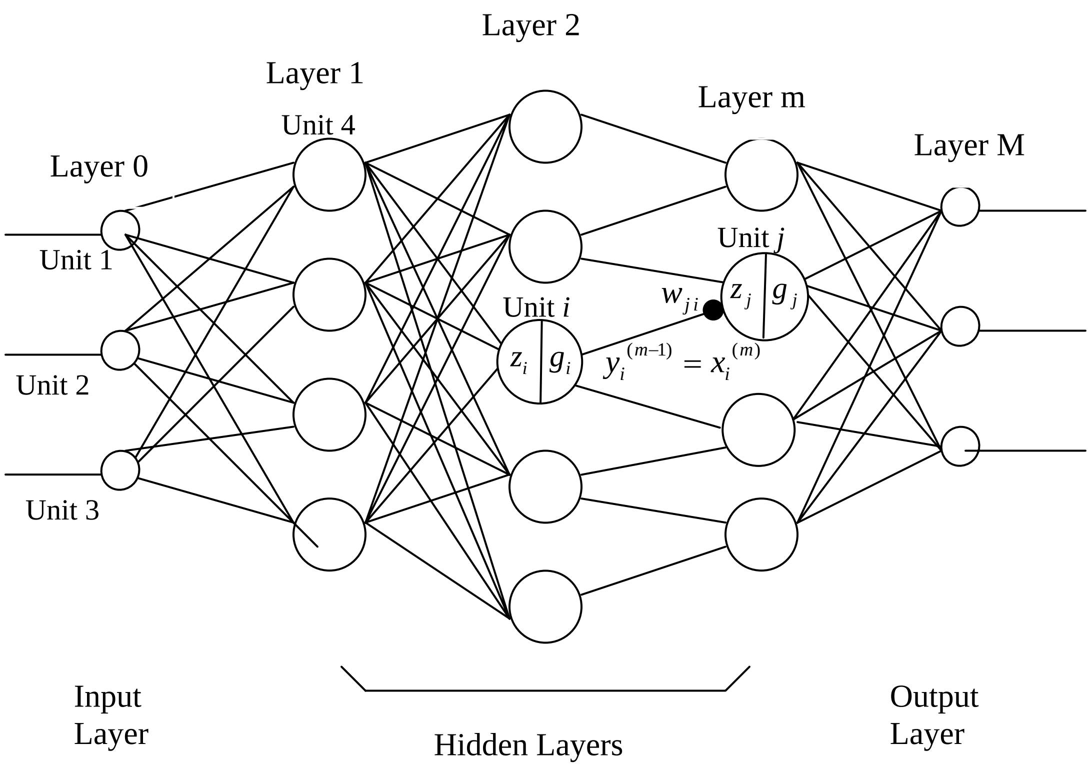
2.160 System Identification, Estimation, and Learning
Lecture Notes No. 14
April 10, 2006

8.4 The Error Back Propagation Algorithm

The Multi-Layer Perception is a universal approximation function that can approximate an arbitrary (measurable) function to any accuracy.



$w_{ji}^{(m)}$ = weight of the connection from unit i to unit j in layer m

$y_j^{(m)}$ = output from unit j in layer m

$x_i^{(m)}$ = input to a unit in layer m from unit i

Forward computation

$$z_j^{(m)} = \sum_i w_{ji}^{(m)} x_i^{(m)} \tag{19}$$

$$y_j^{(m)} = g_j(z_j^{(m)}) = x_j^{(m+1)} \tag{20}$$

$$m = 0, 1, 2, \ldots$$

Starting from $m = 0$, all the units can be computed recursively until $m = M$, output layer.

How do we train the multi-layer perceptron, given training data presented sequentially?

Note: Multi-Layer Perceptrons with nonlinear activation functions, $g(z)$, are nonlinear in parameters w.

- A single-layer neural net is essentially linear in w, although $g(z)$ is nonlinear.
- If two consecutive layers have linear activation functions, they can be combined and replaced by a single layer network.

