Artificial Neural Networks and Nonparametric Methods

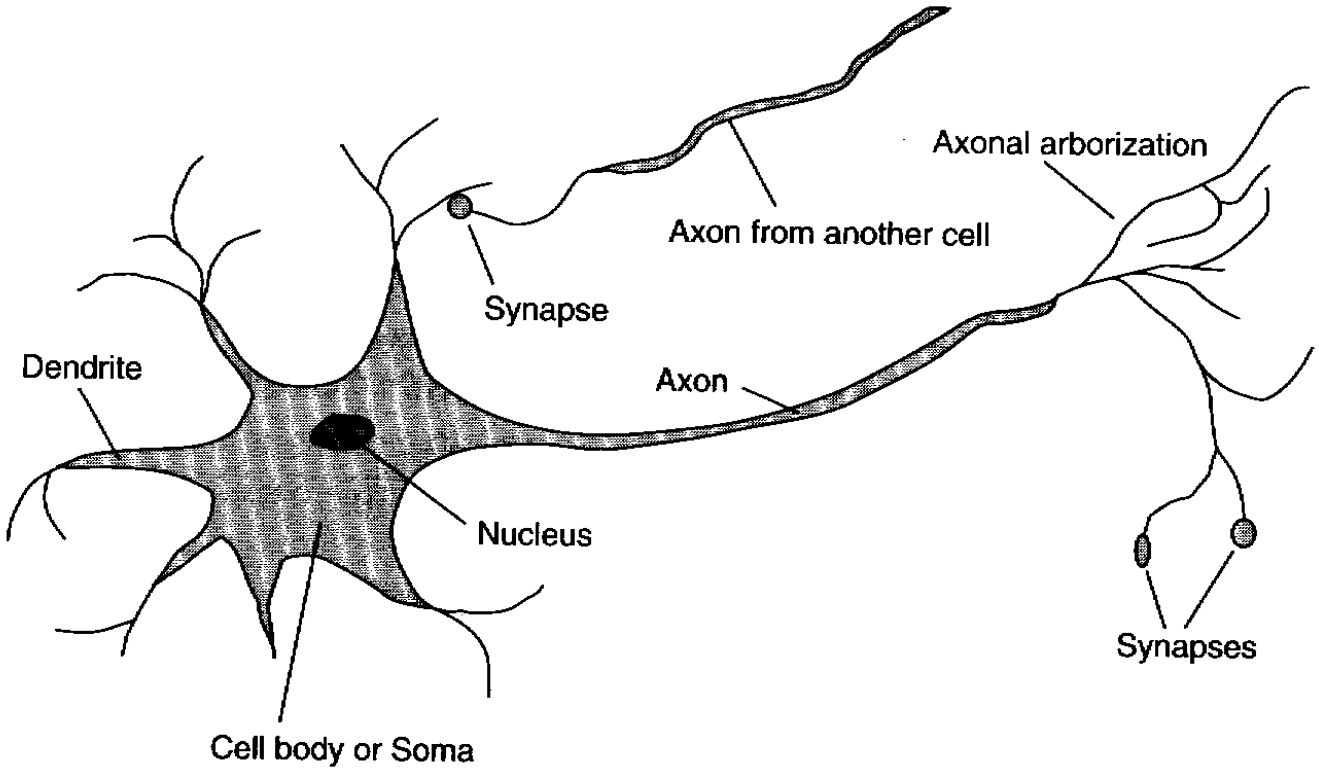
CMPSCI 383
Nov 17, 2011

Today's lecture

- How the brain works (!)
- Artificial neural networks
- Perceptrons
- Multilayer feed-forward networks
  - Error backpropagation algorithm
- Nonparametric methods
  - Nearest neighbor models
  - Nonparametric regression

How the brain works

- Remains a great mystery of science!
- Basic component: the neuron.



The brain and the digital computer

	Computer	Human Brain
Computational units	1 CPU, 10^5 gates	10^{11} neurons
Storage units	10^9 bits RAM, 10^{10} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-8} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec
Neuron updates/sec	10^5	10^{14}

Figure 19.2 A crude comparison of the raw computational resources available to computers (*circa* 1994) and brains.

Even though a computer is a million times faster in raw switching speed, the brain ends up being a billion times faster at what it does.

"Massive parallelism"

Artificial Neural Networks

- **Also called:** neural networks, connectionist systems, neuromorphic systems, parallel distributed processing (PDP) systems, etc.
- Networks of relatively simple processing units, which are very abstract models of neurons; the network does the computation more than the units.

