Bachelor of Computer Science (Hons) Year-2 Sep 2023

Welcome to Intelligent Systems

CAI3204N

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - CO3: Select and apply appropriate intelligent techniques to a given problem.
  - CO4: Critically discuss intelligent system research issues and their applications.

Artificial Neural Networks

Learning Objectives

- At the end of the course, students will be able to:
  - CO1: Identify the types of problem that are amenable to "intelligent" solutions.
  - CO2: Compare and contrast the various intelligent system techniques to solve such problems.
  - CO3: Select and apply appropriate intelligent techniques to a given problem.
  - CO4: Critically discuss intelligent system research issues and their applications.

Neural Networks Fundamentals – A Comprehensive Coverage

This topic gives an introduction to basic neural network architectures and learning rules

Historical Sketch

- Pre-1940: von Hemholtz, Mach, Pavlov, etc.
  - General theories of learning, vision, conditioning
  - No specific mathematical models of neuron operation
- 1940s: Hebb, McCulloch and Pitts
  - Mechanism for learning in biological neurons
  - Neural-like networks can compute any arithmetic function
- 1950s: Rosenblatt, Widrow and Hoff
  - First practical networks and learning rules
- 1960s: Minsky and Papert
  - Demonstrated limitations of existing neural networks, new learning algorithms are not forthcoming, some research suspended
- 1970s: Amari, Anderson, Fukushima, Grossberg, Kohonen
  - Progress continues, although at a slower pace
- 1980s: Grossberg, Hopfield, Kohonen, Rumelhart, etc.
  - Important new developments cause a resurgence in the field

Applications

- Aerospace
  - – High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors
- Automotive
  - – Automobile automatic guidance systems, warranty activity analyzers
- Banking
  - – Check and other document readers, credit application evaluators
- Defense
  - – Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification
- Electronics
  - – Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling

Applications

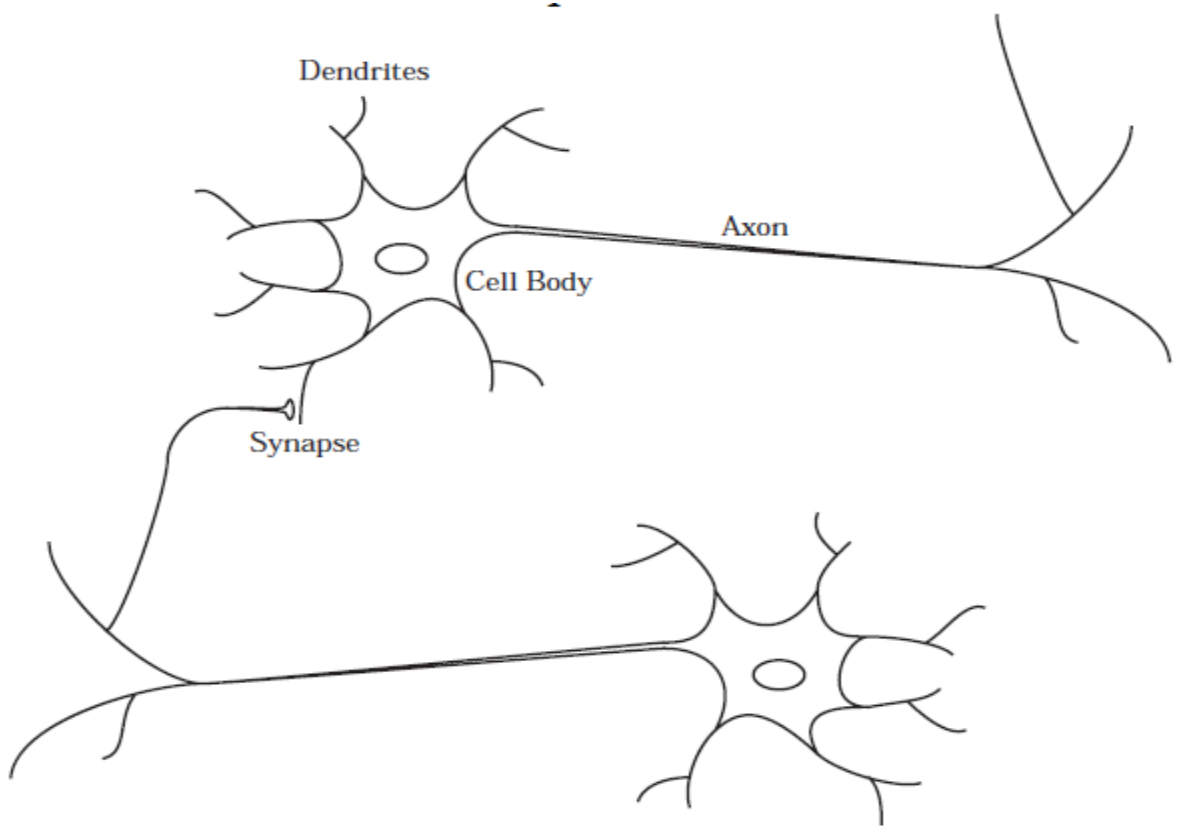
- Financial
  - – Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis, portfolio trading program, corporate financial analysis, currency price prediction
- Manufacturing
  - – Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations, chemical product design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process systems
- Medical
  - – Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency room test advisement

Applications

- Robotics
  - Trajectory control, forklift robot, manipulator controllers, vision systems
- Speech
  - Speech recognition, speech compression, vowel classification, text to speech synthesis
- Securities
  - Market analysis, automatic bond rating, stock trading advisory systems
- Telecommunications
  - Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems
- Transportation
  - Truck brake diagnosis systems, vehicle scheduling, routing systems

Biology

- Neurons respond slowly
  - 10^{-3} s compared to 10^{-9} s for electrical circuits
- The brain uses massively parallel computation
  - $\approx 10^{11}$ neurons in the brain
  - $\approx 10^{4}$ connections per neuron



What is a neural network (NN)?

- Neural networks is a branch of "Artificial Intelligence". **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.

A vague description is as follows:

- An ANN is a network of many very simple processors ("units"), each possibly having a (small amount of) local memory.
- The units are connected by unidirectional communication channels ("connections"), which carry numeric (as opposed to symbolic) data.
- The units operate only on their local data and on the inputs they receive via the connections.
- The design motivation is what distinguishes neural networks from other mathematical techniques
- A neural network is a processing device, either an algorithm, or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof.
- Most neural networks have some sort of "training" rule whereby the weights of connections are adjusted on the basis of presented patterns.
- In other words, neural networks "learn" from examples, just like children learn to recognize dogs from examples of dogs, and exhibit some structural capability for generalization.
- Neural networks normally have great potential for parallelism, since the computations of the components are independent of each other.

Introduction

- Biological neural networks are much more complicated in their elementary structures than the mathematical models we use for ANNs.
- It is an 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.
- **Neural networks are a powerful technique to solve many real world problems. They have the ability to learn from experience in order to improve their performance and to adapt themselves to changes in the environment. In addition to that they are able to deal with incomplete information or noisy data and can be very effective especially in situations where it is not possible to define the rules or steps that lead to the solution of a problem.**
- They typically consist of many simple processing units, which are wired together in a complex communication network.

Introduction

•In principle, NNs can compute any computable function, i.e. they can do everything a normal digital computer can do. Especially anything that can be represented as a mapping between vector spaces can be approximated to arbitrary precision by Neural Networks.

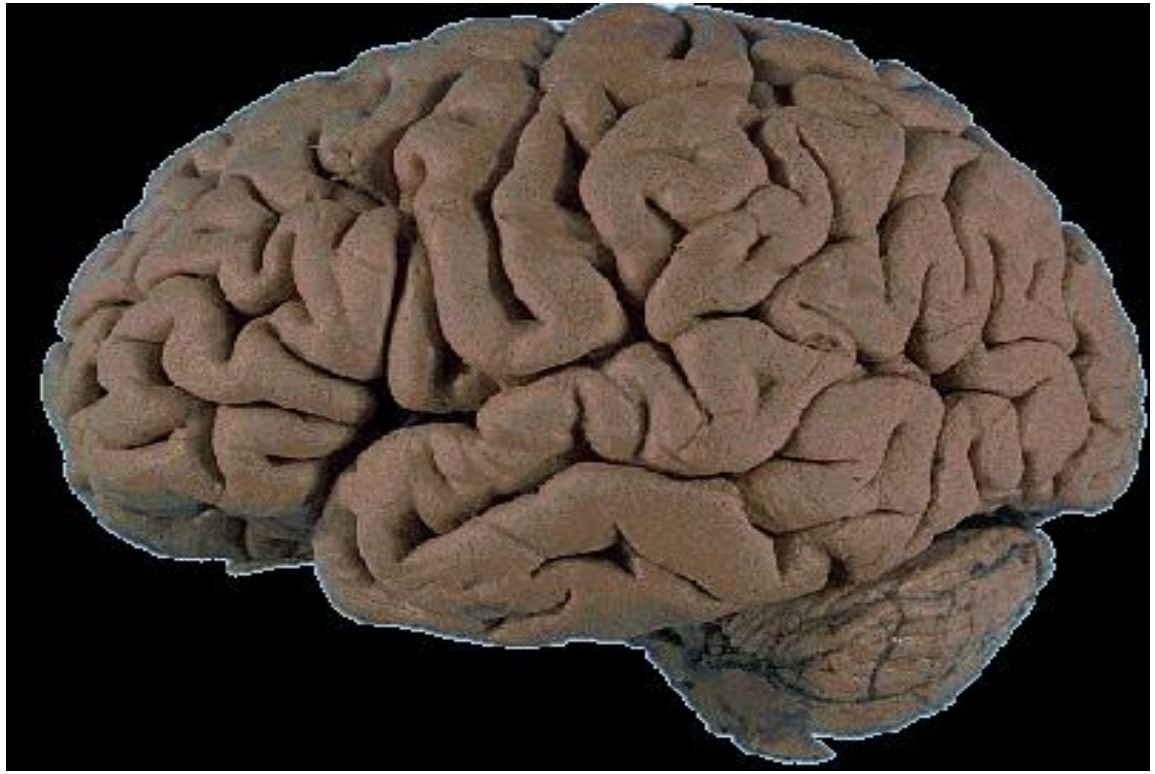
•In practice, NNs are especially useful for mapping problems which are tolerant of some errors and have lots of example data available, but to which hard and fast rules can not easily be applied.

•In a nutshell a Neural network can be considered as a black box that is able to predict an output pattern when it recognizes a given input pattern. Once trained, the neural network is able to recognize similarities when presented with a new input pattern, resulting in a predicted output pattern.

