Neuron Model

Deep Learning & Applications

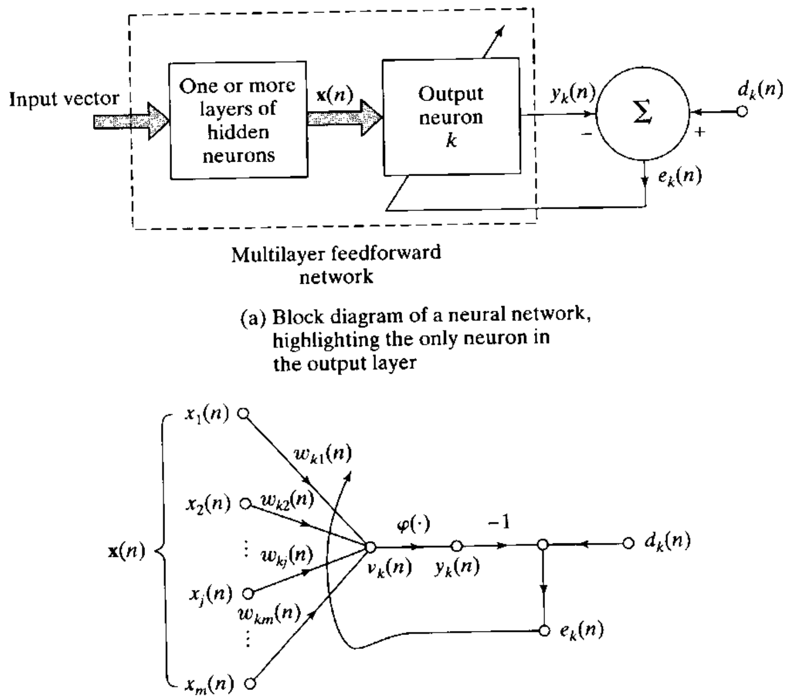
Introduction

- An ANN learns through an interactive process of adjustments to its synaptic weights and bias levels
- A set of well-defined rules for solving the learning problem is called a *learning algorithm*.
- There is no single learning algorithm for all ANNs. We rather have a variety of learning algorithms, each with its own advantages
- Also, different ways for an ANN to relate to its environment (and hence, learn) lead us to different *learning paradigms*

Error-Correction Learning

- Neuron *k* is driven by signal vector ***x****(n)* produced by hidden layers
- *n* denotes discrete time step
- $y_k(n)$ is the output of neuron *k* at time *n*
- $d_k(n)$ denotes desired output at time *n*
- After comparing actual and desired outputs, we obtain an error signal, $e_k(n)$

$$e_k(n) = d_k(n) - y_k(n)$$



(a) Block diagram of a neural network, highlighting the only neuron in the output layer

(b) Signal-flow graph of output neuron

FIGURE 2.1 Illustrating error-correction learning.

Error-Correction Learning

- $e_k(n)$ actuates a control mechanism (a sequence of corrective adjustments to synaptic weights of neuron *k*)
- The aim of these adjustments is to make $y_k(n)$ come closer to $d_k(n)$ step-by-step.
- To do this, we need to minimize a *cost function*

$$\xi(n) = \frac{1}{2} e_k^2(n)$$

(instant value of error energy)