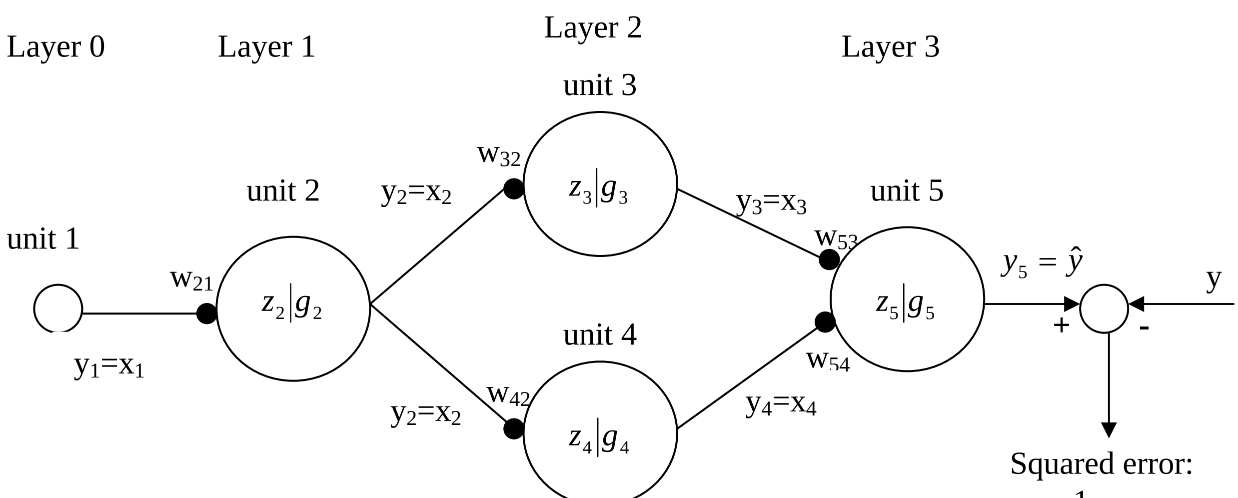
To be able to deal with nonlinear problems, such as the XOR problem, we now focus on a multi-layer perceptron with nonlinear activation functions.

The theory of stochastic approximation is not applicable, since the parameters are not linearly involved in the predictor. However, the Gradient Descent Method (The Widrow-Hoff algorithm) can be extended to multi-layer perceptrons.

The algorithm is called the ***Error Backpropagation*** Algorithm.

Example

Consider a three-layer perceptron in order to derive a basic formula of error backpropagation.



$$E = \frac{1}{2}(\hat{y} - y)^2$$

Gradient Descent

$$\begin{aligned}\Delta w_{53} &= -\rho\, grad_{w_{53}} E \\ &= -\rho \frac{\partial E}{\partial y_5}\frac{dy_5}{dz_5}\frac{\partial z_5}{\partial w_{53}} \qquad (21) \\ &= -\rho \underline{(\hat{y} - y) g'_5(z_5)} \cdot x_3 = \rho \delta_5 x_3\end{aligned}$$

$$-\delta_5 = \frac{\partial E}{\partial z_5}$$

$$\begin{aligned}\Delta w_{32} &= -\rho\, grad_{w_{32}} E \\ &= -\rho \frac{\partial E}{\partial y_5}\frac{dy_5}{dz_5}\overbrace{\frac{\partial z_5}{\partial x_3}}^{w_{53}}\overbrace{\frac{\partial x_3}{\partial z_3}}^{\frac{dg_3}{dz_3}}\frac{\partial z_3}{\partial w_{32}} \\ &= \rho \underbrace{\delta_5\, w_{53}\, g'_3(z_3)}_{\delta_3} x_2\end{aligned} \tag{22}$$

Likewise,

$$\Delta w_{42} = \rho \underbrace{\delta_5 w_{54} g'_4(z_4)}_{\delta_4} x_2 \tag{23}$$

$\Delta w_{21} = -\rho\, grad_{w_{21}} E$: There are two routes between z_5 and w_{21}.

$$\begin{aligned} &= -\rho \frac{\partial E}{\partial z_5}\left(\frac{\partial z_5}{\partial x_3}\frac{\partial x_3}{\partial x_2}\frac{\partial x_2}{\partial w_{21}} + \frac{\partial z_5}{\partial x_4}\frac{\partial x_4}{\partial x_2}\frac{\partial x_2}{\partial w_{21}}\right) \\ &= \rho(\underbrace{\delta_5 w_{53} g'_3(z_3)}_{\delta_3} w_{32} + \underbrace{\delta_5 w_{54} g'_4(z_4)}_{\delta_4} w_{42})\frac{\partial y_2}{\partial w_{21}} \\ &= \rho(\delta_3 w_{32} + \delta_4 w_{42}) g'_2(z_2) x_1 \end{aligned} \tag{24}$$

The above computation can be streamlined by computing δ_j, starting from the final layer back to the first layer.

