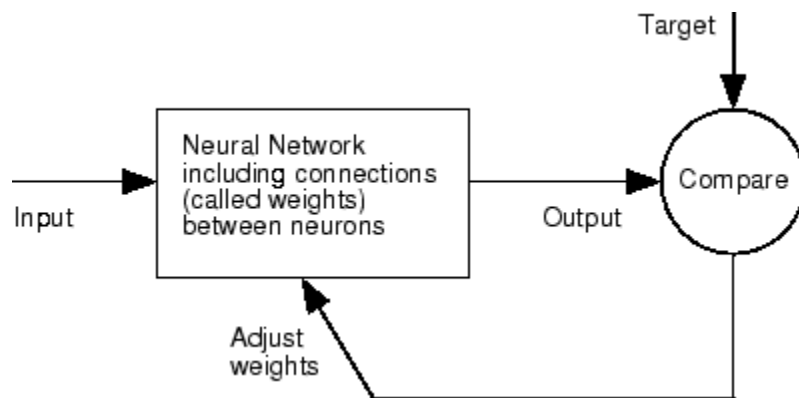


Artificial Neural Networks (ANN)



Artificial neural nets have a number of properties that make them an attractive alternative to traditional problem-solving techniques. The two main alternatives to using neural nets are to develop an algorithmic solution, and to use an expert system.

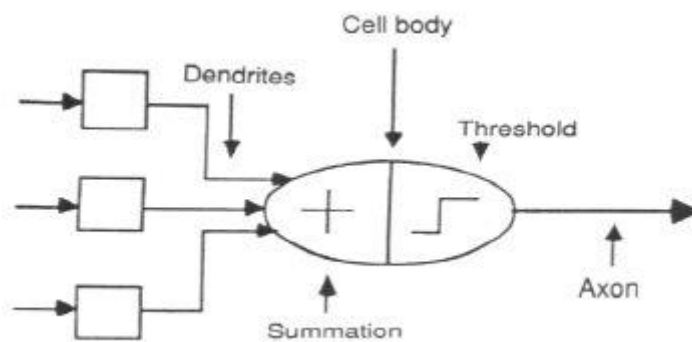
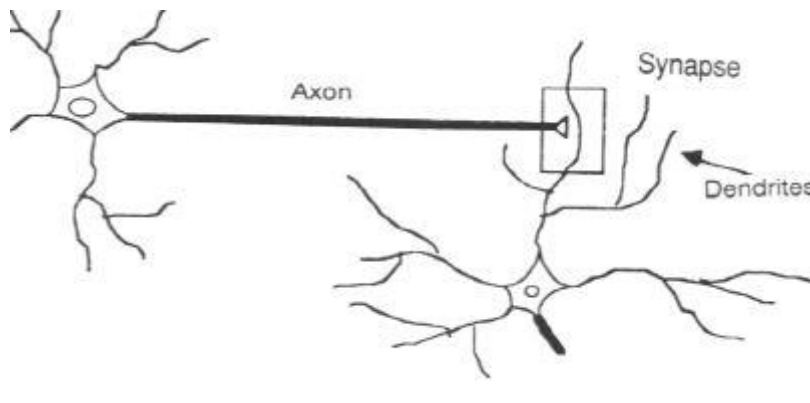
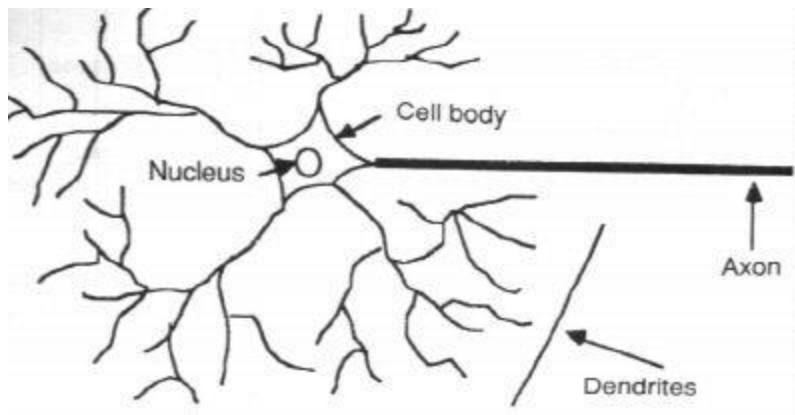
Algorithmic methods arise when there is sufficient information about the data and the underlying theory. By understanding the data and the theoretical relationship between the data, we can directly calculate unknown solutions from the problem space. Ordinary von Neumann computers can be used to calculate these relationships quickly and efficiently from a numerical algorithm.

