11 Backpropagation

11.1 Multilayer Perceptrons

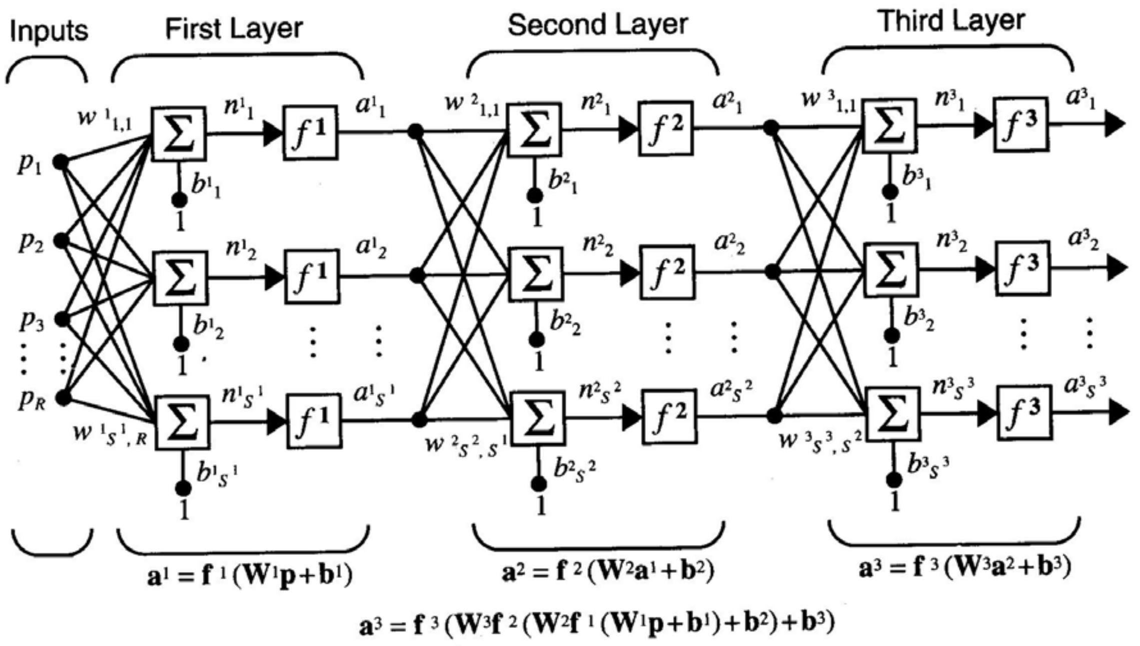

Inputs
First Layer
Second Layer
Third Layer
p_1 p_2 p_3 p_R
$w^1_{1,1}$ $w^1_{S^1,R}$
n^1_1 n^1_2 $n^1_{S^1}$
a^1_1 a^1_2 $a^1_{S^1}$
b^1_1 b^1_2 $b^1_{S^1}$
$w^2_{1,1}$ $w^2_{S^2,S^1}$
a^2_1 a^2_2 $a^2_{S^2}$
$w^3_{1,1}$ $w^3_{S^3,S^2}$
a^3_1 a^3_2 $a^3_{S^3}$
$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$
$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$

Figure 11.1-1 Three-Layer Network

11.1.1 Pattern Classification

The input/target pairs for exclusive-or (XOR) gate,

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\} \quad (11.1.1\text{-}1)$$

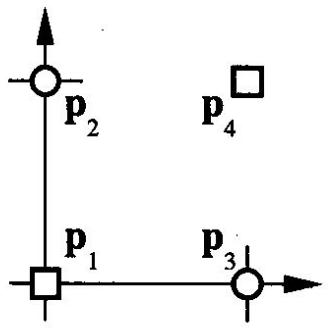


Figure 11.1.1-1 Exclusive-Or Problem

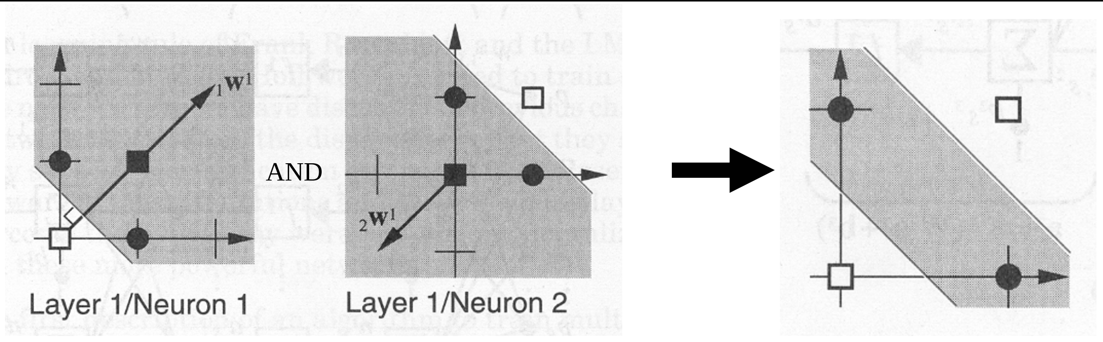


Figure 11.1.1-2 Decision Boundaries for XOR Network

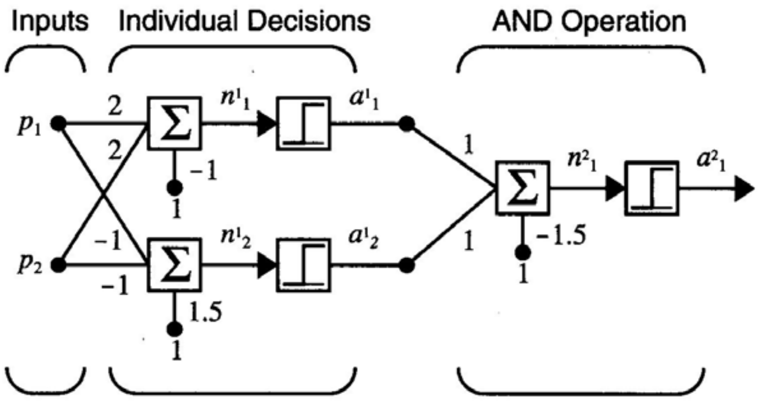


Figure 11.1.1-3 Two-Layer XOR Network

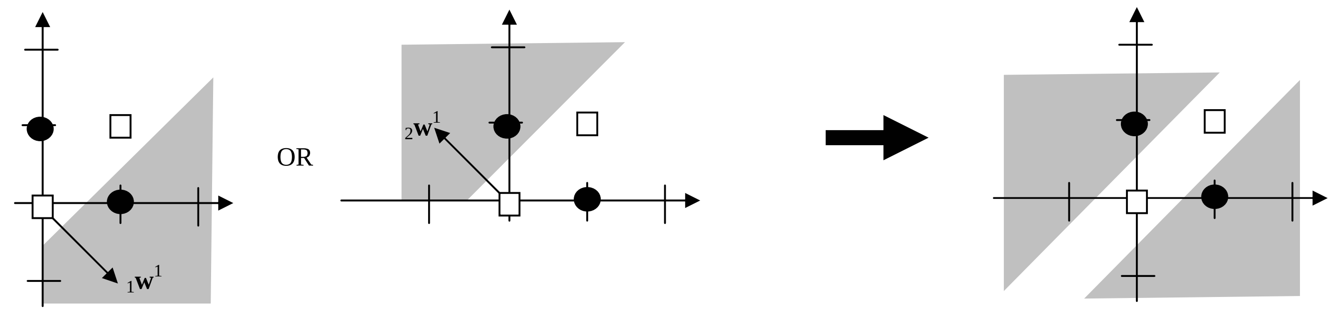


Figure 11.1.1-4 Decision Boundaries for XOR Network

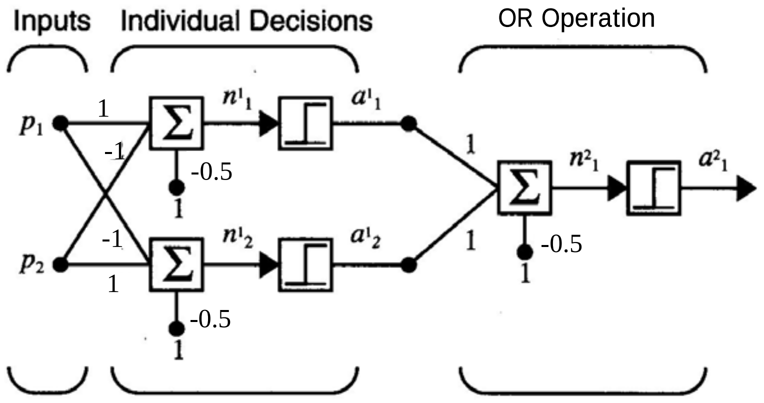


Figure 11.1.1-5 Two-Layer XOR Network

11.1.2 Function Approximation

Two-layer, 1-2-1 network,

$$f^1(n) = \frac{1}{1+e^{-n}} \text{ and } f^2(n) = n \tag{11.1.2-1}$$

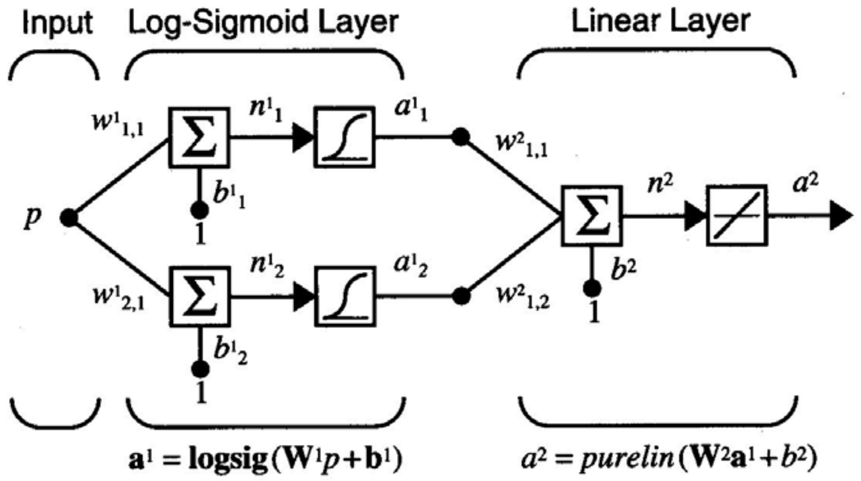


Figure 11.1.2-1 1-2-1 Function Approximation Network

$$w^1_{1,1} = 10,\ w^1_{2,1} = 10,\ b^1_1 = -10,\ b^1_2 = 10 \text{ and } w^2_{1,1} = 1,\ w^2_{1,2} = 1,\ b^2 = 0 \tag{11.1.2-2}$$

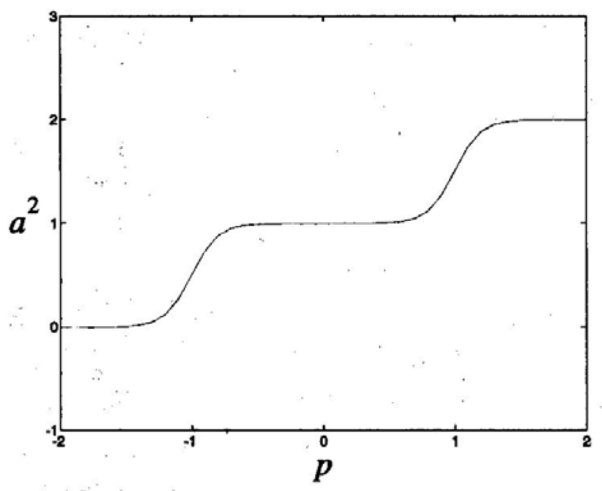


Figure 11.1.2-2 Nominal Response of Network of Figure 11.1.2-1

$$n_1^1 = w_{1,1}^1 p + b_1^1 = 0 \Rightarrow p = -\frac{b_1^1}{w_{1,1}^1} = -\frac{-10}{10} = 1 \tag{11.1.2-3}$$

$$n_2^1 = w_{2,1}^1 p + b_2^1 = 0 \Rightarrow p = -\frac{b_2^1}{w_{2,1}^1} = \frac{10}{10} = -1 \tag{11.1.2-4}$$

$$-1 \le w_{1,1}^2 \le 1,\ -1 \le w_{1,2}^2 \le 1,\ 0 \le b_2^1 \le 20,\ -1 \le b^2 \le 1 \qquad (11.1.2\text{-}5)$$

