Chapter 14: The Future of Neural Networks

- Where are Neural Networks Today?
- To Emulate the Human Brain or the Human Mind
- Understanding Quantum Computing

Neural networks have existed since the 1950s. They have come a long way since the early perceptrons that were easily defeated by problems as simple as the XOR operator; however, they still have a long way to go. This chapter will examine the state of neural networks today, and consider directions they may go in the future.

Neural Networks Today

Many people think the purpose of neural networks is to attempt to emulate the human mind or pass the Turing Test. The turing test is covered later in this chapter. While neural networks are used to accomplish a wide variety of tasks, most fill far less glamorous roles than those filled by systems in science fiction.

Speech and handwriting recognition are two common uses for today's neural networks. Neural networks tend to work well for these tasks, because programs such as these can be trained to an individual user. Chapter 12 presented an example of a neural network used in handwriting recognition.

Data mining is another task for which neural networks are often employed. Data mining is a process in which large volumes of data are "mined" to establish trends and other statistics that might otherwise be overlooked. Often, a programmer involved in data mining is not certain of the final outcome being sought. Neural networks are employed for this task because of their trainability.

Perhaps the most common form of neural network used by modern applications is the feedforward backpropagation neural network. This network processes input by feeding it forward from one layer to the next. Backpropagation refers to the way in which neurons are trained in this sort of neural network. Chapter 5 introduced the feedforward backpropagation neural network.

