Artificial Intelligence
Chapter 3
Neural Networks

Biointelligence Lab
School of Computer Sci. & Eng.
Seoul National University

Outline

- 3.1 Introduction
- 3.2 Training Single TLUs
  - Gradient Descent
  - Widrow-Hoff Rule
  - Generalized Delta Procedure
- 3.3 Neural Networks
  - The Backpropagation Method
  - Derivation of the Backpropagation Learning Rule
- 3.4 Generalization, Accuracy, and Overfitting
- 3.5 Discussion

3.1 Introduction

- TLU (threshold logic unit): Basic units for neural networks
  - Based on some properties of biological neurons
- Training set
  - Input: real value, boolean value, …

$$\mathrm{X}: \text{ n-dim vector}, \mathrm{X} = (x_1, \ldots, x_n)$$

  - Output:
    - d_i: associated actions (Label, Class …)
- Target of training
  - Finding $f(\mathbf{X})$ corresponds "acceptably" to the members of the training set.
  - Supervised learning: Labels are given along with the input vectors.

3.2 Training Single TLUs
3.2.1 TLU Geometry

- Training TLU: Adjusting variable weights
- A single TLU: Perceptron, Adaline (*ada*ptive *lin*ear *e*lement) [Rosenblatt 1962, Widrow 1962]
- Elements of TLU
  - Weight: $\mathbf{W} = (w_1, \ldots, w_n)$
  - Threshold: θ
- Output of TLU: Using weighted sum $s = \mathbf{W} \cdot \mathbf{X}$
  - 1 if $\mathbf{s} - \theta > 0$
  - 0 if $\mathbf{s} - \theta < 0$
- Hyperplane
  - $\mathbf{W} \cdot \mathbf{X} - \theta = 0$