Neuron-like units

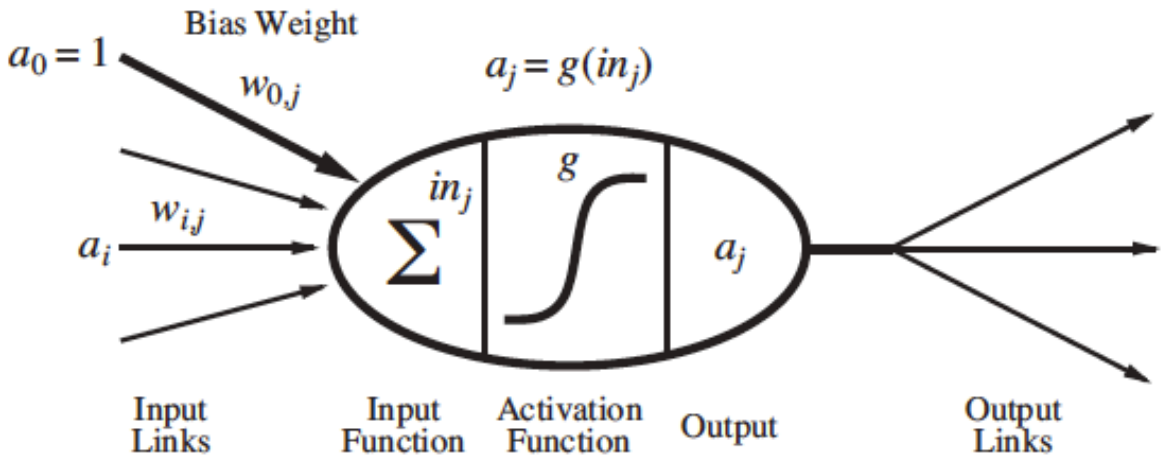
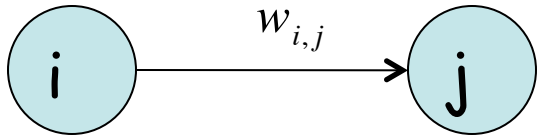
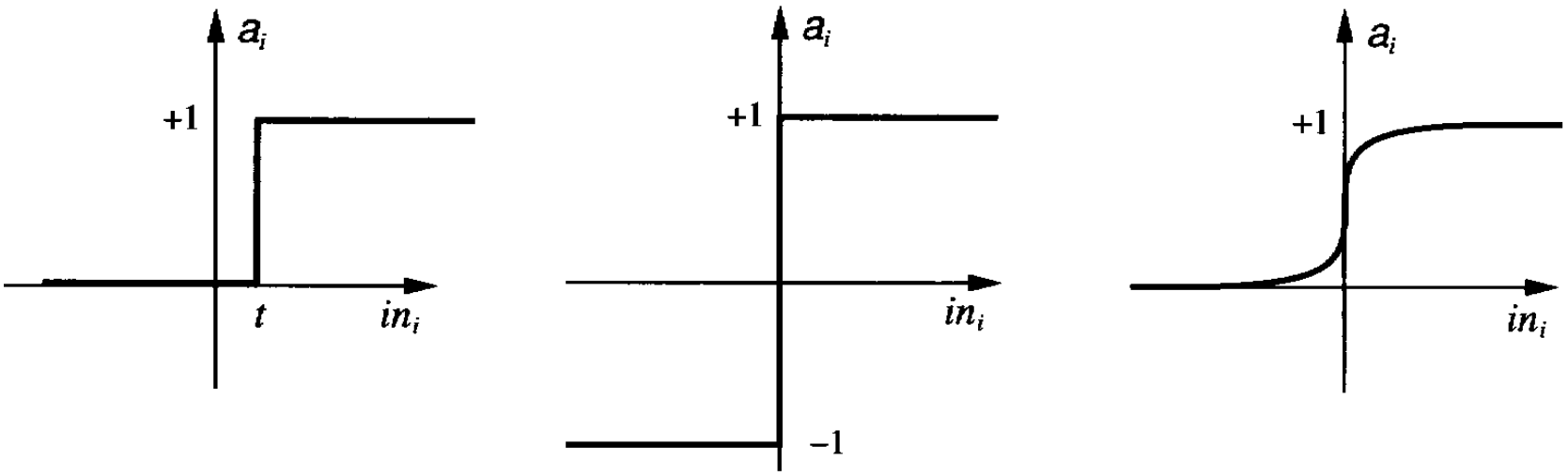


Figure 18.19 **FILES: figures/neuron-unit.eps (Wed Nov 4 11:23:13 2009).** A simple mathematical model for a neuron. The unit's output activation is $a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$, where a_i is the output activation of unit i and $w_{i,j}$ is the weight on the link from unit i to this unit.



Typical activation functions



(a) Step function **(b) Sign function** **(c) Sigmoid function**

Figure 19.5 Three different activation functions for units.

$$\text{step}_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \quad \text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

A useful trick

$$a_j = step_t\left(\sum_{i=1}^{n} w_{i,j} a_i\right) = step_0\left(\sum_{i=0}^{n} w_{i,j} a_i\right)$$

where $w_{0,j} = t$

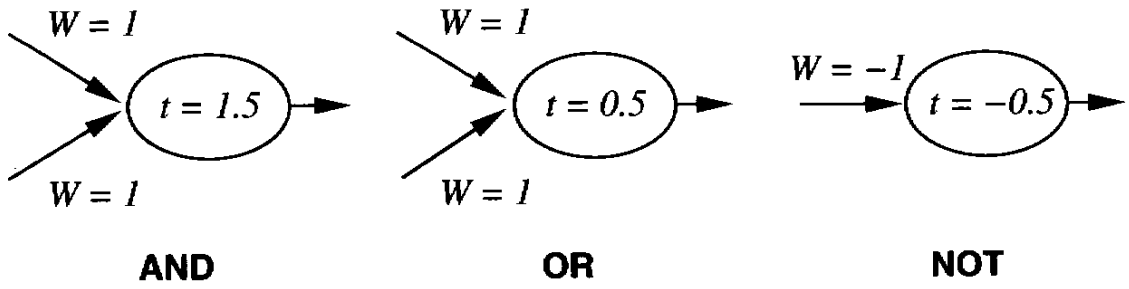
$$a_0 = -1$$

So we can always assume a threshold of 0 if we add an extra input always equal to –1

