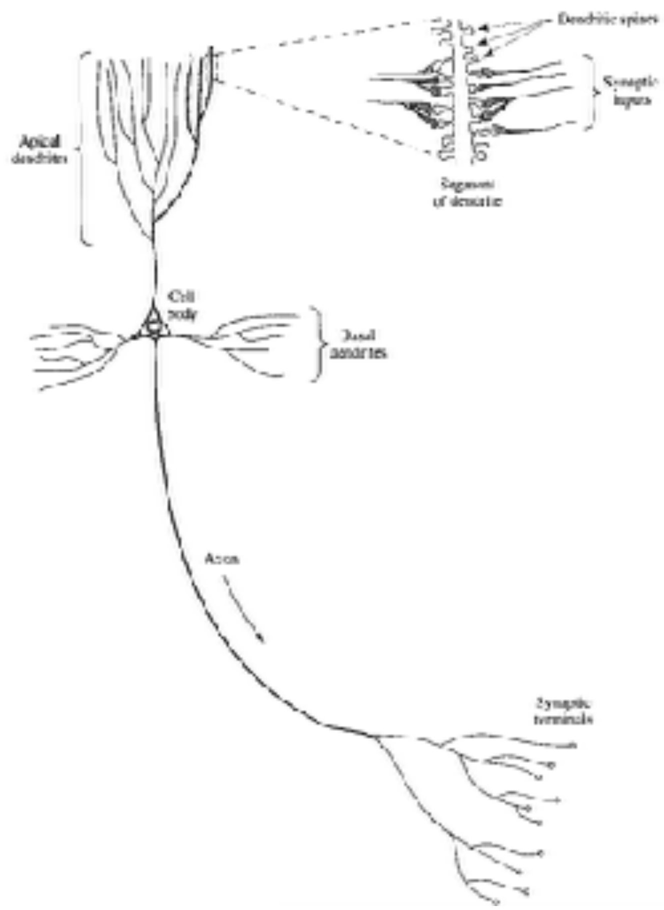
NEURAL NETWORKS

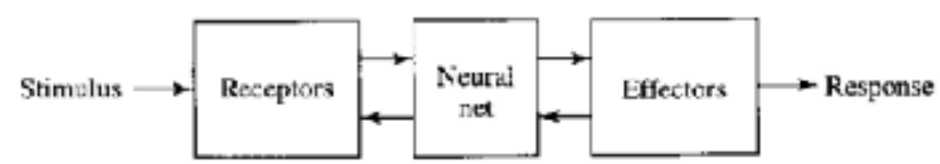
A Comprehensive Foundation
Chapter 1
by Alexander Shultz

