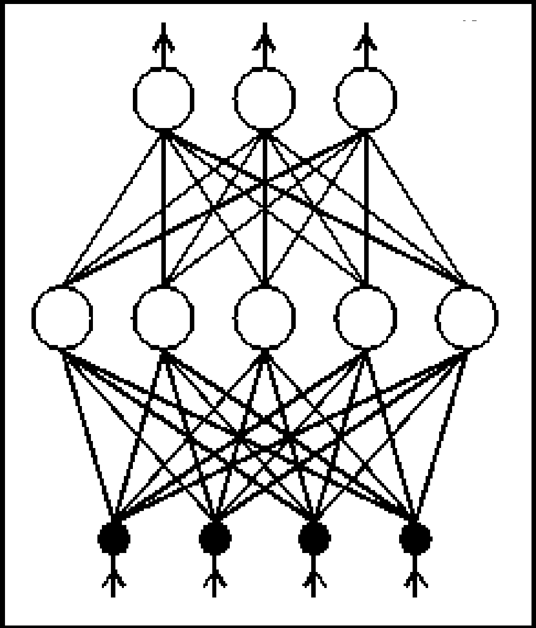
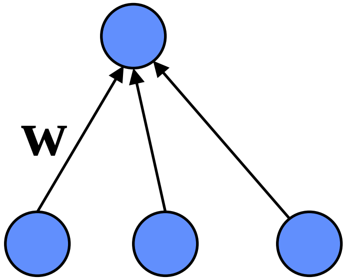
Computational Neuroscience
Supplementary Material



Backpropagation Algorithm for Supervised Learning in Multilayer Networks

Sigmoid Neurons

Output $v = g(\mathbf{w}^T\mathbf{u}) = g(\sum_i w_i u_i)$

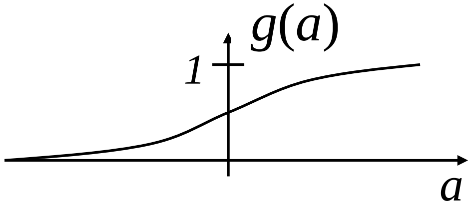


$\mathbf{u} = (u_1 \quad u_2 \quad u_3)^T$

Input nodes

Sigmoid output function:

$$g(a) = \frac{1}{1+e^{-\beta a}}$$

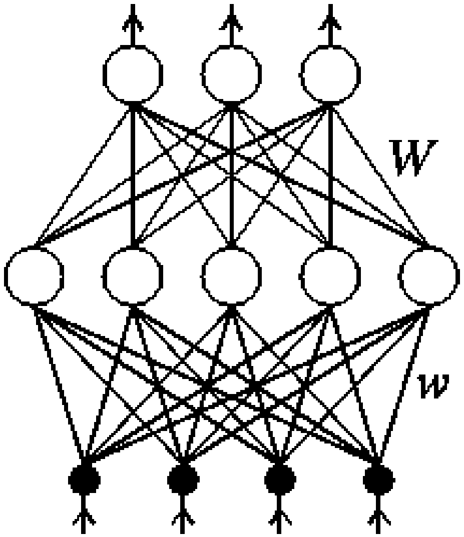


Parameter β controls the slope

Learning Multilayer Sigmoid Networks

$$v_i = g(\sum_j W_{ij} g(\sum_k w_{jk} u_k))$$

Output $\mathbf{v} = (v_1\ v_2\ \ldots\ v_J)^T$



Input $\mathbf{u} = (u_1\ u_2\ \ldots\ u_K)^T$

Desired output **d** also given

Learn weights that minimize output error:

$$E(\mathbf{W}, \mathbf{w}) = \frac{1}{2} \sum_i (d_i - v_i)^2$$

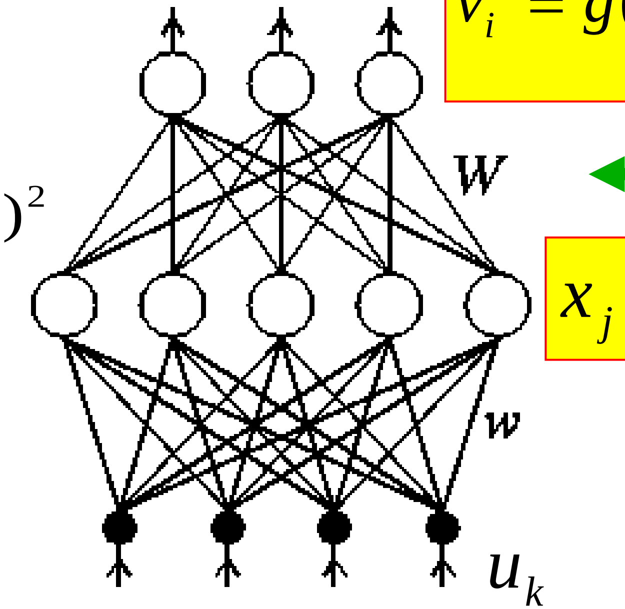
Use gradient descent!