NNs and logic

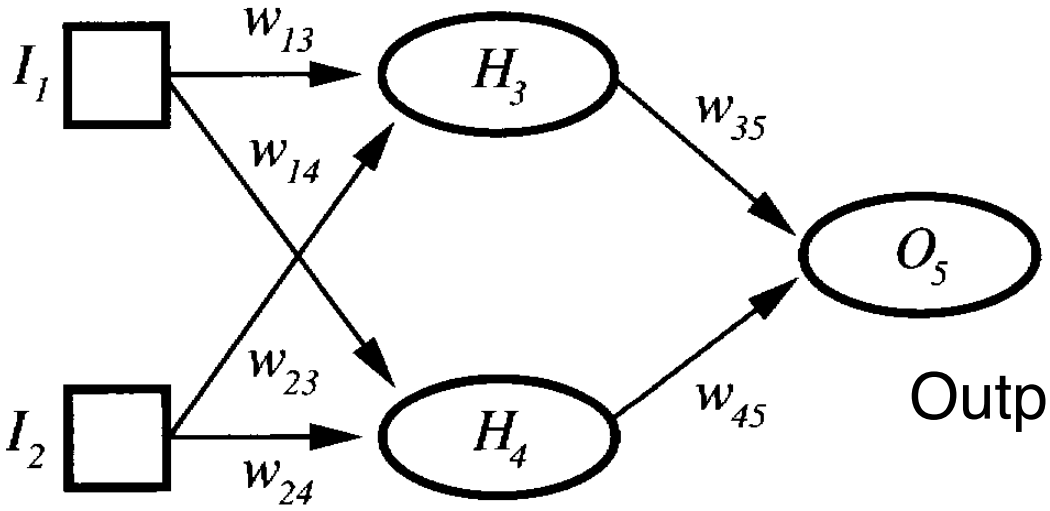
McCulloch and Pitts, 1943: showed that whatever you can do with logic networks, you can do with networks of abstract neuron-like units.

Units with step function activation functions



Network structures

- Feed-forward vs. recurrent networks
- Multi-layer feed-forward networks



Network structures

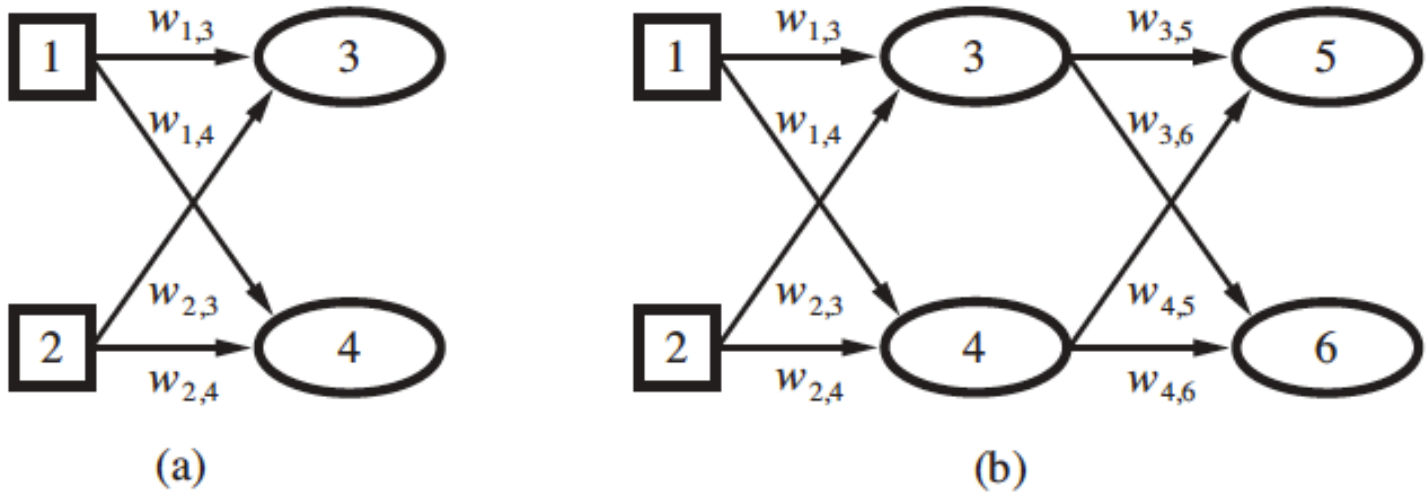
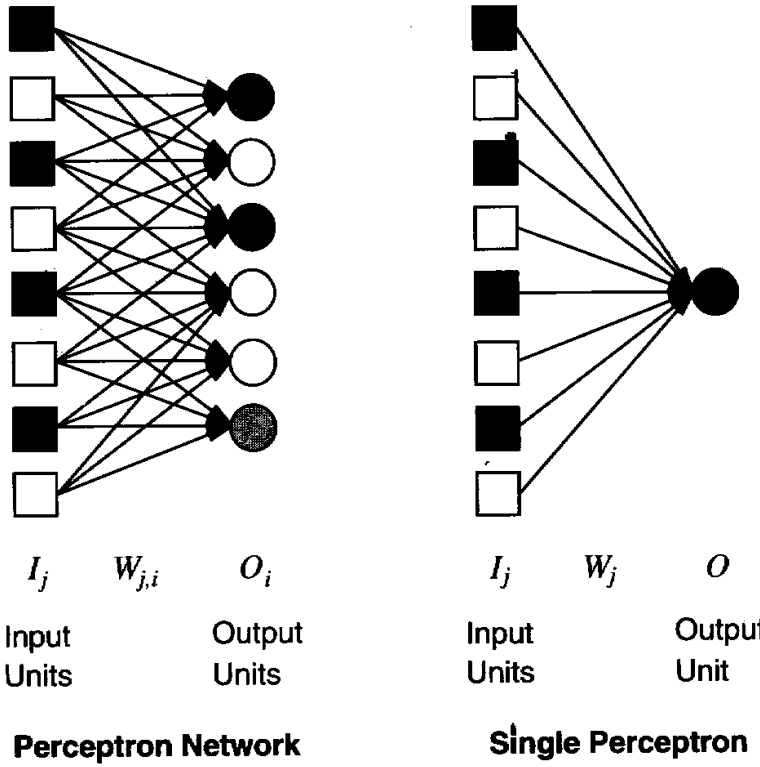


Figure 18.20 FILES: figures/neural-net.eps (Wed Nov 4 11:08:22 2009). (a) A perceptron network with two inputs and two output units. (b) A neural network with two inputs, one hidden layer of two units, and one output unit. Not shown are the dummy inputs and their associated weights.

