**What Is A Neural Network?**

The simplest definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen. He defines a neural network as:

*"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*

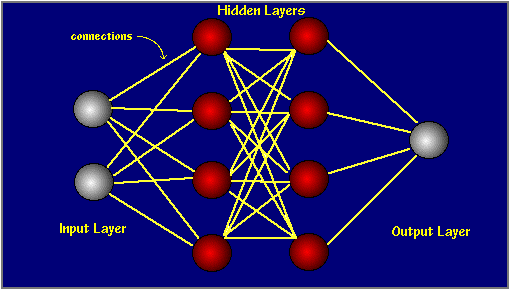
In "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989

ANNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mamalian cerebral cortex but on much smaller scales. A large ANN might have hundreds or thousands of processor units, whereas a mamalian brain has billions of neurons with a corresponding increase in magnitude of their overall interaction and emergent behavior. Although ANN researchers are generally not concerned with whether their networks accurately resemble biological systems, some have. For example, researchers have accurately simulated the function of the retina and modeled the eye rather well.

Although the mathematics involved with neural networking is not a trivial matter, a user can rather easily gain at least an operational understanding of their structure and function

**The Basics of Neural Networks**

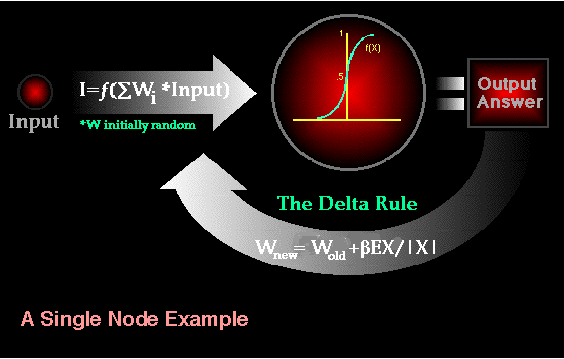
Neural neworks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output .

. 

Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs.

Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The delta rule is often utilized by the most common class of ANNs called 'backpropagational neural networks' (BPNNs). Backpropagation is an abbreviation for the backwards propagation of error.

With the delta rule, as with other types of backpropagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights. More graphically, the process looks something like this:



### Artificial Neural Networks: Introduction and Application

Computer scientists have long been inspired by the human brain. In 1943, Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, developed the first conceptual model of an artificial neural network. In their paper, "A logical calculus of the ideas imminent in nervous activity,” they describe the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs, and generates an output.

Their work, and the work of many scientists and researchers that followed, was not meant to accurately describe how the biological brain works. Rather, an artificial neural network (which we will now simply refer to as a “neural network”) was designed as a computational model based on the brain to solve certain kinds of problems.

It’s probably pretty obvious to you that there are problems that are incredibly simple for a computer to solve, but difficult for you. Take the square root of 964,324, for example. A quick line of code produces the value 982, a number Processing computed in less than a millisecond. There are, on the other hand, problems that are incredibly simple for you or me to solve, but not so easy for a computer. Show any toddler a picture of a kitten or puppy and they’ll be able to tell you very quickly which one is which. Say hello and shake my hand one morning and you should be able to pick me out of a crowd of people the next day. But need a machine to perform one of these tasks? Scientists have already spent entire careers researching and implementing complex solutions.

The most common application of neural networks in computing today is to perform one of these “easy-for-a-human, difficult-for-a-machine” tasks, often referred to as pattern recognition. Applications range from optical character recognition (turning printed or handwritten scans into digital text) to facial recognition. We don’t have the time or need to use some of these more elaborate artificial intelligence algorithms here, but if you are interested in researching neural networks, I’d recommend the books Artificial Intelligence: A Modern Approach by Stuart J. Russell and Peter Norvig and AI for Game Developers by David M. Bourg and Glenn Seemann.

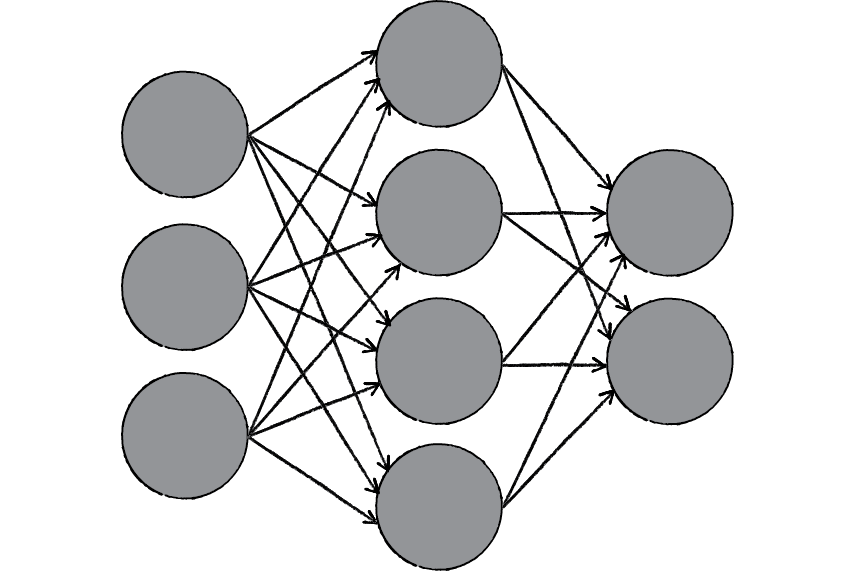


Figure 10.2

A neural network is a “connectionist” computational system. The computational systems we write are procedural; a program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel throughout a network of nodes (the nodes, in this case, being neurons).

