2. Learning Processes

2.1 Introduction

- Central properties of a neural network:
  - **Learning** from its environment.
  - **Improvement** in performance through learning.
- In practice, some suitable **learning algorithm** must be used.
- This adjusts the synaptic weights (free parameters of the network) according to some meaningful rules.

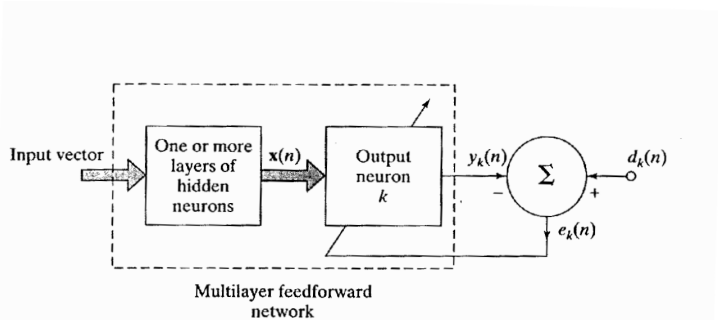
Organization of the Chapter 2

The chapter consists of three parts:

1. In Sections 2.2-2.6, five **basic learning rules** are discussed:
   - error-correction learning;
   - memory-based learning;
   - Hebbian learning;
   - competitive learning;
   - Boltzmann learning.
2. Then various **learning methodologies** are studied:
   - credit-assignment problem (Section 2.7);
   - learning with a teacher (Section 2.8);
   - learning without a teacher (Section 2.9);
3. Learning tasks, memory, and adaptation are studied in Sections 2.10-2.12.

2.2 Error-Correcting Learning

- Consider the simple case of a single neuron k.
- This lies in the output layer of the multilayer feedforward network



- Neuron k is driven by a *signal vector* $\mathbf{x}(n)$ produced by the preceding hidden layer(s).
- The argument n denotes discrete time, or actually the iteration number in the learning algorithm.

- $y_k(n)$ denotes the *output signal* of neuron k.
- This is compared to the *desired response* or *target output* $d_k(n)$.
- Thus the *error signal* is defined by

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

- The error signal $e_k(n)$ is used to adjust the values of the synaptic weights.
- The output signal $y_k(n)$ should approach the desired response $d_k(n)$ in a step-by-step manner.
- This is achieved by minimizing a *cost function* or *index of performance* defined in this case by

$$\mathcal{E}(n) = 0.5[e_k(n)]^2$$

- $\mathcal{E}(n)$ is the instantaneous value of the error energy

- Learning is continued until the synaptic weights are essentially stabilized.
- This type of learning process is called *error-correction learning*.
- Minimization of the cost function $\mathcal{E}(n)$ leads to the so-called **delta rule** or **Widrow-Hoff rule** (1960):

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n). \tag{1}$$

- Here $w_{kj}(n)$ is the j-th element of the weight vector $\mathbf{w}_k(n)$ of the output neuron k.
- $x_j(n)$ is the corresponding j-th component of the signal vector $\mathbf{x}(n)$.
- $\Delta w_{kj}(n)$ is the adjustment (update, correction) made to the weight $w_{kj}(n)$ at iteration step n.
- The *learning-rate parameter* η is a positive constant which determines the amount of correction.
- The error signal must be directly measurable.

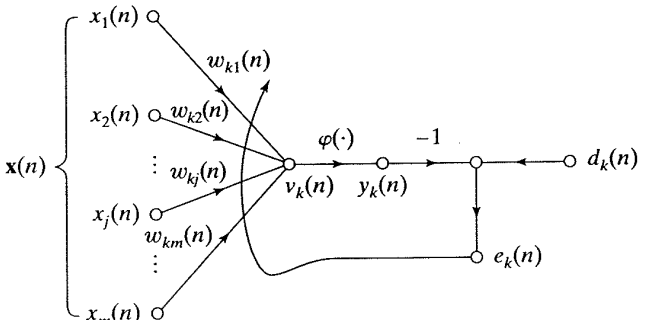
- This means that we must know the desired response $d_k(n)$, or the learning process is supervised.
- Widrow-Hoff learning rule is *local*.
- It uses information directly available to neuron k through its synaptic connections.
- The new, updated value of the synaptic weight $w_{kj}(n)$ is computed from the rule

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n). \tag{2}$$

- This formula could be represented also in the form

$$w_{kj}(n) = z^{-1}[w_{kj}(n+1)] \tag{3}$$

where z^{-1} is the *unit-delay operator (storage element)* used widely in digital signal processing.



