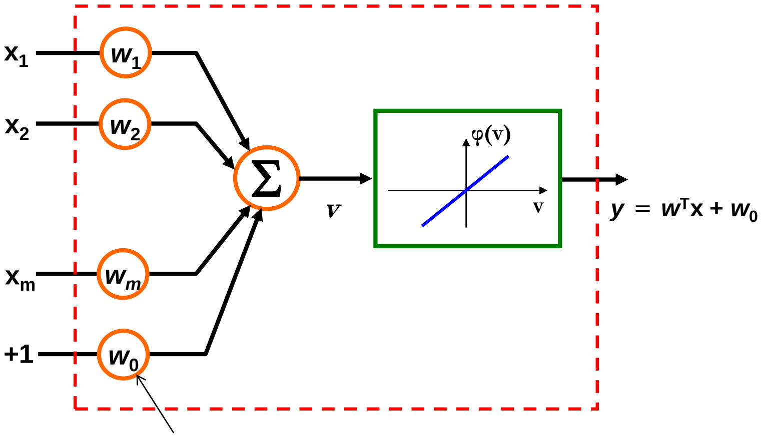
The Adaline* neuron model



*Adaline is for ADAptive LINear Element

Adaline

- Following presentation of the $<x_k, d_k>$ pair from the training set (d_k is the desired output) the actual neuron output y_k is computed as a linear combination of the inputs:

$$y_k = \sum_{i=1}^{m} w_i x_{ki} + w_0$$

- Weight update rule (also known as the *Widrow-Hoff rule* [WH-rule]):

$$w_i^{new} = w_i^{old} + \alpha \, (d_k - y_k) \, x_{ki}$$

- where α is the adaptation gain (or learning rate) and should be chosen either as constant in the (0, 1) interval or as a decreasing function of time satisfying the conditions of stochastic approximation:

$$\sum_{t=1}^{\infty} \alpha(t) = \infty \quad \text{και} \quad \sum_{t=1}^{\infty} \alpha^2(t) < \infty$$

The Adaline training procedure

1. Initialize synaptic weights and bias to small random numbers
2. Present new $<x_k, d_k>$ pair from the training set
3. Compute actual neuron output: $y_k = \sum_{i=1}^{m} w_i x_{ki} + w_0$
4. Update synaptic weights and bias using the *Widrow-Hoff rule*:

   $$w_i(k+1) = w_i(k) + \alpha(k)\,(d_k - y_k)\,x_{ki} \qquad \forall\ 0 \le i \le m$$

   with $0 < \alpha(k) < 1$. In case of a two-class classification problem, $d_k = +1$ for inputs from the first category and -1 for inputs from the second category.
5. While there are weight modifications and the maximum number of presentations has not been reached, go to step 2.

Relation of WH-rule to Least Squares

Define the error to be minimized as the square deviation between actual and desired outputs at iteration k, with $<x_k, d_k>$ being the k-th I/O pair presented to the network:

$$E_k = \frac{1}{2}\,(y_k - d_k)^2 \qquad (1)$$

Recall that neuron output is computed by:

$$y_k = \sum_{i=1}^{m} w_i x_{ki} + w_0 \qquad (2)$$

Relation of WH-rule to Least Squares

GOAL: minimize E_k for all k by *stochastic gradient descent*, i.e. modify weights w_i in the direction of steepest descent in the error surface (the error is a function of all weights and bias) so that E_k is decreased at maximum rate:

$$w_i(k+1) = w_i(k) - \alpha(k)\,\frac{\partial E_k}{\partial w_i} \qquad (3)$$

Weight Update Rule:

From (1): $$\frac{\partial E_k}{\partial w_i} = \frac{1}{2}\,2\,(y_k - d_k)\,\frac{\partial y_k}{\partial w_i}$$

which from (2) becomes:

Relation of WH-rule to Least Squares

$$\frac{\partial E_k}{\partial w_i} = (y_k - d_k)\,x_{ki} \quad \text{for weights } w_i \ (i = 1, \ldots, m) \qquad (4)$$

