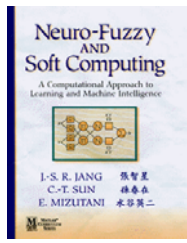
Chapter 8: Adaptive Networks

- Introduction (8.1)
- Architecture (8.2)
- Backpropagation for Feedforward Networks (8.3)



Jyh-Shing Roger Jang et al., *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, First Edition, Prentice Hall, 1997

Introduction (8.1)

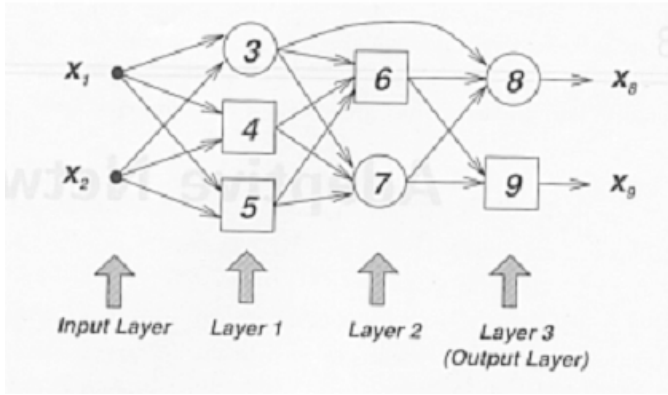
- Almost all kinds of Neural Networks paradigms with supervised learning capabilities are unified through adaptive networks
- Nodes are process units. Causal relationships between the connected nodes are expressed with links
- All or part of the nodes are adaptive, which means that the outputs of these nodes are driven by modifiable parameters pertaining to these nodes

Introduction (8.1) (cont.)

- A learning rule explains how these parameters (or weights) should be updated to minimize a predefined error measure
- The error measure computes the discrepancy between the network's actual output and a desired output
- The steepest descent method is used as a basic learning rule. It is also called backpropagation

Architecture (8.2)

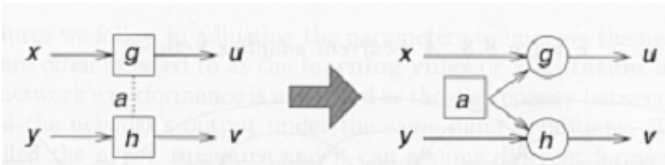
- Definition: An adaptive network is a network structure whose overall input-output behavior is determined by a collection of modifiable parameters
- The configuration of an adaptive network is composed of a set of nodes connected by direct links
- Each node performs a static node function on its incoming signals to generate a single node output
- Each link specifies the direction of signal flow from one node to another



A feedforward adaptive network in layered representation

Example 1 : Parameter sharing in adoptive network

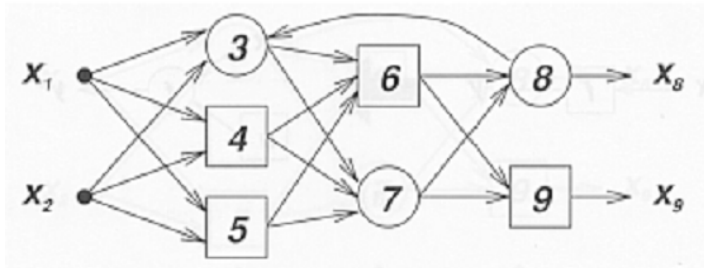
A single node



Parameter sharing problem

Architecture (8.2) cont.)

- There are 2 types of adoptive networks
  - Feedforward (acyclic directed graph such as in fig. 8.1)
  - Recurrent (contains at least one directed cycle)



A recurrent adaptive network

Architecture (8.2) (cont.)

- Static Mapping
  - A feedforward adaptive network is a static mapping between its inputs and output spaces
  - This mapping may be linear or highly nonlinear
  - Our goal is to construct a network for achieving a desired nonlinear mapping

Architecture (8.2) (cont.)

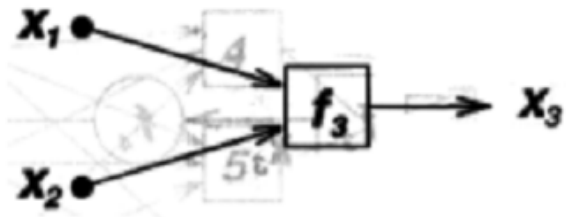
- Static Mapping (cont.)
  - This nonlinear mapping is regulated by a data set consisting of desired input-output pairs of a target system to be modeled: this data set is called training data set
  - The procedures that adjust the parameters to improve the network's performance are called the learning rules

Architecture (8.2) (cont.)