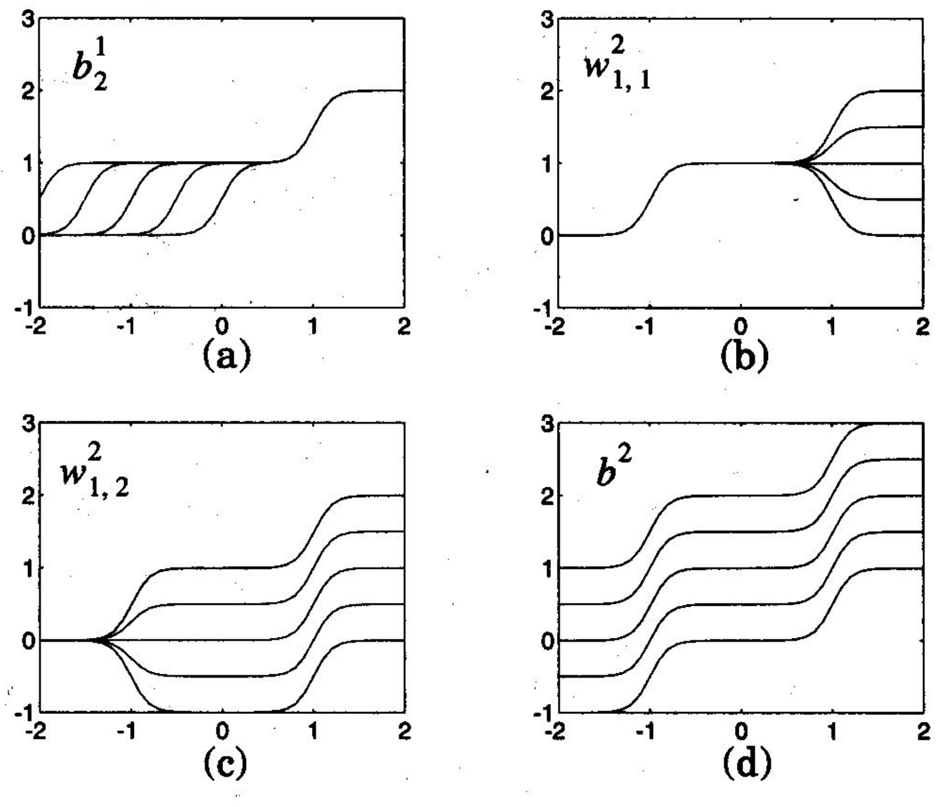


Figure 11.1.2-3 Effect of Parameter Changes on Network Response

11.2 The Backpropagation Algorithm

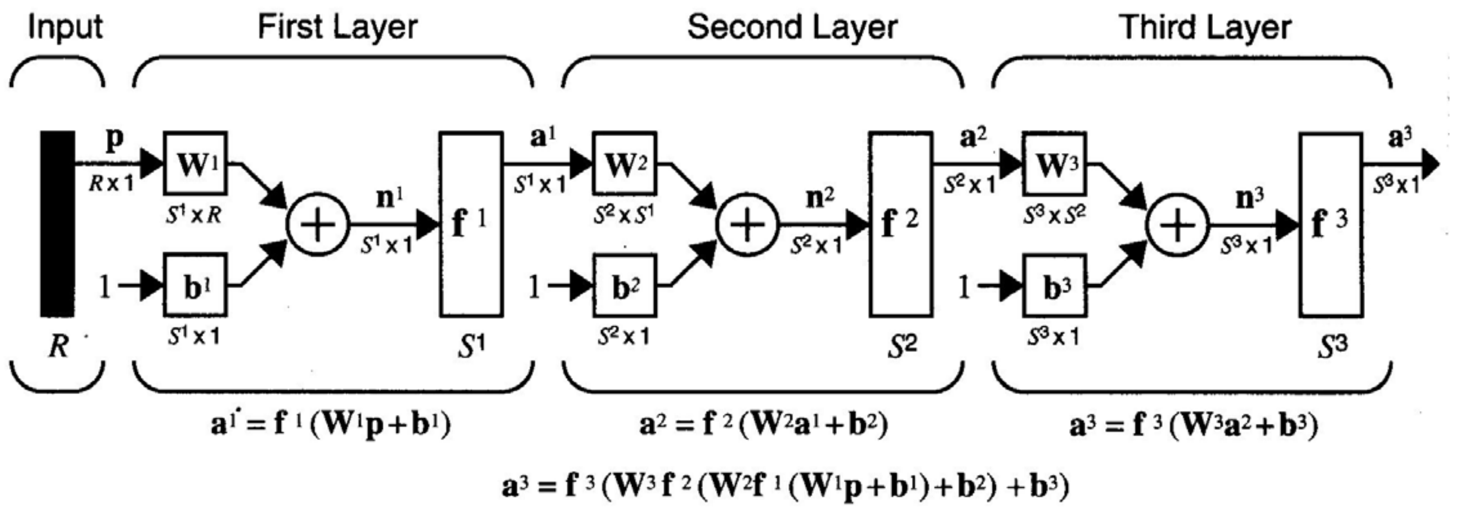


Figure 11.2-1 Three-Layer Network, Abbreviated Notation

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, ..., M\text{-}1 \qquad (11.2\text{-}1)$$

$$\mathbf{a}^{0} = \mathbf{p} \qquad (11.2\text{-}2)$$

$$\mathbf{a} = \mathbf{a}^{M} \qquad (11.2\text{-}3)$$

11.2-1 Performance Index

- Backpropagation algorithm for multilayer networks: generalization of the LMS algorithm
- Performance index: mean square error

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\cdots,\{\mathbf{p}_Q,\mathbf{t}_Q\} \tag{11.2.1-1}$$

For single output,

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] \tag{11.2.1-2}$$

For multiple outputs,

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})] \tag{11.2.1-3}$$

Approximation of mean square error by

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k) \tag{11.2.1-4}$$

By the steepest descent algorithm for the approximate mean square error,

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m} \tag{11.2.1-5}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m} \tag{11.2.1-6}$$

11.2.2 Chain Rule

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} \tag{11.2.2-1}$$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{11.2.2-2}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{11.2.2-3}$$

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \tag{11.2.2-4}$$

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{11.2.2-5}$$

Defining the sensitivity of $\hat{F}$ to changes in the *i*th element of the net input at layer *m*,

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m} \tag{11.2.2-6}$$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{11.2.2-7}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{11.2.2-8}$$

Steepest descent algorithm,

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha s_i^m a_j^{m-1} \tag{11.2.2-9}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{11.2.2-10}$$

In matrix form,

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{11.2.2-11}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \tag{11.2.2-12}$$

where

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \dfrac{\partial \hat{F}}{\partial n_1^m} \\ \dfrac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \dfrac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix} \tag{11.2.2-13}$$

11.2.3 Backpropagating the Sensitivities

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \mathbf{s}^{m+1} \tag{11.2.3-1}$$

Jacobian matrix,

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix} \tag{11.2.3-2}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial \left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1} \right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m(n_j^m) \tag{11.2.3-3}$$

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \dot{\mathbf{F}}^m(\mathbf{n}^m) \tag{11.2.3-4}$$

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix} \tag{11.2.3-5}$$

Recurrence relation for the sensitivity by using the chain rule in matrix form,

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1} \tag{11.2.3-6}$$

- Backpropagation algorithm: The sensitivities are propagated backward through the network from the last layer to the first layer:

$$\mathbf{s}^M \to \mathbf{s}^{M-1} \to \cdots \to \mathbf{s}^2 \to \mathbf{s}^1 \tag{11.2.3-7}$$

Sensitivity at the final layer,

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M}(t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M} \tag{11.2.3-8}$$

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M) \tag{11.2.3-9}$$

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M) \tag{11.2.3-10}$$

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}) \tag{11.2.3-11}$$

11.2.4 Summary

1. The first step: propagate the input forward through the network:

$$\mathbf{a}^0 = \mathbf{p} \qquad (11.2.4\text{-}1)$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, ..., M\text{-}1 \qquad (11.2.4\text{-}2)$$

$$\mathbf{a} = \mathbf{a}^M \qquad (11.2.4\text{-}3)$$

2. The second step: propagate the sensitivities backward through the network:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \qquad (11.2.4\text{-}4)$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}, \text{ for } m = M\text{-}1, \ldots, 2, 1 \qquad (11.2.4\text{-}5)$$

3. The last step: update the weights and biases using the approximate steepest descent rule:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \qquad (11.2.4\text{-}6)$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m \qquad (11.2.4\text{-}7)$$

11.3 Example

1-2-1 network to approximate the function,

$$g(p) = 1 + \sin\left(\frac{\pi}{4} p\right) \text{ for } -2 \le p \le 2 \tag{11.3-1}$$

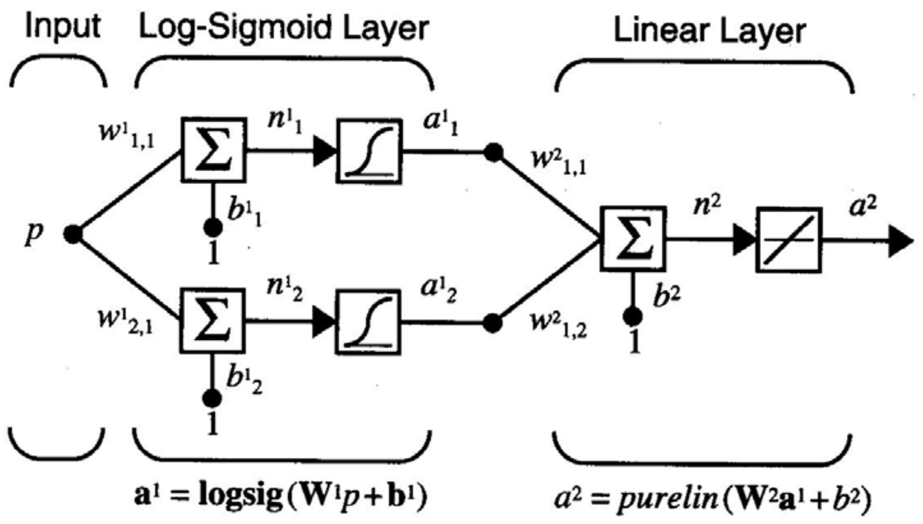


Figure 11.3-1 1-2-1 Function Approximation Network

Initial network,

$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}, \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}, \mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix}, \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix} \tag{11.3-2}$$

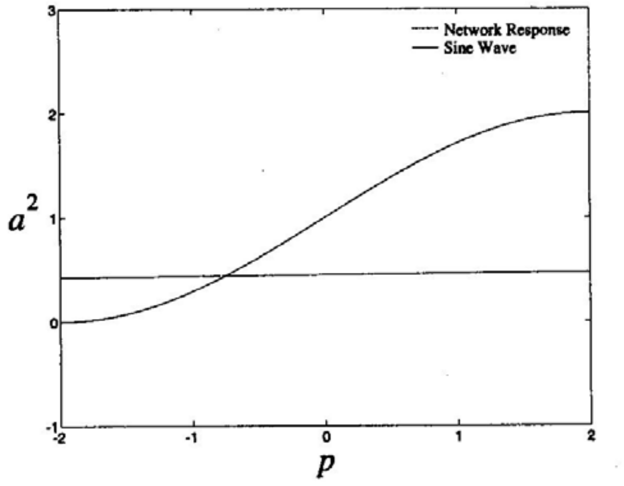


Figure 11.3-2 Initial Network Response and Approximated Sine Function

Presenting p = 1,

$$a^0 = p = 1 \tag{11.3-3}$$

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{a}^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}\begin{bmatrix}1\end{bmatrix} + \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix} -0.75 \\ -0.54 \end{bmatrix}\right) = \begin{bmatrix} \dfrac{1}{1+e^{0.75}} \\ \dfrac{1}{1+e^{0.54}} \end{bmatrix} = \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix} \tag{11.3-4}$$

$$a^2 = f^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) = purelin\left(\begin{bmatrix} 0.09 & -0.17 \end{bmatrix}\begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix} + \begin{bmatrix}0.48\end{bmatrix}\right) = \begin{bmatrix}0.446\end{bmatrix} \tag{11.3-5}$$

$$e = t - a = \left\{1 + \sin\left(\frac{\pi}{4}p\right)\right\} - a^2 = \left\{1 + \sin\left(\frac{\pi}{4}1\right)\right\} - 0.446 = 1.261 \tag{11.3-6}$$

$$\dot{f}^1(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{\left(1+e^{-n}\right)^2} = \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = \left(1 - a^1\right)\left(a^1\right) \tag{11.3-7}$$

$$\dot{f}^2(n) = \frac{d}{dn}(n) = 1 \tag{11.3-8}$$

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t} - \mathbf{a}) = -2\left[\dot{f}^2(n^2)\right](1.261) = -2\left[1\right](1.261) = -2.522 \tag{11.3-9}$$

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

$$= \begin{bmatrix} (1-0.321)(0.321) & 0 \\ 0 & (1-0.368)(0.368) \end{bmatrix}\begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}\begin{bmatrix}-2.522\end{bmatrix} = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} \tag{11.3-10}$$

Learning rate α = 0.1,

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha\mathbf{s}^2(\mathbf{a}^1)^T = \begin{bmatrix}0.09 & -0.17\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix}\begin{bmatrix}0.321 & 0.368\end{bmatrix} = \begin{bmatrix}0.171 & -0.0772\end{bmatrix} \quad (11.3\text{-}11)$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha\mathbf{s}^2 = \begin{bmatrix}0.48\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix} = \begin{bmatrix}0.732\end{bmatrix} \quad (11.3\text{-}12)$$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix}-0.27\\ -0.41\end{bmatrix} - 0.1\begin{bmatrix}-0.0495\\ 0.0997\end{bmatrix}\begin{bmatrix}1\end{bmatrix} = \begin{bmatrix}-0.265\\ -0.420\end{bmatrix} \quad (11.3\text{-}13)$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha\mathbf{s}^1 = \begin{bmatrix}-0.48\\ -0.13\end{bmatrix} - 0.1\begin{bmatrix}-0.0495\\ 0.0997\end{bmatrix} = \begin{bmatrix}-0.475\\ -0.140\end{bmatrix} \quad (11.3\text{-}14)$$

- Proceed to choose another input *p* and perform another iteration of the algorithm.
- Continue to iterate until the difference between the network response and the target function reaches some acceptable level.