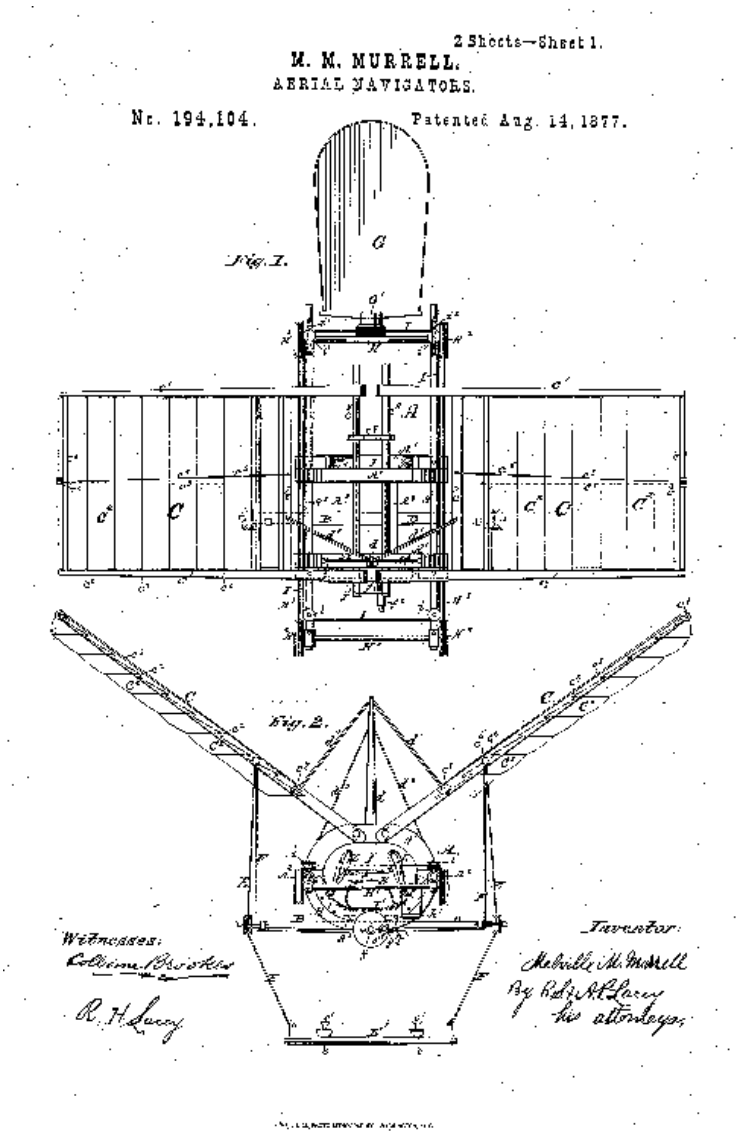
A Fixed Wing Neural Network

The ultimate goal of AI is to produce a thinking machine and we are still a long way from achieving this goal. Some researchers suggest that perhaps the concept of the neural network itself is flawed. Perhaps other methods of modeling human intelligence must be explored. Does this mean that such a machine will have to be constructed exactly like a human brain? To solve the AI puzzle, must we imitate nature? Imitating nature has not always led mankind to the optimal solution—consider the airplane.

Man has been fascinated with the idea of flight since the beginnings of civilization. Many inventors in history worked towards the development of the "Flying Machine." To create a flying machine, most inventors looked to nature. In nature, they found the only working model of a flying machine, the bird. Most inventors who aspired to create a flying machine, developed various forms of ornithopters.

Ornithopters are flying machines that work by flapping their wings. This is how a bird flies, so it seemed logical that this would be the way to create a flying machine. However, no ornithopters were ever successful. Many designs were attempted, but they simply could not generate sufficient lift to overcome their weight. Figure 14.1 shows one such design that was patented in the late 1800s.

Figure 14.1: An ornithopter.



It was not until Wilbur and Orville Wright decided to use a fixed wing design that airplane technology began to truly advance. For years, the paradigm of modeling a bird was pursued. Once the two brothers broke with tradition, this area of science began to move forward. Perhaps AI is no different. Perhaps it will take a new paradigm, different from the neural network, to usher in the next era of AI.

Quantum Computing

One of the most promising areas of future computer research is quantum computing. Quantum computing has the potential to change every aspect of how computers are designed. To understand quantum computers, we must first examine how they are different from computer systems that are in use today.

Von Neumann and Turing Machines

Practically every computer in use today is built upon the Von Neumann principle. A Von Neumann computer works by following simple discrete instructions, which are chip-level machine language codes. This type of machine is implemented using finite state units of data known as "bits," and logic gates that perform operations on the bits. In addition, the output produced is completely predictable and serial. This classic model of computation is essentially the same as Babbage's analytical engine developed in 1834. The computers of today have not strayed from this classic architecture; they have simply become faster and have gained more "bits." The Church-Turing thesis sums up this idea.

The Church-Turing thesis is not a mathematical theorem in the sense that it can be proven. It simply seems correct and applicable. Alonzo Church and Alan Turing created this idea independently. According to the Church-Turing thesis, all mechanisms for computing algorithms are inherently the same. Any method used can be expressed as a computer program. This seems to be a valid thesis. Consider the case in which you are asked to add two numbers. You likely follow a simple algorithm that can easily be implemented as a computer program. If you are asked to multiply two numbers, you follow another approach that could be implemented as a computer program. The basis of the Church-Turing thesis is that there seems to be no algorithmic problem that a computer cannot solve, so long as a solution exists.

The embodiment of the Church-Turing thesis is the Turing machine. The Turing machine is an abstract computing device that illustrates the Church-Turing thesis. The Turing machine is the ancestor from which all existing computers have descended. The Turing computer consists of a read/write head and a long piece of tape. The head can read and write symbols to and from the tape. At each step, the Turing machine must decide its next action by following a very simple algorithm consisting of conditional statements, read/write commands, and tape shifts. The tape can be of any length necessary to solve a particular problem, but the tape cannot be infinite in length. If a problem has a solution, that problem can be solved using a Turing machine and some finite length of tape.

Quantum Computing

Practically every neural network developed thus far has been implemented using a Von Neumann computer. But it is possible that the successor to the Von Neumann computer, quantum computing, may take neural networks to a near human level. A quantum computer is constructed very differently than a Von Neumann computer.

Quantum computers use small particles to represent data. For example, a pebble is a quantum computer for calculating the constant-position function. A quantum computer would also use small particles to represent the neurons of a neural network. Before we consider how a quantum neural network is constructed, we must first consider how a quantum computer is constructed.

