Friday, October 11, 2013

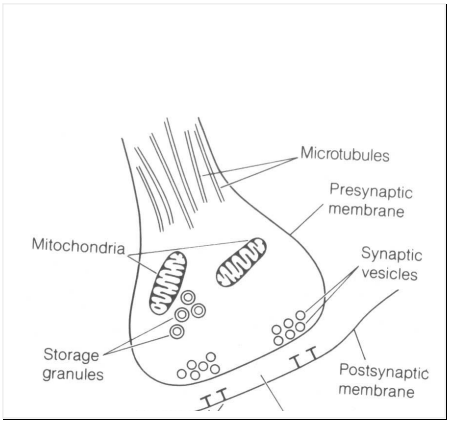
2.6 Introduction to Neural Networks :csdrive

Introduction:

The central nervous system [CNS] is composed entirely of two kinds of specialized cells: neurons and glia. Hence, every information processing system in the CNS is composed of neurons and glia; so too are the networks that compose the systems (and the maps).

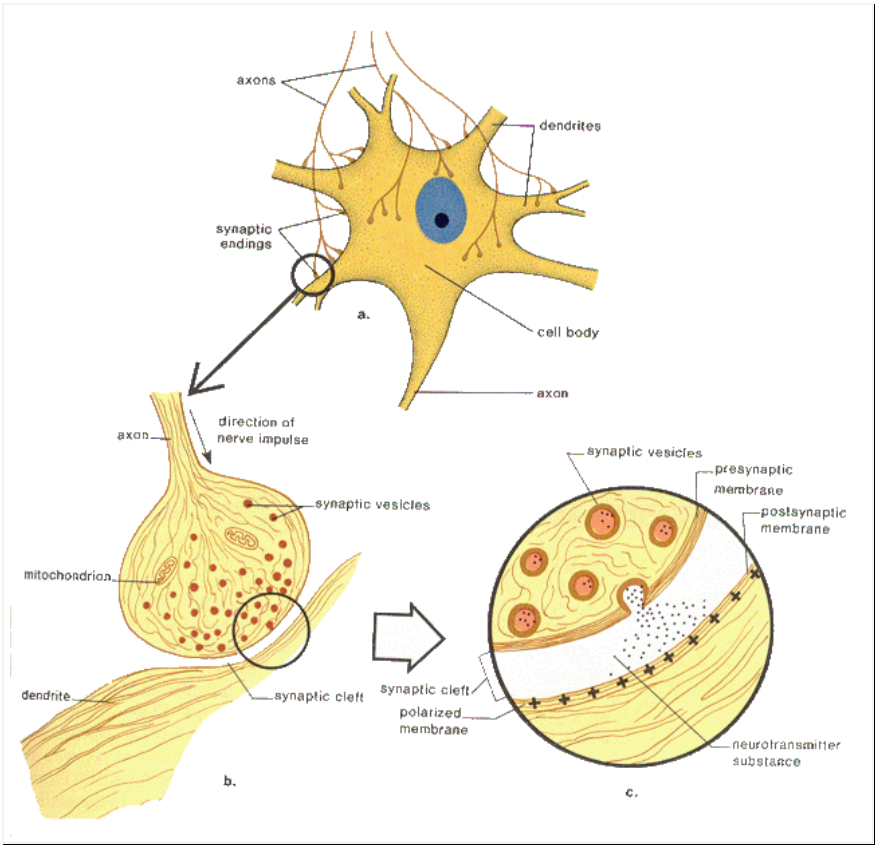
Neurons are the basic information processing structures in the CNS. Everything occurring above the level of neurons qualifies as information processing too. But nothing below the level of neurons does.

A "typical" neuron has four distinct parts (or regions). The first part is the cell body(or soma). This is not only the metabolic "control center" of the neuron, it is also its "manufacturing and recycling plant." (For instance, it is within the cell body that neuronal proteins are synthesized.) The second and third parts are processes — structures that extend away from the cell body. Generally speaking, the function of a process is to be a conduit through which signals flow to or away from the cell body. Incoming signals from other neurons are (typically) received through its dendrites. The outgoing signal to other neurons flows along its axon. A neuron may have many thousands of dendrites, but it will have only one axon. The fourth distinct part of a neuron lies at the end of the axon, the axon terminals. These are the structures that contain neurotransmitters. Neurotransmitters are the chemical medium through which signals flow from one neuron to the next at chemical synapses.

