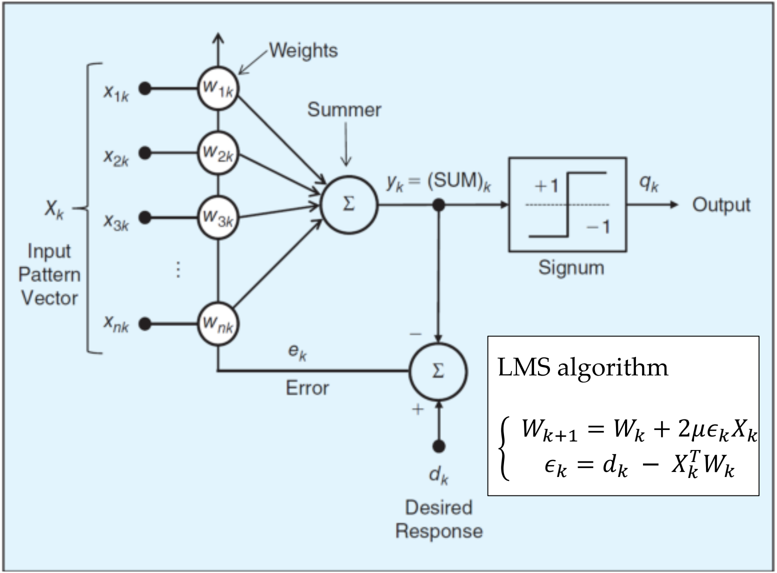
A Brief History of the Development of Artificial Neural Networks

Prof. Bernard Widrow

Department of Electrical Engineering
Stanford University

Baidu • July 18, 2018

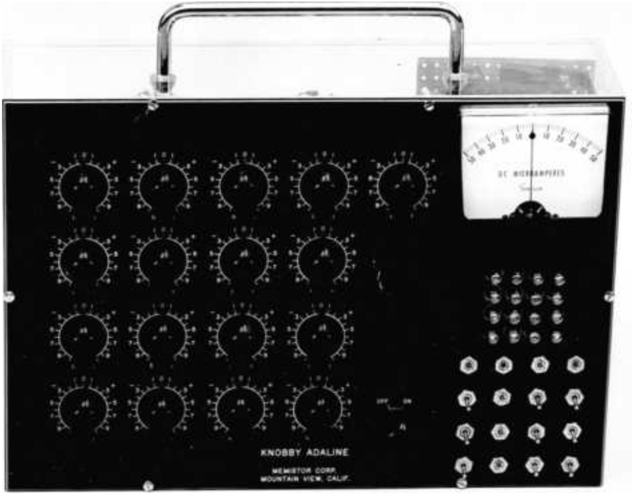
Adaptive linear neuron (Adaline)



LMS algorithm

$$\begin{cases} W_{k+1} = W_k + 2\mu\epsilon_k X_k \\ \epsilon_k = d_k - X_k^T W_k \end{cases}$$

Knobby Adaline, 1959



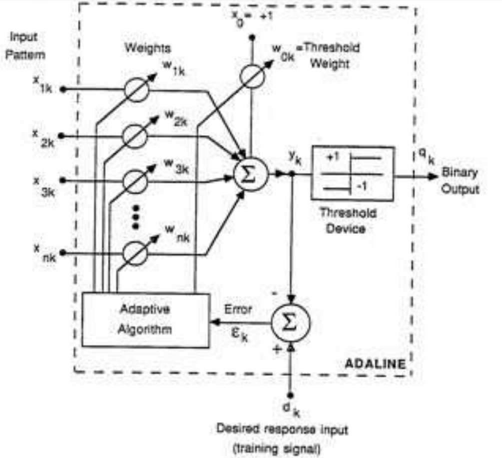
30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation

Bernard Widrow   Michael A. Lehr
Stanford University Department of Electrical Engineering,
Stanford, CA 94305-4055

Abstract

Fundamental developments in feedforward artificial neural networks from the past thirty years are reviewed. The central theme of this paper is a description of the history, origination, operating characteristics, and basic theory of several supervised neural network training algorithms including the Perceptron rule, the LMS algorithm, three Madaline rules, and the backpropagation technique. These methods were developed independently, but with the perspective of history they can all be related to each other. The concept which underlies these algorithms is the "minimal disturbance principle," which suggests that during training it is advisable to inject new information into a network in a manner which disturbs existing information to the smallest extent possible.

An adaptive linear neuron(ADALINE)



$$X_k = [x_{0k}, x_{1k}, x_{2k}, \ldots\ldots x_{nk}]^T$$
$$= [+1, x_{1k}, x_{2k}, \ldots\ldots x_{nk}]^T$$

