Apr 1990 Gradient Descent — Algorithm

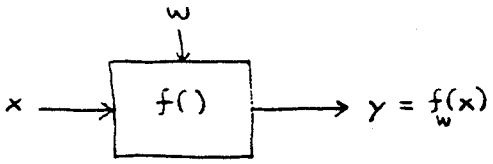
Neil E Cotter

Single weight: $\Delta w = -\eta \frac{\partial E(w)}{\partial w}$

where $\eta \equiv$ learning rate or step size $\eta > 0$

$w \equiv$ synaptic weight (or any other parameter of a box computing a function)

$E(w) \equiv$ Error made by neural network (or any box computing a function)



desired output $\equiv t$ or $t(x)$

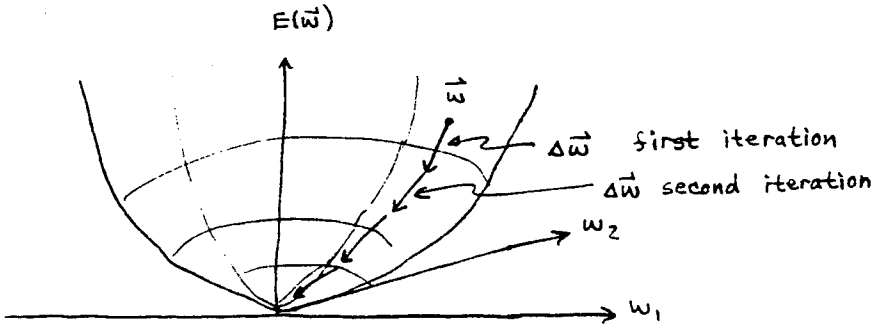
t provided by outside world

Typically, $E(w) = \frac{1}{2}(t-y)^2$ squared error

Multiple weights: $\Delta \vec{w} = -\eta \nabla E(\vec{w})$ where $\nabla E(\vec{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_n} \end{bmatrix}$ = gradient

notn: $\vec{w} = (w_1, \ldots, w_n)$

$\Delta \vec{w} = (\Delta w_1, \ldots, \Delta w_n)$



Algorithm descends toward minimum of $E(\vec{w})$ by following gradient, (which happens to be direction of steepest descent). The algorithm tells us how to adjust w_1, w_2 for min error, $E(\vec{w})$.

In other words, gradient descent is a learning algorithm.