Here we have yet another example of a complex system, much like the ones we examined in Chapters 6, 7, and 8. The individual elements of the network, the neurons, are simple. They read an input, process it, and generate an output. A network of many neurons, however, can exhibit incredibly rich and intelligent behaviors.

One of the key elements of a neural network is its ability to learn. A neural network is not just a complex system, but a complex **adaptive** system, meaning it can change its internal structure based on the information flowing through it. Typically, this is achieved through the adjusting of weights. In the diagram above, each line represents a connection between two neurons and indicates the pathway for the flow of information. Each connection has a**weight**, a number that controls the signal between the two neurons. If the network generates a “good” output (which we’ll define later), there is no need to adjust the weights. However, if the network generates a “poor” output—an error, so to speak—then the system adapts, altering the weights in order to improve subsequent results.

There are several strategies for learning, and we’ll examine two of them in this chapter.

* **Supervised Learning** —Essentially, a strategy that involves a teacher that is smarter than the network itself. For example, let’s take the facial recognition example. The teacher shows the network a bunch of faces, and the teacher already knows the name associated with each face. The network makes its guesses, then the teacher provides the network with the answers. The network can then compare its answers to the known “correct” ones and make adjustments according to its errors. Our first neural network in the next section will follow this model.
* **Unsupervised Learning** —Required when there isn’t an example data set with known answers. Imagine searching for a hidden pattern in a data set. An application of this is clustering, i.e. dividing a set of elements into groups according to some unknown pattern. We won’t be looking at any examples of unsupervised learning in this chapter, as this strategy is less relevant for our examples.
* **Reinforcement Learning** —A strategy built on observation. Think of a little mouse running through a maze. If it turns left, it gets a piece of cheese; if it turns right, it receives a little shock. (Don’t worry, this is just a pretend mouse.) Presumably, the mouse will learn over time to turn left. Its neural network makes a decision with an outcome (turn left or right) and observes its environment (yum or ouch). If the observation is negative, the network can adjust its weights in order to make a different decision the next time. Reinforcement learning is common in robotics. At time t, the robot performs a task and observes the results. Did it crash into a wall or fall off a table? Or is it unharmed? We’ll look at reinforcement learning in the context of our simulated steering vehicles.

This ability of a neural network to learn, to make adjustments to its structure over time, is what makes it so useful in the field of artificial intelligence. Here are some standard uses of neural networks in software today.

* **Pattern Recognition** —We’ve mentioned this several times already and it’s probably the most common application. Examples are facial recognition, optical character recognition, etc.
* **Time Series Prediction** —Neural networks can be used to make predictions. Will the stock rise or fall tomorrow? Will it rain or be sunny?
* **Signal Processing** —Cochlear implants and hearing aids need to filter out unnecessary noise and amplify the important sounds. Neural networks can be trained to process an audio signal and filter it appropriately.
* **Control** —You may have read about recent research advances in self-driving cars. Neural networks are often used to manage steering decisions of physical vehicles (or simulated ones).
* **Soft Sensors** —A soft sensor refers to the process of analyzing a collection of many measurements. A thermometer can tell you the temperature of the air, but what if you also knew the humidity, barometric pressure, dewpoint, air quality, air density, etc.? Neural networks can be employed to process the input data from many individual sensors and evaluate them as a whole.
* **Anomaly Detection** —Because neural networks are so good at recognizing patterns, they can also be trained to generate an output when something occurs that doesn’t fit the pattern. Think of a neural network monitoring your daily routine over a long period of time. After learning the patterns of your behavior, it could alert you when something is amiss.

**What Applications Should Neural Networks Be Used For?**

Neural networks are universal approximators, and they work best if the system you are using them to model has a high tolerance to error. One would therefore not be advised to use a neural network to balance one's cheque book! However they work very well for:

* capturing associations or discovering regularities within a set of patterns;
* where the volume, number of variables or diversity of the data is very great;
* the relationships between variables are vaguely understood; or,
* the relationships are difficult to describe adequately with conventional approaches

**What Are Their Limitations?**

There are many advantages and limitations to neural network analysis and to discuss this subject properly we would have to look at each individual type of network, which isn't necessary for this general discussion. In reference to backpropagational networks however, there are some specific issues potential users should be aware of.

* Backpropagational neural networks (and many other types of networks) are in a sense the ultimate 'black boxes'. Apart from defining the general archetecture of a network and perhaps initially seeding it with a random numbers, the user has no other role than to feed it input and watch it train and await the output. In fact, it has been said that with backpropagation, "you almost don't know what you're doing". Some software freely available software packages (NevProp, bp, Mactivation) do allow the user to sample the networks 'progress' at regular time intervals, but the learning itself progresses on its own. The final product of this activity is a trained network that provides no equations or coefficients defining a relationship (as in regression) beyond it's own internal mathematics. The network 'IS' the final equation of the relationship.
* Backpropagational networks also tend to be slower to train than other types of networks and sometimes require thousands of epochs. If run on a truly parallel computer system this issue is not really a problem, but if the BPNN is being simulated on a standard serial machine (i.e. a single SPARC, Mac or PC) training can take some time. This is because the machines CPU must compute the function of each node and connection separately, which can be problematic in very large networks with a large amount of data. However, the speed of most current machines is such that this is typically not much of an issue.

**What Are Their Advantages Over Conventional Techniques?**

Depending on the nature of the application and the strength of the internal data patterns you can generally expect a network to train quite well. This applies to problems where the relationships may be quite dynamic or non-linear. ANNs provide an analytical alternative to conventional techniques which are often limited by strict assumptions of normality, linearity, variable independence etc. Because an ANN can capture many kinds of relationships it allows the user to quickly and relatively easily model phenomena which otherwise may have been very difficult or imposible to explain otherwise.