$$W_k = [w_{0k}, w_{1k}, w_{2k}, \ldots\ldots w_{nk}]^T$$

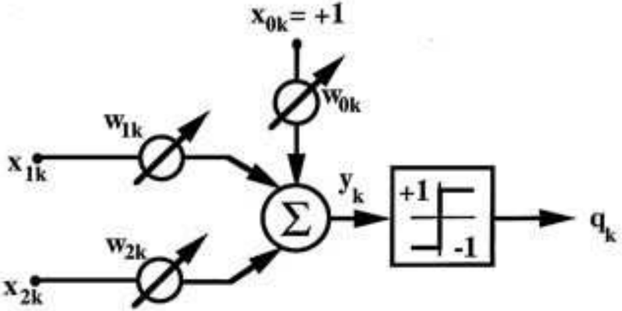
$$y_k = X_k^T W_k = W_k^T X_k$$

$$\varepsilon_k = d_k - y_k$$

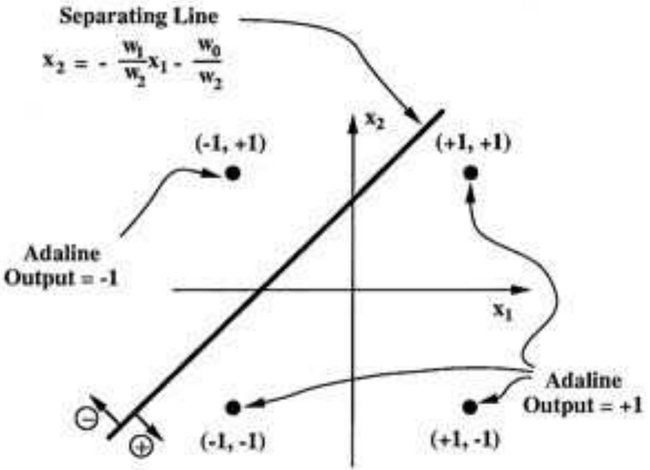
$_k$ is time index

A Two-Input Adaline

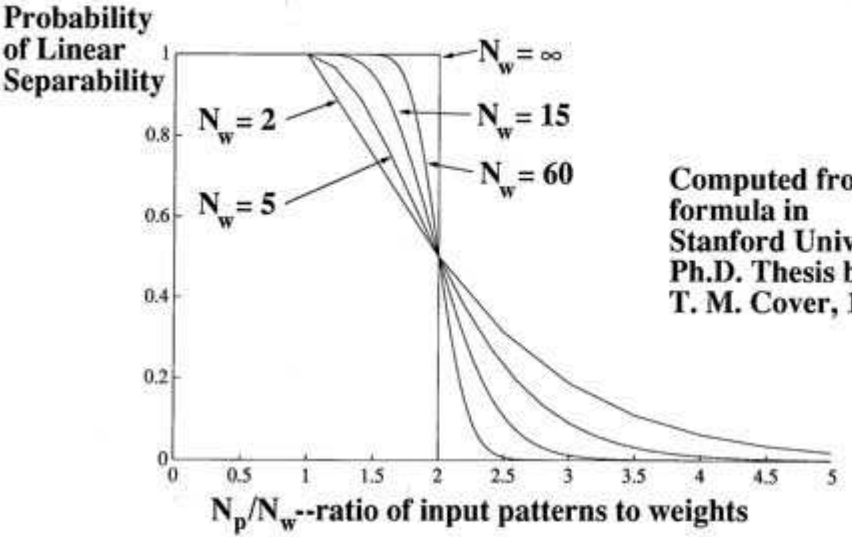
$$y = x_1 w_1 + x_2 w_2 + w_0 = 0$$
$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



Separating line in pattern space

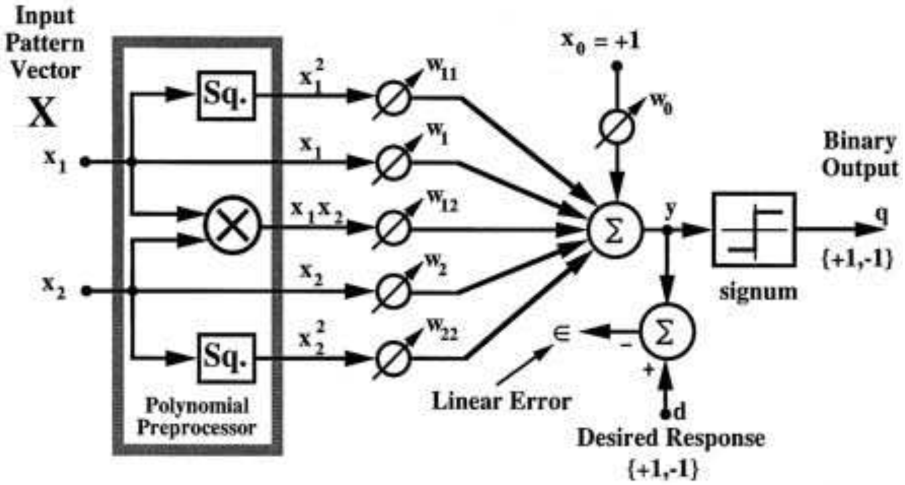


Adaline Capacity

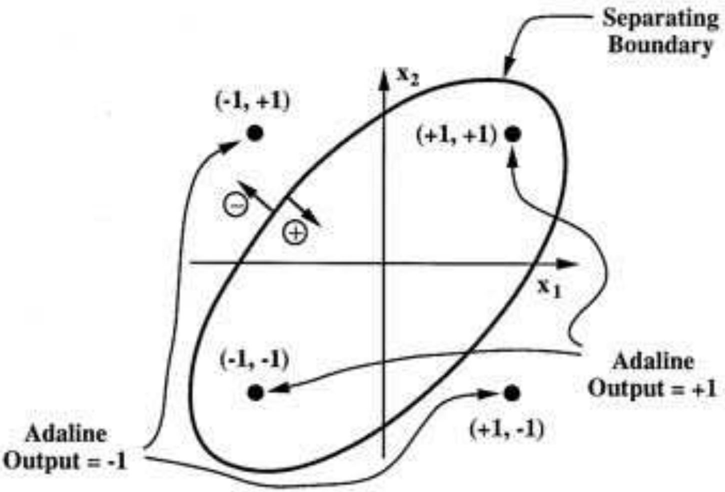


Computed from formula in Stanford Univ. Ph.D. Thesis by T. M. Cover, 1964

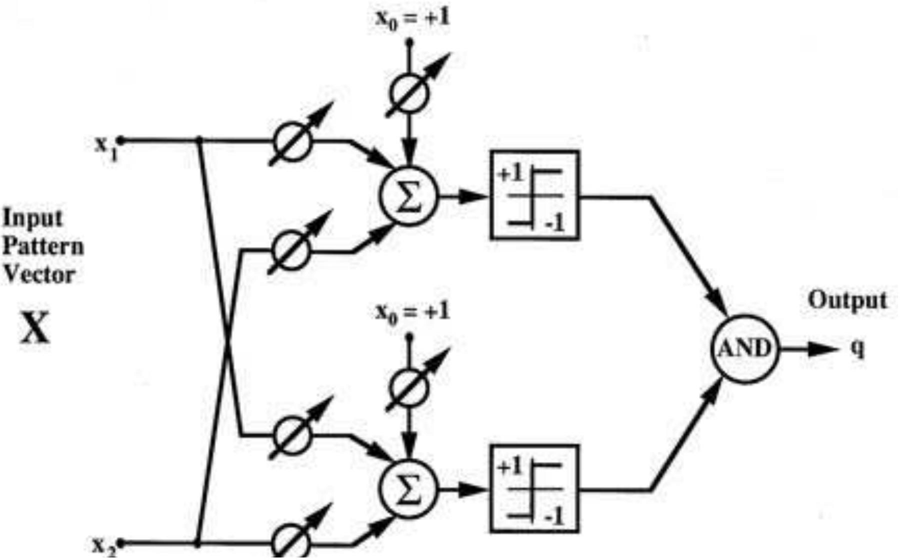
Adaline with polynomial preprocessor



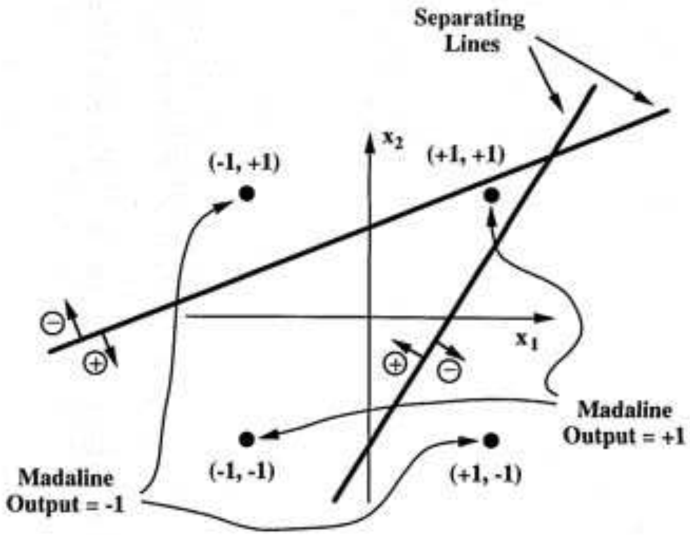
An elliptical separating boundary for the Exclusive NOR Function