Perceptrons

- Name given in 1950s to layered feed-forward networks.

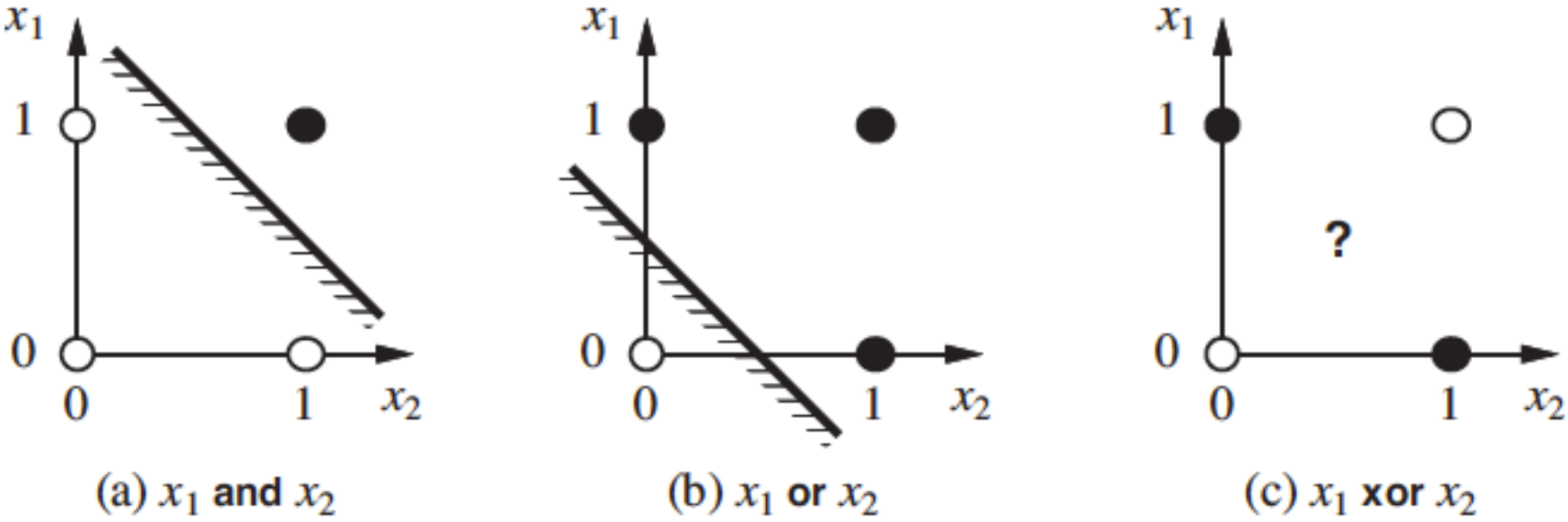


$$O = Step_0(\sum_j W_j I_j)$$

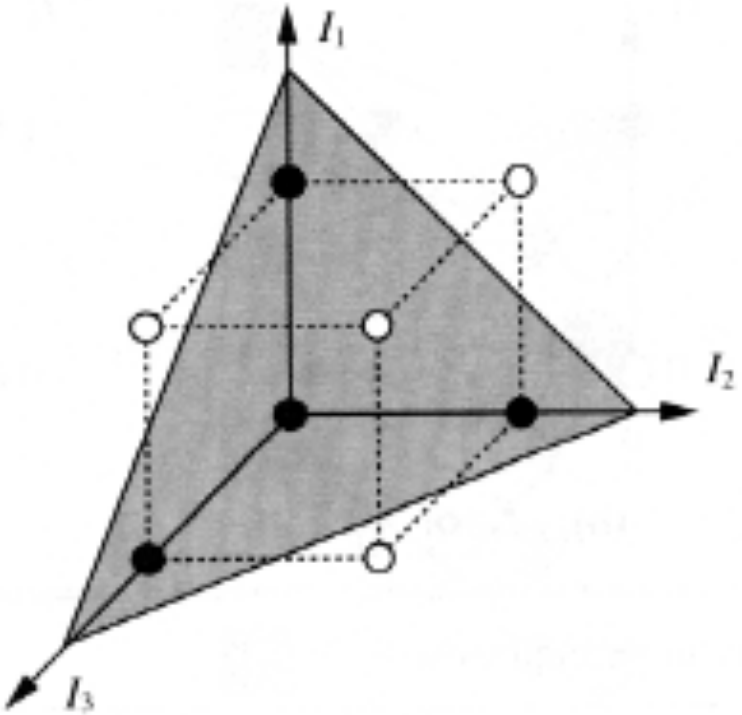
$$= Step_0(W \cdot I)$$

What can perceptrons represent

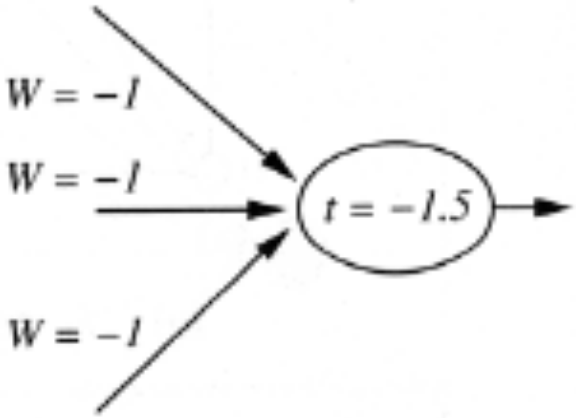
- Only **linearly separable** functions



In three dimensions

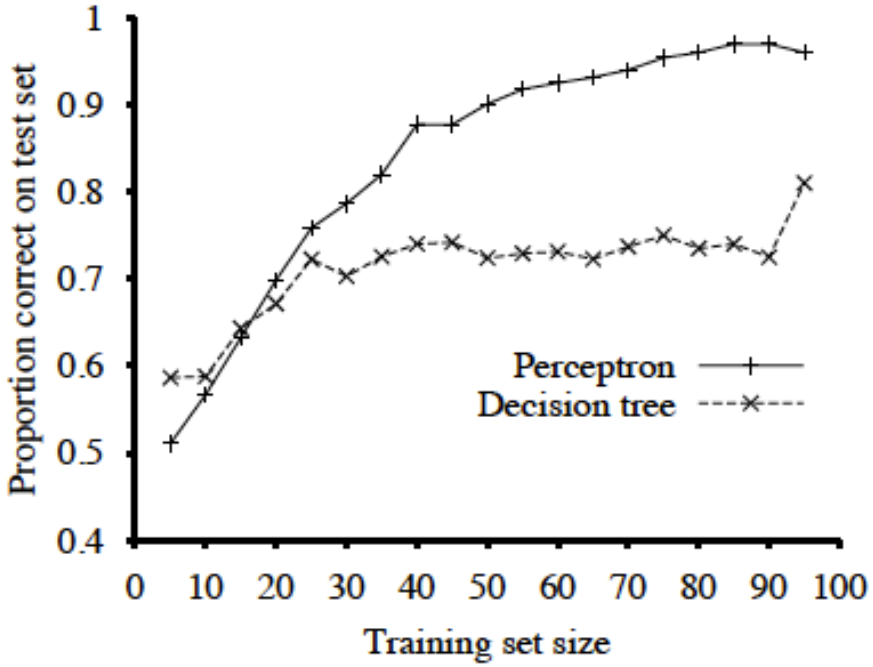


(a) Separating plane

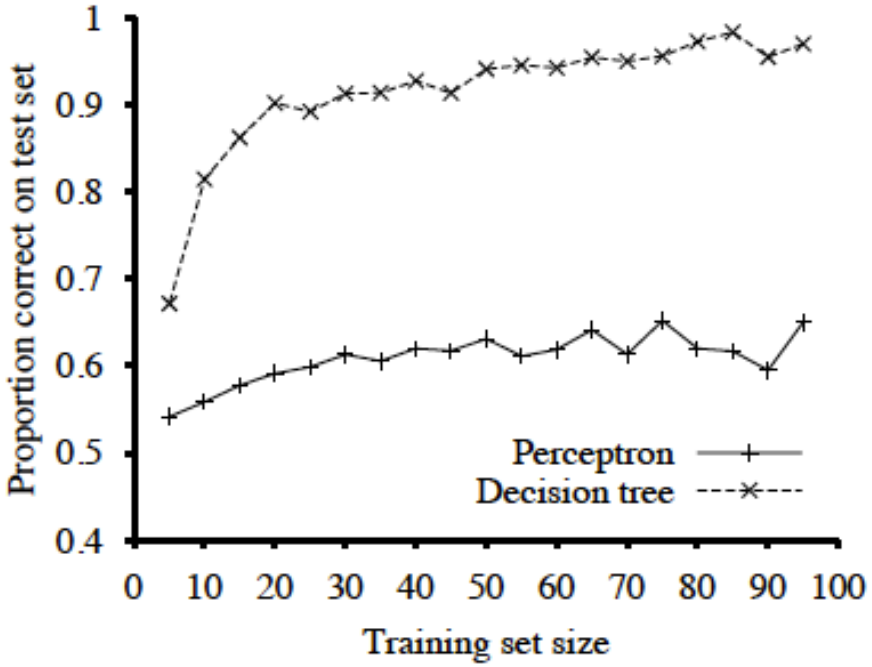


