection of a robust model.

The goal of this paper is to present and discuss comparisons of ANN models, using different learning algorithms, activation functions and network architectures. Using Fisher's Iris data set and the Breast Cancer Wisconsin data set as case studies, we will demonstrate the differences and similarities in supervised learning rules and their effect on the learning process in feed forward ANN's. In most real world applications, once rough comparisons are made between different learning algorithms, other artificial intelligence approaches can be applied to evolve a network and its parameters far more precisely than through manual crafting.

Both the Iris and Cancer data sets are classification data sets. The classification problem is as follows: Given a vector of inputs, and a known set of classes, output what class the input belongs to. Neural networks are extremely useful in classification. Their ability to generalize on unseen data is demonstrable by their wide use in industry on problems ranging from text and speech recognition to image classification.

II. Feed Forward Neural Networks

A neural network is a graph structure, a collection of nodes, joined by weighted edges. Our neural network is implemented as a multi-layer, feed forward network. In general, a feed forward network is one that only connects nodes in one layer to the next layer. It is a directed graph with no cycles. The simplest feed forward network is the single layer perception. Data is passed directly from the input nodes to the output nodes, with no hidden layers. This simple network architecture is impossible to train on non-linearly separable data.

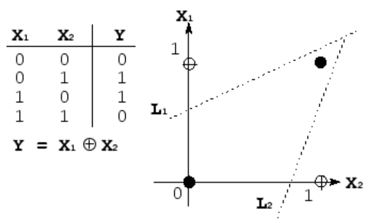

Fig. 1. A visualization of the linear inseparability of the XOR function

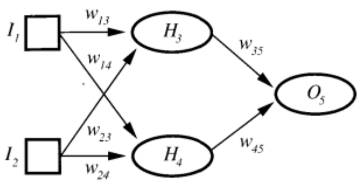

Fig. 2. A very simple, two-layer, feed-forward network with two inputs, two hidden nodes, and one output node[1]

The introduction of hidden layers in feed forward networks alleviates this restriction. The Universal Approximation Theorem states that a feed forward neural network with a single hidden layer can approximate continuous functions on bounded subsets of $\mathbb{R}$.

A. Learning Rules

There are several algorithms for teaching neural nets to adapt their weight based on some calculated error. Training begins with a model with randomly initialized weights, which are modified until some criteria is met. At that point, if the model is acceptable, it is taken, or the training process can reset or continue. The majority of learning rules utilize gradient descent, traveling backwards through the network to compute the required gradients.

Fig. 3. Simple Backprop Skeleton

```
given training example and label
for each training example:

  for o in outputs:
    calculate output_delta

  for h in hidden:
    calculate hidden delta

  update hidden weights
  update input weights
```

Fig. 4. Rprop Skeleton Code

```
  for each epoch
    given training example and label
    for each training example:

    for o in outputs:
    calculate output_delta

    for h in hidden:
    calculate hidden delta

    accumulate gradients
  update weights
```

B. Backpropagation

C. Resilient Propagation

D. Over-fitting in Neural Networks

III. Methods of Analysis

A. McNemar's Test

B. ANOVA

IV. Results and Discussion

A. Comparison of Activation Functions

B. Comparison of Backpropagation Models with Learning Rate and Momentum

C. Effects of the Number of Hidden Nodes

V. Conclusions

The conclusion goes here.

References

[1] S. Russell and P. Norvig, Artificial intelligence. Upper Saddle River, N.J.: Prentice Hall/Pearson Education, 2003.
[2] K. Hornik, ”Approximation capabilities of multilayer feedforward networks”, Neural Netw., vol. 4, no. 2, pp. 251-257, 1991
[3] J Demsar, ”Statistical Comparisons of Classifiers over Multiple Data Sets”, Journal of Machine Learning Research vol .7, 2006