Fundamentals of Artificial Intelligence

Learning in Neural Networks

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

Neural Networks

- Complex networks of simple computing elements
  - The simple arithmetic computing elements correspond to neurons — the cells that perform information processing in the brain.
  - The network as a whole corresponds to a collection of interconnected neurons.
- Capable of learning from examples
  - Through appropriate learning methods.
- Connectionist approach to computation
  - Exhibit complex global behavior, determined by the connections between the processing elements and element parameters.

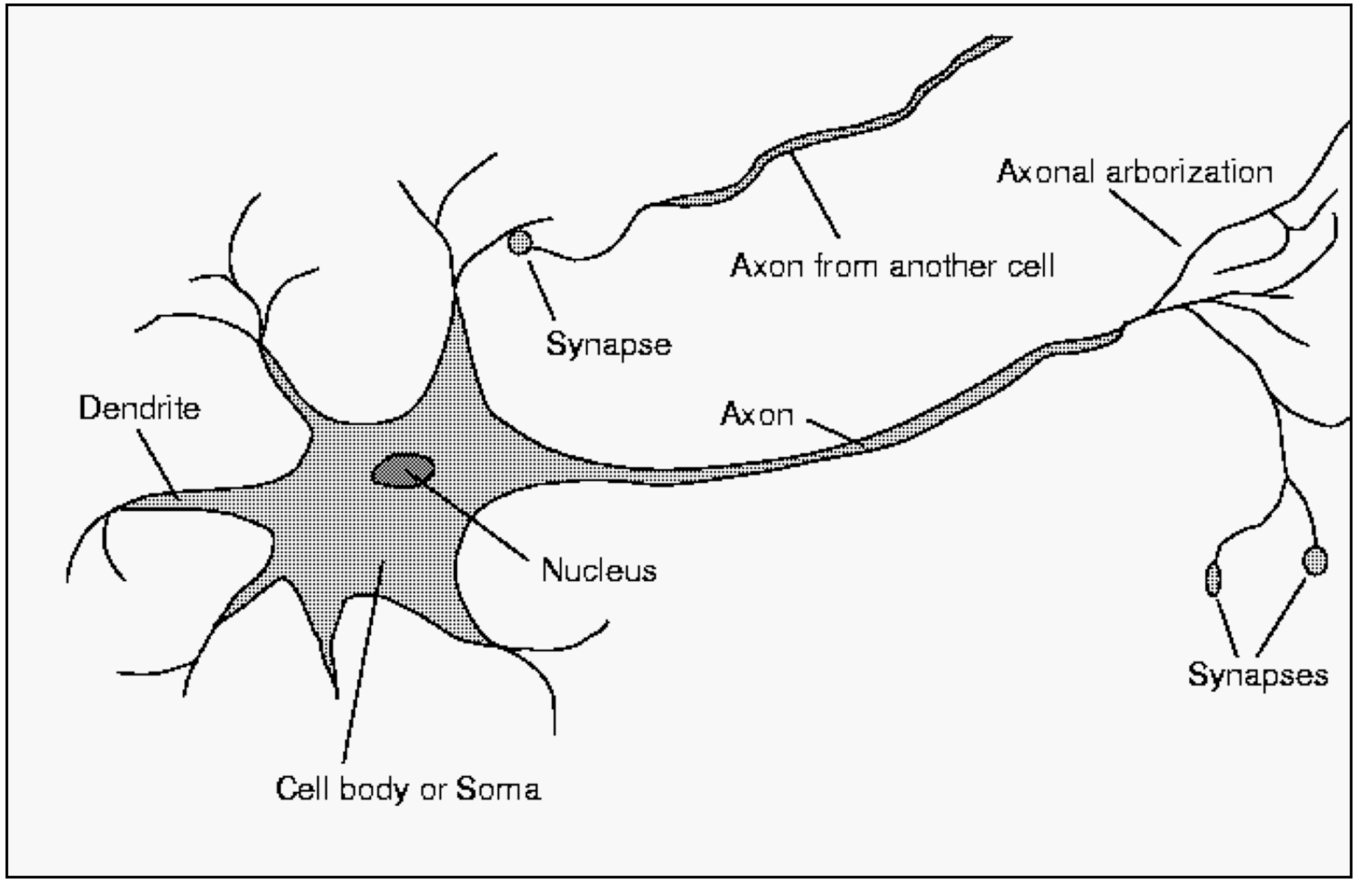
How the Brain Works?

The exact way in which the brain enables thought is one of the great mysteries of science

The neuron is the fundamental functional unit of all nervous system tissue, including the brain.

- Brain
  - Set of interconnected modules
  - Performs information processing operations at various levels
    - sensory input analysis
    - memory storage and retrieval
    - reasoning
    - feelings
    - consciousness
- Neurons
  - basic computational elements
  - heavily interconnected with other neurons

Neuron



Stuart J. Russel and Peter Norvig: Artificial Intelligence – A Modern Approach, Chapter 19, Pages 563-597.

- Soma
  - cell body
- Dendrites
  - Incoming branches
- Axon
  - Single outgoing branch.
- Synapse
  - Junction between a dendrite and an axon from another neuron
  - Each neuron forms synapse with anywhere from a dozen to a hundred thousand other neurons.