Human Brain



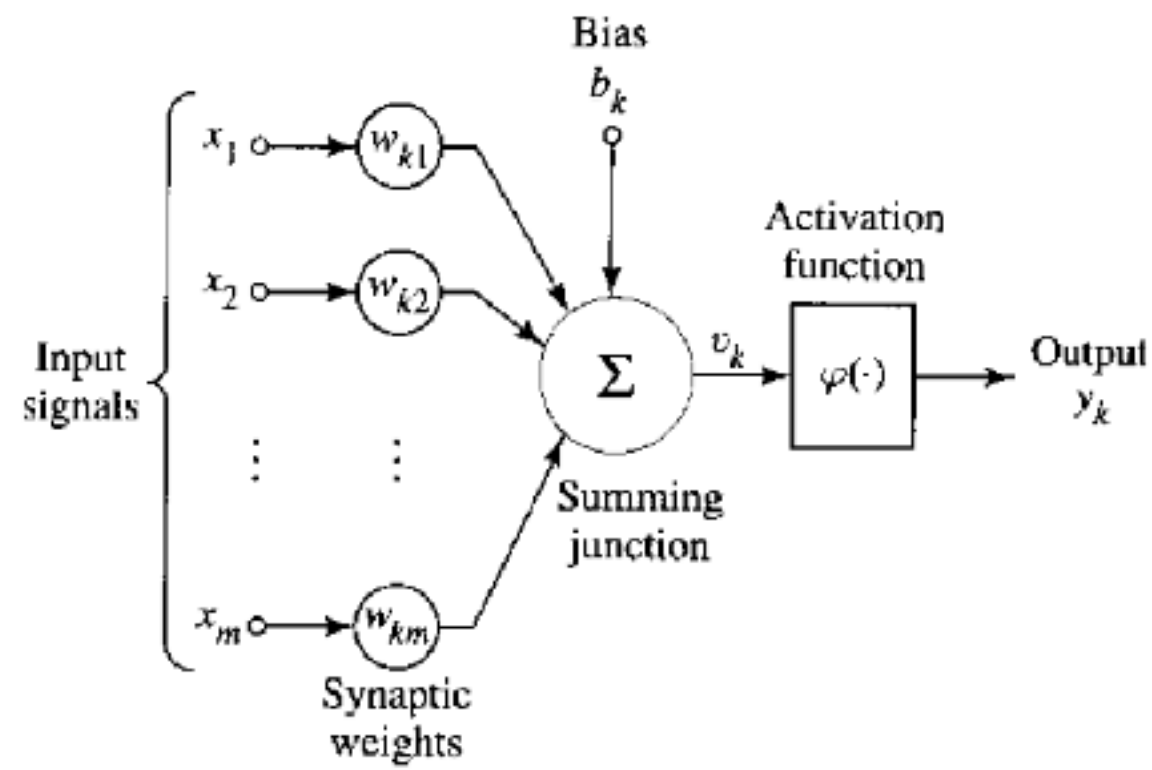
Synaptic

Axon and dendrites



models of a neuron

- synapses
- adder
- activation function
- bias



Description of neuron k

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

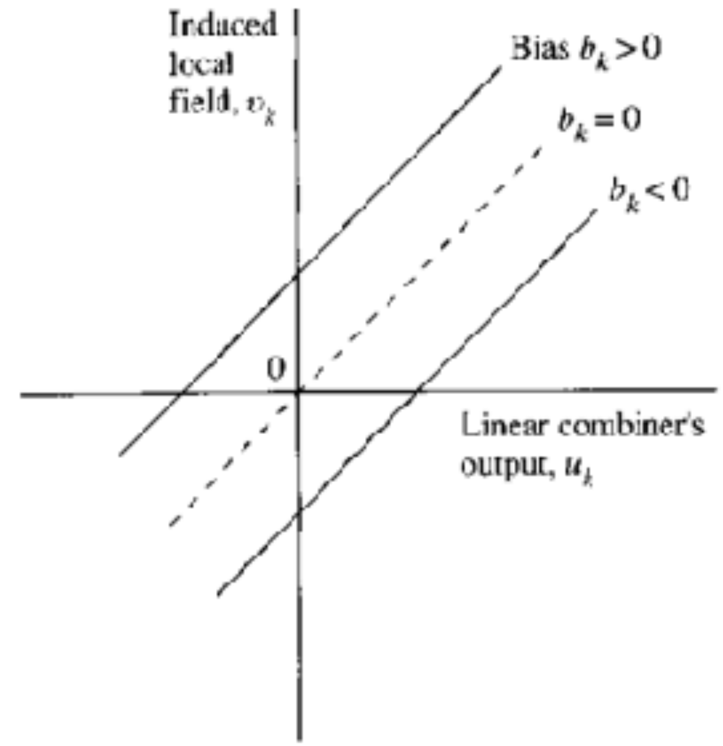
$$y_k = \varphi(u_k + b_k)$$

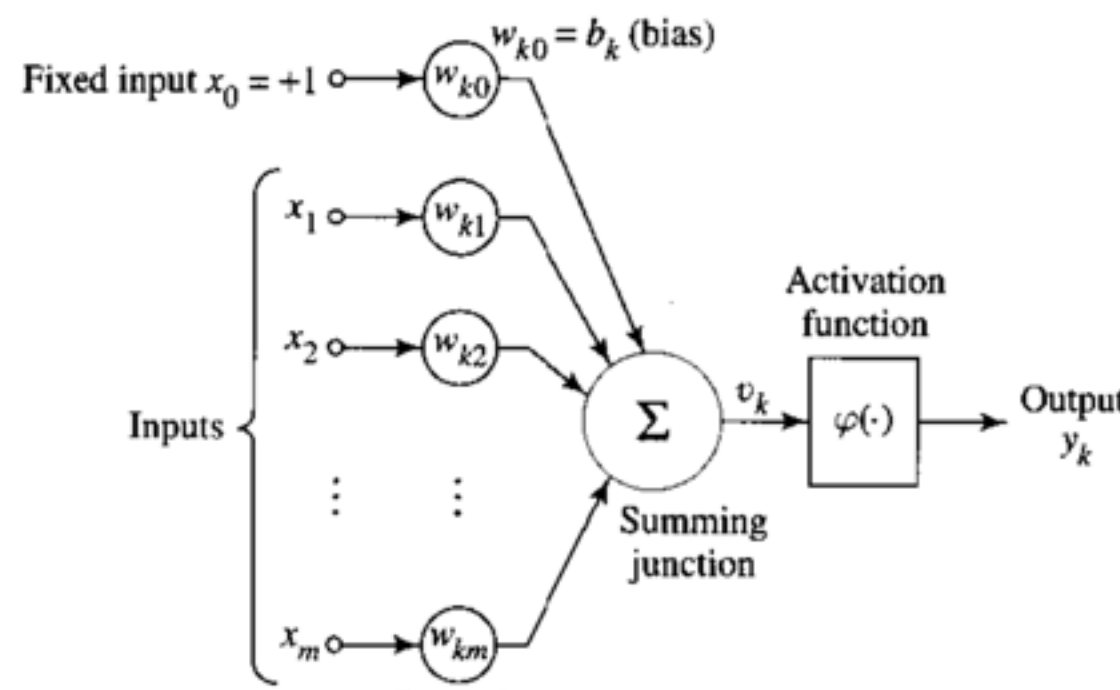
$$v_k = u_k + b_k$$

Vk: local field

$$v_k = \sum_{j=0}^{m} w_{kj} x_j$$

$$y_k = \varphi(v_k)$$



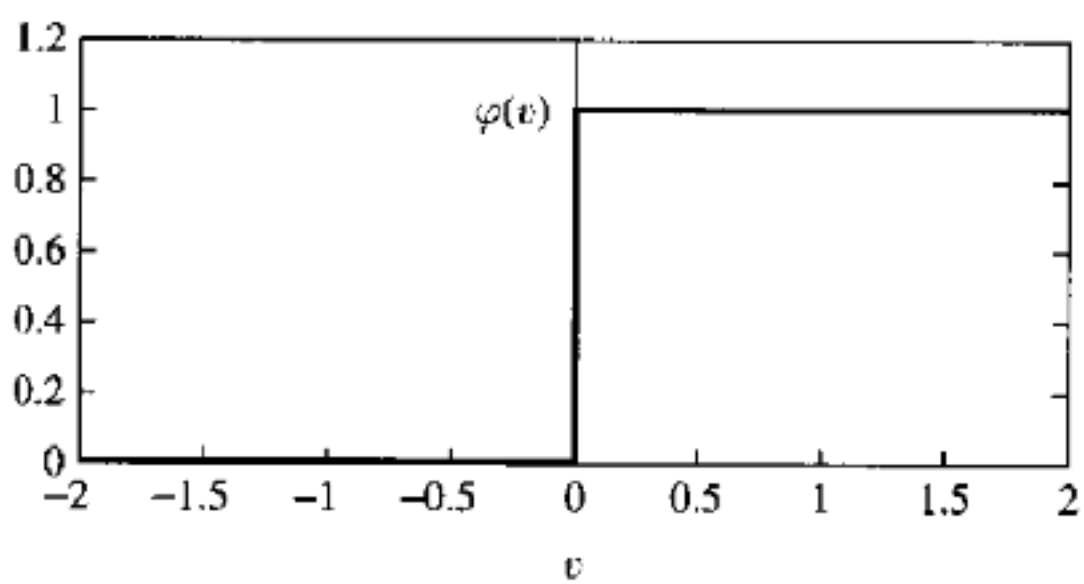


