



Lecture Slides for

INTRODUCTION TO

Machine Learning

ETHEM ALPAYDIN

© The MIT Press, 2004

alpaydin@boun.edu.tr

<http://www.cmpe.boun.edu.tr/~ethem/i2ml>

A decorative graphic on the left side of the slide consisting of a grid of orange squares of varying shades and sizes, arranged in a stepped pattern.

CHAPTER 11:

Multilayer Perceptrons



- a/memory
-
- ```
graph TD; A(()) --> B(()); B --> C(()); C --> D(()); D --> E(()); E --> C; F(()) --> D; G(()) --> E; H(()) --> E;
```



# *Understanding the Brain*

- Levels of analysis (Marr, 1982)
  1. Computational theory
  2. Representation and algorithm
  3. Hardware implementation
- Reverse engineering: From hardware to theory
- Parallel processing: SIMD vs MIMD
  - Neural net: SIMD with modifiable local memory
  - Learning: Update by training/experience

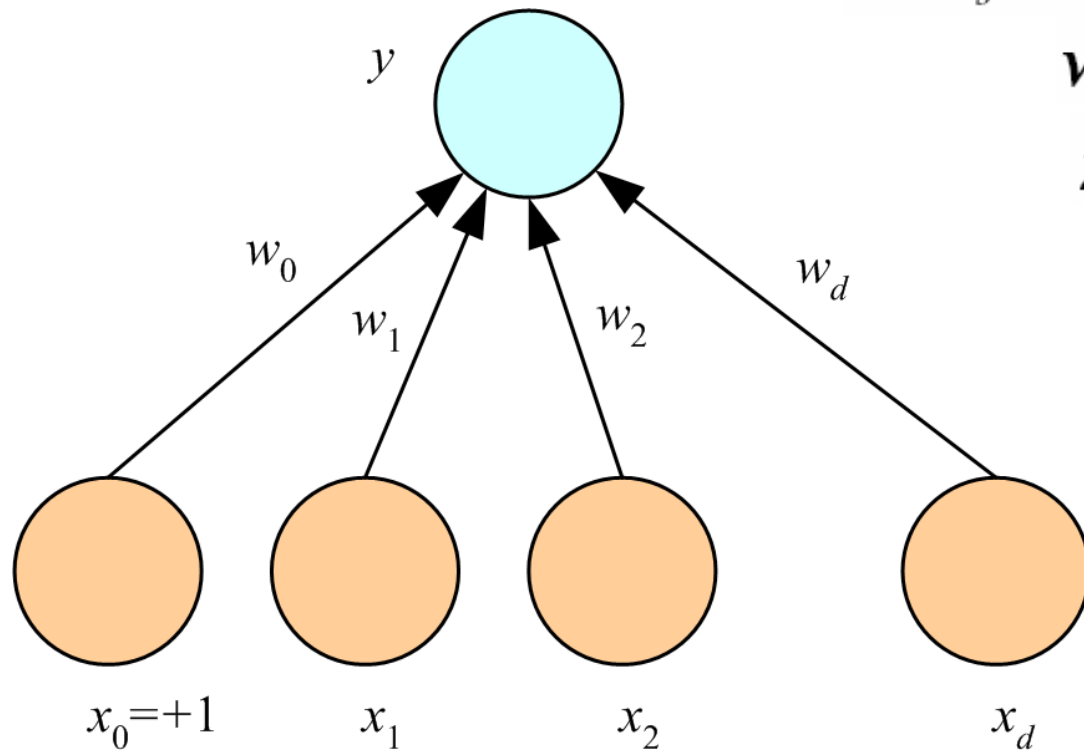
# Perceptron

$$y = \sum_{j=1}^d w_j x_j + w_0 = \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} = [w_0, w_1, \dots, w_d]^T$$

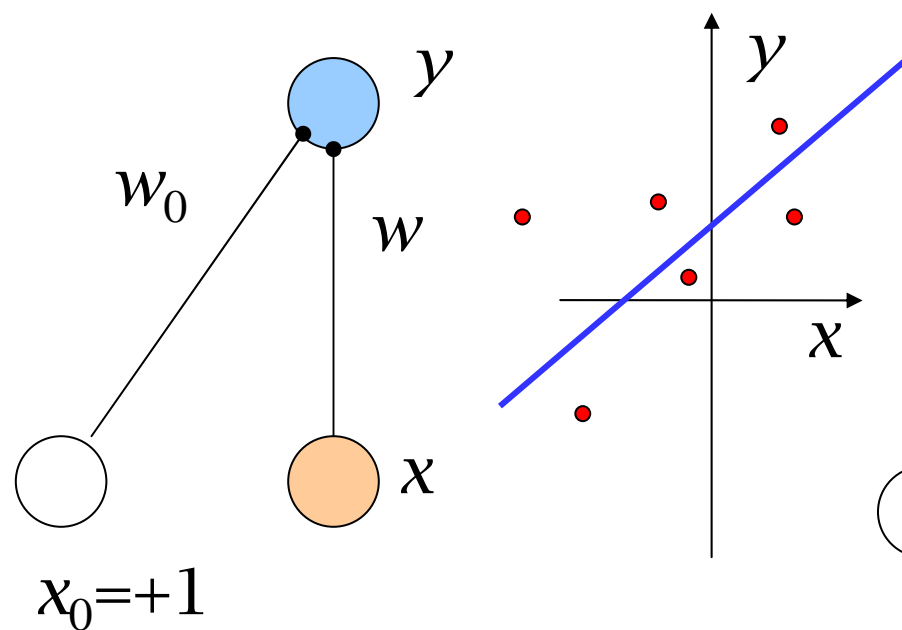
$$\mathbf{x} = [1, x_1, \dots, x_d]^T$$

(Rosenblatt, 1962)

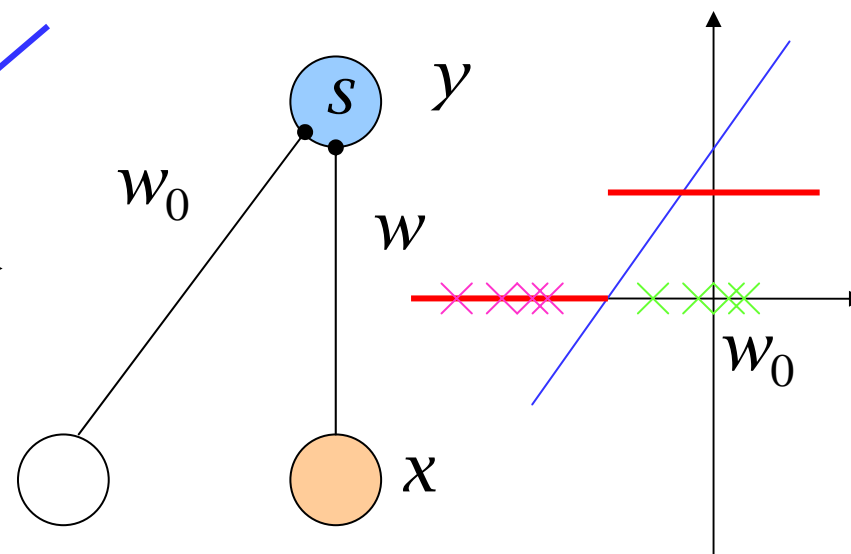


# What a Perceptron Does

■ Regression:  $y = wx + w_0$



■ Classification:  $y = 1(w x + w_0 > 0)$



$$y = \text{sigmoid}(o) = \frac{1}{1 + \exp[-w^T x]}$$

## *K Outputs*

Regression:

$$y_i = \sum_{j=1}^d w_{ij} x_j + w_{i0} = \mathbf{w}_i^T \mathbf{x}$$

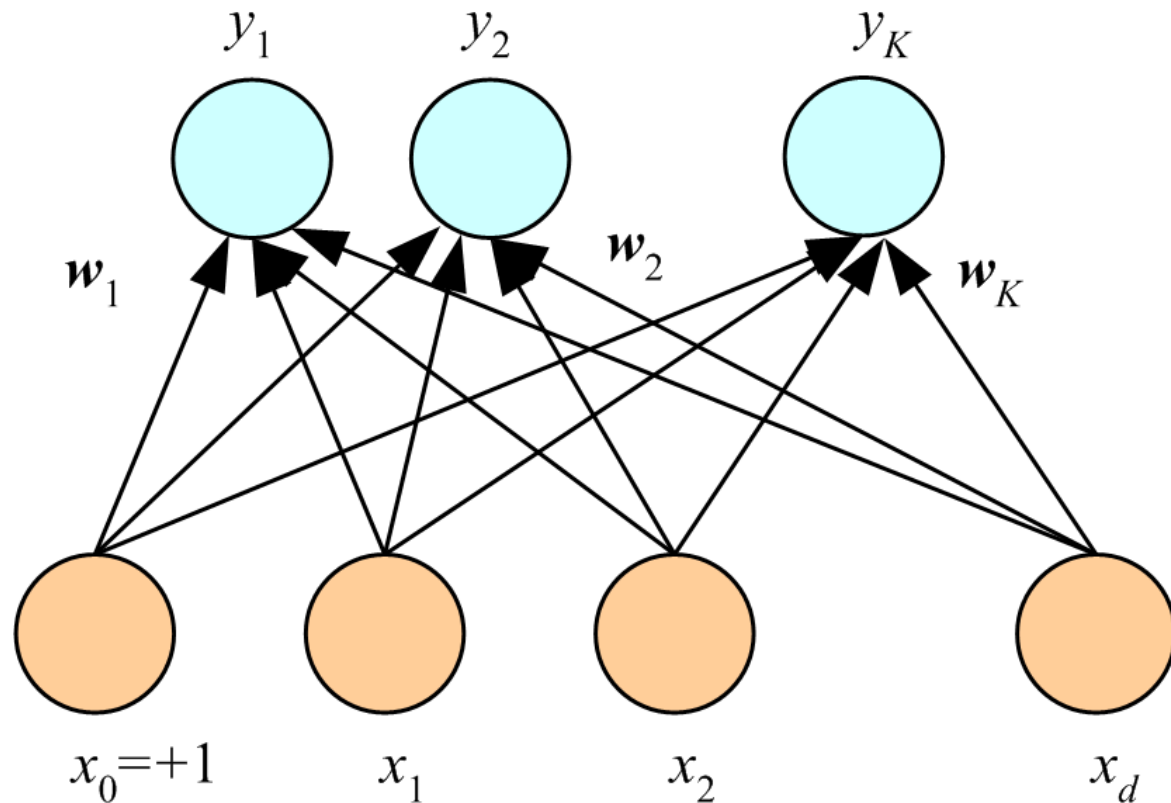
$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

