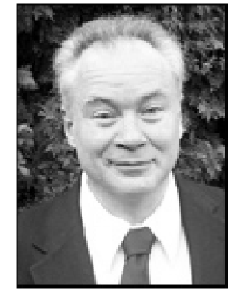
CSC 302 1.5 Neural Networks

Backpropagation

Entrance

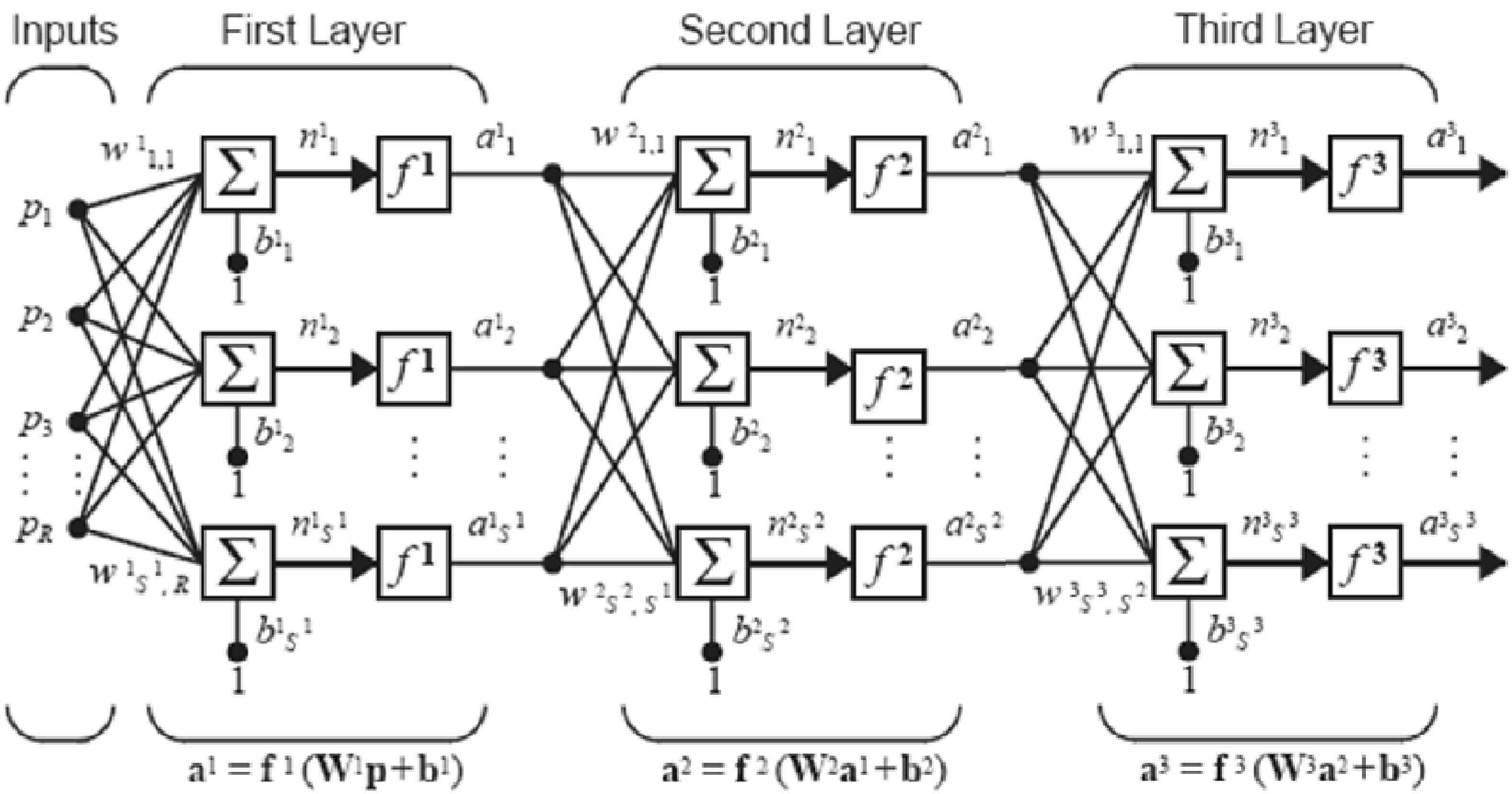
- Continuation of performance learning for multilayer neural networks.
- Introduction of backpropagation.
- Demonstrate how to use chain rule to calculate the derivative of the mean square error of a multilayer neural network.

History



- First algorithm to train a multilayer networks was contained in the thesis of Paul Werbos in 1974.
- But it was not disseminated in the neural network community.
- Rediscovered independently by
  - David Rumelhart, Geoffrey Hinton, and Ronald Williams [1986]
  - David Parker [1985]
  - Yann Le Cun [1985]

Multilayer Perceptrons



$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Notations

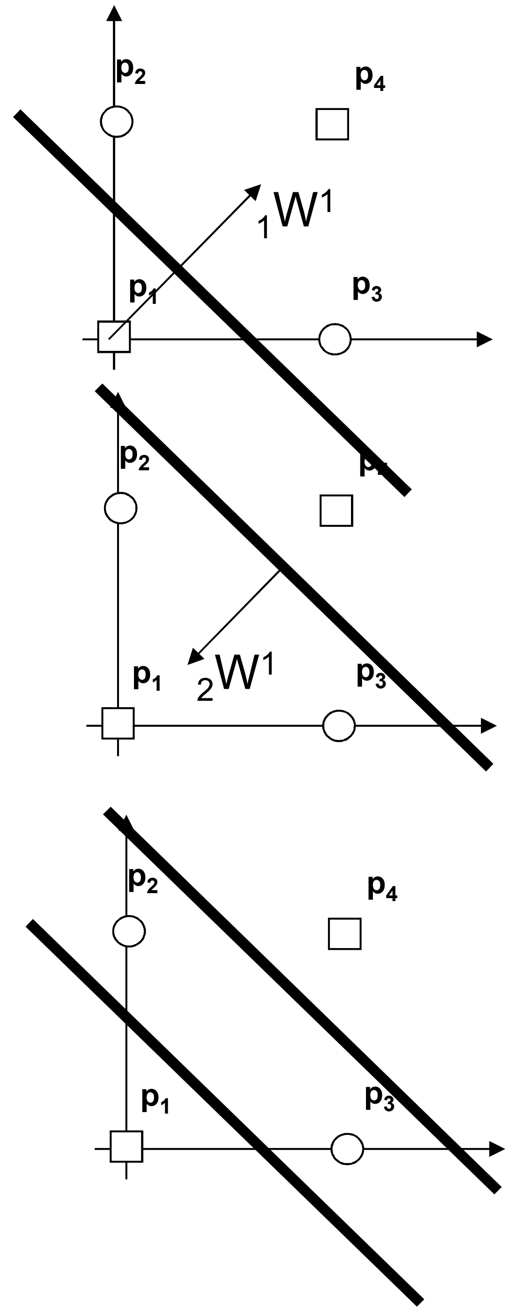
- We use superscripts to identify the layer number.
  - E.g. Weight matrix for first layer W^1
- Shorthand notation for multilayer network
  - R-S^1-S^2-S^3 network
  - Number of inputs followed by number of neurons in each layer.

Pattern Classification

To demonstrate the capabilities of multilayer, lets consider the exclusive-or (XOR) problem.

Exclusive OR (XOR)

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\}$$



Two-layer network can solve the XOR problem.