(b) Weights and threshold

Perceptrons vs. Decision trees



Majority function with 11 inputs



WillWait from restaurant example

Multilayer feed-forward nets

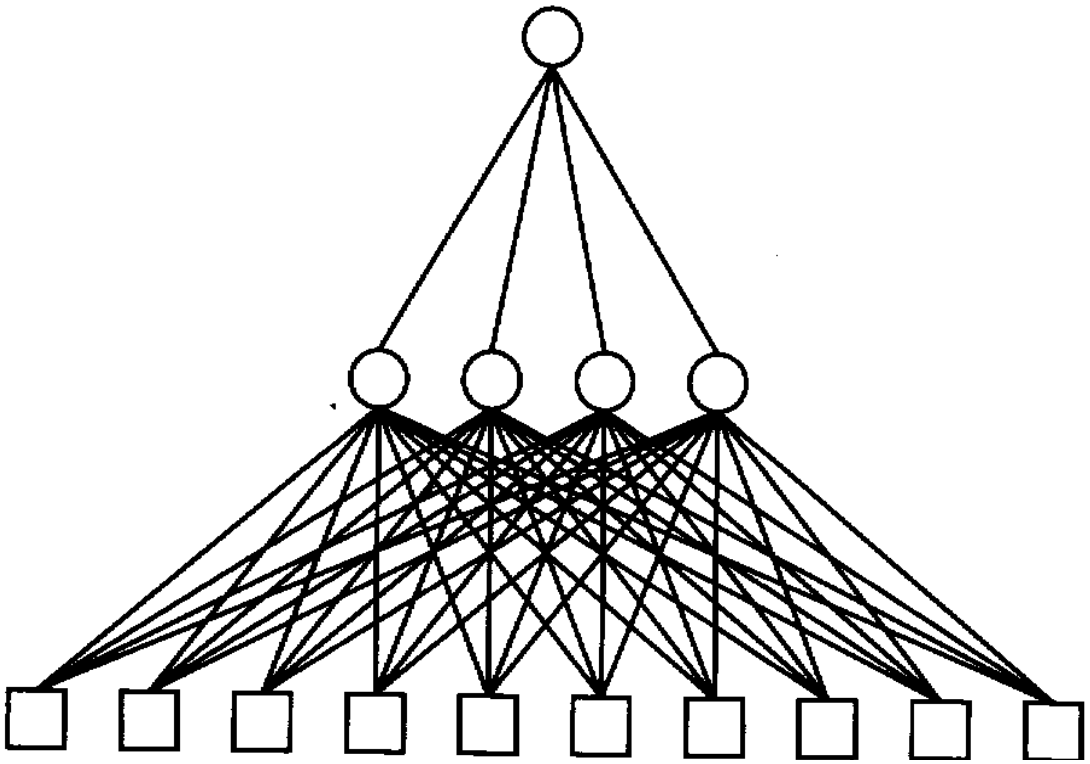
Output units O_i

$W_{j,i}$

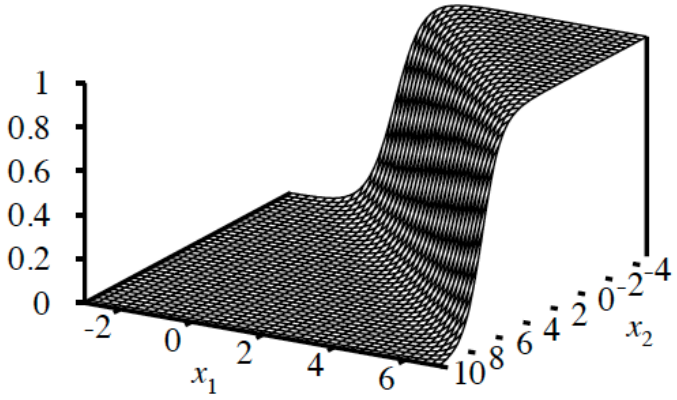
Hidden units a_j

$W_{k,j}$

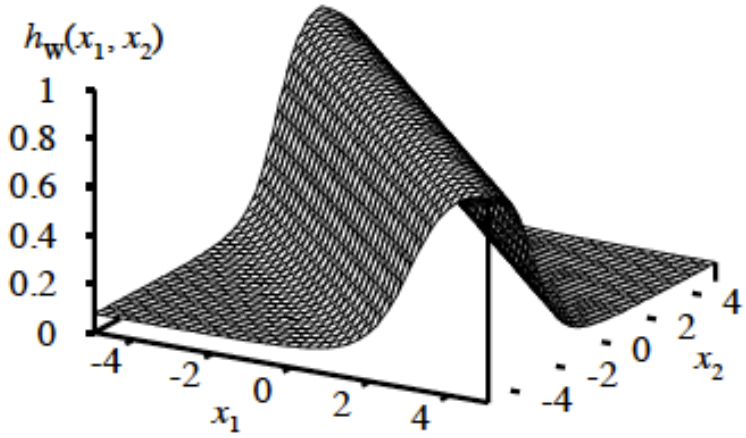
Input units I_k

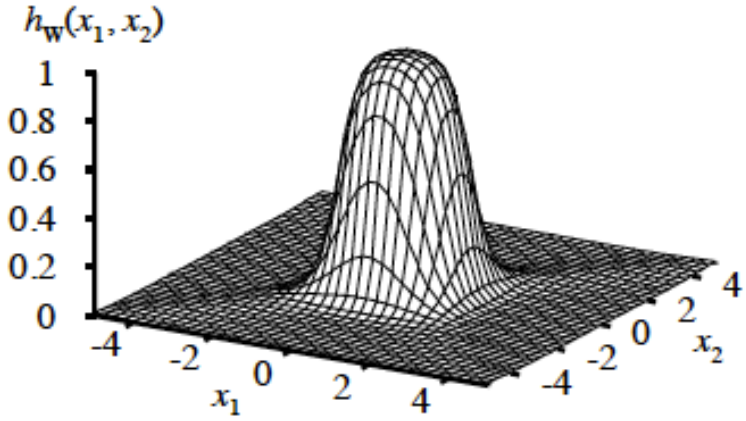


Representing complicated functions



A single logistic unit





