Perceptron Networks and Applications

M. Ali Akcayol
Gazi University
Department of Computer Engineering

Content

- Multilayer networks
- Multilevel discrimination
- Architecture
- Objectives

Multilayer networks

- Perceptrons and one-layer networks, discussed in the preceding lectures, are seriously limited in their capabilities.
- Feedforward multilayer networks with non-linear node functions can overcome these limitations.
- MLPs can be used for many applications successfully.
- The perceptron learning mechanism cannot be used or extended easily for MLPs.
- More powerful supervised learning techniques for MLPs are presented in this lecture.
- The focus of this chapter is on a learning mechanism called error 'backpropagation' for MLPs.

Multilayer networks

- Backpropagation came into prominence in the late 1980's.
- An early version of backpropagation was first proposed by Rosenblatt in 1961.
- His proposal was crippled by the use of perceptrons that compute step functions of their net weighted inputs.
- For successful application of this method, differentiable node functions are required.
- The new algorithm was proposed by Werbos in 1974.
- Parker (1985) and LeCun (1985) rediscovered it, but its modern specification was provided and popularized by Rumelhart, Hinton, and Williams (1986).

Multilayer networks

- Backpropagation is similar to the LMS (least mean squared error) learning algorithm described earlier.
- LMS is based on gradient descent: weights are modified in a direction corresponds to the negative gradient of an error value.
- The choice of everywhere-differentiable node functions allows correct application of this method.
- LMS is straightforward and very similar to the Adaline.
- The major advance of backpropagation over the LMS algorithm is in expressing how an error can be propagated backwards to nodes at lower layers (inputs) of the network.

Multilayer networks

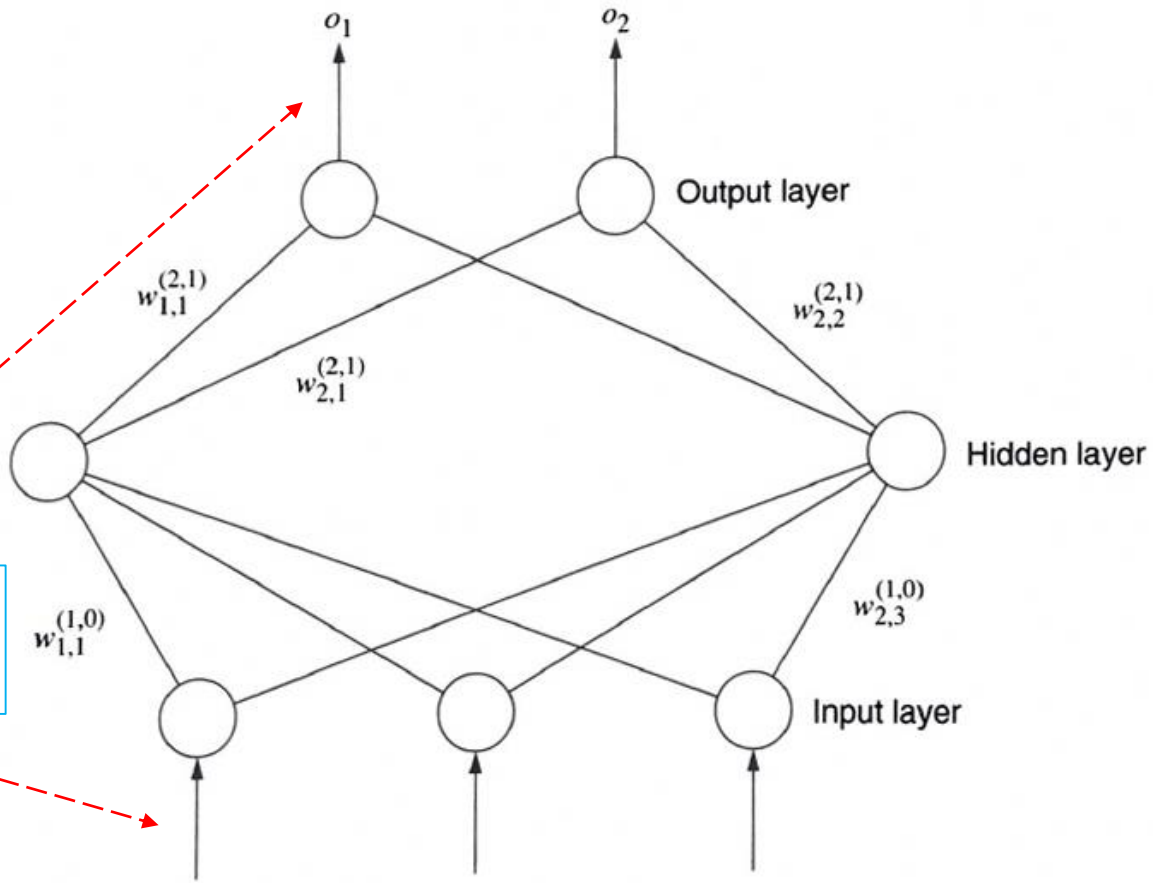
- The gradient of these backward-propagated error measures can then be used to determine the desired weight modifications for connections.
- The backpropagation algorithm has had a major impact on the field of neural networks.
- The backprop has been widely applied to a large number of problems in many disciplines.
- Backpropagation has been used for several kinds of applications including classification, function approximation, forecasting, …