Classification:

$$o_i = \mathbf{w}_i^T \mathbf{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

choose  $C_i$   
if  $y_i = \max_k y_k$





# Training

- Online (instances seen one by one) vs batch (whole sample) learning:
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern
- Generic update rule (LMS rule):

$$\Delta w_{ij}^t = \eta(r_i^t - y_i^t)x_j^t$$

Update = LearningFactor · (DesiredOutput – ActualOutput) · Input





# *Training a Perceptron: Regression*

- Regression (Linear output):

$$E^t(\mathbf{w}|\mathbf{x}^t, r^t) = \frac{1}{2} (r^t - y^t)^2 = \frac{1}{2} [r^t - (\mathbf{w}^T \mathbf{x}^t)]^2$$

$$\Delta w_j^t = \eta (r^t - y^t) x_j^t$$



# Classification

- Single sigmoid output

$$y^t = \text{sigmoid}(\mathbf{w}^T \mathbf{x}^t)$$

$$E^t(\{\mathbf{w}_i\}_i | \mathbf{x}^t, \mathbf{r}^t) = - \sum_i r_i^t \log y_i^t + (1 - r_i^t) \log(1 - y_i^t)$$

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t$$

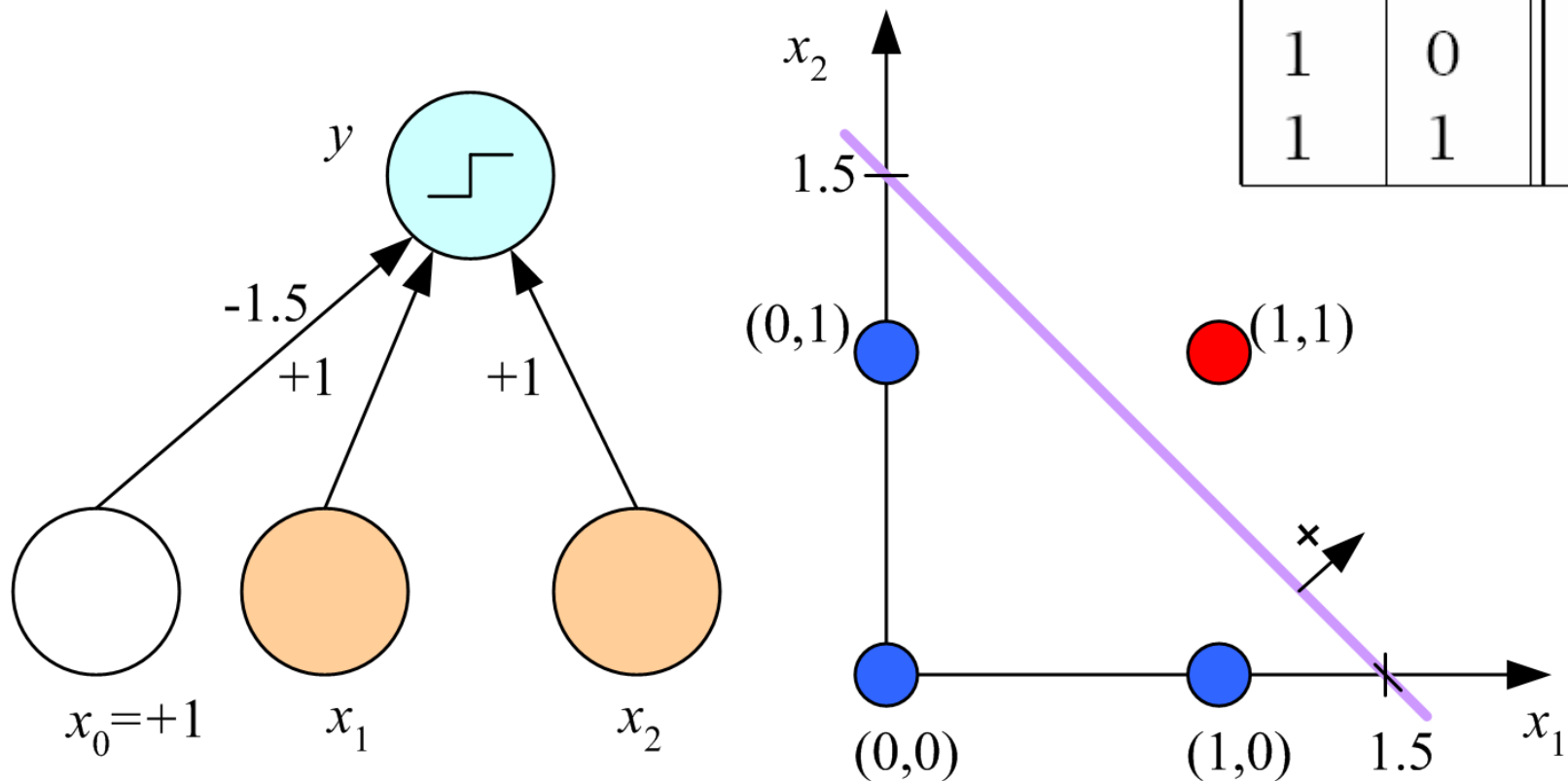
- $K > 2$  softmax outputs

$$y_i^t = \frac{\exp \mathbf{w}_i^T \mathbf{x}^t}{\sum_k \exp \mathbf{w}_k^T \mathbf{x}^t} \quad E^t(\{\mathbf{w}_i\}_i | \mathbf{x}^t, \mathbf{r}^t) = - \sum_i r_i^t \log y_i^t$$

$$\Delta w_{ij}^t = \eta(r_i^t - y_i^t)x_j^t$$

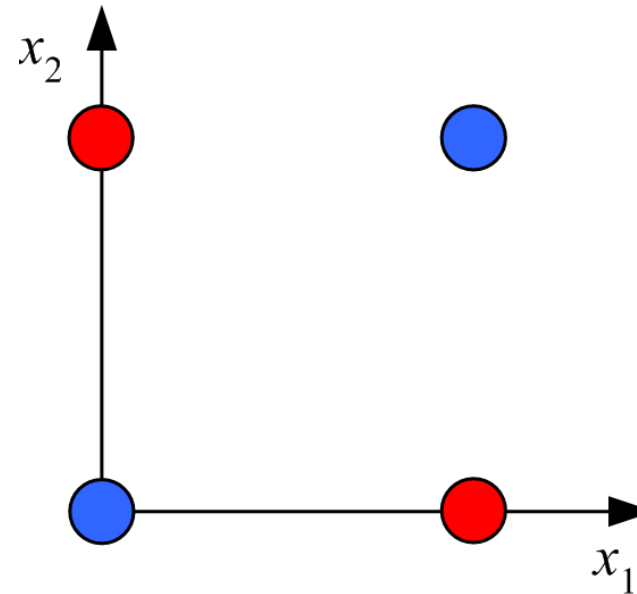
# Learning Boolean AND

| $x_1$ | $x_2$ | $r$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 0     | 1     | 0   |
| 1     | 0     | 0   |
| 1     | 1     | 1   |



# XOR

| $x_1$ | $x_2$ | $r$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 0     | 1     | 1   |
| 1     | 0     | 1   |
| 1     | 1     | 0   |