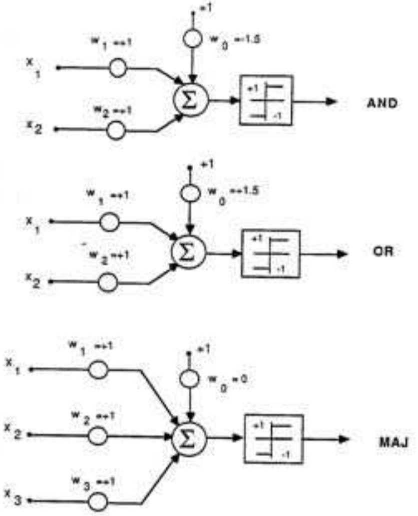
A two-Adaline form of Madaline I



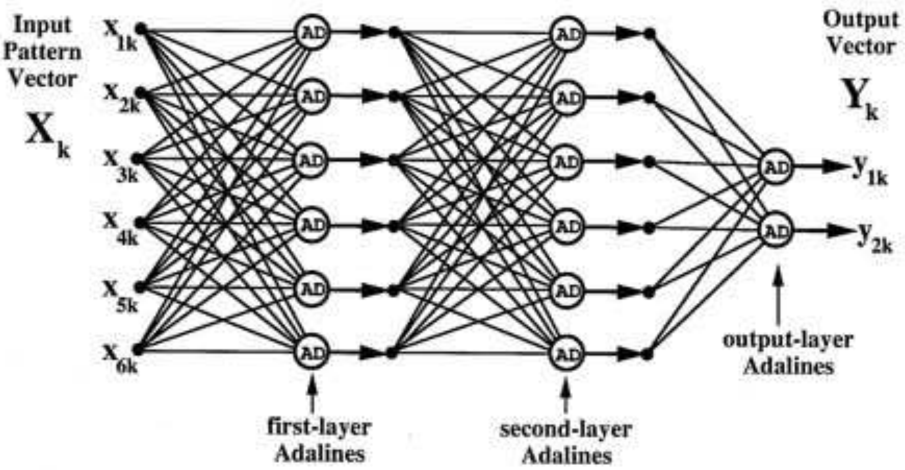
Separating lines for the two-element Madaline



A neuronal implementation of AND, OR, and MAJ logic function



A three-layer adaptive neural network



Principal of Minimal Disturbance

Adapt to reduce the output error for the current training pattern with minimal disturbance to the responses already learned.

Error-Correction Algorithms for the Single Element

1. Linear:

α-LMS

2. Nonlinear:

- Perceptron Rule
- Mays's Rules

α-LMS Algorithm

$$\epsilon_k \triangleq d_k - \mathbf{w}_k^T \mathbf{x}_k. \tag{1}$$

Changing the weights yields a corresponding change in the error:

$$\Delta\epsilon_k = \Delta(d_k - \mathbf{w}_k^T \mathbf{x}_k) = -\mathbf{x}_k^T \Delta\mathbf{w}_k. \tag{2}$$

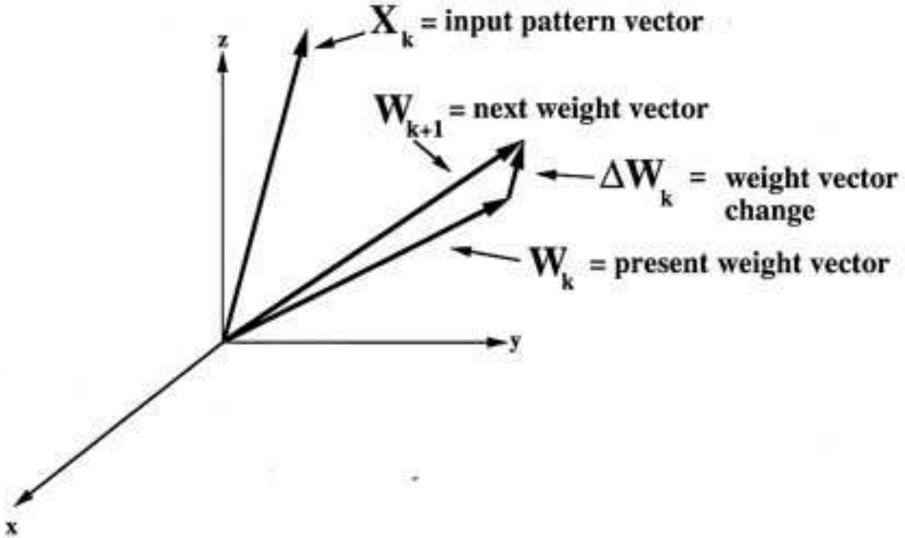
In accordance with the α-LMS rule, the weight change is as follows:

$$\Delta\mathbf{w}_k = \mathbf{w}_{k+1} - \mathbf{w}_k = \alpha \frac{\epsilon_k \mathbf{x}_k}{|\mathbf{x}_k|^2}. \tag{3}$$

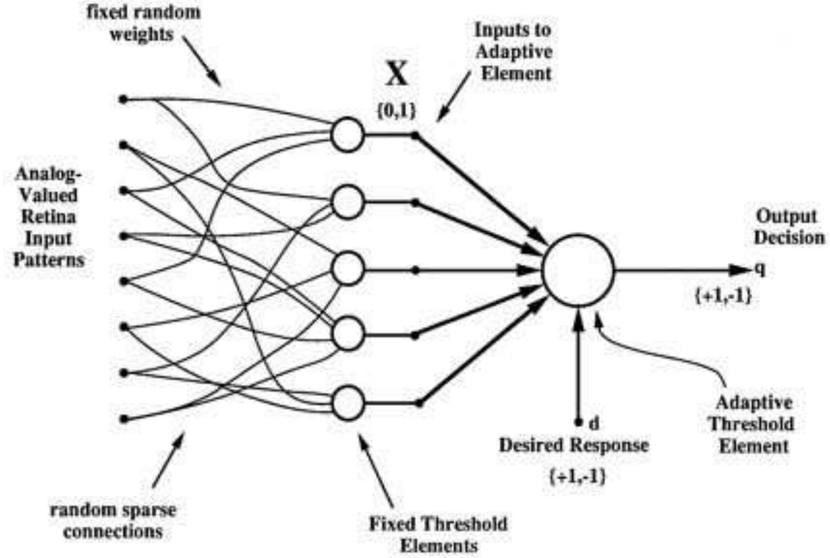
Combining Eqs. (2) and (3), we obtain

$$\Delta\epsilon_k = -\alpha \frac{\epsilon_k \mathbf{x}_k^T \mathbf{x}_k}{|\mathbf{x}_k|^2} = -\alpha\epsilon_k. \tag{4}$$