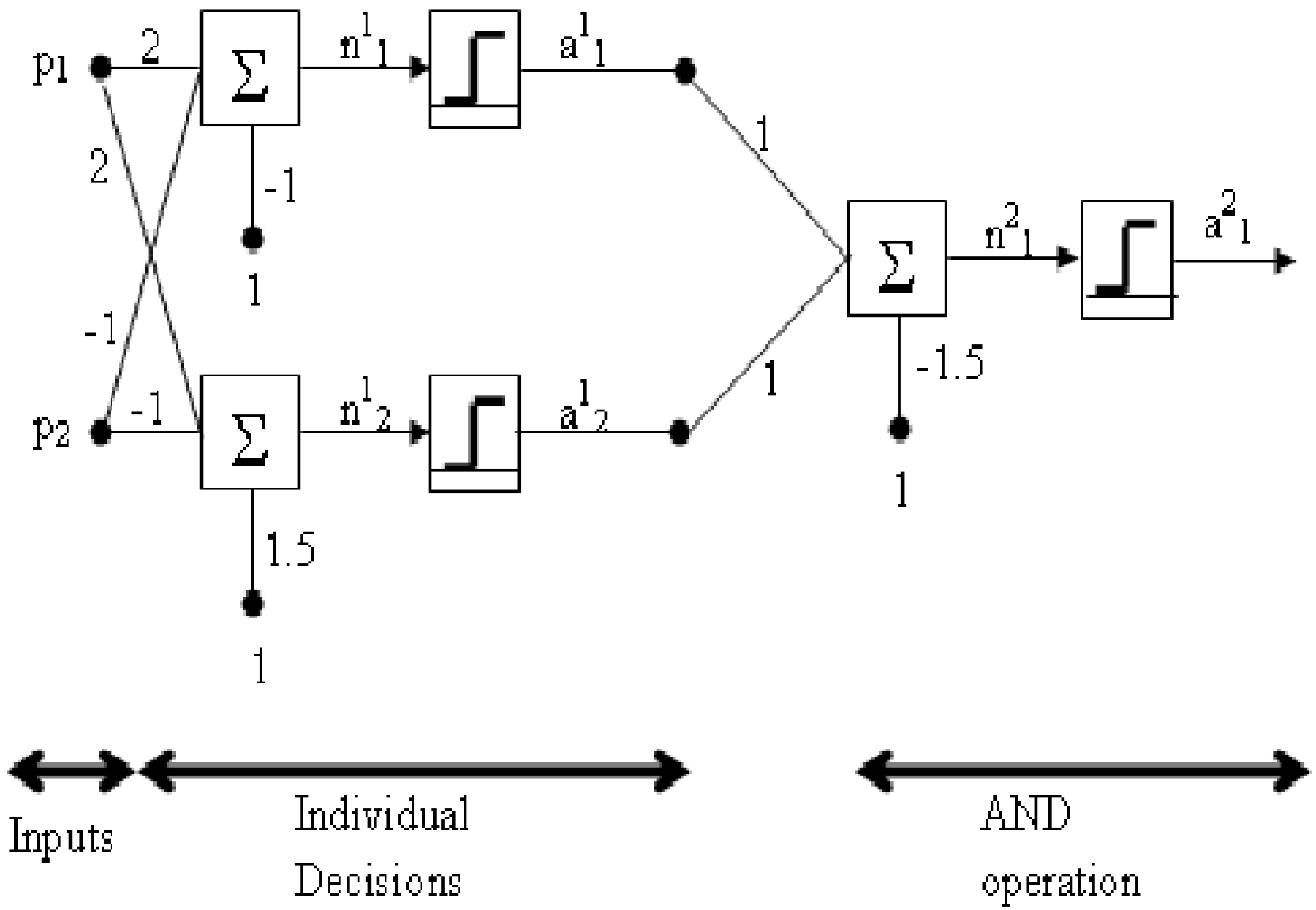
Use two neurons in the first layer to create two decision boundaries.

The first boundary separates $\mathbf{p}_1$ from the other patterns.

The second boundary separates $\mathbf{p}_4$ from the other patterns

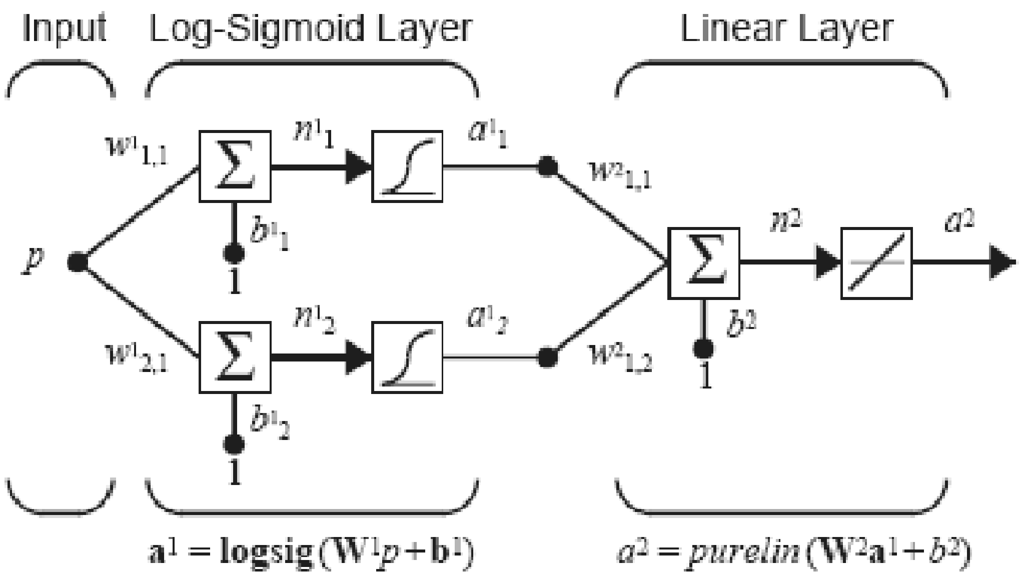
The second layer combines the two boundaries together using an **AND** operation.

Two-Layer XOR Network



2-2-1 Network

Function Approximation



$$f^1(n) = \frac{1}{1 + e^{-n}}$$

(log-sigmoid)

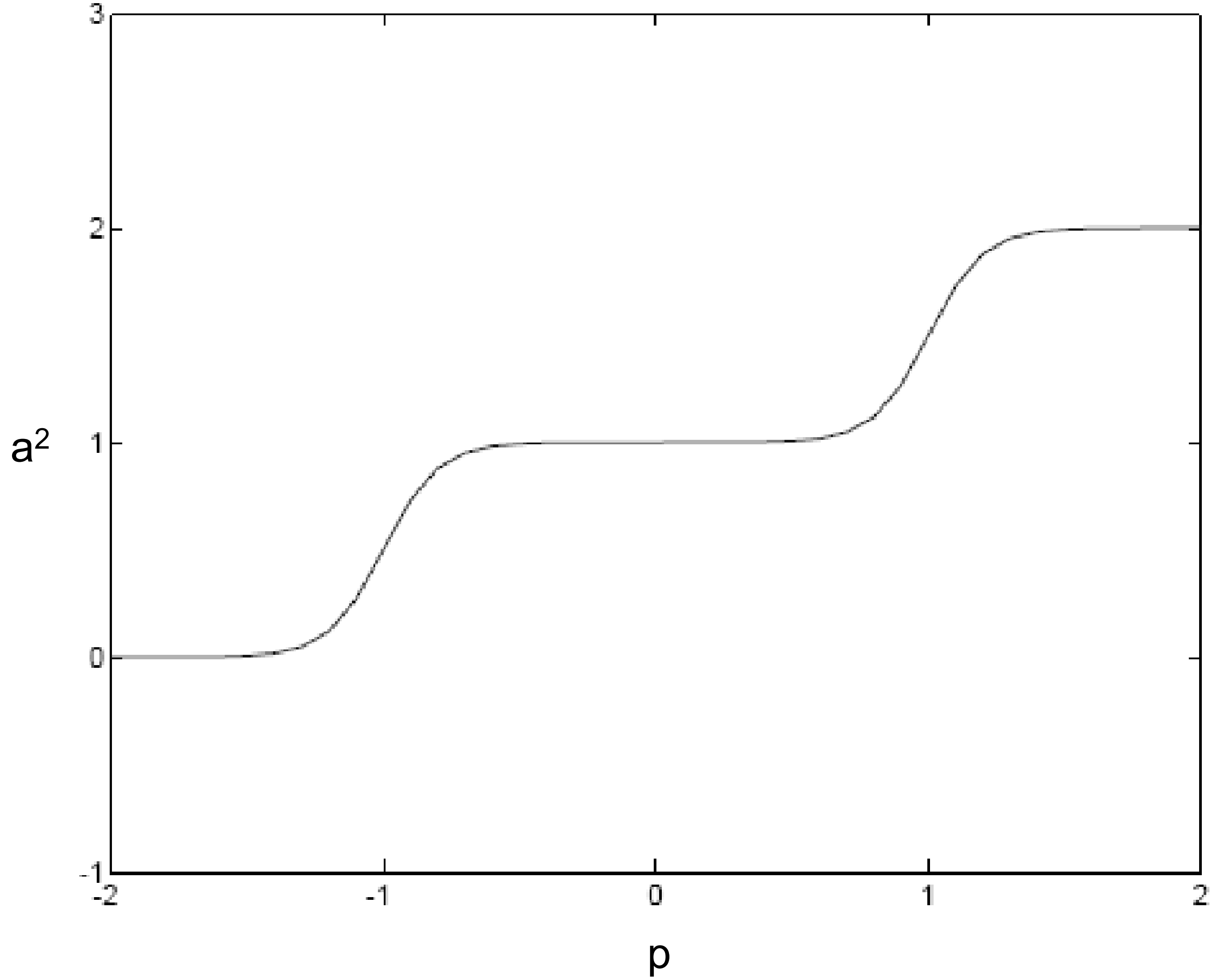
$$f^2(n) = n$$ (linear)