- No  $w_0, w_1, w_2$  satisfy:

$$w_0 \leq 0$$

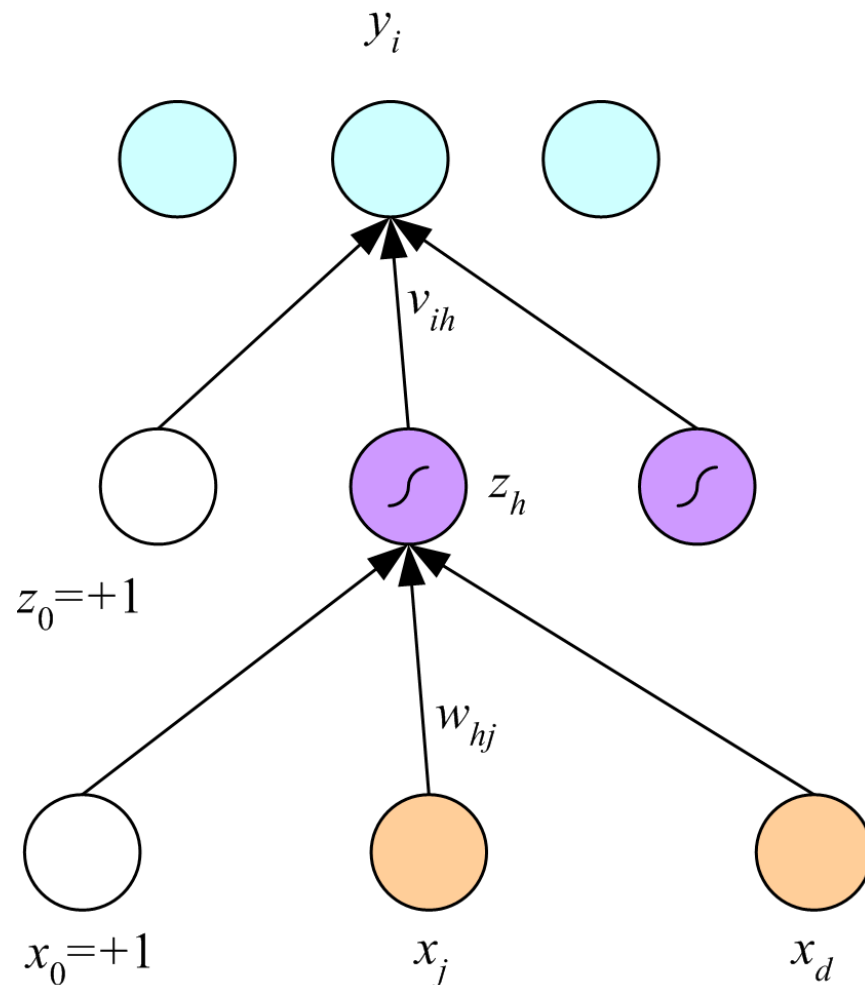
$$w_2 + w_0 > 0$$

$$w_1 + w_0 > 0$$

$$w_1 + w_2 + w_0 \leq 0$$

(Minsky and Papert, 1969)

# Multilayer Perceptrons

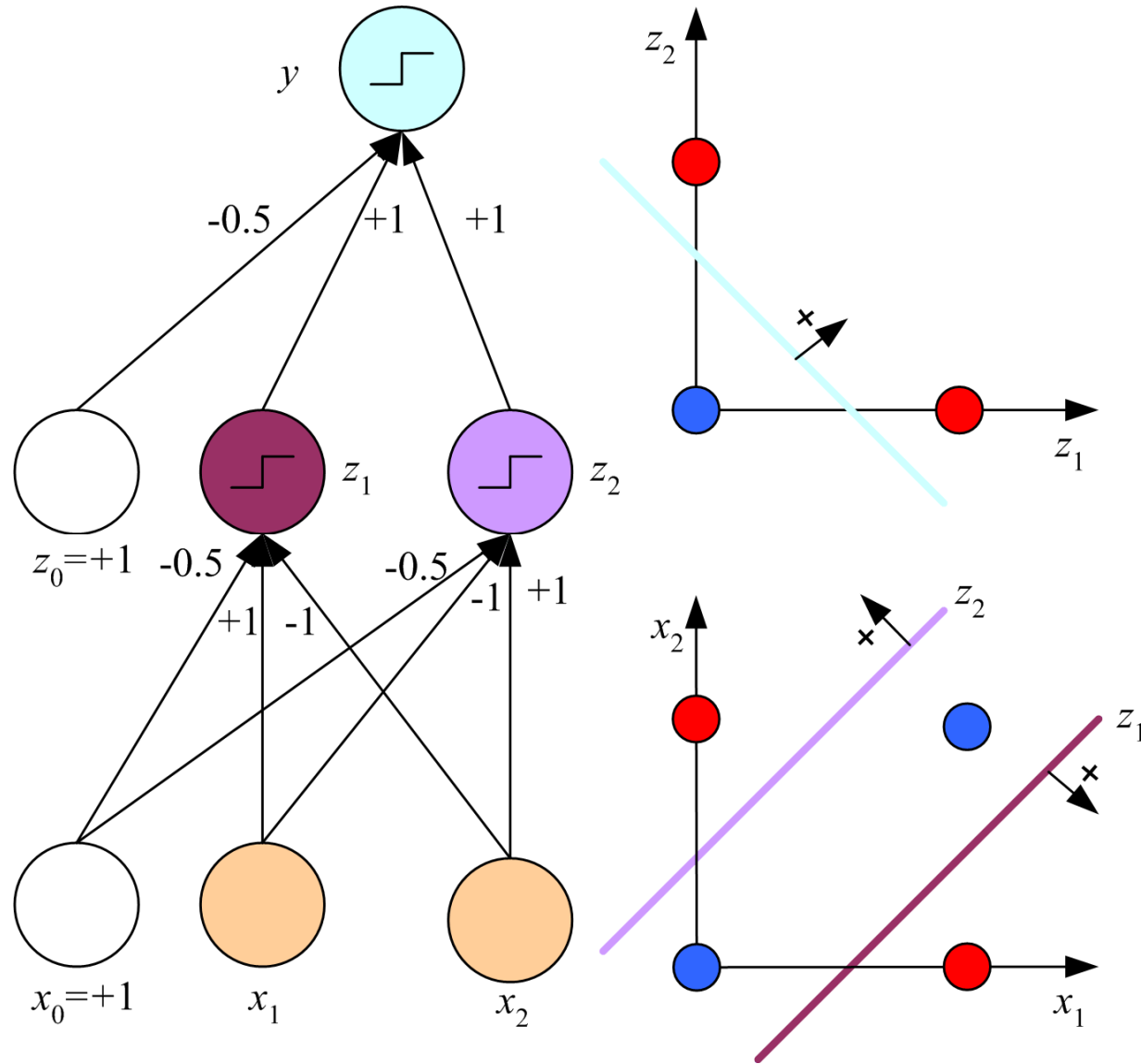


$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^H v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}(\mathbf{w}_h^T \mathbf{x})$$

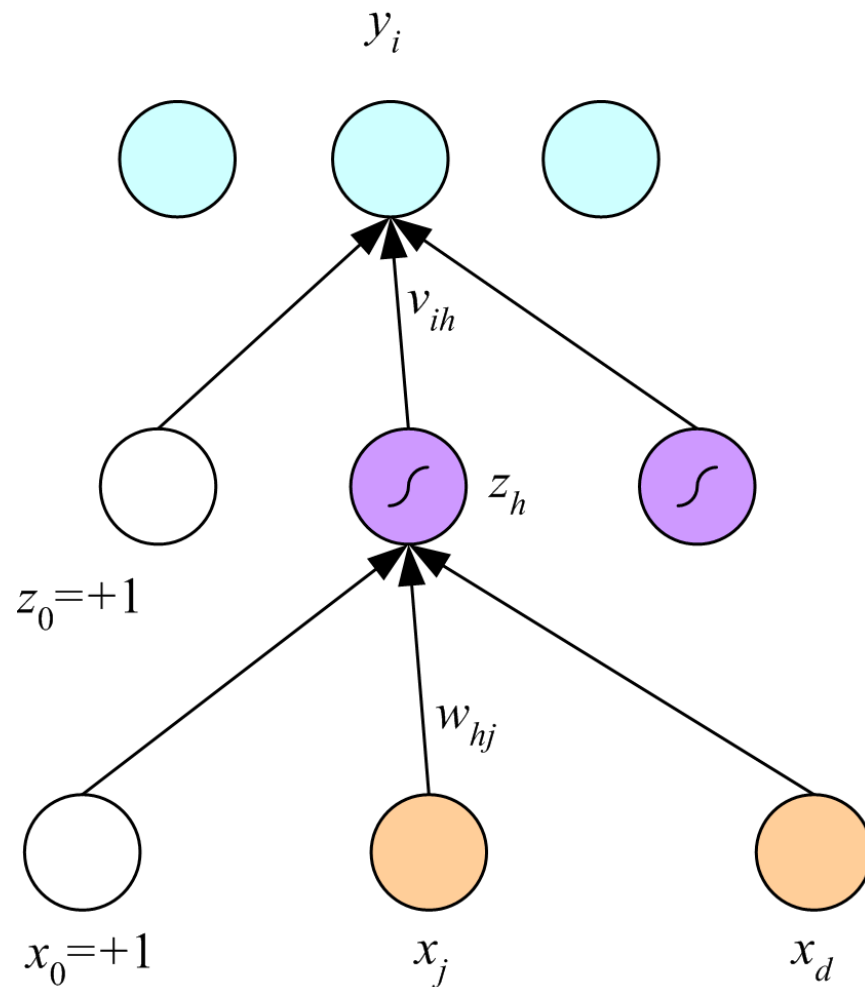
$$= \frac{1}{1 + \exp \left[ - \left( \sum_{j=1}^d w_{hj} x_j + w_{h0} \right) \right]}$$

(Rumelhart et al., 1986)



$$x_1 \text{ XOR } x_2 = (x_1 \text{ AND } \sim x_2) \text{ OR } (\sim x_1 \text{ AND } x_2)$$

# Backpropagation



$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^H v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}(\mathbf{w}_h^T \mathbf{x})$$

$$= \frac{1}{1 + \exp \left[ - \left( \sum_{j=1}^d w_{hj} x_j + w_{h0} \right) \right]}$$

