LMS (Least Mean Square)

Minimize the cost function (error function):

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)x_j^{(k)}$$

$$= -\sum_{k=1}^{p}\left(d^{(k)} - \mathbf{w}^T\mathbf{x}^{(k)}\right)x_j^{(k)} = -\sum_{k=1}^{p}\overbrace{\left(d^{(k)} - y^{(k)}\right)}^{\delta^{(k)}}x_j^{(k)}$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p}\delta^{(k)}x_j^{(k)}$$

$$\delta^{(k)} = d^{(k)} - y^{(k)}$$

Adaline Learning Rule

Minimize the cost function (error function):

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} \delta^{(k)} x_j^{(k)}$$

$$\delta^{(k)} = d^{(k)} - y^{(k)}$$

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m}\right)^T$$

$$\Delta\mathbf{w} = -\eta\nabla_{\mathbf{w}} E(\mathbf{w})$$ — Weight Modification Rule

Learning Modes

- Batch Learning Mode:

$$\Delta w_j = \eta \sum_{k=1}^{p} \delta^{(k)} x_j^{(k)}$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} \delta^{(k)} x_j^{(k)}$$

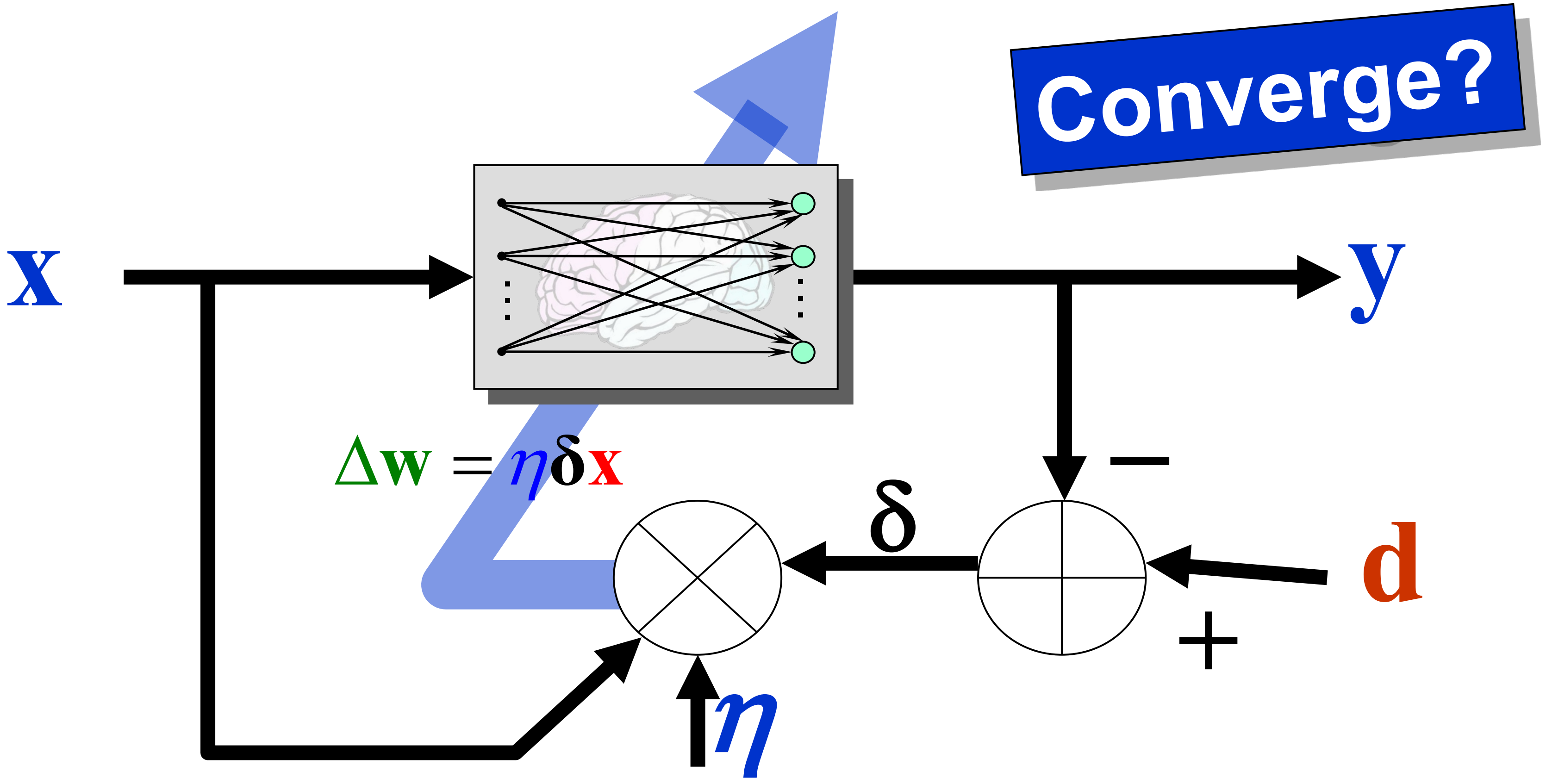
$$\delta^{(k)} = d^{(k)} - y^{(k)}$$

- Incremental Learning Mode:

$$\Delta w_j = \eta \delta^{(k)} x_j^{(k)}$$

Summary – Adaline Learning Rule

δ-Learning Rule
LMS Algorithm
Widrow-Hoff Learning Rule



LMS Convergence

Based on the independence theory (Widrow, 1976).

1. The successive input vectors are statistically independent.
2. At time t, the input vector $\mathbf{x}(t)$ is statistically independent of all previous samples of the desired response, namely $d(1), d(2), \ldots, d(t-1)$.
3. At time t, the desired response $d(t)$ is dependent on $\mathbf{x}(t)$, but statistically independent of all previous values of the desired response.
4. The input vector $\mathbf{x}(t)$ and desired response $d(t)$ are drawn from Gaussian distributed populations.