Error $(\hat{y} - y)$ is propagated backward… Error Backpropagation

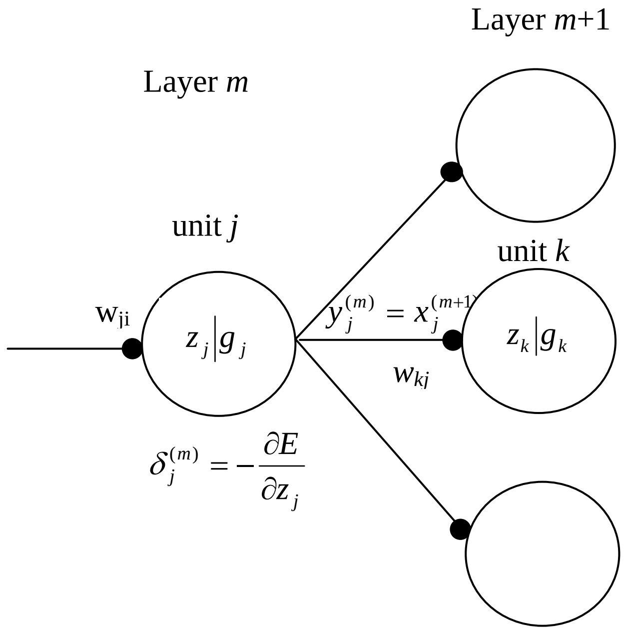
In general,

For the final layer, $m = M$,

$$\Delta w_{ji}^{(M)} = \rho \delta_j^{(M)} x_i^{(M)} \tag{25}$$

$$\delta_j^{(M)} = (y - \hat{y}^{(M)}) g'^{(M)}_j [z_j^{(M)}] \tag{26}$$

For hidden layers, $1 \le m \le M - 1$



$$\Delta w_{ji} = -\rho \, grad_{w_{ji}} E = -\rho \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}} = \rho \delta_j^{(m)} x_i^{(m)} \qquad (27)$$

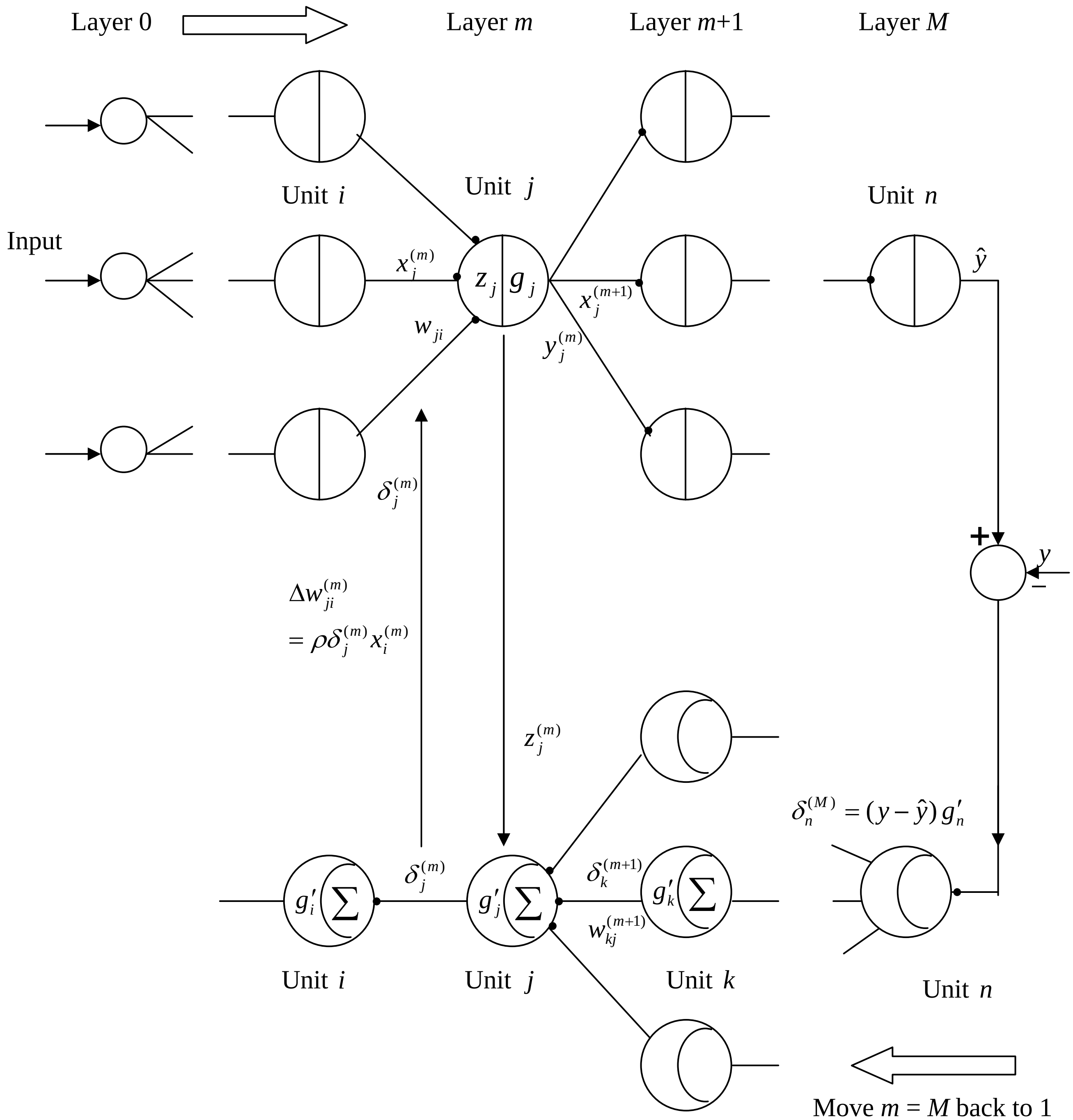
$$\begin{aligned} \delta_j^{(m)} &= -\frac{\partial E}{\partial z_j} \\ &= -\frac{\partial E}{\partial x_j^{(m+1)}} \frac{\partial y_j^m}{\partial z_j} = g_j'(z_j)(-\sum_k \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial x_j^{(m+1)}}) \\ &= g_j'(z_j) \sum_k \delta_k^{(m+1)} w_{kj}^{(m+1)} \end{aligned} \qquad (28)$$