Nominal Parameter Values

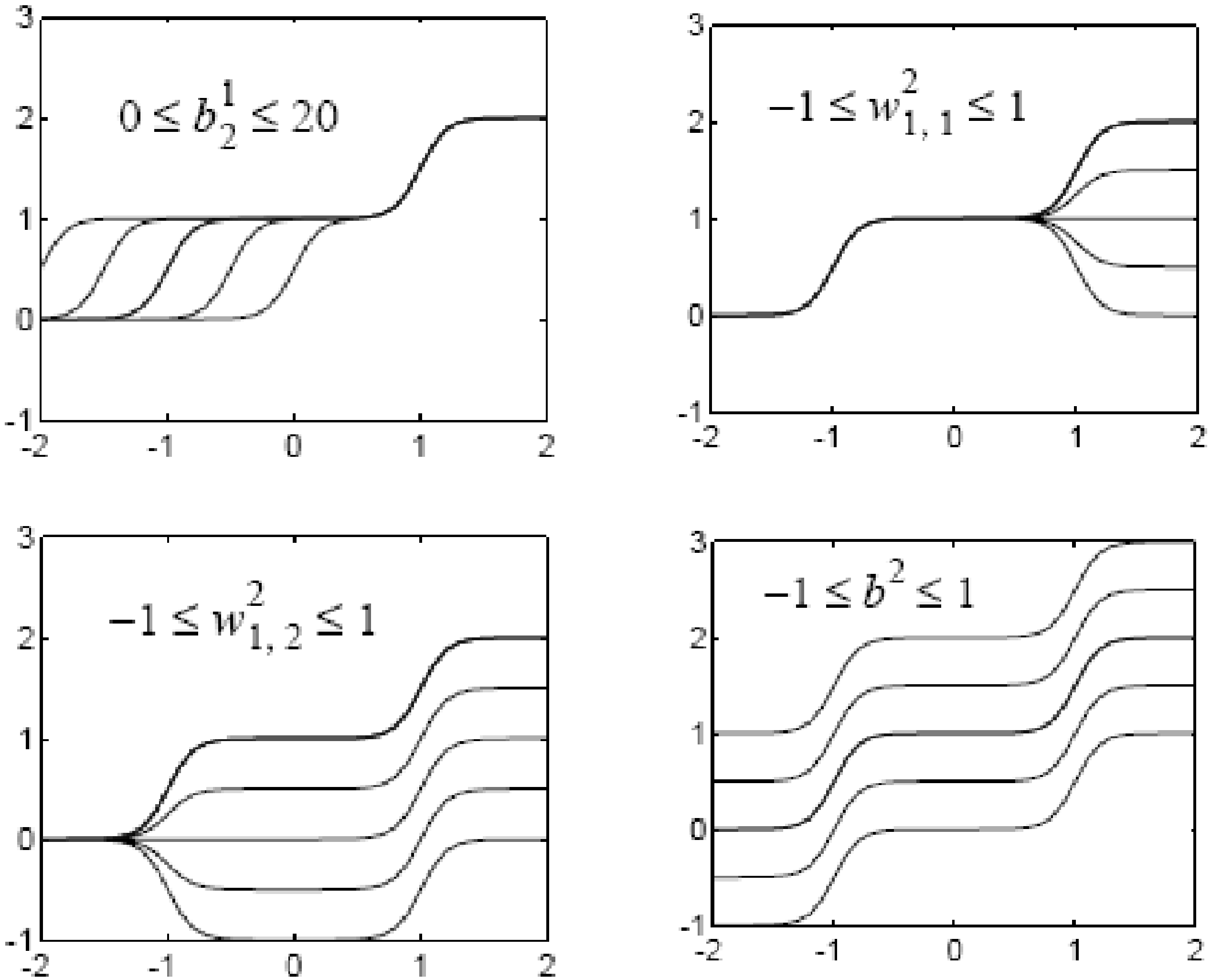
$$w^1_{1,1} = 10 \qquad w^1_{2,1} = 10 \qquad b^1_1 = -10 \qquad b^1_2 = 10$$

$$w^2_{1,1} = 1 \qquad w^2_{1,2} = 1 \qquad b^2 = 0$$

Nominal Response



Parameter Variations



It can approximate almost any function, if we had a sufficient number of neurons in the hidden layer.

It has been shown that two-layer networks, with sigmoid transfer functions in the hidden layer and linear transfer function in the output layer, can approximate virtually any function provided sufficiently many neurons in the hidden layer.

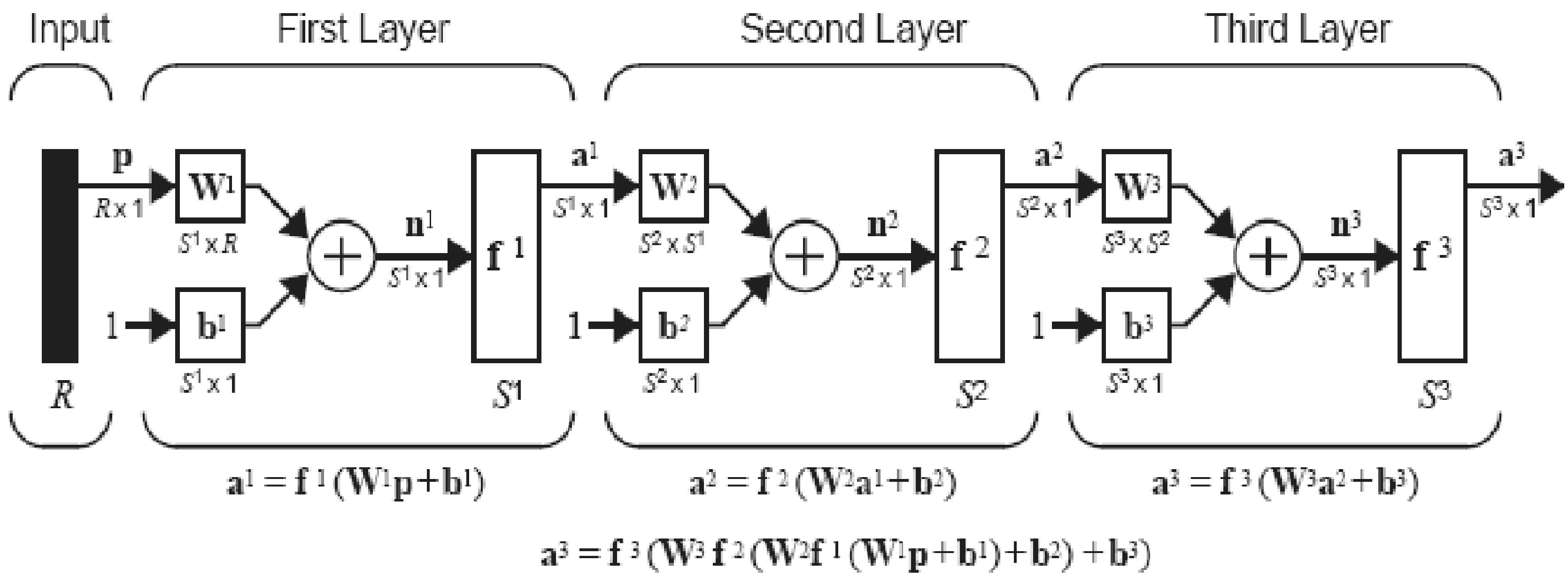
What is next?

Now we have seen the power of multilayer perceptron networks for pattern recognition and function approximation.

Next step …???

> Develop an algorithm to train multilayer neural networks.

The Backpropagation (BP) Algorithm



$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

M – no of layers

$$\mathbf{a}^{0} = \mathbf{p} \qquad \mathbf{a} = \mathbf{a}^{M}$$

Performance Index

Training Set

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

The BP algorithm is a generalization of the LMS algorithm.

