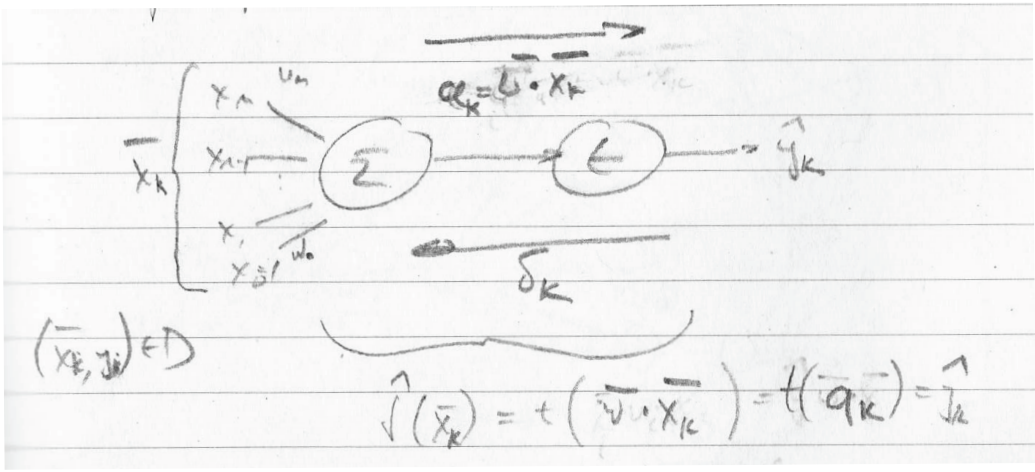
Multi-Layer ANNs

The perceptron revisited



Recall

$$
\begin{aligned}
E_k(\overline{w}) &= \frac{1}{2}(y_k - \hat{y}_k)^2 \\
&= \frac{1}{2}(y_k - t(\overline{w} \bullet \overline{x}_k)^2 \\
&= \frac{1}{2}(y_k - t(a_k))^2
\end{aligned}
$$

Multi-Layer ANNs

We can now look at the gradient,

$$
\begin{aligned}
\nabla E_k(\overline{w}) &= \frac{d}{d\overline{w}} E_k(\overline{w}) \\
&= \frac{1}{2}\frac{d}{d\overline{w}}(y_k - t(a_k))^2 \\
&= -(y_k - t(a_k))\frac{dt}{d\overline{w}}(a_k) \\
&= -(y_k - \hat{y}_k)\frac{dt}{d\overline{w}}(a_k) \\
&= -(y_k - \hat{y}_k)\frac{dt}{da_k}(a_k)\frac{da_k}{d\overline{w}} \text{ (chain rule)} \\
&= -(y_k - \hat{y}_k)t'(a_k)\frac{d}{d\overline{w}}(\overline{w} \bullet \overline{x}_k) \\
&= -(y_k - \hat{y}_k)t'(a_k)\overline{x}_k \\
&= \delta_k \overline{x}_k
\end{aligned}
$$

where $\delta_k = -(y_k - \hat{y}_k)t'(a_k)$ is called the error.

Multi-Layer ANNs

Observation: $\nabla E_k(\overline{w}) = \delta_k \overline{x}_k$, that is, the gradient at $\overline{x}$ is computed by multiplying the error term δ_k with the input $\overline{x}$.

Observation: The error term δ_k is computed by multiplying the observed error $(y_k - \hat{y}_k)$ with the derivative of the activation function evaluated at a_k,