- Signal-flow graph representation of the error-correction learning process.
- Actually this is a *closed-loop feedback system*

- The learning parameter η is crucial to the performance of error-correction learning in practice.
- It affects to the:
  - stability of the learning algorithm;
  - convergence speed;
  - final accuracy achieved.
- Error-correction learning is discussed in much greater detail in Chapters 3 and 4.

2.3 Memory-Based Learning

- In *memory-based learning*, all (or most) of past experiences are explicitly stored in a large memory.
- This consists of correctly classified input-output examples $\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \ldots, (\mathbf{x}_N, d_N)\}$.
- Again, $\mathbf{x}_i$ denotes i-th input vector and d_i the corresponding desired response.
- Without loss of generality, d_i can be restricted to be a scalar.
- Typically, d_i is the number of pattern class.
- Consider now classification of a test vector $\mathbf{x}_{test}$ not seen before.
- This is done by retrieving and analyzing the training data in a local neighborhood of $\mathbf{x}_{test}$.
- All memory-based learning algorithms involve two parts:

1. Criterion used for defining the local neighborhood of the test vector $\mathbf{x}_{test}$.
2. Learning rule applied to the training examples in the local neighborhood of the test vector $\mathbf{x}_{test}$.

- A simple yet effective memory-based learning algorithm is known as the **nearest-neighbor rule**.
- The vector $\mathbf{x}'_N$ belonging to the set of training vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is the nearest neigbor of $\mathbf{x}_{test}$ if

$$\min_i d(\mathbf{x}_i, \mathbf{x}_{test}) = d(\mathbf{x}'_N, \mathbf{x}_{test})$$

  where $d(\mathbf{x}_i, \mathbf{x}_{test})$ is the Euclidean distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_{test}$.
- $\mathbf{x}_{test}$ is classified into the same class as its nearest neigbor $\mathbf{x}'_N$.
- Nearest neigbor rule is independent of the underlying distribution.

- Assume that:
  - The training samples are independently and identically distributed;
  - The sample size N is infinitely large.

- One can then show that the probability of error in nearest neighbor classification is at most twice the *Bayes probability of error* (Cover and Hart, 1967).

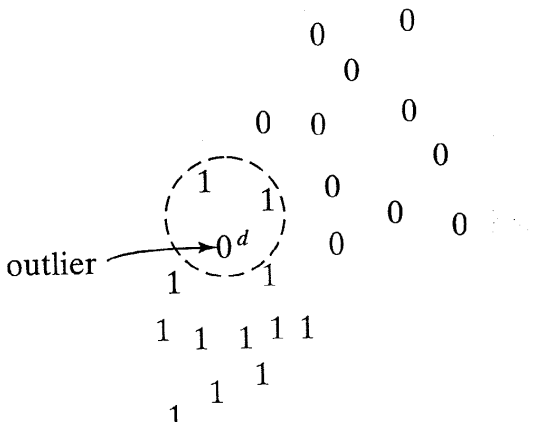
- The Bayes error is the smallest possible (optimal one).

- It is discussed in Chapter 3.

- Thus half the classification information in an infinitely large training set is contained in the nearest neighbor.

K-nearest neighbor classifier

- $\mathbf{x}_{test}$ is classified to the class which is most frequently represented in its k-nearest neighbors (a majority vote).
- The k-nearest neigbor classifier averages information, rejecting single outliers.
- *outlier* = exceptional, often erroneous observation.



- k-nearest neighbor classifier ($k = 3$)
- Another important type of memory-based classifiers: radial-basis function networks (Chapter 5).