Basis for Learning in the Brain

Action potential

- Signals are propagated from neuron to neuron by a complicated electrochemical reaction - transmitter substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body.
- When the potential reaches a threshold, an electrical pulse or action potential is sent down the axon

Excitatory and Inhibitory

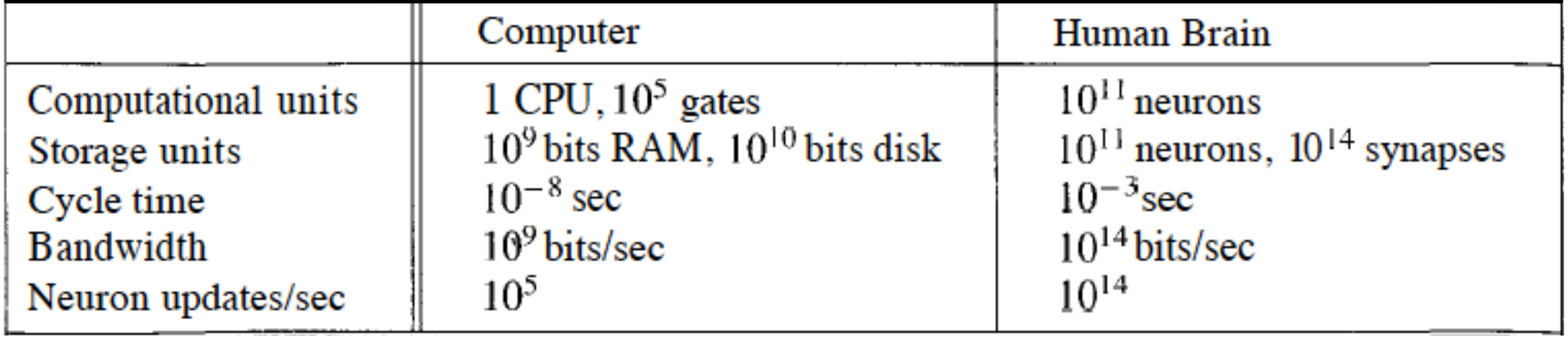
- The pulse spreads out along the branches of the axon, eventually reaching synapses and releasing transmitters into the bodies of other cells.
- Synapses that increase the potential are called excitatory, and those that decrease the potential are called inhibitory.

Plasticity

- Synaptic connections exhibit plasticity — long-term changes in the strength of connections in response to the pattern of stimulation.

Computer vs. Brain

A crude comparison of the raw computational resources available to computers (circa 1994) and brains.



	Computer	Human Brain
Computational units	1 CPU, 10^5 gates	10^{11} neurons
Storage units	10^9 bits RAM, 10^{10} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-8} sec	10^{-3}sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec
Neuron updates/sec	10^5	10^{14}

Stuart J. Russel and Peter Norvig: Artificial Intelligence – A Modern Approach, Chapter 19, Pages 563-597.

Even though a computer is a million times faster in raw switching speed, the brain ends up being a billion times faster at what it does!

Computer chips can execute an instruction in tens of nanoseconds, whereas neurons require milliseconds to fire. Brains more than make up for this, however, because all the neurons and synapses are active simultaneously.

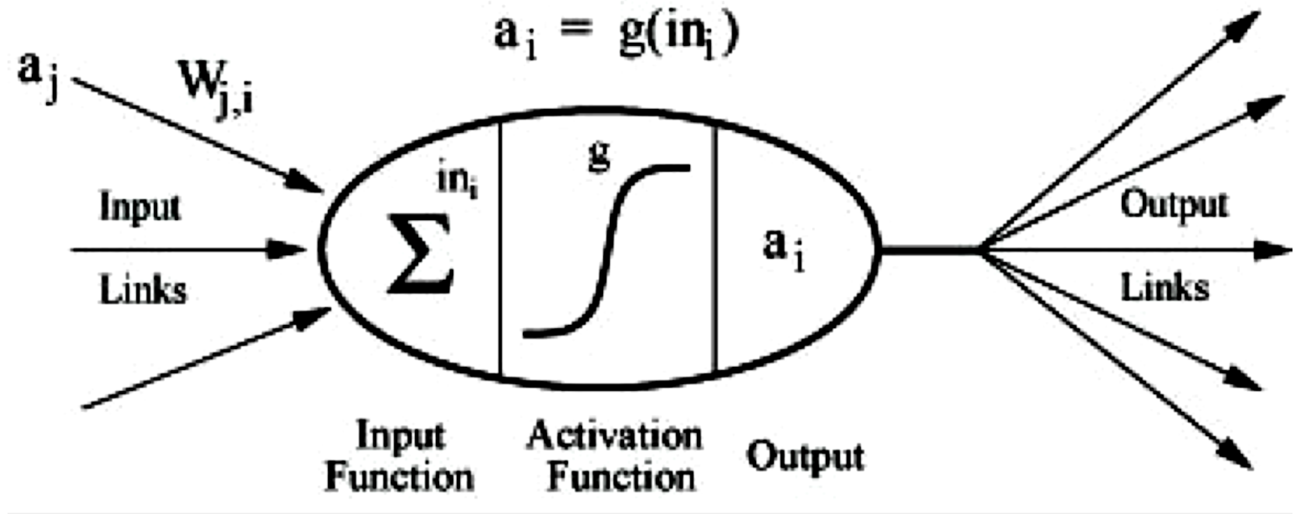
Artificial Neural Networks

- Many neuron-like processing elements
  - Input & output units receive and broadcast signals to the environment, respectively
  - Internal units called hidden units since they are not in contact with external environment
  - Units connected by weighted links (synapses)