- For a network to be able to generalize, it should have fewer parameters than there are data points in the training set.
- Don't use a bigger network when a smaller network will work.
- In general pattern recognition problem
    - First layer: to generate decision boundaries
    - Second layer: to perform AND operation of the decision boundaries
    - Third layer: to perform OR operation of the separated group

12 Variations on Backpropagation

12.1 Drawbacks of Backpropagation

- LMS algorithm is guaranteed to converge to a solution that minimizes the mean squared error, so long as the learning rate is not too large.
- In LMS algorithm, the mean squared error of a single-layer linear network is a quadratic function.
- SDBP (Steepest Descent Backpropagation) is equivalent to the LMS algorithm when used on a single-layer linear network.
- In SDBP, the performance surface of a multilayer network may have many local minimum points, and the curvature can vary widely in different regions of the parameter space.

12.1.1 Performance Surface Example

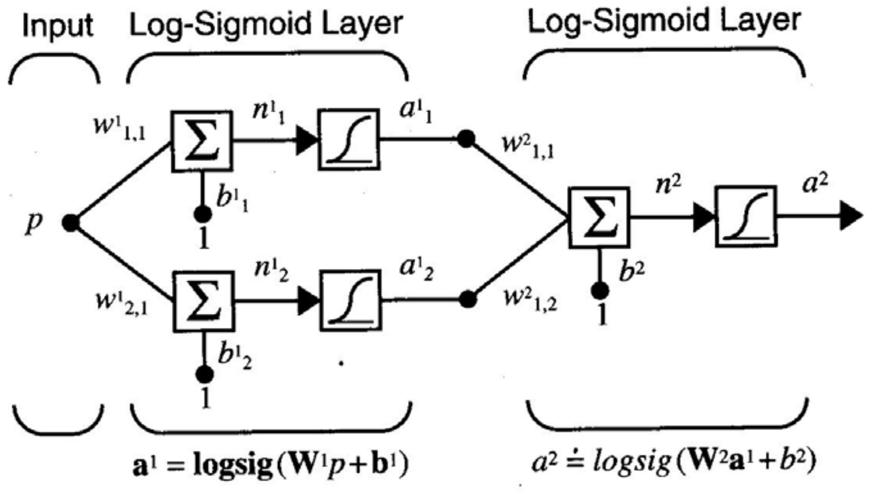


Figure 12.1.1-1 1-2-1 Function Approximation Network

$$w^1_{1,1} = 10, w^1_{2,1} = 10, b^1_1 = -5, b^1_2 = 5, w^2_{1,1} = 1, w^2_{1,2} = 1, b^2 = -1 \quad (12.1.1\text{-}1)$$

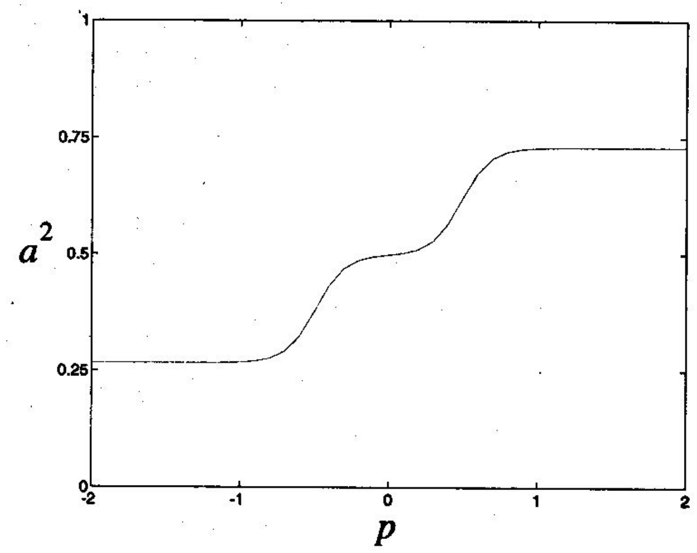


Figure 12.1.1-2 Nominal Function

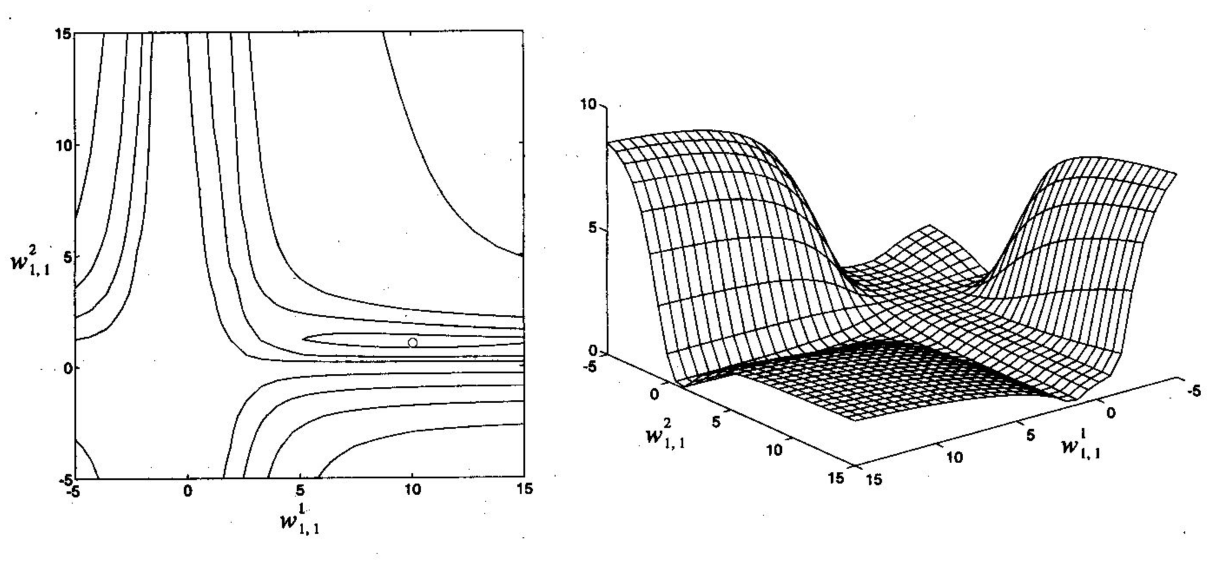
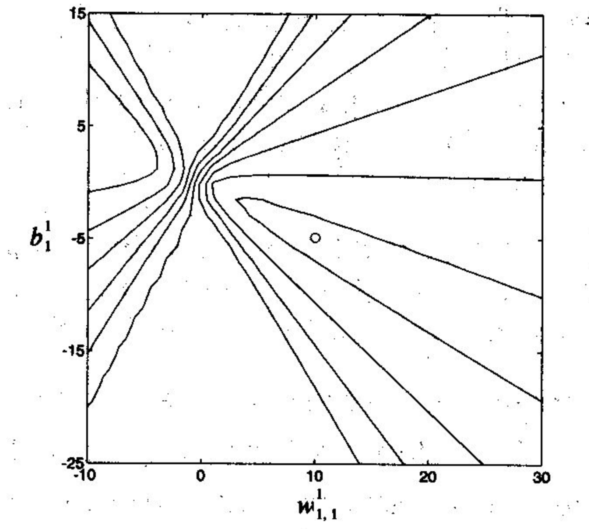


Figure 12.1.1-3 Squared Error Surface Versus $w^1_{1,1}$ and $w^2_{1,1}$

- It is clearly not a quadratic function. The curvature varies drastically over the parameter space.
- In some regions the surface is very flat, which would allow a large learning rate.
- In some regions the curvature is high, which would require a small learning rate.
- The sigmoid is very flat for very large magnitude of inputs.
- There are more than one local minimum points.
- The global minimum point locates at $w^1_{1,1}$= 10 and $w^2_{1,1}$ = 1. A local minimum locates at $w^1_{1,1}$= 0.88 and $w^2_{1,1}$ = 38.6.



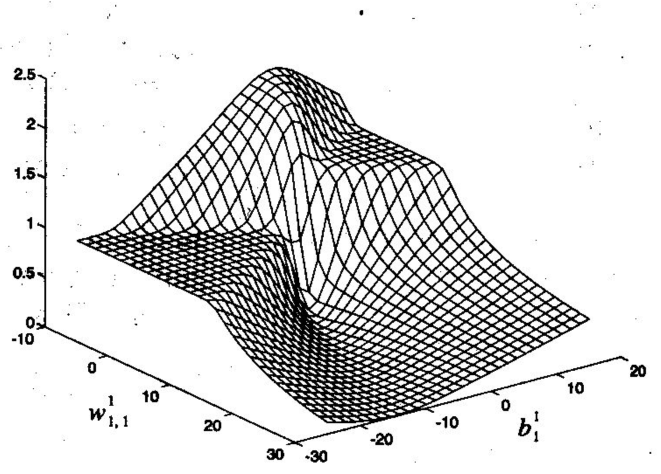


Figure 12.1.1-4 Squared Error Surface Versus $w_{1,1}^1$ and b_1^1

- The minimum error is zero and occurs when $w_{1,1}^1$= 10 and b_1^1=-5.
- With an initial guess of $w_{1,1}^1$= 0 and b_1^1=-10, the gradient is very close to zero, and the steepest descent algorithm would stop, even though it is not close to a local minimum point.

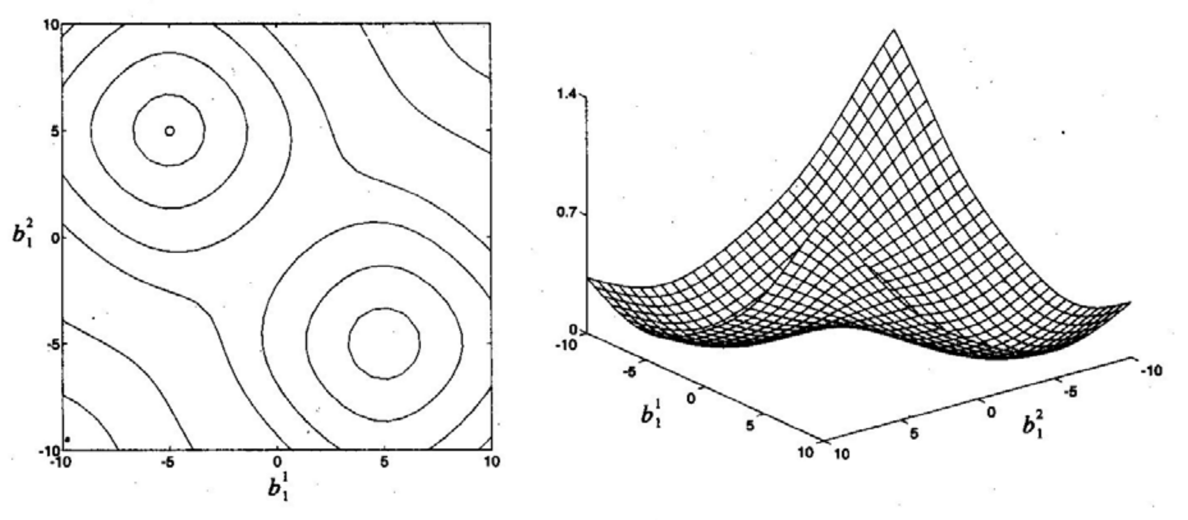


Figure 12.1.1-5 Squared Error Surface Versus b_1^1 and b_1^2

- Two zero error minimum points locate at b_1^1 = -5 with b_1^2 = 5 and b_1^1 = 5 with b_1^2 = -5.
- The surface is symmetry causing zero to be a saddle point of the performance surface.