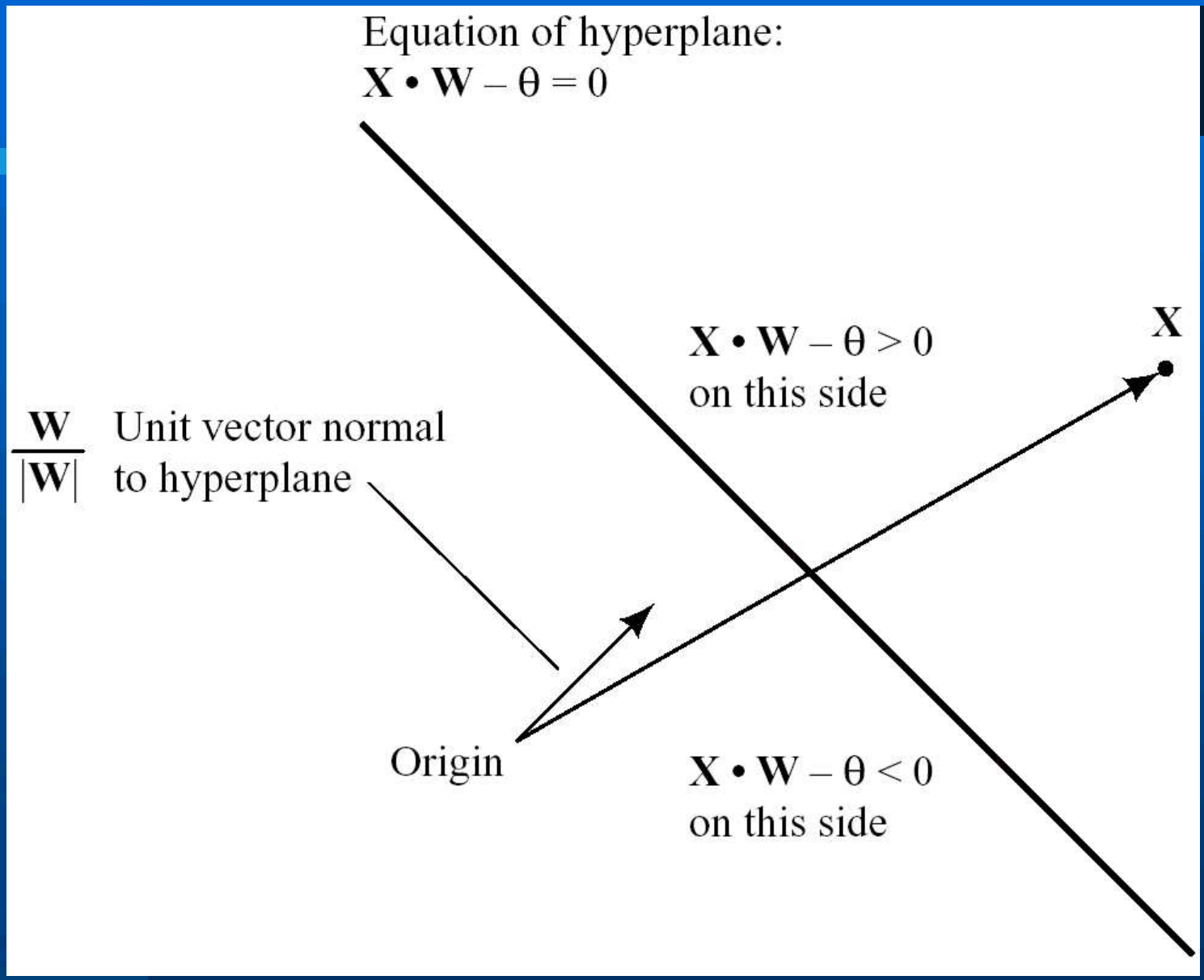

Figure 3.1 TLU Geometry

3.2.2 Augmented Vectors

- Adopting the convention that threshold is fixed to 0.
- Arbitrary thresholds: $(n + 1)$-dimensional vector
- $\mathbf{W} = (w_1, \ldots, w_n, -\theta)$, $\mathbf{X} = (x_1, \ldots, x_n, 1)$
- Output of TLU
  - 1 if $\mathbf{W} \cdot \mathbf{X} \geq 0$
  - 0 if $\mathbf{W} \cdot \mathbf{X} < 0$

3.2.3 Gradient Decent Methods

- Training TLU: minimizing the *error function* by adjusting weight values.
- Batch learning v.s. incremental learning
- Commonly used error function: squared error

$$\varepsilon = (d - f)^2$$

  - Gradient: $\frac{\partial \varepsilon}{\partial \mathbf{W}} \overset{def}{=} \left[\frac{\partial \varepsilon}{\partial w_1}, \ldots, \frac{\partial \varepsilon}{\partial w_i}, \ldots, \frac{\partial \varepsilon}{\partial w_{n+1}} \right]$
  - Chain rule: $\frac{\partial \varepsilon}{\partial \mathbf{W}} = \frac{\partial \varepsilon}{\partial s} \frac{\partial s}{\partial \mathbf{W}} = \frac{\partial \varepsilon}{\partial s} \mathbf{X} = -2(d - f) \frac{\partial f}{\partial s} \mathbf{X}$

- Solution of nonlinearity of $\partial f / \partial s$:
  - Ignoring threshold function: $f = s$
  - Replacing threshold function with differentiable nonlinear function

3.2.4 The Widrow-Hoff Procedure: First Solution

- Weight update procedure:
  - Using $f = s = \mathbf{W} \cdot \mathbf{X}$
  - Data labeled $1 \rightarrow 1$, Data labeled $0 \rightarrow -1$
- Gradient:

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d - f)\frac{\partial f}{\partial s}\mathbf{X} = -2(d - f)\mathbf{X}$$

- New weight vector

$$\mathbf{W} \leftarrow \mathbf{W} + c(d - f)\mathbf{X}$$

- Widrow-Hoff (delta) rule
  - $(d - f) > 0 \rightarrow$ increasing $s \rightarrow$ decreasing $(d - f)$
  - $(d - f) < 0 \rightarrow$ decreasing $s \rightarrow$ increasing $(d - f)$

3.2.5 The Generalized Delta Procedure: Second Solution

- Sigmoid function (**differentiable**): [Rumelhart, et al. 1986]

$$f(s) = 1/(1+e^{-s})$$

- Gradient:

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d-f)\frac{\partial f}{\partial s}\mathbf{X} = -2(d-f)f(1-f)\mathbf{X}$$

- Generalized delta procedure:

$$\mathbf{W} \leftarrow \mathbf{W} + c(d-f)f(1-f)\mathbf{X}$$

  - Target output: 1, 0
  - Output f = output of sigmoid function
  - $f(1-f) = 0$, where $f = 0$ or 1
  - Weight change can occur only within 'fuzzy' region surrounding the hyperplane (near the point $f(s) = ½$).