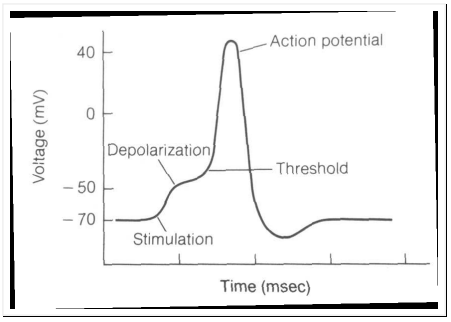


Synapse is a small gap at the end of a neuron that allows information to pass from one neuron to the next. Synapses are found where nerve cells connects with other nerve cells as well as where nerve cells connect with muscles and glands.

Synapses are composed of three main parts:



- The presynaptic ending that contains neurotransmitters
- The synaptic cleft between the two nerve cells
- The postsynaptic ending that contains receptor sites



An electrical impulse travels down the axon of a neuron, and then triggers the release of neurotransmitters. These chemical messengers cross the synaptic cleft and connect with receptor sites in the next nerve cell, triggering an electrical impulse known as an action potential.

Artificial Neural Network is a system loosely modeled based on the human brain. The field goes by many names, such as connectionism, parallel distributed processing, neuro-computing, natural intelligent systems, machine learning algorithms, and artificial neural networks. It is inherently multiprocessor-friendly architecture and without much modification, it goes beyond one or even two processors of the von Neumann architecture. It has ability to account for any functional dependency. The network discovers (learns, models) the nature of the dependency without needing to be prompted. No need to postulate a model, to amend it, etc.

Biological Neural Network	*Artificial Neural Network*
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

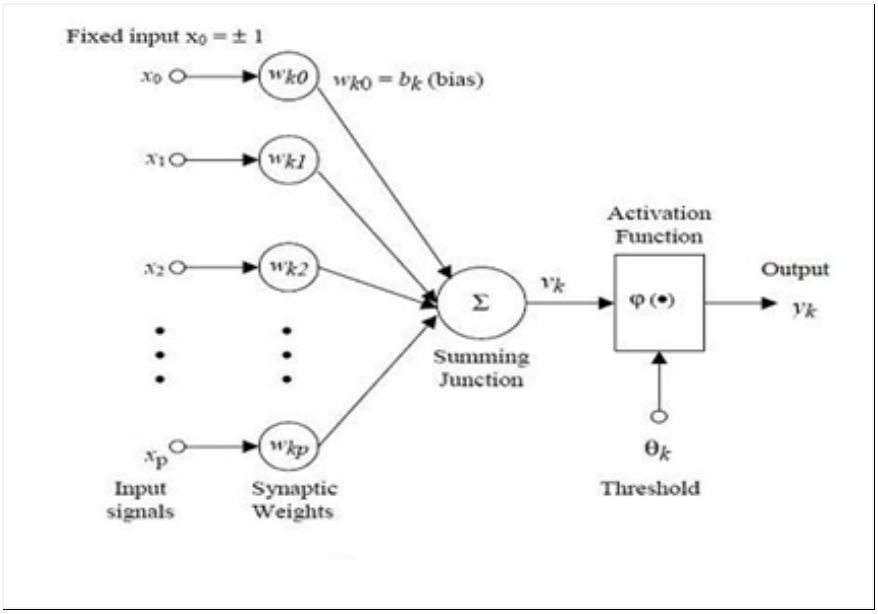
The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

1. A neural network acquires knowledge through learning.

2. A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled. Traditional linear models are simply inadequate when it comes to modeling data that contains non-linear characteristics.

Mathematical Model:



Working Principle:

They typically consist of many simple processing units, which are wired together in a complex communication network. There is no central CPU following a logical sequence of rules - indeed there is no set of rules or program. This structure then is close to the physical workings of the brain and leads to a new type of computer that is rather good at a range of complex tasks. The network "learns" mainly by modifying the weights of the neurons. An adjustment of the threshold can be effected by other neuron.

Various inputs to the network are represented by the mathematical symbol, x(n). Each of these inputs are multiplied by a connection weight. These weights are represented by w(n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output. This process lends itself to physical implementation on a large scale in a small package. This electronic implementation is still possible with other network structures which utilize different summing functions as well as different transfer functions.

Major Components of an Artificial Neuron

This section describes the seven major components which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers.

Component 1. Weighting Factors:

A neuron usually receives many simultaneous inputs. Each input has its own relative weight which gives the input the impact that it needs on the processing element's summation function. These weights perform the same type of function as do the the varying synaptic strengths of biological neurons. In both cases, some inputs are made more important than others so that they have a greater effect on the processing element as they combine to produce a neural response.

Weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. They are a measure of an input's connection strength. These strengths can be modified in response to various training sets and according to a network's specific topology or through its learning rules.