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}$$

# Regression

$$y^t = \sum_{h=1}^H v_h z_h^t + v_0$$

Forward

$$z_h = \text{sigmoid}(\mathbf{w}_h^T \mathbf{x})$$

$\mathbf{x}$

$$E(\mathbf{W}, \mathbf{v} | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - y^t)^2$$

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t$$

Backward

$$\begin{aligned} \Delta w_{hj} &= -\eta \frac{\partial E}{\partial w_{hj}} \\ &= -\eta \sum_t \frac{\partial E^t}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}} \\ &= -\eta \sum_t \underbrace{-(r^t - y^t)}_{\partial E^t / \partial y^t} \underbrace{v_h}_{\partial y^t / \partial z_h^t} \underbrace{z_h^t (1 - z_h^t) x_j^t}_{\partial z_h^t / \partial w_{hj}} \\ &= \eta \sum_t (r^t - y^t) v_h \boxed{z_h^t (1 - z_h^t)} x_j^t \end{aligned}$$



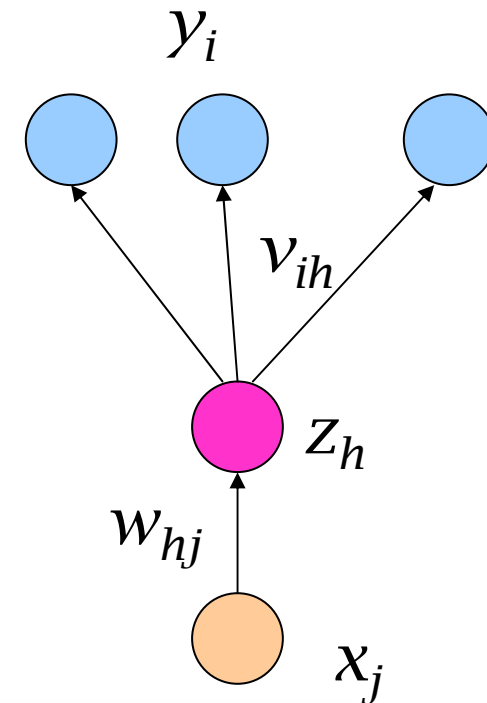
# Regression with Multiple Outputs

$$E(\mathbf{W}, \mathbf{V} | \mathcal{X}) = \frac{1}{2} \sum_t \sum_i (r_i^t - y_i^t)^2$$

$$y_i^t = \sum_{h=1}^H v_{ih} z_h^t + v_{i0}$$

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left[ \sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t$$





Initialize all  $v_{ih}$  and  $w_{hj}$  to  $\text{rand}(-0.01, 0.01)$

Repeat

For all  $(\mathbf{x}^t, r^t) \in \mathcal{X}$  in random order

For  $h = 1, \dots, H$

$$z_h \leftarrow \text{sigmoid}(\mathbf{w}_h^T \mathbf{x}^t)$$

For  $i = 1, \dots, K$

$$y_i = \mathbf{v}_i^T \mathbf{z}$$

For  $i = 1, \dots, K$

$$\Delta \mathbf{v}_i = \eta(r_i^t - y_i^t) \mathbf{z}$$

For  $h = 1, \dots, H$

$$\Delta \mathbf{w}_h = \eta\left(\sum_i (r_i^t - y_i^t) v_{ih}\right) z_h (1 - z_h) \mathbf{x}^t$$

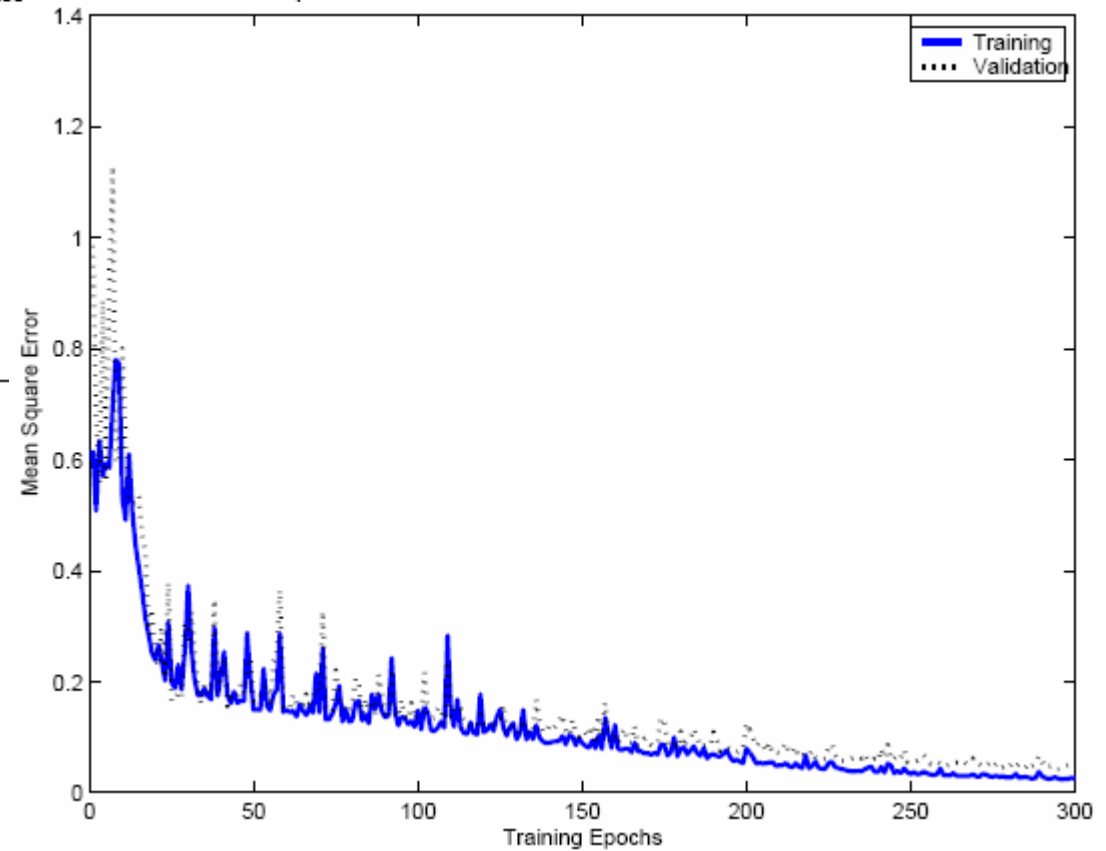
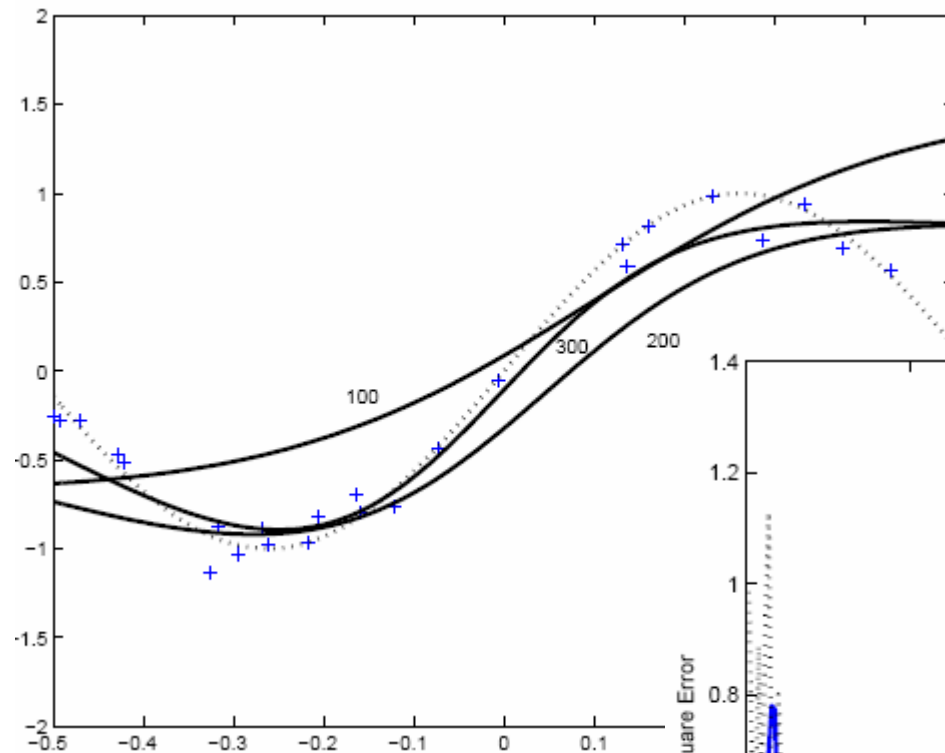
For  $i = 1, \dots, K$