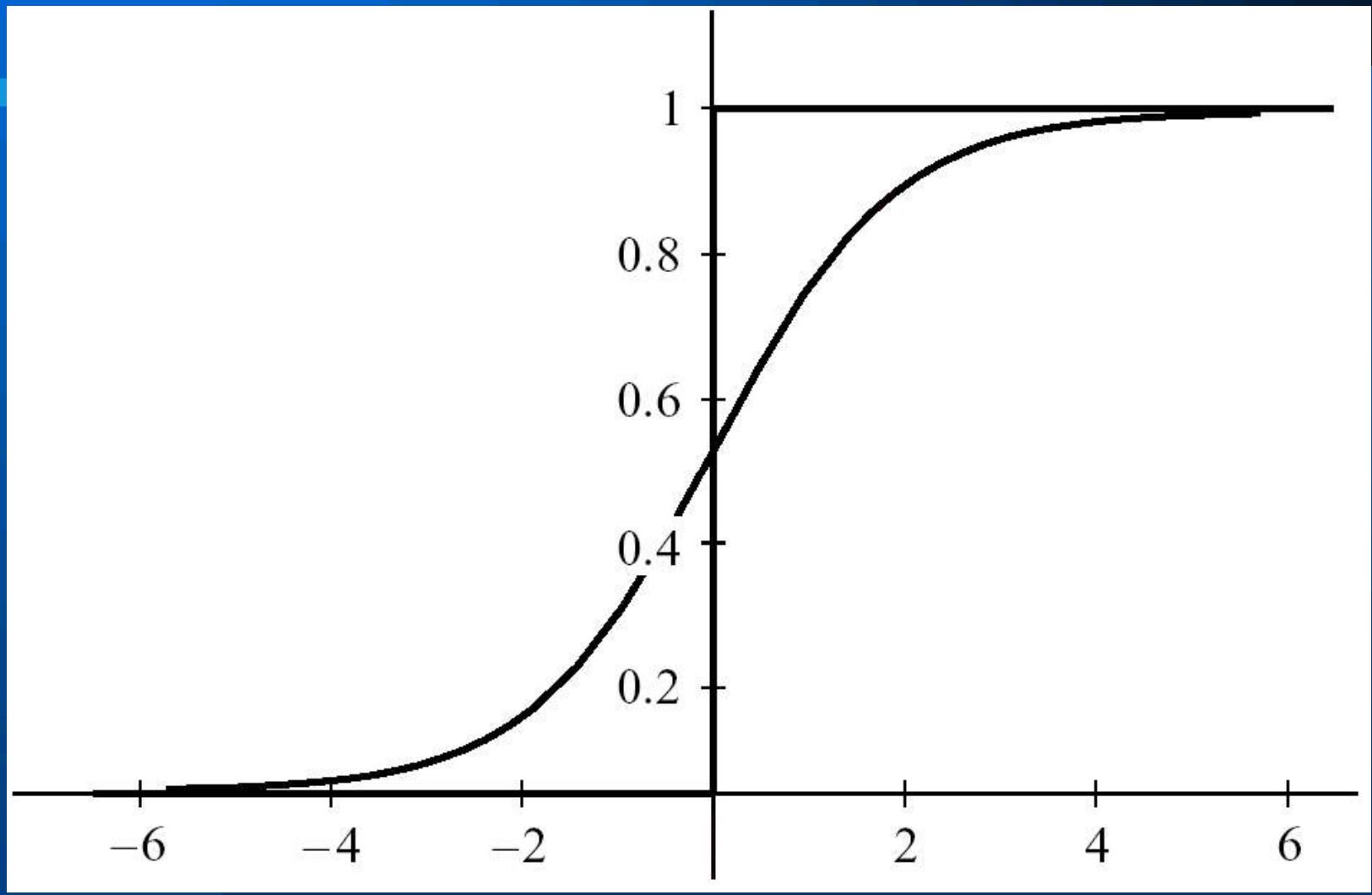

Figure 3.2 A Sigmoid Function

3.2.6 The Error-Correction Procedure

- Using threshold unit: $(d - f)$ can be either 1 or –1.

$$\mathbf{W} \leftarrow \mathbf{W} + c(d - f)\mathbf{X}$$

- In the linearly separable case, after finite iteration, **W** will be converged to the solution.
- In the nonlinearly separable case, **W** will never be converged.
- The Widrow-Hoff and generalized delta procedures will find minimum squared error solutions even when the minimum error is not zero.