Mean Square Error

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2]$$

Vector Case

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})]$$

Approximate Mean Square Error (Single Sample)

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k)$$

$$x_{k+1} = x_k - \alpha\hat{\nabla}F(x)$$

Approximate Steepest Descent

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m} \qquad b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m}$$

Chain Rule

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw}$$

Example

$$f(n) = \cos(n) \qquad n = e^{2w} \qquad f(n(w)) = \cos(e^{2w})$$

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (-\sin(n))(2e^{2w}) = (-\sin(e^{2w}))(2e^{2w})$$

Application to Gradient Calculation

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^{m}} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

Gradient Calculation

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \longrightarrow \frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} \qquad \frac{\partial n_i^m}{\partial b_i^m} = 1$$

Sensitivity

Let's define $s_i^m \equiv \dfrac{\partial \hat{F}}{\partial n_i^m}$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

$\Downarrow$ Gradient $\Downarrow$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = s_i^m$$

Steepest Descent (1)

Approximate Steepest Descent

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m} \qquad b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m}$$

Gradient

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \qquad \frac{\partial \hat{F}}{\partial b_i^m} = s_i^m$$

$$\boxed{w_{i,j}^m(k+1) = w_{i,j}^m(\mathrm{k}) - \alpha s_i^m a_j^{m-1} \qquad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m}$$

where $s_i^m \equiv \dfrac{\partial \hat{F}}{\partial n_i^m}$

Steepest Descent (2)

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha s_i^m a_j^{m-1} \qquad b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$$

$$\boxed{\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m}$$

where $$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

s^m is the number of neurons in the m^{th} layer

Next Step: Compute the Sensitivities (Backpropagation)

Jacobean Matrix

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m(n_j^m)$$

where $\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{n}^m)$$

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix}$$

Backpropogation (Sensitivities)

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$\boxed{\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}}$$

The sensitivities are computed by starting at the last layer, and then propagating backwards through the network to the first layer.

$$\mathbf{s}^M \to \mathbf{s}^{M-1} \to \ldots \to \mathbf{s}^2 \to \mathbf{s}^1$$

Initialization (Last Layer)

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M}$$

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M)$$

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M)$$

$$\boxed{\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a})}$$

Summary

Forward Propagation

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad m = 0, 2, \ldots, M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

Backpropagation

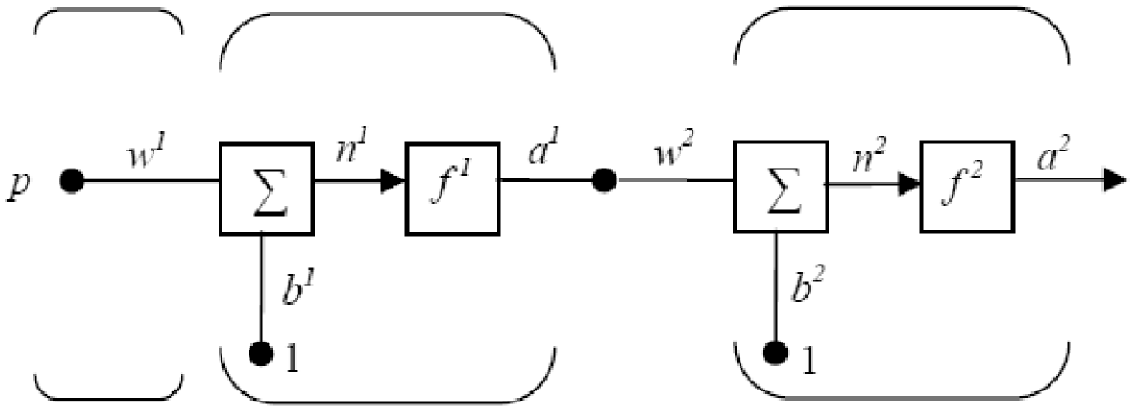
$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \qquad m = M-1, \ldots, 2, 1$$

Weight Update

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \qquad \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

Problem (Exam 2008)



$a^1 = f^1(w^1 p + b^1)$ $a^2 = f^2(w^2 p + b^2)$

The performance index

$$F = e^2 = (t - a)^2$$

a) Show that

$$\frac{\partial F}{\partial w^1} = ps^1, \quad \frac{\partial F}{\partial w^2} = a^1 s^2 \text{ and } \frac{\partial F}{\partial b^m} = s^m \quad (m = 1, 2)$$

where $s^m = \dfrac{\partial F}{\partial n^m}$ $(m = 1, 2)$.

Derivatives

Applying the Chain Rule

$$\frac{\partial F}{\partial w^1} = \frac{\partial n^1}{\partial w^1} \times \frac{\partial F}{\partial n^1} \qquad \text{where} \qquad n^1 = w^1 p + b^1$$

$$\frac{\partial n^1}{\partial w^1} = p \quad \Rightarrow \quad \frac{\partial F}{\partial w^1} = p \times \frac{\partial F}{\partial n^1} \quad \Rightarrow \quad \boxed{\frac{\partial F}{\partial w^1} = ps^1 \text{ where } s^1 = \frac{\partial F}{\partial n^1}}$$

Similarly applying the Chain Rule, we can obtain

$$\frac{\partial F}{\partial w^2} = a^1 s^2 \text{ and } \frac{\partial F}{\partial b^m} = s^m \ (m = 1, 2) \qquad \text{where } s^m = \frac{\partial F}{\partial n^m} \ (m = 1, 2)$$

Sensitivities

b) Show that

$$s^2 = -2(t-a)\frac{\partial f^2(n^2)}{\partial n^2} \text{ and } s^1 = w^2 s^2 \frac{\partial f^1(n^1)}{\partial n^1}.$$

$$s^2 = \frac{\partial F}{\partial n^2} = \frac{\partial (t-a)^2}{\partial n^2} = \frac{\partial (t-a^2)^2}{\partial n^2} = -2(t-a^2)\frac{\partial a^2}{\partial n^2}$$

$$\boxed{= -2(t-a)\frac{\partial f^2(n^2)}{\partial n^2}} \quad (since\ a^2 = a\ and\ a^2 = f^2(n^2))$$

$$s^1 = \frac{\partial F}{\partial n^1} = \frac{\partial n^2}{\partial n^1} . \frac{\partial F}{\partial n^2} = \boxed{\frac{\partial n^2}{\partial n^1}} . s^2$$

$$\frac{\partial n^2}{\partial n^1} = \frac{\partial}{\partial n^1}(w^2 a^1 + b^2) = w^2 \frac{\partial a^1}{\partial n^1}$$

$$\frac{\partial n^2}{\partial n^1} = w^2 \frac{\partial f^1(n^1)}{\partial n^1}$$

$$\boxed{s^1 = w^2 s^2 \frac{\partial f^1(n^1)}{\partial n^1}}$$

New weights and bias

From the Steepest Descent Algorithm

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha \nabla \mathbf{F}\big|_{\mathbf{x}=\mathbf{x_k}}$$

$$w^1(k+1) = w^1(k) - \alpha \frac{\partial F}{\partial w^1} = w^1(k) - \alpha p s^1$$

$$w^2(k+1) = w^2(k) - \alpha \frac{\partial F}{\partial w^2} = w^2(k) - \alpha a^1 s^2$$

$$b^m(k+1) = b^m(k) - \alpha \frac{\partial F}{\partial b^m} \, (m = 1, 2)$$

$$= b^m(k) - \alpha s^m \, (m = 1, 2)$$

Performing the First Iteration

$$w^1(0) = 1,\ b^1(0) = -2,\ w^2(0) = 1\ and\ b^2(0) = 1.$$

$$f^1(n) = (n)^2, \quad f^2(n) = \frac{1}{n} \qquad \{p = 1, t = 1\}.$$

given

$$\frac{\partial f^1(n^1)}{\partial n^1} = 2(n^1)^2\ and\ \frac{\partial f^2(n^2)}{\partial n^2} = \frac{-1}{(n^2)^2}$$

Forward propagation

$$\begin{aligned} a^1 &= f^1(w^1 p + b^1) \\ &= f^1(1.1 - 2) \\ &= f^1(-1) \\ &= (-1)^2 \\ &= 1 \end{aligned}$$

$$\begin{aligned} a^2 &= f^2(w^2 a^1 + b^2) \\ &= f^2(1.1 + 1) \\ &= f^2(2) \\ &= 1/2 \\ &= 0.5 \end{aligned}$$

$$\begin{aligned} & t - a \\ &= t - a^2 \\ &= 1 - 0.5 \\ &= 0.5 \end{aligned}$$

$$\begin{aligned} \frac{\partial f^1(n^1)}{\partial n^1} &= 2(n^1)^2 \\ &= 2(-1)^2 \\ &= 2 \end{aligned}$$

$$\begin{aligned} \frac{\partial f^2(n^2)}{\partial n^2} &= \frac{-1}{(n^2)^2} \\ &= \frac{-1}{(2)^2} \\ &= -0.25 \end{aligned}$$

