Neural Network Design

This paper looks at various neural network designs, with some emphasis on back-error propagation algorithm and analyzes to what degree each model is capable of emulating intelligence.

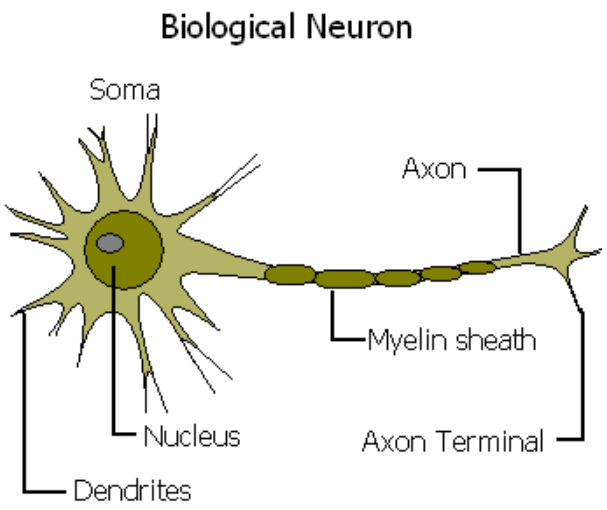
University of Arkansas

Kenneth Cason, James Barnes, Ryan Meier

2008 December 12

1. Biological Neural Network

Before trying to design an artificial neural network it is important to understand some basic principles about how biological neurons function. It should be noted that this is a very brief overview of biological neural networks. The neuron is a core component of the brain functioning primarily to transmit data from one cell to another. There are many variations of neurons each ranging in function and structure depending on what their purpose is. Neurons generate and electric signal, also known as an action potential, of which propagates through the cell, along the axon to other connecting neuron cells. The propagation of the signals is made possible by the flow of ions through the cell membrane. Some of the important ions include sodium (Na^+), potassium (K^+), chloride (Cl^-), and calcium (Ca^{2+}). As well as the electric potential across the cell membrane, there are other factors also affecting the activity of a neuron including; various types of neurons, and various types of neurotransmitters each with a different function and purpose. Some neurons excite their target neurons using such transmitters as acetylcholine, while others inhibit their target neurons. Some inhibitory neurotransmitters include GABA and glycine. There are also groups of neurons called modulatory neurons that are invoked with more complexity. Such neurotransmitters are dopamine, acetylcholine, serotonin and others. [1] (Wikipdedia)



For example, dopamine has the function of controlling the flow of information from other areas of the brain to the frontal lobe. Dopamine is also commonly known for its association with feelings and emotions. [2](Wikipedia) The above image gives a simple representation of a typical neuron cell. Labeled are a few of its functional components.

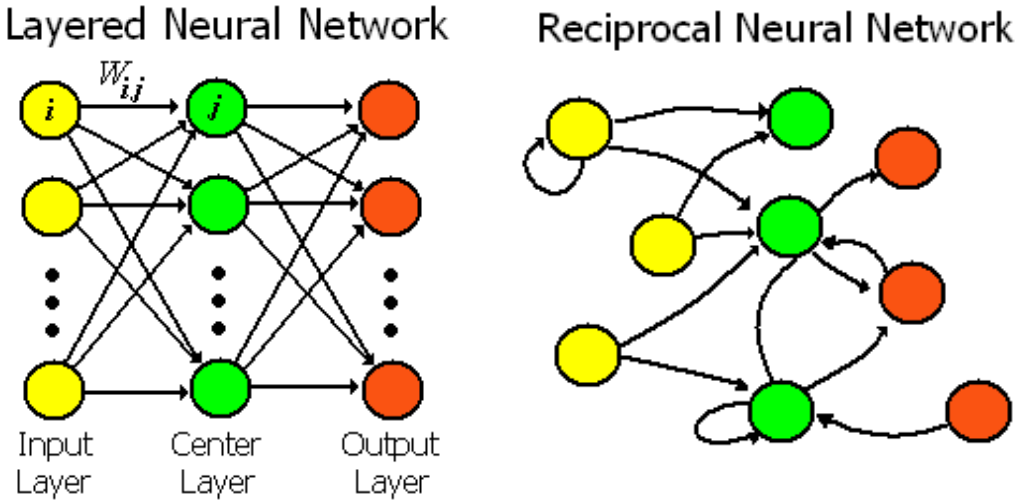
Just to give an idea of approximately how many neurons and synapses are present in biological organisms refer to the table below. Even if a perfect model of a human brain is developed, one can see that simply due to the extremely large number of neurons and synapses that perform meaningful computations would be near impossible.

