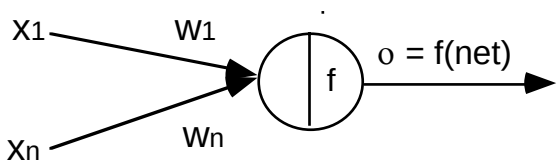
Neural networks III: The delta learning rule with semilinear activation function

The standard delta rule essentially implements gradient descent in sum-squared error for linear activation functions.

We shall describe now the delta learning rule with semilinear activation function. For simplicity we explain the learning algorithm in the case of a single-output network.



Single neuron network.

The output of the neuron is computed by unipolar sigmoidal activation function

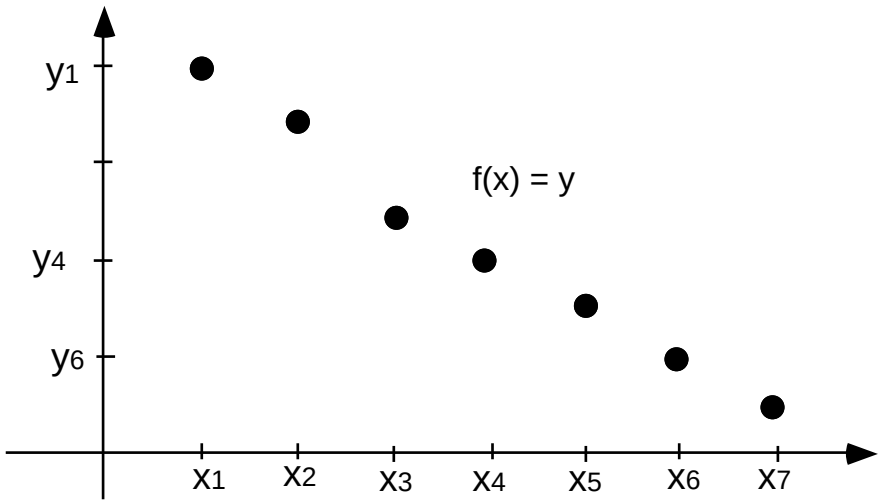
$$o(< w, x >) = \frac{1}{1 + \exp(-w^T x)}.$$

Suppose we are given the following training set

No.	input values	desired output value
1.	$x^1 = (x_1^1, \ldots x_n^1)$	y^1
2.	$x^2 = (x_1^2, \ldots x_n^2)$	y^2
$\vdots$	$\vdots$	$\vdots$
K.	$x^K = (x_1^K, \ldots x_n^K)$	y^K

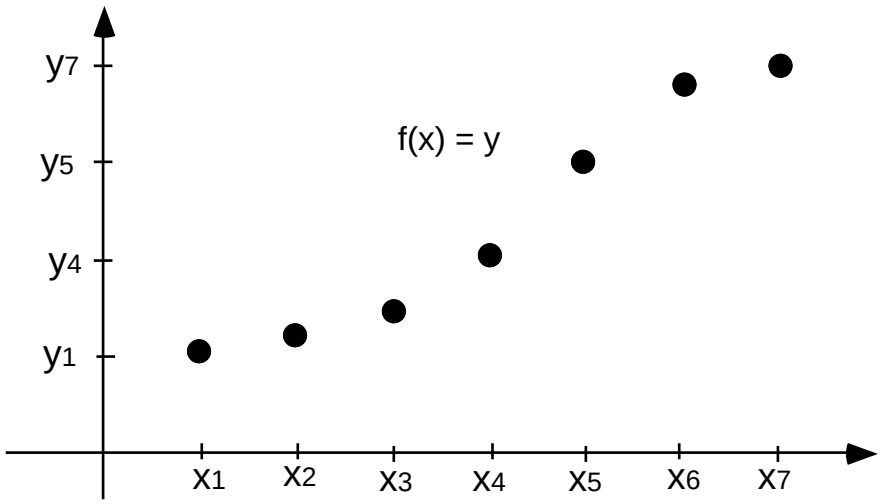
The system first uses the input vector, x^k, to produce its own output vector, o^k, and then compares this with the desired output, y^k.

If we use a linear output unit then whatever is the final weight vector, the output function of the network is a line (in two-dimensional case).



A pattern set from an almost linear function f.

It means that the delta learning rule with linear output function can approximate only a pattern set derived from an almost linear function.



A pattern set from a nonlinear function f.

However, if the patterns are derived from a nonlinear function f, then the chances for a good approximation are small. It is why we use semilinear activation functions.

Let

$$E_k = \frac{1}{2}(y^k - o^k)^2 = \frac{1}{2}(y^k - o(< w, x^k >))^2 =$$

$$\frac{1}{2} \times \left[y^k - \frac{1}{1 + \exp\left(-w^T x^k\right)} \right]^2$$

be our measure of the error on input/output pattern k and let

$$E = \sum_{k=1}^{K} E_k = E_1 + E_2 + \cdots + E_K,$$

be our overall measure of the error.