Performing the First Iteration

Sensitivities

$$
\begin{aligned}
&s^2 \\
&= -2(t-a)\frac{\partial f^2(n^2)}{\partial n^2} \\
&= -2(0.5)(-0.25) \\
&= 0.25
\end{aligned}
$$

$$
\begin{aligned}
&s^1 \\
&= w^2 s^2 \frac{\partial f^1(n^1)}{\partial n^1} \\
&= (1)(0.25)(2) \\
&= 0.5
\end{aligned}
$$

New values of weights and bias (α = 1)

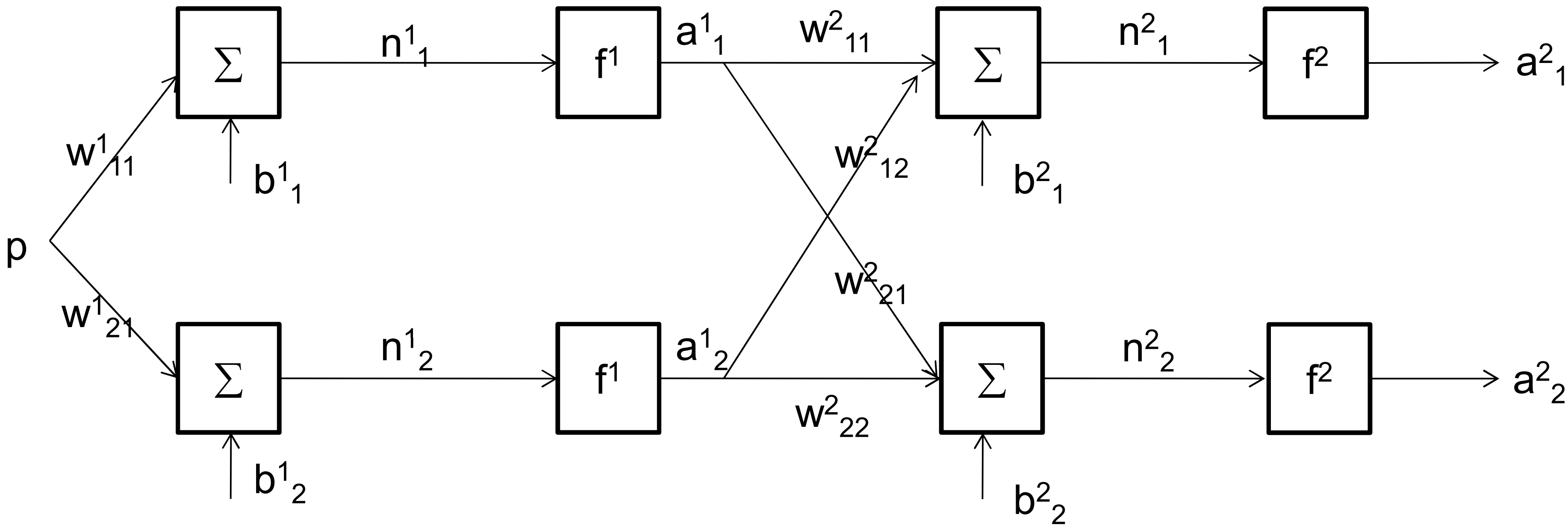
$$
\begin{aligned}
&w^1(1) \\
&= w^1(0) - \alpha p s^1 \\
&= 1 - 1.1.(0.5) \\
&= 1 - 0.5 \\
&= 0.5
\end{aligned}
$$

$$
\begin{aligned}
&w^2(1) \\
&= w^2(0) - \alpha a^1 s^2 \\
&= 1 - 1.1.(0.25) \\
&= 0.75
\end{aligned}
$$

$$
\begin{aligned}
&b^1(1) \\
&= b^1(0) - \alpha s^1 \\
&= -2 - 1.(0.5) \\
&= -2.5
\end{aligned}
$$

$$
\begin{aligned}
&b^2(1) \\
&= b^2(0) - \alpha s^2 \\
&= 1 - 1.(0.25) \\
&= 0.75
\end{aligned}
$$

1-2-2 Network



$$\mathbf{F} = (\mathbf{t} - \mathbf{a})^T(\mathbf{t} - \mathbf{a}) = \mathbf{e}^T\mathbf{e}$$

Derivatives

Steepest Descent Algorithm

$$\mathbf{W}^{\mathbf{1}}(k+1) = \mathbf{W}^{\mathbf{1}}(k) - \alpha \frac{\partial \mathbf{F}}{\partial \mathbf{W}^{\mathbf{1}}} \qquad \mathbf{b}^{\mathbf{1}}(k+1) = \mathbf{b}^{\mathbf{1}}(k) - \alpha \frac{\partial \mathbf{F}}{\partial \mathbf{b}^{\mathbf{1}}}$$

$$\mathbf{W}^{\mathbf{2}}(k+1) = \mathbf{W}^{\mathbf{2}}(k) - \alpha \frac{\partial \mathbf{F}}{\partial \mathbf{W}^{\mathbf{2}}} \qquad \mathbf{b}^{\mathbf{2}}(k+1) = \mathbf{b}^{\mathbf{2}}(k) - \alpha \frac{\partial \mathbf{F}}{\partial \mathbf{b}^{\mathbf{2}}}$$

We need to find the following derivatives

$$\frac{\partial \mathbf{F}}{\partial \mathbf{W}^{\mathbf{1}}} = ?, \ \frac{\partial \mathbf{F}}{\partial \mathbf{b}^{\mathbf{1}}} = ?, \quad \frac{\partial \mathbf{F}}{\partial \mathbf{W}^{\mathbf{2}}} = ?, \ and \ \frac{\partial \mathbf{F}}{\partial \mathbf{b}^{\mathbf{2}}} = ?$$

Derivatives …

$$\frac{\partial \mathbf{F}}{\partial \mathbf{W^1}} = \frac{\partial \mathbf{F}}{\partial \mathbf{n^1}}.\left[\frac{\partial \mathbf{n^1}}{\partial \mathbf{W^1}}\right]^T$$

$$= \mathbf{s^1}.\left[\frac{\partial[\mathbf{W^1 a^0} + \mathbf{b^1}]}{\partial \mathbf{W^1}}\right]^T$$

$$= \mathbf{s^1}.[\mathbf{a^0}]^T$$

$$\boxed{= p\mathbf{s^1}}$$

$$\frac{\partial \mathbf{F}}{\partial \mathbf{W^2}} = \frac{\partial \mathbf{F}}{\partial \mathbf{n^2}}.\left[\frac{\partial \mathbf{n^2}}{\partial \mathbf{W^2}}\right]^T$$

$$= \mathbf{s^2}.\left[\frac{\partial[\mathbf{W^1 a^1} + \mathbf{b^2}]}{\partial \mathbf{W^2}}\right]^T$$

$$\boxed{= \mathbf{s^2}.[\mathbf{a^1}]^T}$$

Similarly

$$\boxed{\frac{\partial \mathbf{F}}{\partial \mathbf{b^m}} = \mathbf{s^m}\ (m = 1, 2)}$$

where

$$\mathbf{s^1} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial \mathbf{n^1_1}} \\ \frac{\partial \mathbf{F}}{\partial \mathbf{n^1_2}} \end{bmatrix} \qquad \mathbf{s^2} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial \mathbf{n^2_1}} \\ \frac{\partial \mathbf{F}}{\partial \mathbf{n^2_2}} \end{bmatrix}$$

Sensitivities

$$\mathbf{s}^2 = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial \mathbf{n}_1^2} \\ \frac{\partial \mathbf{F}}{\partial \mathbf{n}_2^2} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{F} &= (\mathbf{t} - \mathbf{a})^T.(\mathbf{t} - \mathbf{a}) \\
&= [(t_1 - a_1^2) \;\; (t_2 - a_2^2)] \begin{bmatrix} (t_1 - a_1^2) \\ (t_2 - a_2^2) \end{bmatrix} \\
&= (t_1 - a_1^2)^2 + (t_2 - a_2^2)^2 \\
&= [t_1 - f^2(n_1^2)]^2 + [t_2 - f^2(n_2^2)]^2
\end{aligned}$$

$$\boxed{\frac{\partial \mathbf{F}}{\partial n_1^2} = -2(t_1 - a_1^2)\frac{\partial f^2(n_1^2)}{\partial n_1^2}}$$

$$\boxed{\frac{\partial \mathbf{F}}{\partial n_2^2} = -2(t_2 - a_2^2)\frac{\partial f^2(n_2^2)}{\partial n_2^2}}$$