Hints: used to set initial guess for the SDBP algorithm

- The initial parameters should not be set to zero. This is because the origin of the parameter space tends to be a saddle point for the performance surface.
- The initial parameters should not be set to large values. This is because the performance surface tends to have very flat regions as we move far away from the optimum point.

- Typically, the initial weights and biases should be set to small random values.
- It is also useful to try several different initial guesses, in order to be sure that the algorithm converges to a global minimum point.

In **batching mode** of SDBP, weights and biases are updated every after all the inputs/targets presented to the network.

$$\Delta\mathbf{W} = \sum_{\mathbf{i=1}}^{\mathbf{Q}} \Delta\mathbf{W}_i \tag{12.1.1-2}$$

$$\Delta\mathbf{b} = \sum_{\mathbf{i=1}}^{\mathbf{Q}} \Delta\mathbf{b}_i \tag{12.1.1-3}$$

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \Delta\mathbf{W} \tag{12.1.1-4}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \Delta\mathbf{b} \tag{12.1.1-5}$$

12.2 Heuristic Modifications of Backpropagation

Heuristic Backpropagation

- Momemtum Backpropagation
- Variable Learning Rate Backpropagation

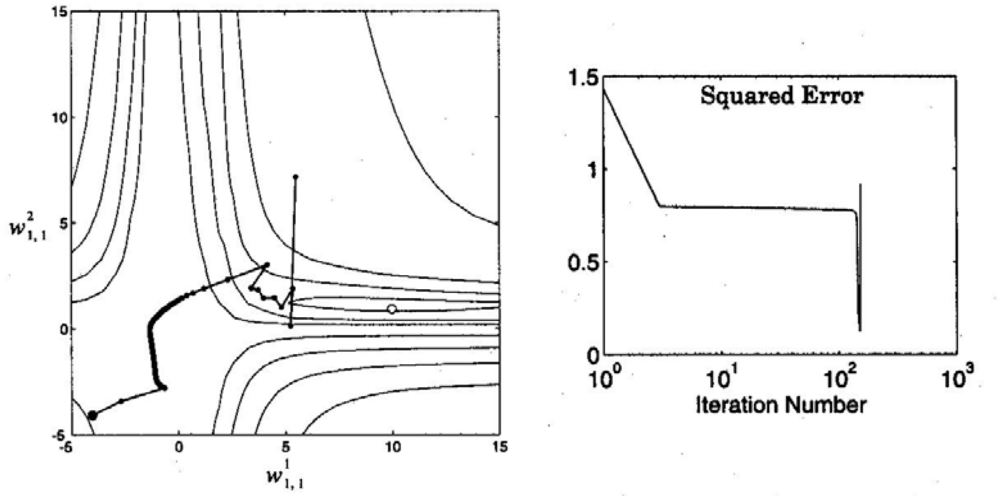


Figure 12.2-1 Trajectory with Too Large Learning

12.2.1 Momentum

First-order low-pass filter,

$$y(k) = \gamma y(k-1) + (1-\gamma)w(k) \tag{12.2.1-1}$$

where *w*(*k*): the input to the filter, *y*(*k*): the output of the filter, γ: the momentum coefficient

$$0 \le \gamma < 1 \tag{12.2.1-2}$$

Sine wave input,

$$w(k) = 1 + \sin\left(\frac{2\pi k}{16}\right) \tag{12.2.1-3}$$

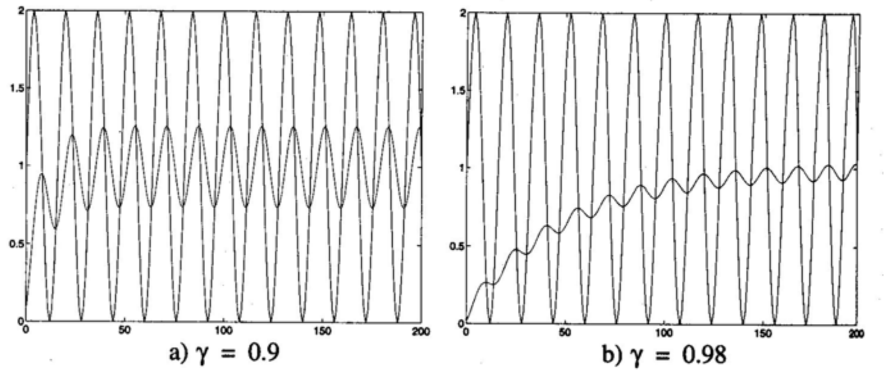


Figure 12.2.1-1 Smoothing Effect of Momentum

- The oscillation of the filter output is less than the oscillation in the filter input.
- As γ is increased the oscillation in the filter output is reduced.
- The average filter output is the same as the average filter input.
- As γ is increased the filter output is slower to respond.

Parameter updates for SDBP,

$$\Delta \mathbf{W}^m(k) = -\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{12.2.1-4}$$

$$\Delta \mathbf{b}^m(k) = -\alpha \mathbf{s}^m \tag{12.2.1-5}$$

Parameter updates for momentum modification to backpropagation (MOBP),

$$\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{12.2.1-6}$$

$$\Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m \tag{12.2.1-7}$$

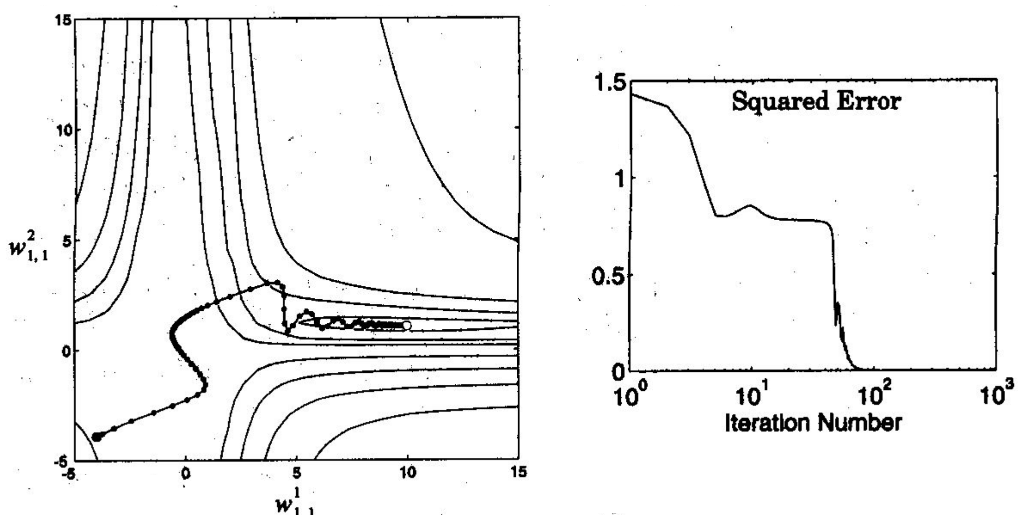


Figure 12.2.1-2 Trajectory with Momentum, $\gamma = 0.8$

- By the use of momentum, a larger learning rate can be used, while maintaining the stability of the algorithm.
- Momentum tends to make the trajectory continue in the same direction.
- The larger the value of γ, the more "momentum" the trajectory has.

12.2.2 Variable Learning Rate

The rules of the variable learning rate backpropagation algorithm (VLBP):

1. If the squared error (over the entire training set) increases by more than some set percentage ζ (typically one to five percent) after a weight update, then the weight update is discarded, the learning rate is multiplied by some factor $0 < \rho < 1$, and the momentum coefficient γ (if it is used) is set to zero.
2. If the squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by some factor $\eta > 1$. If γ has been previously set to zero, it is reset to its original value.
3. If the squared error increases by less than ζ, then the weight update is accepted but the learning rate and the momentum coefficient are unchanged.

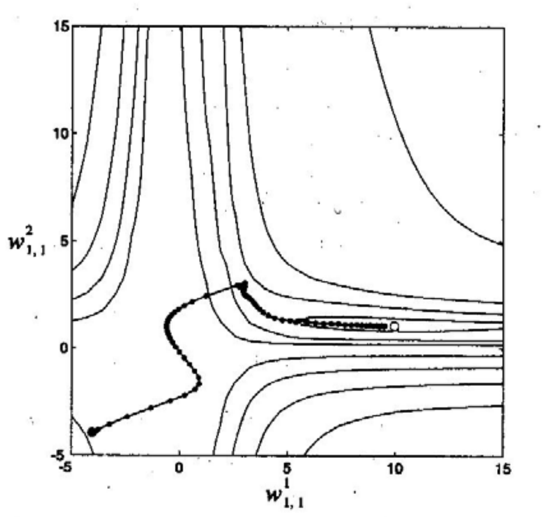


Figure 12.2.2-1 Variable Learning Rate Trajectory, $\eta = 1.05, \rho = 0.7, \zeta = 4\%$, $\gamma = 0.8$

- The learning rate, step size, tends to increase when the trajectory is traveling in a straight line with constantly decreasing error.
- When the trajectory reaches a narrow valley, the learning rate is rapidly decreased and the momentum is eliminated, which allows the trajectory to make the quick turn to follow the valley toward the minimum point.

12.3 Numerical Optimization Techniques

Numerical Backpropagation

- Conjugate Gradient Backpropagation
- Levenberg-Marquardt Backpropagation

12.3.1 Conjugate Gradient

General procedure for locating the minimum of a function in a specified direction:

1. Interval location: to find some initial interval that contains a local minimum

2. Interval reduction: to reduces the size of the initial interval until the minimum is located to the desired accuracy

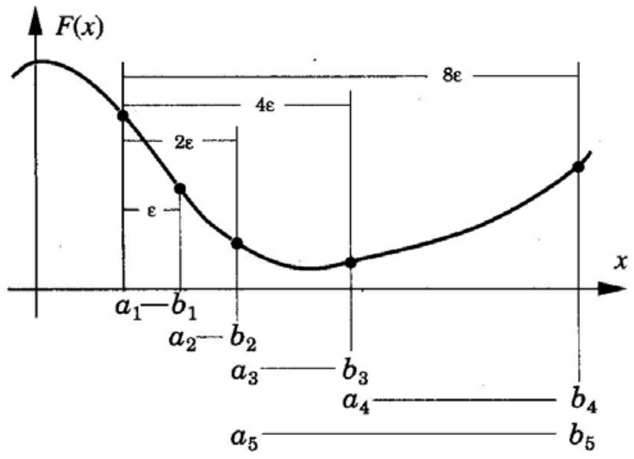


Figure 12.3.1-1 Interval Location

1. Evaluating the performance index at an initial point, represented by a_1 in the figure.

$$F(\mathbf{x}_0) \tag{12.3.1-1}$$

2. Evaluating the function at a second point, represented by b_1 in the figure, which is a distance ε from the initial point, along the first search direction $\mathbf{p}_0$.

$$F(\mathbf{x}_0+\varepsilon\mathbf{p}_0) \tag{12.3.1-2}$$

4. Continuing to evaluate the performance index at new points b_i, successively doubling the distance between points. This process stops when the function increases between two consecutive evaluations.

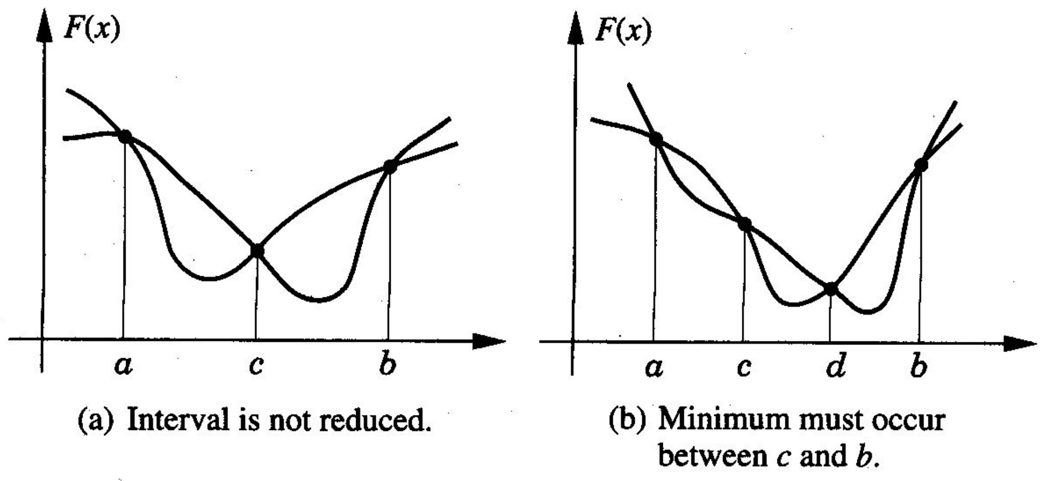

F(x)
a
c
b
(a) Interval is not reduced.
F(x)
a
c
d
b
(b) Minimum must occur between c and b.