- A parallel computation system because
  - Signals travel independently on weighted channels and units can update their state in parallel.
  - However, most ANNs can be simulated in serial computers

Single Computing Element

Artificial Neuron

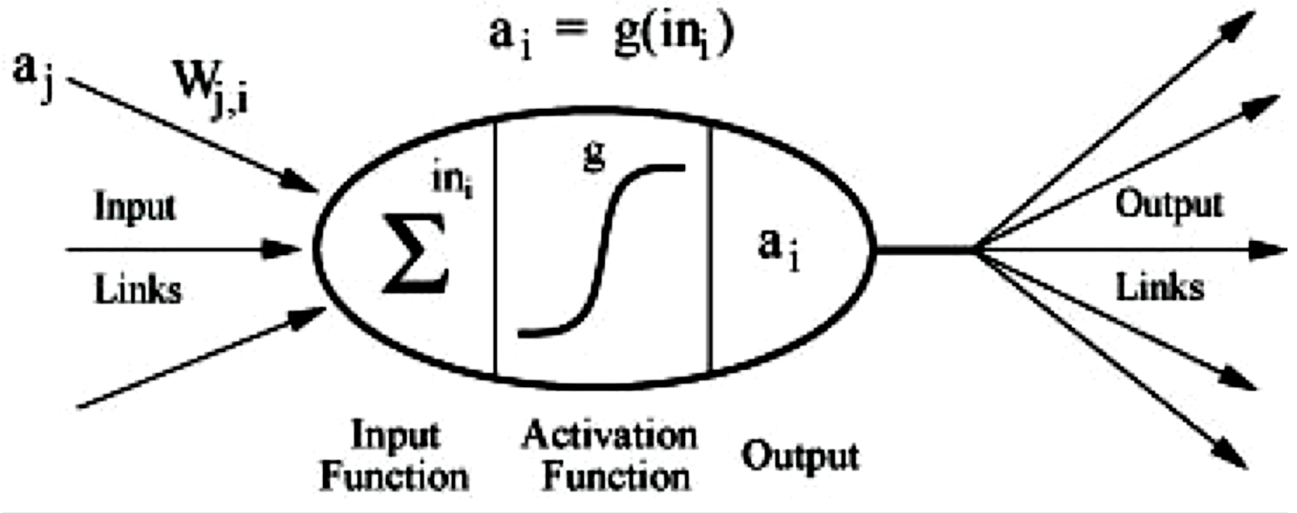


- Receives n-inputs
- Multiplies each input by its weight
- Applies activation function to the sum of results
- Outputs result

Single Computing Element

Artificial Neuron

The computation is split into two components. First is a linear component, called the input function, that computes the weighted sum of the unit's input values.



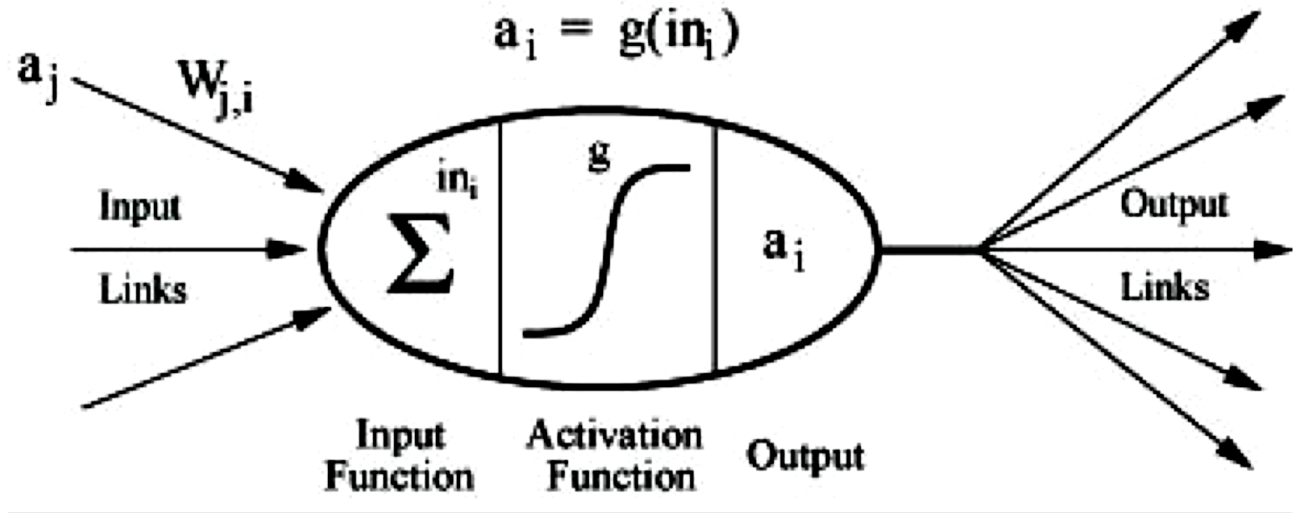
Second is a nonlinear component called the activation function, g, that transforms the weighted sum into the final value that serves as the unit's activation value, a_i.