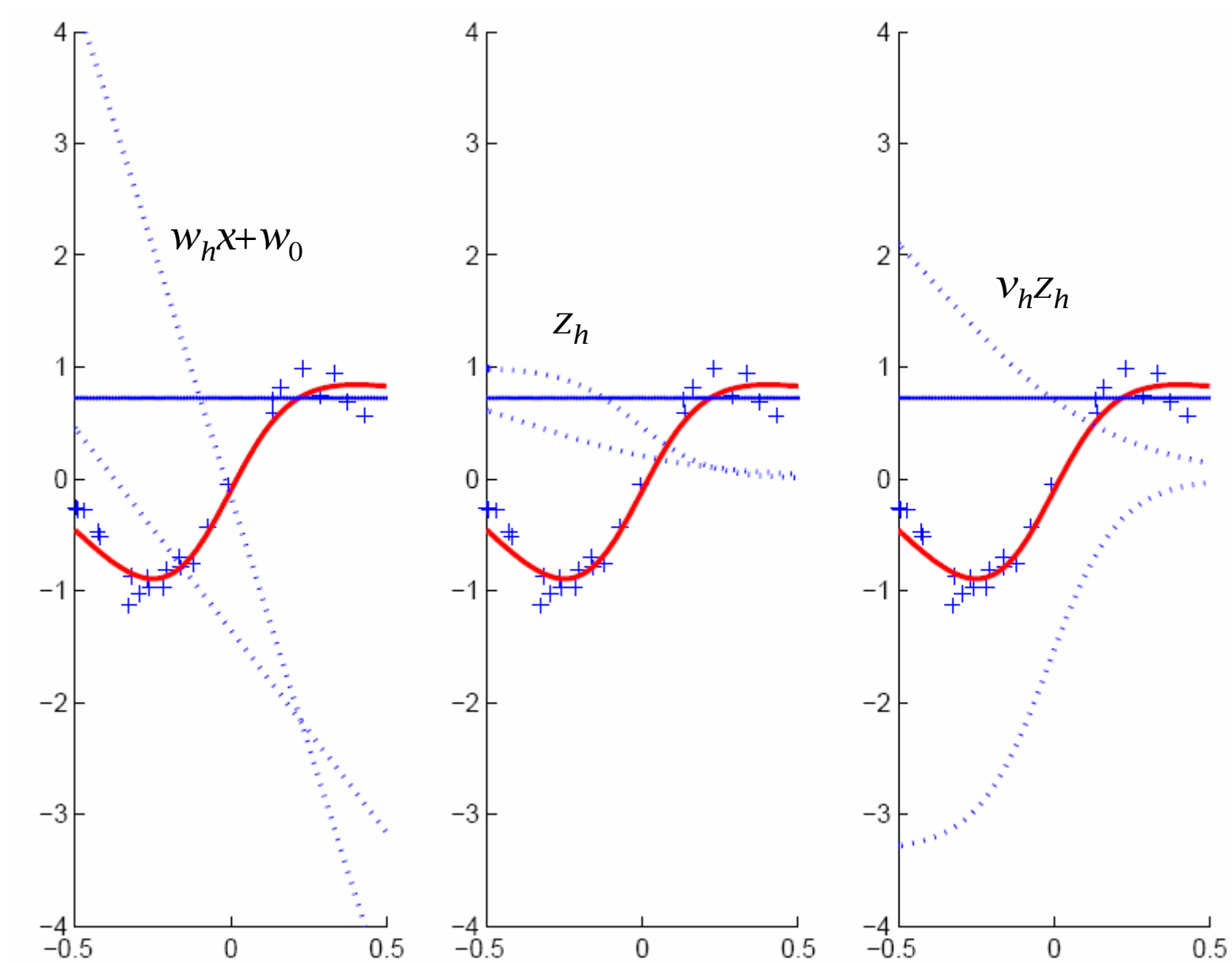
$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta \mathbf{v}_i$$

For  $h = 1, \dots, H$

$$\mathbf{w}_h \leftarrow \mathbf{w}_h + \Delta \mathbf{w}_h$$

Until convergence







## *Two-Class Discrimination*

- One sigmoid output for  $P(C_1|\mathbf{x}^t)$  and  $\hat{P}(C_2|\mathbf{x}^t) \equiv 1 - y^t$

$$y^t = \text{sigmoid} \left( \sum_{h=1}^H v_h z_h^t + v_0 \right)$$

$$E(\mathbf{W}, \mathbf{v}|\mathcal{X}) = - \sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t)$$

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

## *K > 2 Classes*

$$o_i^t = \sum_{h=1}^H v_{ih} z_h^t + v_{i0}$$

$$y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t}$$

approximates  $P(C_i | \mathbf{x}^t)$

$$E(\mathbf{W}, \mathbf{V} | \mathcal{X}) = - \sum_t \sum_i r_i^t \log y_i^t$$

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left[ \sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t$$



# Multiple Hidden Layers

- MLP with one hidden layer is a **universal approximator** (Hornik et al., 1989), but using multiple layers may lead to simpler networks

$$z_{1h} = \text{sigmoid}(\mathbf{w}_{1h}^T \mathbf{x}) = \sum_{j=1}^d w_{1hj} x_j + w_{1h0}, \quad h = 1, \dots, H_1$$

$$z_{2l} = \text{sigmoid}(\mathbf{w}_{2l}^T \mathbf{z}_1) = \sum_{h=0}^{H_1} w_{2lh} z_{1h} + w_{2l0}, \quad l = 1, \dots, H_2$$