- Static Mapping (cont.)
  - Example 2: An adaptive network with a single linear node

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3$$

where: x_1, x_2 are inputs; a_1, a_2 and a_3 are modifiable parameters



A linear single-node adaptive network

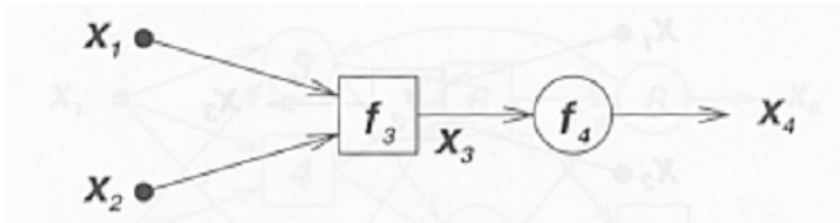
Architecture (8.2) (cont.)

- Static Mapping (cont.)
  - Identification of parameters can be performed through the linear least-squares estimation method of chapter 5
  - Example 3: Perception network

$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3$ and

$$x_4 = f_4(x_3) = \begin{cases} 1 & \text{if } x_3 \geq 0 \\ 0 & \text{if } x_3 < 0 \end{cases}$$

(f_4 is called a step function)



A nonlinear single-node adaptive network

Architecture (8.2) (cont.)

- The step function is discontinuous at one point (origine) and flat at all other points, it is not suitable for derivative based learning procedures ⇒ use of sigmoidal function
- Sigmoidal function has values between 0 and 1 and is expressed as:

$$x_4 = f_4(x_3) = \frac{1}{\left(1 + e^{-x_3}\right)}$$

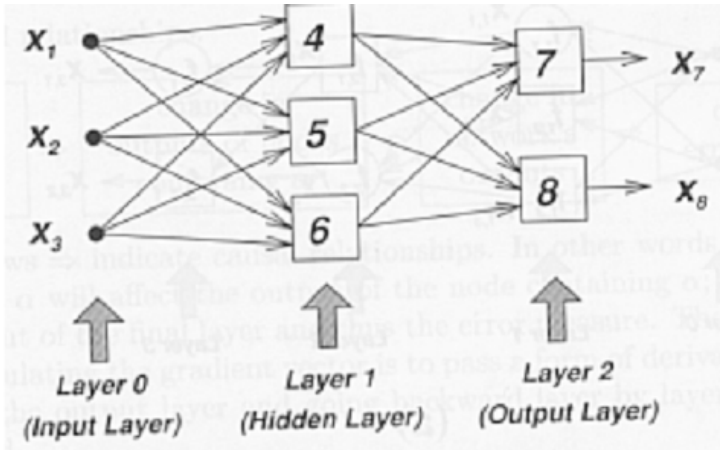
- Sigmoidal function is the building-block of the multi-layer perceptron

Architecture (8.2) (cont.)

- Example 4: A multilayer perceptron

$$x_7 = \frac{1}{1 + \exp\left[-\left(w_{4,7}x_4 + w_{5,7}x_5 + w_{6,7}x_6 + t_7\right)\right]}$$

where x_4, x_5 and x_6 are outputs from nodes 4, 5 and 6 respectively and the set of parameters of node 7 is $\{w_{4,7}, w_{5,7}, w_{6,7}, t_7\}$



A 3-3-2 neural network

Backpropagation for Feedforward Networks (8.3)

- Basic learning rule for adaptive networks
- It is a steepest descent-based method discussed in chapter 6
- It is a recursive computation of the gradient vector in which each element is the derivative of an error measure with respect to a parameter
- The procedure of finding a gradient vector in a network structure is referred to as backpropagation because the gradient vector is calculated in the direction opposite to the flow of the output of each node

Backpropagation for Feedforward Networks (8.3) (cont.)