The total weighted input is the sum of the input activations times their respective weights

$$in_i = \sum_j W_{j,i} a_j = \mathbf{W}_i \cdot \mathbf{a}_i$$

Single Computing Element

Artificial Neuron



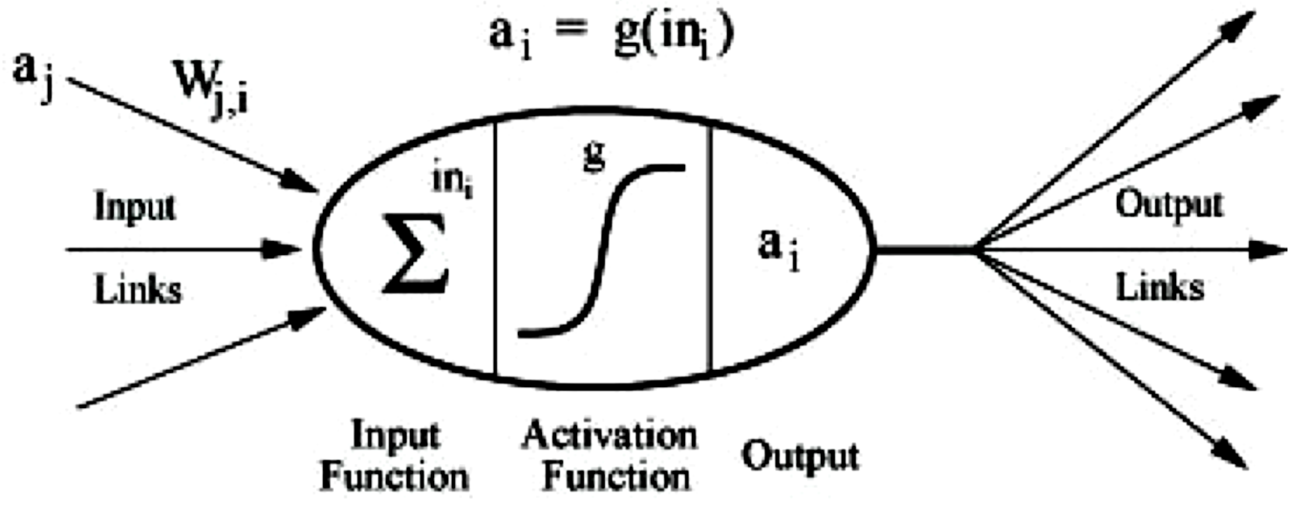
The elementary computation step in each unit computes the new activation value for the unit by applying the activation function, g, to the result of the input function:

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

Single Computing Element

Artificial Neuron

Allows for a simpler learning element because it need only worry about adjusting weights, rather than adjusting both weights and thresholds.



Replace the threshold with an extra input weight. Add an extra input; activation fixed at -1. Weight serves the function of threshold at t.

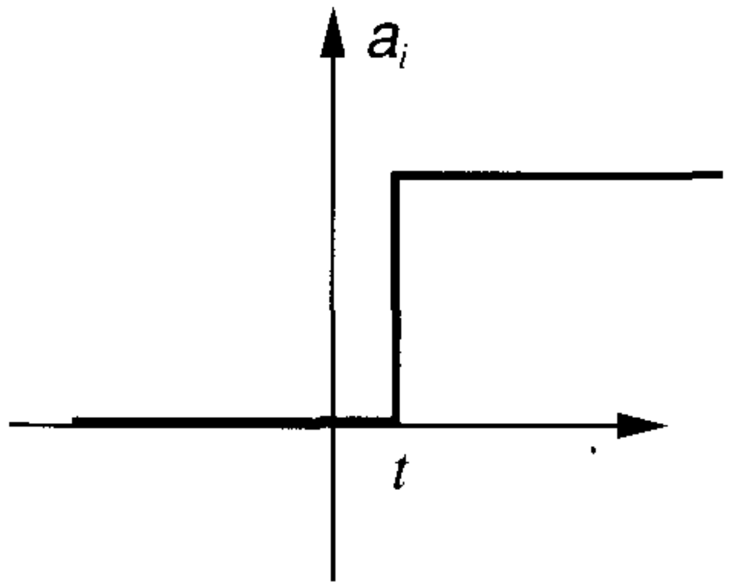
$$a_i = \text{step}_t\left(\sum_{j=1}^{n} W_{j,i}a_j\right) = \text{step}_0\left(\sum_{j=0}^{n} W_{j,i}a_j\right) \text{ where } W_{0,i} = t \text{ and } a_0 = -1$$

Common Activation Functions

Different models are obtained by using different mathematical functions for g. Three common choices are the step, sign, and sigmoid functions.

Step function

The step function has a threshold t such that it outputs a 1 when the input is greater than its threshold, and outputs a 0 otherwise.