Figure 12.3.1-2 Reducing the Size of the Interval of Uncertainty

The algorithm for the Golden Section search:

```
                $\tau = 0.618$
Set             $c_1 = a_1 + (1-\tau)(b_1 - a_1), F_c = F(c_1)$
                $d_1 = b_1 - (1-\tau)(b_1 - a_1), F_d = F(d_1)$
For $k$ = 1, 2, ... repeat
    If $Fc < F_d$ then
        Set     $a_{k+1} = a_k; b_{k+1} = d_k; d_{k+1} = c_k$
                $c_{k+1} = a_{k+1} + (1-\tau)(b_{k+1} - a_{k+1})$
                $F_d = F_c; F_c = F(c_{k+1})$
    else
        Set     $a_{k+1} = c_k; b_{k+1} = b_k; c_{k+1} = d_k$
                $d_{k+1} = b_{k+1} - (1-\tau)(b_{k+1} - a_{k+1})$
                $F_c = F_d; F_d = F(d_{k+1})$
    end
end  until $b_{k+1} - a_{k+1} < tol$
```

where *tol* is the accuracy tolerance set by the user.

- For quadratic functions the algorithm will converge to the minimum in at most *n* iterations, where *n* is the number of parameters being optimized.
- The mean squared error performance index for multilayer networks is not quadratic, therefore the algorithm would not normally converge in *n* iterations.
- In conjugate gradient backprogagation, the search direction is reset to the steepest descent direction (negative of the gradient) after *n* iterations.

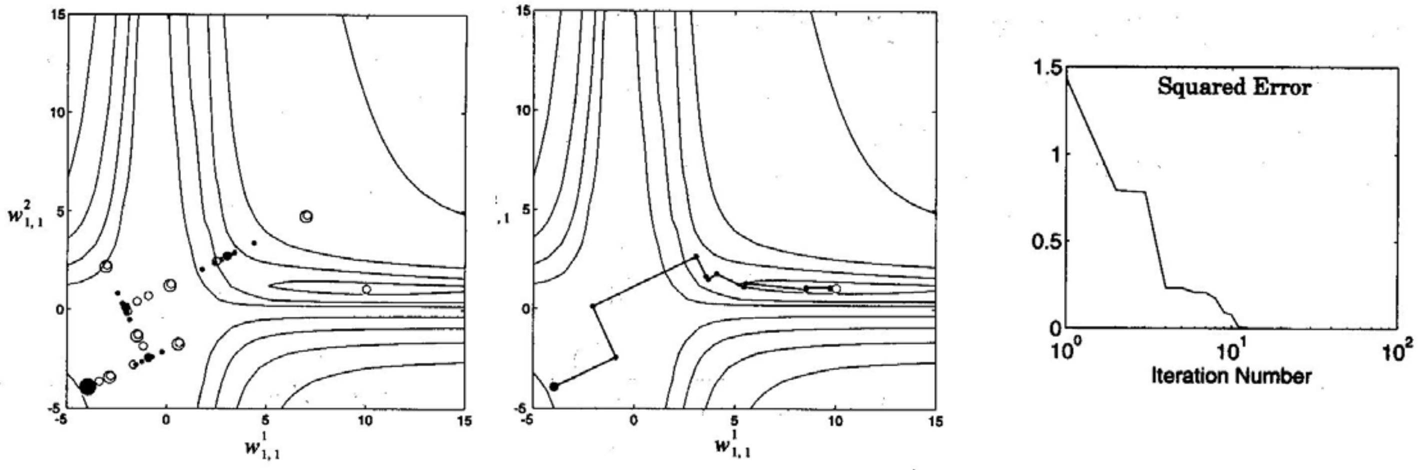


Figure 12.3.1-3 Steps and Trajectory of CGBP

12.3.2 Levenberg-Marquardt Algorithm

Newton's method,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \qquad (12.3.2\text{-}1)$$

where $\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}$ and $\mathbf{g}_k \equiv \nabla F(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}$.

$F(\mathbf{x})$: a sum of squares function,

$$F(\mathbf{x}) = \sum_{i=1}^{N} v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \qquad (12.3.2\text{-}2)$$

$$[\nabla F(\mathbf{x})]_j = \frac{\partial F(\mathbf{x})}{\partial x_j} = 2\sum_{i=1}^{N} v_i(\mathbf{x})\frac{\partial v_i(\mathbf{x})}{\partial x_j} \qquad (12.3.2\text{-}3)$$

$$\nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \qquad (12.3.2\text{-}4)$$

where Jacobian matrix,

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1(\mathbf{x})}{\partial x_1} & \frac{\partial v_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial v_2(\mathbf{x})}{\partial x_1} & \frac{\partial v_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_N(\mathbf{x})}{\partial x_1} & \frac{\partial v_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \qquad (12.3.2\text{-}5)$$

$$[\nabla^2 F(\mathbf{x})]_{k,j} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_j} = 2\sum_{i=1}^{N}\left\{\frac{\partial v_i(\mathbf{x})}{\partial x_k}\frac{\partial v_i(\mathbf{x})}{\partial x_j} + v_i(\mathbf{x})\frac{\partial^2 v_i(\mathbf{x})}{\partial x_k \partial x_j}\right\} \tag{12.3.2-6}$$

$$\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x}) \tag{12.3.2-7}$$

where

$$\mathbf{S}(\mathbf{x}) = \sum_{i=1}^{N} v_i(\mathbf{x})\nabla^2 v_i(\mathbf{x}) \tag{12.3.2-8}$$

S(x) is assumed small,

$$\nabla^2 F(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) \tag{12.3.2-9}$$

Gauss-Newton method,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1} 2\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \tag{12.3.2-10}$$

- The advantage of Gauss-Newton over the standard Newton's method is that it does not require calculation of second derivatives.
- One problem with the Gauss-Newton method is that the matrix $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ may not be invertible.

Modification to the approximate Hessian matrix,

$$\mathbf{G} = \mathbf{H} + \mu\mathbf{I} \tag{12.3.2-11}$$

$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$: eigenvalues of **H,** $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$: eigenvectors of **H**,

$$\mathbf{G}\mathbf{z}_i = [\mathbf{H} + \mu\mathbf{I}]\mathbf{z}_i = \mathbf{H}\mathbf{z}_i + \mu\mathbf{z}_i = \lambda_i\mathbf{z}_i + \mu\mathbf{z}_i = (\lambda_i + \mu)\mathbf{z}_i \qquad (12.3.2\text{-}12)$$

- The eigenvectors of **G** are the same as the eigenvectors of **H**, and the eigenvalues of **G** are $(\lambda_i + \mu)$.
- **G** can be made positive definite by increasing μ until $(\lambda_i + \mu) > 0$ for all *i*, and therefore the matrix will be invertible.

Levenberg-Marquardt algorithm,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \qquad (12.3.2\text{-}13)$$

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \qquad (12.3.2\text{-}14)$$

- As μ_k is increased it approaches the steepest descent algorithm with small learning rate:

$$\mathbf{x}_{k+1} \cong \mathbf{x}_k - \frac{1}{\mu_k}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - \frac{1}{2\mu_k}\nabla F(\mathbf{x}) \text{ for large } \mu_k. \qquad (12.3.2\text{-}15)$$

- As μ_k is decreased to zero the algorithm becomes Gauss-Newton.

Procedure and Rules in Levenberg-Marquardt algorithm

1. The algorithm begins with μ_k set to some small value (e.g., $\mu_k = 0.01$).
2. If a step does not yield a smaller value for $F(\mathbf{x})$, then the step is repeated with μ_k multiplied by some factor $\upsilon > 1$ (e.g., $\upsilon = 10$). Eventually $F(\mathbf{x})$ should decrease.
3. If a step does produce a smaller value for $F(\mathbf{x})$, then μ_k is divided by υ for the next step.

$$F(\mathbf{x}) = \sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q)^T(\mathbf{t}_q - \mathbf{a}_q) = \sum_{q=1}^{Q}\mathbf{e}_q^T\mathbf{e}_q = \sum_{q=1}^{Q}\sum_{j=1}^{S^M}(e_{j,q})^2 = \sum_{i=1}^{N}(v_i)^2 \qquad (12.3.2\text{-}16)$$

where $e_{j,q}$ is the *j*th element of the error for the *q*th input/target pair.

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1} & \frac{\partial e_{1,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w_{1,1}^1} & \frac{\partial e_{2,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{2,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{2,1}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ \frac{\partial e_{S^M,1}}{\partial w_{1,1}^1} & \frac{\partial e_{S^M,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{S^M,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{S^M,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1} & \frac{\partial e_{1,2}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,2}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,2}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix} \qquad (12.3.2\text{-}17)$$

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial x_l} = \frac{\partial \mathbf{e}_q^T\mathbf{e}_q}{\partial x_l} \qquad (12.3.2\text{-}18)$$

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial x_l} \qquad (12.3.2\text{-}19)$$

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \qquad (12.3.2\text{-}20)$$

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m} \qquad (12.3.2\text{-}21)$$

Marquardt sensitivity,

$$\tilde{s}_{i,h}^{m} \equiv \frac{\partial v_h}{\partial n_{i,q}^{m}} = \frac{\partial e_{k,q}}{\partial n_{i,q}^{m}} \qquad (12.3.2\text{-}22)$$

where $h = (q-1)S^M + k$.

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^{m}} = \frac{\partial e_{k,q}}{\partial n_{i,q}^{m}} \times \frac{\partial n_{i,q}^{m}}{\partial w_{i,j}^{m}} = \tilde{s}_{i,h}^{m} \times \frac{\partial n_{i,q}^{m}}{\partial w_{i,j}^{m}} = \tilde{s}_{i,h}^{m} \times a_{j,q}^{m-1} \qquad (12.3.2\text{-}23)$$

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b_{i}^{m}} = \frac{\partial e_{k,q}}{\partial n_{i,q}^{m}} \times \frac{\partial n_{i,q}^{m}}{\partial b_{i}^{m}} = \tilde{s}_{i,h}^{m} \times \frac{\partial n_{i,q}^{m}}{\partial b_{i}^{m}} = \tilde{s}_{i,h}^{m} \qquad (12.3.2\text{-}24)$$

For the Marquardt sensitivities at the final layer,

$$\tilde{s}_{i,h}^{M} = \frac{\partial v_h}{\partial n_{i,q}^{M}} = \frac{\partial e_{k,q}}{\partial n_{i,q}^{M}} = \frac{\partial (t_{k,q} - a_{k,q}^{M})}{\partial n_{i,q}^{M}} = -\frac{\partial a_{k,q}^{M}}{\partial n_{i,q}^{M}} = \begin{cases} -\dot{f}^{M}(n_{i,q}^{M}), \; for(i=k) \\ 0, \; for(i \neq k) \end{cases} \qquad (12.3.2\text{-}25)$$

$$\tilde{\mathbf{S}}_{q}^{M} = -\dot{\mathbf{F}}^{M}(\mathbf{n}_{q}^{M}) \qquad (12.3.2\text{-}26)$$

$$\tilde{\mathbf{S}}_{q}^{m} = \dot{\mathbf{F}}^{m}(\mathbf{n}_{q}^{m})(\mathbf{W}^{m+1})^{T}\tilde{\mathbf{S}}_{q}^{m+1} \qquad (12.3.2\text{-}27)$$

$$\tilde{\mathbf{S}}^{m} = [\tilde{\mathbf{S}}_{1}^{m} | \tilde{\mathbf{S}}_{2}^{m} | \cdots | \tilde{\mathbf{S}}_{Q}^{m}] \qquad (12.3.2\text{-}28)$$

The iterations of the Levenberg-Marquardt backpropagation algorithm (LMBP),

1. Present all inputs to the network and compute the corresponding network outputs and the errors. Compute the sum of squared errors over all inputs, $F(\mathbf{x})$.
2. Compute the Jacobian matrix. Calculate the sensitivities with the recurrence relations, after initializing. Augment the individual matrices into the Marquardt sensitivities. Compute the elements of the Jacobian matrix.
3. Solve to obtain $\Delta\mathbf{x}_k$.
4. Recompute the sum of squared errors using $\mathbf{x}_k + \Delta\mathbf{x}_k$. If this new sum of squares is smaller than that computed in step 1, then divide μ by υ, let $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$ and go back to step 1. If the sum of squares is not reduced, then multiply μ by υ and go back to step 3.