Content

- Multilayer networks
- Multilevel discrimination
- Architecture
- Objectives

Multilevel discrimination

- A layered structure of nodes can be used to solve linearly nonseparable classification problems.
- MLPs may have more than one hidden layer.
- The MLP contains hidden nodes in a hidden layer.



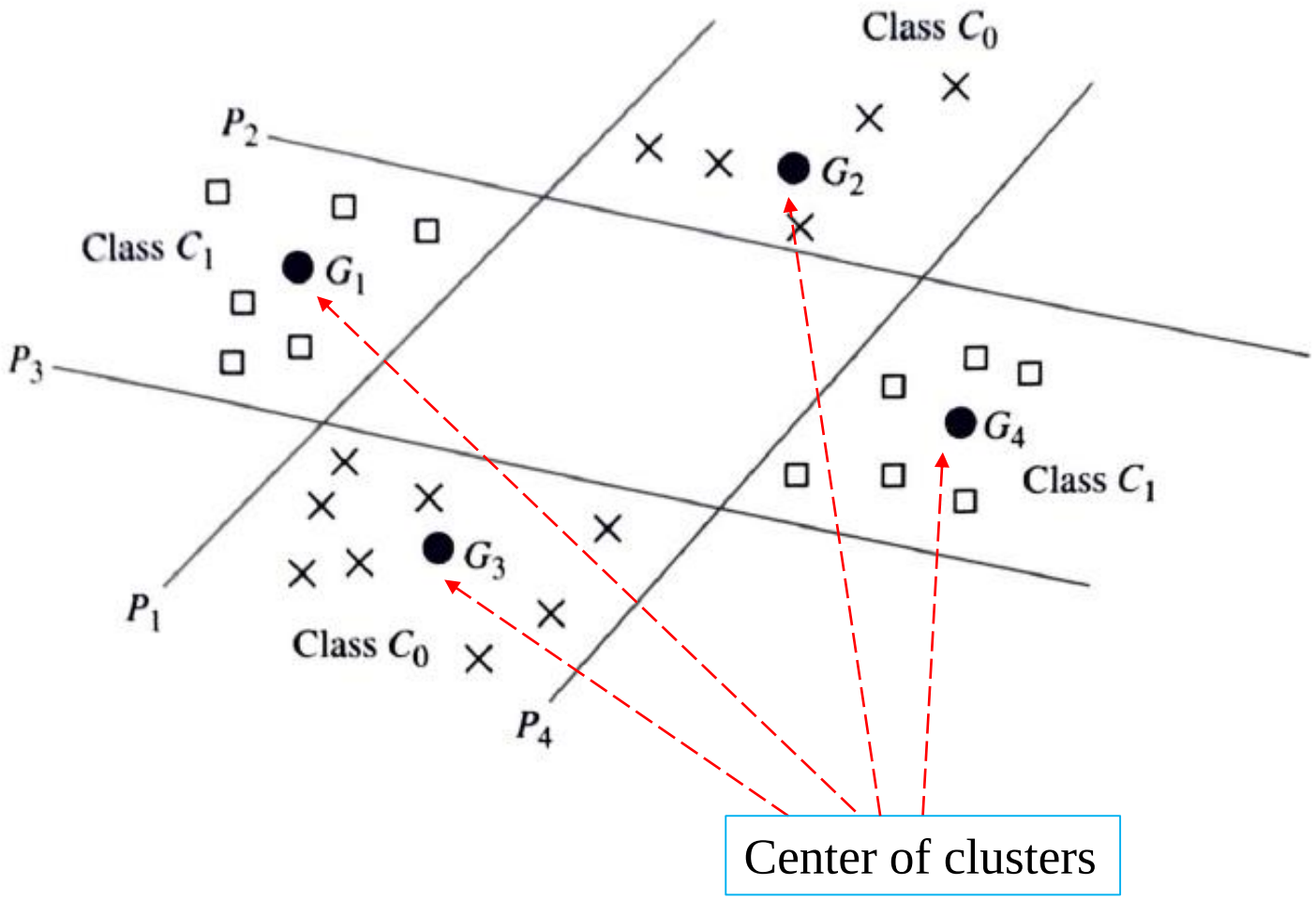
3 inputs, 2 outputs
1 hidden layer

Multilevel discrimination

- It is not easy to train such a network since the ideal weight change rule is more complex than the single layer nets.
- The network calculates an error value for some input sample.
- Which weights in the network must be modified?
- Are there any differences between changes depending on connections?
- What are the change of the weight values for each connection?
- How can we decide the value of changing for each connections?

Multilevel discrimination

- Two-class problem, which cannot be separated by a straight line, can be separated using multiple straight lines.



Content

- Multilayer networks
- Multilevel discrimination
- Architecture
- Objectives

Architecture

- The backpropagation algorithm assumes a feedforward neural network architecture.
- In this architecture, nodes are partitioned into layers numbered 0 to L.
- The lowermost layer is the input layer numbered as layer 0, and the topmost layer is the output layer numbered as layer L.
- Backpropagation addresses networks which $L > 2$.
- The hidden layers are numbered 1 to L - 1.

Architecture

- Hidden nodes do not directly receive inputs or send outputs to the external environment.
- The presentation of the algorithm also assumes that the network is strictly feedforward, only nodes in adjacent layers are directly connected.
- Input layer nodes transmit input values to the hidden layer nodes and do not perform any computation.
- The number of input nodes equals the dimensionality of input patterns.
- The number of nodes in the output layer is dictated by the problem.