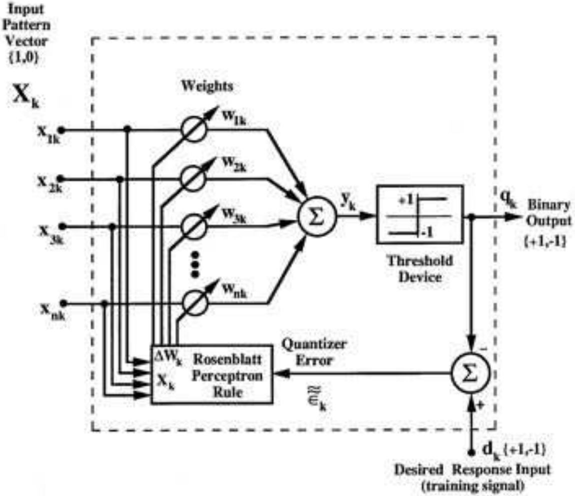
Weight correction by the LMS rule



Rosenblatt's Perceptron



Adaptive Threshold Element in the Perceptron



Perceptron Rule:

$$W_{k+1} = W_k + \mu \tilde{\tilde{\epsilon}}_k X_k$$

μ normally set to 1/2

Perceptron Rule

- If response is OK, do not adapt weights.
- Otherwise adapt weights by a fixed distance along the X-Vector to reduce error

Good Features

- Guaranteed to converge to solution if problem is linearly separable

Bad Features

- Performs poorly if training set is not linearly separable.

May’s Rule

Mays’s Increment Adaptation Rule

$$\mathbf{W}_{k+1} = \begin{cases} \mathbf{W}_k + \alpha \, \tilde{\tilde{\epsilon}}_k \dfrac{\mathbf{X}_k}{2|\mathbf{X}_k|^2} & \text{if } |s_k| \geq \gamma \\ \mathbf{W}_k + \alpha d_k \dfrac{\mathbf{X}_k}{|\mathbf{X}_k|^2} & \text{if } |s_k| < \gamma \end{cases}, \qquad (1)$$

Mays’s Modified Relaxation Rule

$$\mathbf{W}_{k+1} = \begin{cases} \mathbf{W}_k & \text{if } \tilde{\tilde{\epsilon}}_k = 0 \text{ and } |s_k| \geq \gamma \\ \mathbf{W}_k + \alpha \epsilon_k \dfrac{\mathbf{X}_k}{|\mathbf{X}_k|^2} & \text{otherwise} \end{cases}, \qquad (2)$$

C. H. Mays Stanford Univ. Ph.D. Dissertation:

Adaptive Threshold Logic, 1963

May's Increment Adaptation Rule

- If linear output of Adaline falls outside dead zone, adapt weights by Perceptron Rule.
- If linear output of Adaline falls inside dead zone, adapt weights by the Perceptron Rule as though the response were incorrect, whether or not this is the case.

Good Features

- Guaranteed to converge to solution if problem is linearly separable.
- Solutions less sensitive to weight perturbations than those of the Perceptron Rule.
- When the training set is not linearly separable, usually achieves better solution (fewer errors) than that of the Perceptron Rule.

Bad Features

- When the training set is linearly separable, generally takes slightly longer to converge than the Perceptron Rule

May’s Modified Relaxation Rule

- If response is correct and linear output of Adaline falls outside dead zone, do not adapt.
- Otherwise, adapt weights by α-LMS.

Good Features

- Same as Mays's Increment Adaptation Rule.

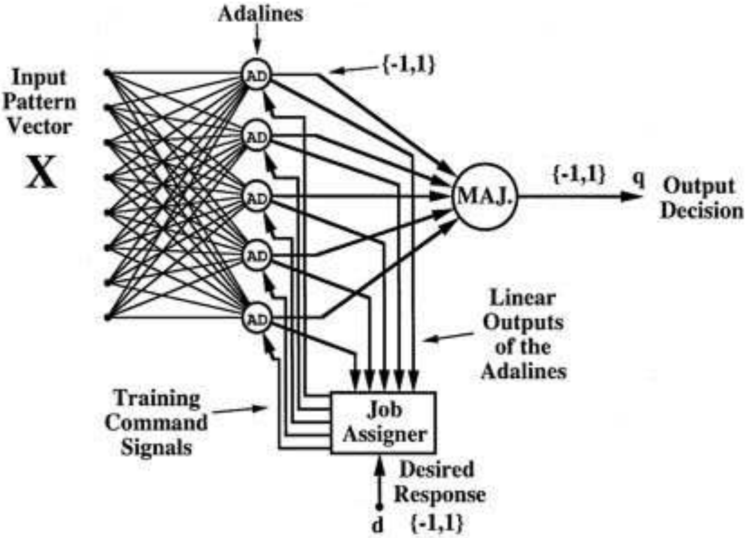
Bad Features

- Same as Mays's Increment Adaptation Rule.

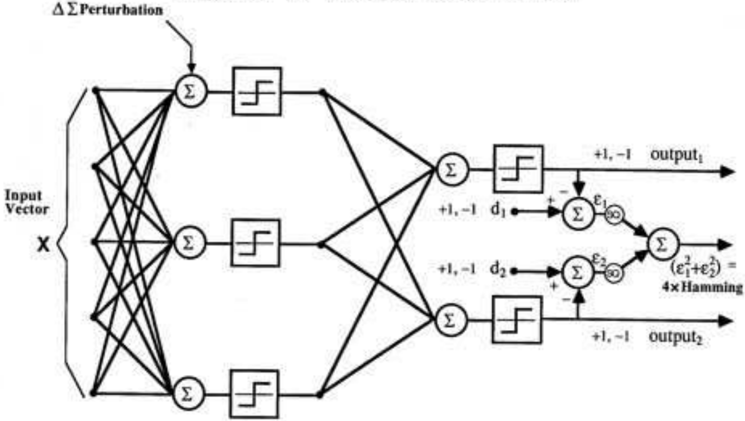
Error-Correction Algorithms for Multi-Element Networks

- Madaline Rule I (MRI)
- Madaline Rule II (MRII)

A five-Adaline example of the MR I Architecture



MR II of B. Widrow and R. Winter



For each layer, beginning with layer 1:

Toggle output of neuron with sum closest to zero. If output Hamming error is reduced, adapt neuron. Repeat for neuron whose sum is next closest to zero, etc. Can also adapt two at a time, etc. Adaptation reduces Hamming error.