The most basic level of a Von Neumann computer is the bit. Similarly, the most basic level of the quantum computer is the "qubit." A qubit, or quantum bit, differs from a normal bit in one very important way. Where a normal bit can only have a value of 0 or 1, a qubit can have a value of 0, 1, or both simultaneously. To demonstrate how this is possible, first you will be shown how a qubit is constructed.

A qubit is constructed with an atom of some element. Hydrogen makes a good example. The hydrogen atom consists of a nucleus and one orbiting electron. For the purposes of quantum computing, only the orbiting electron is important. This electron can exist in different energy levels, or orbits about the nucleus. The different energy levels are used to represent the binary 0 and 1. When the atom is in its lowest orbit, the ground state, the qubit might have a value of 0. The next highest orbit could be represented by the value 1. The electron can be moved to different orbits by subjecting it to a pulse of polarized laser light. This has the effect of adding photons to the system. To flip a qubit from 0 to 1, enough light is added to move the electron up one orbit. To flip from 1 to 0, we do the same thing, since overloading the electron will cause the it to return to its ground state. Logically, this is equivalent to a NOT gate. Using similar ideas, other gates can be constructed, such as AND and OR.

Thus far, there is no qualitative difference between qubits and regular bits. Both are capable of storing the values 0 and 1. What is different is the concept of a superposition. If only half of the light necessary to move an electron is added, the electron will occupy both orbits simultaneously. A superposition allows two possibilities to be computed at once. Further, if you have one "qubyte," that is, 8 qubits, then 256 numbers can be represented simultaneously.

Calculations with superpositions can have certain advantages. For example, to perform a calculation with the superpositional property, a number of qubits are raised to their superpositions. The calculation is then performed on these qubits. When the algorithm is complete, the superposition is collapsed. This results in the true answer being revealed. You can think of the algorithm as being run on all possible combinations of the definite qubit states (i.e., 0 and 1) in parallel. This is called quantum parallelism.

Quantum computers clearly process information differently than their Von Neumann counterparts. But does quantum computing offer anything not already provided by classical computers? The answer is yes. Quantum computing provides tremendous speed advantages over the Von Neumann architecture.

To see this difference in speed, consider a problem that takes an extremely long time to compute on a classical computer, such as factoring a 250-digit number. It is estimated that this would take approximately 800,000 years to compute with 1400 present-day Von Neumann computers working in parallel. Even as Von Neumann computers improve in speed, and methods of large-scale parallelism improve, the problem is still exponentially expensive to compute. This same problem presented to a quantum computer would not take nearly as long. A quantum computer is able to factor a 250-digit number in just a few million steps. The key is that by using the parallel properties of superposition, all possibilities can be computed simultaneously.

It is not yet clear whether or not the Church-Turing thesis is true for all quantum computers. The quantum computer previously mentioned processes algorithms much the way Von Neumann computers do, using bits and logic gates; however, we could use other types of quantum computer models that are more powerful. One such model may be a quantum neural network, or QNN. A QNN could be constructed using qubits. This would be analogous to constructing an ordinary neural network on a Von Neumann computer. The result would only offer increased speed, not computability advantages over Von Neumann-based neural networks. To construct a QNN that is not restrained by Church-Turing, a radically different approach to qubits and logic gates must be sought. As of yet, there does not seem to be any clear way of doing this.

Quantum Neural Networks

How might a QNN be constructed? Currently, there are several research institutes around the world working on QNNs. Two examples are Georgia Tech and Oxford University. Most research groups are reluctant to publish details of their work. This is likely because building a QNN is potentially much easier than building an actual quantum computer; thus, a quantum race may be underway.

A QNN would likely gain exponentially over classic neural networks through superposition of values entering and exiting a neuron. Another advantage might be a reduction in the number of neuron layers required. This is because neurons can be used to calculate many possibilities, using superposition. The model would therefore require fewer neurons to learn. This would result in networks with fewer neurons and greater efficiency.