3.3 Neural Networks
3.3.1 Motivation

- Need for use of multiple TLUs
  - Feedforward network: no cycle
  - Recurrent network: cycle (treated in a later chapter)
  - Layered feedforward network
    - j_{th} layer can receive input only from $j - 1_{th}$ layer.
- Example : $f = x_1 x_2 + \bar{x}_1 \bar{x}_2$

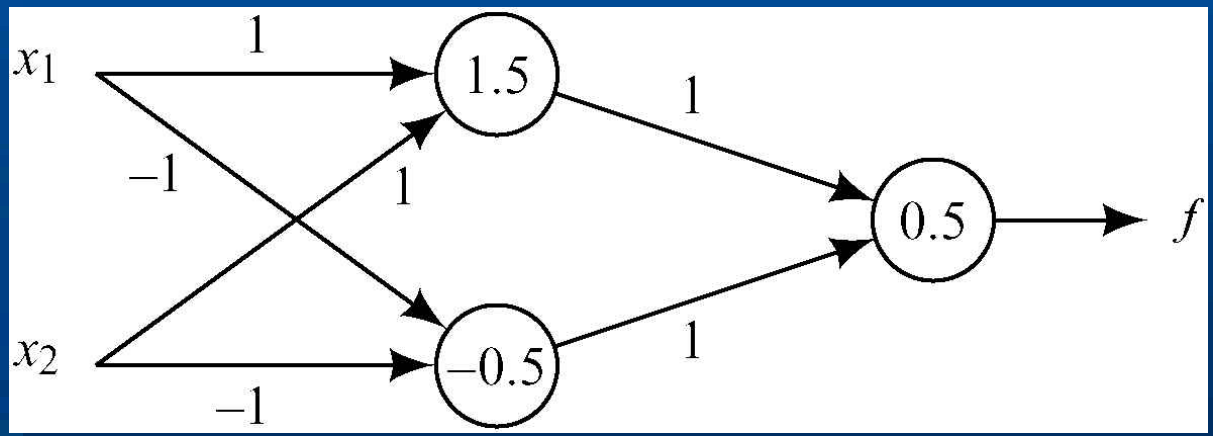

Figure 3.4 A Network of TLUs That Implements the Even-Parity Function

3.3.2 Notation

- Hidden unit: neurons in all but the last layer
- Output of j-th layer: $\mathbf{X}^{(j)} \rightarrow$ input of $(j+1)$-th layer
- Input vector: $\mathbf{X}^{(0)}$
- Final output: f
- The weight of i-th sigmoid unit in the j-th layer: $\mathbf{W}_i^{(j)}$
- Weighted sum of i-th sigmoid unit in the j-th layer: $s_i^{(j)}$

$$s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$$

- Number of sigmoid units in j-th layer: m_j

$$\mathbf{W}_i^{(j)} = (w_{1,i}^{(j)}, \ldots, w_{l,i}^{(j)}, \ldots, w_{m_{(j-1)}+1,i}^{(j)})$$