Types of Activation Function

1. Threshold Function

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases}$$



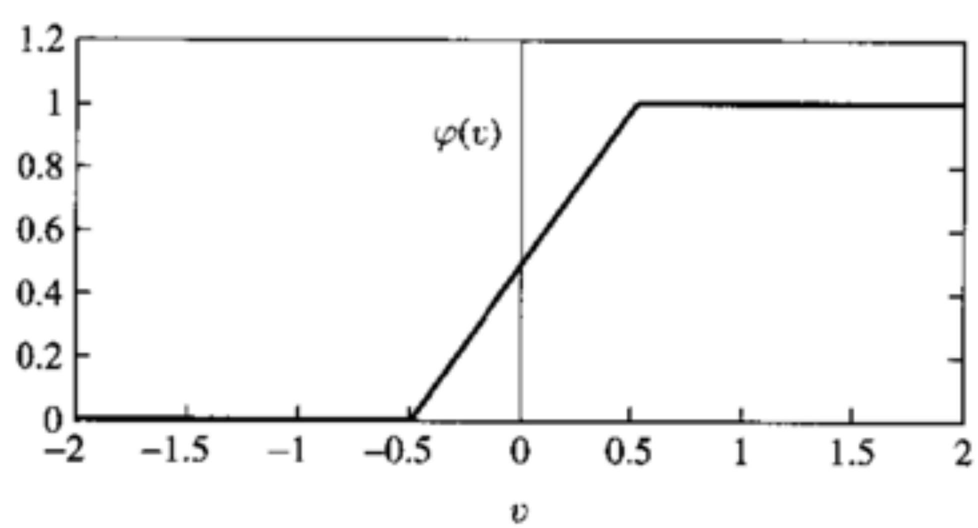
Types of Activation Function

2. Piecewise-Linear

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases}$$

A linear combiner arises if the linear region of operation is maintained without running into saturation

The piecewise-linear function reduces to a threshold function if the amplification factor of the linear region is made infinitely large