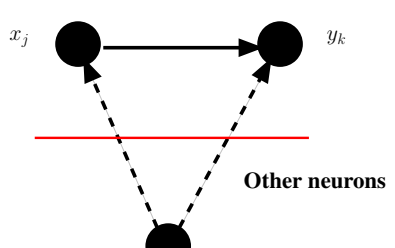
2.4 Hebbian Learning

- *Hebb's postulate of learning* (1949) is the oldest and most famous neural learning rule
- A modern, expanded version of Hebb's learning rule consists of two parts:
  1. If two neurons on either side of a synapse (connection) are activated simultaneously (synchronously), then the strength of that synapse is selectively increased.
  2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.
- Such a synapse is called a *Hebbian synapse*.

- **Key properties of a Hebbian synapse:**

  1. *Time-dependent mechanism*
     - Modification depends on the exact time of occurrence of presynaptic and postsynaptic signals.
  2. *Local mechanism*
     - A Hebbian synapse uses only local information available for a neuron.
  3. *Interactive mechanism*
     - Change in a Hebbian synapse depends both on presynaptic and postsynaptic signals.
     - Interaction between these signals can be either deterministic or statistical in nature.
  4. *Conjunctional or correlational mechanism*
     - Correlation over time between presynaptic and postsynaptic signals is responsible for a synaptic change.

- Synaptic modifications can be classified as *Hebbian*, *anti-Hebbian*, *and non-Hebbian* ones.
- A Hebbian synapse increases its strength for positively correlated presynaptic and postsynaptic signals.
- It decreases the strength for uncorrelated and negatively correlated signals.
- An *anti-Hebbian* synapse operates just in the reverse manner.
- A *non-Hebbian* synapse does not use Hebbian type learning.



Mathematical Models of Hebbian Learning

- Consider a synaptic weight w_{kj} of neuron k.
- The respective presynaptic (input) signal is denoted by x_j.
- The postsynaptic (output) signal is denoted by y_k.
- The change (update) of w_{kj} at time step n has the general form

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

  where $F(y, x)$ is a function of both postsynaptic and presynaptic signals.

- Consider two specific forms of the general Hebbian learning rule.

1. **Standard Hebbian learning rule:**

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

- Here η is again the learning rate or parameter.

- Repeated application of the input (presynaptic) signal x_j leads to an increase in the output signal y_k.

- Finally this leads to an *exponential growth* and saturation of the weight value.

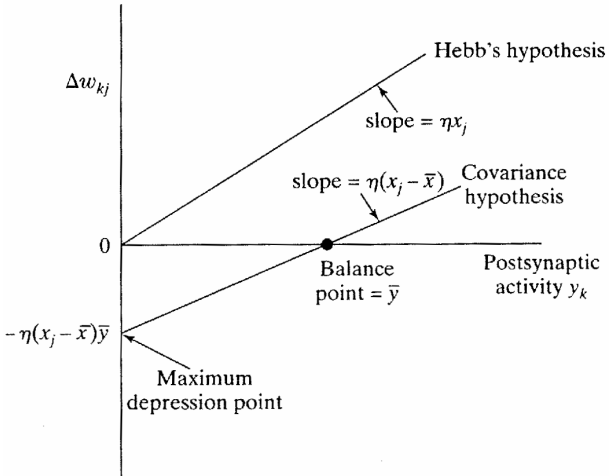
2. **Covariance Hebbian rule:**

$$\Delta w_{kj}(n) = \eta[x_j(n) - m_x][y_k(n) - m_y]$$

Here m_x and m_y are time averages of the presynaptic input signal x_j and postsynaptic output signal y_k, respectively.

Covariance rule can converge to a nontrivial state $x_j = m_x$, $y_k = m_y$ (error in Haykin's book here!?).

The synaptic strength can both increase and decrease.



Standard Hebbian rule and the covariance rule.

- In both cases, the weight update depends on the output signal y_k linearly.
- Hebbian learning has strong physiological evidence.