The Brain

The Brain as an Information Processing System

The human brain contains about 10 billion nerve cells, or **neurons**. On average, each neuron is connected to other neurons through about 10 000 **synapses**. (The actual figures vary greatly, depending on the local neuroanatomy.)

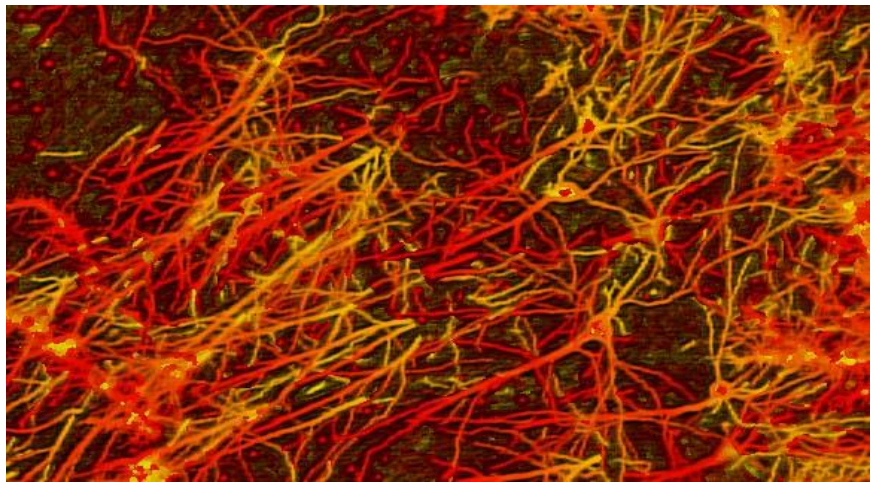


Computation in the brain

- The brain's network of neurons forms a massively parallel information processing system. This contrasts with conventional computers, in which a single processor executes a single series of instructions.
- Against this, consider the time taken for each elementary operation: neurons typically operate at a maximum rate of about 100 Hz, while a conventional CPU carries out several hundred million machine level operations per second. Despite of being built with very slow hardware, the brain has quite remarkable capabilities:
- Its performance tends to degrade gracefully under partial damage. In contrast, most programs and engineered systems are brittle: if you remove some arbitrary parts, very likely the whole will cease to function.
- It can learn (reorganize itself) from experience.
- This means that partial recovery from damage is possible if healthy units can learn to take over the functions previously carried out by the damaged areas.
- It performs massively parallel computations extremely efficiently. For example, complex visual perception occurs within less than 100 ms, that is, 10 processing steps!
- It supports our intelligence and self-awareness. (Nobody knows yet how this occurs.)
- As a discipline of Artificial Intelligence, Neural Networks attempt to bring computers a little closer to the brain's capabilities by imitating certain aspects of information processing in the brain, in a highly simplified way.

Neural Networks in the Brain

- The brain is not homogeneous. At the largest anatomical scale, we distinguish cortex, midbrain, brainstem, and cerebellum. Each of these can be hierarchically subdivided into many regions, and areas within each region, either according to the anatomical structure of the neural networks within it, or according to the function performed by them.



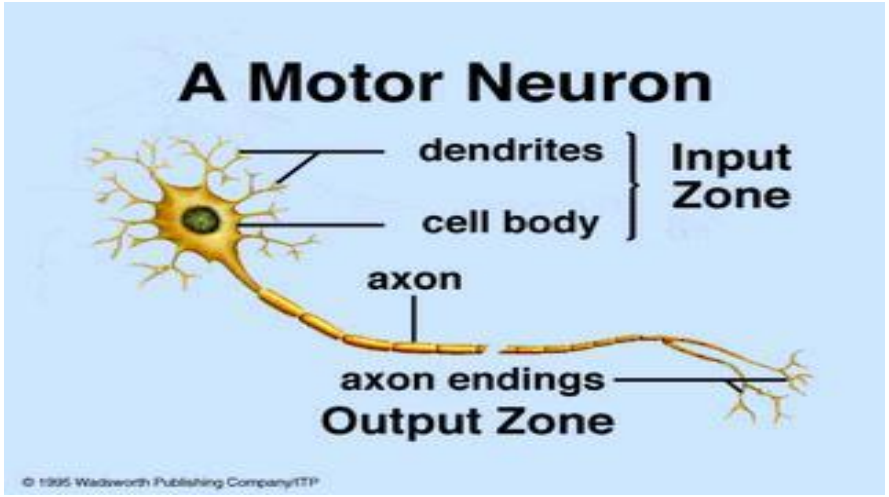
•The overall pattern of projections (bundles of neural connections) between areas is extremely complex, and only partially known. The best mapped (and largest) system in the human brain is the visual system, where the first 10 or 11 processing stages have been identified. We distinguish feedforward projections that go from earlier processing stages (near the sensory input) to later ones (near the motor output), from feedback connections that go in the opposite direction.

•In addition to these long-range connections, neurons also link up with many thousands of their neighbours. In this way they form very dense, complex local networks

Neurons and Synapses

•The basic computational unit in the nervous system is the nerve cell, or neuron. A neuron has:

- •Dendrites (inputs)
- •Cell body
- •Axon (output)



•A neuron receives input from other neurons (typically many thousands). Inputs sum (approximately). Once input exceeds a critical level, the neuron discharges a spike - an electrical pulse that travels from the body, down the axon, to the next neuron(s) (or other receptors). This spiking event is also called depolarization, and is followed by a refractory period, during which the neuron is unable to fire.

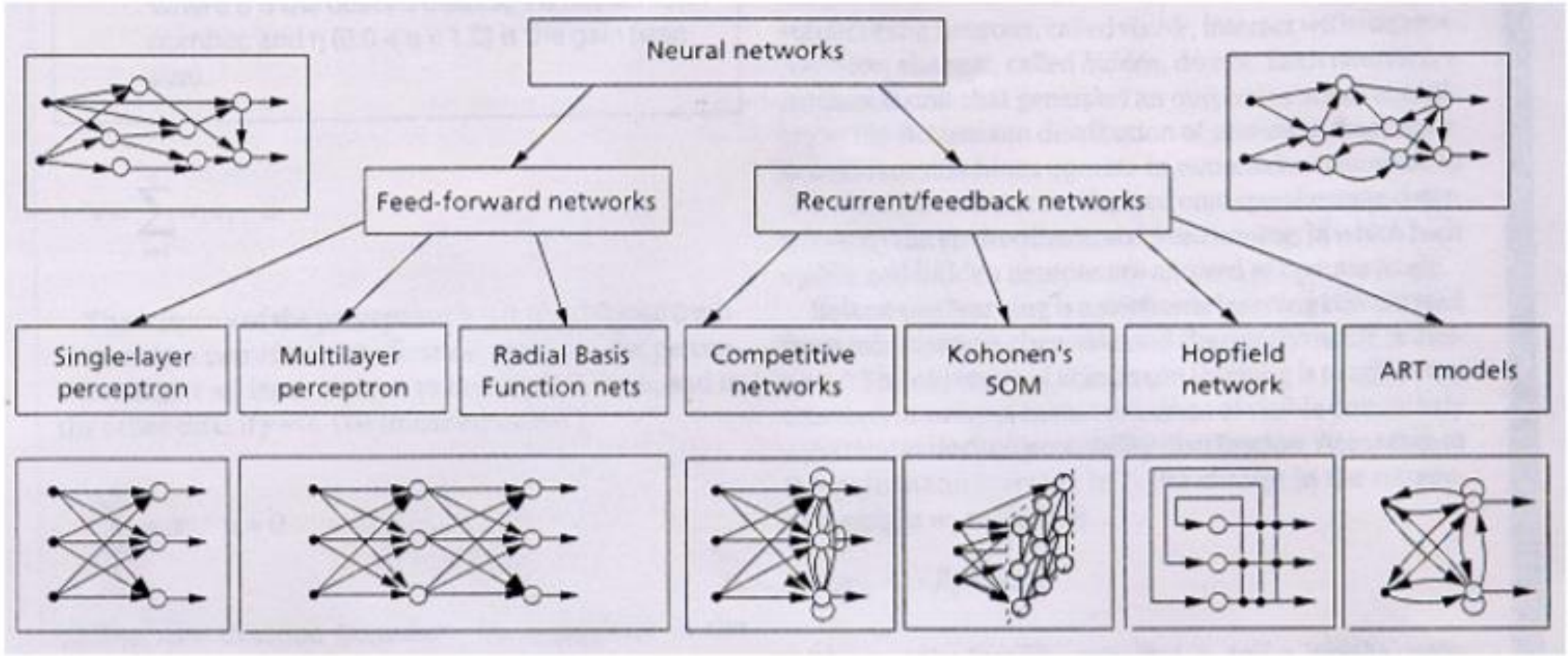
•The axon endings (Output Zone) almost touch the dendrites or cell body of the next neuron. Transmission of an electrical signal from one neuron to the next is effected by neurotransmittors, chemicals which are released from the first neuron and which bind to receptors in the second. This link is called a synapse. The extent to which the signal from one neuron is passed on to the next depends on many factors, e.g. the amount of neurotransmittor available, the number and arrangement of receptors, amount of neurotransmittor reabsorbed, etc.

Artificial Neuron Models

- Computational neurobiologists have constructed very elaborate computer models of neurons in order to run detailed simulations of particular circuits in the brain. As Computer Scientists, we are more interested in the general properties of neural networks, independent of how they are actually "implemented" in the brain. This means that we can use much simpler, abstract "neurons", which (hopefully) capture the essence of neural computation even if they leave out much of the details of how biological neurons work.

- People have implemented model neurons in hardware as electronic circuits, often integrated on VLSI chips. Remember though that computers run much faster than brains - we can therefore run fairly large networks of simple model neurons as software simulations in reasonable time. This has obvious advantages over having to use special "neural" computer hardware.

TOPOLOGY



A Simple Artificial Neuron

•The basic computational element (model neuron) is often called a node or unit. It receives input from some other units, or perhaps from an external source. Each input has an associated weight w, which can be modified so as to model synaptic learning. The unit computes some function f of the weighted sum of its inputs

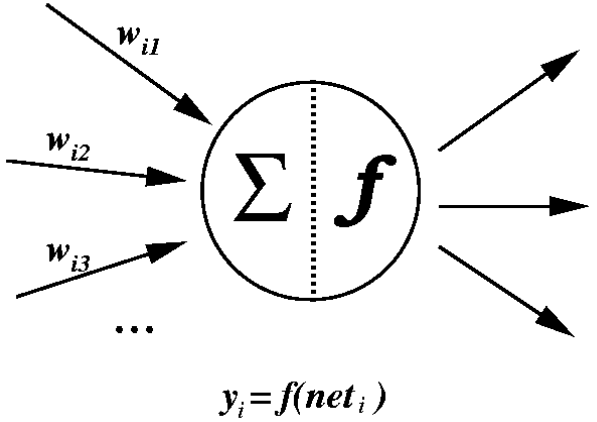
$$y_i = f\left(\sum_j w_{ij}\, y_j\right)$$

•Its output, in turn, can serve as input to other units.