Back-propagation learning

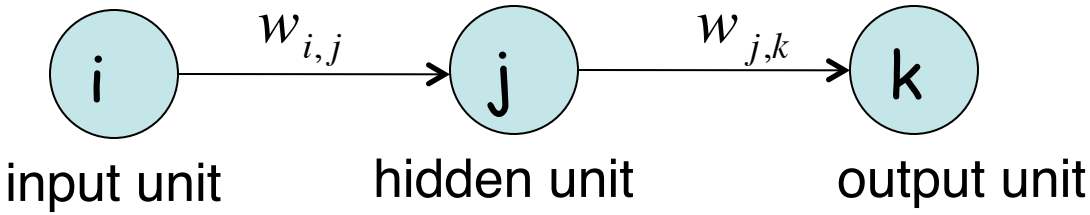
To update weights from hidden units to output unit

$$Err_k = k^{th} \text{ component of } \mathbf{y} - \mathbf{h_w}$$

$$w_{j,k} \leftarrow w_{j,k} + \alpha \times a_j \times Err_k \times g'(in_k)$$

letting $\Delta_k = Err_k \, g'(in_k)$ this becomes

$$w_{j,k} \leftarrow w_{j,k} + \alpha \times a_j \times \Delta_k$$

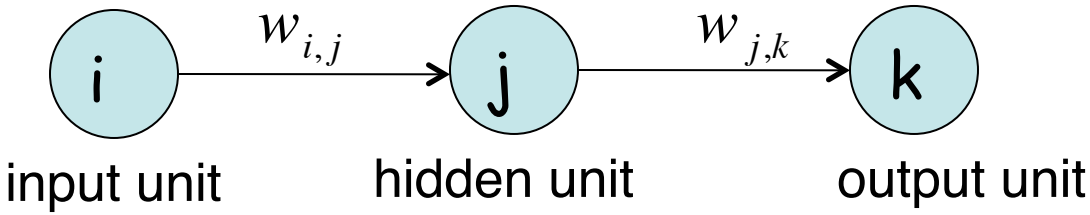


Back-prop contd.