Reusable Neural Network Frameworks

This book introduced many classes that implement many aspects of neural networks. These classes can be reused in other applications; however, they have been designed to serve primarily as educational tools. They do not attempt to hide the nature of neural networks, as the purpose of this book is to teach the reader about neural networks, not simply how to use a neural network framework. As a result, this book did not use any third-party neural frameworks.

Unfortunately, there is not much available in terms of free open-source software (FOSS) for neural network processing in Java. One project that once held a great deal of potential was the Java Object Oriented Neural Engine, which can be found at **http://www.jooneworld.com**. However, the project appears to be dying. As of the writing of this book, the latest version was a release candidate for version 2.0. There have been no updates beyond RC1 for over a year. Additionally, RC1 seems to have several serious bugs and architectural issues. Support requests go unanswered.

The first edition of this book used JOONE for many of the examples. However, JOONE was removed from the second edition, because it no longer seems to be a supported project.

Another option for a third-party neural framework is Encog. Encog is a free open-source project I have organized. Encog is based on classes used to create this book. It is essentially the classes provided by the books Introduction to Neural Networks with Java and HTTP Programming Recipes for Java Bots. Encog provides a neural network-based framework that has the ability to access web data for processing. If the project does not call for web access, the HTTP classes do not need to be used.

Encog is a new project. The classes presented in this book will evolve beyond the learning examples provided. Specifically, Encog will be extended in the following areas:

- Advanced hybrid training
- Consolidation of classes for all types of neural architectures
- Improved support for multicore and grid computing
- More efficient, but somewhat less readable code
- Integration of the pruning and training processes

Adding all of these elements into the instructional classes provided in this book would have only obscured the key neural network topics this book presents. This book provided an introduction to how neural networks are constructed and trained. If you require more advanced training and capabilities, I encourage you to take a look at the Encog project. More information on the Encog project can be found at the Heaton Research website:

http://www.heatonresearch.com/

As you will see, Encog is available in both Java and C#. Because Encog is an open-source project, we are always looking for additional contributions. If you would like to contribute to a growing open-source neural network project, I encourage you to take a look. Sub-projects are available for all levels of neural network experience.

Chapter Summary

Computers can process information considerably faster than human beings. Yet, a computer is incapable of performing many of the tasks that a human can easily perform. For processes that cannot easily be broken into a finite number of steps, a neural network can be an ideal solution.

The term neural network usually refers to an artificial neural network. An artificial neural network attempts to simulate the real neural networks contained in the brains of all animals. Neural networks were introduced in the 1950s and have experienced numerous setbacks; they have yet to deliver on the promise of simulating human thought.

The future of artificial intelligence programming may reside with quantum computers or perhaps a framework other than neural networks. Quantum computing promises to speed computing to levels that are unimaginable with today's computer platforms.

Early attempts at flying machines model birds. This was done because birds were the only working models of flight. It was not until Wilbur and Orville Wright broke from the model of nature and created the first fixed wing aircraft that success in flight was finally achieved. Perhaps modeling AI programs after nature is analogous to modeling airplanes after birds; a model that is much better than the neural network may exist, but only time will tell.

There are not many open-source neural network frameworks available. One early and very promising project was the Java Object Oriented Neural Engine. Unfortunately, the project seems to be unsupported and somewhat unstable at this time. A project called Encog is underway based on the classes developed in this book. Encog has added advanced features that were beyond the scope of this introductory book. If you would like to evaluate the Encog framework, more information can be found at the Heaton Research website at **http://www.heatonresearch.com**.

Neural networks are a fascinating topic. This book has provided an introduction. To continue learning about neural networks I encourage you to look at the Encog project and look for additional books and articles from Heaton Research.

I hope you have found this book enjoyable and informative. I would love to hear your comments, suggestions, and criticisms of this book. You can contact me at **support@heatonresearch.com**. There is also a forum at **http://www.heatonresearch.com,** where all of our books and software can be discussed.

Vocabulary

Ornithopter

Perceptron

Quantum Computer

Quantum Neural Network

Qubit

Turing Machines

Von Neumann Machine