Steepest-Descent Algorithms for the Single Element

1. **Linear:**

    μ-LMS

2. **Nonlinear:**

    - Backpropagation for the single element
    - MRIII for the single element

Mean Square Error Surface

$$\epsilon_k^2 = (d_k - X_k^T W_k)^2 = d_k^2 - 2 d_k X_k^T W_k + W_k^T X_k X_k^T W_k$$

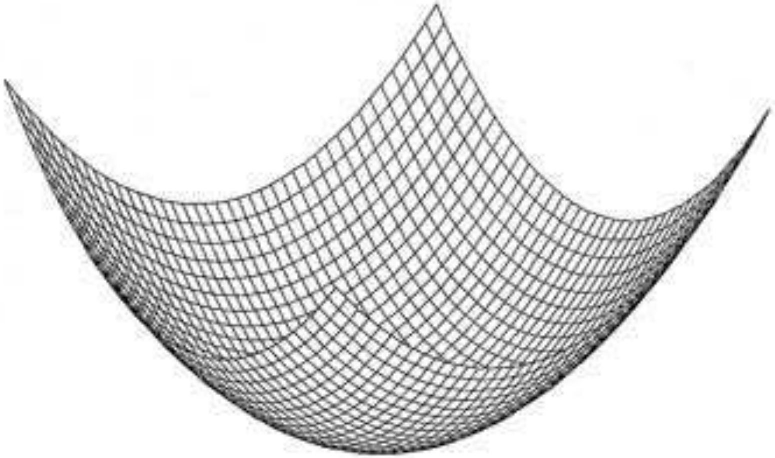
$$E\left[\epsilon_k^2\right]_{W=W_k} = E\left[d_k^2\right] - 2E\left[d_k X_k^T\right] W_k + W_k^T E\left[X_k X_k^T\right] W_k$$

$$P^T \triangleq E\left[d_k X_k^T\right] = E\left[d_k, d_k x_{1k}, \ldots d_k x_{nk}\right]^T$$

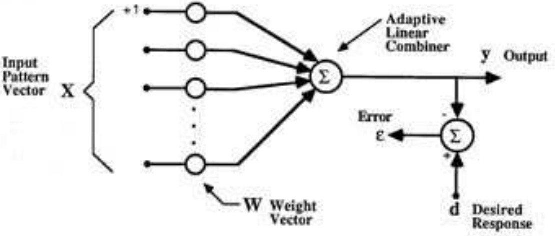
$$R \triangleq E\left[X_k X_k^T\right] = E\begin{bmatrix} 1 & x_{1k} & \ldots & x_{nk} \\ x_{1k} & x_{1k}x_{1k} & \ldots & x_{1k}x_{nk} \\ \vdots & \vdots & & \vdots \\ x_{nk} & x_{nk}x_{1k} & \ldots & x_{nk}x_{nk} \end{bmatrix}$$

$$\xi_k \triangleq E\left[\epsilon_k^2\right]_{W=W_k} = E\left[d_k^2\right] - 2P^T W_k + W_k^T R W_k$$

Mean Square Error Surface



Conventional LMS



Method of Steepest Descent $\rightarrow \mathbf{W}_{k+1} = \mathbf{W}_k + \mu(-\nabla_k)$

$$\text{GRAD.} = \nabla = \frac{\partial E[\varepsilon^2]}{\partial \mathbf{W}}$$

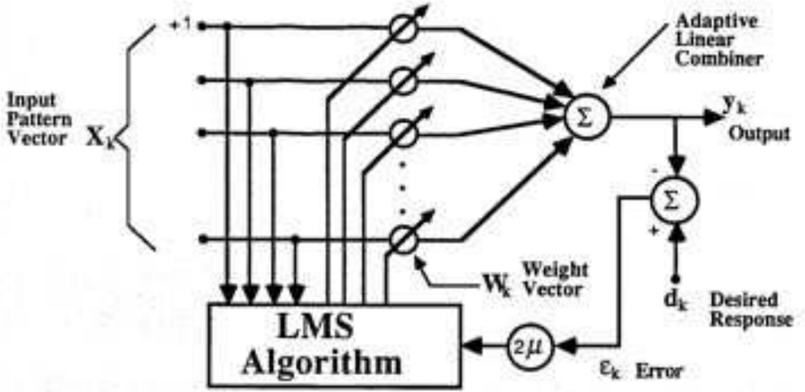
$$\text{ERROR} = \varepsilon = d - \mathbf{X}^T\mathbf{W}$$

$$\text{INST. GRAD.} = \hat{\nabla} = \frac{\partial \varepsilon^2}{\partial \mathbf{W}} = 2\varepsilon\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2\varepsilon\mathbf{X}$$

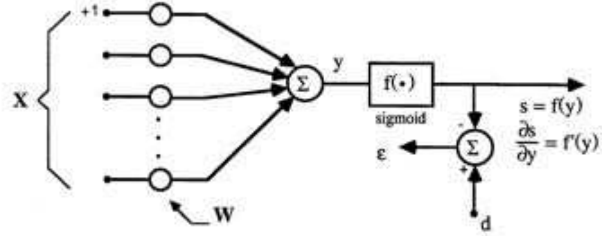
$$\text{LMS} \rightarrow \mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu\varepsilon_k\mathbf{X}_k$$

Implementation of Conventional LMS

$$\text{LMS} \rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k$$



“Sigmoid” LMS(Back-Prop)



GRAD. $= \nabla = \dfrac{\partial E[\varepsilon^2]}{\partial W}$

ERROR $= \varepsilon = d - f(X^T W)$

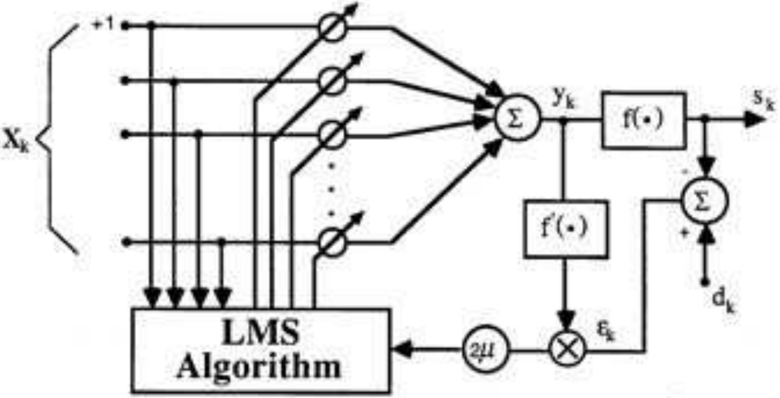
INST. GRAD. =

$$\hat{\nabla} = \frac{\partial \varepsilon^2}{\partial W} = 2\varepsilon \frac{\partial \varepsilon}{\partial W} = -2\varepsilon f'(X^T W)\frac{\partial (X^T W)}{\partial W} = -2\varepsilon X f'(X^T W)$$

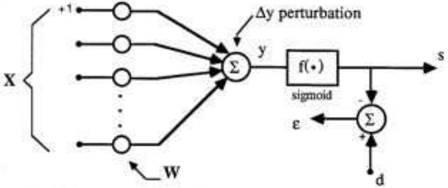
SIGMOID LMS (BACK-PROP) $\rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k f'(X_k^T W_k)$

Implementation of “Sigmoid” LMS(Back-Prop)

$$\text{SIGMOID LMS} \rightarrow W_{k+1} = W_k + 2\mu\varepsilon_k X_k f'(X_k^T W_k)$$