Sensitivities …

$$\mathbf{s}^1 = \frac{\partial \mathbf{F}}{\partial \mathbf{n}^1} = \left[\frac{\partial \mathbf{n}^2}{\partial \mathbf{n}^1}\right]^T . \frac{\partial \mathbf{F}}{\partial \mathbf{n}^2}$$

$$= \left[\frac{\partial \mathbf{n}^2}{\partial \mathbf{n}^1}\right]^T . \mathbf{s}^2$$

where

$$\left[\frac{\partial \mathbf{n}^2}{\partial \mathbf{n}^1}\right]^T = \begin{bmatrix} \frac{\partial n_1^2}{\partial n_1^1} & \frac{\partial n_1^2}{\partial n_2^1} \\ \frac{\partial n_2^2}{\partial n_1^1} & \frac{\partial n_2^2}{\partial n_2^1} \end{bmatrix}$$

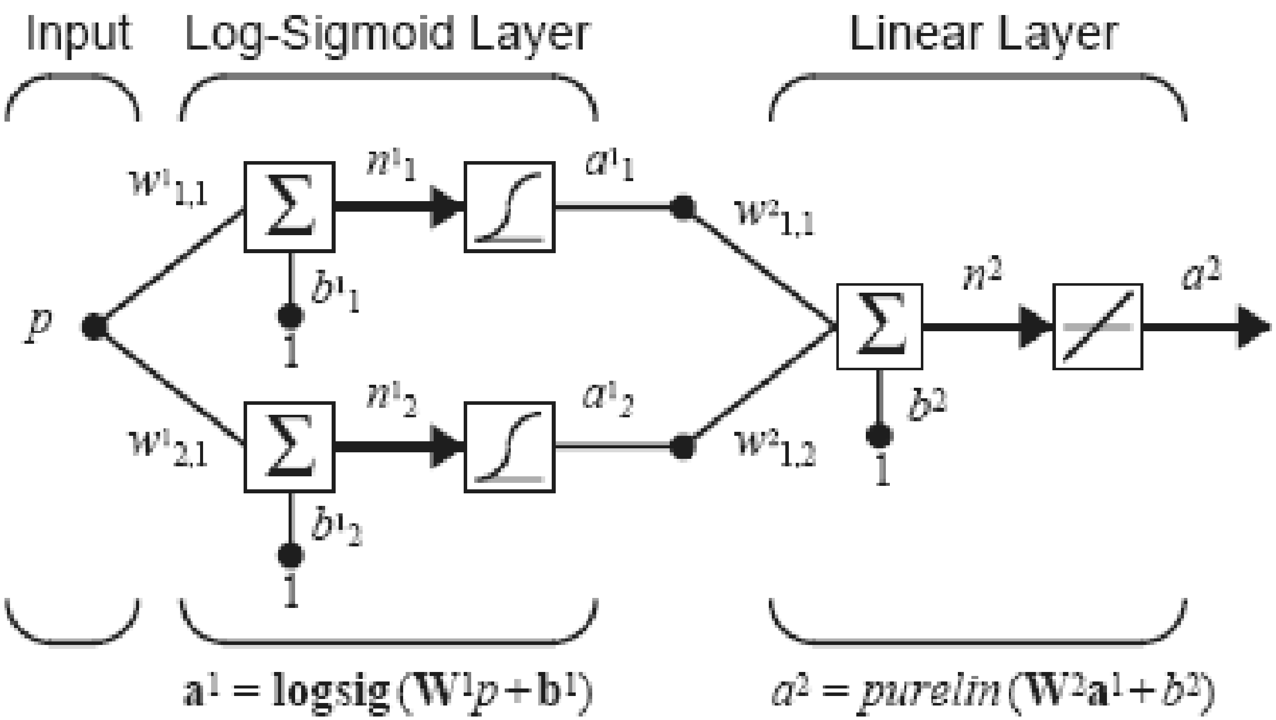
$$\frac{\partial n_1^2}{\partial n_1^1} = \frac{\partial[w_{11}^2 a_1^1 + w_{12}^2 a_2^1 + b_1^2]}{\partial n_1^1} = \frac{\partial[w_{11}^2 f^1(n_1^1) + w_{12}^2 f^1(n_2^1) + b_1^2]}{\partial n_1^1} \boxed{= w_{11}^2 \frac{\partial f^1(n_1^1)}{\partial n_1^1}}$$

Similarly

$$\boxed{\frac{\partial n_1^2}{\partial n_2^1} = w_{12}^2 \frac{\partial f^1(n_2^1)}{\partial n_2^1}} \qquad \boxed{\frac{\partial n_2^2}{\partial n_1^1} = w_{21}^2 \frac{\partial f^1(n_1^1)}{\partial n_1^1}} \qquad \boxed{\frac{\partial n_2^2}{\partial n_2^1} = w_{22}^2 \frac{\partial f^1(n_2^1)}{\partial n_2^1}}$$

Example – Function Approximation

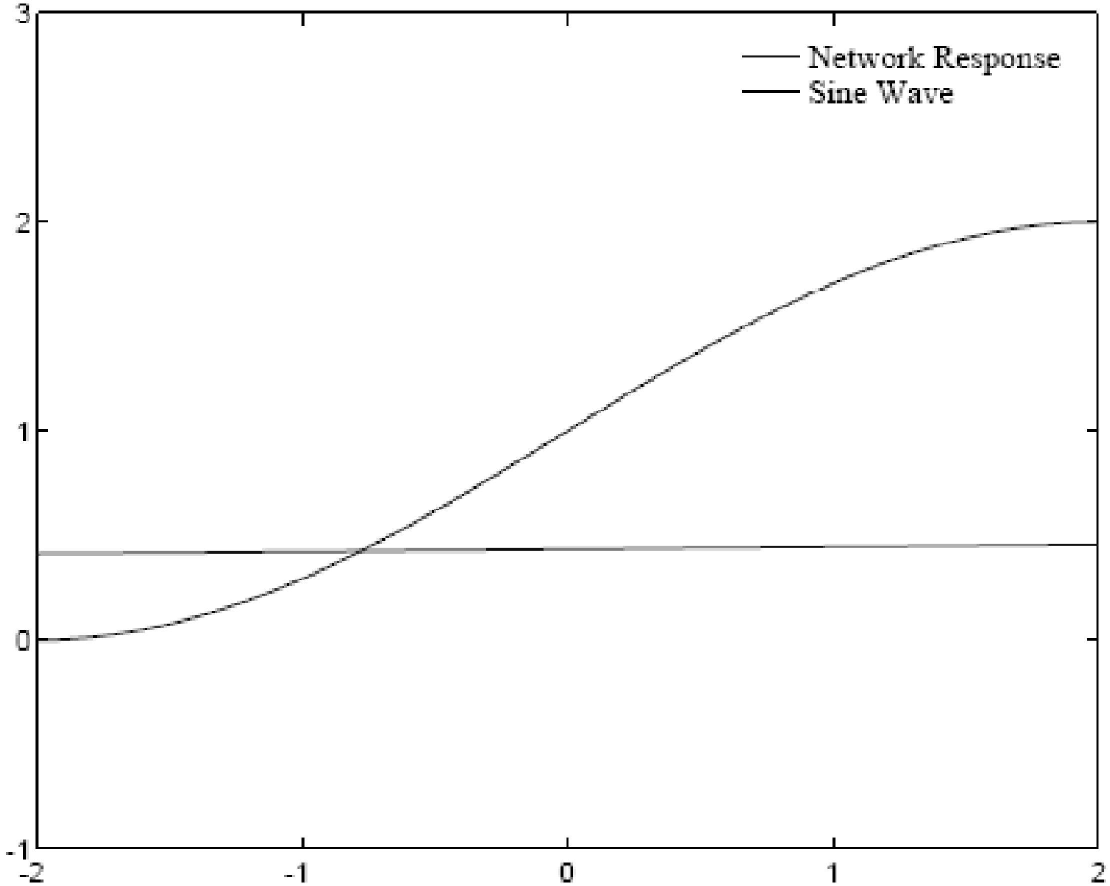
$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \qquad (-2 \le p \le 2)$$



1-2-1
Network

Initial Conditions

$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} \quad \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} \quad \mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix} \quad \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix}$$



Forward Propagation

$$a^0 = p = 1$$

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{a}^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix}-0.27\\-0.41\end{bmatrix}\begin{bmatrix}1\end{bmatrix} + \begin{bmatrix}-0.48\\-0.13\end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix}-0.75\\-0.54\end{bmatrix}\right)$$

$$\mathbf{a}^1 = \begin{bmatrix}\dfrac{1}{1+e^{0.75}}\\[2ex]\dfrac{1}{1+e^{0.54}}\end{bmatrix} = \begin{bmatrix}0.321\\0.368\end{bmatrix}$$

$$a^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) = purelin\left(\begin{bmatrix}0.09 & -0.17\end{bmatrix}\begin{bmatrix}0.321\\0.368\end{bmatrix} + \begin{bmatrix}0.48\end{bmatrix}\right) = \begin{bmatrix}0.446\end{bmatrix}$$

$$e = t - a = \left\{1 + \sin\left(\frac{\pi}{4}p\right)\right\} - a^2 = \left\{1 + \sin\left(\frac{\pi}{4}1\right)\right\} - 0.446 = 1.261$$