2.5 Competitive Learning

- In competitive learning, the neurons in the output layer compete to become active (fired).
- Only a single output neuron is active at any one time.
- In Hebbian learning, several output neurons may be active simultaneously.
- Competitive learning is highly suitable for finding relevant features for classification tasks.

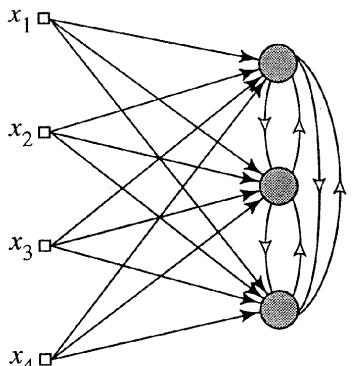
- **Three basic elements** of a competitive learning rule:

  1. A set of similar neurons except for some randomly distributed synaptic weights.
     - Therefore the neurons *respond differently* to input signals.
  2. A *limit* imposed on the strength of each neuron.
  3. A competing mechanism for the neurons.
     - Only one output neuron has the right to respond to an input signal.
     - The winner of the competition is called a *winner-takes-all neuron*.

- As a result of competition, the neurons become specialized.

- They respond to certain type of inputs, becoming *feature detectors* for different input classes.

- Simplest form of a competitive neural network



- Feedback connections between the competing output neurons perform *lateral inhibition*.
- Each neuron tends to inhibit the neuron to which it is laterally connected.
- A neuron k is the winning neuron if its induced local field v_k for a given input pattern $\mathbf{x}$ is the largest one.
- Mathematically, the output signal

$$y_k = 1, \text{ if } v_k > v_j \text{ for all } j, j \neq k.$$

  For other than the winning neuron, the output signal $y_k = 0$.
- The local field v_k represents the combined action of all the forward and feedback inputs to neuron k.
- Typically, all the synaptic weights w_{kj} are positive.

- Normalization condition giving equal portion of synaptic weight "mass" to each neuron:

$$\sum_j w_{kj} = 1 \text{ for all } k$$

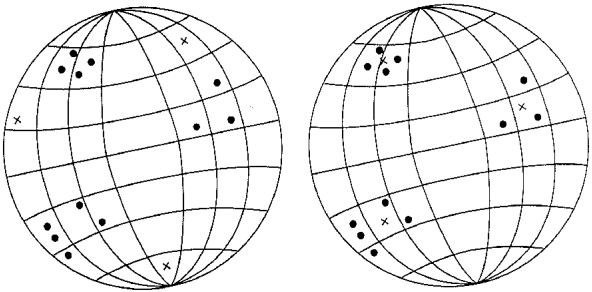
- A neuron learns by shifting synaptic weights from its inactive input nodes to the active ones.

- **The standard competitive learning rule:**

$$\Delta w_{kj} = \eta(x_j - w_{kj}) \text{ if neuron k wins;}$$

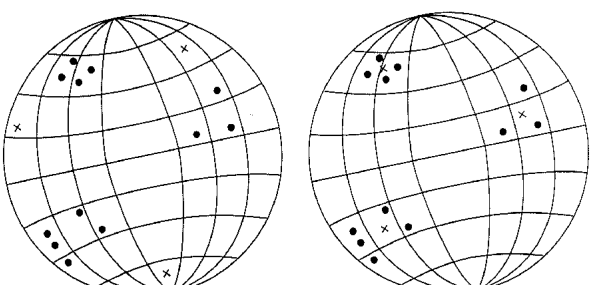
$$\Delta w_{kj} = 0 \text{ if neuron k loses the competition.}$$

- This learning rule moves the weight vector $\mathbf{w}_k$ of the winning neuron k toward the input pattern $\mathbf{x}$.



- Here both the input vectors $\mathbf{x}$ and the weight vectors $\mathbf{w}_k$ are scaled to have unit length (Euclidean norm).
- Then they are points on the surface of an N-dimensional hypersphere (assuming N-dimensional vectors).
- Initial state (Fig. a) shows three clusters of data points (black dots) and initial values of three weight vectors (crosses).