$$y = \mathbf{v}^T \mathbf{z}_2 = \sum_{l=1}^{H_2} v_l z_{2l} + v_0$$



# *Improving Convergence*

- Momentum

$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1}$$

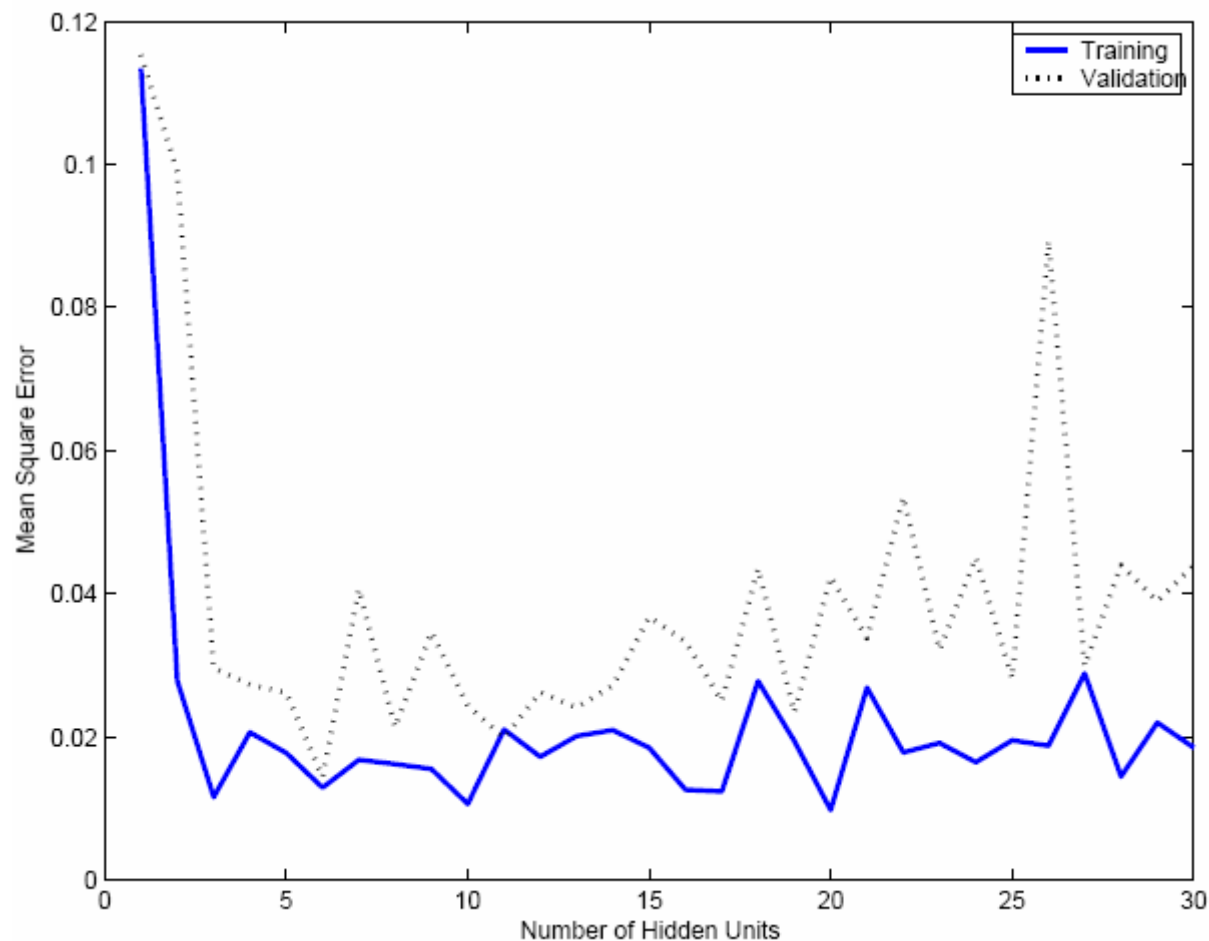
- Adaptive learning rate

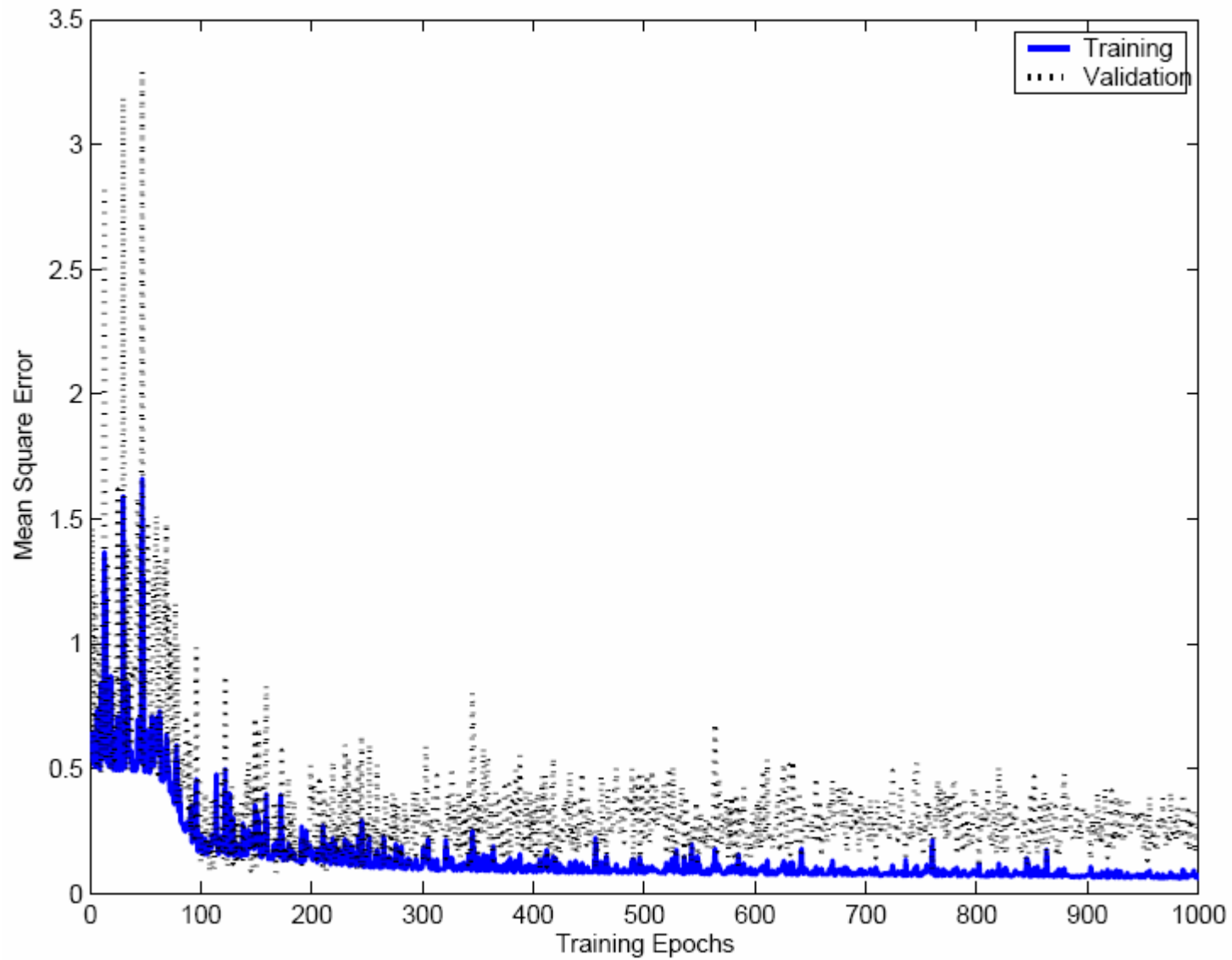
$$\Delta \eta = \begin{cases} +a & \text{if } E^{t+\tau} < E^t \\ -b\eta & \text{otherwise} \end{cases}$$



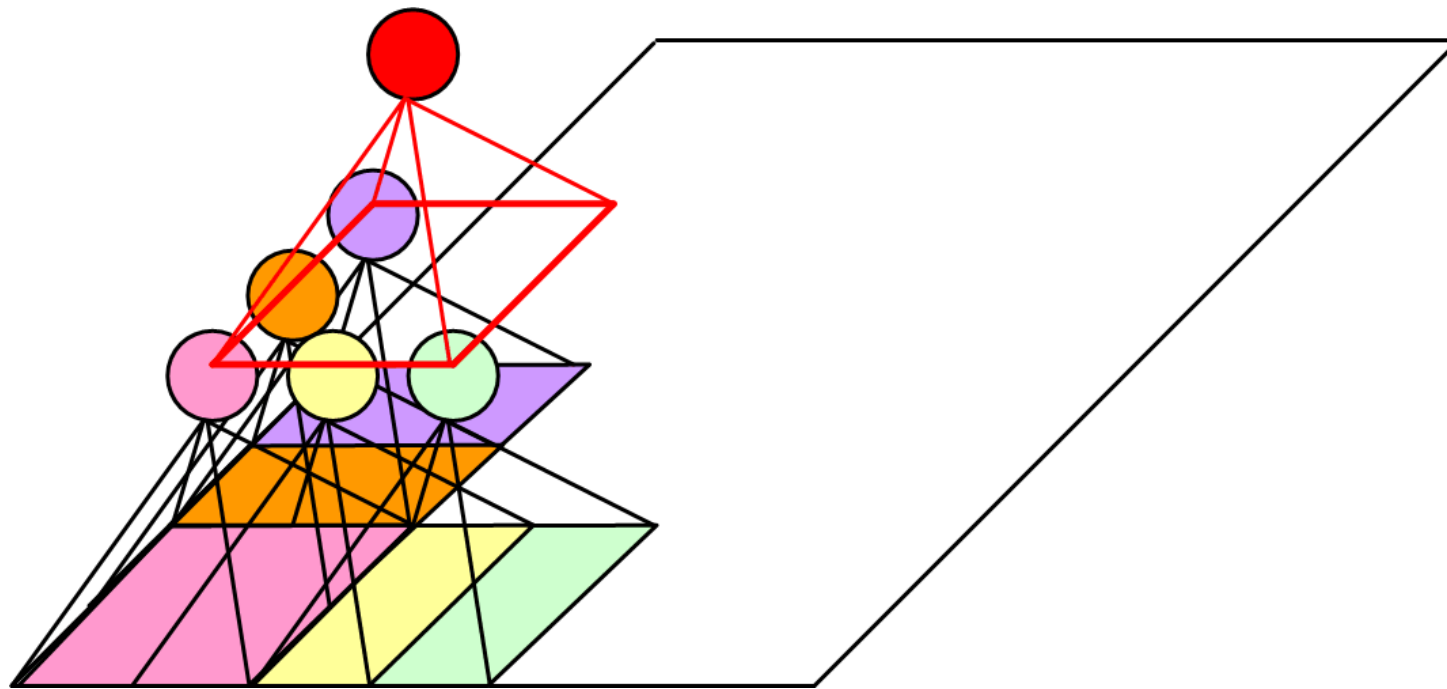
# Overfitting/Overtraining

Number of weights:  $H(d+1) + (H+1)*K$



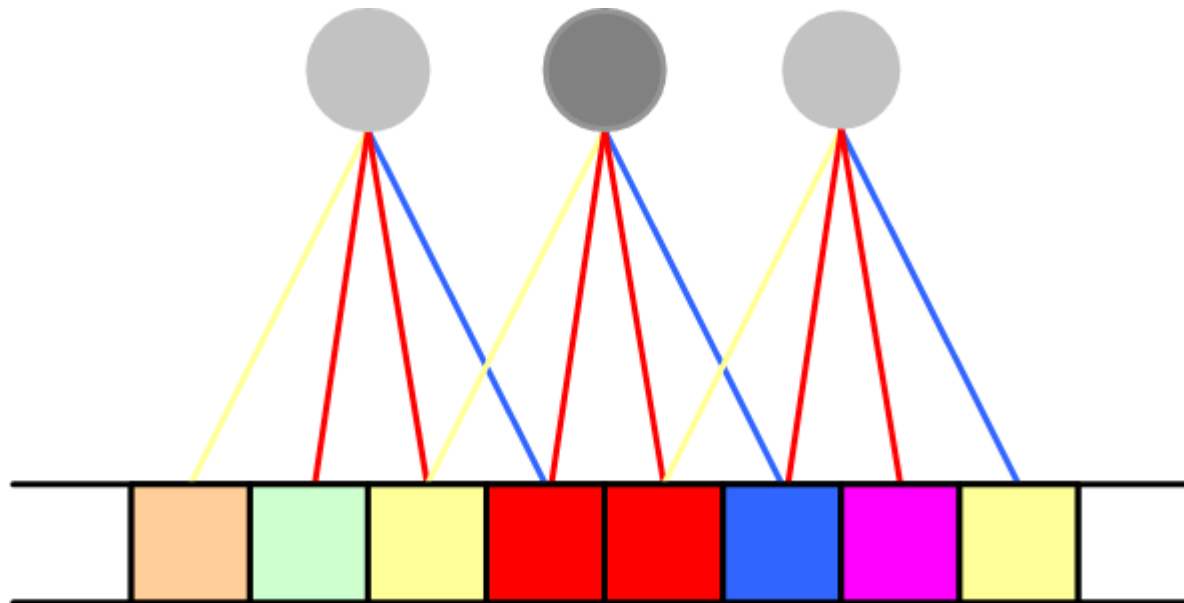


# *Structured MLP*



(Le Cun et al, 1989)

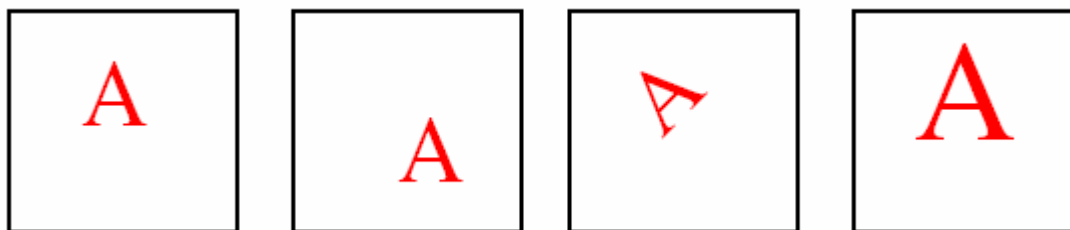
# Weight Sharing



# Hints

(Abu-Mostafa, 1995)