- Once the gradient is computed, regression techniques are used to update parameters (weights, links)
- Notations:

  Assume we have L layers ($l = 0, 1, \ldots, L - 1$)

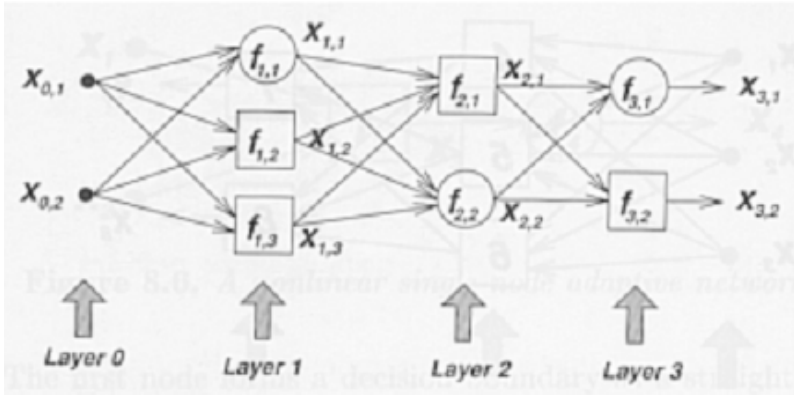
  - $N(l)$ represents the number of nodes in layer l
  - $x_{l,I}$ represents the output of node i in layer l ($i = 1, \ldots, N(l)$)
  - $f_{l,i}$ represents the function of node i

Backpropagation for Feedforward Networks (8.3) (cont.)

- Principle
  - Since the output of a node depends on the incoming signals and the parameter set of the node, we can write:

$$x_{l,i} = f_{l,i}(x_{l-1,1}, x_{l-1,2}, \ldots, x_{l-1,N(l-1)}, \alpha, \beta, \gamma, \ldots)$$

where α, β, γ, …, are the parameters of this node.



A layered representation

Backpropagation for Feedforward Networks (8.3) (cont.)

- Principle (cont.)
  - Let assume that the training set has P patterns, therefore we define an error for the pth pattern as:

$$\mathbf{E_p} = \sum_{\mathbf{k=1}}^{\mathbf{N(L)}} \left(\mathbf{d_k} - \mathbf{x_{L,k}}\right)^2$$

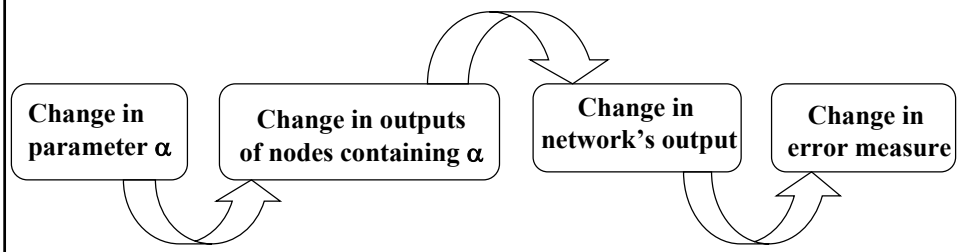
Where: d_k is the k-th component of the pth desired output vector and $x_{L,K}$ is the k-th component of the predicted output vector produced by presenting the pth input vector to the network

  - The task is to minimize an overall measure defined as:

$$\mathbf{E} = \sum_{\mathbf{p=1}}^{\mathbf{P}} \mathbf{E_p}$$

Backpropagation for Feedforward Networks (8.3) (cont.)

- Principle (cont.)
  - To use the steepest descent to minimize E, we have to compute ∇E (gradient vector)
  - Causal Relationships



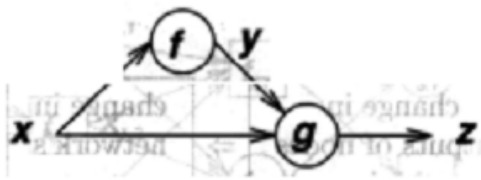
Backpropagation for Feedforward Networks (8.3) (cont.)

- Principle (cont.)
  - Therefore, the basic concept in calculating the gradient vector is to pass a form of derivative information starting from the output layer and going backward layer by layer until the input layer is attained
  - Let's define $\varepsilon_{l,i} = \frac{\partial^+ E_p}{\partial x_{l,i}}$ as the error signal on
  - the node i in layer l (ordered derivative)

Backpropagation for Feedforward Networks (8.3) (cont.)