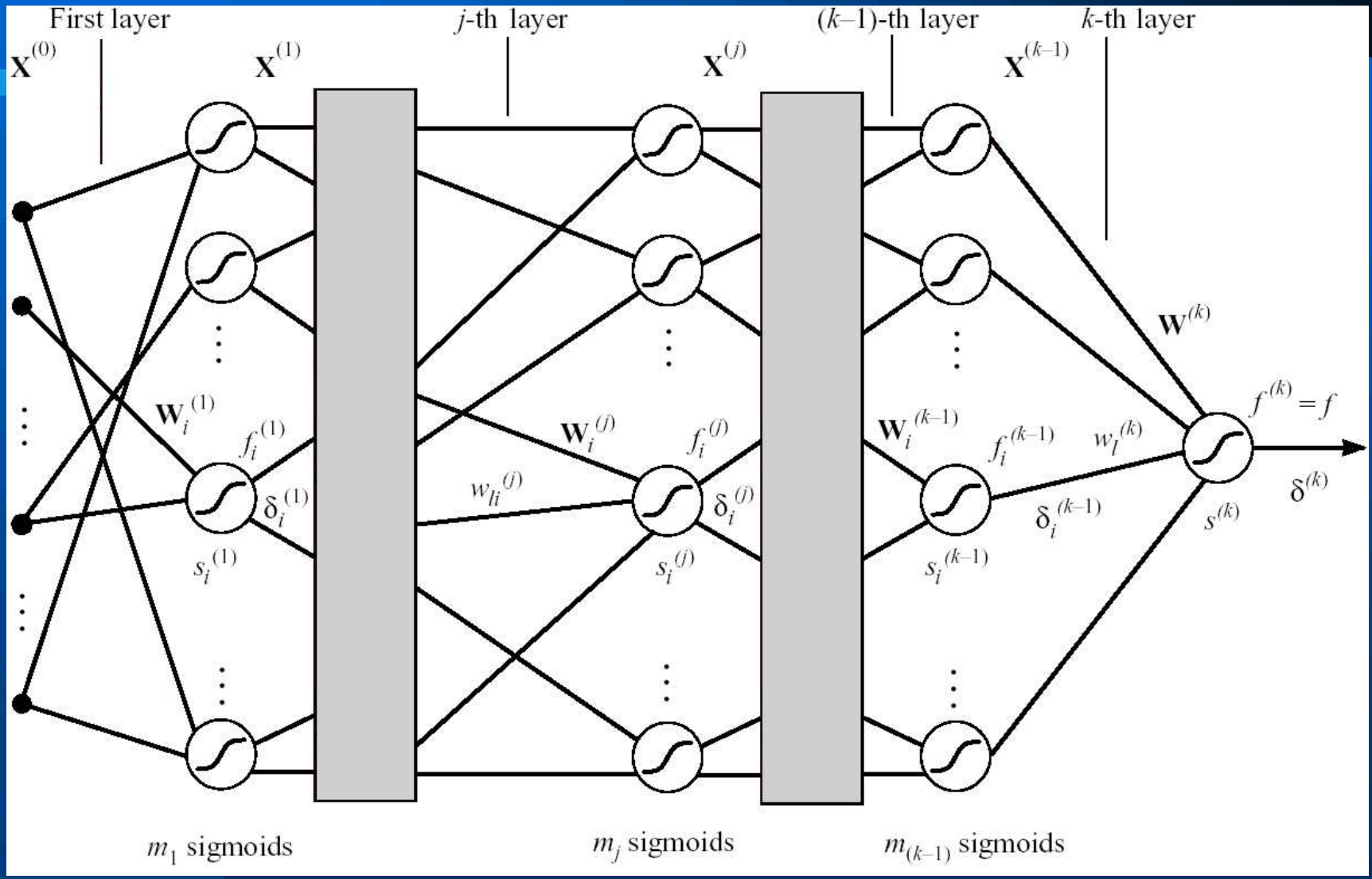

Figure 3.4 A *k*-layer Network of Sigmoid Units

3.3.3 The Backpropagation Method

- Gradient of $\mathbf{W}_i^{(j)}$:

$$s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$$

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \frac{\partial s_i^{(j)}}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)}$$

$$= -2(d-f)\frac{\partial f}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)} = -2\delta_i^{(j)} \mathbf{X}^{(j-1)}$$

Local gradient

$$\frac{\partial \varepsilon}{\partial s_i^{(j)}} = \frac{\partial (d-f)^2}{\partial s_i^{(j)}} = -2(d-f)\frac{\partial f}{\partial s_i^{(j)}}$$

- Weight update:

$$\mathbf{W}_i^{(j)} \leftarrow \mathbf{W}_i^{(j)} + c_i^{(j)} \delta_i^{(j)} \mathbf{X}^{(j-1)}$$

3.3.4 Computing Weight Changes in the Final Layer

- Local gradient:

$$\delta^{(k)} = (d - f)\frac{\partial f}{\partial s_i^{(j)}}$$
$$= (d - f) f (1 - f)$$

- Weight update:

$$\mathbf{W}^{(k)} \leftarrow \mathbf{W}^{(k)} + c^{(k)}(d - f) f (1 - f)\mathbf{X}^{(k-1)}$$

3.3.5 Computing Changes to the Weights in Intermediate Layers

- Local gradient:

$$\delta_i^{(j)} = (d-f)\frac{\partial f}{\partial s_i^{(j)}}$$

$$= (d-f)\left[\frac{\partial f}{\partial s_1^{(j+1)}}\frac{\partial s_1^{(j+1)}}{\partial s_i^{(j)}} + \ldots \frac{\partial f}{\partial s_l^{(j+1)}}\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} + \ldots + \frac{\partial f}{\partial s_{m_{j+1}}^{(j+1)}}\frac{\partial s_{m_{j+1}}^{(j+1)}}{\partial s_i^{(j)}}\right]$$

$$= \sum_{l=1}^{m_{j+1}} (d-f)\frac{\partial f}{\partial s_l^{(j+1)}}\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)}\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}}$$

  - The final ouput f, depends on $s_i^{(j)}$ through of the summed inputs to the sigmoids in the $(j+1)$-th layer.