•The weighted sum is called the net input to unit i, often written neti.

•Note that wij refers to the weight from unit j to unit i (not the other way around).

•The function f is the unit's activation function. In the simplest case, f is the identity function, and the unit's output is just its net input. This is called a linear unit.



Applications:

Neural Network Applications can be grouped in following categories:

- **Clustering: (NEXT WEEK)**
  A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.

- **Classification/Pattern recognition: (NEXT WEEK)**
  The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.

- **Function approximation:**
  The tasks of function approximation is to find an estimate of the unknown function f() subject to noise. Various engineering and scientific disciplines require function approximation.

- **Prediction/Dynamical Systems:**
  The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from Function approximation by considering time factor. Here the system is dynamic and may produce different results for the same input data based on system state (time).

Types of Neural Networks

Neural Network types can be classified based on following attributes:

- **Applications**
  - -Classification
  - -Clustering
  - -Function approximation
  - -Prediction
- **Connection Type**
  - - Static (feedforward)
  - - Dynamic (feedback)
- **Topology**
  - - Single layer
  - - Multilayer
  - - Recurrent
  - - Self-organized

- **Learning Methods**
  - - Supervised
  - - Unsupervised

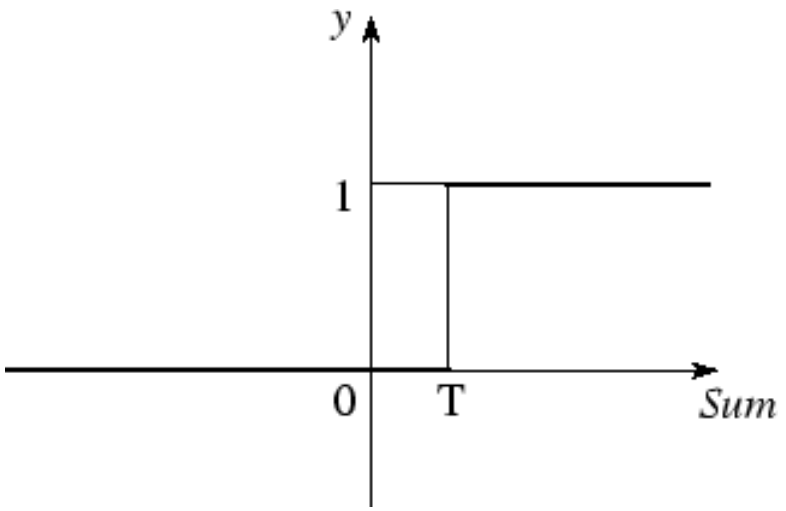
The McCulloch-Pitts Model of Neuron

The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs ***I1,I2,I3…Im*** and one output ***y*** . The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output y is binary. Such a function can be described mathematically using these equations:

$$Sum = \sum_{i=1}^{N} I_i W_i,$$

$$y = f(Sum).$$

W1,W2…Wm are weight values normalized in the range of either (0,1) or (-1,1) and associated with each input line, ***Sum*** is the weighted sum, and ***T*** is a threshold constant. The function ***f*** is a linear step function at threshold ***T*** as shown in figure



The Perceptron

In late 1950s, Frank Rosenblatt introduced a network composed of the units that were enhanced version of McCulloch-Pitts Threshold Logic Unit (TLU) model. Rosenblatt's model of neuron, a *perceptron*, was the result of merger between two concepts from the 1940s, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights. In addition to the variable weight values, the perceptron model added an extra input that represents *bias*. Thus, the modified equation is now as follows:

$$Sum = \sum_{i=1}^{N} I_i W_i + b,$$

where ***b*** represents the bias value.

The McCulloch-Pitts Model of Neuron

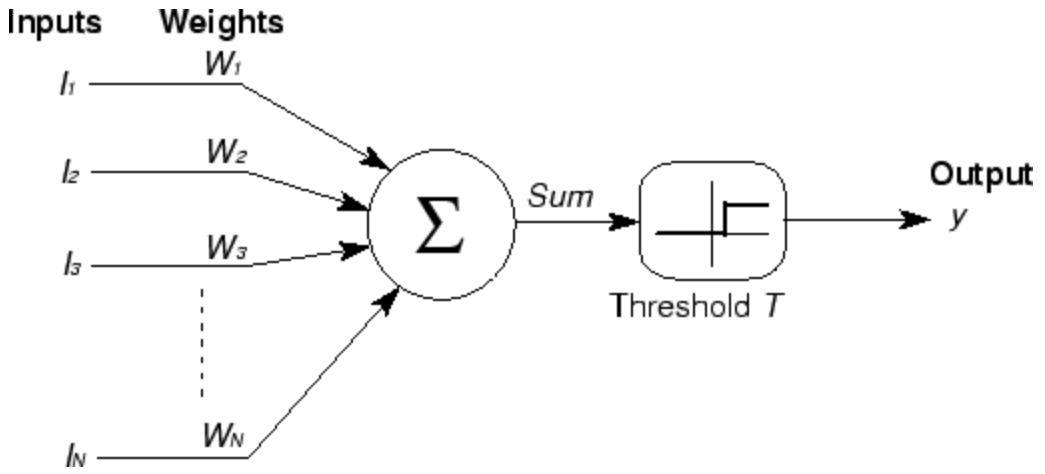


Figure :Symbolic Illustration of Linear Threshold Gate

- The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse features for various applications . Thus, we need to obtain the neural model with more flexible computational features.

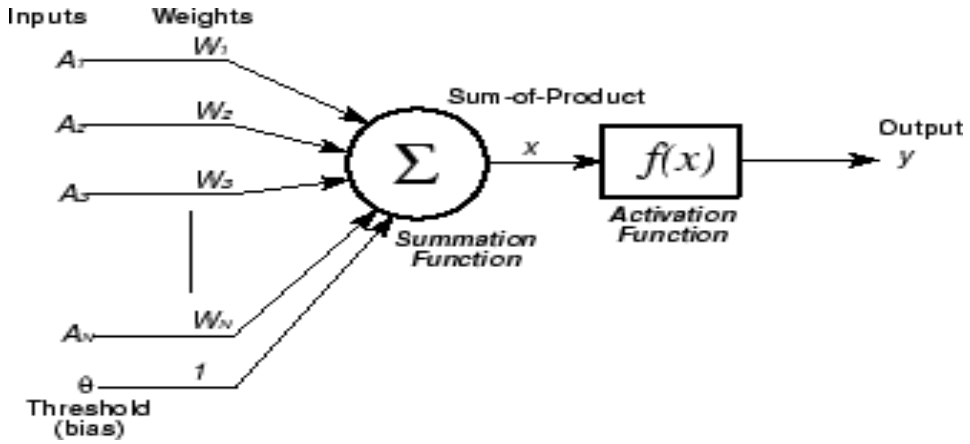
Artificial Neuron with Continuous Characteristics

- Based on the McCulloch-Pitts model described previously, the general form an artificial neuron can be described in two stages shown in figure. In the first stage, the *linear combination* of inputs is calculated. Each value of input array is associated with its weight value, which is normally between 0 and 1. Also, the summation function often takes an extra input value ***Theta*** with weight value of 1 to represent threshold or *bias* of a neuron. The summation function will be then performed as

$$x = \sum_{i=1}^{N} A_i W_i + \theta.$$

•The sum-of-product value is then passed into the second stage to perform the activation function which generates the output from the neuron. The activation function ``squashes'' the amplitude the output in the range of [0,1] or [-1,1] alternately. The behavior of the activation function will describe the characteristics of an artificial neuron model.

Artificial Neuron with Continuous Characteristics



•The signals generated by actual biological neurons are the action-potential spikes, and the biological neurons are sending the signal in *patterns* of spikes rather than simple absence or presence of single spike pulse. For example, the signal could be a continuous stream of pulses with various frequencies. With this kind of observation, we should consider a signal to be continuous with bounded range. The linear threshold function should be ``softened".

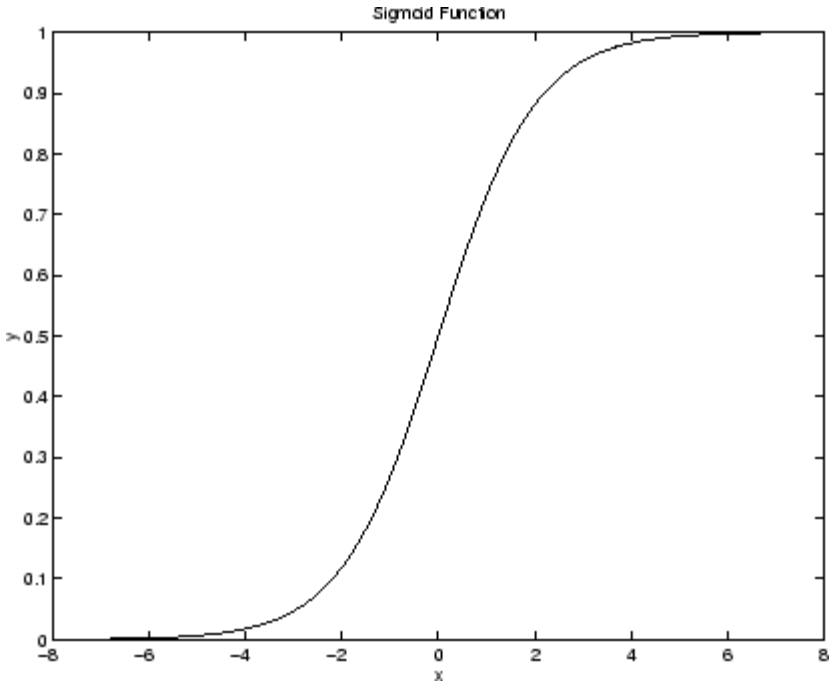
•One convenient form of such ``semi-linear" function is the *logistic sigmoid* function, or in short, *sigmoid* function as shown in figure. As the input ***x*** tends to large positive value, the output value ***y*** approaches to 1. Similarly, the output gets close to 0 as **x** goes negative. However, the output value is neither close to 0 nor 1 near the threshold point.

$$y = \frac{1}{1 + exp(-x)}.$$

- This function is expressed mathematically as follows:

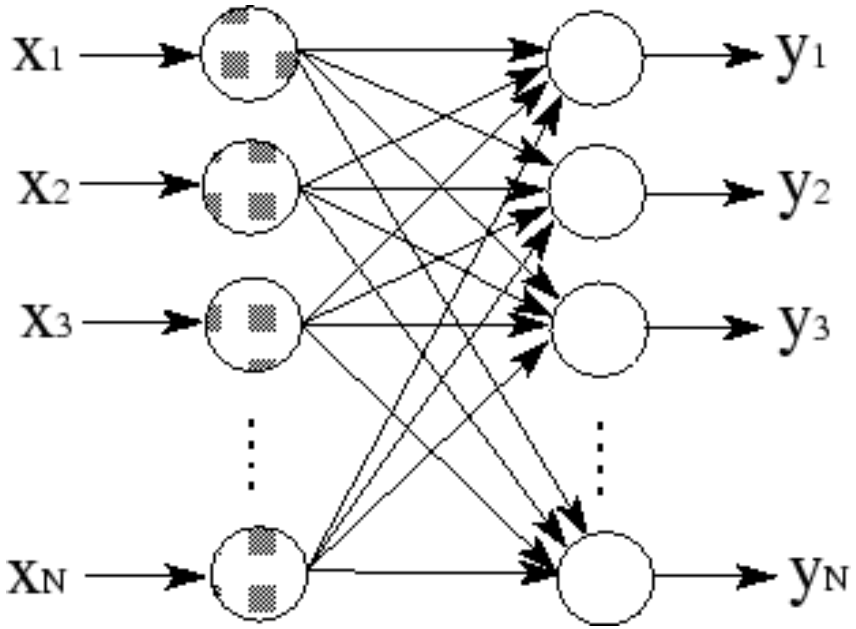
- Additionally, the sigmoid function describes the ``closeness'' to the threshold point by the slope. As ***x*** approaches to

  - *infinity* or **+ *infinity*** , the slope is zero; the slope increases as ***x*** approaches to 0. This characteristic often plays an important role in learning of neural networks.



Single-Layer Network

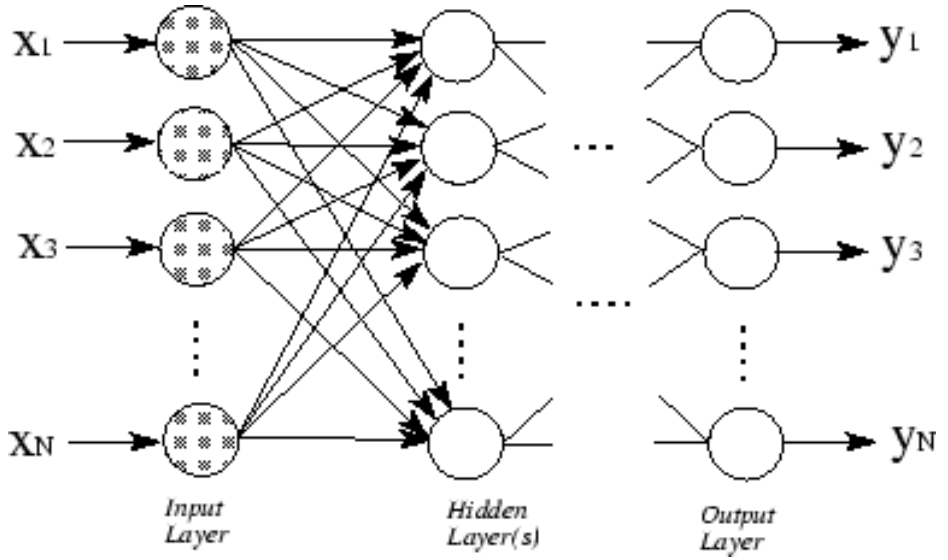
- By connecting multiple neurons, the true computing power of the neural networks comes, though even a single neuron can perform substantial level of computation.
- The most common structure of connecting neurons into a network is by layers. The simplest form of layered network is shown in figure. The shaded nodes on the left are in the so-called *input layer*. The input layer neurons are to only pass and distribute the inputs and perform no computation.
- Thus, the only true layer of neurons is the one on the right. Each of the inputs ***x1,x2,…xN*** is connected to every artificial neuron in the output layer through the connection weight. Since every value of outputs ***y1,y2,…yN*** is calculated from the same set of input values, each output is varied based on the connection weights.
- Although the presented network is *fully connected*, the true biological neural network may not have all possible connections - the weight value of zero can be represented as ``no connection".



Multilayer Network

- To achieve higher level of computational capabilities, a more complex structure of neural network is required. Figure shows the *multilayer neural network* which distinguishes itself from the single-layer network by having one or more *hidden layers*. In this multilayer structure, the input nodes pass the information to the units in the first hidden layer, then the outputs from the first hidden layer are passed to the next layer, and so on.

- Multilayer network can be also viewed as cascading of groups of single-layer networks. The level of complexity in computing can be seen by the fact that many single-layer networks are combined into this multilayer network. The designer of an artificial neural network should consider how many hidden layers are required, depending on complexity in desired computation.

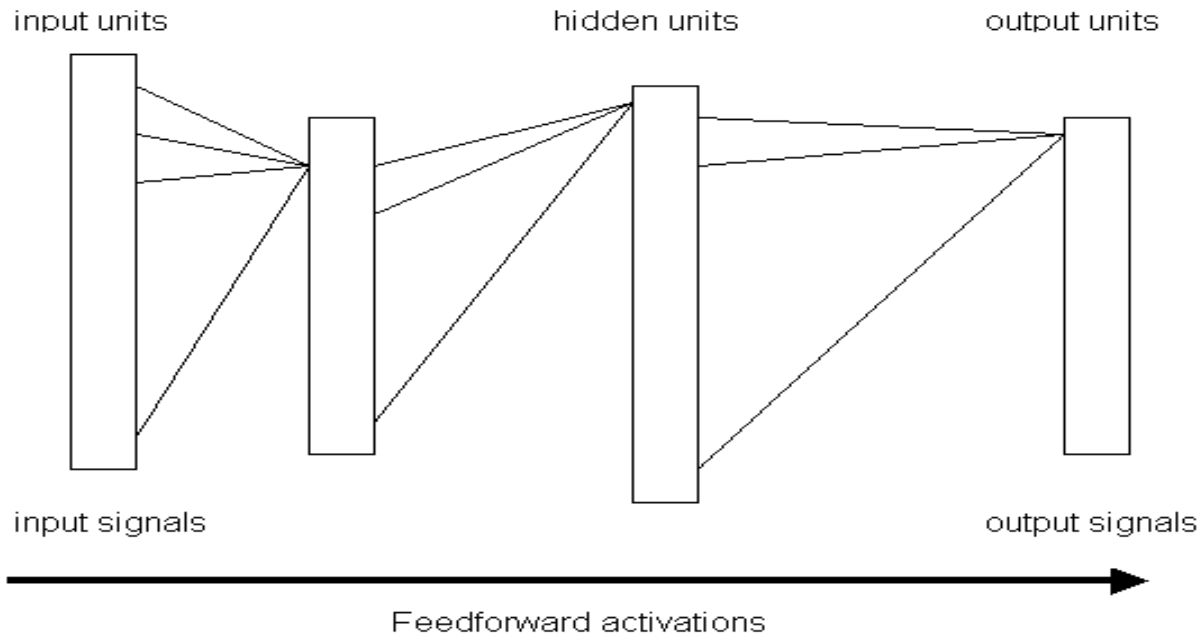


Backpropagation Networks

- Backpropagation networks, and multi layered perceptrons, in general, are *feedforward networks* with distinct input, output, and hidden layers. The units function basically like perceptrons, except that the transition (output) rule and the weight update (learning) mechanism are more complex.

- The figure on next page presents the architecture of backpropagation networks. There may be any number of hidden layers, and any number of hidden units in any given hidden layer. Input and output units can be binary {0, 1}, bi-polar {-1, +1}, or may have real values within a specific range such as [-1, 1].
- Note that units within the same layer are not interconnected.

Backpropagation Networks



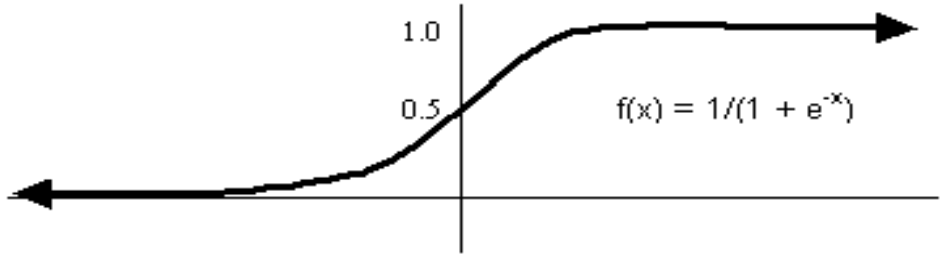
Architecture of backpropagation Networks. There may be any number of hidden layers, and any number of hidden units in any given hidden layer. Only some of the connections are shown. In general, every unit in a given layer is connected to every other unit in the preceding layer. There are no interconnections among units in the same layer. Activation of the network proceeds from the input units, layer by layer, up to the output units. Input and output units can be binary (0 or 1), bi-polar (-1 or +1), or may have real values within a specific range such as [-1, 1].