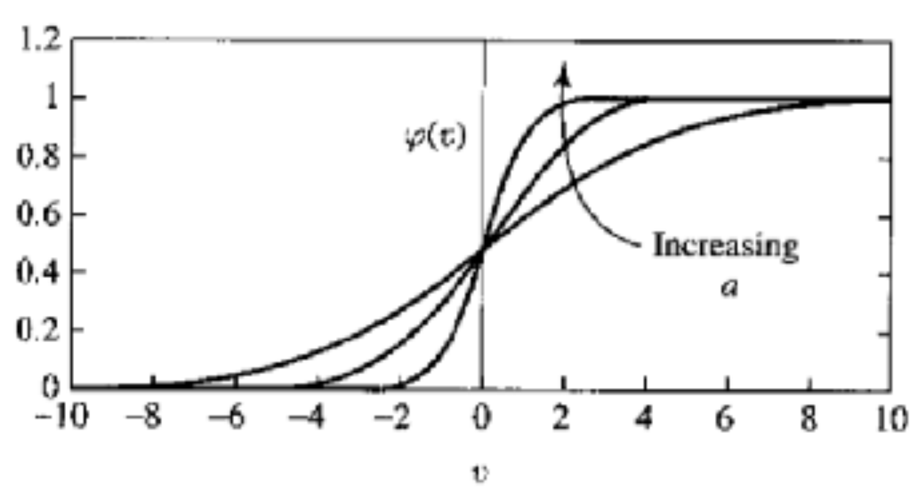
Types of Activation Function

3.Sigmoid Function

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

By varying the parameter a, wen obtain sigmoid functions of different slopes

Sigmoid function is differentiable, whereas the threshold function is not



Stochastic Model of a Neuron

It is desirable to base the analysis on a stochastic neuronal model

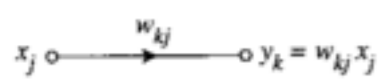
$$x = \begin{cases} +1 & \text{with probability } P(v) \\ -1 & \text{with probability } 1 - P(v) \end{cases}$$

$$P(v) = \frac{1}{1 + \exp(-v/T)}$$

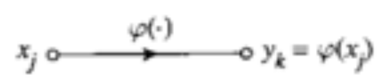
T: pseudotemperature that is used to control the noise level and therefore the uncertainty in firing

Neural networks viewed as directed graphs

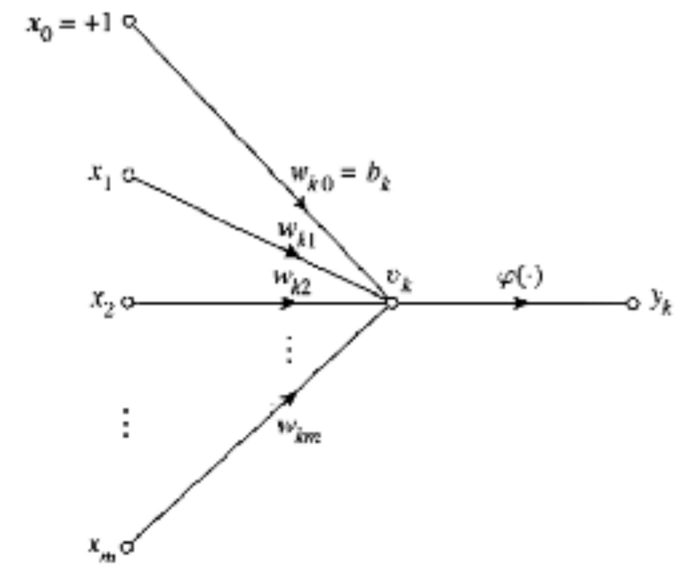
A signal flows along a link only in the direction defined by the arrow in the link



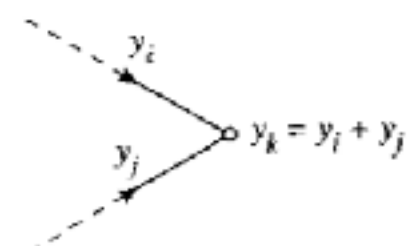
(a)



(b)

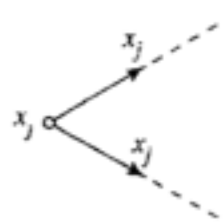


A node signal equals the algebraic sum of all signals entering the pertinent node via the incoming links