To update weights from input units to hidden units

$$\Delta_j = g'(in_j)\sum_k w_{j,k}\Delta_k$$

$$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta_j$$

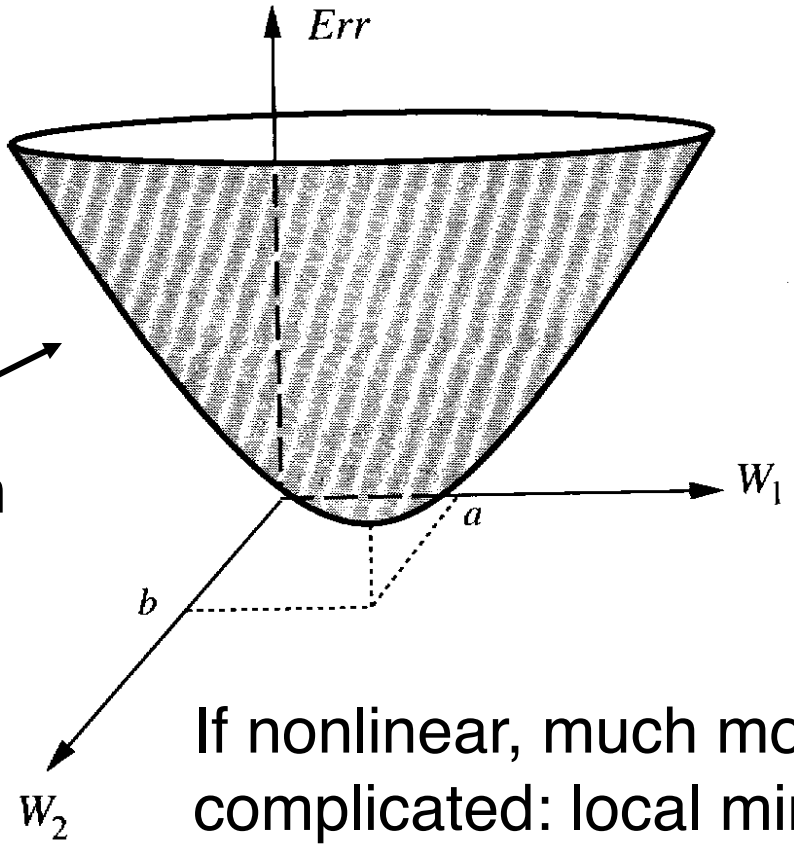


Back-prop as gradient descent

$$Err = \sum_k (y_k - a_k)^2$$

sum squared error

An error surface for net with linear activation functions



If nonlinear, much more complicated: local minima

Multilayer net vs. decision tree

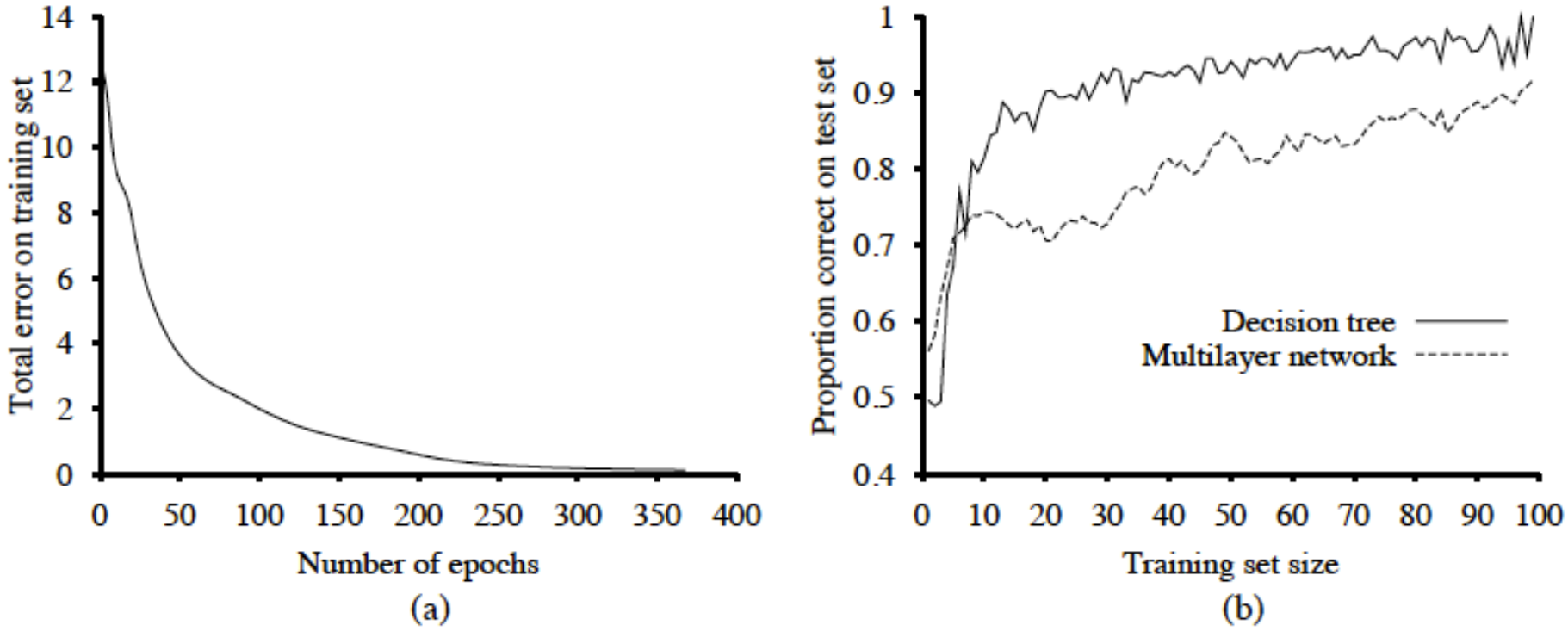


Figure 18.25 **FILES: .** (a) Training curve showing the gradual reduction in error as weights are modified over several epochs, for a given set of examples in the restaurant domain. (b) Comparative learning curves showing that decision-tree learning does slightly better on the restaurant problem than back-propagation in a multilayer network.

Comments on network learning

- **Expressiveness**: given enough hidden units, can represent any function (almost).
- **Computational efficiency**: generally slow to train, but fast to use once trained.
- **Generalization**: good success in a number of real-world problems.
- **Sensitivity to noise**: very tolerant to noise in data

Comments contd.

- **Transparency**: not good!
- **Prior knowledge**: not particularly easy to insert prior knowledge, although possible.

Nonparametric Methods

- Parametric model: a learning model that has a set of parameters of fixed size