$$-(y_k - \hat{y}_k)t'(a_k)$$

Multi-Layer ANNs

Now recall our update rule,

$$\overline{w} \leftarrow \overline{w} + \Delta\overline{w}$$

From before we have

$$\overline{w} \leftarrow \overline{w} + \eta\nabla E_k(\overline{w})$$

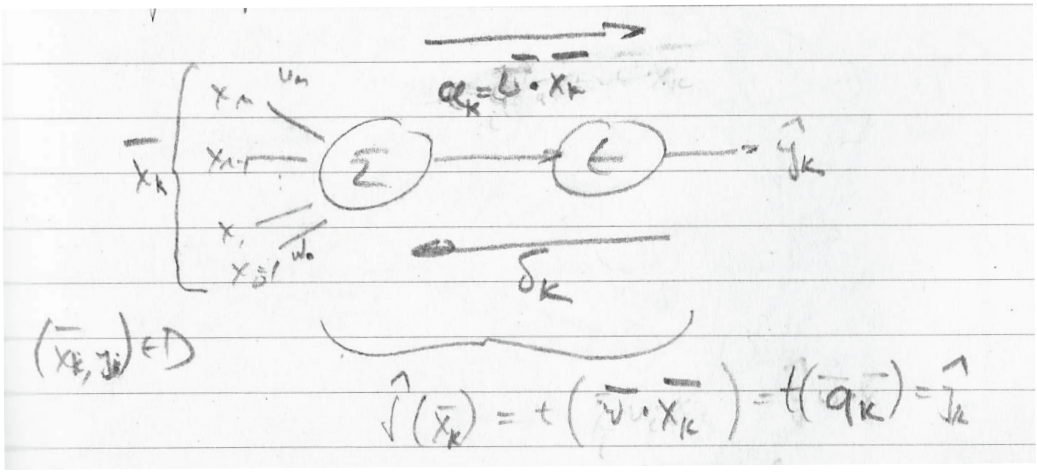
From our discussion above it follows that

$$\overline{w} \leftarrow \overline{w} + \eta\delta_k\overline{x}_k$$

Observation: The weights are updated using a scaled version of the input vector. It is also easy to see that the weights are scaled proportional to the error.