Expert systems, by contrast, are used in situations where there is insufficient data and theoretical background to create any kind of a reliable problem model. In these cases, the knowledge and rationale of human experts is codified into an expert system. Expert systems emulate the deduction processes of a human expert, by collecting information and traversing the solution space in a directed manner. Expert systems are typically able to perform very well in the absence of an accurate problem model and complete data. However, where sufficient data or an algorithmic solution is available, expert systems are a less than ideal choice.

Artificial neural nets are useful for situations where there is an abundance of data, but little underlying theory. The data, which typically arises through extensive experimentation may be non-linear, non-stationary, or chaotic, and so may not be easily modeled. Input-output spaces may be so complex that a reasonable traversal with an expert system is not a satisfactory option.

Importantly, neural nets do not require any a priori assumptions about the problem space, not even information about statistical distribution. Though such assumptions are not required, it has been found that the addition of such a priori information as the statistical distribution of the input space can help to speed training. Many mathematical problem models tend to assume that data lies in a standard distribution pattern, such as Gaussian or Maxwell-Boltzmann distributions. Neural networks require no such assumption. During training, the

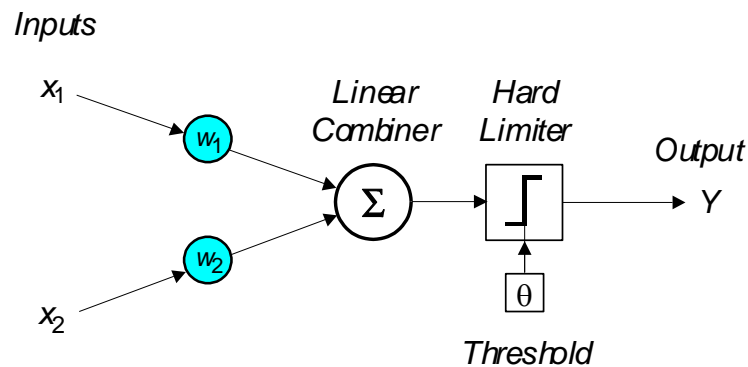
neural network performs the necessary analytical work, which would require non-trivial effort on the part of the analyst if other methods were to be used.



Perceptron

The perceptron is the simplest form of a neural network. It consists of a single neuron with adjustable synaptic weights and a hard limiter

Single-layer two-input perceptron



Can be used to compute AND, OR but not exclusive or (XOR). This is because it is geometrically equivalent to a hyper plane and the inputs can only be on one side or the other of a plane, XOR would require a region between 2 planes.

Perceptron Training

Algorithm

1. Start a training set of inputs x_i and expected outputs Y .
2. Pick some random values from $[-0.5, +0.5]$ for initial weights w_i
3. Compute output Y with $X = \sum_1^n x_i w_i = \bar{x} \cdot \bar{w}$ and $Y = \text{sign}_\theta(X)$
4. Compare Y with expected output Y_d to find the error $e = Y_d - Y$
5. Use error to calculate adjustments to w_i , $\Delta w_i = \alpha e x_i$
6. Then update the weights for the next iteration $w_i(p + 1) = w_i(p) + \Delta w_i(p)$
7. Iterate steps 3 to 6 to minimise the error. Then the weights arrived at can be used in your perceptron

Logical AND

To be completed by the student

Threshold $\theta = 0.2$, learning rate $\alpha = 0.1$.

[illegible]

Logical OR

To be completed by the student

Threshold $\theta = 0.2$, learning rate $\alpha = 0.1$.

[illegible]

Logical AND

Threshold $\theta = 0.2$, learning rate $\alpha = 0.1$.

[illegible]

Logical OR

Threshold $\theta = 0.2$, learning rate $\alpha = 0.1$.

Epoch	Inputs		Desired output Y_d	Weights		Actual Output Y	Error e	Weight adjustments	
	x_1	x_2		w_1	w_2			Δw_1	Δw_2
1	0	0	0	0.3	-0.1	0	0	0	0
	0	1	1	0.3	-0.1	0	1	0	0.1
	1	0	1	0.3	0.0	1	0	0	0
	1	1	1	0.3	0.0	1	0	0	0
2	0	0	0	0.3	0.0	0	0	0	0
	0	1	1	0.3	0.0	0	1	0	0.1
	1	0	1	0.3	0.1	1	0	0	0
	1	1	1	0.3	0.1	1	0	0	0
3	0	0	0	0.3	0.1	0	0	0	0
	0	1	1	0.3	0.1	0	1	0	0.1
	1	0	1	0.3	0.2	1	0	0	0
	1	1	1	0.3	0.2	1	0	0	0
4	0	0	0	0.3	0.2	0	0	0	0
	0	1	1	0.3	0.2	1	0	0	0
	1	0	1	0.3	0.2	1	0	0	0
	1	1	1	0.3	0.2	1	0	0	0