Organism	Neurons	Synapse
Worm	302	5600
Ant	10,000	-
Cockroach	1,000,000	-
Mouse	10^7 10 million	$8x10^{10}$ 80 million
Octopus	30 million~100 million	-
Human (Adult）	50–100 billion	10^{14} - $5x10^{14}$
Human (Child)	50–100 billion	10^{16}
Elephant	>200 billion	-
Blue Whale	>200 billion	-

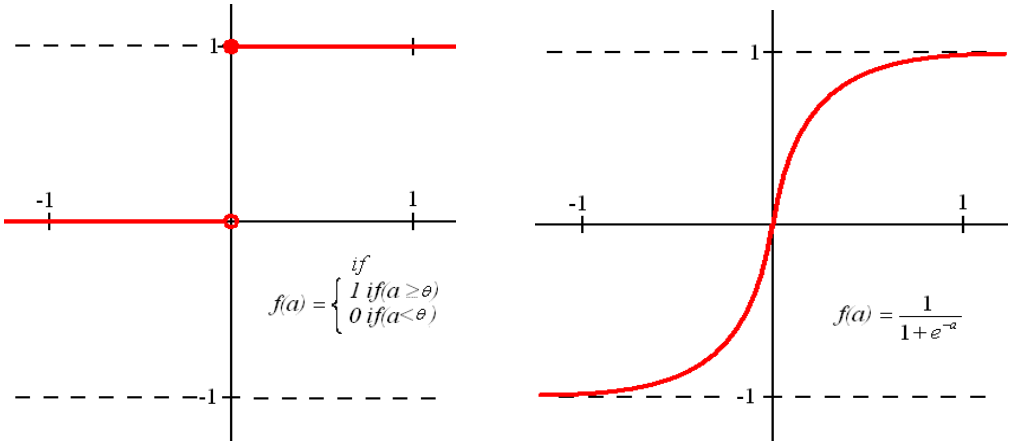
It is also interesting to observe that the number of synapses present in a human child is many times larger, approximately 20-30 times, than that of an adult. This happens as organisms learn, unused and unimportant synapses die.

2. Current Artificial Neural Networks

An artificial neural network is a computational model based on a biological neural network, commonly referred to as a neural network or NN. It is an interconnected webbing of artificial neurons used to model complex relationships between inputs and outputs or to find patterns in data. There are many successful models already in existence. Before analyzing current learning algorithms, let's first look at ways to structure a neural network. The Below models represent the two most common ways to represent neuron connections in a neural network; A layered neural network, meaning that neurons in one layer are connected only to neurons in the next layer. I.e. neurons in $layer_i$ are connected to $layer_{i+1}$. In the Reciprocal model, neurons do not have any particular rule governing which neuron that they are connected to. This means that it is possible for a neuron in one layer to make connections to a neuron in the same layer, a previous layer, the next layer, or even to itself. While these models are harder to apply current learning algorithm models to, they offer much more flexibility and computational ability of the neural networks.

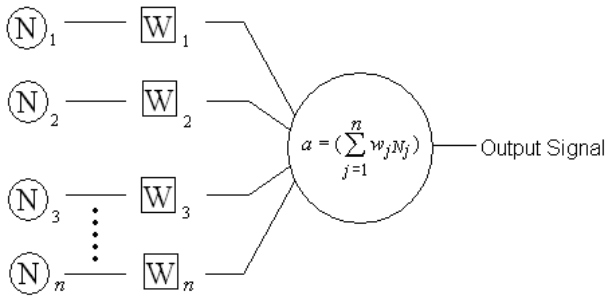


After the designing the layout of the neural network, you must next define how each of the nodes, or neurons, will function in the system. Developing an artificial neuron entails deciding how the neurons manage incoming signals, fire, and respond to other stimuli, all directly depending on the model. A common attribute of artificial neurons includes an activation function. There are two main types, they can be seen in the below two graphs:



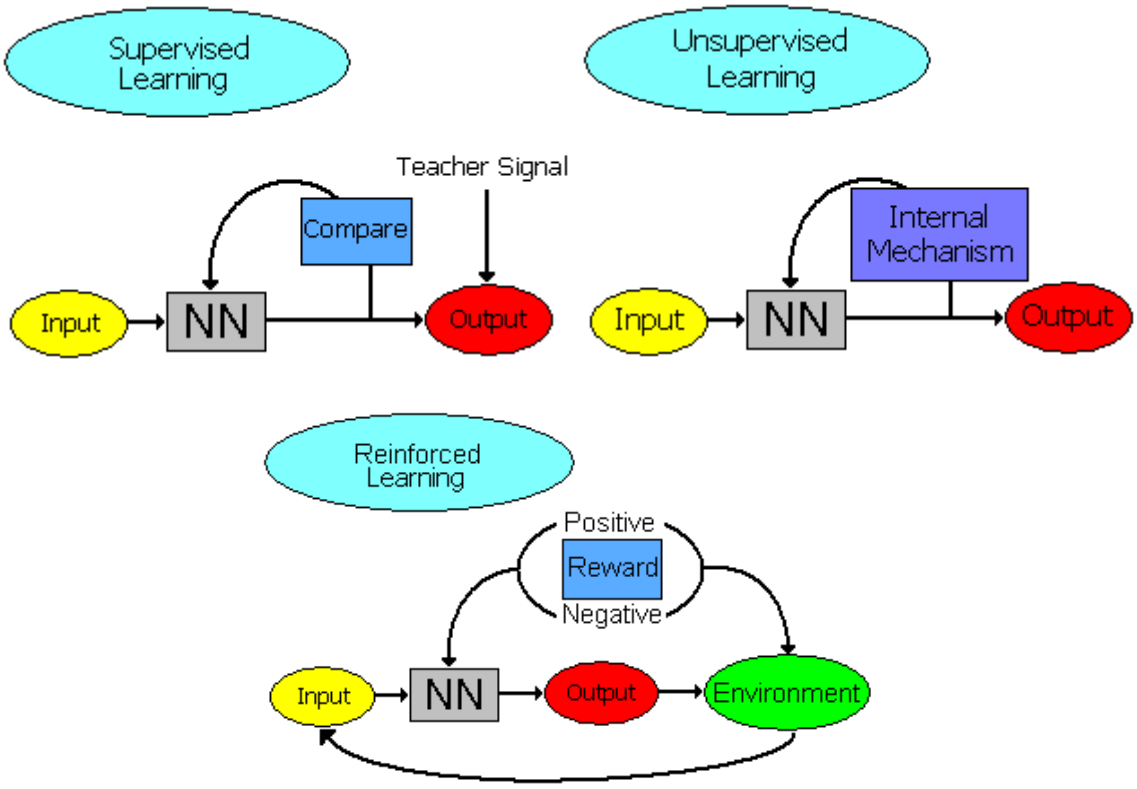
The first graph represents an “all-or-none” type of function meaning that if the incoming signals do not sum up to a specific point the neuron will not fire, i.e. produce an output signal. A neuron containing an activation function as represented in the right graph, will always produce an output regardless of the sum of the input signals.

The below graph displays a simple activation function, where N_i = neuron value $neuron_i$, and W_i = $weight_i$, and a = the output value of the neuron.



3. Learning Algorithms – Back-Error Propagation

There are a few main categories of which neural network learning algorithms are broken into. Some primary methods include: supervised learning, unsupervised learning, reinforced learning, as well as many other models such as genetic algorithms. The below diagrams show a visual representation of three commonly used models. Of the three, supervised learning and reinforced learning being the more frequently used.



Algorithms such as supervised learning are commonly used in solving problems like teaching a neural network a specific function. For example, if you input 1+1, the correct output is 2. Thus, in supervised learning the teacher signal would be 2 and would repeat until the learning algorithm output the desired value of 2.

In reinforced learning, the neural network has a problem to solve, for example, learning to walk, or playing tic-tac-toe. The environmental state would first be input into

the neural network, as well as being processed to determine whether to punish or reward the neural network based on the state of the environment. Then, based on the new input and how the neural network was rewarded or punished it would produce another output, thus again changing the state of the environment. This process is continuously repeated until a desired stage is reached or is manually stopped.

To better understand how some of these learning algorithms work, lets analyze how back-error propagation functions, first from a mathematical stand-point, then use diagrams to walk through a simple case in order to better visualize the process.

Variable Definition: Input Layer i; Center Layer j; connection weight between j and i is represented by $w^h_{j,i}$; Output Layer value y; and Teacher Signal t.