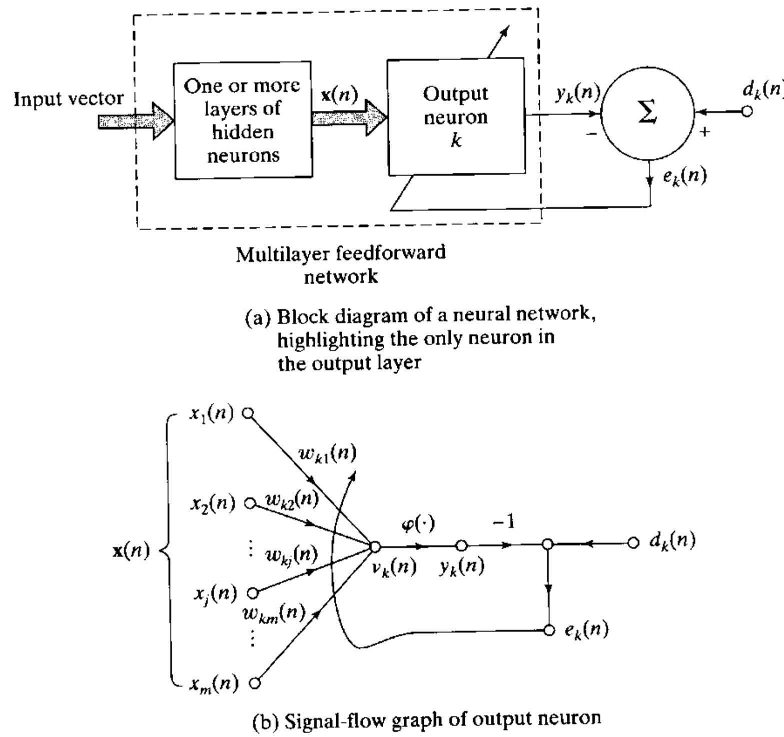


FIGURE 2.1 Illustrating error-correction learning.

Error-Correction Learning

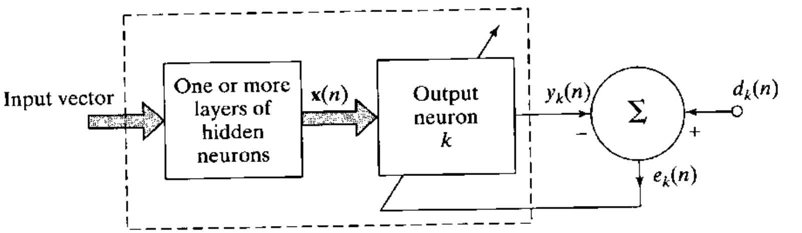
- The adjustments to the weights are continued until system reaches a steady state
- Delta rule: the adjustment $\Delta w_{kj}(n)$ for the weight w_{kj} at time step n is

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

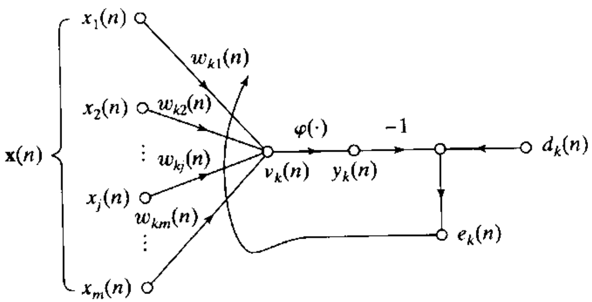
where η is the learning rate parameter

- When this is calculated, synaptic weight is updated with

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$



(a) Block diagram of a neural network, highlighting the only neuron in the output layer



(b) Signal-flow graph of output neuron

FIGURE 2.1 Illustrating error-correction learning.

Memory-Based Learning

- All (or most) past experiences are stored as correctly classified input-output examples $\{(\mathbf{x}_i, d_i)\}^N_{i=1}$
- When a new input signal, $\mathbf{x}_{\text{test}}$ is given, system responds by <u>looking at</u> <u>nearby</u> known data
  - E.g., nearest neighbor, k-nearest neighbors, radial-basis function network, etc.

Hebbian Learning

- Neuropsychologist Hebb's postulate of learning (1949) says (in short) that, when cell A repeatedly and persistently takes part in firing cell B, changes take place so that A fires B better.
- In ANN context, this is expressed as a two-part rule
  - If neurons on either side of a synapse are activated simultaneously, then synapse strength is increased
  - If neurons on either side of a synapse are activated asynchronously, then synapse strength is decreased.
- Such a synapse is called a Hebbian synapse.
- A Hebbian synapse uses a *time-dependent*, *highly local*, and strongly interactive mechanism to increase synaptic efficiency as a function of correlation between presynaptic and postsynaptic activities.

Hebbian Learning

- Synaptic weight w_{kj} for neuron k with presynaptic signal x_j and postsynaptic signal y_k. The adjustment to w_{kj} at time n (in general form) is

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n))$$

where *F(.,.)* is a function of both pre and post synaptic signals.