- Need for computation of $\dfrac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}}$

Weight Update in Hidden Layers (cont.)

$$s_l^{(j+1)} = \mathbf{X}^{(j)} \cdot \mathbf{W}_l^{(j+1)} = \sum_{v=1}^{m_j+1} f_v^{(j)} w_{vl}^{(j+1)}$$

$$\frac{\partial s_l^{(j+1)}}{\partial s_i^{(j)}} = \frac{\partial \left[\sum_{v=1}^{m_j+1} f_v^{(j)} w_{vl}^{(j+1)} \right]}{\partial s_i^{(j)}} = \sum_{v=1}^{m_j+1} w_{vl}^{(j+1)} \frac{\partial f_v^{(j)}}{\partial s_i^{(j)}} I(i=v)$$

$$= w_{il}^{(j+1)} f_i^{(j)} (1 - f_i^{(j)})$$

- $v \neq i$: $\dfrac{\partial f_v^{(j)}}{\partial s_i^{(j)}} = 0$

$v = i$: $\dfrac{\partial f_v^{(j)}}{\partial s_i^{(j)}} = f_v^{(j)} (1 - f_v^{(j)})$

- Conseqeuntly, $\delta_i^{(j)} = f_i^{(j)} (1 - f_i^{(j)}) \sum_{l=1}^{m_{j+1}} \delta_l^{(j+1)} w_{il}^{(j+1)}$

Weight Update in Hidden Layers (cont.)

- Attention to **recursive equation** of local gradient!

$$\delta^{(k)} = f(1-f)(d-f)$$

$$\delta_i^{(j)} = f_i^{(j)}(1-f_i^{(j)})\sum_{l=1}^{m_{j+1}} \delta_i^{(j+1)} w_{il}^{(j+1)}$$

- Backpropagation:
  - Error is back-propagated from the output layer to the input layer
  - Local gradient of the latter layer is used in calculating local gradient of the former layer.

3.3.5 (con't)

- Example (even parity function)

input			target
1	0	1	0
0	0	1	1
0	1	1	0
1	1	1	1

  - Learning rate: 1.0

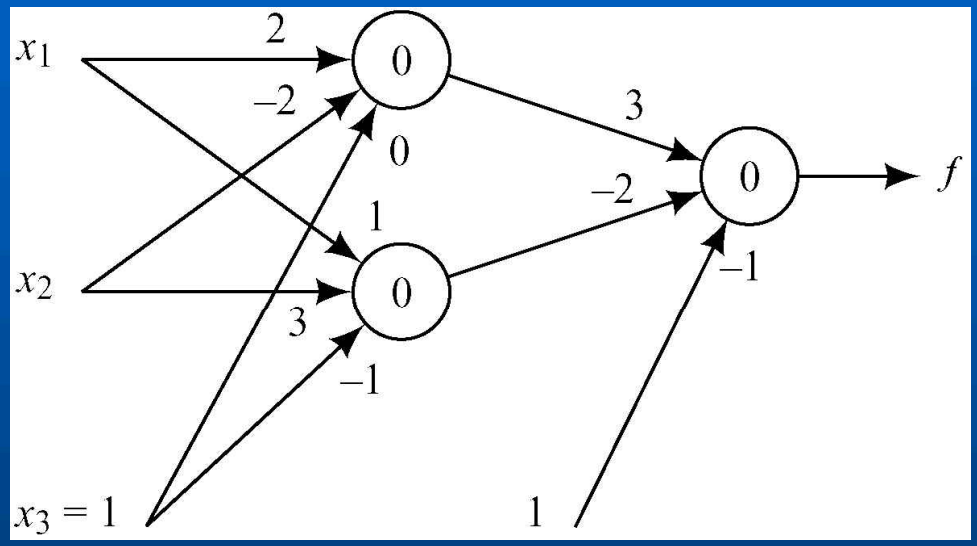

Figure 3.6 A Network to Be Trained by Backprop

3.4 Generalization, Accuracy, and Overfitting

- Generalization ability:
  - NN appropriately classifies vectors not in the training set.
  - Measurement = accuracy
- Curve fitting
  - Number of training input vectors ≥ number of degrees of freedom of the network.
  - In the case of *m* data points, is (*m*-1)-degree polynomial best model? **No, it can not capture any special information.**
- Overfitting
  - Extra degrees of freedom are essentially just fitting the noise.
  - Given sufficient data, the *Occam's Razor* principle dictates to choose the lowest-degree polynomial that adequately fits the data.