Backpropagation Networks

- In feedforward activation, units of hidden layer 1 compute their activation and output values and pass these on to the next layer, and so on until the output units will have produced the network's actual response to the current input. The activation value ak of unit k is computed as follows.
- This is basically the same activation function of *linear threshold units* (McCulloch and Pitts model).
- As illustrated above, xi is the input signal coming from unit i at the other end of the incoming connection. wki is the weight of the connection between unit k and unit i. Unlike in the linear threshold unit, the output of a unit in a backpropagation network is no longer based on a threshold. The output yk of unit k is computed as follows:
- The function f(x) is referred to as the output function. It is a continuously increasing function of the *sigmoid* type, asymptotically approaching 0 as x decreases, and asymptotically approaches 1 as x increases. At x = 0, f(x) is equal to 0.5.

$$a_k = \sum_{i=1,n} W_{ik} \cdot X_i$$

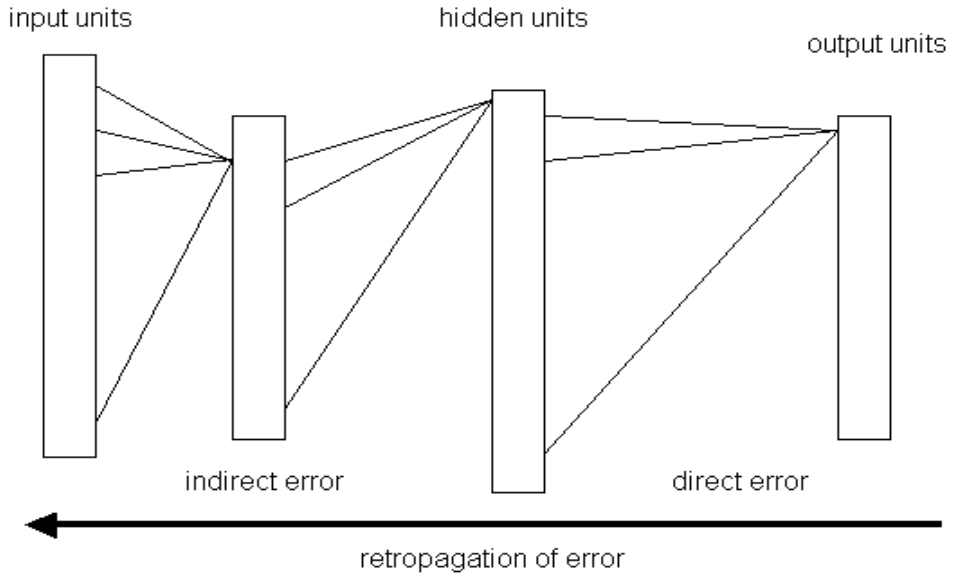
$$y_k = f(a_k) \text{ and } f(x) = 1/(1 + e^{-x})$$



The output function f(x) is a continuously increasing function of the "sigmoid" type, asymptotically approaching 0 as x decreases, and asymptotically approaches 1 as x increases. At x equal to 0, f(x) is equal to 0.5.

Backpropagation Networks- Errors

- Essentially, the error at the output layer is used to compute for the error at the hidden layer immediately preceding the output layer. Once this is computed, this is used in turn to compute for the error of the next hidden layer immediately preceding the last hidden layer. This is done sequentially until the error at the very first hidden layer is computed. The retropropagation of error is illustrated in the figure below:



The error at the output layer is used to compute for the error at the last hidden layer. Error at this layer is used to compute the error at the second to the last hidden layer. This is done in sequence until the error of the very first hidden layer is computed.

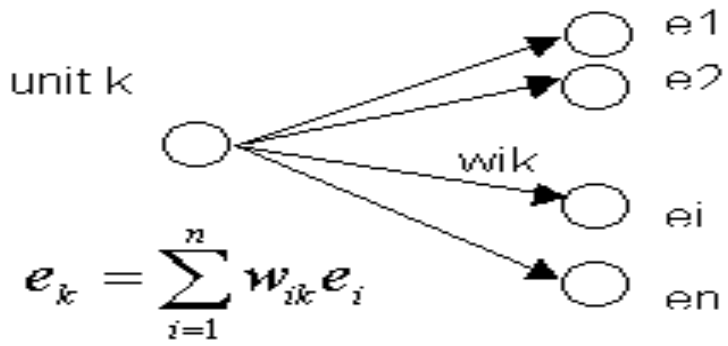
Backpropagation Networks

•Computation of errors ei at a hidden layer is done as follows:

$$e_h = \sum_{i=1,n} W_{ih} \cdot e_i$$

•The errors at the other end of the outgoing connections of the hidden unit h have been earlier computed. These could be error values at the output layer or at a hidden layer. These error signals are multiplied by their corresponding outgoing connection weights and the sum of these is taken.

Backpropagation Networks



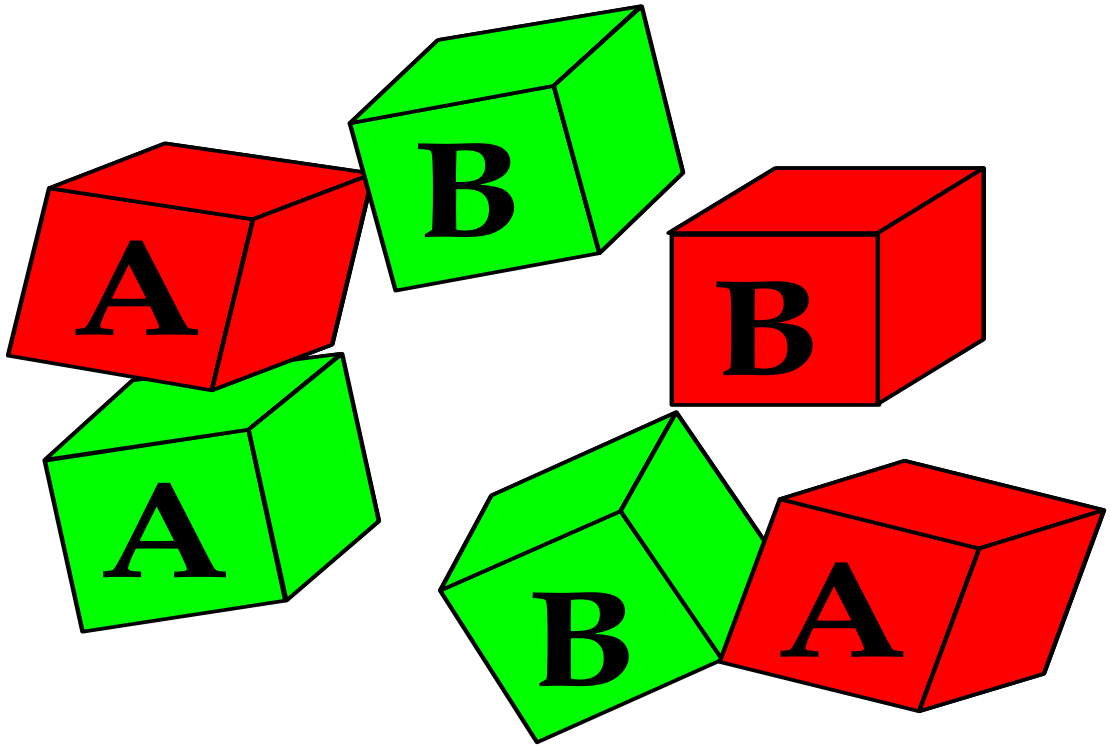
The errors at the other end of the outgoing connections of the hidden unit h have been earlier computed. These could be error values at the output layer or at a hidden layer. These error signals are multiplied by their corresponding outgoing connection weights and their sum is computed.

- The errors at the other end of the outgoing connections of the hidden unit h have been earlier computed. These could be error values at the output layer or at a hidden layer. These error signals are multiplied by their corresponding outgoing connection weights and the sum of these is taken.

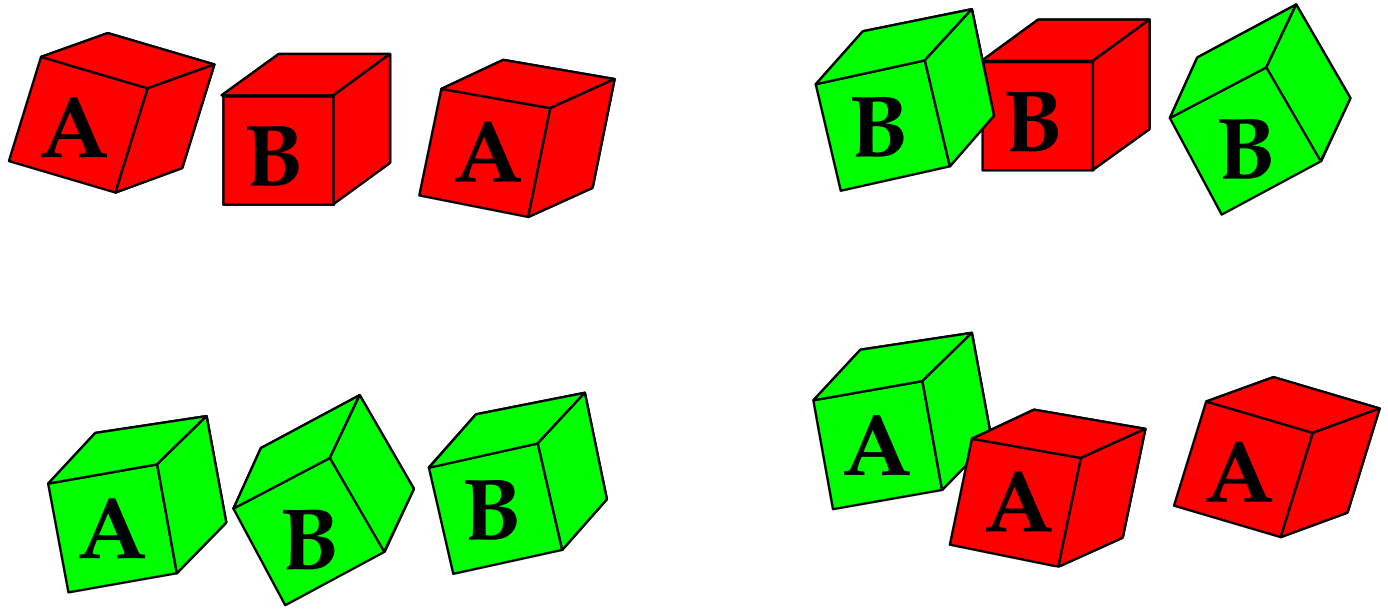
Learning Process

- One of the most important aspects of Neural Network is the learning process. The learning process of a Neural Network can be viewed as reshaping a sheet of metal, which represents the output (range) of the function being mapped.
- The training set (domain) acts as energy required to bend the sheet of metal such that it passes through predefined points. However, the metal, by its nature, will resist such reshaping. So the network will attempt to find a low energy configuration (i.e. a flat/non-wrinkled shape) that satisfies the constraints (training data).
- Learning can be done in **supervised** or **unsupervised** training.
- In supervised training, both the inputs and the outputs are provided.
  The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then calculated, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked.