LMS Convergence

It can be shown that LMS is convergent if

$$0 < \eta < \frac{2}{\lambda_{\max}}$$

where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}_\mathbf{x}$ for the inputs.

$$\mathbf{R}_\mathbf{x} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} \mathbf{x}_i \mathbf{x}_i^T$$

LMS Convergence

It can be shown that LMS is convergent if

$$0 < \eta < \frac{2}{\lambda_{\max}}$$

Where $\boxed{\lambda_{max} \text{ is the largest eigenvalue of}}$ the correlation matrix $\mathbf{R_x}$ for the inputs.

$$\mathbf{R_x} = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{\infty} \mathbf{x}_i \mathbf{x}_i^T$$

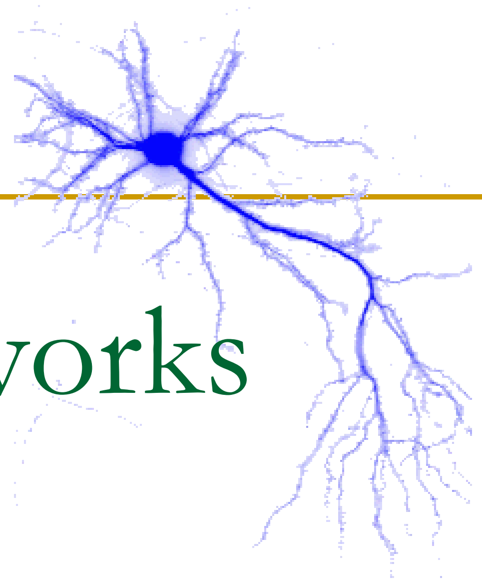
Since λ_{max} is hardly available, we commonly use

$$\boxed{0 < \eta < \frac{2}{tr(\mathbf{R_x})}}$$

Comparisons

	Perceptron Learning Rule	Adaline Learning Rule (Widrow-Hoff)
Fundamental	Hebbian Assumption	Gradient Descent
Convergence	In finite steps	Converge Asymptotically
Constraint	Linearly Separable	Linear Independence