“Sigmoid” LMS(MR III)



$$\text{GRAD.} = \nabla = \frac{\partial E[\varepsilon^2]}{\partial W}$$

$$y = X^T W$$

INST. GRAD. =

$$\hat{\nabla} = \frac{\partial \varepsilon^2}{\partial W} = \frac{\partial \varepsilon^2}{\partial y} \cdot \frac{\partial y}{\partial W} = X \frac{\partial \varepsilon^2}{\partial y} \approx X \frac{\Delta \varepsilon^2}{\Delta y}$$

or

$$\hat{\nabla} = \frac{\partial \varepsilon^2}{\partial W} = 2\varepsilon \frac{\partial \varepsilon}{\partial W} = 2\varepsilon \frac{\partial \varepsilon}{\partial y} \cdot \frac{\partial y}{\partial W} = 2\varepsilon X \frac{\partial \varepsilon}{\partial y} \approx 2\varepsilon X \frac{\Delta \varepsilon}{\Delta y}$$

SIGMOID LMS (MRIII) → $W_{k+1} = W_k - \mu X_k \left(\frac{\Delta \varepsilon^2}{\Delta y}\right)_k$

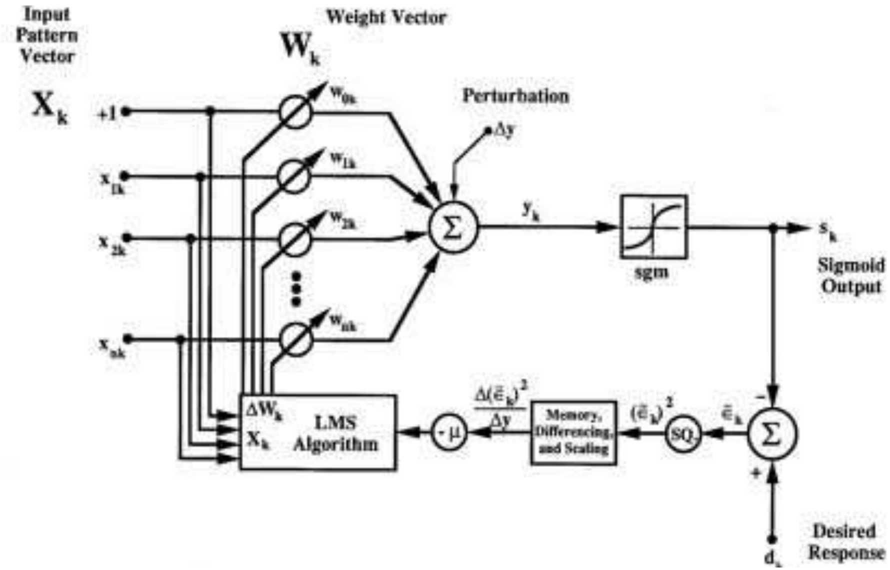
or

$$W_{k+1} = W_k - 2\mu \varepsilon_k X_k \left(\frac{\Delta \varepsilon}{\Delta y}\right)_k$$

SIGMOID LMS (BACK-PROP) → $W_{k+1} = W_k + 2\mu \varepsilon_k X_k f'(X_k^T W_k)$

and $\frac{\Delta \varepsilon}{\Delta y} = \frac{-\Delta s}{\Delta y} \approx -f'(y)$