- Figure b shows a typical final state of a network resulting from competitive learning.
- The weight vectors have moved to the gravity centers of clusters.
- In more difficult cases, competitive learning algorithms may fail to find stable clusters.



2.7 Credit-Assignment Problem

- *Credit assignment* (Minsky, 1961) is a useful concept in studying learning algorithms for distributed systems.
- Basic problem: Assign *credit* or *blame* to internal decisions made by a learning machine.
- The reward or punishment is based on the quality of the overall output provided by the learning machine.
- Often the outputs of a learning machine depend directly on some actions and only indirectly on the internal decisions.

- In these situations the credit-assignment problem may be decomposed into two subproblems (Sutton, 1984):

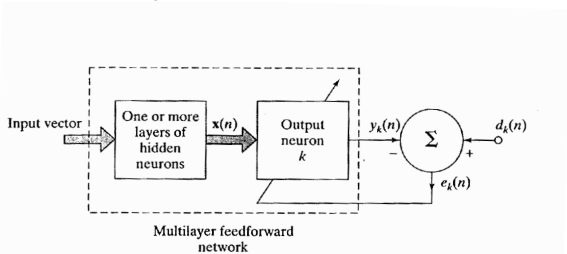
    1. *Temporal credit-assignment problem*
       - The assignment of credits to actions for outcomes.
       - Involves the instants of time when the actions that deserve credit were actually taken.
    2. *Structural credit-assignment problem*
       - Credit assignment to the internal structures of the actions generated by the system.

- The structural credit-assignment problem is relevant when we must determine precisely which component of the system should alter its behavior and how much for improving overall system performance.

- The temporal credit-assignment problem is relevant when we must determine which of the actions were responsible for the outcomes obtained.

- Often both problems encounter simultaneously.

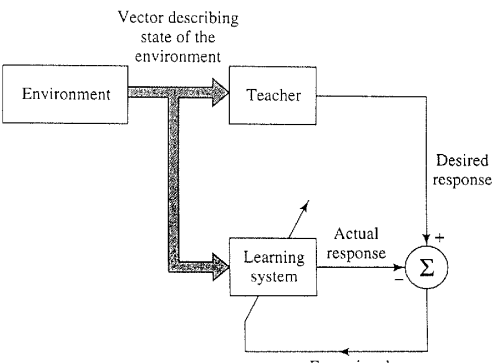
- The credit-assignment problem arises for example when error-correction learning is applied to a multilayer feedforward neural network.
- Both the neurons in the hidden layers and in the output layer are responsible for the overall behavior of the network.
- As an example, consider this



- It is straightforward to adjust the synaptic weights of the output neuron using the known desired response and error-correction learning
- Fundamental question: how the weights of hidden neurons are adjusted? — This will be discussed later on in Chapter 4.

2.8 Learning with a Teacher

- Called also **supervised learning**.
- A block diagram of a supervised learning system.



- Assumption: teacher has knowledge about the environment.

- This knowledge is represented by the known input-output pairs (training data).
- The environment is unknown to the neural network.
- Using error-correction learning (for example), knowledge of the teacher is transferred to the neural network.
- After learning, the neural network should be able to process new data independently without a teacher.
- The learning system is a closed-loop feedback system.
- Typical error measure: mean-square error, as a function of the free parameters of the system.
- This function can be described geometrically using a multidimensional *error surface*.
  - Coordinates: the free parameters to be optimized.
- The error surface is *averaged* over all possible input-output examples.

- Any supervised operation corresponds to a point on the error surface.
- The optimum operating point is the global minimum of the error surface.
- In supervised learning, this global minimum is searched iteratively using the *gradient* (derivative) of the error surface.
- The (negative) gradient vector shows the direction of *steepest descent* at any point of the error surface.
- In practice, an *instantaneous estimate* of the gradient vector is often used.
- Results in statistical fluctuations in learning.
- However, the correct minimum may be achieved using enough training data and iterations.
- Gradient type learning may result in a local minimum.