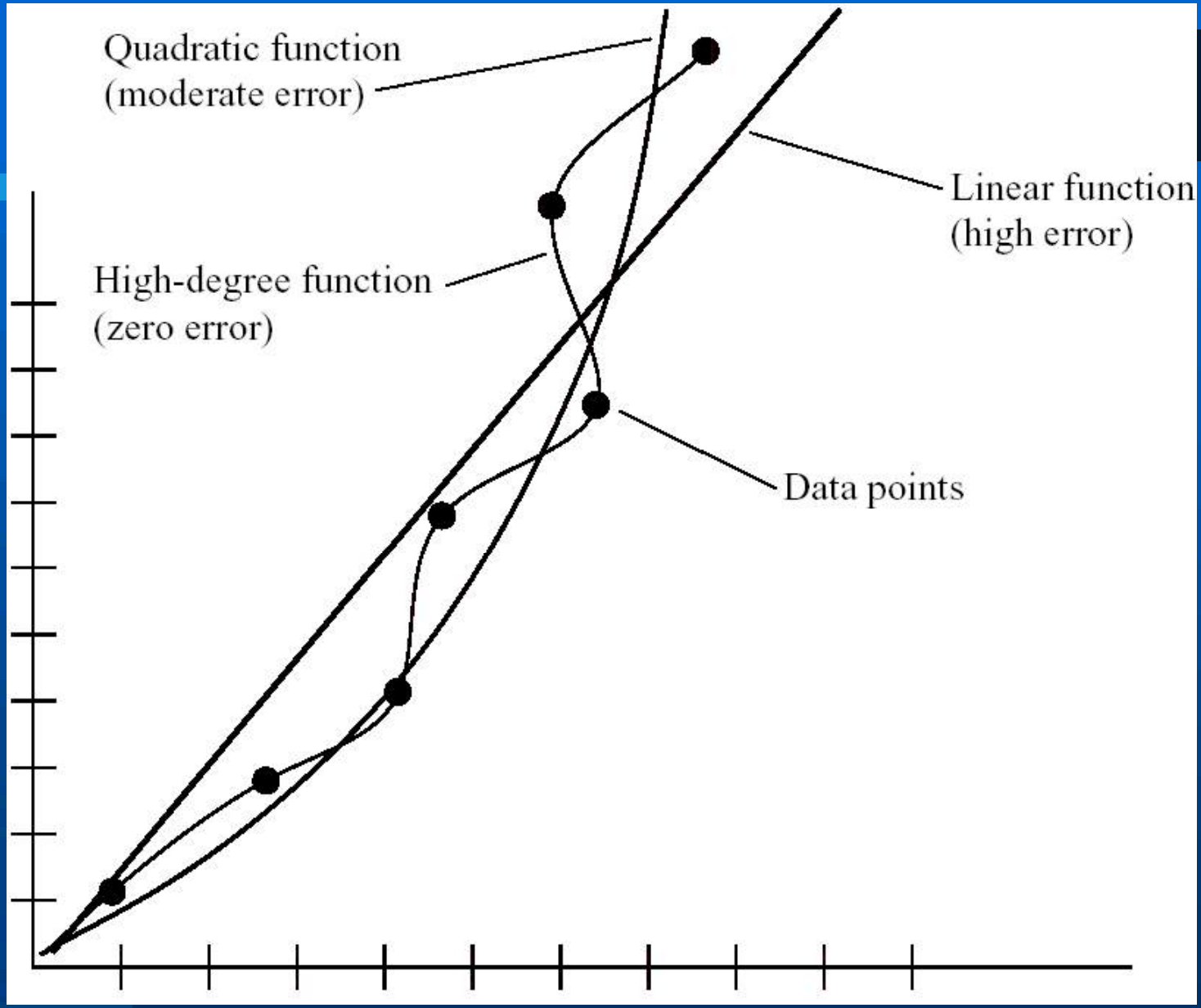

Figure 3.7 Curve Fitting

3.4 (cont'd)

- Out-of-sample-set error rate
  - Error rate on data drawn from the same underlying distribution of training set.
- Dividing available data into a t*raining* set and a *validation* set
  - Usually use 2/3 for training and 1/3 for validation
- *k*-fold cross validation
  - *k* disjoint subsets (called folds).
  - Repeat training *k* times with the configuration: one validation set, *k*-1 (combined) training sets.
  - Take average of the error rate of each validation as the out-of-sample error.
  - Empirically 10-fold is preferred.