Transfer Function Derivative

$$\dot{f}^1(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1 - a^1)(a^1)$$

$$\dot{f}^2(n) = \frac{d}{dn}(n) = 1$$

Backpropagation

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t}-\mathbf{a}) = -2\left[\dot{f}^2(n^2)\right](1.261) = -2\left[1\right](1.261) = -2.522$$

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T\mathbf{s}^2 = \begin{bmatrix}(1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1)\end{bmatrix}\begin{bmatrix}0.09 \\ -0.17\end{bmatrix}\begin{bmatrix}-2.522\end{bmatrix}$$

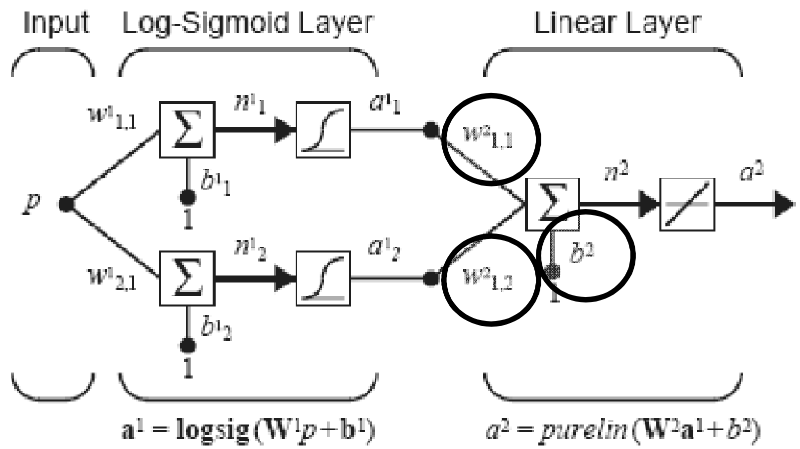
$$\mathbf{s}^1 = \begin{bmatrix}(1-0.321)(0.321) & 0 \\ 0 & (1-0.368)(0.368)\end{bmatrix}\begin{bmatrix}0.09 \\ -0.17\end{bmatrix}\begin{bmatrix}-2.522\end{bmatrix}$$

$$\mathbf{s}^1 = \begin{bmatrix}0.218 & 0 \\ 0 & 0.233\end{bmatrix}\begin{bmatrix}-0.227 \\ 0.429\end{bmatrix} = \begin{bmatrix}-0.0495 \\ 0.0997\end{bmatrix}$$

Weight Update (1)

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$



$\alpha = 0.1$ **s²** = -2.522 $\mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix}$ $\mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix}$ $\mathbf{a}^1 = \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix}$

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha \mathbf{s}^2 (\mathbf{a}^1)^T = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix} - 0.1 \begin{bmatrix} -2.522 \end{bmatrix} \begin{bmatrix} 0.321 & 0.368 \end{bmatrix}$$

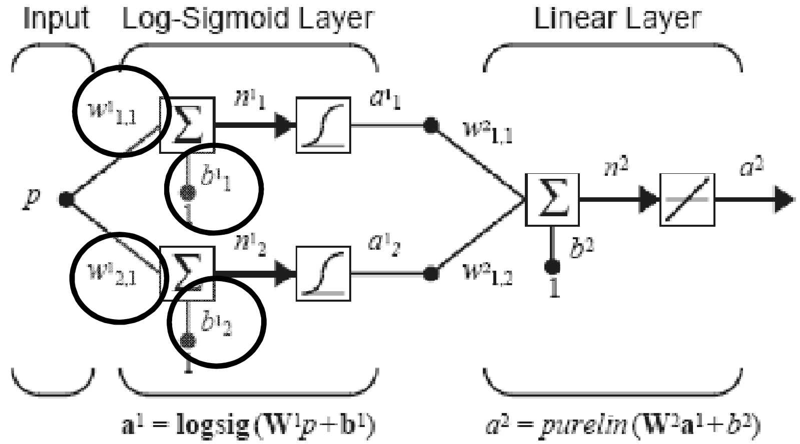
$$\mathbf{W}^2(1) = \begin{bmatrix} 0.171 & -0.0772 \end{bmatrix}$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha \mathbf{s}^2 = \begin{bmatrix} 0.48 \end{bmatrix} - 0.1 \begin{bmatrix} -2.522 \end{bmatrix} = \begin{bmatrix} 0.732 \end{bmatrix}$$

Weight Update (2)

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m$$



$$\alpha = 0.1 \quad \mathbf{s}^1 = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} \quad \mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} \quad \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}$$

$\mathbf{a}^0 = [1]$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha \mathbf{s}^1 (\mathbf{a}^0)^T = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} - 0.1 \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} -0.265 \\ -0.420 \end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha \mathbf{s}^1 = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} - 0.1 \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} = \begin{bmatrix} -0.475 \\ -0.140 \end{bmatrix}$$

This completes the first iteration of the BP algorithm.