Back-error propagation algorithm: [3](Wikipedia)

1. Input the training data into the neural network.

2. Calculate the error between the actual output and the desired output.

$$\delta = \frac{1}{2}(t - y)^2$$

3. For each neuron, calculate what the output should have been, and a scaling factor to correct the value to match the desired output. This is commonly referred to as the local error.

$$\Delta w^o_{k,j} = -\varepsilon \frac{\partial \delta}{\partial w^o_{k,j}}$$

4. Adjust each the weights of each neuron to lower the local error.

$$w^o_{k,j} \leftarrow w^o_{k,j} + \Delta w_{k,j}$$

5. Assign "blame" for the local error to neurons in the previous layer, giving greater responsibility to neurons connected by stronger weights.

$$\Delta w^h_{j,i} = -\varepsilon \frac{\partial \delta}{\partial w^h_{j,i}}$$

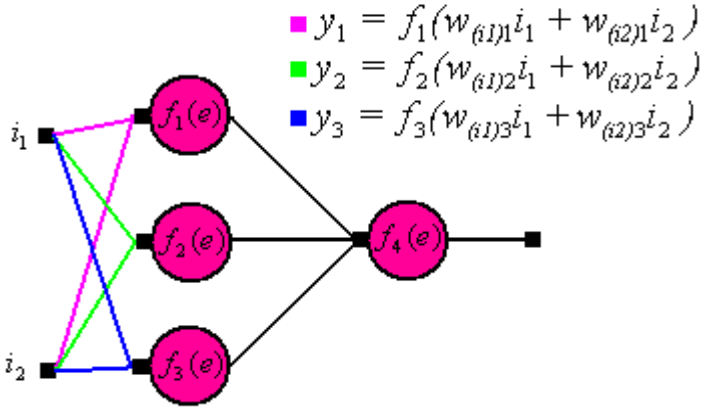
6. Repeat from step 3 on the neurons at the previous level using each one's "blame" as its error.

$$w^h_{j,i} \leftarrow w^h_{j,i} + \Delta w^h_{j,i}$$

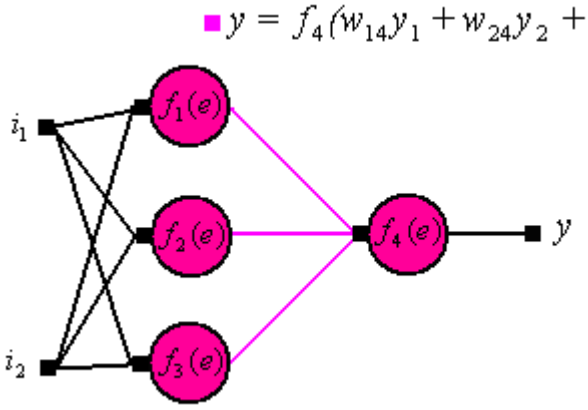
The below series of graphs give a visual representation of back-error propagation algorithm.

η = Learning Coefficient, δ = Error, y = Output, $f_i(e)$ = Bias Function, w = connection weight, i = Input Data

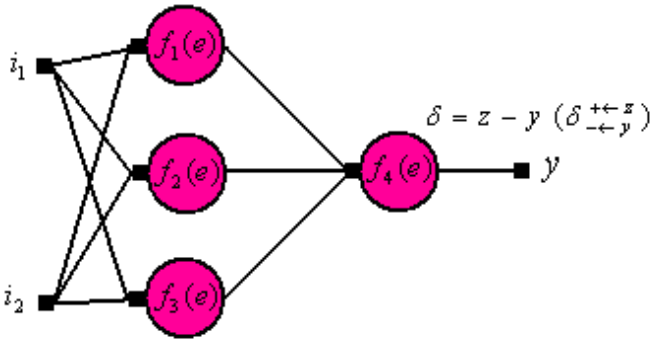
First, input the data and calculate the middle layer's neuron values.



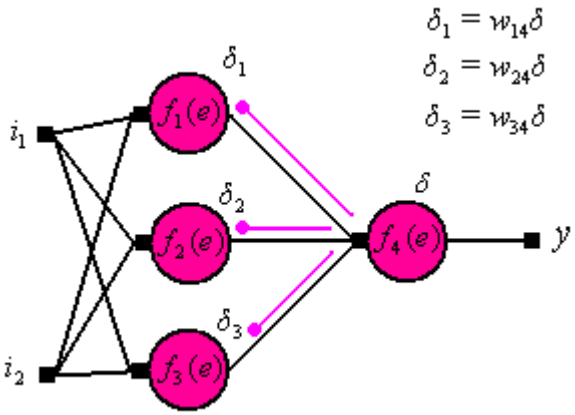
Calculate the output layer's neuron values.