Understanding Supervised and Unsupervised Learning

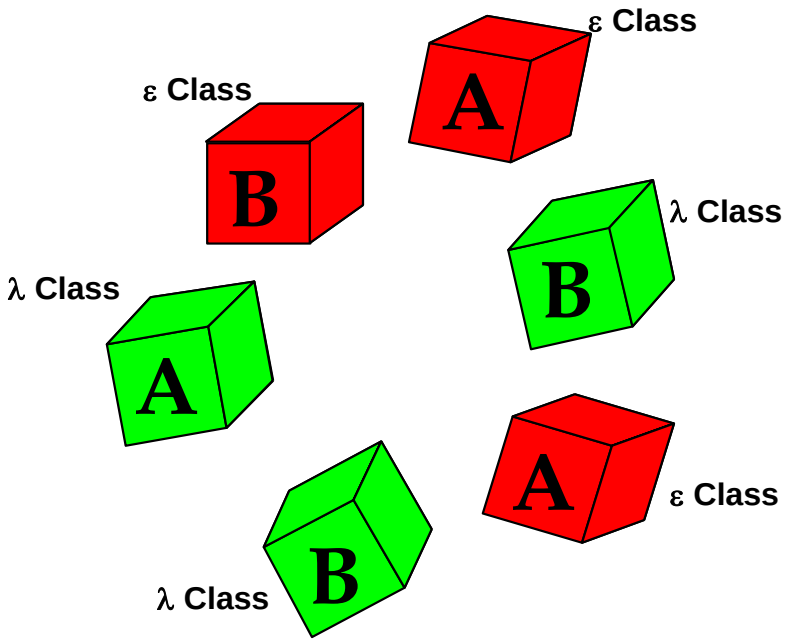


Two possible Solutions…



Supervised Learning

- It is based on a labeled training set.
- The class of each piece of data in training set is known.
- Class labels are pre-determined and provided in the training phase.



Supervised Vs Unsupervised

- Task performed

  Classification

  Pattern Recognition

- NN model :

  Preceptron

  Feed-forward NN

"What is the class of this data point?"

- Task performed

  Clustering

- NN Model :

  Self Organizing Maps

"What groupings exist in this data?"

"How is each data point related to the data set as a whole?"

Unsupervised Learning

- Input : set of patterns P, from n-dimensional space S, but little/no information about their classification, evaluation, interesting features, etc.

  It must learn these by itself! :)

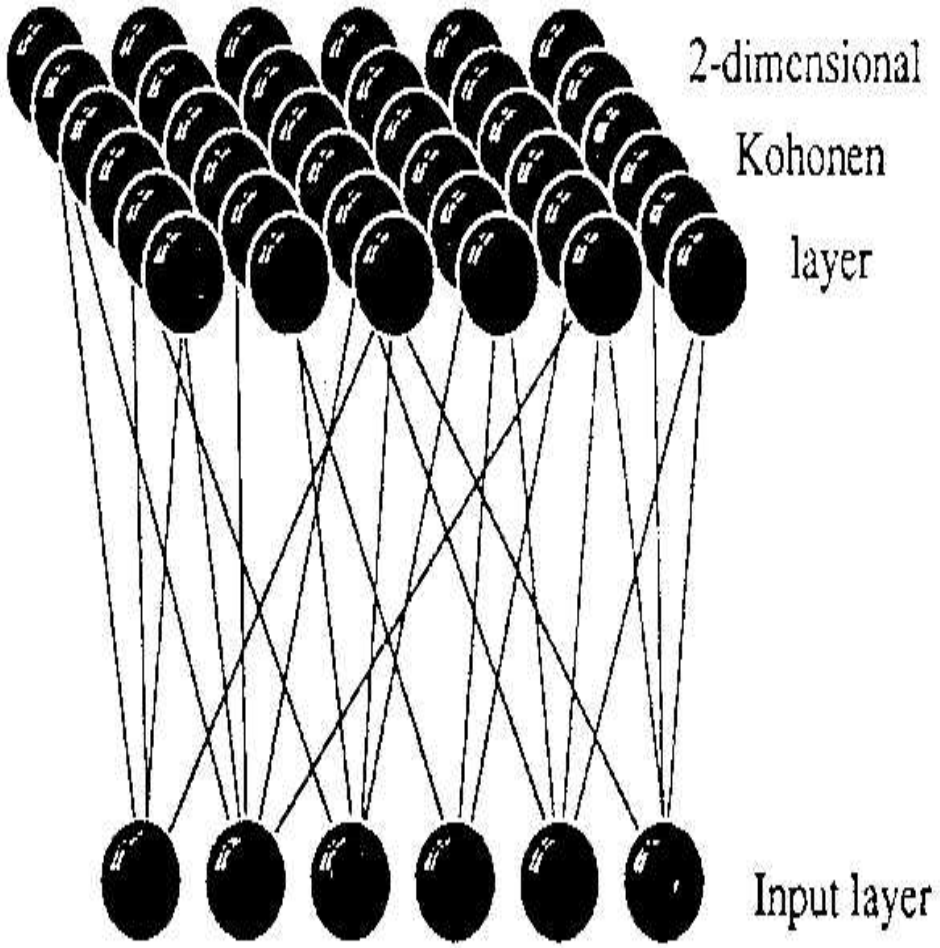
- Tasks:
  - Clustering - Group patterns based on similarity
  - Vector Quantization - Fully divide up S into a small set of regions (defined by codebook vectors) that also helps cluster P.
  - Feature Extraction - Reduce dimensionality of S by removing unimportant features (i.e. those that do not help in clustering P)

Unsupervised Neural Networks –Kohonen Learning

- Also defined – Self Organizing Map
- Learn a categorization of input space
- Neurons are connected into a 1-D or 2-D lattice.
- Each neuron represents a point in N-dimensional pattern space, defined by N weights
- During training, the neurons move around to try and fit to the data
- Changing the position of one neuron in data space influences the positions of its neighbors via the lattice connections

Self Organizing Map – Network Structure

- All inputs are connected by weights to each neuron size of neighbourhood changes as net learns.
- Aim is to map similar inputs (sets of values) to similar neuron positions.
- Data is clustered because it is mapped to the same node or group of nodes



SOM-Algorithm

1. Initialization :Weights are set to unique random values
2. Sampling : Draw an input sample x and present in to network
3. Similarity Matching : The winning neuron i is the neuron with the weight vector that best matches the input vector

$$i = \text{argmin}(j)\{ x - w_j \}$$

SOM - Algorithm

4. Updating : Adjust the weights of the winning neuron so that they better match the input.

   Also adjust the weights of the neighbouring neurons.

$$\Delta w_j = \eta \, . \, h_{ij} \, (x - w_j)$$

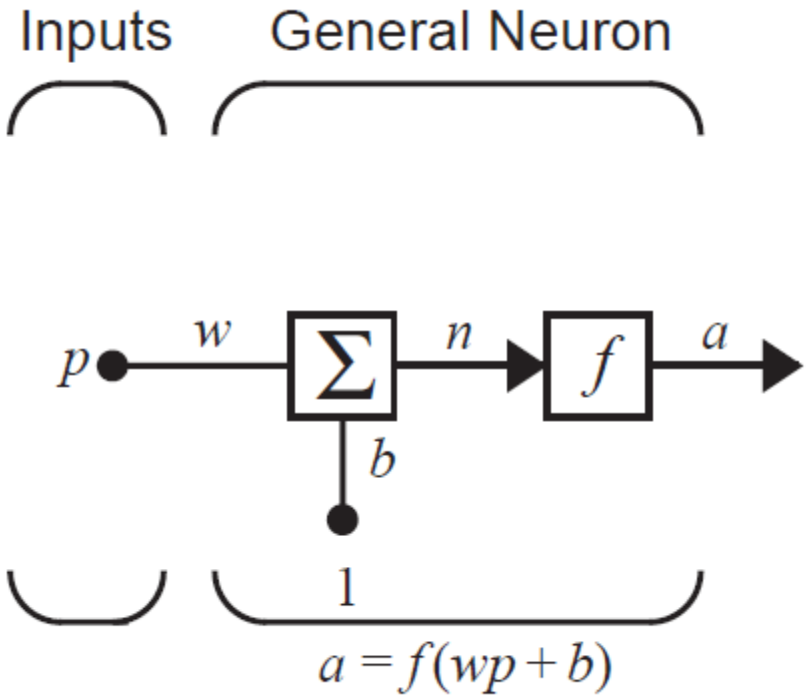
   neighbourhood function : h_{ij}

- over time neigbourhood function gets smaller

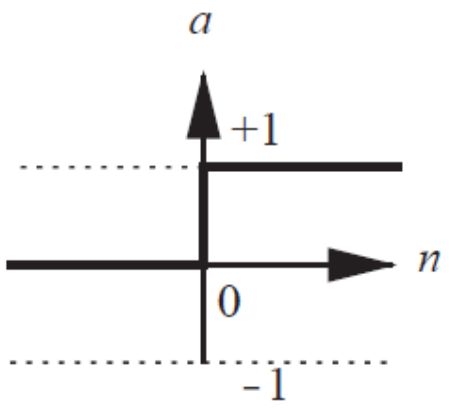
Result: The neurons provide a good approximation of the input space and correspond