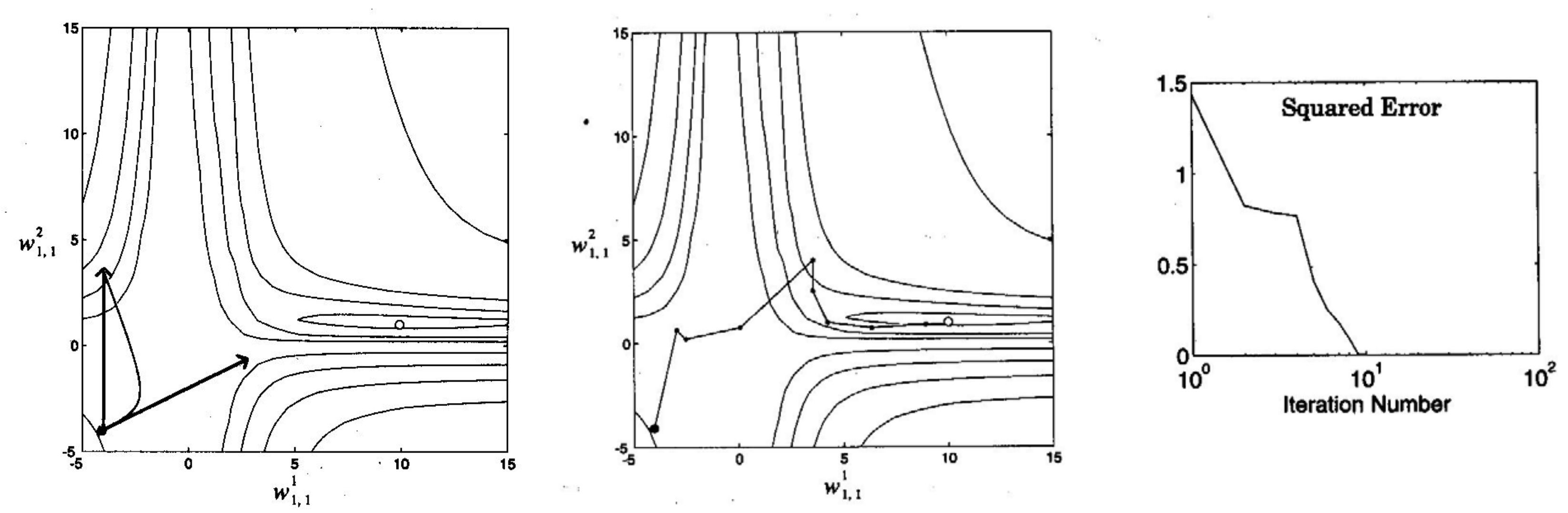
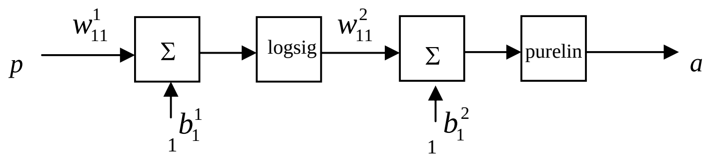


Figure 12.3.2-1 Levenberg-Marquardt Step and Trajectory, $\mu_0 = 0.01$, $\upsilon = 5$

12.3.3 Example of Levenberg-Marquardt Algorithm

A 1-1-1 network is used in a function approximation problem by Levenberg-Marquardt method,



The initial weights and biases,

$$w_{11}^1(0)=0.1, b_1^1(0)=0.2, w_{11}^2(0)=0.3, b_1^2(0)=0.4 \tag{12.3.3-1}$$

The relation between inputs and targets,

$$\{p_1=5,t_1=15\},\{p_2=10,t_2=25\},\{p_3=15,t_3=25\},\{p_4=20,t_4=30\},\{p_5=35,t_5=35\} \tag{12.3.3-2}$$

when $\mu_k=0.01, \upsilon=10$,

$$f^1=(1+e^{-n})^{-1},\ \dot{f}^1=(1-a_1^1)(a_1^1),\ f^2=n,\ \dot{f}^2=1 \tag{12.3.3-3}$$

Present p_1,

$$p = p_1 = 5, n_1^1 = w_{11}^1 p + b_1^1 = 0.1\times 5 + 0.2 = 0.7, a_1^1 = (1+e^{-n_1^1})^{-1} = (1+e^{-0.7})^{-1} = 0.668 \quad (12.3.3\text{-}4)$$

$$n_1^2 = w_{11}^2 a_1^1 + b_1^2 = 0.3\times 0.668 + 0.4 = 0.600, a_1^2 = a = n_1^2 = 0.600, e = e_1 = t - a = 15 - 0.600 = 14.400 \quad (12.3.3\text{-}5)$$

$$\tilde{S}_1^2 = -\dot{f}^2 = -1, \tilde{S}_1^1 = \dot{f}^1 w_{11}^2 \tilde{S}_1^2 = [(1-0.668)0.668](0.3)(-1) = -0.067 \quad (12.3.3\text{-}6)$$

Present p_2,

$$p = p_2 = 10, n_1^1 = w_{11}^1 p + b_1^1 = 0.1\times 10 + 0.2 = 1.2, a_1^1 = (1+e^{-n_1^1})^{-1} = (1+e^{-1.2})^{-1} = 0.769 \quad (12.3.3\text{-}7)$$

$$n_1^2 = w_{11}^2 a_1^1 + b_1^2 = 0.3\times 0.769 + 0.4 = 0.631, a_1^2 = a = n_1^2 = 0.631, e = e_2 = t - a = 25 - 0.631 = 24.369 \quad (12.3.3\text{-}8)$$

$$\tilde{S}_2^2 = -\dot{f}^2 = -1, \tilde{S}_2^1 = \dot{f}^1 w_{11}^2 \tilde{S}_1^2 = [(1-0.769)0.769](0.3)(-1) = -0.053 \quad (12.3.3\text{-}9)$$

Present p_3,

$$p = p_3 = 15, n_1^1 = w_{11}^1 p + b_1^1 = 0.1\times 15 + 0.2 = 1.7, a_1^1 = (1+e^{-n_1^1})^{-1} = (1+e^{-1.7})^{-1} = 0.846 \quad (12.3.3\text{-}10)$$

$$n_1^2 = w_{11}^2 a_1^1 + b_1^2 = 0.3\times 0.846 + 0.4 = 0.654, a_1^2 = a = n_1^2 = 0.654, e = e_3 = t - a = 25 - 0.654 = 24.346 \quad (12.3.3\text{-}11)$$

$$\tilde{S}_3^2 = -\dot{f}^2 = -1, \tilde{S}_3^1 = \dot{f}^1 w_{11}^2 \tilde{S}_1^2 = [(1-0.846)0.846](0.3)(-1) = -0.039 \quad (12.3.3\text{-}12)$$

Present p_4,

$$p = p_4 = 20, n_1^1 = w_{11}^1 p + b_1^1 = 0.1 \times 20 + 0.2 = 2.2, a_1^1 = (1 + e^{-n_1^1})^{-1} = (1 + e^{-2.2})^{-1} = 0.900 \tag{12.3.3-13}$$

$$n_1^2 = w_{11}^2 a_1^1 + b_1^2 = 0.3 \times 0.900 + 0.4 = 0.670, a_1^2 = a = n_1^2 = 0.670, e = e_4 = t - a = 30 - 0.670 = 29.330 \tag{12.3.3-14}$$

$$\tilde{S}_4^2 = -\dot{f}^2 = -1, \tilde{S}_4^1 = \dot{f}^1 w_{11}^2 \tilde{S}_1^2 = [(1 - 0.900)0.900](0.3)(-1) = -0.027 \tag{12.3.3-15}$$

Present p_5,

$$p = p_5 = 35, n_1^1 = w_{11}^1 p + b_1^1 = 0.1 \times 35 + 0.2 = 3.7, a_1^1 = (1 + e^{-n_1^1})^{-1} = (1 + e^{-3.7})^{-1} = 0.976 \tag{12.3.3-16}$$

$$n_1^2 = w_{11}^2 a_1^1 + b_1^2 = 0.3 \times 0.976 + 0.4 = 0.693, a_1^2 = a = n_1^2 = 0.693, e = e_5 = t - a = 35 - 0.693 = 34.307 \tag{12.3.3-17}$$

$$\tilde{S}_5^2 = -\dot{f}^2 = -1, \tilde{S}_5^1 = \dot{f}^1 w_{11}^2 \tilde{S}_1^2 = [(1 - 0.976)0.976](0.3)(-1) = -0.007 \tag{12.3.3-18}$$

Summation of error square,

$$F(x) = \sum e^2 = 14.400^2 + 24.369^2 + 24.346^2 + 29.330^2 + 34.307^2 = 3431.155 \tag{12.3.3-19}$$

$$J(x)=\begin{bmatrix} \frac{\partial e_1}{\partial w_{11}^1} & \frac{\partial e_1}{\partial b_1^1} & \frac{\partial e_1}{\partial w_{11}^2} & \frac{\partial e_1}{\partial b_1^2} \\ \frac{\partial e_2}{\partial w_{11}^1} & \frac{\partial e_2}{\partial b_1^1} & \frac{\partial e_2}{\partial w_{11}^2} & \frac{\partial e_2}{\partial b_1^2} \\ \frac{\partial e_3}{\partial w_{11}^1} & \frac{\partial e_3}{\partial b_1^1} & \frac{\partial e_3}{\partial w_{11}^2} & \frac{\partial e_3}{\partial b_1^2} \\ \frac{\partial e_4}{\partial w_{11}^1} & \frac{\partial e_4}{\partial b_1^1} & \frac{\partial e_4}{\partial w_{11}^2} & \frac{\partial e_4}{\partial b_1^2} \\ \frac{\partial e_5}{\partial w_{11}^1} & \frac{\partial e_5}{\partial b_1^1} & \frac{\partial e_5}{\partial w_{11}^2} & \frac{\partial e_5}{\partial b_1^2} \end{bmatrix} = \begin{bmatrix} -0.067\times 5 & -0.067 & -1\times 0.668 & -1 \\ -0.053\times 10 & -0.053 & -1\times 0.769 & -1 \\ -0.039\times 15 & -0.039 & -1\times 0.846 & -1 \\ -0.027\times 20 & -0.027 & -1\times 0.900 & -1 \\ -0.007\times 35 & -0.007 & -1\times 0.976 & -1 \end{bmatrix} \qquad (12.3.3\text{-}20)$$

$$J(x)=\begin{bmatrix} -0.335 & -0.067 & -0.668 & -1 \\ -0.530 & -0.053 & -0.769 & -1 \\ -0.585 & -0.039 & -0.846 & -1 \\ -0.540 & -0.027 & -0.900 & -1 \\ -0.245 & -0.007 & -0.976 & -1 \end{bmatrix} \qquad (12.3.3\text{-}21)$$

$$J^T J=\begin{bmatrix} -0.335 & -0.067 & -0.668 & -1 \\ -0.530 & -0.053 & -0.769 & -1 \\ -0.585 & -0.039 & -0.846 & -1 \\ -0.540 & -0.027 & -0.900 & -1 \\ -0.245 & -0.007 & -0.976 & -1 \end{bmatrix}^T \begin{bmatrix} -0.335 & -0.067 & -0.668 & -1 \\ -0.530 & -0.053 & -0.769 & -1 \\ -0.585 & -0.039 & -0.846 & -1 \\ -0.540 & -0.027 & -0.900 & -1 \\ -0.245 & -0.007 & -0.976 & -1 \end{bmatrix} = \begin{bmatrix} 1.087 & 0.090 & 1.851 & 2.235 \\ 0.090 & 0.010 & 0.150 & 0.193 \\ 1.851 & 0.150 & 3.516 & 4.159 \\ 2.235 & 0.193 & 4.159 & 5.000 \end{bmatrix} \qquad (12.3.3\text{-}22)$$

$$J^T v = \begin{bmatrix} -0.335 & -0.067 & -0.668 & -1 \\ -0.530 & -0.053 & -0.769 & -1 \\ -0.585 & -0.039 & -0.846 & -1 \\ -0.540 & -0.027 & -0.900 & -1 \\ -0.245 & -0.007 & -0.976 & -1 \end{bmatrix}^T \begin{bmatrix} 14.400 \\ 24.369 \\ 24.346 \\ 29.330 \\ 34.307 \end{bmatrix} = \begin{bmatrix} -56.225 \\ -4.238 \\ -108.836 \\ -126.752 \end{bmatrix} \tag{12.3.3-23}$$

$$\Delta x = -[J^T J + \mu_k I]^{-1} J^T v \tag{12.3.3-24}$$

$$\begin{bmatrix} \Delta w_{11}^1 \\ \Delta b_1^1 \\ \Delta w_{11}^2 \\ \Delta b_1^2 \end{bmatrix} = -\begin{bmatrix} 1.087 + 0.01 & 0.090 & 1.851 & 2.235 \\ 0.090 & 0.010 + 0.01 & 0.150 & 0.193 \\ 1.851 & 0.150 & 3.516 + 0.01 & 4.159 \\ 2.235 & 0.193 & 4.159 & 5.000 + 0.01 \end{bmatrix}^{-1} \begin{bmatrix} -56.225 \\ -4.238 \\ -108.836 \\ -126.752 \end{bmatrix} = \begin{bmatrix} -0.899 \\ -11.364 \\ 47.605 \\ -13.380 \end{bmatrix} \tag{12.3.3-25}$$

$$\begin{bmatrix} w_{11}^1(1) \\ b_1^1(1) \\ w_{11}^2(1) \\ b_1^2(1) \end{bmatrix} = \begin{bmatrix} w_{11}^1(0) \\ b_1^1(0) \\ w_{11}^2(0) \\ b_1^2(0) \end{bmatrix} + \begin{bmatrix} \Delta w_{11}^1 \\ \Delta b_1^1 \\ \Delta w_{11}^2 \\ \Delta b_1^2 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} + \begin{bmatrix} -0.899 \\ -11.364 \\ 47.605 \\ -13.380 \end{bmatrix} = \begin{bmatrix} -0.799 \\ -11.164 \\ 47.905 \\ -12.980 \end{bmatrix} \tag{12.3.3-26}$$

13 Associative Learning

- An association is any link between a system's input and output such that when a pattern A is presented to the system it will respond with pattern B.
- The input pattern is referred to as the stimulus.
- The output pattern is referred to as the response.