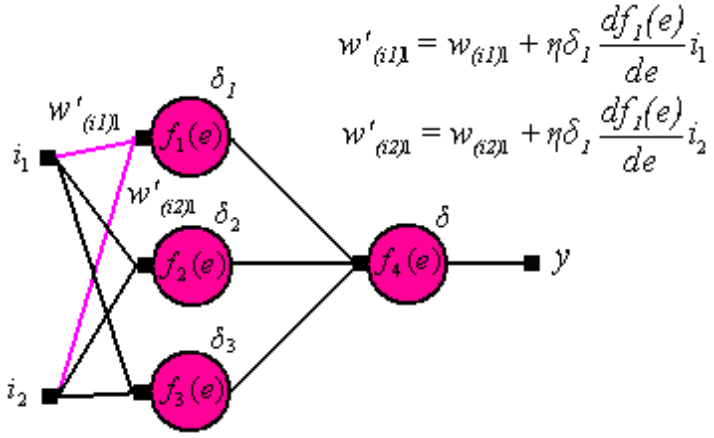
Calculate the difference between the actual output and the expected output.



Back-propagate, calculating the error of each of the connection weights in the center and input layers.

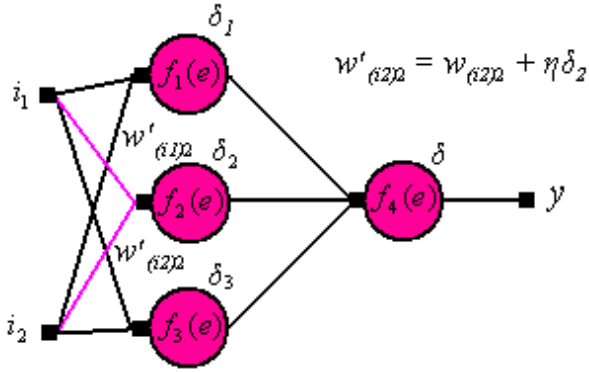


Next, forward propagate, adjusting each of the connection weights based on their errors.



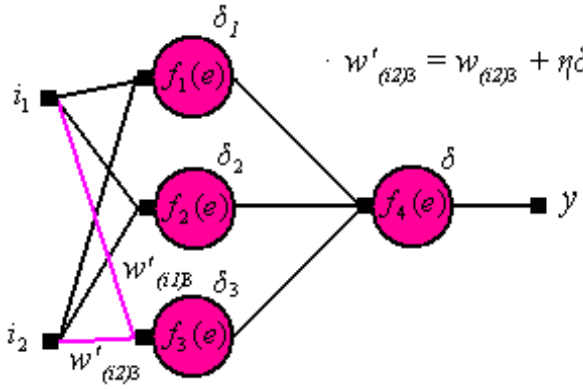
$$w'_{(i1)2} = w_{(i1)2} + \eta\delta_2 \frac{df_2(e)}{de} i_1$$

$$w'_{(i2)2} = w_{(i2)2} + \eta\delta_2 \frac{df_2(e)}{de} i_2$$



$$w'_{(i1)3} = w_{(i1)3} + \eta\delta_3 \frac{df_3(e)}{de} i_1$$

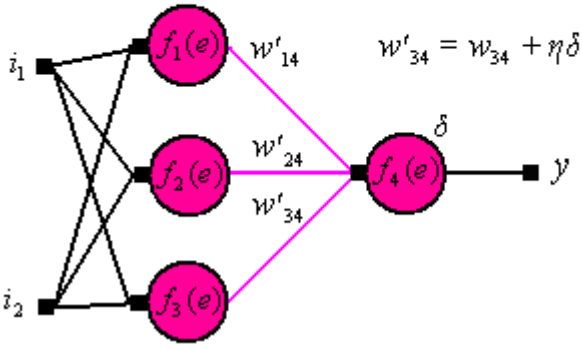
$$w'_{(i2)3} = w_{(i2)3} + \eta\delta_3 \frac{df_3(e)}{de} i_2$$



$$w'_{14} = w_{14} + \eta\delta \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta \frac{df_4(e)}{de} y_3$$



4. Analysis of Back-Error Propagation

The following tables summarize the results of the training of the Logical AND function under various conditions. For this simple experiment a two input Logical AND gate was taught. Its truth table can be seen below.

