Emergence and Applications of Artificial Neural Networks

By: Greg Thompson

For: PHYS 569 (ESM) Term Paper

Dec. 11, 2007

Abstract: Although conventional computers are capable of performing many tasks at rates far exceeding human capabilities their abilities in solving certain classes of complex problems are still dwarfed by the massively parallel human brain. This motivated the advent of artificial neural networks in which a large number of equivalent components communicate using weighted connections attempt to mimic these properties of the brain. Computational properties such as stable memories, ability to generalize, and error robustness emerge from the simple network. The spontaneous appearance and various applications of these properties are discussed in this paper.

Introduction:

Computers are prevalent in society; and for good reason. Many tasks that would be excessively tedious or practically impossible for the human brain to deal with are quite tidily handled by these machines. Processing a large quantity of data, retrieving a record from a large database, or factoring a large number are just a few examples of these sorts of problems. But consider another set of problems, something like recognizing a face in a crowd[1], and you immediately recognize that conventional computer architecture pales in comparison to the massively parallel human brain. Since the brain is itself just a massive network of relatively simple neurons, it may be possible to apply these principles to computers and create an Artificial Neural Network (ANN). This paper discusses the emergent computational properties of these ANNs focussing stable memory, ability to generalize and robustness against errors. It also discusses how these properties emerge from simple networks of elementary components.

Biological Brain:

The fundamental unit of the human nervous system is the neuron. Figure 1 (a) is a schematic of a neuron that is intended to give the reader a sense of how the signals are processed in a biological system. The essential components, in terms of signal processing, of the neuron are the dendrites, axon, and synapses. The dendrites receive electrical impulses from neighbouring neurons, this is the signal that is to be processed. It is passed through the cell body to the axon. The axon carries the signal away and to dendrites of neighbouring neurons through the synapses. The synapse is a microscopic gap that is illustrated in further detail in Figure 1 (b). The signal that transfers through the axon is transferred through the synapse to the accepting dendrite by a neurotransmitter.

The strength of the signal transmitted depends on several factors. Firstly, the strength of the incoming signal is proportional to the amount of neurotransmitter emitted into the synapse, and thus after diffusing across the gap, determines the strength of the signal received at the dendrite. Secondly, the accepting dendrite has a threshold to determine whether a signal is accepted or not, this is often modeled as a binary threshold with no signal transmitted below a certain minimum bias and full signal transmitted above this bias. Thirdly, since each neuron has a multiplicity of dendrites, it can be sending and receiving many signals simultaneously. These signals will either encourage or discourage the neuron to turn on and propagate the signal[2].

Analogy to Artificial Neuron:

The ANN is comprised of nodes with weighted connections. In the simplest model, the Perceptron, all the nodes feed forward in layers[2]. Most modern networks

feature connections back through layers, so it is possible for every node to be able to send and receive signals from every other node. Figure 2 shows a schematic of the perceptron to illustrate how these connections work. Once this diagram is understood, it is relatively easy to generalize and visualize how more complicated networks are constructed. We will understand this diagram in analogy to the biological neuron described above.

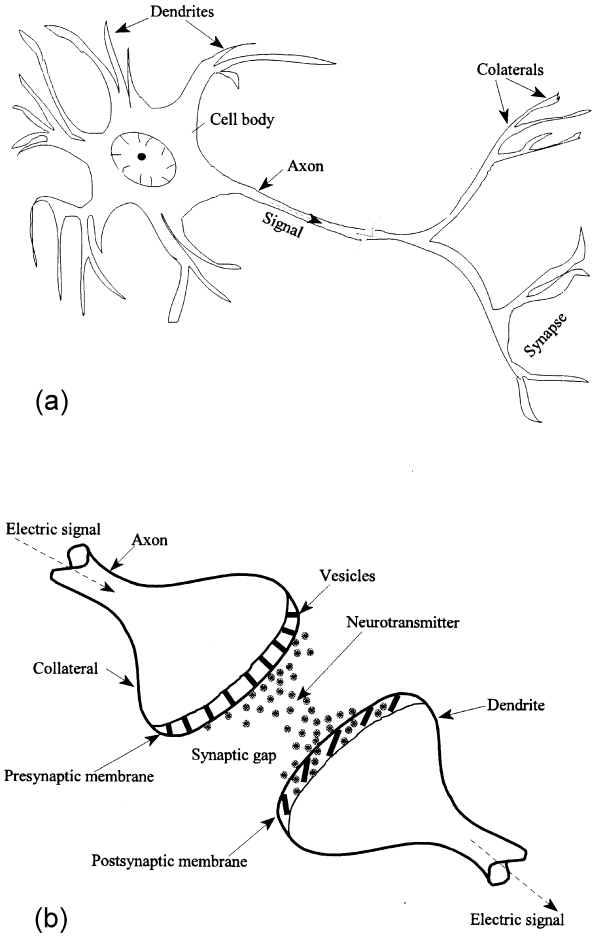

Figure 1. a) Schematic of the biological neuron.
b) Schematic of the synapse, illustrating the method of transmitting signals between neurons. (From [2])

Each node represents a single neuron itself, with the connections between nodes acting in analogy to the axons and dendrites, the weights applied to the connections acting like the synaptic gap and a threshold gate in the node acting like the threshold on the accepting dendrite. The capabilities of both the biological neuron and the artificial neuron hinge on the ability to modify the weights of the connections [3].