Backpropagation Learning: Uppermost layer

Minimize output error:

$$E(\mathbf{W},\mathbf{w}) = \frac{1}{2}\sum_i (d_i - v_i)^2$$

$$v_i = g(\sum_j W_{ij} x_j)$$



Learning rule for <u>hidden-output weights **W**</u>:

$$W_{ij} \leftarrow W_{ij} - \varepsilon \frac{dE}{dW_{ij}}$$ {***gradient descent***}

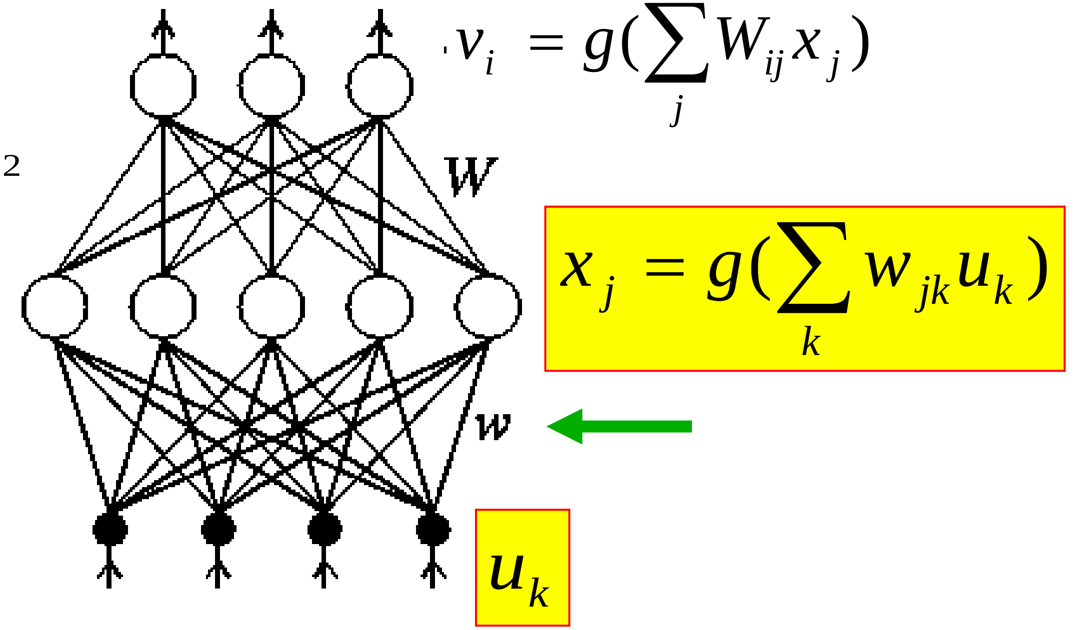
$$\frac{dE}{dW_{ij}} = -(d_i - v_i) g'(\sum_j W_{ij} x_j) x_j$$ {***delta rule***}

$g' =$ derivative of sigmoid function

Backpropagation: Inner layer

Minimize output error:

$$E(\mathbf{W},\mathbf{w}) = \frac{1}{2}\sum_i (d_i - v_i)^2$$



Learning rule for input-hidden weights **w**:

$$w_{jk} \leftarrow w_{jk} - \varepsilon \frac{dE}{dw_{jk}} \qquad \text{But}: \frac{dE}{dw_{jk}} = \frac{dE}{dx_j} \cdot \frac{dx_j}{dw_{jk}} \quad \{\textit{chain rule}\}$$

$$\frac{dE}{dw_{jk}} = \left[-\sum_i (d_i - v_i) g'(\sum_j W_{ij} x_j) W_{ij} \right] \cdot \left[g'(\sum_k w_{jk} u_k) u_k \right]$$