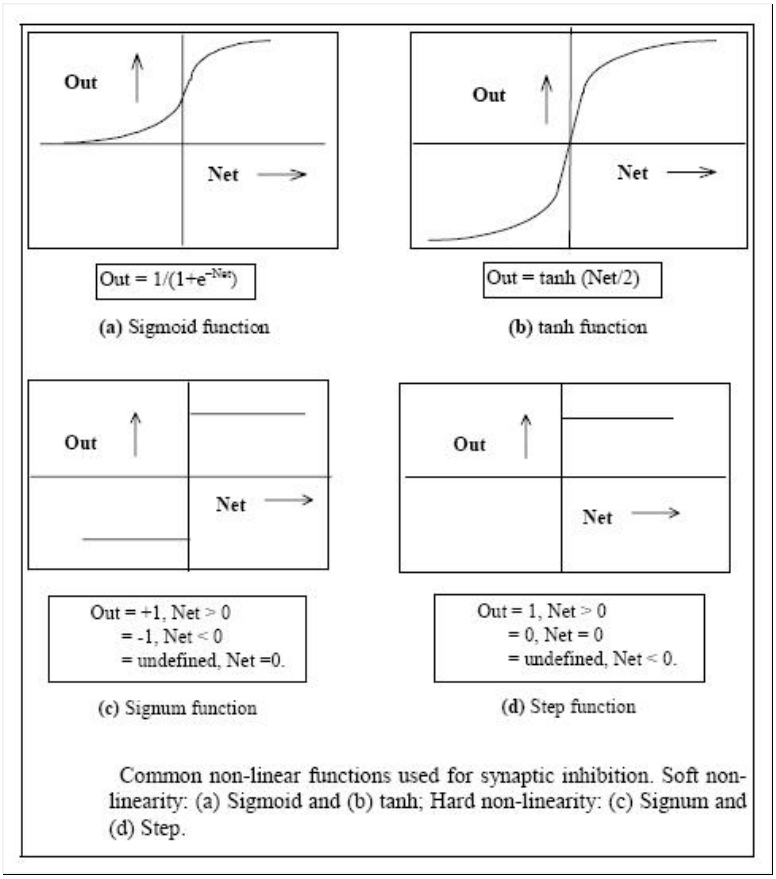
Component 2. Summation Function:

The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (i1, i2 . . . i*n*) and (w1, w2 . . . w*n*). The total input signal is the dot, or inner, product of these two vectors. This simplistic summation function is found by multiplying each component of the i vector by the corresponding component of the w vector and then adding up all the products. Input1 = i1 * w1, input2 = i2 * w2, etc., are added as input1 + input2 + . . . + input *n*. The result is a single number, not a multi-element vector.

Geometrically, the inner product of two vectors can be considered a measure of their similarity. If the vectors point in the same direction, the inner product is maximum; if the vectors point in opposite direction (180 degrees out of phase), their inner product is minimum.
The summation function can be more complex than just the simple input and weight sum of products. The input and weighting coefficients can be combined in many different ways before passing on to the transfer function. In addition to a simple product summing, the summation function can select the minimum, maximum, majority, product, or several normalizing algorithms. The specific algorithm for combining neural inputs is determined by the chosen network architecture and paradigm.

Some summation functions have an additional process applied to the result before it is passed on to the transfer function. This process is sometimes called the activation function. The purpose of utilizing an activation function is to allow the summation output to vary with respect to time. Activation functions currently are pretty much confined to research. Most of the current network implementations use an "identity" activation function, which is equivalent to not having one. Additionally, such a function is likely to be a component of the network as a whole rather than of each individual processing element component.

Component 3. Transfer Function:



Common non-linear functions used for synaptic inhibition. Soft non-linearity: (a) Sigmoid and (b) tanh; Hard non-linearity: (c) Signum and (d) Step.

The result of the summation function, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the processing element generates a signal. If the sum of the input and weight products is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of response are significant.

The threshold, or transfer function, is generally non-linear. Linear (straight-line) functions are limited because the output is simply proportional to the input. Linear functions are not very useful. That was the problem in the earliest network models as noted in Minsky and Papert's book *Perceptrons*.

The transfer function could be something as simple as depending upon whether the result of the summation function is positive or negative. The network could output zero and one, one and minus one, or other numeric combinations. The transfer function would then be a "hard limiter" or step function. See Figure above for sample transfer functions.