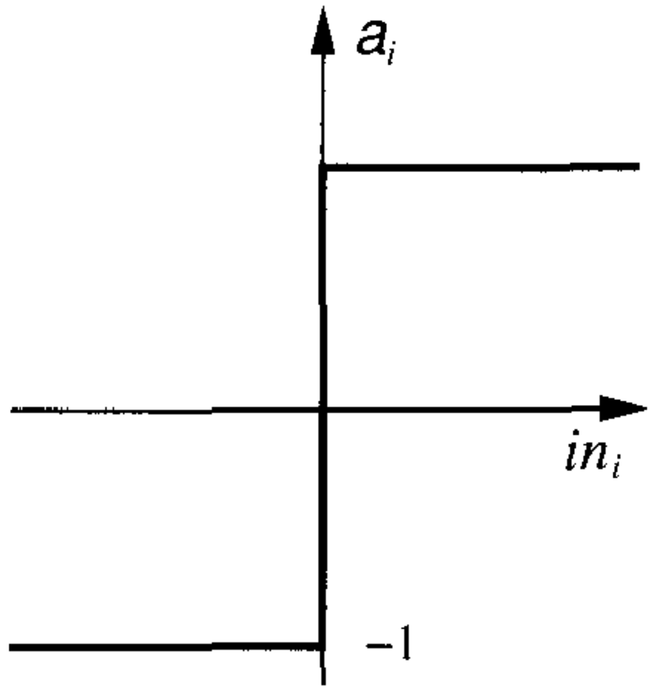
The biological motivation is that a 1 represents the firing of a pulse down the axon, and a 0 represents no firing. The threshold represents the minimum total weighted input necessary to cause the neuron to fire

Common Activation Functions

Different models are obtained by using different mathematical functions for g. Three common choices are the step, sign, and sigmoid functions.

Sign function

Threshold function outputs 1 when input is positive and 0 otherwise

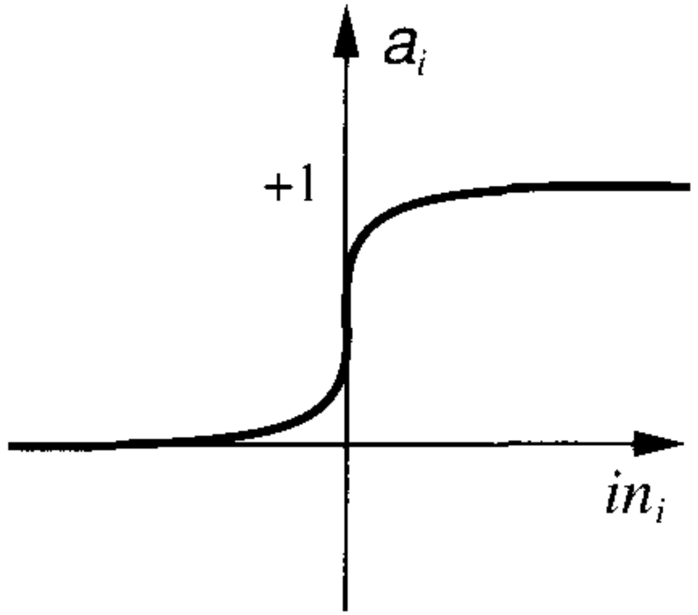


Common Activation Functions

Different models are obtained by using different mathematical functions for g. Three common choices are the step, sign, and sigmoid functions.

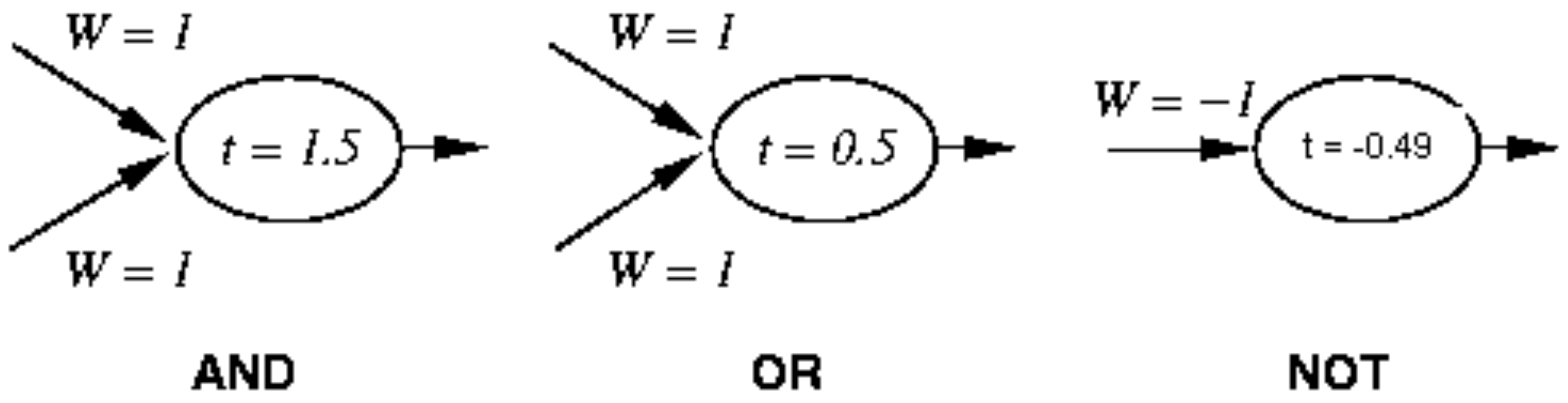
Sigmoid function

$$\text{Sigmoid}(x) = \frac{1}{(1+e^{-X})}$$



Basic Boolean Functions

One of the original motivations for the design of individual units (McCulloch and Pitts, 1943) was their ability to represent basic Boolean functions



Units with a step function for the activation function can act as logic gates, given appropriate thresholds and weights.