13.1 Simple Associative Network

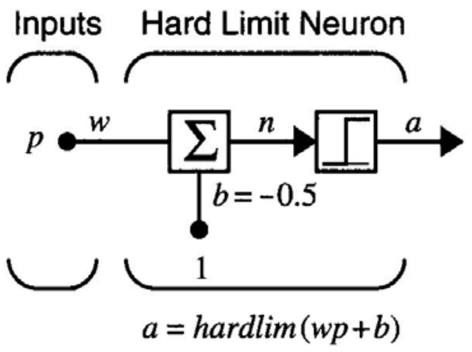


Figure 13.1-1 Single-Input Hard Limit Associator

$$a = hardlim(wp+b) = hardlim(wp-0.5) \tag{13.1-1}$$

$$p = \begin{cases} 1, & stimulus \\ 0, & no\_stimulus \end{cases} \qquad a = \begin{cases} 1, & respone \\ 0, & no\_response \end{cases} \tag{13.1-2}$$

Stimulus:

1. The unconditioned stimulus, $\mathbf{p}^0$
2. The conditioned stimulus, $\mathbf{p}$

- The weights associated with $\mathbf{p}^0$ are fixed, but that the weights associated with $\mathbf{p}$ are adjusted according to the relevant learning rule.

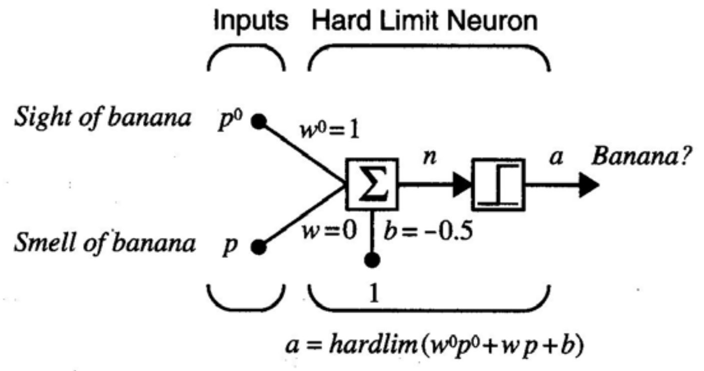


Figure 13.1-2 Banana Associator

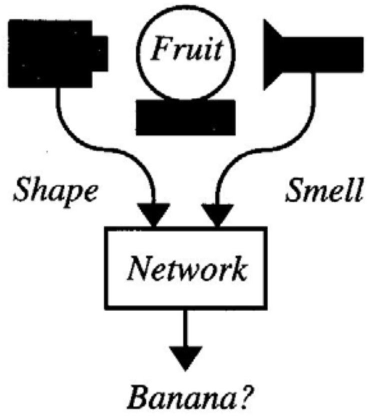


Figure 13.1-3 Bananas Recognition System

Unconditioned Stimulus (Banana Shape) and Conditioned Stimulus (Banana Smell)

The definitions of the unconditioned and conditioned inputs for this network are

$$p^0 = \begin{cases} 1, & shape\_detected \\ 0, & shape\_not\_detected \end{cases} \qquad p = \begin{cases} 1, & smell\_detected \\ 0, & smell\_not\_detected \end{cases} \tag{13.1-3}$$

13.2 Unsupervised Hebb Rule

Unsupervised Hebb rule,

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) \tag{13.2-1}$$

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q) \tag{13.2-2}$$

$$\mathbf{p}(1), \mathbf{p}(2), \dots, \mathbf{p}(Q) \tag{13.2-3}$$

Initial weights in banana associator,

$$w^0 = 1, w(0) = 0 \tag{13.2-4}$$

The training sequence consists of repetitions of two sets of inputs,

$$\{p^0(1) = 0, p(1) = 1\}, \{p^0(2) = 1, p(2) = 1\}, \dots \tag{13.2-5}$$

w^0: the weight for the unconditioned stimulus p^0: constant,

w: the weight for the conditioned stimulus p: updated at each iteration,

Unsupervised Hebb rule with a learning rate of 1,

$$w(q) = w(q-1) + a(q)p(q) \tag{13.2-6}$$

$$a(1) = hardlim(w^0 p^0(1) + w(0)p(1) - 0.5) = hardlim(1 \cdot 0 + 0 \cdot 1 - 0.5) = 0 \text{ (no response)} \tag{13.2-7}$$

$$w(1) = w(0) + a(1)p(1) = 0 + 0 \cdot 1 = 0 \tag{13.2-8}$$

$$a(2) = hardlim(w^0 p^0(2) + w(1)p(2) - 0.5) = hardlim(1 \cdot 1 + 0 \cdot 1 - 0.5) = 1 \text{ (banana)} \qquad (13.2\text{-}9)$$

$$w(2) = w(1) + a(2)p(2) = 0 + 1 \cdot 1 = 1 \qquad (13.2\text{-}10)$$

$$a(3) = hardlim(w^0 p^0(3) + w(2)p(3) - 0.5) = hardlim(1 \cdot 0 + 1 \cdot 1 - 0.5) = 1 \text{ (banana)} \qquad (13.2\text{-}11)$$

$$w(3) = w(2) + a(3)p(3) = 1 + 1 \cdot 1 = 2 \qquad (13.2\text{-}12)$$

- If the inputs are continuously presented and w is updated, the weight w will become arbitrarily large.
- Synapses cannot grow without bound.
- There is no mechanism for weights to decrease. Every weight will grow until the network responds to any stimulus.

13.2.1 Hebb Rule with Decay

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha\mathbf{a}(q)\mathbf{p}^T(q) - \gamma\mathbf{W}(q-1) = (1-\gamma)\mathbf{W}(q-1) + \alpha\mathbf{a}(q)\mathbf{p}^T(q) \qquad (13.2.1\text{-}1)$$

γ: decay rate, a positive constant less than one

- As γ approaches zero, the learning law becomes the standard rule.
- As γ approaches one, the learning law quickly forgets old inputs and remembers only the most recent patterns.

The maximum weight value $w_{ij}^{\max}$,

$$w_{ij} = (1-\gamma)w_{ij} + \alpha a_i p_j = (1-\gamma)w_{ij} + \alpha, \text{ or } w_{ij} = \frac{\alpha}{\gamma} \qquad (13.2.1\text{-}2)$$

With γ of 0.1 in banana associator,

$$a(1) = hardlim(w^0 p^0(1) + w(0)p(1) - 0.5) = hardlim(1\cdot 0 + 0\cdot 1 - 0.5) = 0 \text{ (no response)} \qquad (13.2.1\text{-}3)$$

$$w(1) = w(0) + a(1)p(1) - \gamma w(0) = 0 + 0\cdot 1 + 0 = 0 \qquad (13.2.1\text{-}4)$$

$$a(2) = hardlim(w^0 p^0(2) + w(1)p(2) - 0.5) = hardlim(1\cdot 1 + 0\cdot 1 - 0.5) = 1 \text{ (banana)} \qquad (13.2.1\text{-}5)$$

$$w(2) = w(1) + a(2)p(2) - \gamma w(1) = 0 + 1\cdot 1 - 0 = 1 \qquad (13.2.1\text{-}6)$$

$$a(3) = hardlim(w^0 p^0(3) + w(2)p(3) - 0.5) = hardlim(1\cdot 0 + 1\cdot 1 - 0.5) = 1 \text{ (banana)} \qquad (13.2.1\text{-}7)$$

$$w(3) = w(2) + a(3)p(3) - \gamma w(2) = 1 + 1\cdot 1 - 0.1\cdot 1 = 1.9 \qquad (13.2.1\text{-}8)$$

$$w_{ij}^{\max} = \frac{\alpha}{\gamma} = \frac{1}{0.1} = 10 \tag{13.2.1-9}$$

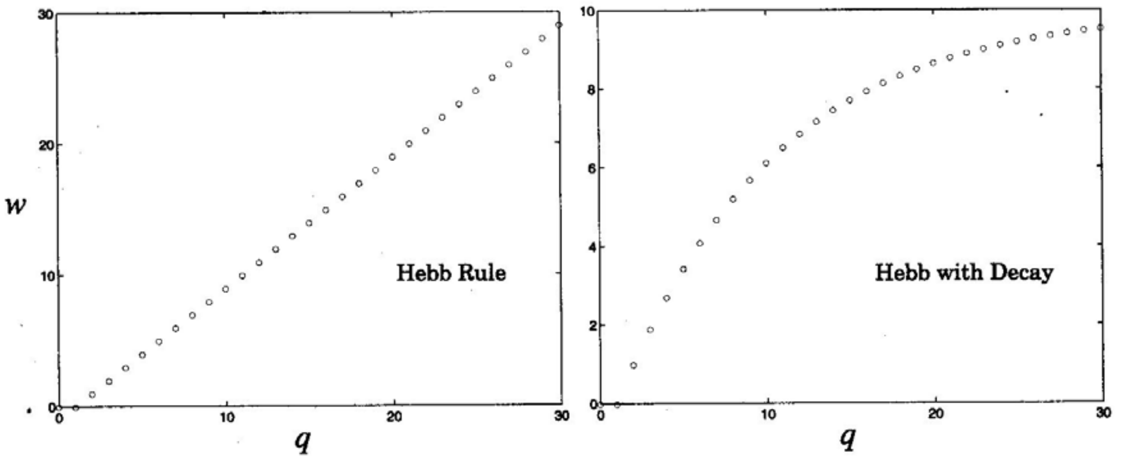


Figure 13.2.1-1 Response of Hebb Rule, With and Without Decay

Without reinforcement, a_i = 0, associations decays away.

$$w_{ij}(q) = (1-\gamma)w_{ij}(q-1) \tag{13.2.1-8}$$

γ = 0.1,

$$w_{ij}(q) = 0.9w_{ij}(q-1) \tag{13.2.1-9}$$

13.3 Simple Recognition Network

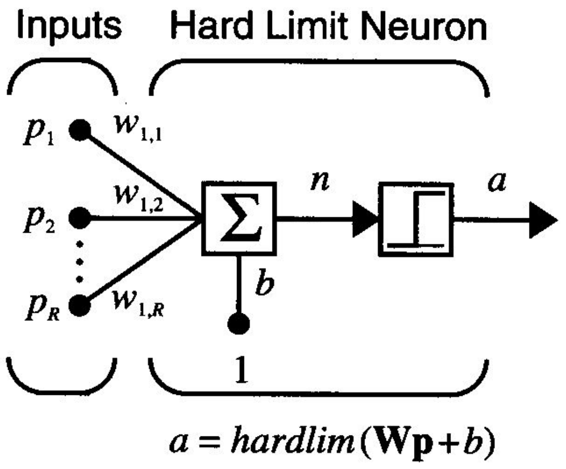


Figure 13.3-1 Instar

$$a = hardlim(\mathbf{Wp} + b) = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) \tag{13.3-1}$$

The instar is active whenever the inner product between the weight vector and the input is greater than or equal to $-b$:

$${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\|\|\mathbf{p}\|\cos\theta \geq -b \tag{13.3-2}$$

θ: the angle between two vectors.

- The inner product is maximized when the angle θ is 0.
- The instar will be active when **p** is "close" to $_1\mathbf{w}$.
- By setting the bias *b* appropriately, we can select how close the input vector must be to the weight vector in order to activate the instar.
- The larger the value of *b*, the more patterns there will be that can activate the instar, thus making it the less discriminatory.