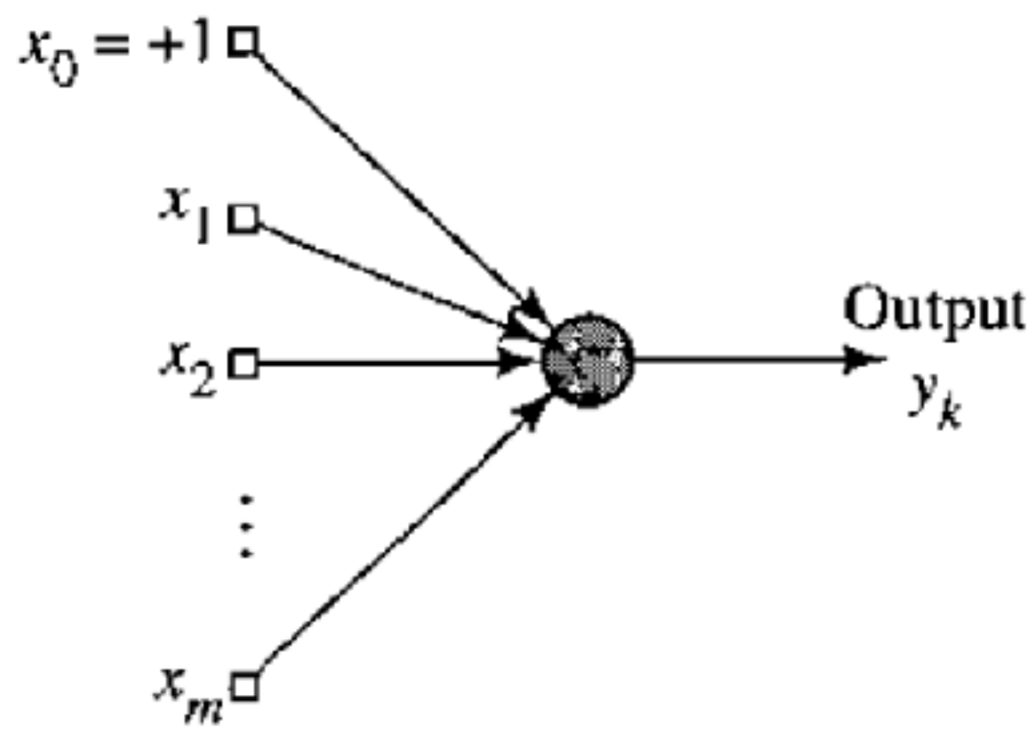


(c)

The signal at a node is transmitted to each outgoing link originating from that node, with the transmission being entirely independent of the transfer functions of outgoing links

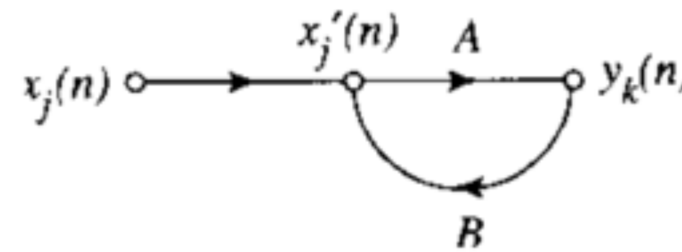


Architectural graph



Feedback

Xj(n): input signal
Xj'(n): internal signal
yk(n): output signal
A,B: operator



$$y_k(n) = A[x_j'(n)]$$

$$x_j'(n) = x_j(n) + B[y_k(n)]$$

$$y_k(n) = \frac{A}{1 - AB}[x_j(n)]$$

A/(1-AB): closed-loop operator
AB: open-loop operator

$$\frac{A}{1 - AB} = \frac{w}{1 - wz^{-1}}$$

$$= w(1 - wz^{-1})^{-1}$$

$$\left(\frac{1}{1+x}\right)^{(n)} = (-1)^n \cdot n! \cdot \frac{1}{(1+x)^{n+1}}$$

$$\therefore\ f^{(n)}(0) = (-1)^n \cdot n!$$

$$\therefore\ \frac{1}{1+x} = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n = \sum_{n=0}^{\infty} (-1)^n \cdot x^n$$