- Simple neurons with can act as logic gates
  - Appropriate choice of activation function, threshold, and weights

Network Structures

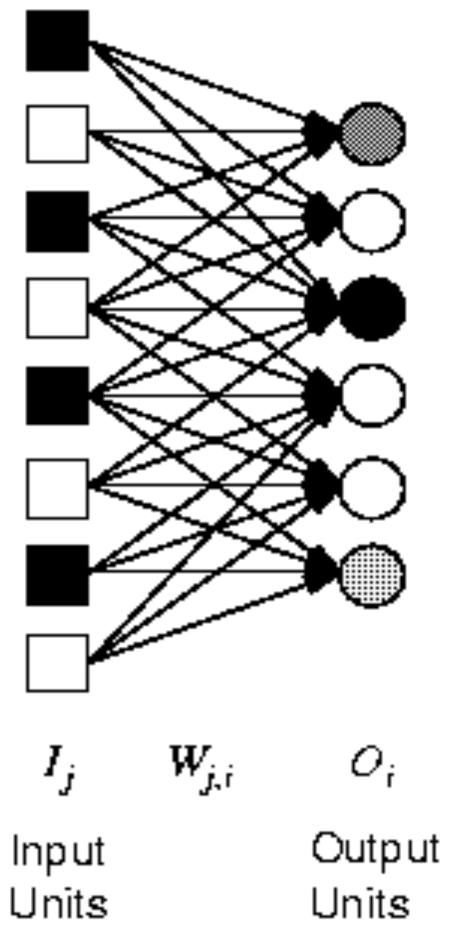
There are a variety of kinds of network structure, each of which results in very different computational properties

- In principle, networks can be arbitrarily connected
  - occasionally done to represent specific structures
    - Semantic networks
    - Logical sentences
  - Makes learning rather difficult
- Layered structures
  - Networks are arranged into layers
  - Interconnections mostly between two layers
  - Some networks may have feedback connections

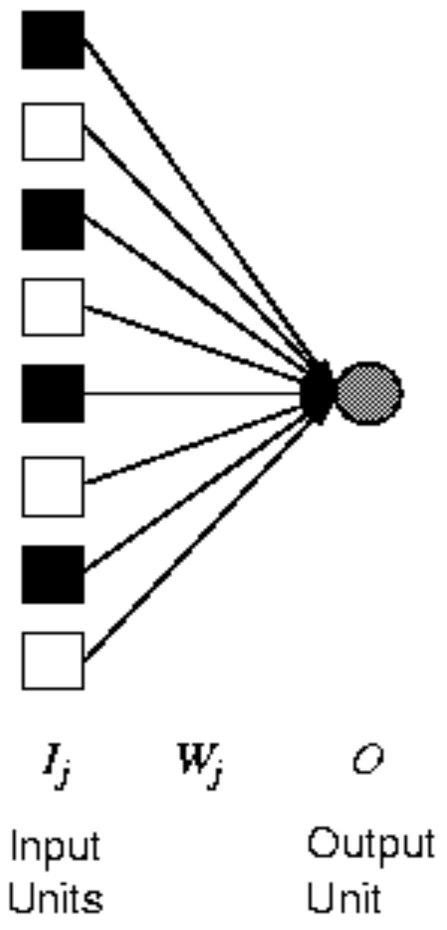
Network Structures

- Feed-forward
  - In a feed-forward network, links are unidirectional, and there are no cycles.
  - A feed-forward network is a directed acyclic graph.
  - Layered feed-forward network - each unit is linked only to units in the next layer; there are no links between units in the same layer, no links backward to a previous layer, and no links that skip a layer.
- Recurrent Network
  - In a recurrent network, the links can form arbitrary topologies.

