Nick Handelman

CMPS 588 Spring 2012

February 16, 2012

**Back Propagation in an Artificial Neural Network for Learning Propositional Logic**

**Interest and Background**

I propose a Java implementation of an artificial neural network (ANN) using the back propagation algorithm for learning propositional logic. My interest in this topic stems from my interest in artificial intelligence and machine learning, both of which can be augmented through the use of the biologically-inspired ANN. Propositional logic provides a fairly simple and well understood platform on which ANNs can be further studied. Additionally, propositional logic provides a basis on which more complex models of logic, such as first-order logic, can be extended. As such, developing ANN's to operate on propositional logic is a basic step in modeling ANN's to operate in a complex real-life environment.

**Proposal**

My chosen topic is based on "Java Project: Part 1", found on Dr. Anthony Maida's website, and calls for students to "write an artificial neural network program which can be trained to recognize simple two-input boolean patterns such as *AND*, *OR*, AND *XOR*." More specifically, the program calls for the use of a back propagating ANN to achieve this goal and includes pseudo code to help guide students through the project. In fact, a simple internet search reveals that the problem posed in the guidelines has, predictably, been implemented many times in various programming languages and is widely and freely available. Since I do find the topic interesting but not particularly complex, I plan to not only implement the ANN as laid out in "Java Project: Part 1", but to also extend on the idea by developing an ANN to handle an arbitrarily complex propositional logic.

**Approach**

For example, consider the proposition "p *AND* q", an elementary example. A feed-forward ANN can easily be developed to determine the output, given the two inputs p and q. Developing a back-propagating ANN for this proposition is certainly more complex, but examples of this are readily available. In fact, a single neuron with two inputs is sufficient to tackle the problem. Other propositional operators, such as *OR*, can be handled by similar methods.

Now consider the proposition p *XOR* q. An equivalent proposition, that is, one whose output is the same for a given input, is "(p *AND* (*NOT* q)) *OR* (q *AND* (*NOT* p))". This equivalent proposition is of seemingly greater complexity than *XOR*, but it allows *XOR* to be expressed using the basic propositional building blocks of *AND*, *OR*, and *NOT*.

To the point, I plan to implement a program that will read in some arbitrary proposition along with relevant training data, convert that proposition into an ANN and use the back-propagation algorithm to learn on the training data. The two previous examples of *AND* and *XOR* were to demonstrate the extensibility of propositional logic through the use of ANNs. Since the basic propositional building blocks can be implemented in an ANN, I propose that any arbitrary combination of these basic propositions can be implemented through an ANN.