EE 434
02/08/05

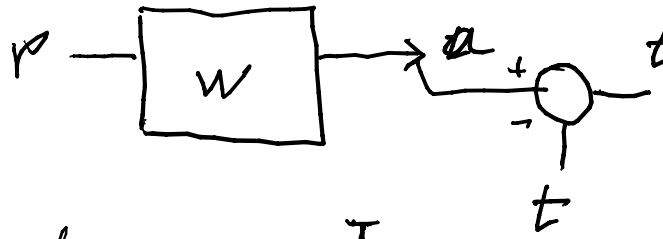
Backpropagation

Summary on pp. 11-24-25

scheme for training

t = desired outputs for an input p

a = output of net for input p



form $(t-a)^T(t-a)$; V^T means transpose

= sum of squares

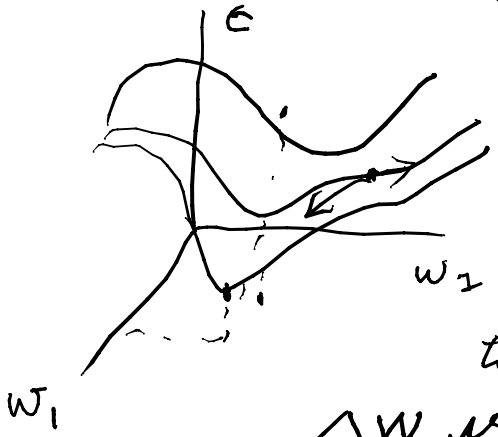
adjust the weights (including biases) to try to minimize $\epsilon = (t-a)^T(t-a)$

use Newton iteration, index by integers k

for k-th step

$$W(k+1) = W(k) + \Delta W(k+1)$$

" change in weights

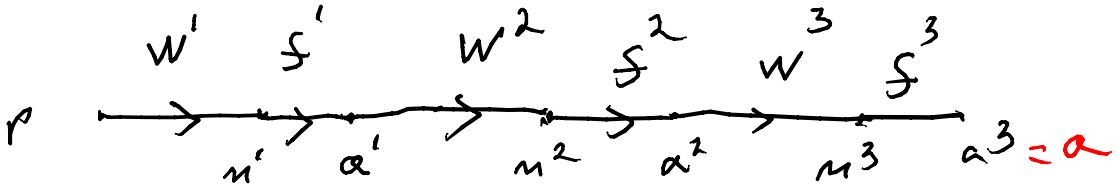


desire smallest ϵ by choice of $W \rightarrow \Delta W$ makes ϵ smaller

to go downhill desire ΔW proportional to $-\nabla_W \epsilon$

as desire to not overshoot the minimum of ϵ put in a scale factor α, $0<\alpha<1$

This works ok if see the weights directly in ϵ, but for many sectioned feedforward networks ϵ is not a direct function of the layers weights of internal layers. So have backpropagation.



$$a^3 = f^3(n^3) = f^3(W^3 a^2) = f^3(W^3 f^2(n^2))$$

$$= f^3(W^3 f^2(W^2 a^1)) = f^3(W^3 f^2(W^2 f^1(W^1 p)))$$

desire to set the W^1. $\frac{\partial \epsilon}{\partial w^1_{i,j}}$

$$\frac{\partial \epsilon}{\partial w^3_{i,j}} \Rightarrow (t-a)^T(t-a) = \epsilon = t^T t - a^T t - t^T a + a^T a$$

$a^T t$ and $t^T a$ equal as $(a^T t)^T = t^T a^{T^T} = t^T a$

$$\epsilon = t^T t - 2t^T a + a^T a$$

$t^T t$: fixed by choice

$$a = f^3(n^3)$$

$$\frac{\partial \epsilon}{\partial w^3_{i,j}} = 0 - 2t^T \frac{\partial a}{\partial w^3_{i,j}} + \frac{\partial a^T}{\partial w^3_{i,j}} \cdot a + a^T \frac{\partial a}{\partial w^3_{i,j}}$$

$$= \left(-2t^T + 2a^T\right)\frac{\partial a}{\partial w^3_{i,j}}; \quad \frac{\partial a}{\partial w^3_{i,j}} = \sum_{k=1} \frac{\partial f^3(n^3_k)}{\partial n^3_k} \cdot \frac{\partial n^3_k}{\partial w^3_{i,j}}$$

$n^3 = w^3 \cdot a^2 \Rightarrow n^3_k = \sum_{j=1} w^3_{k,j} a^2_j$ only get if $k = i$

here $n^3_i = \sum_{l=1} w^3_{i,l} a^2_l$, $\frac{\partial n^3_i}{\partial w^3_{i,j}} = 1 \cdot a^2_j = a^2_j$

$$\frac{\partial \epsilon}{\partial w^3_{i,j}} = -2(t-a)^T \cdot \frac{\partial a^3_{i,j}}{\partial w^3_{i,j}} = -2\epsilon^T \cdot \frac{\partial f^3_i(n^3_i)}{\partial n^3_i} \cdot a^2_j$$

known from running on p.

understood as only one activation function on each neuron

need the activation function $f^3_i(\cdot)$

Ex: $f^3_i(x) = \tanh x$

need $\frac{\partial f_i(x)}{\partial x} = \frac{d f_i(x)}{dx} = \frac{d}{dx}\left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$

$$= \frac{e^x + e^{-x}}{e^x + e^{-x}} - \frac{(e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})} = 1 - (\tanh x)^2$$

in our case a^3_i = known, $\frac{\partial a^3_i}{\partial n^3_i} = 1 - (a^3_i)^2$

---

works well for output layer, but $\frac{\partial \epsilon}{\partial w^2_{i,j}}$, $\frac{\partial \epsilon}{\partial w^1_{i,j}}$

are not so easy to get as don't know desired outputs for 1st & second layer

$s^M = -2\dot{F}^M(n^M)(t-a)$ = sensitivity vectors for last (M) layer

M ← last one

derivative

$s^m = \dot{F}^m(n^m)(W^{m+1})^T s^{m+1}$ for $m = M-1, \ldots, 1$

p. 11-25

$$F^m = \begin{bmatrix} f(n_1^m) & & \\ & \ddots & \\ & & f(n_{S^m}^m) \end{bmatrix} = \text{diag}\left[f(n_i^m)\right]$$

$$\dot{F}^m = \begin{bmatrix} \frac{\partial f(n_1^m)}{\partial n_1^m} & & \\ & \ddots & \\ & & \frac{\partial f(n_{S^m}^m)}{\partial n_{S^m}^m} \end{bmatrix}$$

S^m = size of the mth n^m vector = # of neurons in m^{th} layer

$W^m(k+1) = W^m(k) - \alpha s^m (a^{m-1})^T$; $0 < \alpha < 1$ up to us to choose