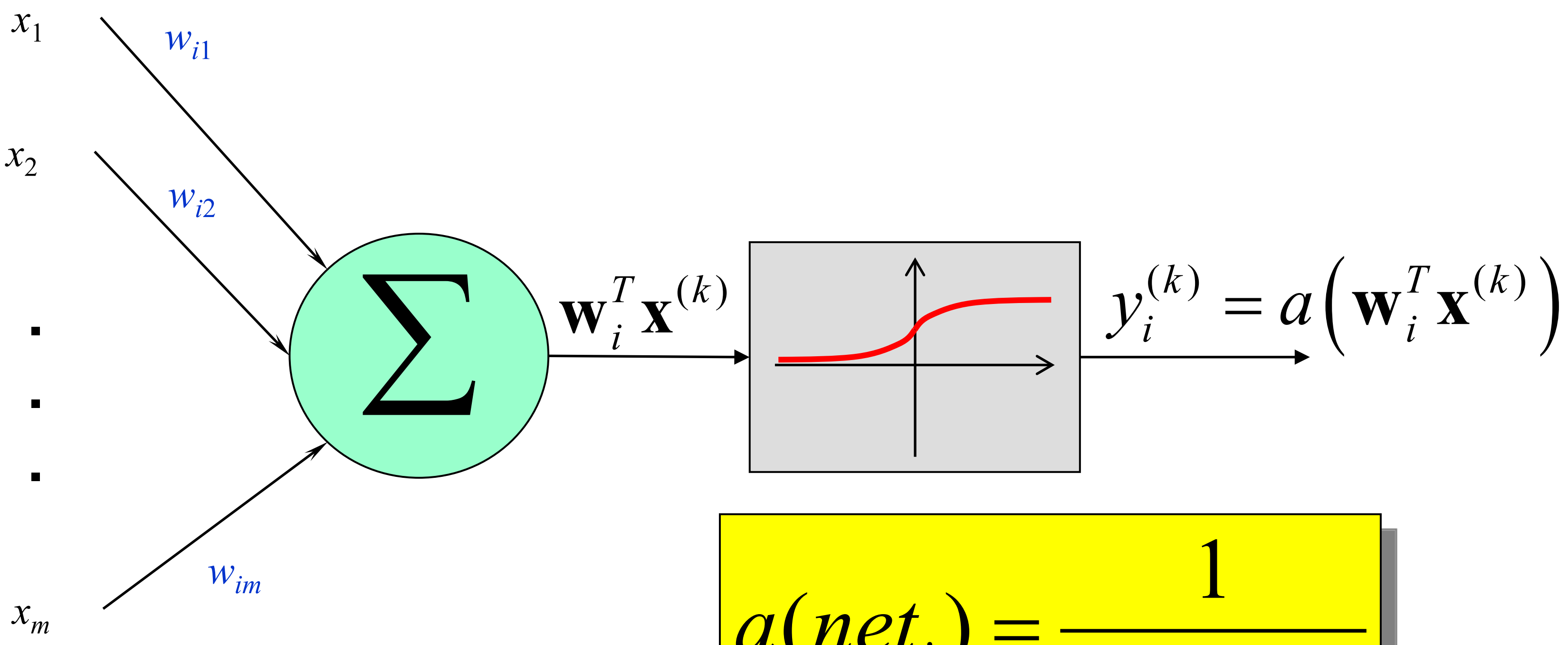
Feed-Forward Neural Networks



Learning Rules for
Single-Layered Perceptron Networks

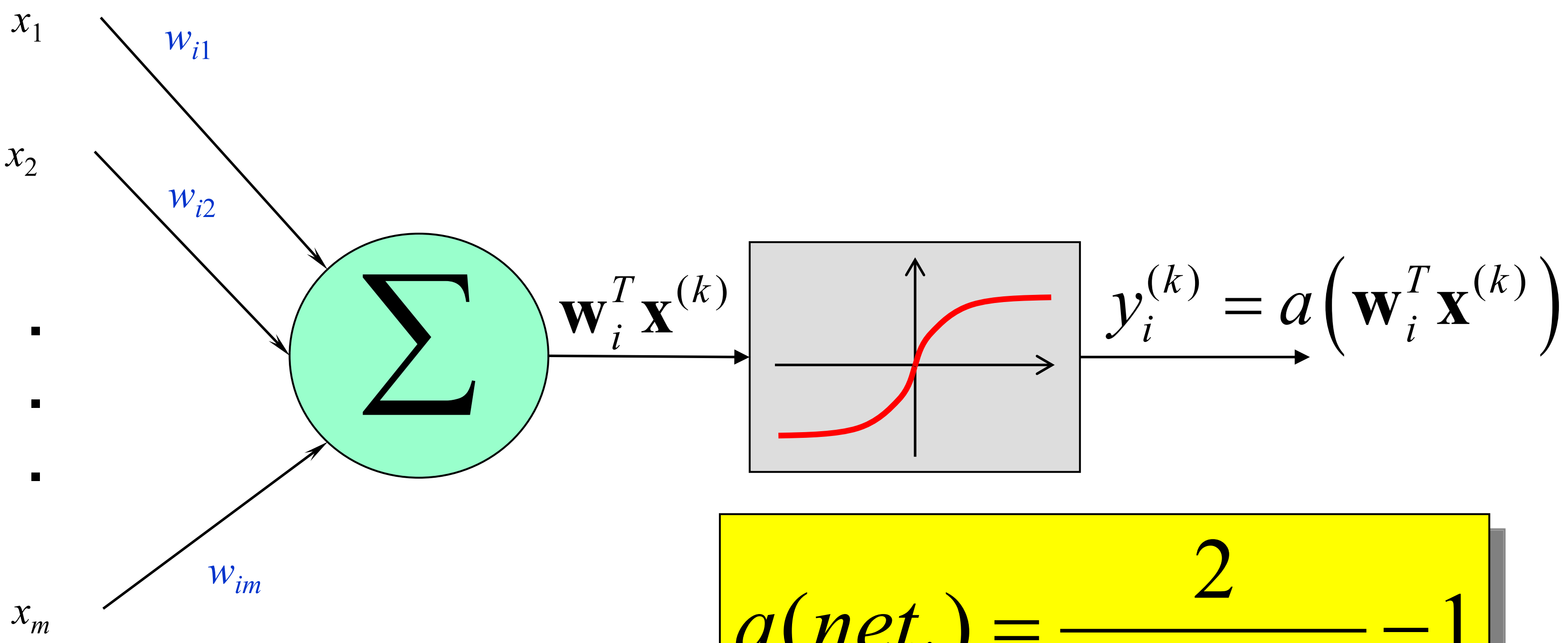
- Perceptron Learning Rule
- Adaline Leaning Rule
- δ-Learning Rule

Unipolar Sigmoid



$$a(net_i) = \frac{1}{1 + e^{-\lambda net_i}}$$

Bipolar Sigmoid



$$a(net_i) = \frac{2}{1+e^{-\lambda net_i}} - 1$$

Goal

Minimize
$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - y^{(k)}\right)^2$$

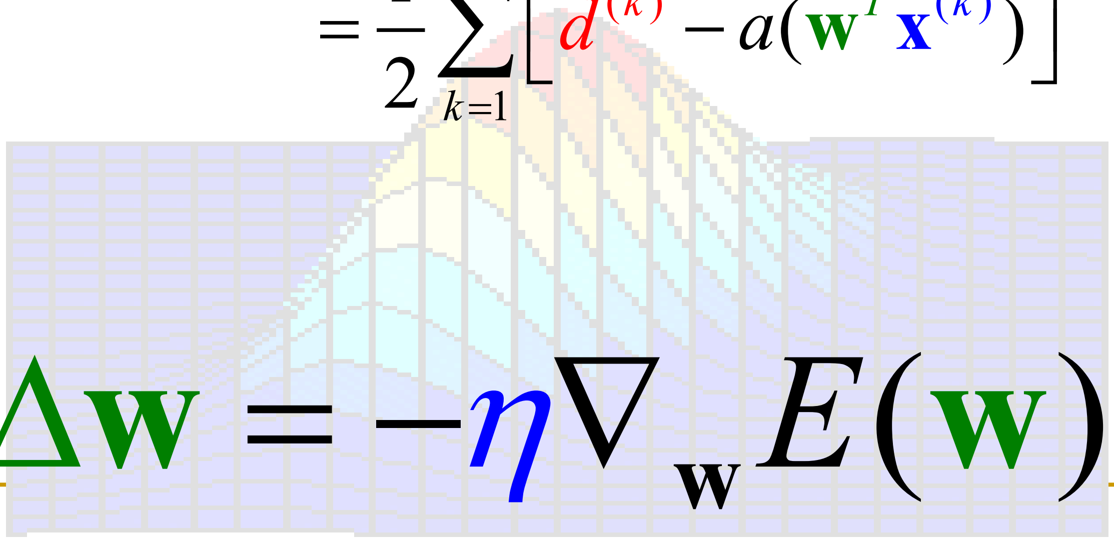
$$= \frac{1}{2}\sum_{k=1}^{p}\left[d^{(k)} - a\left(\mathbf{w}^T\mathbf{x}^{(k)}\right)\right]^2$$

Gradient Descent Algorithm

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m} \right)^T$$

Minimize $$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{p} (d^{(k)} - y^{(k)})^2$$

$$= \frac{1}{2} \sum_{k=1}^{p} \left[d^{(k)} - a(\mathbf{w}^T \mathbf{x}^{(k)}) \right]^2$$



$$\Delta \mathbf{w} = -\eta \nabla_{\mathbf{w}} E(\mathbf{w})$$

The Gradient

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m} \right)^T$$

$$\boxed{y^{(k)} = a\left(\mathbf{w}^T \mathbf{x}^{(k)}\right)}$$

Minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{p} (d^{(k)} - y^{(k)})^2$$

Depends on the activation function used.