This is called *back propagation.*

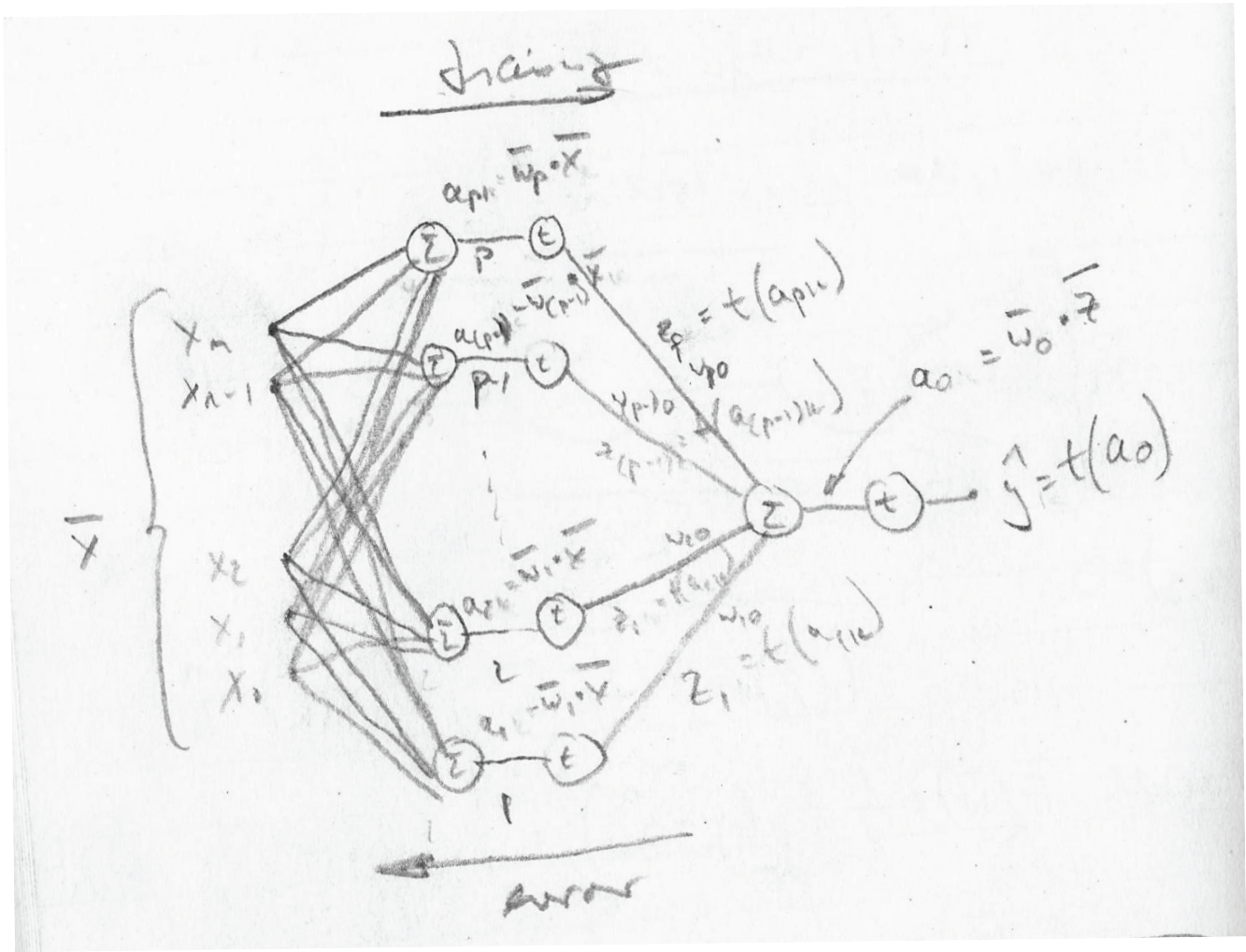
Multi-Layer ANNs



Feed forward and backprop neural networks.

Multi-Layer ANNs

A simple ANN with a single hidden layer of p nodes and a differentiable transfer function t,