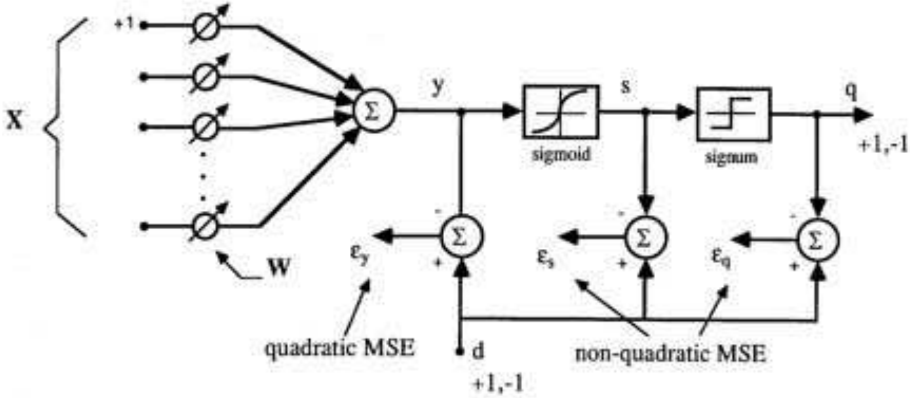
Implementation of Sigmoid LMS-MRIII



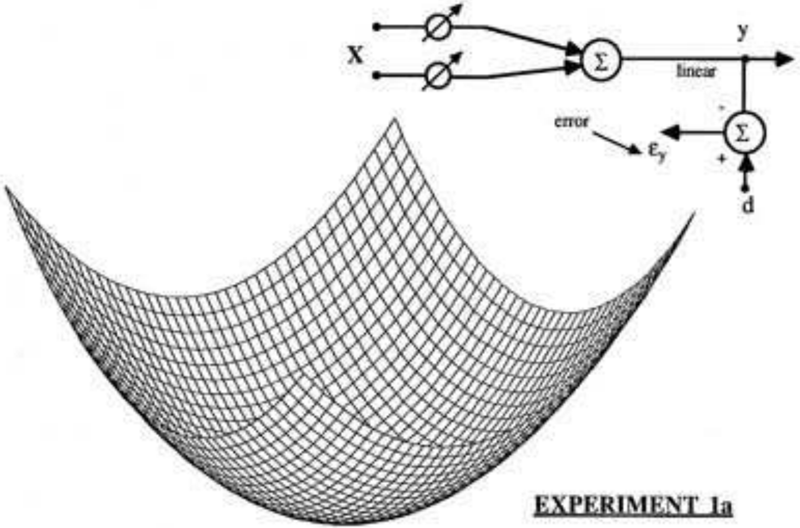
Error Surfaces for the Single Element

- Linear Error
- Sigmoid Error
- Signum Error

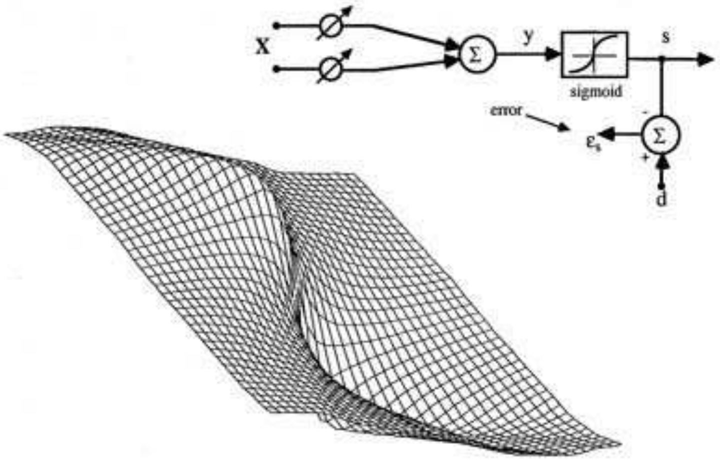
Linear MSE, Sigmoid MSE, and Signum MSE



Linear Mean Square Error Surface

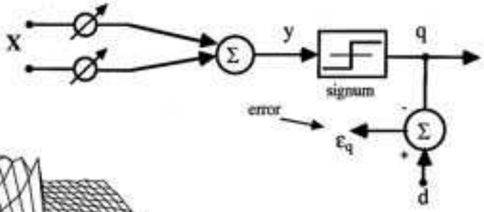


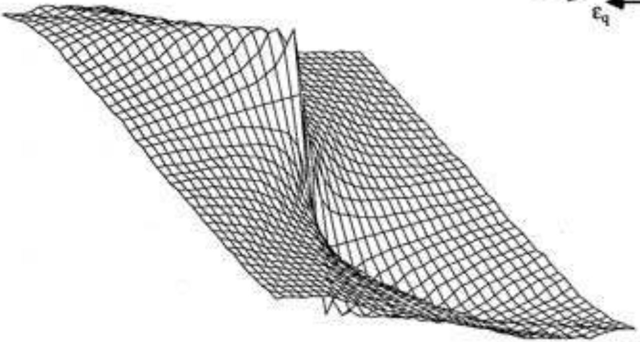
Sigmoid Mean Square Error Surface



EXPERIMENT 1b

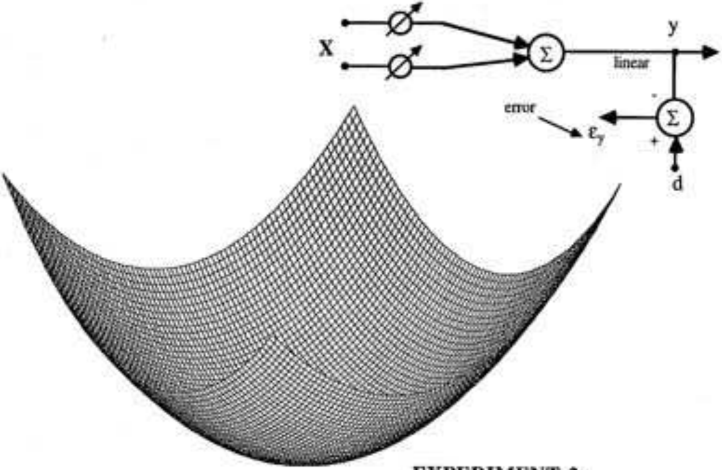
Signum Mean Square Error Surface





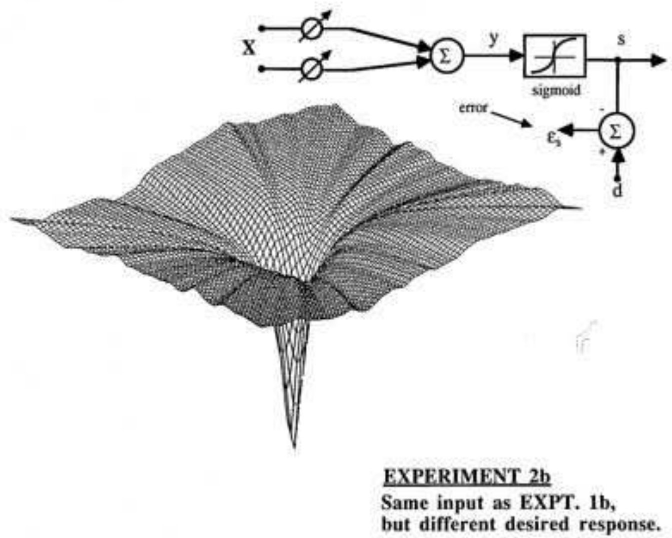
EXPERIMENT 1c

Linear Mean Square Error Surface

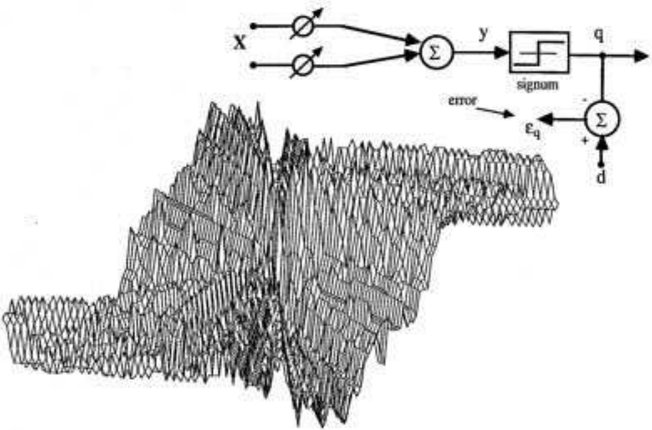


EXPERIMENT 2a
Same input as EXPT. 1a,
but different desired response.

Sigmoid Mean Square Error Surface



Signum Mean Square Error Surface



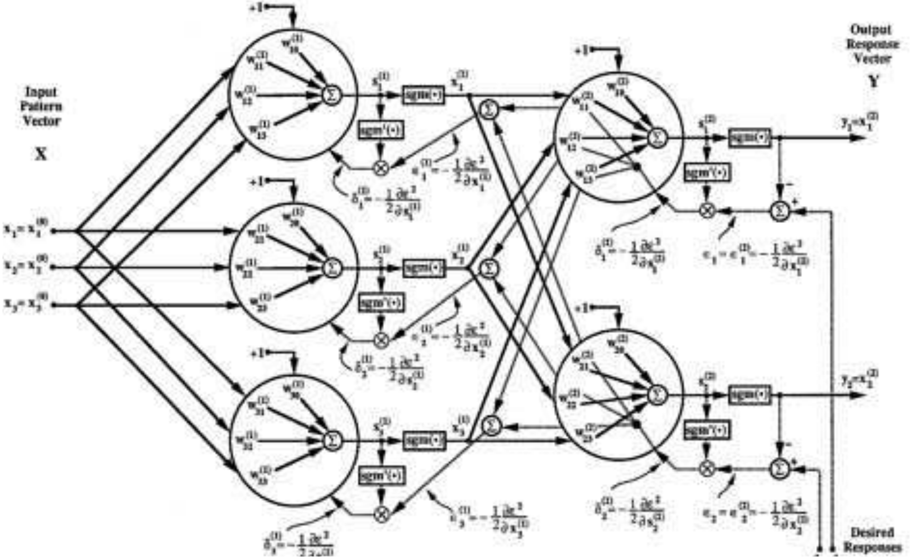
EXPERIMENT 2c

Same input as EXPT. 1c, but different desired response.