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} (d^{(k)} - y^{(k)}) \frac{\partial y^{(k)}}{\partial w_j}$$

$$= -\sum_{k=1}^{p} (d^{(k)} - y^{(k)}) \underbrace{\frac{\partial a\left(net^{(k)}\right)}{\partial net^{(k)}}}_{?} \underbrace{\frac{\partial net^{(k)}}{\partial w_j}}_{?}$$

$$net^{(k)} = \mathbf{w}^T \mathbf{x}^{(k)} = \sum_{i=1}^{m} w_i x_i^{(k)} \Rightarrow \frac{\partial net^{(k)}}{\partial w_j} = x_j^{(k)}$$

Weight Modification Rule

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m} \right)^T$$

$$\boxed{y^{(k)} = a\left(net^{(k)}\right)}$$

$$\boxed{\delta^{(k)} = d^{(k)} - y^{(k)}}$$

Minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{p} (d^{(k)} - y^{(k)})^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} (d^{(k)} - y^{(k)}) x_j^{(k)} \frac{\partial a\left(net^{(k)}\right)}{\partial net^{(k)}}$$

Learning Rule

Batch

$$\Delta w_j = \eta \sum_{k=1}^{p} \delta^{(k)} x_j^{(k)} \frac{\partial a\left(net^{(k)}\right)}{\partial net^{(k)}}$$

Incremental

$$\Delta w_j = \eta \delta^{(k)} x_j^{(k)} \frac{\partial a\left(net^{(k)}\right)}{\partial net^{(k)}}$$

The Learning Efficacy

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m} \right)^T$$

$$y^{(k)} = a\left(net^{(k)}\right)$$

Minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{p} (d^{(k)} - y^{(k)})^2$$

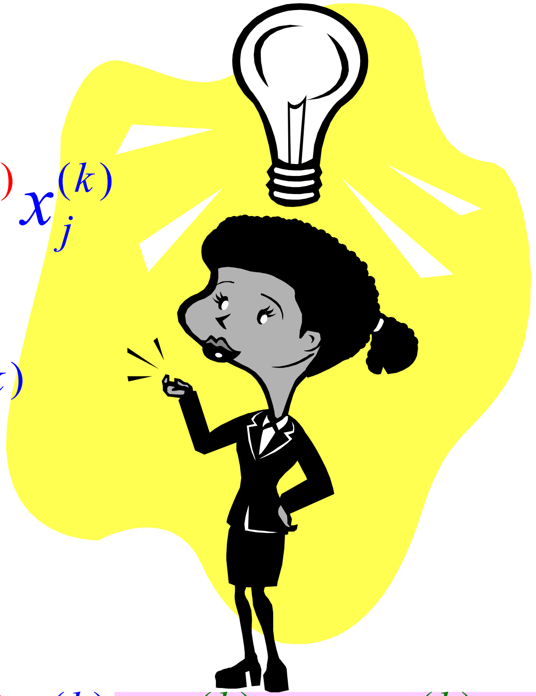
$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} (d^{(k)} - y^{(k)}) x_j^{(k)} \frac{\partial a\left(net^{(k)}\right)}{\partial net^{(k)}}$$

Sigmoid

Adaline	Unipolar	Bipolar
$a(net) = net$	$a(net) = \frac{1}{1+e^{-\lambda net}}$	$a(net) = \frac{2}{1+e^{-\lambda net}} - 1$
$\frac{\partial a(net)}{\partial net} = 1$	$\frac{\partial a(net)}{\partial net} = \lambda y^{(k)}(1-y^{(k)})$	**Exercise**

Comparisons

$$\lambda y^{(k)}(1-y^{(k)})$$



Adaline

Batch $\Delta w_j = \eta \sum_{k=1}^{p} \delta^{(k)} x_j^{(k)}$

Incremental $\Delta w_j = \eta \delta^{(k)} x_j^{(k)}$

Sigmoid

Batch $\Delta w_j = \eta \sum_{k=1}^{p} \delta^{(k)} x_j^{(k)} \lambda y^{(k)}(1-y^{(k)})$

Incremental $\Delta w_j = \eta \delta^{(k)} x_j^{(k)} \lambda y^{(k)}(1-y^{(k)})$

Learning on Output Neurons

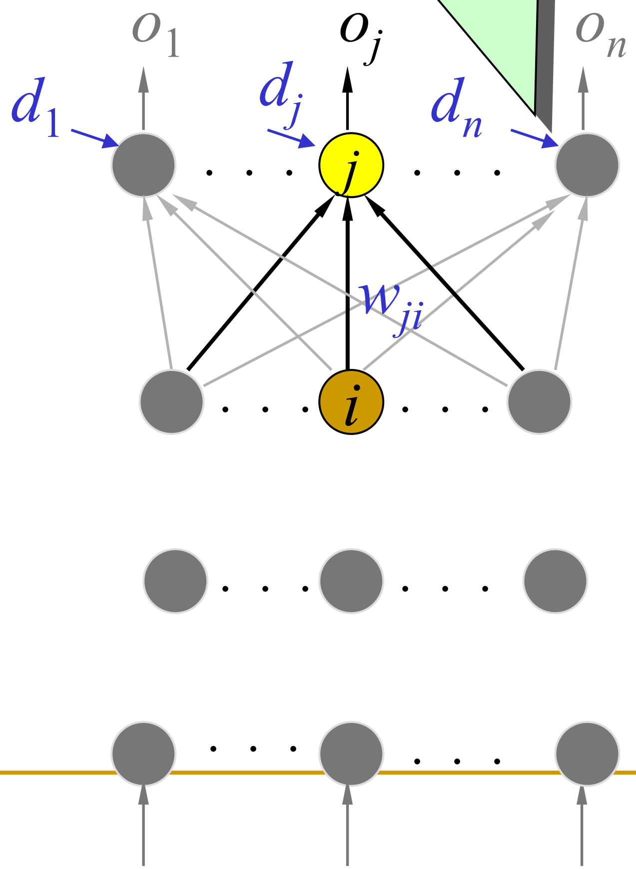
$$\delta_j^{(l)} = \frac{\partial E^{(l)}}{\partial net_j^{(l)}} = -(d_j^{(l)} - o_j^{(l)})\lambda o_j^{(l)}(1-o_j^{(l)})$$