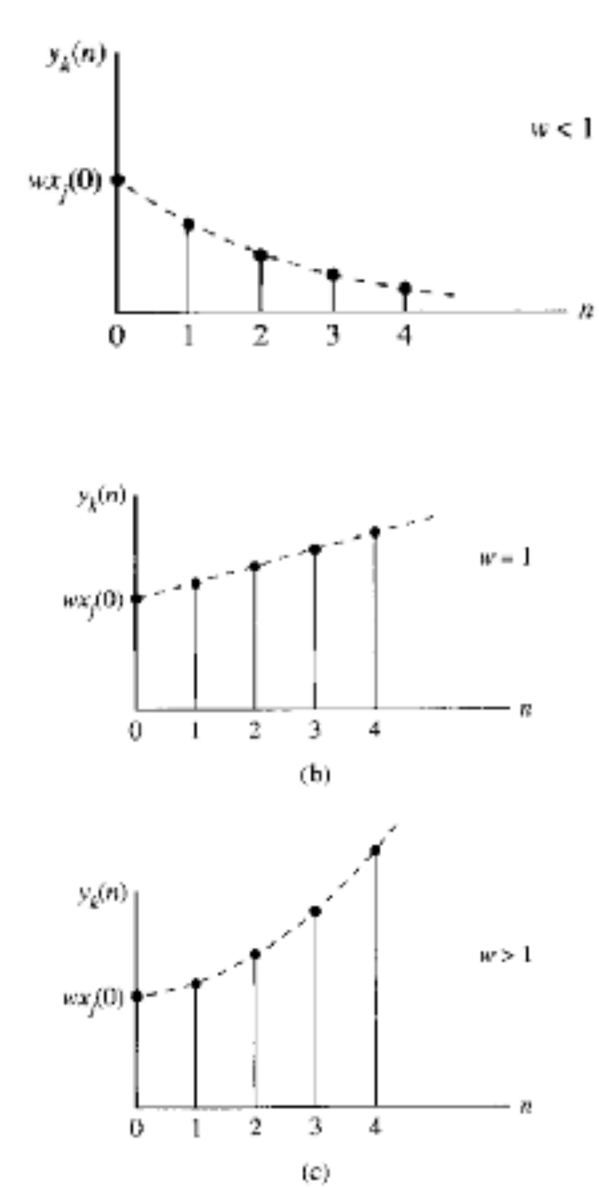
$$= w \sum_{l=0}^{\infty} w^l z^{-l}$$

$$y_k(n) = w \sum_{l=0}^{\infty} w^l z^{-l}[x_j(n)]$$

$$y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n - l)$$

|w| < 1

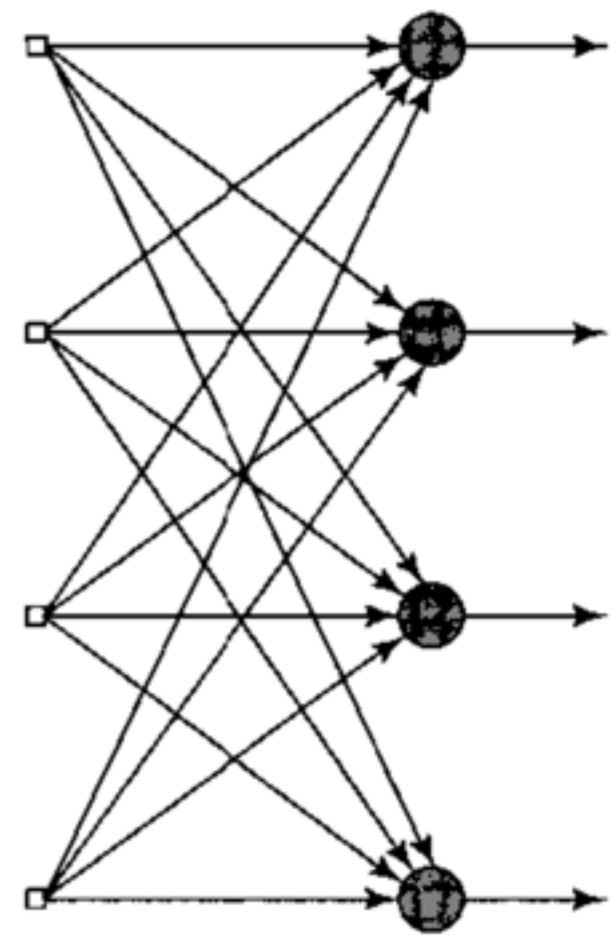
Yk(n) is exponentially convergent corresponds to a system with infinite memory the memory is fading in that the influence of a past sample is reduced exponentially with time n