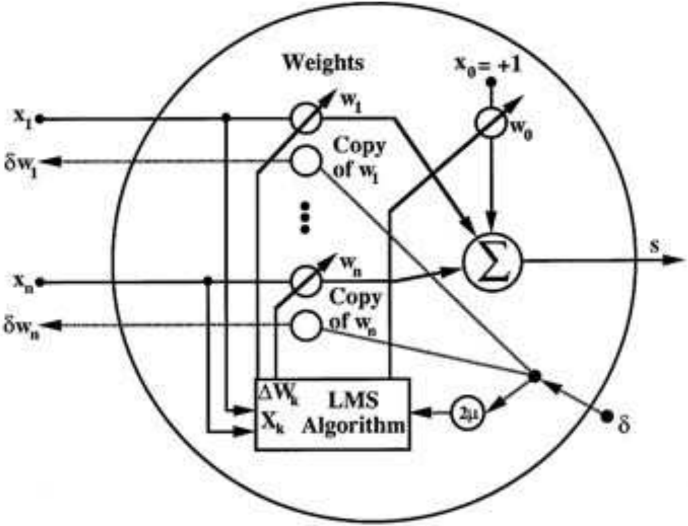
Steepest-Descent Algorithms for Multi-Element Networks

- Backpropagation
- Madaline Rule III (MRIII)

Backpropagation Network

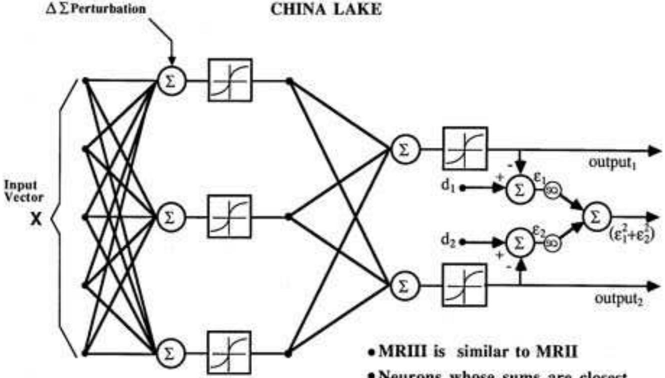


Detail of Backpropagation Node



MRIII of David Andes

MRIII of DAVID ANDES
CHINA LAKE



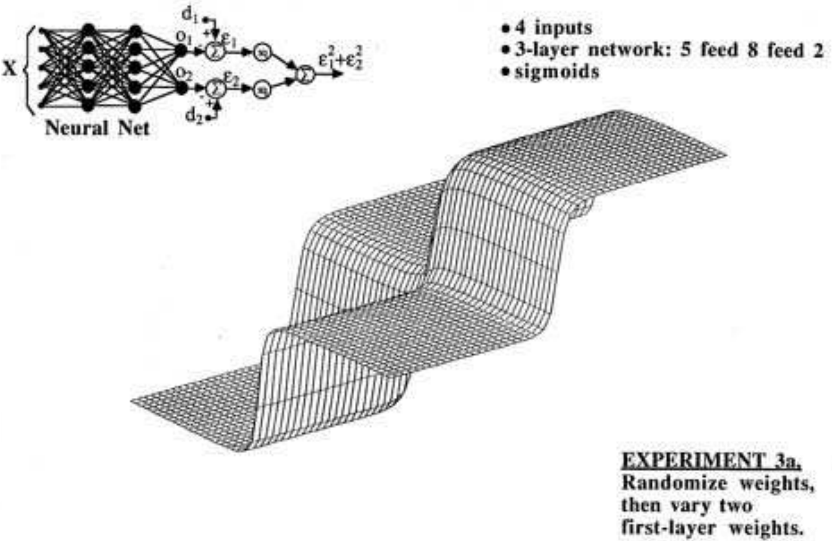
- MRIII is similar to MRII
- Neurons whose sums are closest to zero tend to be adapted most strongly
- Neurons adapted one at a time
- Adaptation reduces Euclidean error

For each neuron:

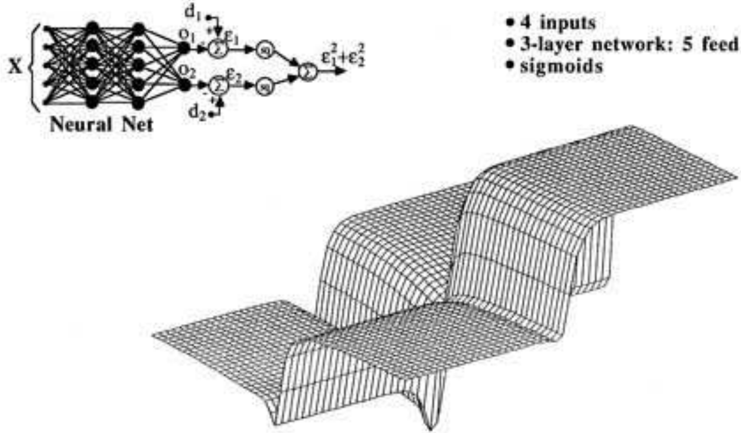
$$W_{k+1} = W_k - \mu X_k \left(\frac{\Delta(\varepsilon_1^2+\varepsilon_2^2)}{\Delta\Sigma} \right)$$

Error Surfaces for Sigmoidal Multilayer Networks

Mean Square Error Surface



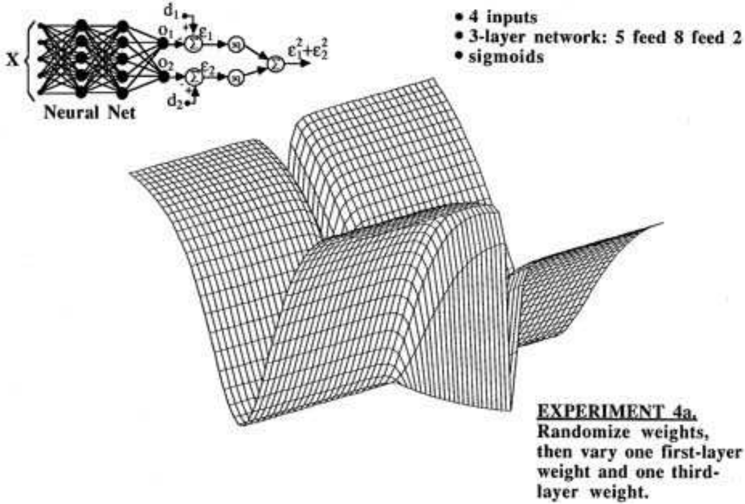
Mean Square Error Surface



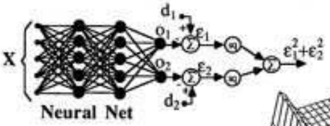
- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids

EXPERIMENT 3b.
Adapt weights by backprop, then vary two first-layer weights.

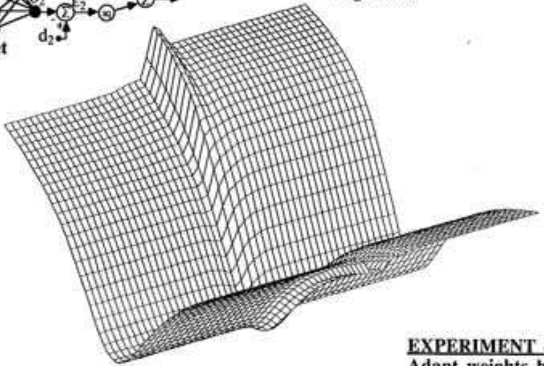
Mean Square Error Surface



Mean Square Error Surface



- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids



EXPERIMENT 4b.
Adapt weights by backprop, then vary one first-layer weight and one third-layer weight.

Learning Rules