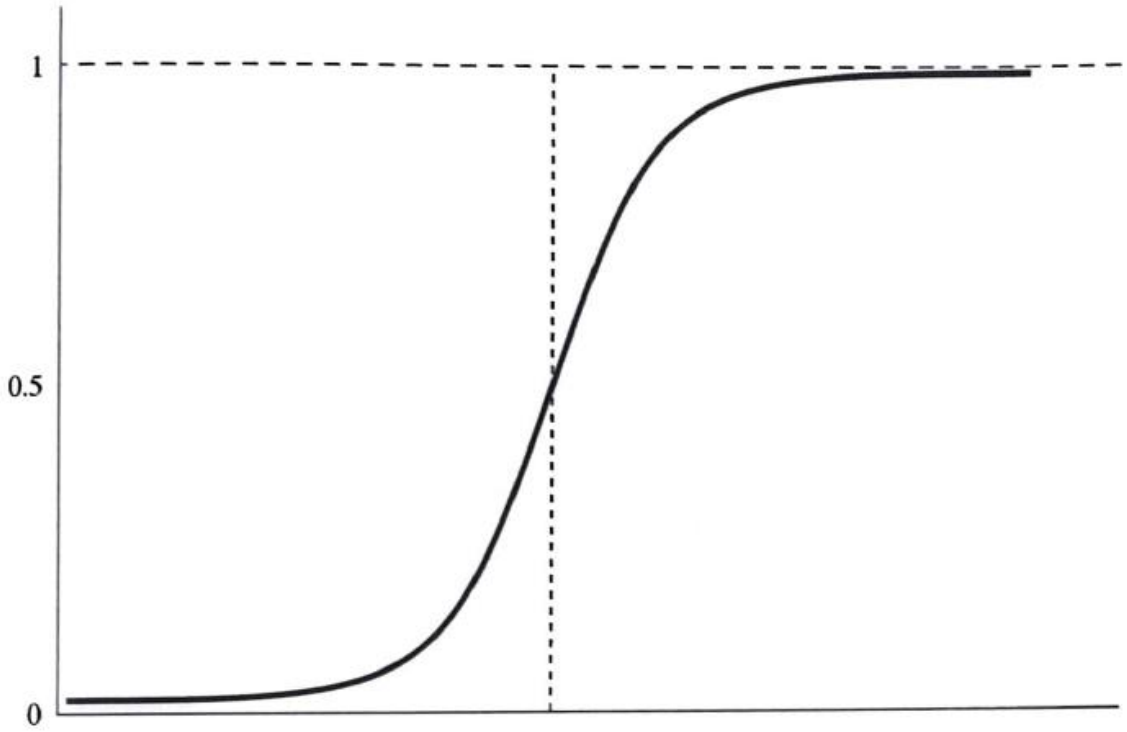
Architecture

- If the task is to approximate a function mapping n-dimensional input vectors to m-dimensional output vectors, the network contains n input nodes and m output nodes.
- An additional "dummy" input node with constant input (= 1) is also often used with a weight value.
- The number of nodes in the hidden layer is up to generally depends on problem complexity.
- Each hidden node and output node applies a sigmoid function to its net input.

Architecture

- The main reasons the use of the sigmoidal function are: continuous, monotonicaly increasing, invertible, everywhere differentiable, and asymptotically approaches its saturation values as $net \rightarrow \pm\infty$.

$$S(net) = \frac{1}{1 + e^{(-net)}}$$



Content

- Multilayer networks
- Multilevel discrimination
- Architecture
- Objectives

Objectives

- The algorithm is a supervised learning algorithm trained using $\boldsymbol{P}$ input patterns.
- For each input vector $\boldsymbol{x_p}$, we have the corresponding desired K-dimensional output vector;

$$\boldsymbol{d}_p = (d_{p,1}, d_{p,2}, \ldots, d_{p,K}) \qquad 1 \leq p \leq P$$

- Collection of input-output pairs constitutes the training set;

$$\{(x_p, d_p) : p = 1, \ldots, P\}$$