Logical AND		
p	q	Output
0	0	0
0	1	0
1	0	0
1	1	1

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	2	0.1	0.1	844
1	2	0.1	0.01	2688
1	2	0.1	0.001	9685
1	2	0.1	0.0001	38063

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	2	0.2	0.1	407
1	2	0.2	0.01	1250
1	2	0.2	0.001	4543
1	2	0.2	0.0001	29578

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	2	0.3	0.1	289
1	2	0.3	0.01	740
1	2	0.3	0.001	2768
1	2	0.3	0.0001	19915

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	2	0.4	0.1	241
1	2	0.4	0.01	555
1	2	0.4	0.001	9507
1	2	0.4	0.0001	11953

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	2	0.5	0.1	157
1	2	0.5	0.01	425
1	2	0.5	0.001	1595
1	2	0.5	0.0001	10269

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles

1	3	0.1	0.1	672
1	3	0.1	0.01	2035
1	3	0.1	0.001	7819
1	3	0.1	0.0001	52187

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.2	0.1	266
1	3	0.2	0.01	997
1	3	0.2	0.001	4174
1	3	0.2	0.0001	23568

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.3	0.1	191
1	3	0.3	0.01	659
1	3	0.3	0.001	2504
1	3	0.3	0.0001	15789

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.4	0.1	147
1	3	0.4	0.01	509
1	3	0.4	0.001	1847
1	3	0.4	0.0001	15233

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.5	0.1	116
1	3	0.5	0.01	394
1	3	0.5	0.001	1601
1	3	0.5	0.0001	9215

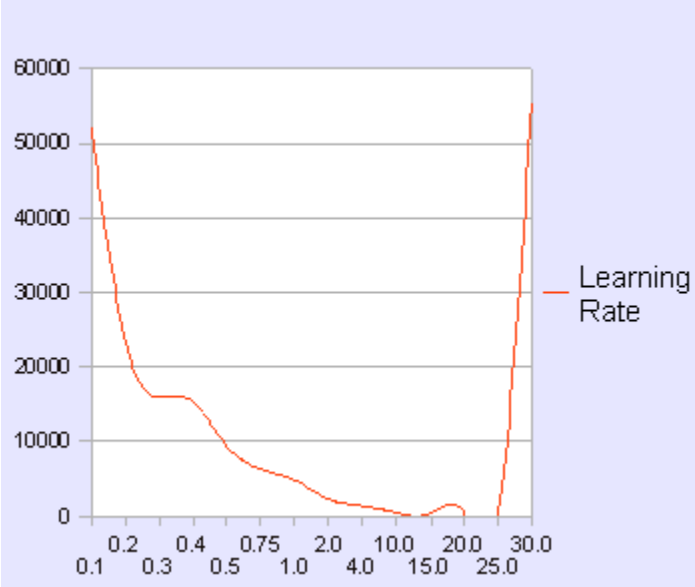
Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.75	0.1	103
1	3	0.75	0.01	299
1	3	0.75	0.001	1046
1	3	0.75	0.0001	6261

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	1.0	0.1	106
1	3	1.0	0.01	243
1	3	1.0	0.001	843
1	3	1.0	0.0001	470

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles

	Neurons			
1	3	2.0	0.1	40
1	3	2.0	0.01	98
1	3	2.0	0.001	339
1	3	2.0	0.0001	2183

Results using back-error propagation algorithm to train Logical AND				
Number of Center Layers	Number of Center Layer Neurons	Learning Rate (Teacher Signal)	Error	Average number of necessary training cycles
1	3	0.1	0.0001	52187
1	3	0.2	0.0001	23568
1	3	0.3	0.0001	15789
1	3	0.4	0.0001	15233
1	3	0.5	0.0001	9215
1	3	0.75	0.0001	6261
1	3	1.0	0.0001	4701
1	3	2.0	0.0001	2183
1	3	4.0	0.0001	1293
1	3	10.0	0.0001	415
1	3	15.0	0.0001	302
1	3	20.0	0.0001	294
1	3	25.0	0.0001	273
1	3	30.0	0.0001	80143
1	3	> 30.0	0.0001	Unable to train



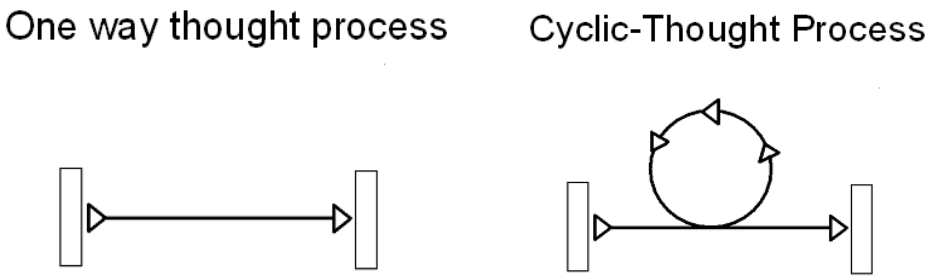
The overall conclusion of the data suggests that the higher the learning rate is the faster that the neural network learns the supplied data. However, if the learning rate is too high the neural network can not learn. This can be looked at as training a kid. The more seriously you scold a kid, the stronger he/she will react, but if you punish the kid too severely, i.e. extreme physical abuse, the kid cannot associate the punishment with what it has down wrong.