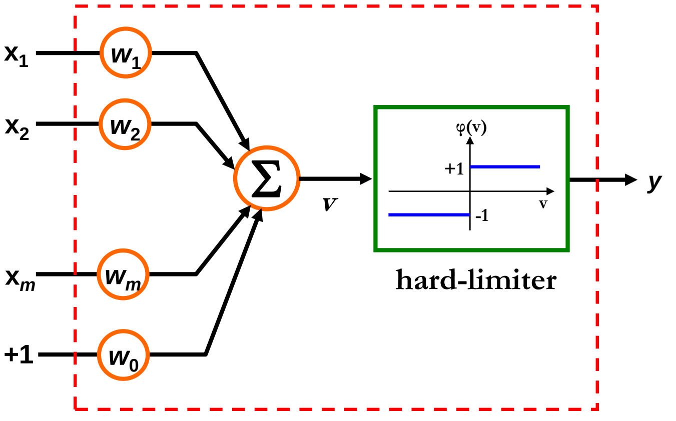
$$\frac{\partial E_k}{\partial w_0} = (y_k - d_k) \quad \text{for bias } w_0 \qquad (5)$$

Finally, from (3) & (4) or (3) & (5) we obtain the update rules:

$$\left.\begin{array}{l} w_i(k+1) = w_i(k) - \alpha(k)\,(y_k - d_k)\,x_{ki} \quad \forall\, i = 1, \ldots, m \\ \text{and} \\ w_0(k+1) = w_0(k) - \alpha(k)\,(y_k - d_k) \end{array}\right\} \qquad (6)$$

which are the same as the Widrow-Hoff rule.

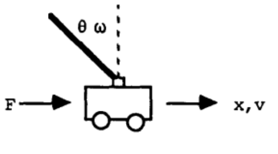
Adaline* used as a classifier



The trained Adaline is now used in the testing phase as a classifier

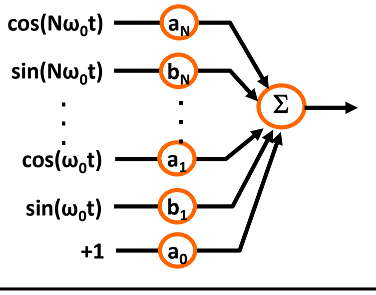
Adaline applications

- **Adaptive controller**



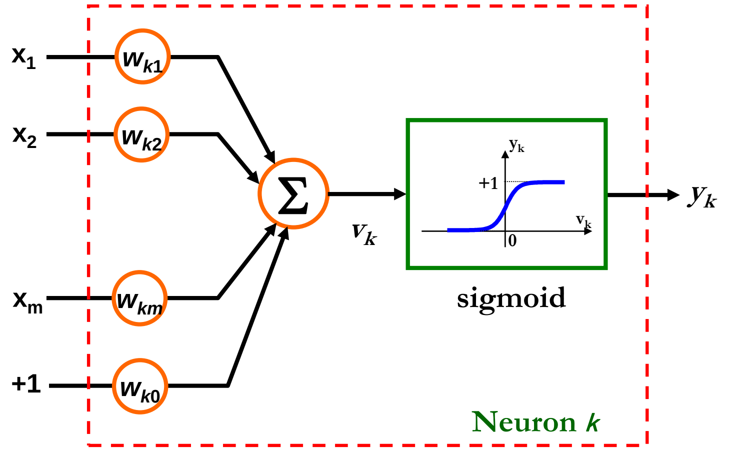

```
θ = pendulum angle
ω = pendulum velocity
x = cart position
v = cart velocity
F = reaction force
```

- **Fourier Series Analysis**



$$x_p(t) \approx a_0 + \sum_{n=1}^{N} a_n \cos(n\omega_0 t) + \sum_{n=1}^{N} b_n \sin(n\omega_0 t)$$

The "logistic regressor" neuron model



Error and neuron activation functions

- **Error to be minimized:**

$$E_k = \frac{1}{2} (y_k - d_k)^2 \qquad (1)$$

where $<x_k, d_k>$ **is the** k**-th I/O pair presented at iteration** k**.**

- **Neuron activation:**

$$v_k = \sum_{i=1}^{m} w_i x_{ki} + w_0 \qquad (2)$$

- **Neuron output:**

$$y_k = \frac{1}{1 + e^{-v_k}} \qquad (3)$$

Weight update rule

GOAL: minimize E_k $\forall k$ by ***stochastic gradient descent*** **(SGD):**

$$w_i(k+1) = w_i(k) - \alpha(k) \frac{\partial E_k}{\partial w_i} \qquad (4)$$

Weight Update Rule:

From (1): $$\frac{\partial E_k}{\partial w_i} = (y_k - d_k) \frac{\partial y_k}{\partial w_i}$$

From (2) & (3): $$\frac{\partial y_k}{\partial w_i} = \frac{\partial y_k}{\partial v_k} \frac{\partial v_k}{\partial w_i} = y_k (1 - y_k)\, x_{ki}$$

Hence: $$\boxed{w_i(k+1) = w_i(k) - \alpha(k)\, (y_k - d_k)\, y_k (1 - y_k)\, x_{ki}}$$