- The length of the input vector $\boldsymbol{x_p}$ is equal to the number of inputs of the application.
- The length of the output vector $\boldsymbol{d_p}$ is equal to the number of outputs of the application.

Objectives

- The training algorithm should work irrespective of the initial weight values.
- The goal of training is to modify the weights.
- The network's output vector $\boldsymbol{o}_p$ should be as close as possible to the desired output $\boldsymbol{d}_p$.

$$\boldsymbol{o}_p = (o_{p,1}, o_{p,2}, \ldots, o_{p,K})$$

- To achieve this goal, the cumulative error of the network should be minimized.

$$\text{Error} = \sum_{p=1}^{P} Err(\boldsymbol{o}_p, \boldsymbol{d}_p)$$

- There are many possible choices for the error function.

Objectives

- The error function may be cross-entropy function.

$$Err(\boldsymbol{o}_p, \boldsymbol{d}_p) = \sum_p (\boldsymbol{d}_p \log \boldsymbol{o}_p + (1 - \boldsymbol{d}_p) \log(1 - \boldsymbol{o}_p))$$

- The goal of this lecture will be to find weights that minimize Sum Square Error (SSE) or Mean Squared Error (MSE).

$$\text{Sum Square Error} = \sum_{p=1}^{P} \sum_{j=1}^{K} (\ell_{p,j})^2$$

$$MSE = \frac{1}{P} \sum_{p=1}^{P} \sum_{j=1}^{K} (\ell_{p,j})^2$$

$$\ell_{p,j} = |o_{p,j} - d_{p,j}|$$