  - Hebb's hypothesis

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

  - Covariance hypothesis

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y})$$

    - $\bar{x}$ and $\bar{y}$ are time averaged values

Hebbian Learning

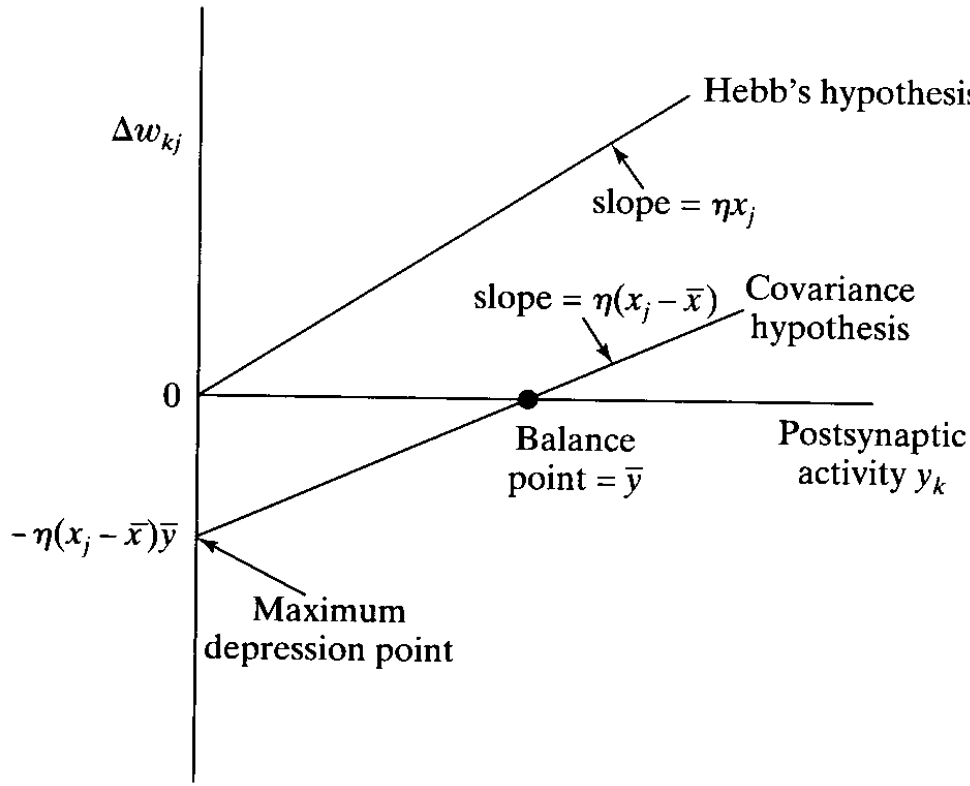


FIGURE 2.3 Illustration of Hebb's hypothesis and the covariance hypothesis.

Competitive Learning

- The output neurons of a neural network compete among themselves to become active.
  - a set of neurons that are all the same (except for synaptic weights)
  - a limit imposed on the strength of each neuron
  - a mechanism that permits the neurons to compete -> a winner-takes-all

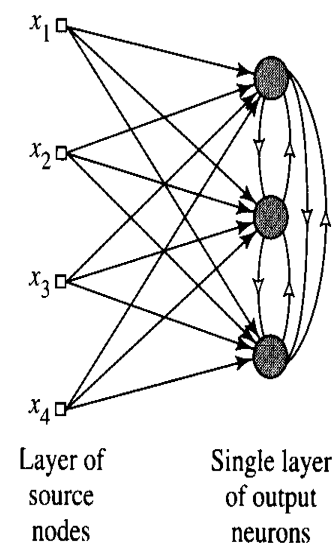


FIGURE 2.4 Architectural graph of a simple competitive learning network with feedforward (excitatory) connections from the source nodes to the neurons, and lateral (inhibitory) connections among the neurons; the lateral connections are signified by open arrows.

Competitive Learning

- The standard competitive learning rule

Δw_{kj} $= \eta(x_j - w_{kj})$ if neuron k wins the competition
$= 0$ if neuron k loses the competition

Note: all the neurons in the network are constrained to have the same length.

Boltzmann Learning

- The neurons constitute a recurrent structure and they operate in a binary manner. The machine is characterized by an energy function E where x_k and x_j are neuron states

$$E = -\tfrac{1}{2}\Sigma_j\Sigma_k\ w_{kj}x_kx_j\ ,\ j\neq k$$

- Machine operates by choosing a neuron at random then flipping the state of neuron k from state x_k to state $-x_k$ at some temperature T with probability

$$P(x_k \rightarrow -x_k) = 1/(1+\exp(-\Delta E_k/T))$$

where ΔE_k is the energy change and T is a pseudotemperature

Boltzmann Learning

Clamped condition: the visible neurons are all clamped onto specific states determined by the environment

Free-running condition: all the neurons (=visible and hidden) are allowed to operate freely

- The Boltzmann learning rule:

$$\Delta w_{kj} = \eta(\rho^{+}_{kj} - \rho^{-}_{kj}),\ j \neq k,$$

note that both ρ^{+}_{kj} and ρ^{-}_{kj} range in value from –1 to +1.