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$

$$o_j^{(l)} = a(net_j^{(l)})$$

$$net_j^{(l)} = \sum w_{ji} o_i^{(l)}$$



$$\delta_j^{(l)}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p}\frac{\partial E^{(l)}}{\partial w_{ji}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ji}} = \boxed{\frac{\partial E^{(l)}}{\partial net_j^{(l)}}}\frac{\partial net_j^{(l)}}{\partial w_{ji}}$$

$$\frac{\partial E^{(l)}}{\partial net_j^{(l)}} = \frac{\partial E^{(l)}}{\partial o_j^{(l)}}\frac{\partial o_j^{(l)}}{\partial net_j^{(l)}}$$

$$-(d_j^{(l)} - o_j^{(l)})$$

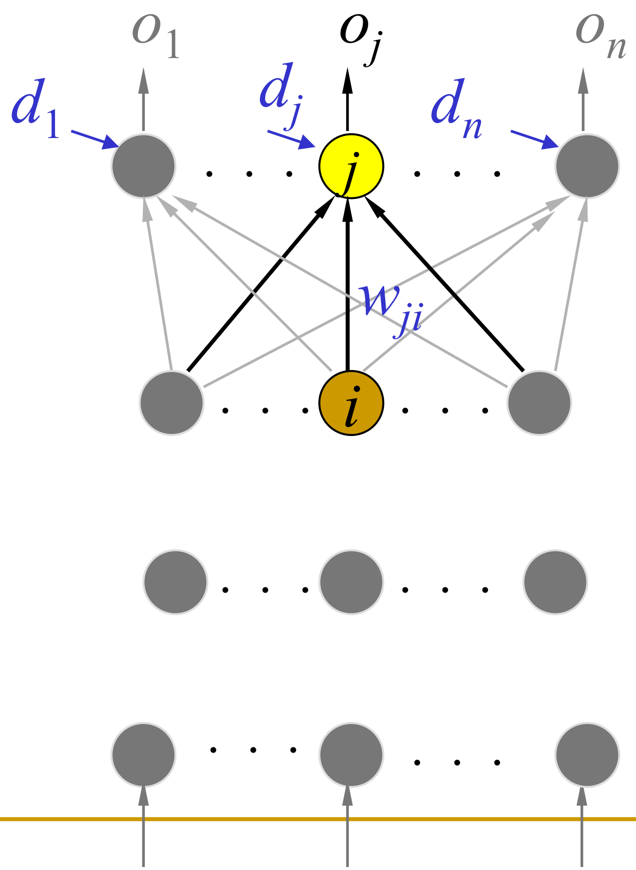
Using sigmoid,

$$\lambda o_j^{(l)}(1-o_j^{(l)})$$

Learning on Output Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$



$$o_j^{(l)} = a(net_j^{(l)})$$

$$net_j^{(l)} = \sum w_{ji} o_i^{(l)}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p} \frac{\partial E^{(l)}}{\partial w_{ji}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ji}} = \frac{\partial E^{(l)}}{\partial net_j^{(l)}} \boxed{\frac{\partial net_j^{(l)}}{\partial w_{ji}}} \rightarrow o_i^{(l)}$$

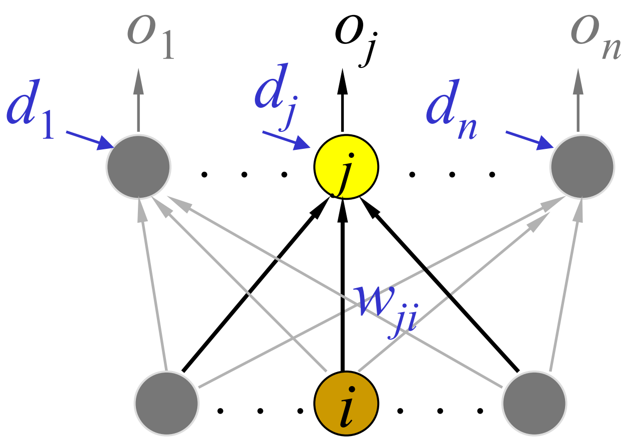
$$\frac{\partial E^{(l)}}{\partial w_{ji}} = \delta_j^{(l)} o_i^{(l)}$$

$$= -(d_j^{(l)} - o_j^{(l)})\lambda o_j^{(l)}(1 - o_j^{(l)}) o_i^{(l)}$$

Learning on Output Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$



$$o_j^{(l)} = a(net_j^{(l)})$$

$$net_j^{(l)} = \sum w_{ji} o_i^{(l)}$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p} \frac{\partial E^{(l)}}{\partial w_{ji}}$$

How to train the weights connecting to output neurons?

$$\frac{\partial E^{(l)}}{\partial w_{ji}} = \frac{\partial E^{(l)}}{\partial net_j^{(l)}} \boxed{\frac{\partial net_j^{(l)}}{\partial w_{ji}}} \rightarrow o_i^{(l)}$$

$$\frac{\partial E}{\partial w_{ji}} = \sum_{l=1}^{p} \delta_j^{(l)} o_i^{(l)}$$

$$\frac{\partial E^{(l)}}{\partial w_{ji}} = \delta_j^{(l)} o_i^{(l)}$$

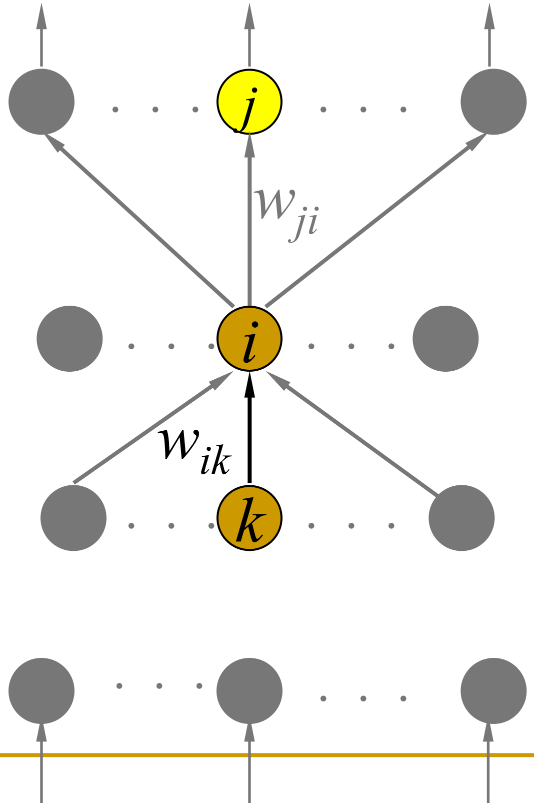
$$\Delta w_{ji} = -\eta \sum_{l=1}^{p} \delta_j^{(l)} o_i^{(l)}$$