- Invariance to translation, rotation, size



- Virtual examples
- Augmented error:  $E' = E + \lambda_h \cdot E_h$

If  $\mathbf{x}'$  and  $\mathbf{x}$  are the “same”:  $E_h = [g(\mathbf{x}|\theta) - g(\mathbf{x}'|\theta)]^2$

Approximation hint:

$$E_h = \begin{cases} 0 & \text{if } g(\mathbf{x}|\theta) \in [a_x, b_x] \\ (g(\mathbf{x}) - a_x)^2 & \text{if } g(\mathbf{x}|\theta) < a_x \\ (g(\mathbf{x}) - b_x)^2 & \text{if } g(\mathbf{x}|\theta) > b_x \end{cases}$$

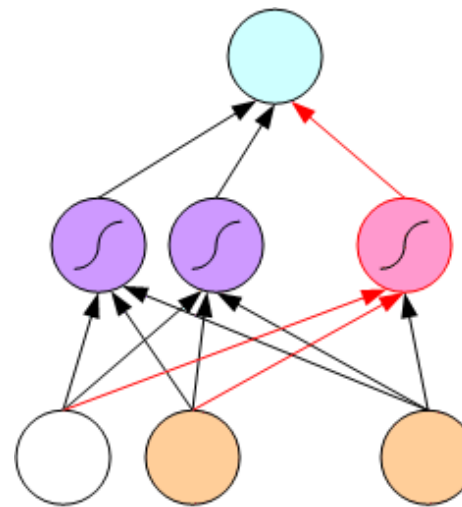
# Tuning the Network Size

- Destructive
- Weight decay:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda w_i$$

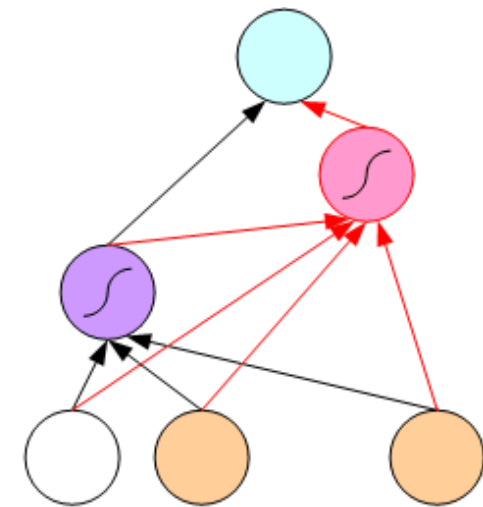
$$E' = E + \frac{\lambda}{2} \sum_i w_i^2$$

- Constructive
- Growing networks



*Dynamic Node Creation*

(Ash, 1989)



*Cascade Correlation*

(Fahlman and Lebiere, 1989)

# Bayesian Learning

- Consider weights  $w_i$  as random vars, prior  $p(w_i)$

$$p(\mathbf{w}|\mathcal{X}) = \frac{p(\mathcal{X}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{X})} \quad \hat{\mathbf{w}}_{MAP} = \arg \max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{X})$$

$$\log p(\mathbf{w}|\mathcal{X}) = \log p(\mathcal{X}|\mathbf{w}) + \log p(\mathbf{w}) + C$$

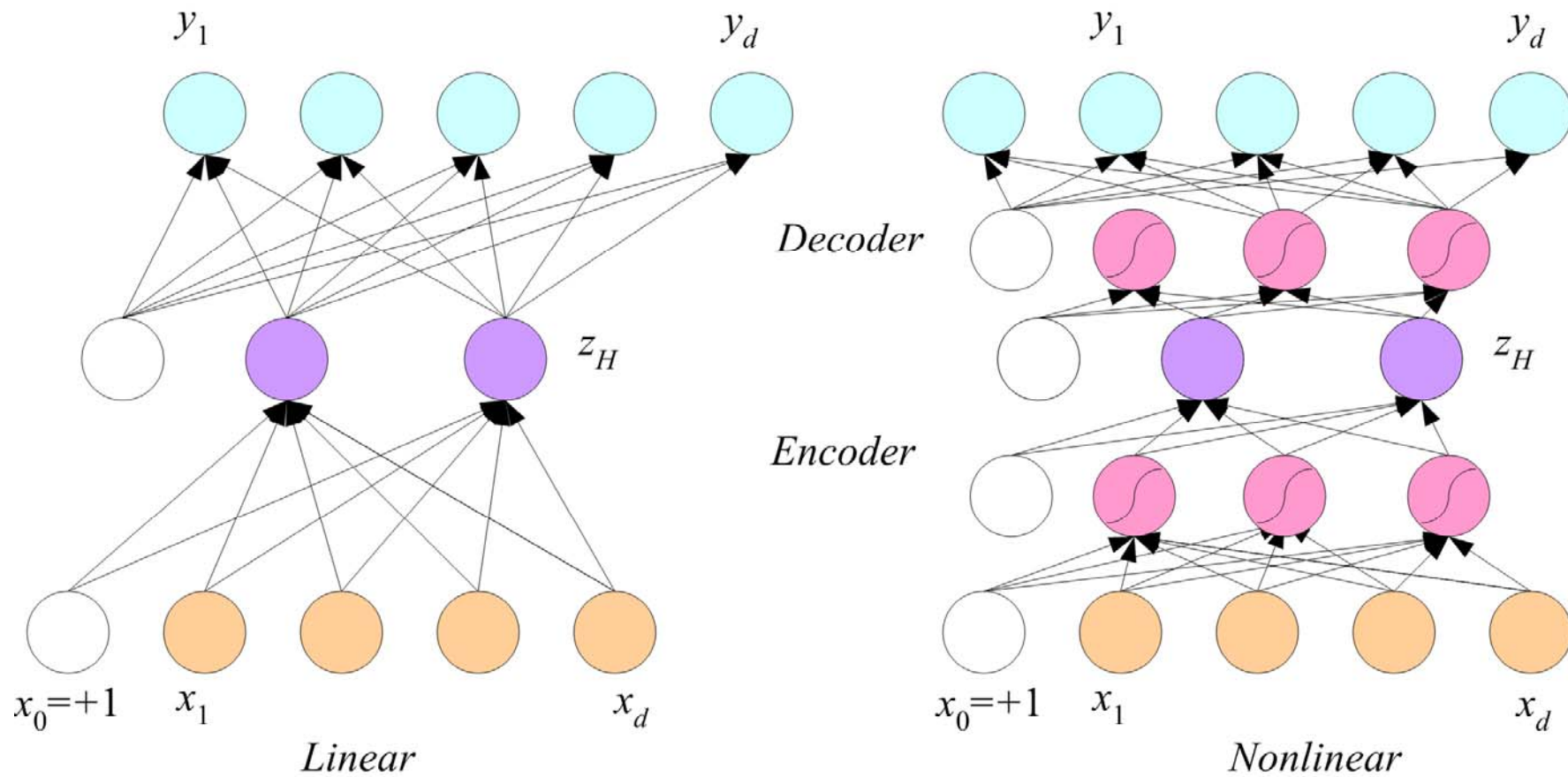
$$p(\mathbf{w}) = \prod_i p(w_i) \text{ where } p(w_i) = c \cdot \exp \left[ -\frac{w_i^2}{2(1/2\lambda)} \right]$$