Perceptrons



Perceptron Network

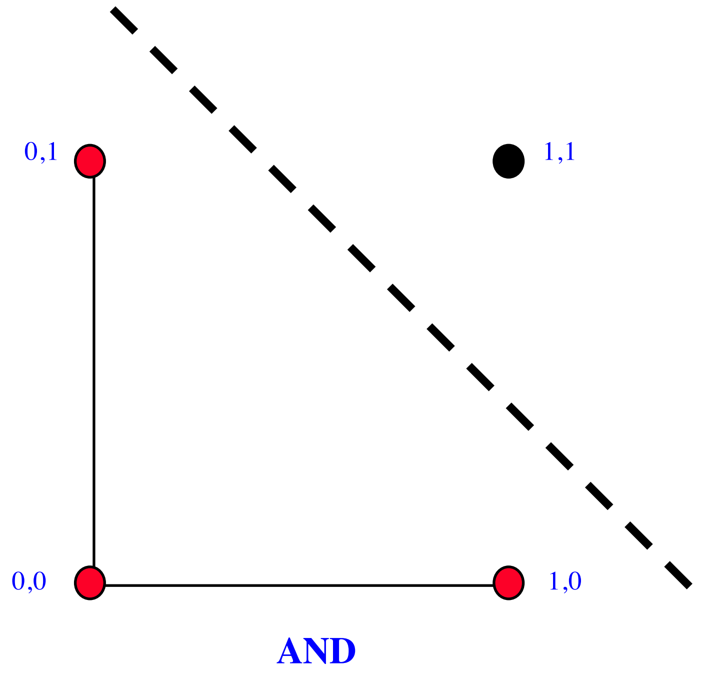


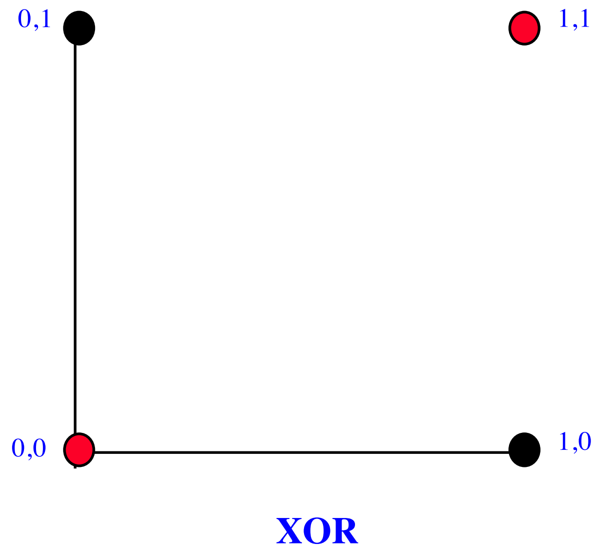
Single Perceptron

- Single layer, feed-forward network
- One of the first types of neural networks
  - Late 1950s
- Output is calculated as a step function applied weighted sum of inputs
- Capable of learning simple functions
  - Linearly separable.

Perceptrons and Linear Separability

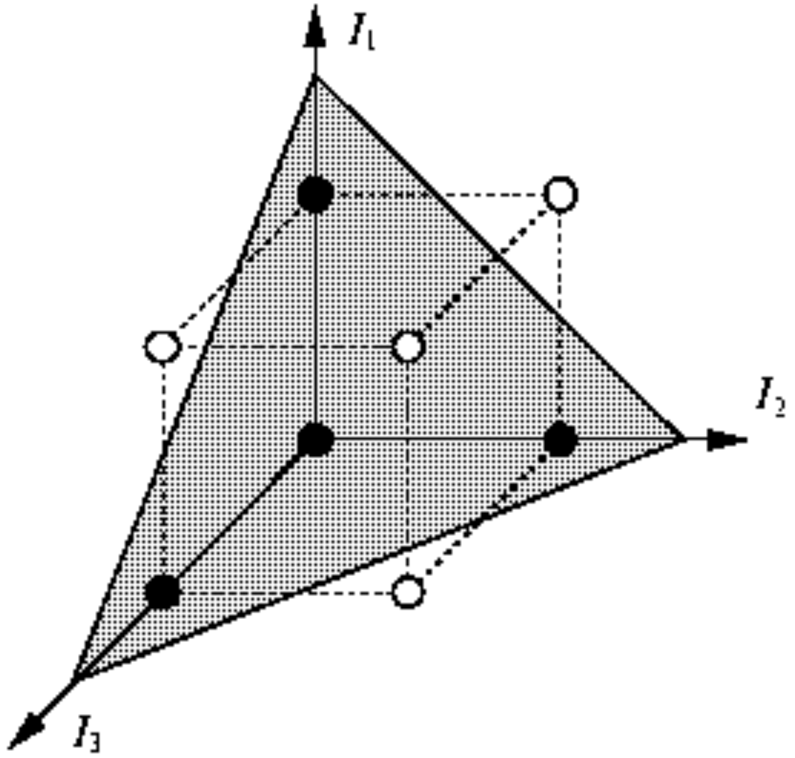
The fact that a perceptron can only represent linearly separable functions follows directly from the function computed by a perceptron. A perceptron outputs a 1 only if W • I > 0. This means that the entire input space is divided in two along a boundary defined by W • I = 0



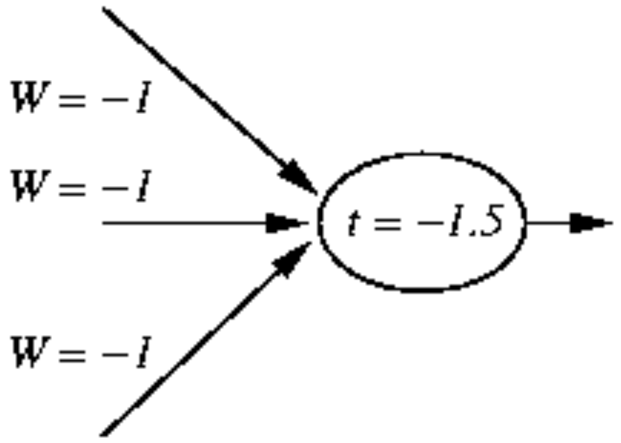


Perceptrons can deal with linearly separable functions; some simple functions are *not* linearly separable XOR function

Perceptrons and Linear Separability



Separating Plane



Weights and Threshold

Linear separability can be extended to more than two dimensions; more difficult to visualize

Perceptrons and Learning

- Perceptrons can learn from examples through a simple learning rule
  - Calculate the error of a unit Err_i as the difference between the correct output T_i and the calculated output O_i

    $Err_i = T_i - O_i$
  - Adjust the weight W_j of the input I_j such that the error decreases