It is important to note that each node is a very simple element. It performs the elementary task of taking several inputs and producing an output based on whether or not a certain threshold is met after integrating over the inputs. But when connected in a massively parallel network, ranging from hundreds to tens of thousands of nodes, and trained using specific rules, certain computational properties emerge out of the network. These properties include but are not limited to stable memories including robustness against error[2], and the ability to generalize[3].

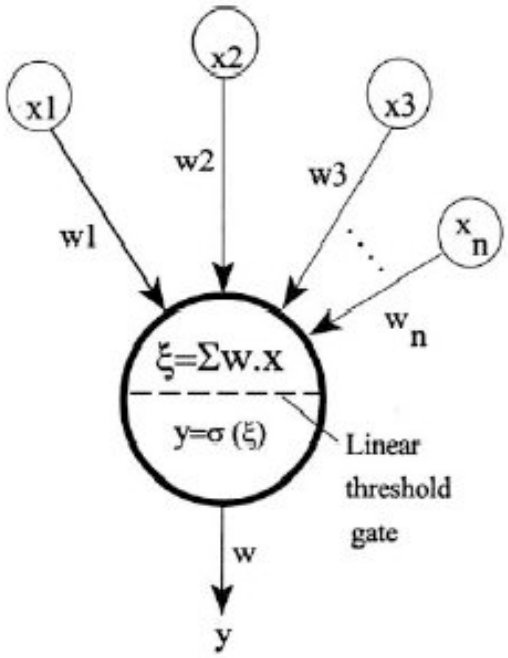

Figure 2. A schematic of the “Perceptron”, a common node in artificial neural networks. (From [2])

Stable Memories with Robustness Against Error:

A computational experiment to demonstrate how stable memories persist in biologicial neural networks was performed by J.J. Hopfield in 1982 [3]. This experiment also has the added benefit of demonstrating the emergence of stable memories in ANNs. His model consisted of a network of N = 30 and N = 100 nodes. Each neuron has a value assigned to it, $V^i = 0 \text{ or } 1$. At a fixed mean rate, but randomly in time, the neuron then adjusts its value according to:

$$\begin{matrix} V_i \rightarrow 1 \\ V_i \rightarrow 0 \end{matrix} \quad \text{if} \quad \sum_{j \neq i} T_{ij} V_j \quad \begin{matrix} > U_i \\ < U_i \end{matrix} \quad . \qquad (1)$$

Where T_{ij} is the connection weight between nodes i and j. The storage of a set of state vectors V^s, where $s = 1..n$ is accomplished by setting the weights according to the following algorithm:

$$T_{ij} = \sum_s (2V_i^s - 1)(2V_j^s - 1) \qquad (2)$$

This defines the weights of the connections between the neurons. Assigning a random initial state, the network will then evolve according to the dynamics described above. There are three possible outcomes based on the input. Firstly, the system could settle into a single stable state. Secondly, the system could cycle between a couple of the stable states. Lastly, the system could wander chaotically in a small sample of state space.

To test the memory capabilities of the system, a random set of state vectors V^s were assigned and the connection weights were computed according to the above equation. Then each nominal state vector was assigned as an initial condition of the ANN and the network was allowed to evolve until it was stationary. For n=5 initial memories in a N=100 node network, almost every memory establishes a stable state that is perfectly recallable. As the number of memories to be stored increases, the number of memories that establishes a stable state decreases and the error rate increases. This is shown in the following figure:

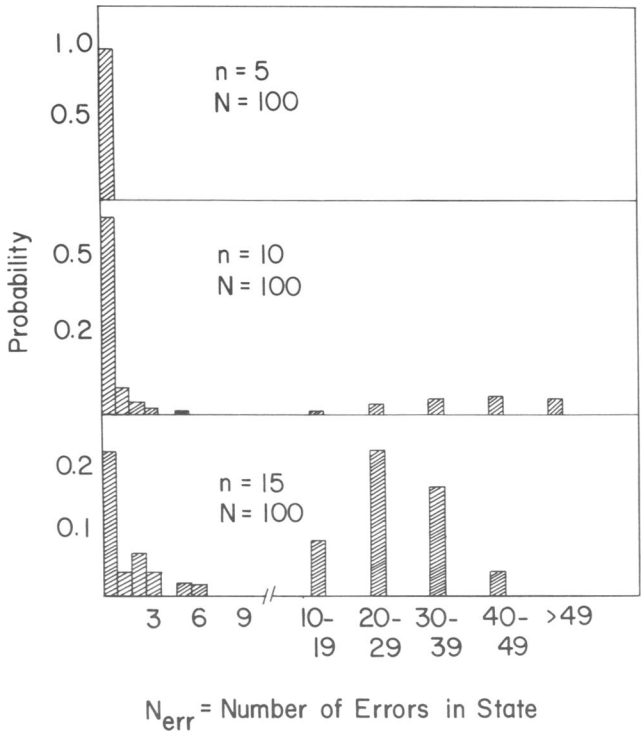

Figure 3. The probability distribution of errors between the stable states and the nominal memories.(From [3])

Once these memories are stored as stable states in the system, it is now possible to retrieve them using partial subsets of the memory, even containing errors. Define a state vector X might have fewer elements than the system has nodes, and thus fewer elements than the vectors stored as memories in the network. Recalling that the stable memories are defined as V^s, then some $X = V^a + \delta$ defines a partial knowledge of the entry we wish to retrieve. Then δ represents some deviation from the desired record, it could be a lack of information, an error in the entry, or both. If δ is sufficiently small, the system will evolve to the stable state V^a that we wished to retrieve.

The ability to store memory in this way emerged as a collective property of the many simple elements with very little network structure, and the ability to store more memories will increase as the system size increases. Hopfield estimates that the number of states that can be stored before serious error in recall is 0.15N for sufficiently large N. Beyond the emergence of the ability to store information, some other collective properties show some presence in this experiment.

Generalization:

Another collective property of the ANNs is generalization. Instead of training a network by giving it a state vector to store, it is possible to create an input-output map [4]. These map ANNs are usually described in layers. In this case we have an input layer which takes a multidimensional state vector then hidden layers which contain the weighted connections and an output layer from which an output state vector can be read. So the network can be described as a map M from the input vector X to the output vector V.

The ANN must be trained to produce a map M that is as close as possible to the desired map M_D. This is again accomplished by modifying the weighted connections between nodes. However, in this case, since we don't necessarily know what information we wish to store in the ANN (this is why the middle layer is described as hidden) we need another way to determine the couplings. How this is accomplished is by providing an input vector for which we know what the output vector should be. Given the input X, the map M provides and output V, and the dissimilarity between V and the desired output V_D gives us a dissimilarity or *generalization error* between M and M_D. The couplings are then modified to minimize this *generalization error* [4]. This process is repeated, with the number of example input vectors playing the role of time in the dynamical evolution of the map.

One example of ANNs being used for generalization is in high energy physics. In the simplest data analyses events are selected based on some hard criteria cuts, that is if a particle is reconstructed to have a certain set of kinematic parameters even if all the parameters but one are well within their desired ranges, the one that could fall just slightly outside the predetermined cut will cause the event be thrown away, reducing the statistical power of the sample. If we can train an ANN to recognize likely good candidate events, and reject likely background events we can, in some cases, significantly increase the effective statistical[5] size of the sample.

The network is trained using Monte Carlo simulations of signal and background events. The output can be as simple as a value between 0 and 1 which indicates the probability that the event is a signal event. The plot below (Figure 4) shows an example of output from a trained neural network being used to select events. Figure 4 (b) is the output from Monte Carlo signal events while Figure 4 (a) is from data. You can see from the data that the sample is mostly background events, making this a good example of a sample that would benefit from this method of signal extraction.

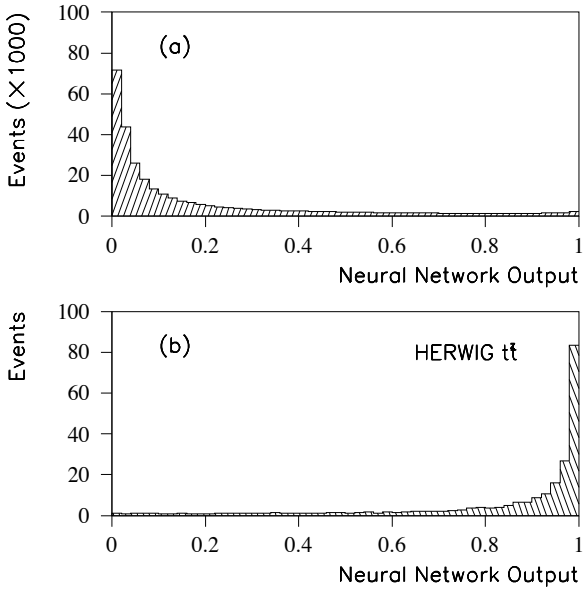

Figure 4. Neural Network Output for a) Data, b) Monte Carlo for a high energy experiment. (From [5])