For the neuron that recognizes only the pattern $_1\mathbf{w}$

$$b = -\|{}_1\mathbf{w}\|\|\mathbf{p}\| \tag{13.3-3}$$

13.4 Instar Rule

The original unsupervised Hebb rule,

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) \tag{13.4-1}$$

To get the benefits of weight decay, while limiting the forgetting problem, a decay term is proportional to $a_i(q)$.

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) - \gamma a_i(q) w_{ij}^{old} \tag{13.4-2}$$

In instar rule, $\gamma = \alpha$,

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q)(p_j(q) - w_{ij}^{old}) \tag{13.4-3}$$

$$_i\mathbf{w}(q) = {_i\mathbf{w}}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - {_i\mathbf{w}}(q-1)) \tag{13.4-4}$$

If the instar is active ($a_i = 1$),

$$_i\mathbf{w}(q) = {_i\mathbf{w}}(q-1) + \alpha(\mathbf{p}(q) - {_i\mathbf{w}}(q-1)) = (1-\alpha)\,{_i\mathbf{w}}(q-1) + \alpha\mathbf{p}(q) \tag{13.4-5}$$

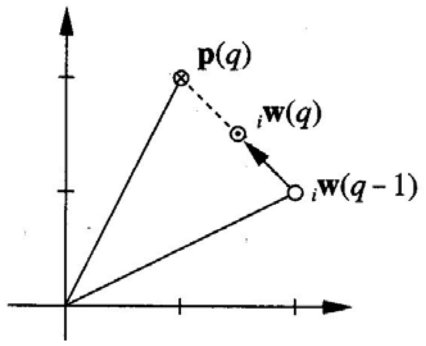


Figure 13.4-1 Graphical Representation of the Instar Rule

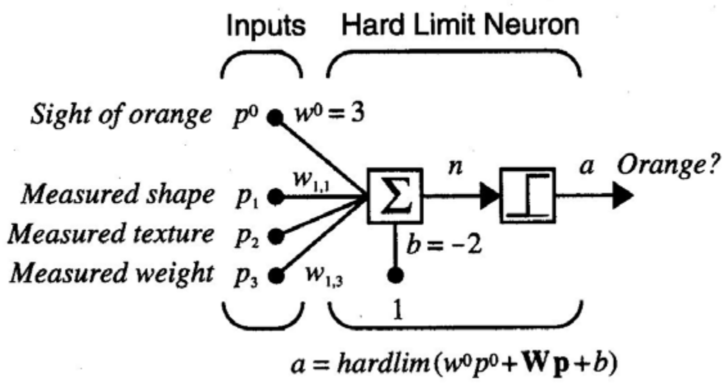


Figure 13.4-2 Orange Recognizer

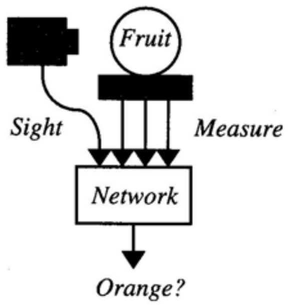


Figure 13.4-3 Orange Recogniton System

$$a = hardlim(w^0 p^0 + \mathbf{W}\mathbf{p} + b) \qquad (13.4\text{-}6)$$

$$p^0 = \begin{cases} 1, & orange\_detected\_visually \\ 0, & orange\_not\_detected \end{cases} \qquad \mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \qquad (13.4\text{-}7)$$

$$w^0 = 3, \mathbf{W}(0) = {}_1\mathbf{w}^T(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \qquad (13.4\text{-}8)$$

Learning rate of $\alpha = 1$,

$${}_1\mathbf{w}(q) = {}_1\mathbf{w}(q-1) + a(q)(\mathbf{p}(q) - {}_1\mathbf{w}(q-1)) \qquad (13.4\text{-}8)$$

$$\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \left\{ p^0(2) = 1, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \ldots \qquad (13.4\text{-}9)$$

$$a(1) = hardlim(w^0 p^0(1) + \mathbf{W}\mathbf{p}(1) - 2) = hardlim\left(3 \cdot 0 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2 \right) = 0 \text{ (no response)} \qquad (13.4\text{-}10)$$

$${}_1\mathbf{w}(1) = {}_1\mathbf{w}(0) + a(1)(\mathbf{p}(1) - {}_1\mathbf{w}(0)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad (13.4\text{-}11)$$

$$a(2) = hardlim(w^0 p^0(2) + \mathbf{Wp}(2) - 2) = hardlim\left(3 \cdot 1 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \text{ (orange)} \qquad (13.4\text{-}12)$$

$$_1\mathbf{w}(2) = {}_1\mathbf{w}(1) + a(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \qquad (13.4\text{-}13)$$

$$a(3) = hardlim(w^0 p^0(3) + \mathbf{Wp}(3) - 2) = hardlim\left(3 \cdot 0 + \begin{bmatrix} 1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \text{ (orange)} \qquad (13.4\text{-}14)$$

$$_1\mathbf{w}(3) = {}_1\mathbf{w}(2) + a(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2)) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \qquad (13.4\text{-}15)$$

13.5 Simple Recall Network

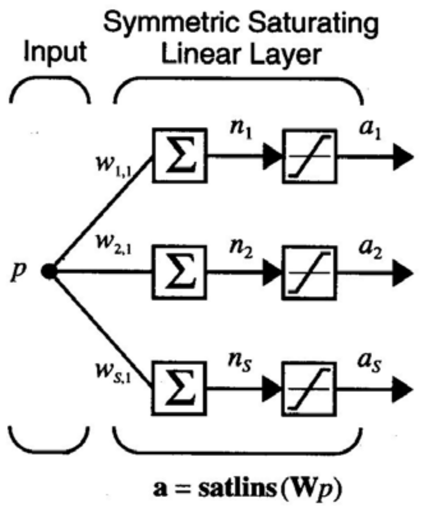

Input
Symmetric Saturating Linear Layer
$w_{1,1}$
$w_{2,1}$
$w_{S,1}$
p
n_1
n_2
n_S
a_1
a_2
a_S
$\mathbf{a} = \mathbf{satlins}(\mathbf{W}p)$

Figure 13.5-1 Outstar Network

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}p) \tag{13.5-1}$$

If we would like the network to associate a stimulus (an input of 1) with a particular output vector **a***, we can simply set **W** (which contains only a single column vector) equal to **a***. Then, if *p* is 1, the output will be **a***:

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}\mathrm{p}) = \mathbf{satlins}(\mathbf{a}^*\cdot 1) = \mathbf{a}^* \tag{13.5-2}$$

13.6 Outstar Rule

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) - \gamma p_j(q) w_{ij}(q-1) \qquad (13.6\text{-}1)$$

In outstar rule, $\gamma = \alpha$,

$$w_{ij}(q) = w_{ij}(q-1) + \alpha(a_i(q) - w_{ij}(q-1)) p_j(q) \qquad (13.6\text{-}2)$$

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha(\mathbf{a}(q) - \mathbf{w}_j(q-1)) p_j(q) \qquad (13.6\text{-}3)$$

If the outstar is stimulated ($p_i = 1$),

$$_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{a}(q) - {}_i\mathbf{w}(q-1)) = (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{a}(q) \qquad (13.6\text{-}4)$$

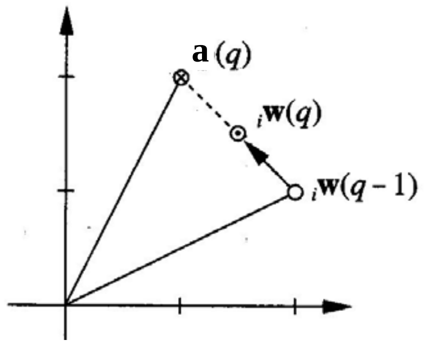


Figure 13.6-1 Graphical Representation of the Outstar Rule

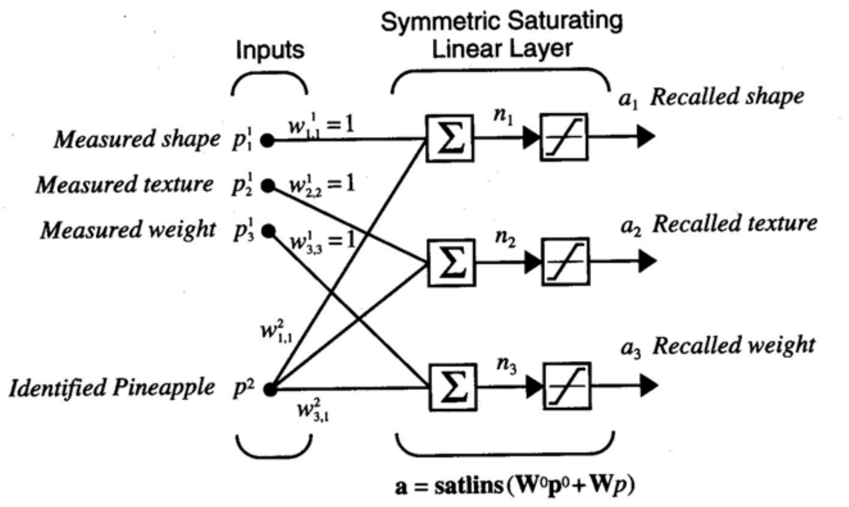


Figure 13.6-1 Pineapple Recaller

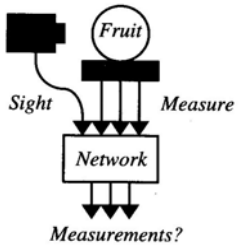


Figure 13.6-2 Pineapple Recalling System

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}^0\mathbf{p}^0+\mathbf{W}p) \tag{13.6-4}$$

$$\mathbf{W}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13.6-5}$$

$$\mathbf{p}^0 = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \qquad p = \begin{cases} 1, & if\_a\_pineappe\_can\_be\_seen \\ 0, & otherwise \end{cases} \tag{13.6-6}$$

Learning rate of α = 1,

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + (\mathbf{a}(q) - \mathbf{w}_j(q-1))\,p_j(q) \tag{13.6-7}$$

$$\mathbf{p}^{pineapple} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \tag{13.6-8}$$

$$\left\{\mathbf{p}^0(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, p(1) = 1\right\}, \left\{\mathbf{p}^0(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, p(2) = 1\right\}, \ldots \tag{13.6-9}$$

$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{W}^0\mathbf{p}^0(1)+\mathbf{W}p(1)) = \mathbf{satlins}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}+\begin{bmatrix}0\\0\\0\end{bmatrix}1\right)=\begin{bmatrix}0\\0\\0\end{bmatrix} \text{ (no response)} \qquad (13.6\text{-}10)$$

$$\mathbf{w}_1(1)=\mathbf{w}_1(0)+(\mathbf{a}(1)-\mathbf{w}_1(0))\,p(1)=\begin{bmatrix}0\\0\\0\end{bmatrix}+\left(\begin{bmatrix}0\\0\\0\end{bmatrix}-\begin{bmatrix}0\\0\\0\end{bmatrix}\right)1=\begin{bmatrix}0\\0\\0\end{bmatrix} \qquad (13.6\text{-}11)$$

$$\mathbf{a}(2)=\mathbf{satlins}\left(\begin{bmatrix}-1\\-1\\1\end{bmatrix}+\begin{bmatrix}0\\0\\0\end{bmatrix}1\right)=\begin{bmatrix}-1\\-1\\1\end{bmatrix} \text{ (measurements given),} \qquad (13.6\text{-}12)$$

$$\mathbf{w}_1(2)=\mathbf{w}_1(1)+(\mathbf{a}(2)-\mathbf{w}_1(1))\,p(2)=\begin{bmatrix}0\\0\\0\end{bmatrix}+\left(\begin{bmatrix}-1\\-1\\1\end{bmatrix}-\begin{bmatrix}0\\0\\0\end{bmatrix}\right)1=\begin{bmatrix}-1\\-1\\1\end{bmatrix} \qquad (13.6\text{-}13)$$

$$\mathbf{a}(3)=\mathbf{satlins}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}+\begin{bmatrix}-1\\-1\\1\end{bmatrix}1\right)=\begin{bmatrix}-1\\-1\\1\end{bmatrix} \text{ (measurements recalled)} \qquad (13.6\text{-}14)$$

$$\mathbf{w}_1(3)=\mathbf{w}_1(2)+(\mathbf{a}(3)-\mathbf{w}_1(2))\,p(3)=\begin{bmatrix}-1\\-1\\1\end{bmatrix}+\left(\begin{bmatrix}-1\\-1\\1\end{bmatrix}-\begin{bmatrix}-1\\-1\\1\end{bmatrix}\right)1=\begin{bmatrix}-1\\-1\\1\end{bmatrix} \qquad (13.6\text{-}15)$$