Another type of transfer function, the threshold or ramping function, could mirror the input within a given range and still act as a hard limiter outside that range. It is a linear function that has been clipped to minimum and maximum values, making it non-linear. Yet another option would be a sigmoid or S-shaped curve. That curve approaches a minimum and maximum value at the asymptotes. It is common for this curve to be called a sigmoid when it ranges between 0 and 1, and a hyperbolic tangent when it ranges between -1 and 1. Mathematically, the exciting feature of

these curves is that both the function and its derivatives are continuous. This option works fairly well and is often the transfer function of choice. Other transfer functions are dedicated to specific network architectures and will be discussed later.

Prior to applying the transfer function, uniformly distributed random noise may be added. The source and amount of this noise is determined by the learning mode of a given network paradigm. This noise is normally referred to as "temperature" of the artificial neurons. The name, temperature, is derived from the physical phenomenon that as people become too hot or cold their ability to think is affected. Electronically, this process is simulated by adding noise. Indeed, by adding different levels of noise to the summation result, more brain-like transfer functions are realized. To more closely mimic nature's characteristics, some experimenters are using a gaussian noise source. Gaussian noise is similar to uniformly distributed noise except that the distribution of random numbers within the temperature range is along a bell curve. The use of temperature is an ongoing research area and is not being applied to many engineering applications.

NASA just announced a network topology which uses what it calls a temperature coefficient in a new feed-forward, back-propagation learning function. But this temperature coefficient is a global term which is applied to the gain of the transfer function. It should not be confused with the more common term, temperature, which is simple noise being added to individual neurons. In contrast, the global temperature coefficient allows the transfer function to have a learning variable much like the synaptic input weights. This concept is claimed to create a network which has a significantly faster (by several order of magnitudes) learning rate and provides more accurate results than other feedforward, back-propagation networks.

Component 4. Scaling and Limiting:

After the processing element's transfer function, the result can pass through additional processes which scale and limit. This scaling simply multiplies a scale factor times the transfer value, and then adds an offset. Limiting is the mechanism which insures that the scaled result does not exceed an upper or lower bound. This limiting is in addition to the hard limits that the original transfer function may have performed.

This type of scaling and limiting is mainly used in topologies to test biological neuron models, such as James Anderson's brain-state-in-the-box.

Component 5. Output Function (Competition):

Each processing element is allowed one output signal which it may output to hundreds of other neurons. This is just like the biological neuron, where there are many inputs and only one output action. Normally, the output is directly equivalent to the transfer function's result. Some network topologies, however, modify the transfer result to incorporate competition among neighboring processing elements. Neurons are allowed to compete with each other, inhibiting processing elements unless they have great strength. Competition can occur at one or both of two levels. First, competition determines which artificial neuron will be active, or provides an output. Second, competitive inputs help determine which processing element will participate in the learning or adaptation process.

Component 6. Error Function and Back-Propagated Value:

In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match a particular network architecture. The most basic architectures use this error directly, but some square the error while retaining its sign, some cube the error, other paradigms modify the raw error to fit their specific purposes. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is sometimes called the current error.

The current error is typically propagated backwards to a previous layer. Yet, this back-propagated value can be either the current error, the current error scaled in some manner (often by the derivative of the transfer function), or some other desired output depending on the network type. Normally, this back-propagated value, after being scaled by the learning function, is multiplied against each of the incoming connection weights to modify them before the next learning cycle.

Component 7. Learning Function:

The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result can also be called the adaption function, as well as the learning mode. There are two types of learning: supervised and unsupervised. Supervised learning requires a teacher. The teacher may be a training set of data or an observer who grades the performance of the network results. Either way, having a teacher is learning by reinforcement. When there is no external teacher, the system must organize itself by some internal criteria designed into the network. This is learning by doing. [7]

Conclusion:

A neural network can perform tasks that a linear program cannot. When an element of the neural network fails, it can

continue without any problem by their parallel nature. A neural network learns and does not need to be reprogrammed. But, the neural network needs training to operate. The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated. Plus, it requires high processing time for large neural networks.[2]

Reference:

[1]http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
[2]http://www.nd.com/welcome/whatisnn.htm
[3]http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html
[4]http://www.makhfi.com/tutorial/introduction.htm
[5]http://thecustomizewindows.com/2012/01/artificial-neural-network-ann/
[6]http://www.learnartificialneuralnetworks.com/introduction-to-neural-networks.html
[7]http://irrigation.rid.go.th/rid15/ppn/Knowledge/Artificial%20Neuron%20Networks%20Technology/4.0%20Neural%20Networks%20Components.htm

+1 Recommend this on Google

Labels: ics, neural networks, neurons, synapse

No comments:

Post a Comment

Enter your comment...

Comment as: sagar giri (Goo

Sign out

Publish Preview Notify me

3.1 Computation and Cognition : csdrive

2.5 Expert System :csdrive