Neuron Model and Network Architectures

Single –Input Neuron

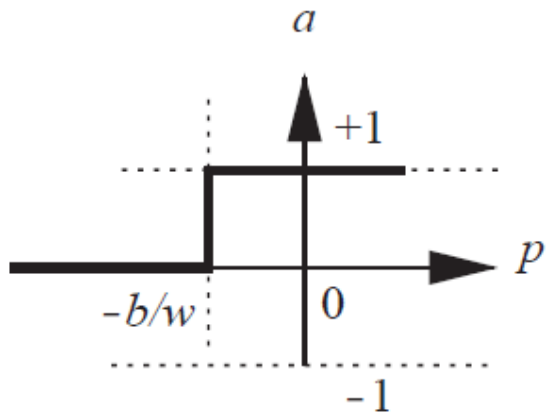


Transfer Functions



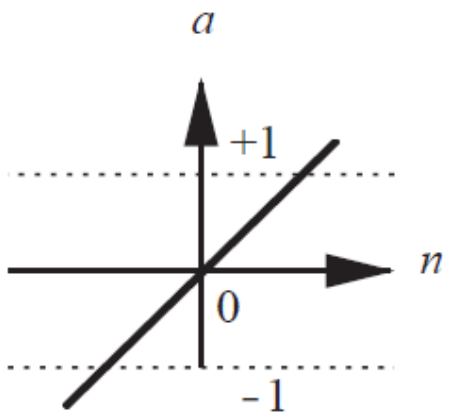
$a = hardlim(n)$

Hard Limit Transfer Function



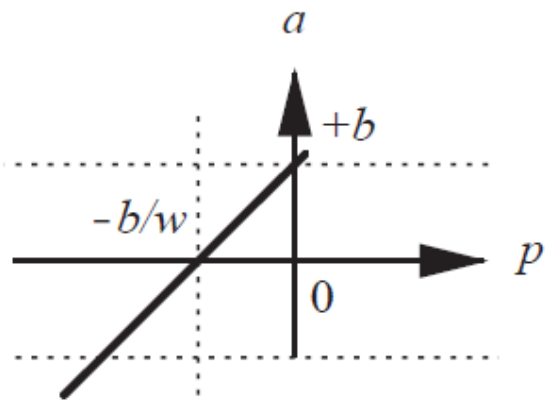
$a = hardlim(wp + b)$

Single-Input *hardlim* Neuron



$a = purelin(n)$

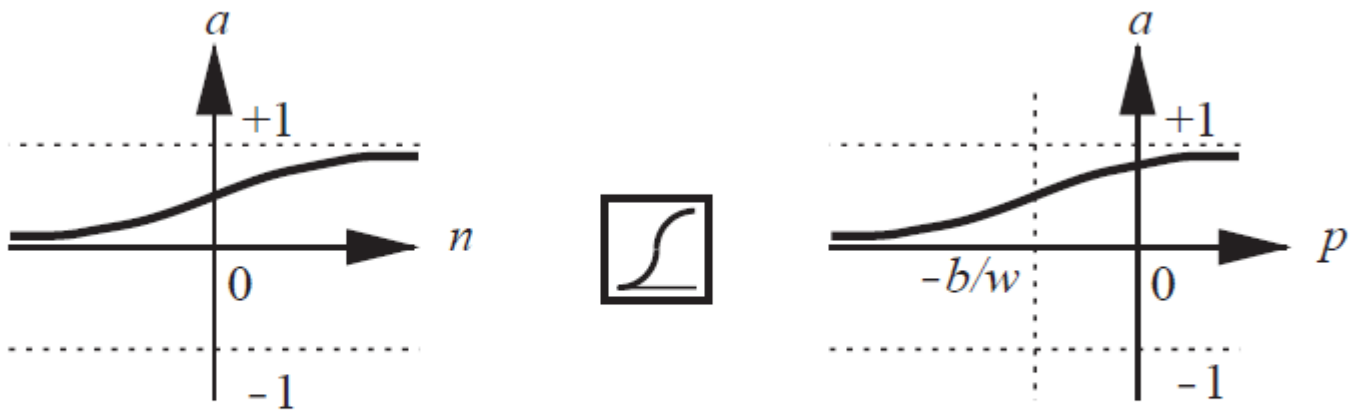
Linear Transfer Function



$a = purelin(wp + b)$

Single-Input *purelin* Neuron

Transfer Functions



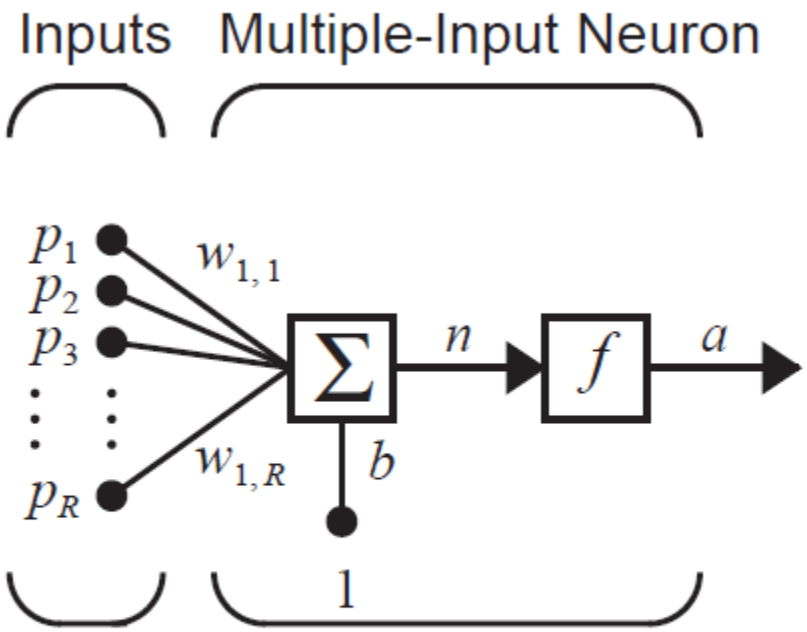
$a = logsig(n)$

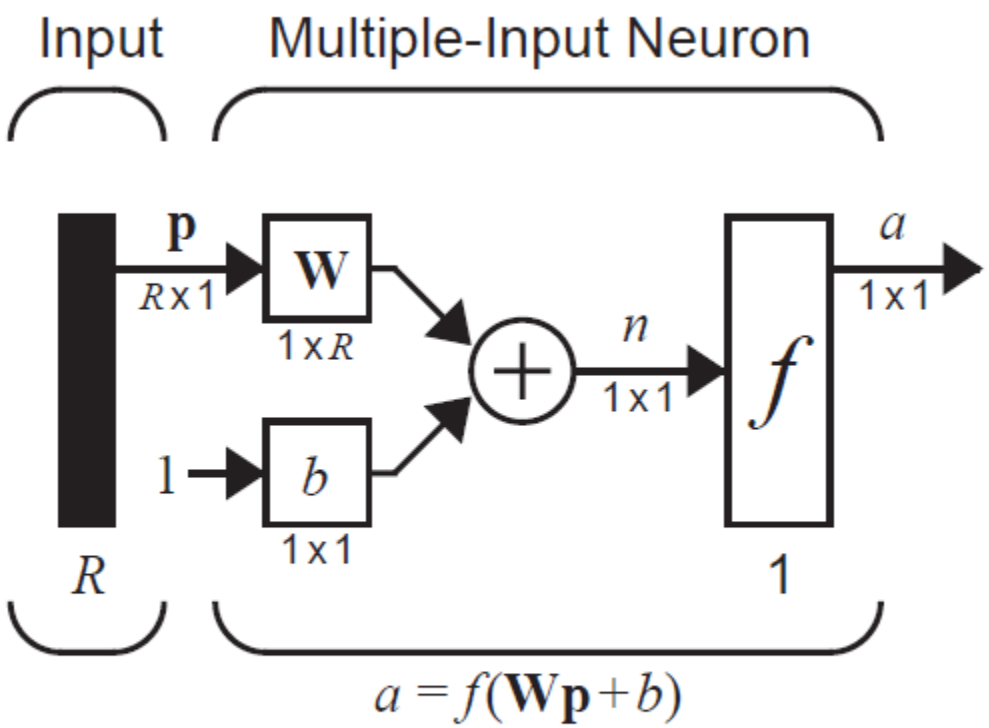
Log-Sigmoid Transfer Function

$a = logsig(wp + b)$

Single-Input *logsig* Neuron

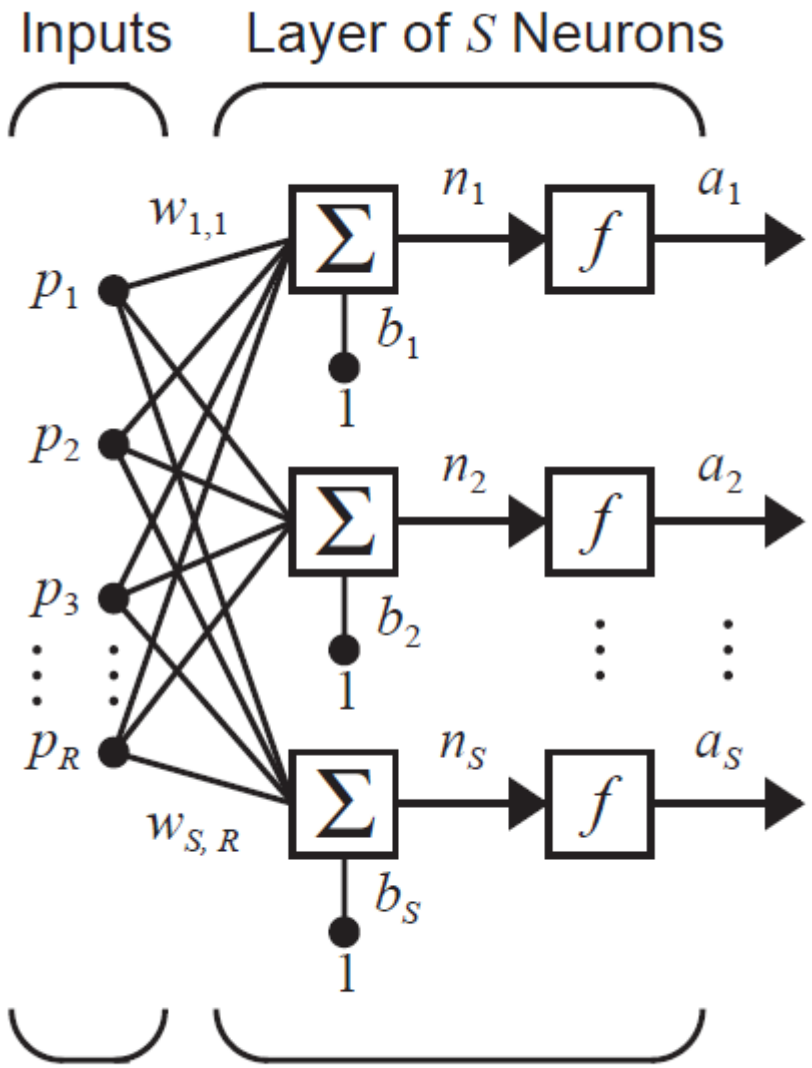
Multiple Input Neuron





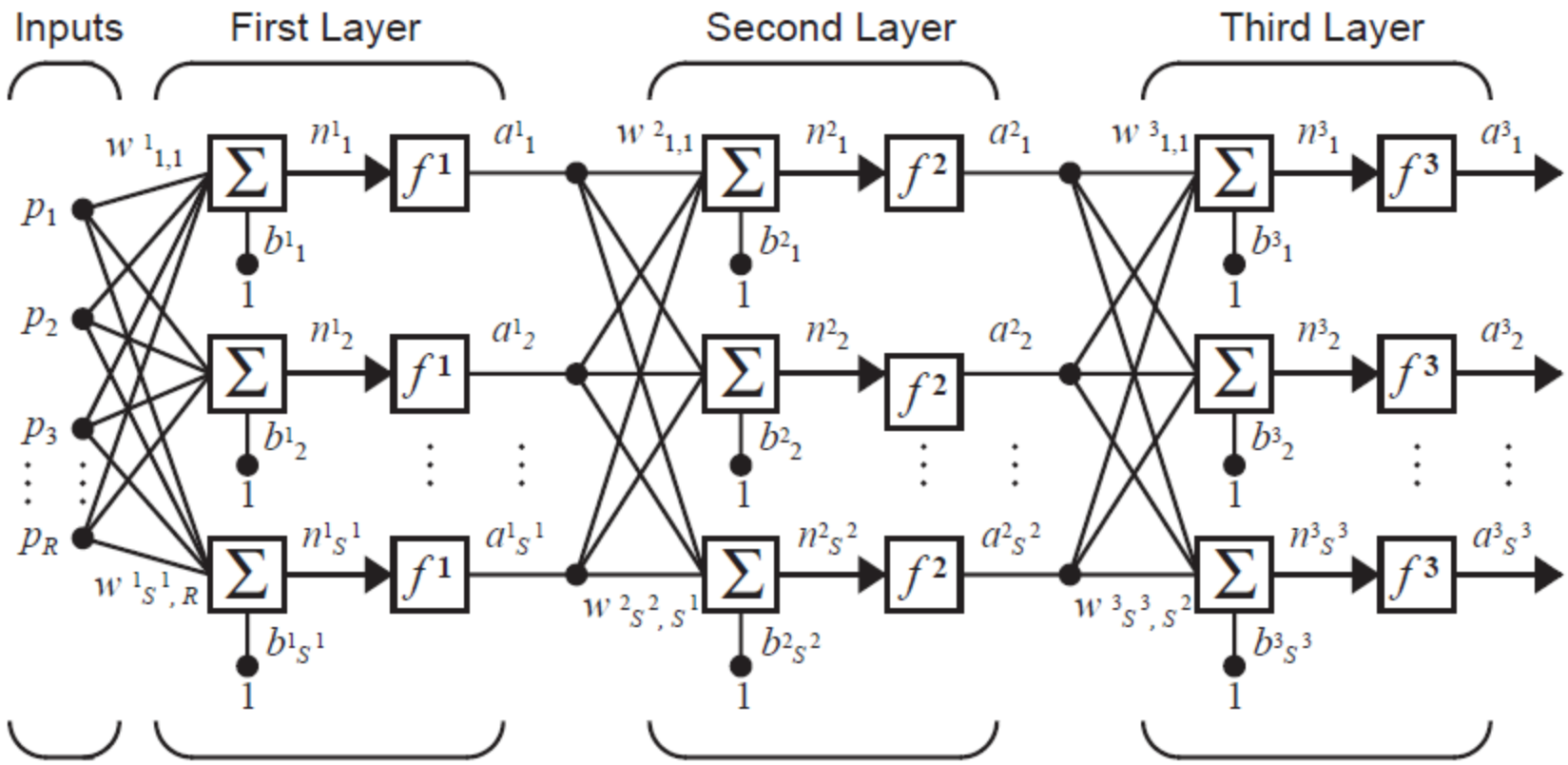
Abreviated Notation

Layers of Neurons



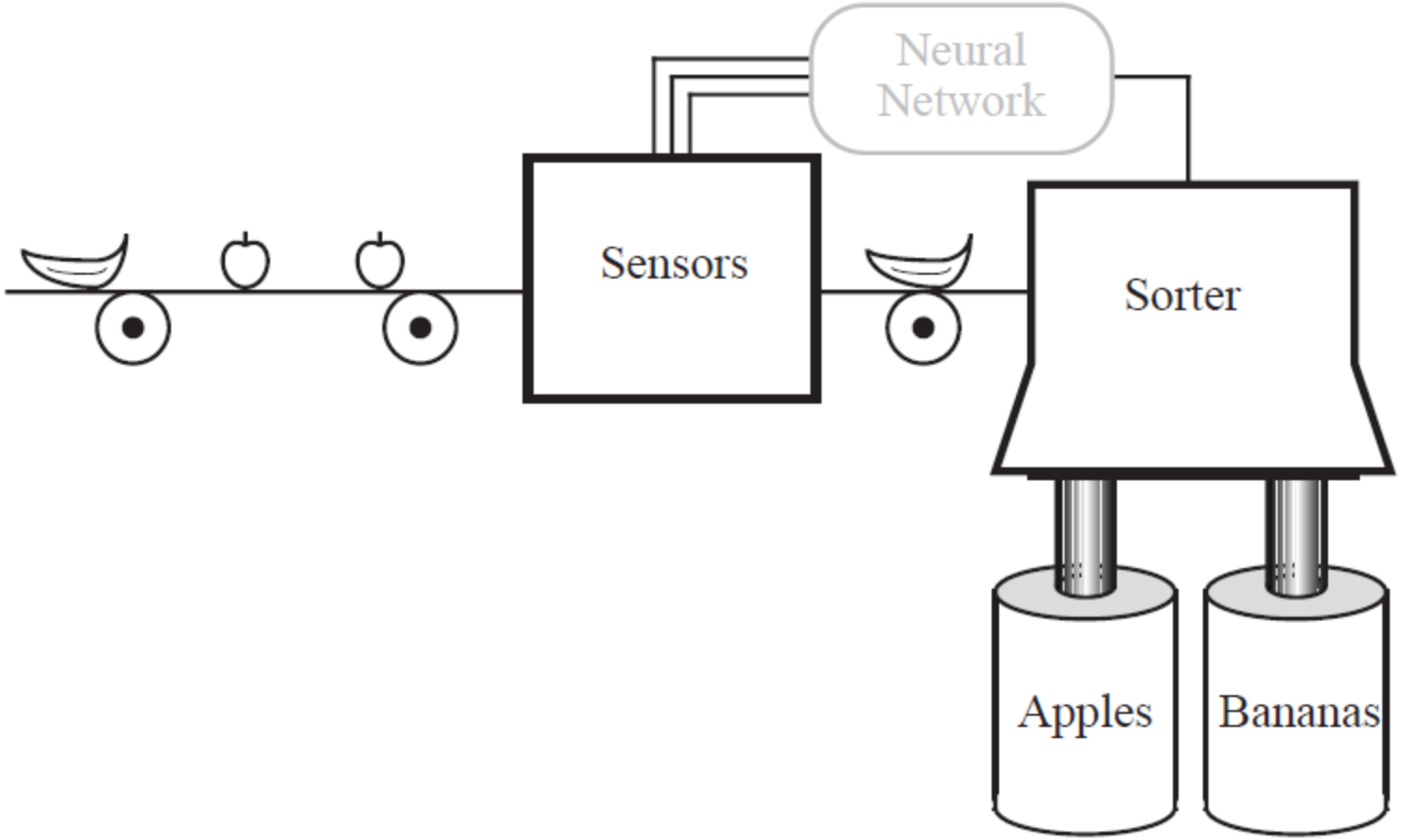
$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p}+\mathbf{b})$$