$$= -(d_j^{(l)} - o_j^{(l)})\lambda o_j^{(l)}(1 - o_j^{(l)}) o_i^{(l)}$$

Learning on Hidden Neurons

$$\boxed{E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2}$$

$$\boxed{E = \sum_{l=1}^{p} E^{(l)}}$$



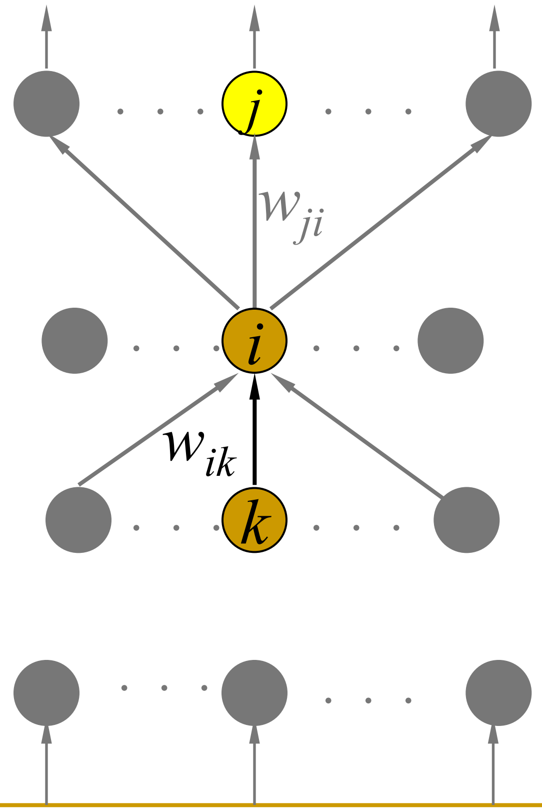
$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}}\sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p}\frac{\partial E^{(l)}}{\partial w_{ik}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ik}} = \underbrace{\frac{\partial E^{(l)}}{\partial net_i^{(l)}}}_{?}\ \underbrace{\frac{\partial net_i^{(l)}}{\partial w_{ik}}}_{?}$$

Learning on Hidden Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$



$$\delta_i^{(l)}$$

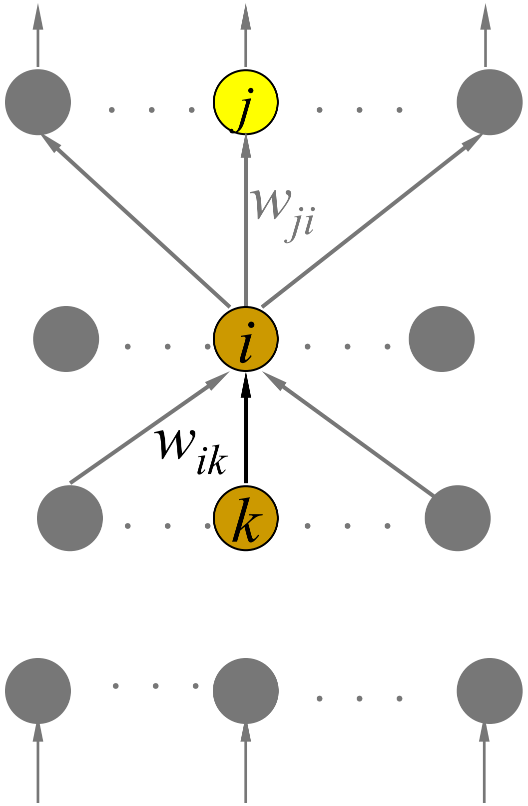
$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}} \sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p} \frac{\partial E^{(l)}}{\partial w_{ik}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ik}} = \frac{\partial E^{(l)}}{\partial net_i^{(l)}} \boxed{\frac{\partial net_i^{(l)}}{\partial w_{ik}}} \rightarrow o_k^{(l)}$$

Learning on Hidden Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$



$\delta_i^{(l)}$

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}}\sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p}\frac{\partial E^{(l)}}{\partial w_{ik}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ik}} = \boxed{\frac{\partial E^{(l)}}{\partial net_i^{(l)}}}\frac{\partial net_i^{(l)}}{\partial w_{ik}} \rightarrow o_k^{(l)}$$

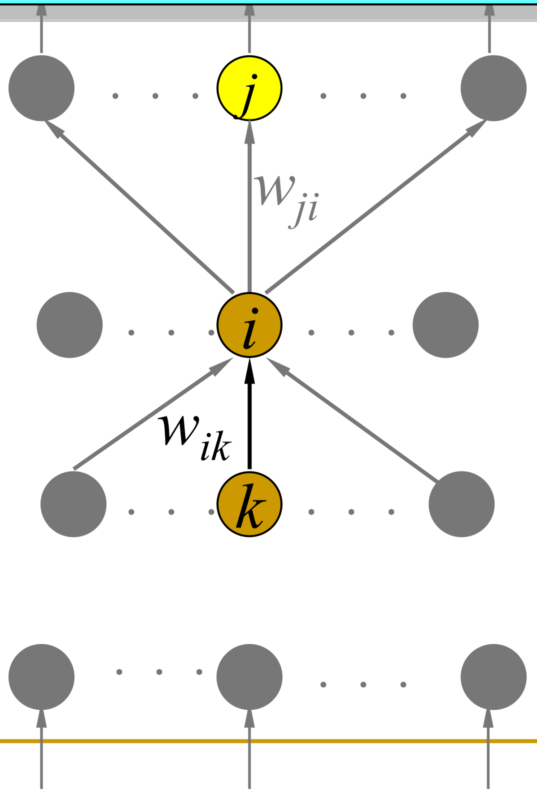
$$\frac{\partial E^{(l)}}{\partial net_i^{(l)}} = \underbrace{\frac{\partial E^{(l)}}{\partial o_i^{(l)}}}_{?}\frac{\partial o_i^{(l)}}{\partial net_i^{(l)}} \rightarrow \lambda o_i^{(l)}(1-o_i^{(l)})$$