Other Classes Of Problems:

There are many other classes of problems at which ANNs perform exceptionally well. Pattern classification is performed using unknown data on a network that has been trained to recognize properties that characterize a class[1],[2]. Clustering assigns input patterns with similar inter-correlations to the same classes[1],[2]. Function approximation[1],[2],[7] is also well suited to ANNs, in fact multilayer ANNs are described as universal approximators[2],[6] with the ability to approximate any function to arbitrary degree of accuracy. They are also robust forecasters given a time series[1], [2]. Some ANNs are also more efficient than other mathematical tools at optimizing nonlinear systems[1],[2],[7].

Collective Behaviour of the System:

All of these impressive properties of the network are a direct result of the massive parallelism[2]. The nodes of the ANN are simple elements that perform elementary functions. The connections between them provide very little structure[1], and are not necessarily important to the behaviour of the system. The computational properties are only mildly sensitive to the model used to construct the network[3].

To illustrate how these properties arise we look again to our first example. Consider the system described by Hopfield [3] with connections between neurons described by equation 2. Further make the constraint that $T_{ij} = T_{ji}$, that is that the matrix defining the connections is symmetric. Then define:

$$E = -\frac{1}{2}\sum_{i \neq j}\sum T_{ij} V_i V_j \qquad (3)$$

Hopfield observes that for a change in the state vector δV_i, we observe the following change in E:

$$\Delta E = -\Delta V_i \sum_{j \neq i'} T_{ij} V_j \ . \qquad (4)$$

Further, we can see that the algorithm for modifying V makes E a monotonically decreasing function[3]. So we see that the system can be taken in analogy to an Ising spin model system. In fact, the two problems are isomorphic[3]. If we describe E as the "energy" of the system, Hopfield networks operate by minimizing the energy. So we see that this simple example of a ANN is analogous to a familiar physical model that also exhibits spontaneous emergence.

Also recall that this entire system is, at its core, modeled upon the biological neuron and their networks found inside the nervous systems of intelligent beings. We can see that certain traits of thought seem to begin to manifest themselves [2], including the stable memories, ability to generalize and robustness against error. Taking the fact that these properties are a direct result of the massive parallelism and then considering that most modern ANNs are systems with 10^5 nodes, making from 5 to 100 connections per node, where the human nervous system contains upwards of 10^{11} neurons and between 100 and 10000 connections per neuron [2], perhaps this implies that even more sophisticated computational abilities emerge as the size of the network increases by several orders of magnitude.

Summary

Although I only briefly described them, it is important to note that there are many computational properties that ANNs exhibit. The emergence of stable memories including robustness against error and the ability to generalize were focussed upon in this paper because they are the most relatable to the human conscious, and easiest to demonstrate that these networks go well beyond the computational ability of their simple

elements. But what is really important is that these networks demonstrate an ability to learn. In a conventional computer constructed of simple logic gates may seem equally impressive, but the difference there is that the logic is coded by the person who designed the hardware or software. In an ANN, the logic is “learned” by the network and an input-output map is constructed that is out of the hands of the designer[2].

This paper barely scratched the surface of the incredibly rich field of research into Artificial Neural Networks. There are many different models to base the nodes and their connections on, many of which weren’t even mentioned in the scope of this paper. And while some are better suited to specific classes of problems than others, this is largely irrelevant to the spontaneous emergence of the computational abilities we observe. The underlying dynamics are not near as important as the large number of nodes in these networks and this is why we describe this system as emergent.

References:

1. K.M. Mohiuddin, J. Mao and A. K. Jain, "Artificial Neural Networks: A Tutorial", Computer, March 1996.

2. I.A. Basheer, M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", Journal of Microbiological Methods 43 (2000) 3-31.

3. J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci. USA, Vol. 79 pp. 2554-2558, April 1982.

4. D. Saad and S. A. Solla, "Exact Solution for On-Line Learning in Multilayer Neural Networks", Phys. Rev. Lett., Vol. 74 pp. 4337--4340, May 1995.

5. B. Abbott et. al., "Measurement of the top quark pair production cross section in pp collisions using multijet final states", arXiv:hep-ex/9808034v2 2 Sep 1998.

6. J. M. Benitez, J. L. Castro, and I. Requena, "Are Artificial Neural Networks Black Boxes?", IEEE Transactions On Neural Networks, Vol. 8, No. 5, September 1997.

7. J. Mao and A. K. Jain, "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection", IEEE Transactions On Neural Networks. Vol 6. No. 2. March 1995.