$$E' = E + \lambda \|\mathbf{w}\|^2$$

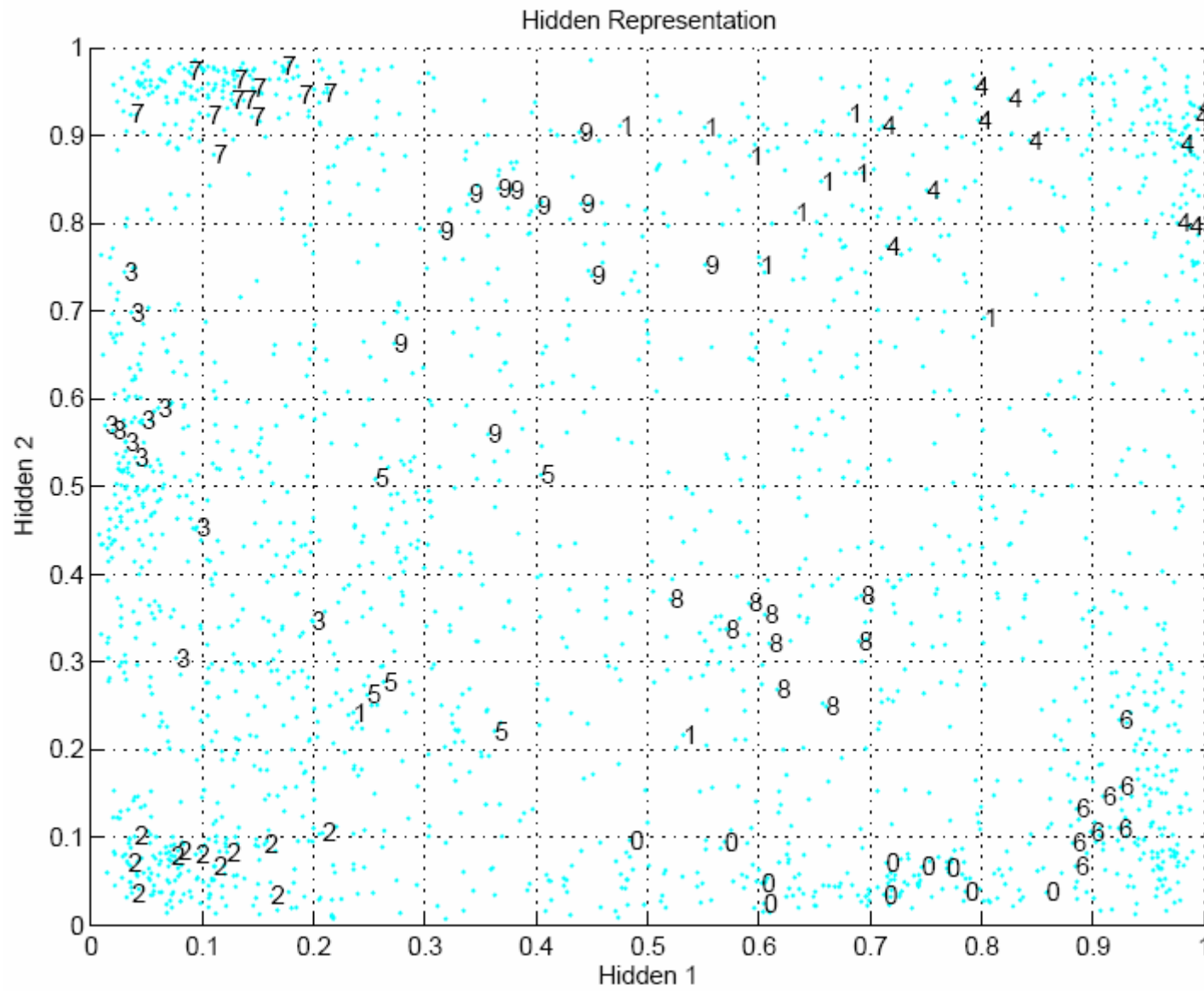
- Weight decay, ridge regression, regularization

$$\text{cost} = \text{data-misfit} + \lambda \cdot \text{complexity}$$

# Dimensionality Reduction





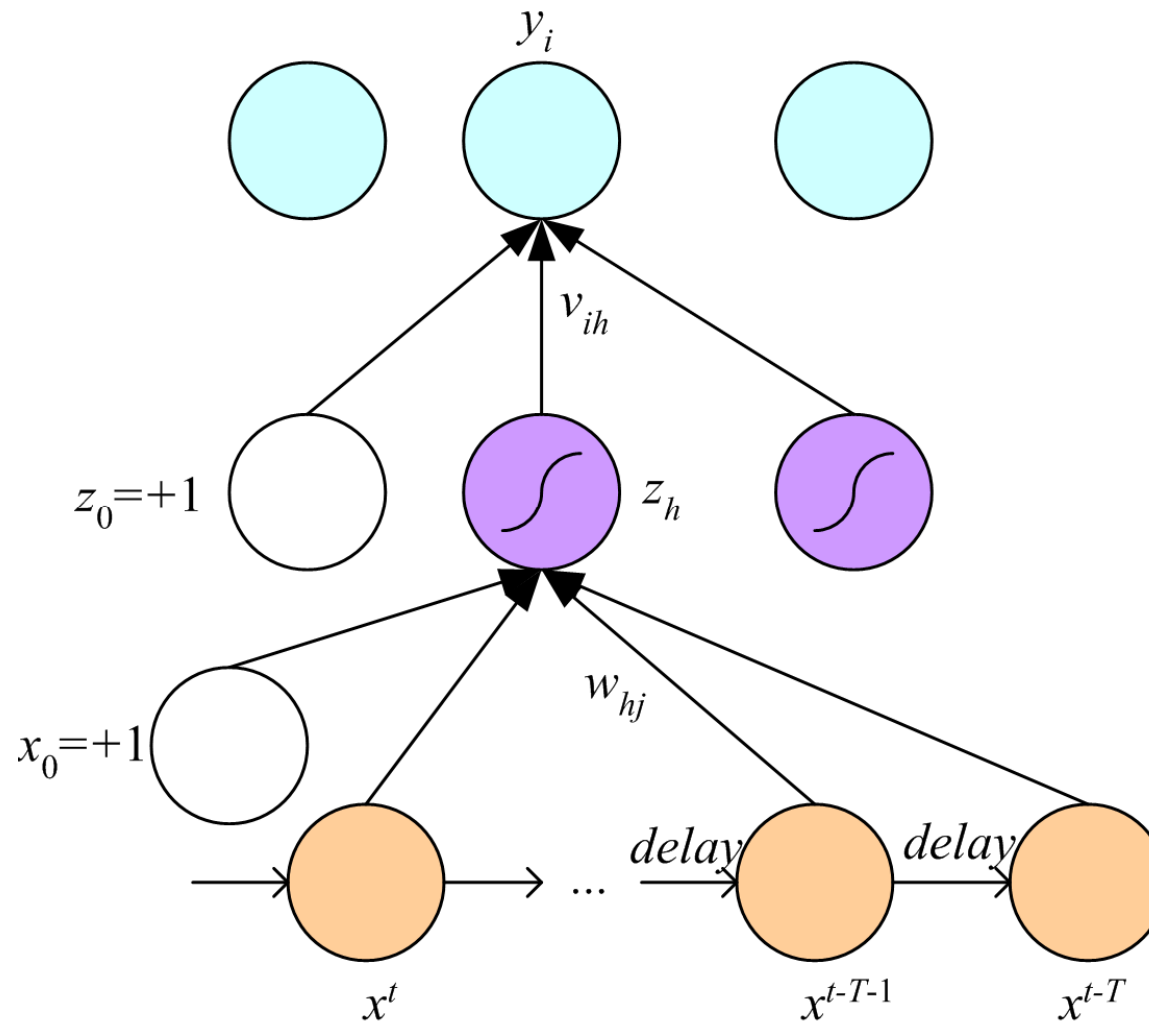




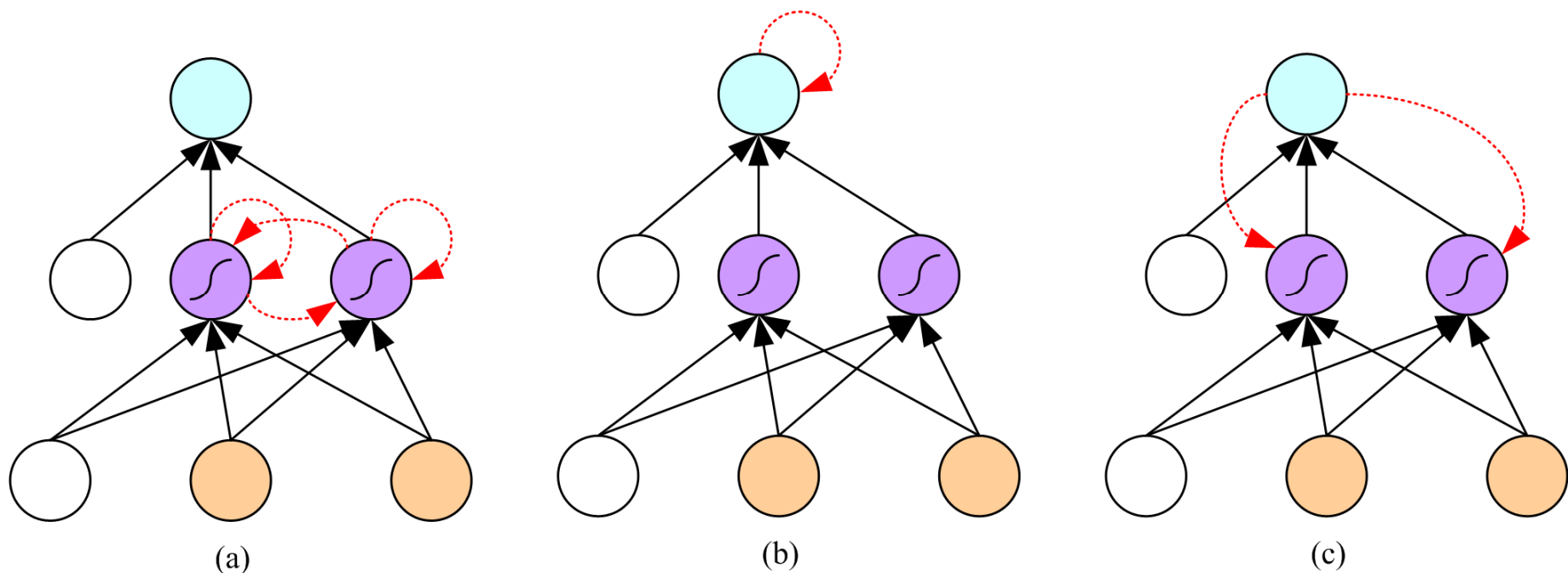
# *Learning Time*

- Applications:
  - Sequence recognition: Speech recognition
  - Sequence reproduction: Time-series prediction
  - Sequence association
- Network architectures
  - Time-delay networks (Waibel et al., 1989)
  - Recurrent networks (Rumelhart et al., 1986)

# Time-Delay Neural Networks



# Recurrent Networks



# Unfolding in Time