This result holds true for neural networks in which the number of center layers and neurons in each of the center layers is low. In the case of larger neural networks different conditions are required to elicit an optimal result. For example, maybe the maximum learning rate is a low value like 0.1. whereas in our previous example a value of 12~14 provides an optimal result.

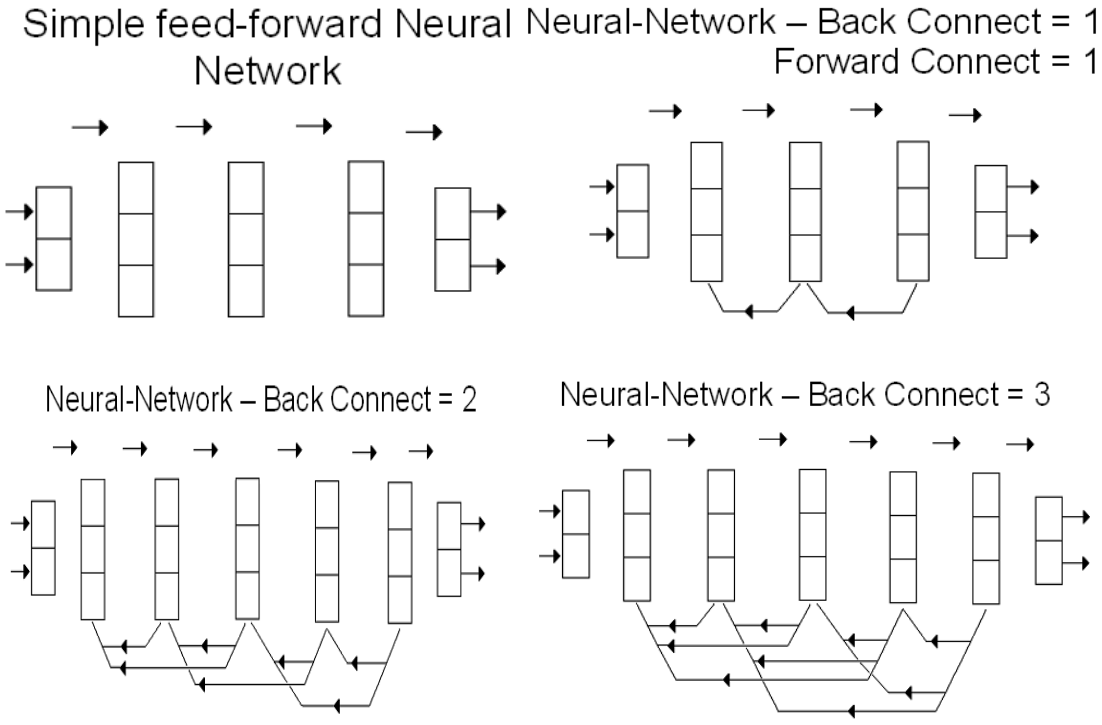
5. Other Designs

Why is it important to think about other designs considering there are already many working models? It is simple. In many neural network learning algorithms, for example, back-error propagation is more or less just trying to minimize error by adjusting matrices values. This is not how biological neurons function. Also, biological neurons more or less function as individual entities and not necessarily as coupled as they are in many artificial neural network models. So looking at biological functionality more before designing an artificial neural network will provide more insight on designing a better learning algorithm. In fact, instead of programming a learning algorithm, perhaps only programming a neuron that emulates a biological neuron, then putting the neurons together will be enough to recreate artificial intelligence, and then again maybe it will not. Other food for thought is that there are many features present in a biological neural network that may not be necessary to create an equally functional artificial neuron. For example, neuron shape in biology plays an enormous role in neuron function. However, in an artificial neural network it may be possible to ignore this property, as long as other functionality is not hindered.

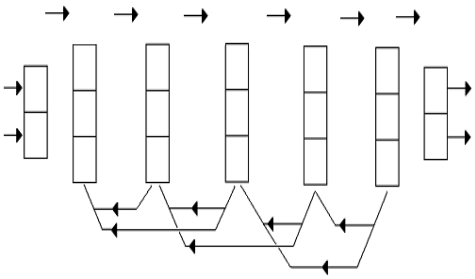
Below are some diagrams that demonstrate types of thought processes. In simple feed-forward networks where data only travels in one direction, an internal cyclic thought cycle is not possible. This means that if input from the environment is stopped at any point, the neural network can not longer “think,” i.e. produce output. This is not the case in organisms, such as human beings, where even if all input is cut off, we are still capable of maintaining a thought cycle. For example if you shut your eyes and go into a quiet neutral room, you can still count and think of other complex logical problems.