- Principle (cont.)
  - Example: Ordered derivatives and ordering partial derivatives

    Consider the following adaptive network



Ordered derivatives & ordinary partial derivatives

$$\begin{cases} z = g(x, y) \\ y = f(x) \end{cases}$$

Backpropagation for Feedforward Networks (8.3) (cont.)

– For the ordinary partial derivative,

$$\frac{\partial z}{\partial x} = \frac{\partial g(x,y)}{\partial x}$$

x and y are assumed independent without paying attention that y = f(x)

– For the ordered derivative, we take the indirect causal relationship into account,

$$\frac{\partial^+ z}{\partial x} = \frac{\partial g(x,f(x))}{\partial x} = \frac{\partial g(x,y)}{\partial x}\Big|_{y=f(x)} + \frac{\partial g(x,y)}{\partial y}\Big|_{y=f(x)} * \frac{\partial f}{\partial x}$$

– The gradient vector is defined as the derivative of the error measure with respect to the parameter variables. If α is a parameter of the ith node at layer l, we can write:

$$\frac{\partial^+ E_p}{\partial \alpha} = \frac{\partial^+ E_p}{\partial x_{l,i}} * \frac{\partial f_{l,i}}{\partial \alpha} = \varepsilon_{l,i} \frac{\partial f_{l,i}}{\partial \alpha}$$

Where $\varepsilon_{l,i}$ is computed through The indirect causal relationship

Backpropagation for Feedforward Networks (8.3) (cont.)

– The derivative of the overall error measure E with respect to α is:

$$\frac{\partial^+ E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial^+ E_p}{\partial \alpha}$$

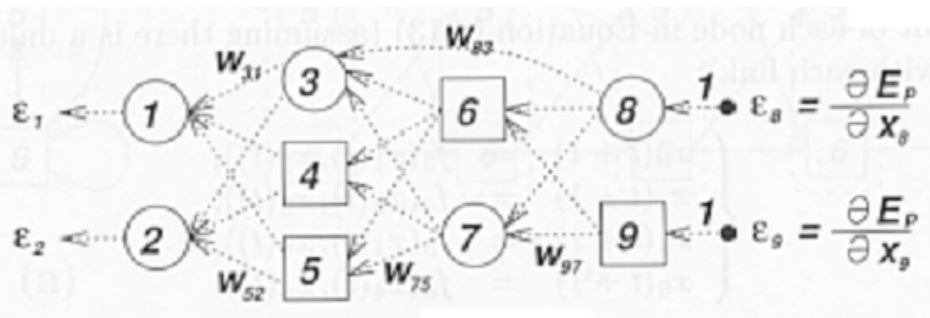
– Using a steepest descent scheme, the update for α is:

$$\alpha_{next} = \alpha_{now} - \eta \frac{\partial^+ E}{\partial \alpha}$$

Backpropagation for Feedforward Networks (cont.)

– Example 8.6 Adaptive network and its error propagation model

In order to calculate the error signals at internal nodes, an error-propagation network is built



Error propagation model

Backpropagation for Feedforward Networks (8.3) (cont.)

$$\varepsilon_9 = \frac{\partial^+ E_p}{\partial x_9} = \frac{\partial E_p}{\partial x_9} = -2(d_9 - x_9)$$

$$\varepsilon_8 = \frac{\partial^+ E_p}{\partial x_8} = \frac{\partial E_p}{\partial x_8} = -2(d_8 - x_8)$$

$$\varepsilon_7 = \frac{\partial^+ E_p}{\partial x_7} = \frac{\partial^+ E_p}{\partial x_8} * \frac{\partial f_8}{\partial x_7} + \frac{\partial^+ E_p}{\partial x_9} * \frac{\partial f_9}{\partial x_7} = \varepsilon_8 \underbrace{\frac{\partial f_8}{\partial x_7}}_{w_{8,7}} + \varepsilon_9 \underbrace{\frac{\partial f_9}{\partial x_7}}_{w_{9,7}}$$

Backpropagation for Feedforward Networks (8.3) (cont.)

– There are 2 types of learning algorithms:

- Batch learning (off-line learning): the update for α takes place after the whole training data set has been presented (one epoch)
- On-line learning (pattern-by-pattern learning): the parameters α are updated immediately after each input output pair has been presented