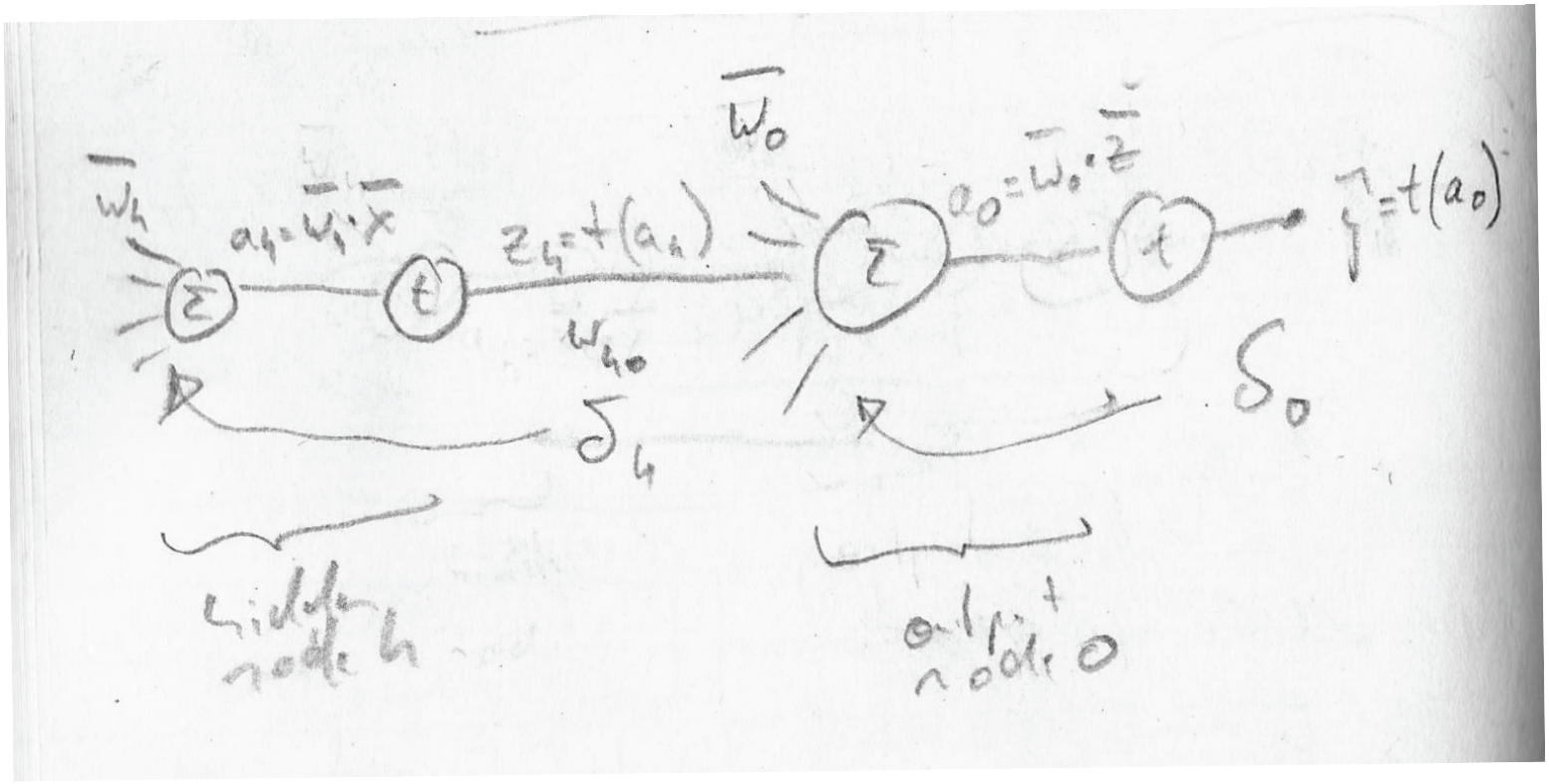
Multi-Layer ANNs

How do we train this kind of network? Observed errors for the hidden layers are no longer available.

Solution: We systematically backprop the error and use the backprop error instead of the observed error when training the hidden layer.

Multi-Layer ANNs

Consider a single path through the network:



Training instance $\overline{x}$ is fed forward and the error is back propagated.

Multi-Layer ANNs

Now consider the update rule for the weights at the output node o,

$$\overline{w}_o \leftarrow \overline{w}_o + \eta \nabla E(\overline{w})$$

with

$$\begin{aligned} \nabla E(\overline{w}) &= \frac{\partial E(\overline{w})}{\partial \overline{w}_o} \\ &= \frac{\partial E(\overline{w})}{\partial a_o} \frac{\partial a_o}{\partial \overline{w}_o} \\ &= \delta_o \overline{z} \end{aligned}$$

where

$$\frac{\partial a_o}{\partial \overline{w}_o} = \frac{\partial (\overline{w}_o \bullet \overline{z})}{\partial \overline{w}_o} = \overline{z} \text{ (see diagram above)}$$

and

$$\frac{\partial E(\overline{w})}{\partial a_o} = \frac{1}{2} \frac{\partial}{\partial \overline{w}_o} (y - t(a_o))^2 = -(y - t(a_o)) t'(a_o) = -(y - \hat{y}) t'(a_o) = \delta_o$$

This means the update rule for our output node weights becomes,

$$\overline{w}_o \leftarrow \overline{w}_o + \eta \delta_o \overline{z}$$

Multi-Layer ANNs

Now consider updating the weights for the hidden node h,

$$\overline{w}_h \leftarrow \overline{w}_h + \eta \nabla E(\overline{w})$$

with

$$\begin{aligned}
\nabla E(\overline{w}) &= \frac{\partial E(\overline{w})}{\partial \overline{w}_h} \\
&= \frac{\partial E(\overline{w})}{\partial a_h} \frac{\partial a_h}{\partial \overline{w}_h} \\
&= \frac{\partial E(\overline{w})}{\partial a_h} \frac{\partial (\overline{w}_h \bullet \overline{x})}{\partial \overline{w}_h} \\
&= \delta_h \overline{x}
\end{aligned}$$

with

$$\delta_h = \frac{\partial E(\overline{w})}{\partial a_h} = \frac{\partial E(\overline{w})}{\partial a_o} \frac{\partial a_o}{\partial a_h} = \delta_o \frac{\partial a_o}{\partial a_h} = w_{ho} t'(a_h) \delta_o$$

Note:

$$\begin{aligned}
a_o &= \overline{w}_o \bullet \overline{z} = \ldots + w_{ho} z_h + \ldots = \ldots w_{ho} t(a_h) \ldots \\
\frac{\partial a_o}{\partial a_h} &= 0 + \ldots + 0 + w_{ho} t'(a_h) + 0 + \ldots + 0 = w_{ho} t'(a_h)
\end{aligned}$$

Multi-Layer ANNs

Therefore:

$$\overline{w}_h \leftarrow \overline{w}_h + \eta \delta_h \overline{x}$$

or

$$\overline{w}_h \leftarrow \overline{w}_h + \eta w_{ho} t'(a_h) \delta_o \overline{x}$$

The updates to the hidden weights is now accomplished with entities that are easily computed.

Also, since we are stacking neurons we are no longer confined to liner surfaces! BUT, the error surface is also no longer convex and therefore a global minimum is much harder to find since there might be lots of local minima.