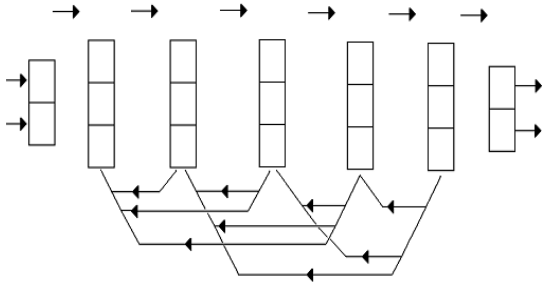
The above diagrams demonstrate a one way thought process vs. a cyclic-thought cycle. Closely observe the below diagrams and note that as the number of layers that each layer is connected to increases, the ability to maintain an internal cyclic thought also increases. These sort of neural networks are referred to as recurrent neural networks.



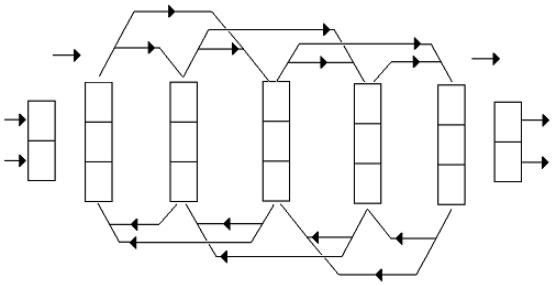
Neural-Network – Back Connect = 2



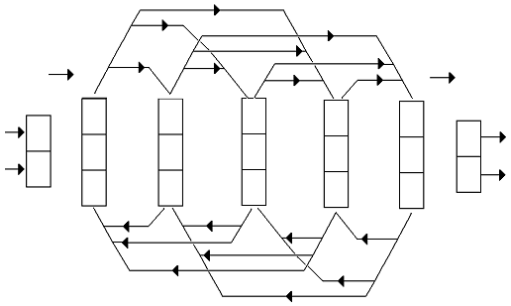
Neural-Network – Back Connect = 3



Neural-Network – Back Connect = 2
Forward Connect = 2

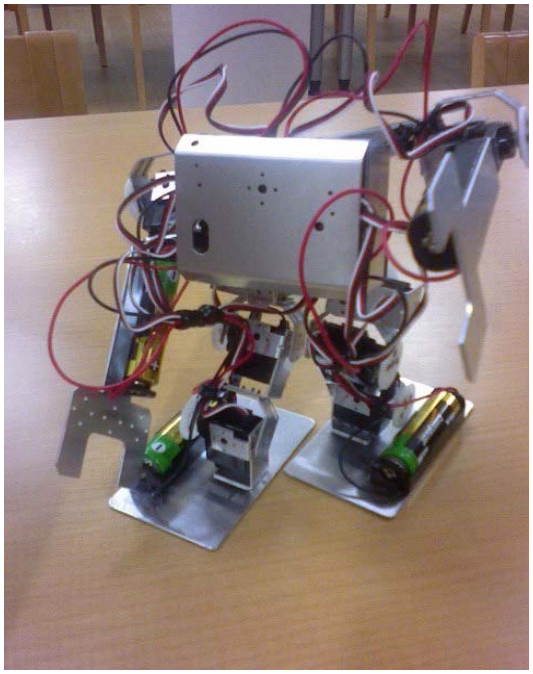


Neural-Network – Back Connect = 3
Forward Connect = 3



6. Conclusion

The purpose of this document is to analyze many aspects and factors to consider when designing an artificial neural network. While there are many models available, each has its limitations. In a world where technology is growing fast, so is the desire to have software and robots capable of high level thought. Nevertheless, in the world of AI the biggest challenge lies more in the limitations of the hardware than in our ability to develop hardy learning algorithms.



7. Sources

1. Wikipedia - http://en.wikipedia.org/wiki/Neurons

2. Wikipedia - http://en.wikipedia.org/wiki/Dopamine

3. Wikipedia - http://en.wikipedia.org/wiki/Backpropagation

8. Neural Network Source Code

(See attached Source)

Included source:

NN, Single Center Layer, back-error propagation

NN, Multiple Center Layers, back-error propagation

NN, Multiple Center Layers, Neuron objects, back-error propagation