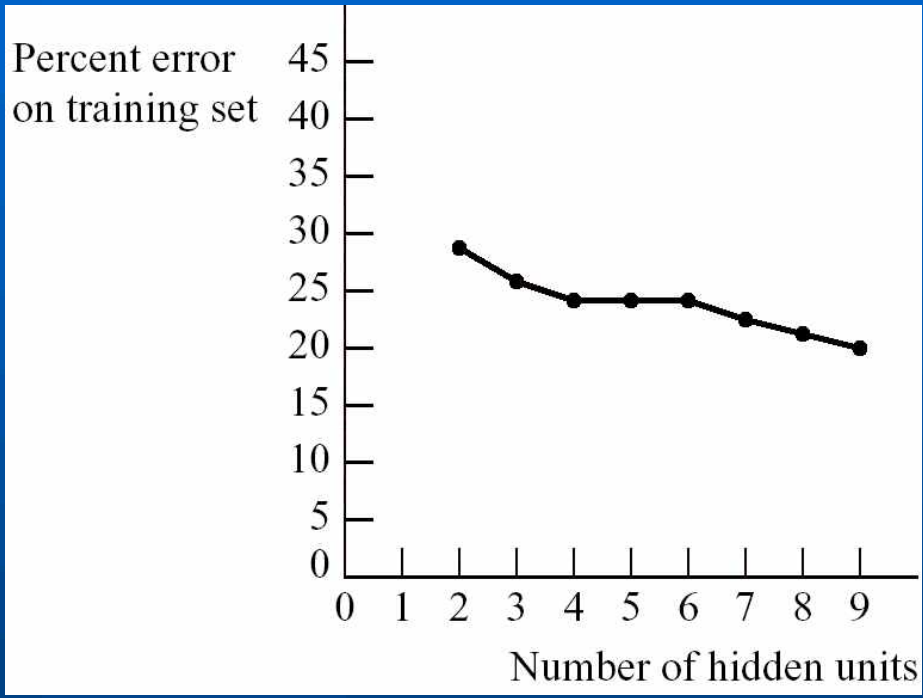

Figure 3.8 Error Versus Number of Hidden Units

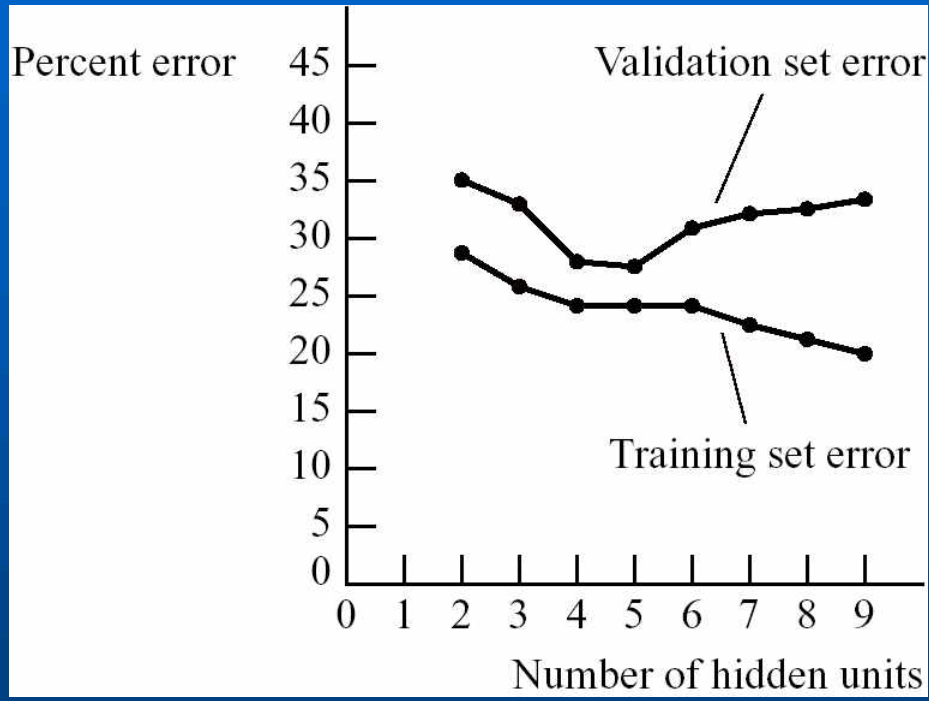

Fig 3.9 Estimate of Generalization Error Versus Number of Hidden Units

3.5 Additional Readings and Discussion

- Applications
  - Pattern recognition, automatic control, brain-function modeling
  - Designing and training neural networks still need experience and experiments.
- Major annual conferences
  - Neural Information Processing Systems (NIPS)
  - International Conference on Machine Learning (ICML)
  - Computational Learning Theory (COLT)
- Major journals
  - Neural Computation
  - IEEE Transactions on Neural Networks
  - Machine Learning