Learning on Hidden Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$

$$\delta_i^{(l)} = \frac{\partial E^{(l)}}{\partial net_i^{(l)}} = \lambda o_i^{(l)}(1-o_i^{(l)})\sum_j w_{ji}\delta_j^{(l)}$$



$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}}\sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p}\frac{\partial E^{(l)}}{\partial w_{ik}}$$

$\delta_i^{(l)}$

$$\frac{\partial E^{(l)}}{\partial w_{ik}} = \boxed{\frac{\partial E^{(l)}}{\partial net_i^{(l)}}}\,\frac{\partial net_i^{(l)}}{\partial w_{ik}} \quad \frac{\partial net_i^{(l)}}{\partial w_{ik}} \rightarrow o_k^{(l)}$$

$$\frac{\partial E^{(l)}}{\partial net_i^{(l)}} = \boxed{\frac{\partial E^{(l)}}{\partial o_i^{(l)}}}\,\frac{\partial o_i^{(l)}}{\partial net_i^{(l)}} \quad \frac{\partial o_i^{(l)}}{\partial net_i^{(l)}} \rightarrow \lambda o_i^{(l)}(1-o_i^{(l)})$$

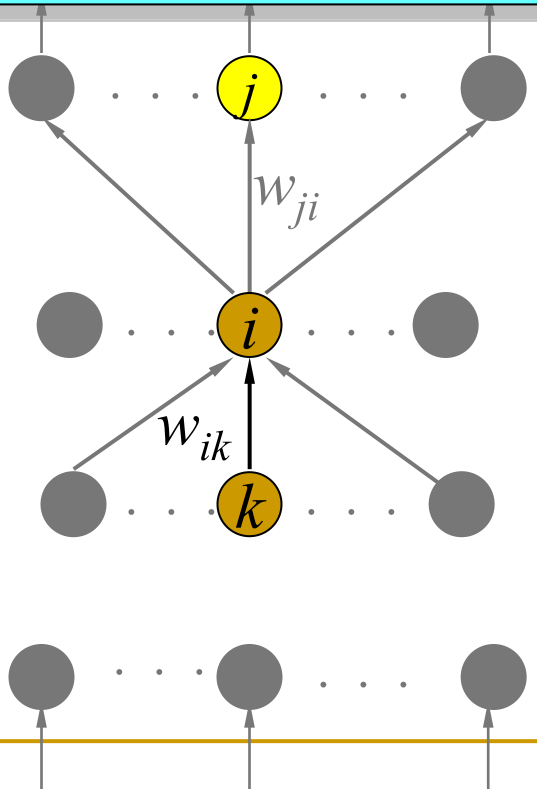
$$\frac{\partial E^{(l)}}{\partial o_i^{(l)}} = \sum_j \underbrace{\frac{\partial E^{(l)}}{\partial net_j^{(l)}}}_{\delta_j^{(l)}}\,\underbrace{\frac{\partial net_j^{(l)}}{\partial o_i^{(l)}}}_{w_{ji}}$$

Learning on Hidden Neurons

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

$$E = \sum_{l=1}^{p} E^{(l)}$$

$$\delta_i^{(l)} = \frac{\partial E^{(l)}}{\partial net_i^{(l)}} = \lambda o_i^{(l)}(1 - o_i^{(l)})\sum_j w_{ji}\delta_j^{(l)}$$



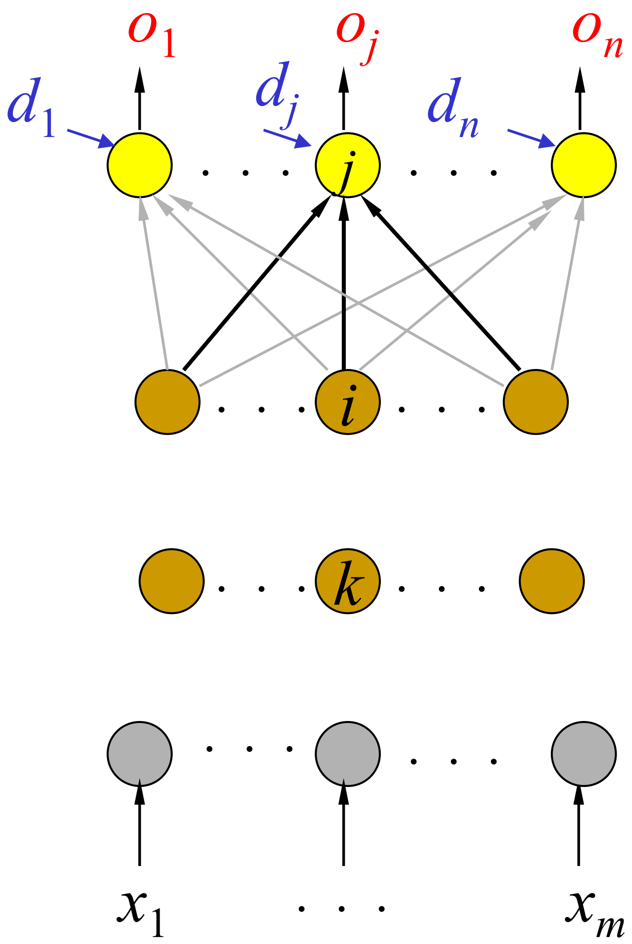
$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}}\sum_{l=1}^{p} E^{(l)} = \sum_{l=1}^{p}\frac{\partial E^{(l)}}{\partial w_{ik}}$$

$$\frac{\partial E^{(l)}}{\partial w_{ik}} = \boxed{\frac{\partial E^{(l)}}{\partial net_i^{(l)}}}\frac{\partial net_i^{(l)}}{\partial w_{ik}} \rightarrow o_k^{(l)}$$