Network Architectures

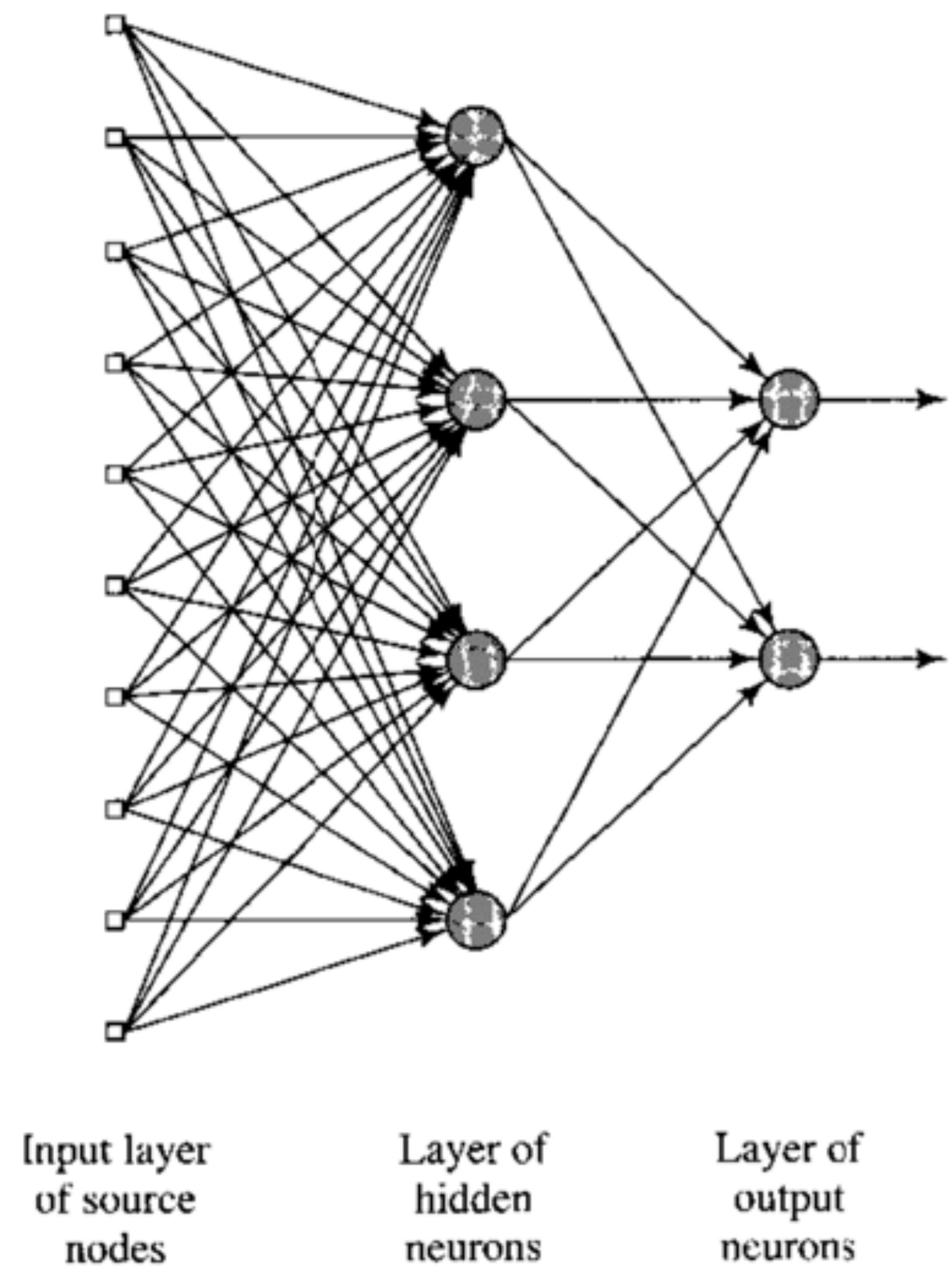
1. Single-Layer Feedforward Networks

In the simplest form of a layered network, we have an input layer of source nodes that project onto an output layer of neurons



Network Architectures

2. Multilayer Feedforward Networks



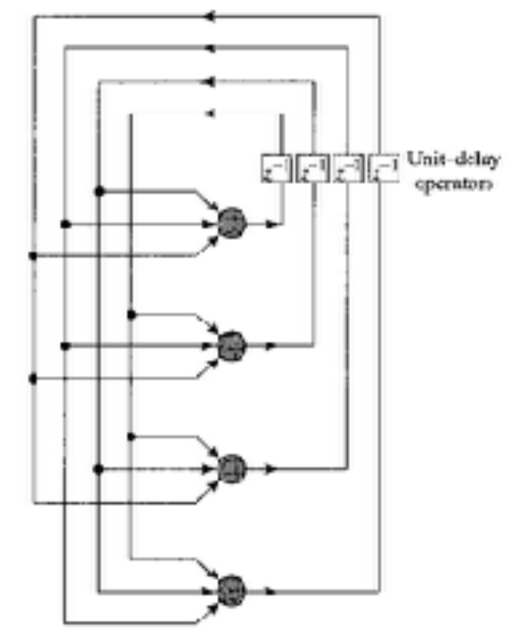
presence of one or more hidden layers, called hidden neurons or hidden units
more hidden layers, the networks is enable to extract higher-order statistics

m-h1-h2-q network

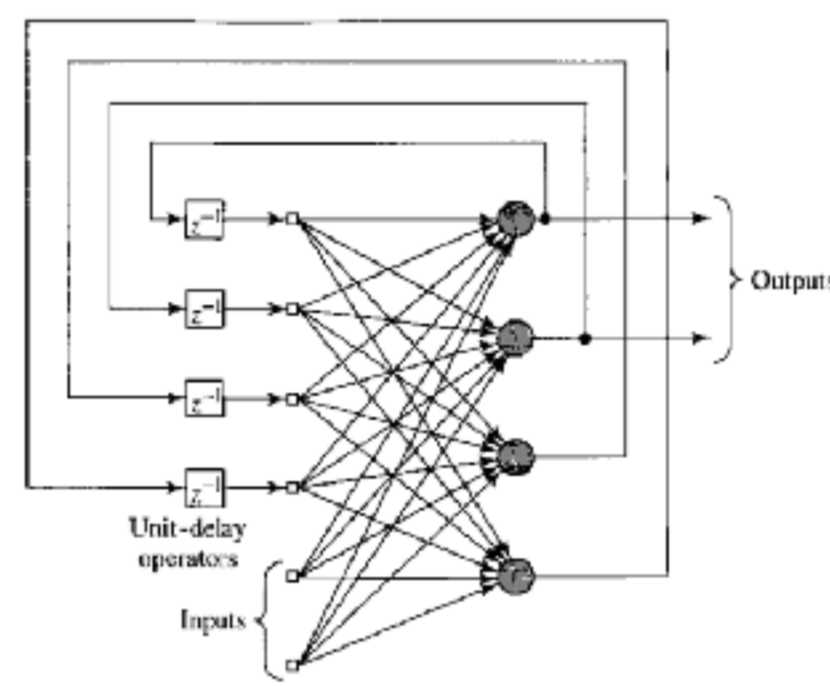
Fully connected or partially connected

Network Architectures

3.Recurrent Networks



feedforward neural network in that it has at least one feedback loop, which has a profound impact on the learning capability of the network and on its performance



Knowledge Representation

Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world.