Multilayer Network

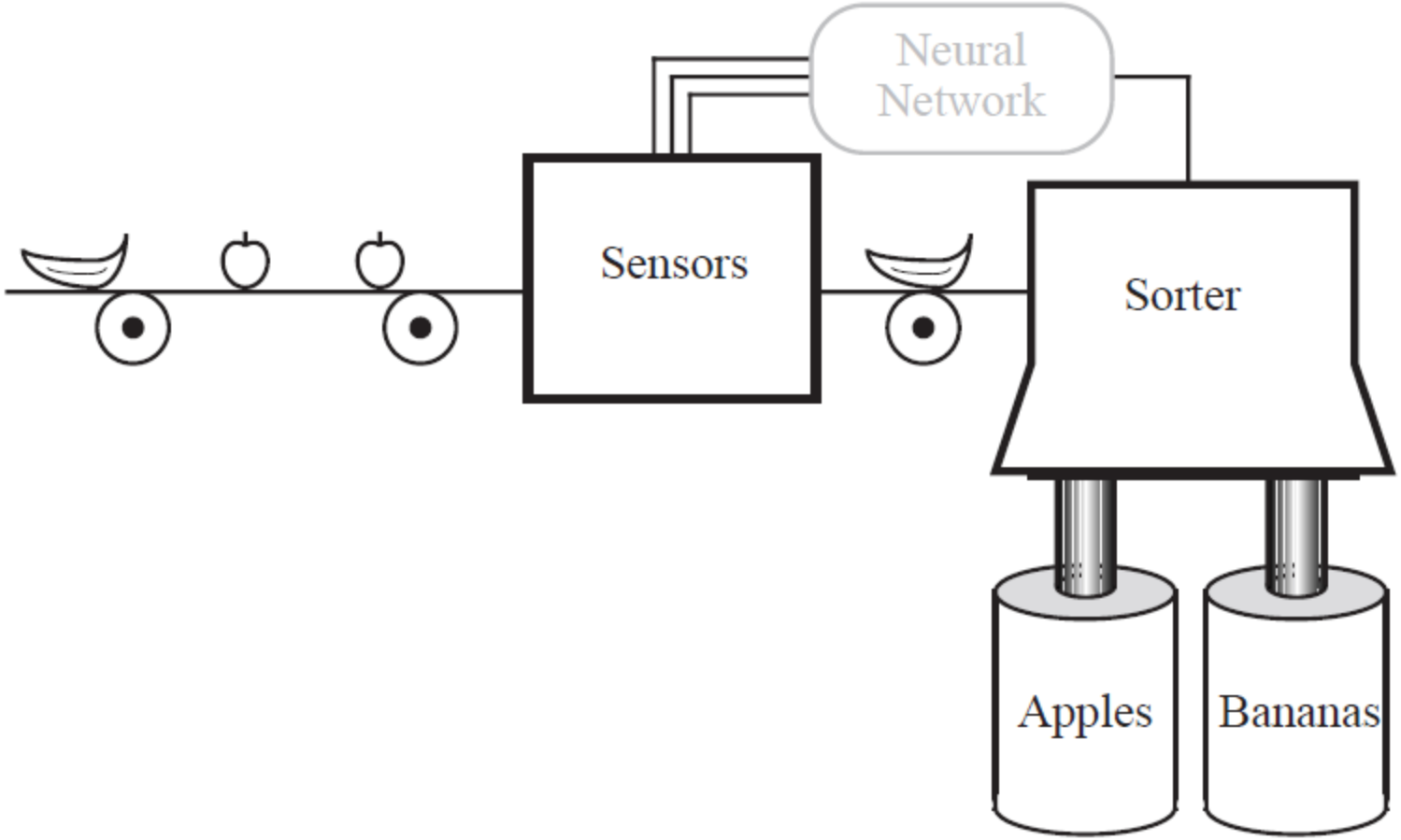


$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

An Example



An Example



Prototype Vectors

Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$

Shape: {1 : round ; -1 : eliptical}
Texture: {1 : smooth ; -1 : rough}
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

Prototype Banana

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Prototype Apple

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Neural Network Implementation in JAVA

https://www.youtube.com/watch?v=ZJNklhq1zvg

Seminar:

Neural Network Implementation in Intrusion Detection with Data Mining

Article:

2004 IEEE International Conference on Systems, Man and Cybernetics" A Hybrid Training Mechanism for Applying Neural Networks to Web-based Applications” Ko-Kang Chu Maiga Chang Yen-Teh Hsia

Stretch Break!

Thank you