$$\frac{\partial E}{\partial w_{ik}} = \sum_{l=1}^{p}\delta_i^{(l)} o_k^{(l)}$$

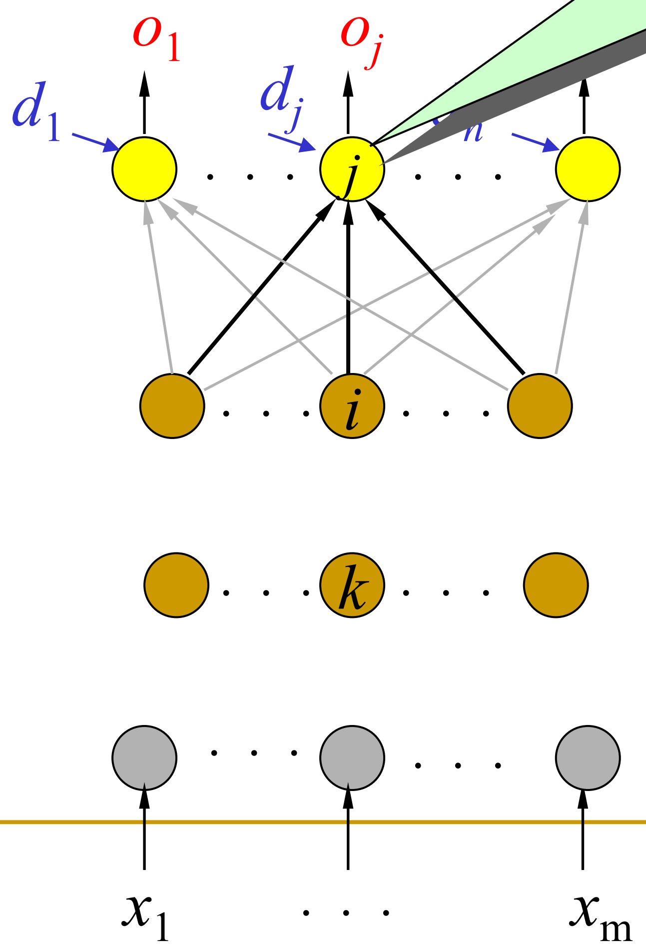
$$\Delta w_{ik} = -\eta\sum_{l=1}^{p}\delta_i^{(l)} o_k^{(l)}$$

Back Propagation



Back Propagation

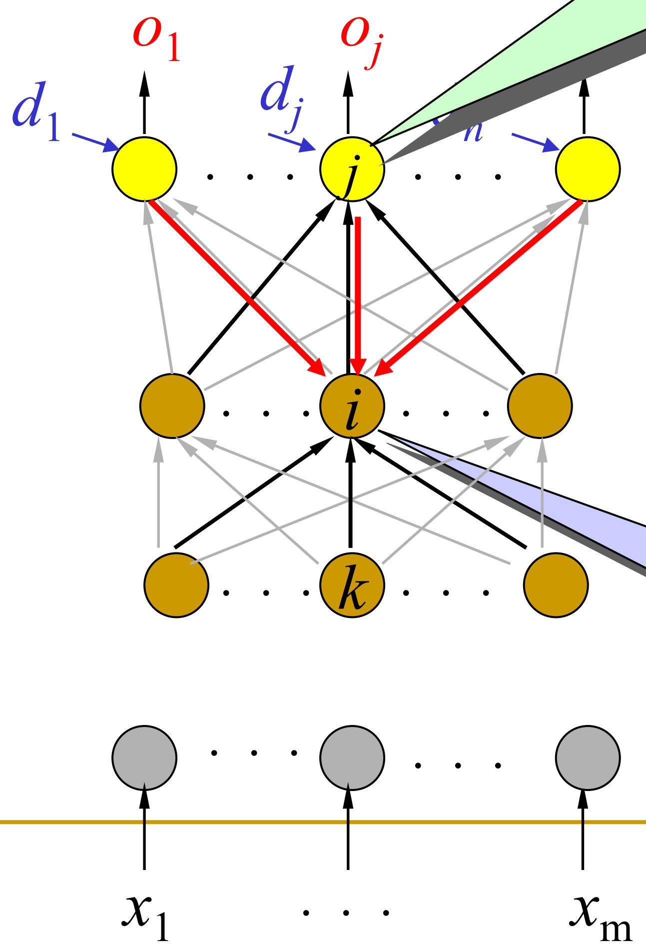
$$\delta_j^{(l)} = \frac{\partial E^{(l)}}{\partial net_j^{(l)}} = -\lambda (d_j^{(l)} - o_j^{(l)}) o_j^{(l)} (1 - o_j^{(l)})$$



$$\Delta w_{ji} = -\eta \sum_{l=1}^{p} \delta_j^{(l)} o_i^{(l)}$$

Back Propagation

$$\delta_j^{(l)} = \frac{\partial E^{(l)}}{\partial net_j^{(l)}} = -\lambda(d_j^{(l)} - o_j^{(l)})o_j^{(l)}(1 - o_j^{(l)})$$



$$\Delta w_{ji} = -\eta \sum_{l=1}^{p} \delta_j^{(l)} o_i^{(l)}$$

$$\Delta w_{ik} = -\eta \sum_{l=1}^{p} \delta_i^{(l)} o_k^{(l)}$$

$$\delta_i^{(l)} = \frac{\partial E^{(l)}}{\partial net_i^{(l)}} = \lambda o_i^{(l)}(1 - o_i^{(l)}) \sum_j w_{ji} \delta_j^{(l)}$$

Learning Factors

- Initial Weights
- Learning Constant (η)
- Cost Functions
- Momentum
- Update Rules
- Training Data and Generalization
- Number of Layers
- Number of Hidden Nodes

Reading Assignments

- Shi Zhong and Vladimir Cherkassky, "Factors Controlling Generalization Ability of MLP Networks." In Proc. IEEE Int. Joint Conf. on Neural Networks, vol. 1, pp. 625-630, Washington DC. July 1999. (http://www.cse.fau.edu/~zhong/pubs.htm)

- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). "Learning Internal Representations by Error Propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. I, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. MIT Press, Cambridge (1986).
  (http://www.cnbc.cmu.edu/~plaut/85-419/papers/RumelhartETAL86.backprop.pdf).