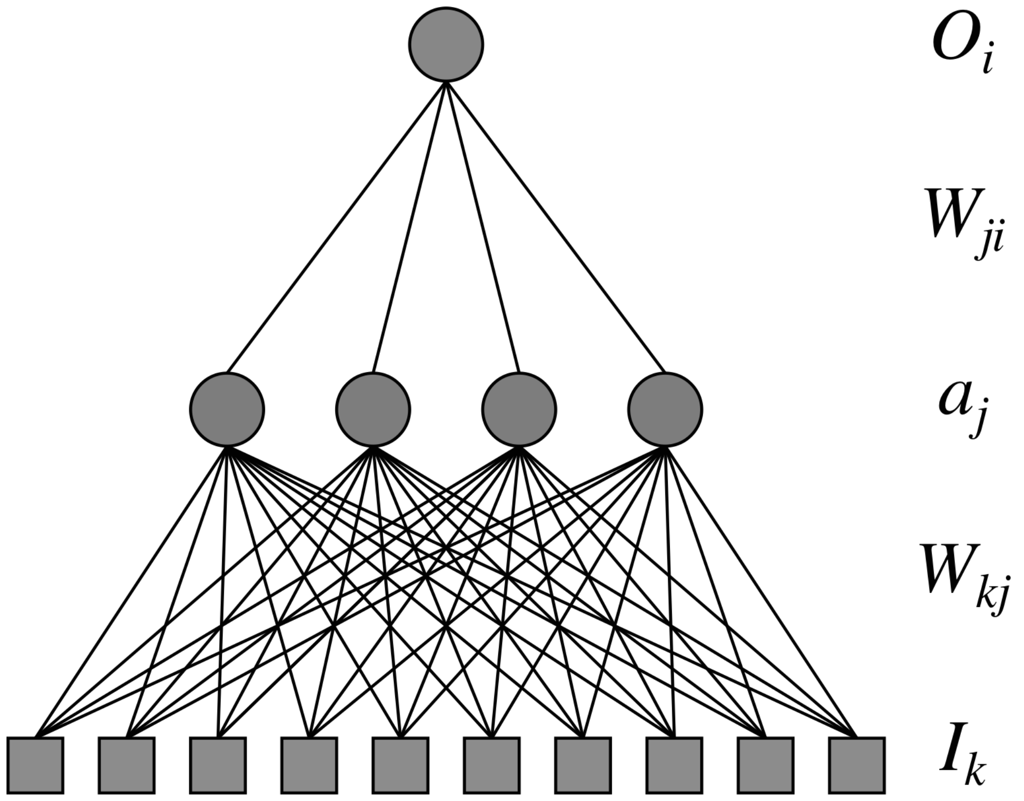
    $W_{ij} := W_{ij} + \alpha * I_{ij} * Err_{ij}$
    - α is the learning rate
    - This is a gradient descent search through the weight space
- Great enthusiasm in the late 50s and early 60s;
  - Minsky & Papert in 1969 analyzed the class of representable functions and found the linear separability problem

Multi-Layer Networks

Rosenblatt and others described multilayer feed-forward networks in the late 1950s, but concentrated their research on single-layer perceptrons. This was mainly because of the difficulty of finding a sensible way to update the weights between the inputs and the hidden units.

- Research in the more complex networks with more than one layer was very limited until the 1980s
  - Learning in such networks is much more complicated
  - The problem is to assign the blame for an error to the respective units and their weights in a constructive way.
- The back-propagation learning algorithm can be used to facilitate learning in multi-layer networks

Multi-Layer Network



- Two-layer network
  - Input units I_k
    - usually not counted as a separate layer
  - Hidden units a_j
  - Output units O_i
- Usually all nodes of one layer have **weighted connections** to all nodes of the next layer.

Back-Propagation Algorithm

Learning in such a network proceeds the same way as for perceptrons; If there is an error (a difference between the output and target), then the weights are adjusted to reduce this error. The trick is to assess the blame for an error and divide it among the contributing weights.

- Assigns blame to individual units in the respective layers
  - Essentially based on the connection strength
  - Proceeds from the output layer to the hidden layer(s)
  - Updates the weights of the units leading to the layer
- Essentially performs gradient-descent search on the error surface
  - Relatively simple since it relies only on local information from directly connected units
  - Has convergence and efficiency problems

Multi-Layer Network

In practice, time to convergence is highly variable, and a vast array of techniques have been developed to try to speed up the process using an assortment of tunable parameters.

- Expressiveness
  - Weaker than predicate logic
  - Good for continuous inputs and outputs
- Computational efficiency
  - training time can be exponential in the number of inputs
  - depends critically on parameters like the learning rate
  - local minima are problematic
    - can be overcome by simulated annealing, at additional cost
- Generalization
  - works reasonably well for some functions (classes of problems)
    - no formal characterization of these functions

Multi-Layer Network

- Sensitivity to noise
  - very tolerant
  - they perform nonlinear regression
- Transparency
  - neural networks are essentially black boxes
  - there is no explanation or trace for a particular answer
  - tools for the analysis of networks are very limited
  - some limited methods to extract rules from networks
- Prior Knowledge
  - very difficult to integrate since the internal representation of the networks is not easily accessible

Applications

- Domains and tasks where neural networks are successfully used
  - Handwriting recognition
  - Control problems
    - juggling, truck backup problem
  - Series prediction
    - weather, financial forecasting
  - Categorization
    - sorting of items (fruit, characters, phonemes, …)

Neural networks provide ability to provide more human-like AI. Takes rough approximation and hard-coded reactions out of AI design. Still require a lot of fine-tuning during development.