The Error Backpropagation Algorithm

[Wabos 1974, 1994] [Rumelhart, Hinton, & Williams,1986]

Forward Input Propagation
Move from m = 1 to M

$$z_j^{(m)} = \sum_j w_{ji}^{(m)} x_i^{(m)}, \; y_j^{(m)} = g_j(z_j^{(m)}) = x_j^{(m+1)} \qquad (29)$$



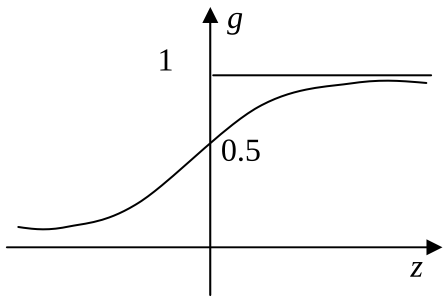
Error Back propagation

$$\delta_j^{(m)} = g'_j(z_j^{(m)})\sum_k \delta_k^{(m+1)} w_{kj}^{(m+1)} \qquad (30)$$

$$\delta_n^{(M)} = \left(y - \hat{y}^{(M)}\right) g'_n(z_n^{(M)})$$

8.5 Stabilizing Techniques

1). Properties of the sigmoid function



$$g(z) = \frac{1}{1+e^{-z}} \qquad (31)$$

Nonlinear, differentiable

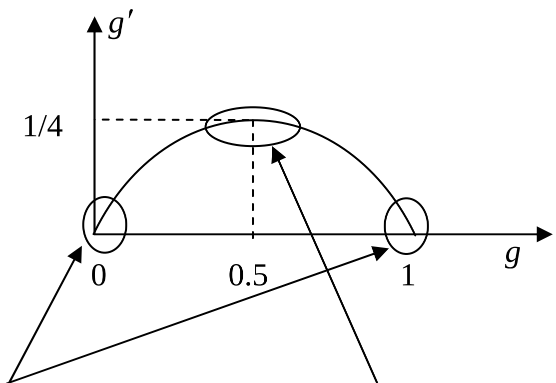
$$g'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = g(1-g) \qquad (32)$$

$$\Delta w_{ji} = \rho \delta_j^{(m)} x_i^{(m)}$$

$$\delta_j^{(m)} = g'_j \sum_k \delta_k^{(m+1)} w_{kj}^{(m+1)}$$

(33) $\Delta w_{ji} \propto g'_j(z_j)$

The incremental weight change is proportional to the derivative of $g(z)$.