Knowledge of the world consists of two kinds of information

1. **The known world state, represented by facts about what is and what has been known, referred to as prior information**
2. **Observations of the world, inherently noisy, being subject to errors due to sensor noise and system imperfection**

The examples can be labeled on unlabeled
can be positive or negative

Rules of knowledge

1. similar inputs from similar classes should usually produce similar representations inside the network, and should therefore be classified as belonging to the same category

The measurement of similarity

Euclidean distance

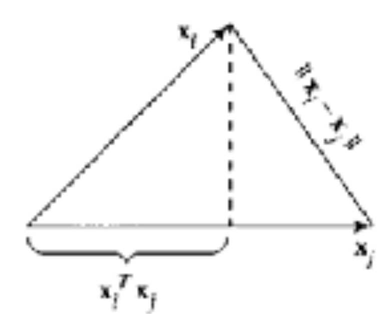
$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T$$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \left[\sum_{k=1}^{m} (x_{ik} - x_{jk})^2 \right]^{1/2}$$

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = 2 - 2\mathbf{x}_i^T\mathbf{x}_j$$

dot product or inner product

$$(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j = \sum_{k=1}^{m} x_{ik}x_{jk}$$



The more similar the vectors xi and xj are, the larger the inner product it will be

If Xi, Xj are drawn from two different populations of data

$$\boldsymbol{\mu}_i = E[\mathbf{x}_i]$$

$$d_{ij}^2 = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_j)$$

$$cov(X_i, X_j) = E[(X_i - E(X_i))(X_j - E(X_j))].$$

$$: \begin{bmatrix} cov(X_1, X_1) & cov(X_1, X_2) & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2, X_2) & \cdots & cov(X_2, X_n) \\ & & \ddots & \\ cov(X_n, X_1) & cov(X_n, X_2) & \cdots & cov(X_n, X_n) \end{bmatrix}_{n \times n} ,$$

Rule 2. Items to be categorized as separate classes should be given widely different representations in the network

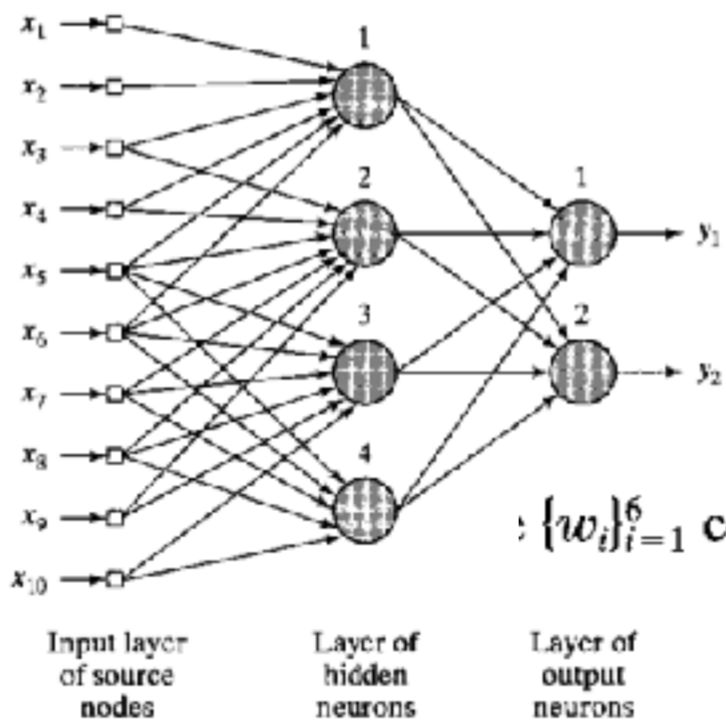
Rule 3. If a particular feature is important, then there should be a large number of a neurons involved in the representations of that item in the network

Rule 4. Prior information and invariances should be built into the design of a neural network, thereby simplifying the network design by not having to learn them

How to build prior information into neural design design

1. Restricting the network architecture through the use of local connections known as receptive fields

2. Constraining the choice of synaptic weights through the use of weight-sharing



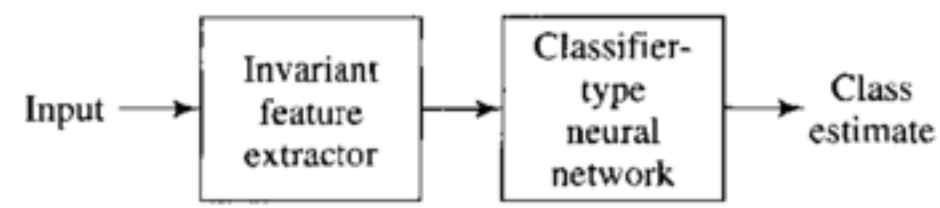
$$v_j = \sum_{i=1}^{6} w_i x_{i+j-1}, \quad j = 1, 2, 3, 4$$

: $\{w_i\}_{i=1}^{6}$ constitute the same set of weights shared by all four hidden neurons.

Build Invariances into Neural Network Design

A primary requirement of pattern recognition is to design a classifier by an output of the classifier must not be affected by transformations of the observed signal applied to the classifier input

1. Invariance by Structure



2. Invariance by Training

3. Invariant Feature Space

$$x(n) = \sum_{i=1}^{M} a_i^* x(n-i) + e(n)$$