The rule for changing weights following presentation of input/output pair k is given by the gradient descent method, i.e. we minimize the quadratic error function by using the following iteration process

$$w := w - \eta E_k'(w).$$

Let us compute now the gradient vector of the error function E_k at point w:

$$E_k'(w) = \frac{d}{dw}\left(\frac{1}{2} \times \left[y^k - \frac{1}{1+\exp\left(-w^T x^k\right)}\right]^2\right) =$$

$$\frac{1}{2} \times \frac{d}{dw}\left[y^k - \frac{1}{1+\exp\left(-w^T x^k\right)}\right]^2 =$$

$$-(y^k - o^k)o^k(1 - o^k)x^k$$

where $o^k = 1/(1 + \exp(-w^T x^k))$.

Therefore our learning rule for w is

$$w := w + \eta(y^k - o^k)o^k(1 - o^k)x^k$$

which can be written as

$$w := w + \eta\delta_k o^k(1 - o^k)x^k$$

where $\delta_k = (y^k - o^k)o^k(1 - o^k)$.

Summary 1. *The delta learning rule with unipolar sigmoidal activation function.*

Given are K training pairs arranged in the training set

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}$$

where $x^k = (x_1^k, \ldots, x_n^k)$ *and* $y^k \in \mathbf{R}$, $k = 1, \ldots, K$.

- **Step 1** $\eta > 0$, $E_{\max} > 0$ are chosen

- **Step 2** Weigts w are initialized at small random values, $k := 1$, and the running error E is set to 0
- **Step 3** Training starts here. Input x^k is presented, $x := x^k$, $y := y^k$, and output o is computed

$$o = o(< w, x >) = \frac{1}{1 + \exp\left(-w^T x\right)}$$

- **Step 4** Weights are updated

$$w := w + \eta(y - o)o(1 - o)x$$

- **Step 5** Cumulative cycle error is computed by adding the present error to E

$$E := E + \frac{1}{2}(y - o)^2$$

- **Step 6** If $k < K$ then $k := k+1$ and we continue the training by going back to **Step 3**, otherwise we go to **Step 7**
- **Step 7** The training cycle is completed. For $E < E_{\max}$ terminate the training session. If $E > E_{\max}$

then E is set to 0 and we initiate a new training cycle by going back to **Step 3**

In this case, without hidden units, the error surface is shaped like a bowl with only one minimum, so gradient descent is guaranteed to find the best set of weights. With hidden units, however, it is not so obvious how to compute the derivatives, and the error surface is not concave upwards, so there is the danger of getting stuck in local minima.

We illustrate the delta learning rule with bipolar sigmoidal activation function

$$f(t) = \frac{2}{(1 + \exp -t) - 1}.$$

Example 1. *The delta learning rule with bipolar sigmoidal activation function. Given are K training pairs arranged in the training set*

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}$$

where $x^k = (x_1^k, \ldots, x_n^k)$ *and* $y^k \in \mathbf{R}$, $k = 1, \ldots, K$.

- **Step 1** $\eta > 0$, $E_{\max} > 0$ are chosen
- **Step 2** Weigts w are initialized at small random values, $k := 1$, and the running error E is set to 0
- **Step 3** Training starts here. Input x^k is presented, $x := x^k$, $y := y^k$, and output o is computed

$$o = o(< w, x >) = \frac{2}{1 + \exp(-w^T x)} - 1$$

- **Step 4** Weights are updated

$$w := w + \frac{1}{2}\eta(y - o)(1 - o^2)x$$

- **Step 5** Cumulative cycle error is computed by adding the present error to E

$$E := E + \frac{1}{2}(y - o)^2$$

- **Step 6** If $k < K$ then $k := k+1$ and we continue the training by going back to **Step 3**, otherwise we go to **Step 7**
- **Step 7** The training cycle is completed. For $E < E_{\max}$ terminate the training session. If $E > E_{\max}$ then E is set to 0 and we initiate a new training cycle by going back to **Step 3**

Exercise 1. *The error function to be minimized is given by*

$$E(w_1, w_2) = \frac{1}{2}[(w_2 - w_1)^2 + (1 - w_1)^2]$$

Find analytically the gradient vector

$$E'(w) = \begin{bmatrix} \partial_1 E(w) \\ \partial_2 E(w) \end{bmatrix}$$

Find analytically the weight vector w^* that minimizes the error function such that

$$E'(w) = 0.$$

Derive the steepest descent method for the minimization of E.

Solution 1. *The gradient vector of E is*

$$E'(w) = \begin{bmatrix} (w_1 - w_2) + (w_1 - 1) \\ (w_2 - w_1) \end{bmatrix} = \begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix}$$

and $w^*(1,1)^T$ is the unique solution to the equation

$$\begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The steepest descent method for the minimization of E reads

$$\begin{bmatrix} w_1(t+1) \\ w_2(t+1) \end{bmatrix} = \eta \begin{bmatrix} 2w_1(t) - w_2(t) - 1 \\ w_2(t) - w_1(t) \end{bmatrix}.$$

where $\eta > 0$ is the learning constant and t indexes the number of iterations.

That is,

$$w_1(t+1) = w_1(t) - \eta(2w_1(t) - w_2(t) - 1),$$

$$w_2(t+1) = w_2(t) - \eta(w_2(t) - w_1(t)),$$

or, equivalently,

$$w_1 := w_1 - \eta(2w_1(t) - w_2 - 1),$$

$$w_2 := w_2 - \eta(w_2(t) - w_1).$$