For $-\infty < z < \infty$.
g varies $0 < g < 1$.
Max $g' = 1/4$
at $z = 0 \quad g = 0.5$

In these ranges weight changes are small.
$g \cong 0$ or $g \cong 1$
$|z| >> 1$.
$z_j = \sum_i w_{ji} x_i$

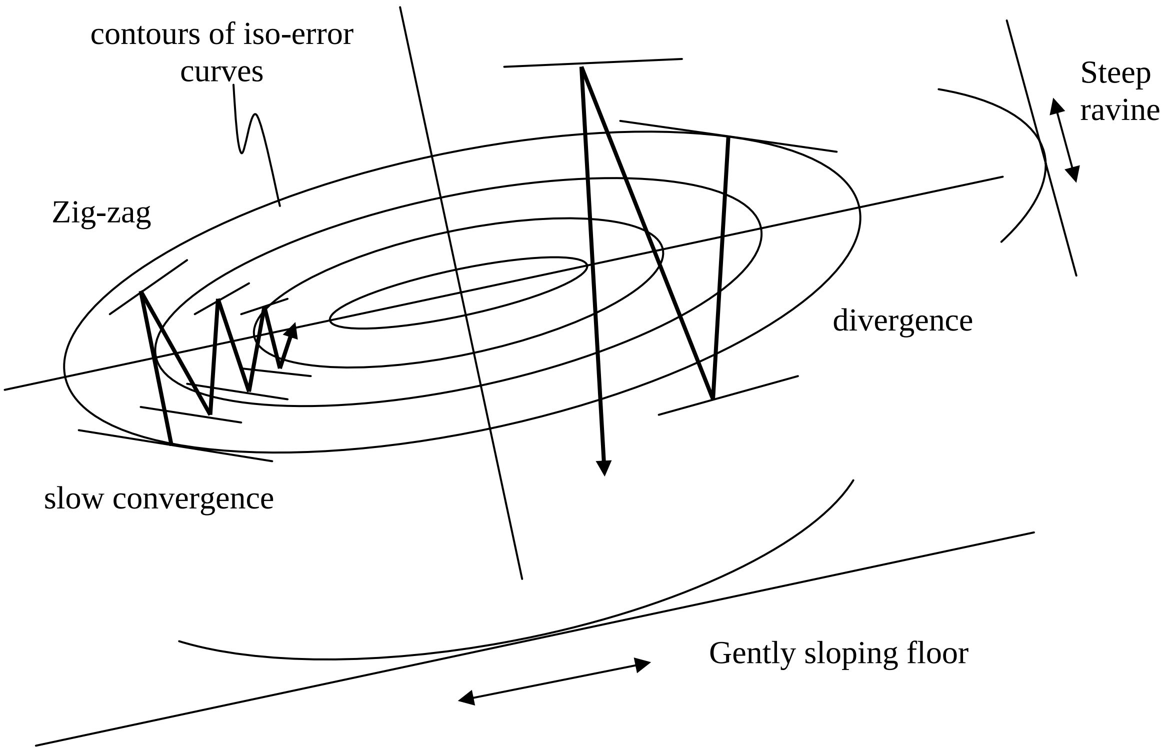
The largest weight change occurs in this range.
$g = 0.5, z = 0$
The unit has committed to neither 0 nor 1. The error backpropagation algorithm forces the unit to react significantly to that input.

Once the unit (j) has committed to take an output value of either “0” or “1”, the weight w_{ji} will no longer change very much for that inputs.

These features contribute to stabilizing the learning process

2) Smoothing by adding a momentum term

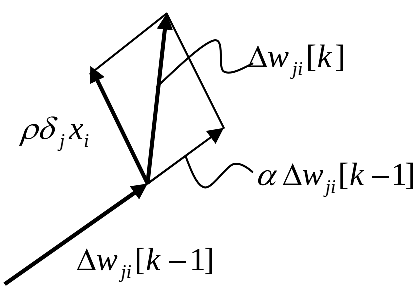
Ravine: a typical failure scenario of convergence



(34) $$\Delta w_{ji}[k] = \rho\, \delta_j x_i + \alpha\, \Delta w_{ji}[k-1]$$

proportionality constant — Previous weight change

Momentum = filters out High frequency oscillations



3) How to get rid of local minima

- Increase the number of hidden units
- Randomize the initial weights and repeat learning, then take the best one.