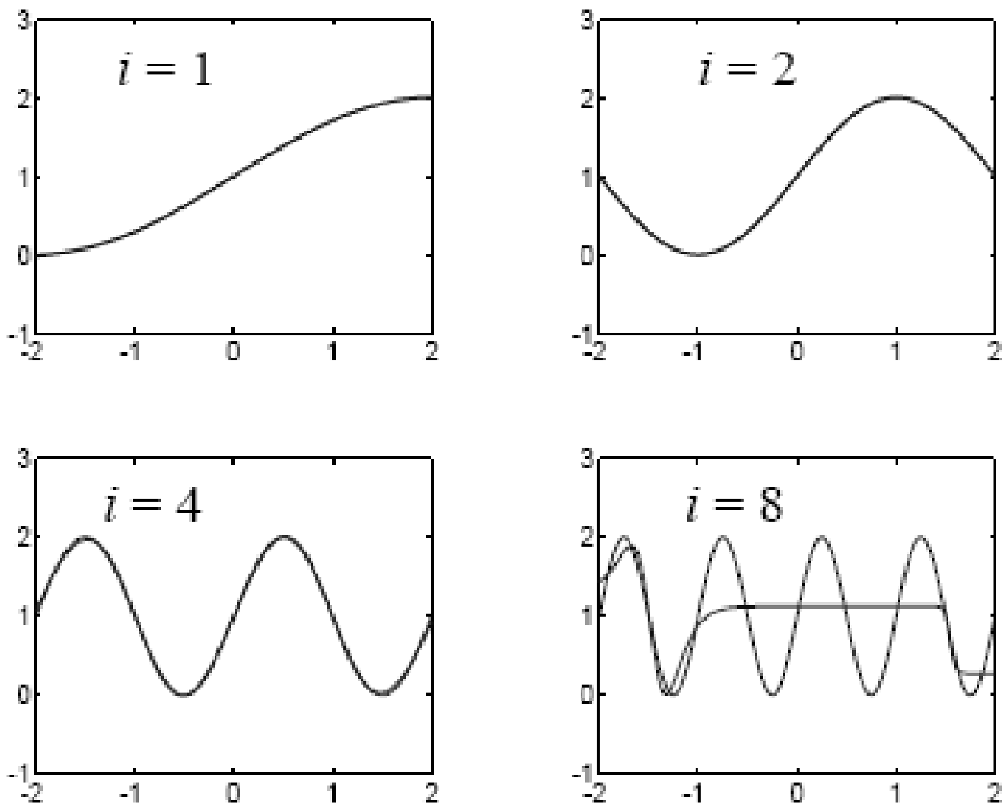
Choice of Network Architecture

$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right)$$

Transfer function for first layer → log-sigmoid

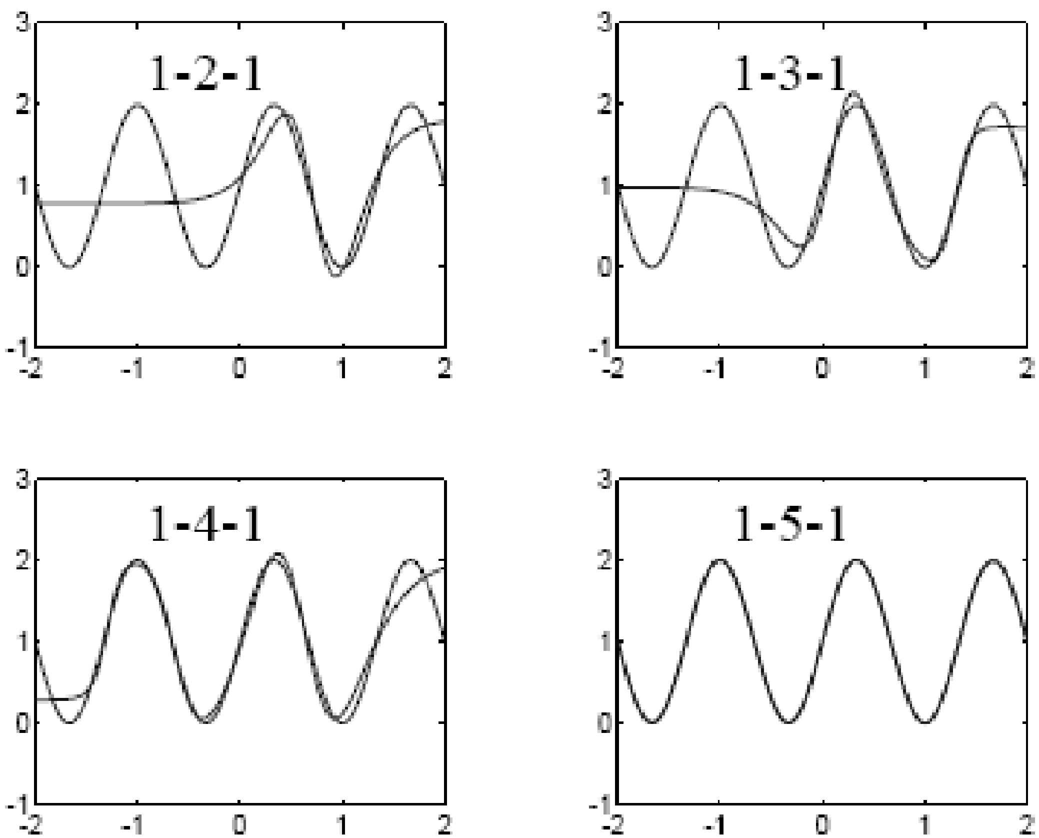
Transfer function for the second layer → linear

1-3-1 Network



Choice of Network Architecture

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right)$$

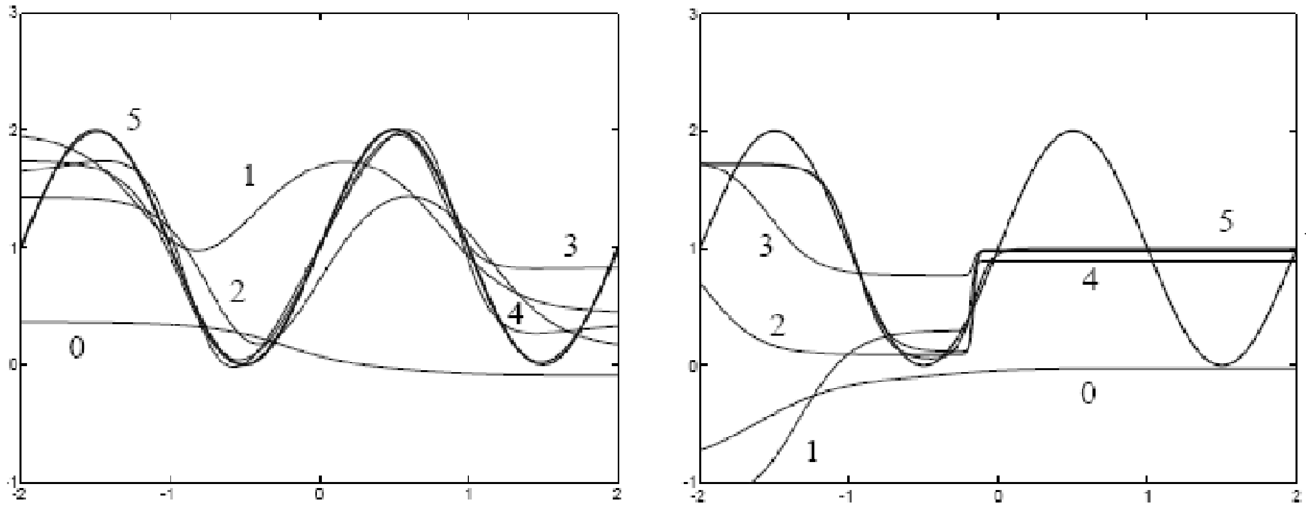


To approximate a function with a large number of inflection points, we need to have a large number of neurons in the hidden layer.

Convergence

$$g(p) = 1 + \sin(\pi p)$$

1-3-1 network



Results obtained for two different initial conditions

→ Second one converged to a local minimum.

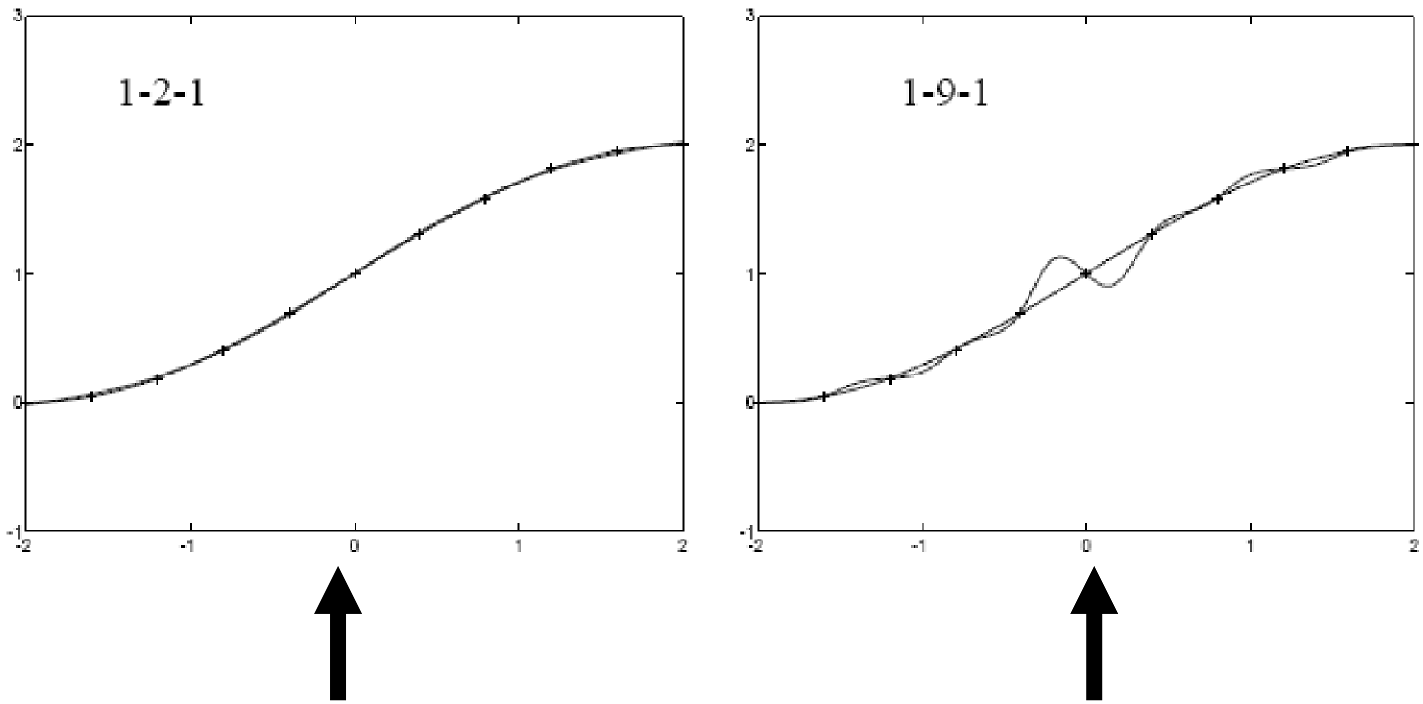
→ Algorithm does not guarantee to converge to the global minimum

Try several different initial conditions in order to ensure that an optimum solution has been obtained.

Generalization

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \qquad p = -2, -1.6, -1.2, \ldots, 1.6, 2$$



11 data points and 7 parameters

11 data points and 28 parameters

Generalization

- Should have fewer parameters than there are data points in the training set.
- Should use the simplest network that can adequately represent the training set.

Don’t use a bigger network when a smaller network will work.
(Ockham's razor)