  - e.g., linear models, neural networks (of fixed size)
- Nonparametric model: a learning model whose set of parameters is not bounded
  - Parameter set grows with the number of training examples
  - e.g., just save the examples in a lookup table

Nearest Neighbor Models

- K-nearest neighbors algorithm:
  - Save all the training examples
  - For classification: find k nearest neighbors of the input and take a vote (make k odd)
  - For regression: take mean or median of the k nearest neighbors, or do a local regression on them
- How do you measure distance?
- How do you efficiently find the k nearest neighbors?

Distance Measures

Minkowski distance

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = \left(\sum_{\mathrm{i}} \left| x_{j,i} - x_{q,i} \right|^p \right)^{1/p}$$

$p = 1$ Manhattan distance

$p = 2$ Euclidean distance

Hamming distance for Boolean attribute values

k-nearest neighbor for k=1 and k=5

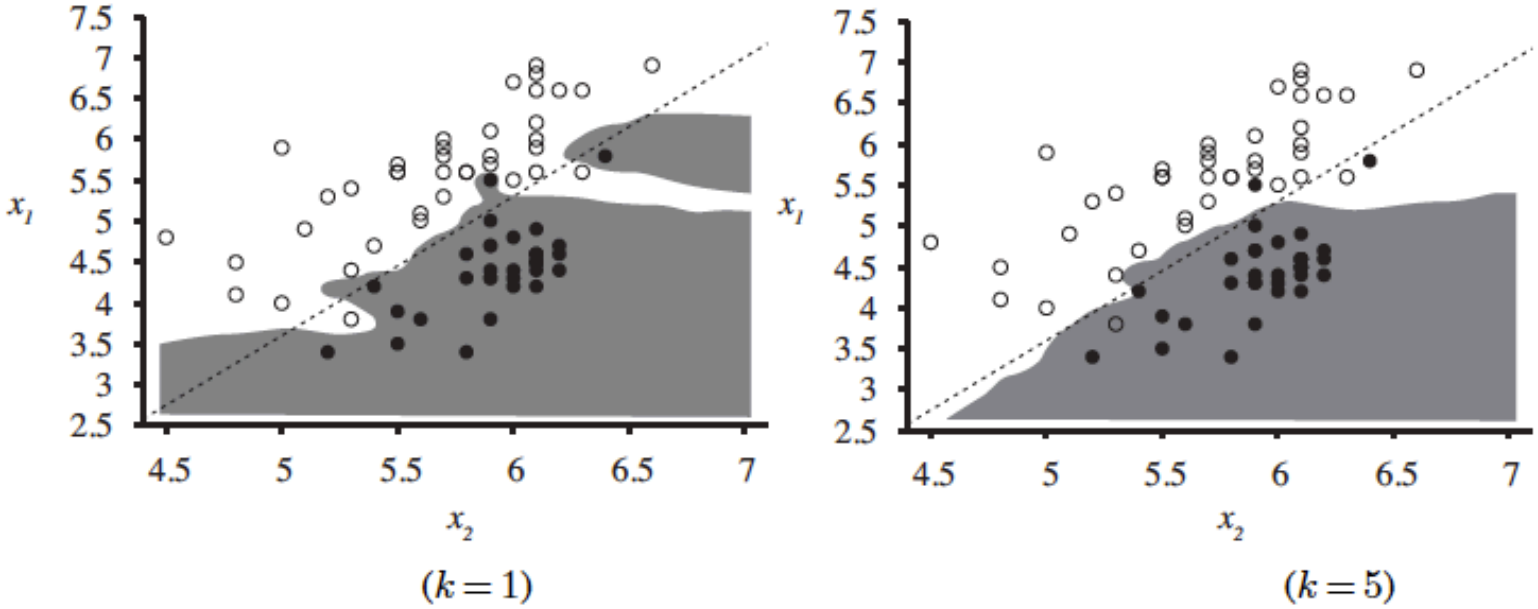


Figure 18.26 FILES: figures/earthquake-nn1.eps (Tue Nov 3 16:22:38 2009) figures/earthquake-nn5.eps (Tue Nov 3 16:22:38 2009). (a) A k-nearest-neighbor model showing the extent of the explosion class for the data in Figure 18.14, with $k=1$. Overfitting is apparent. (b) With $k=5$, the overfitting problem goes away for this data set.

Curse of Dimensionality

In high dimensions, the nearest points tend to be far away.

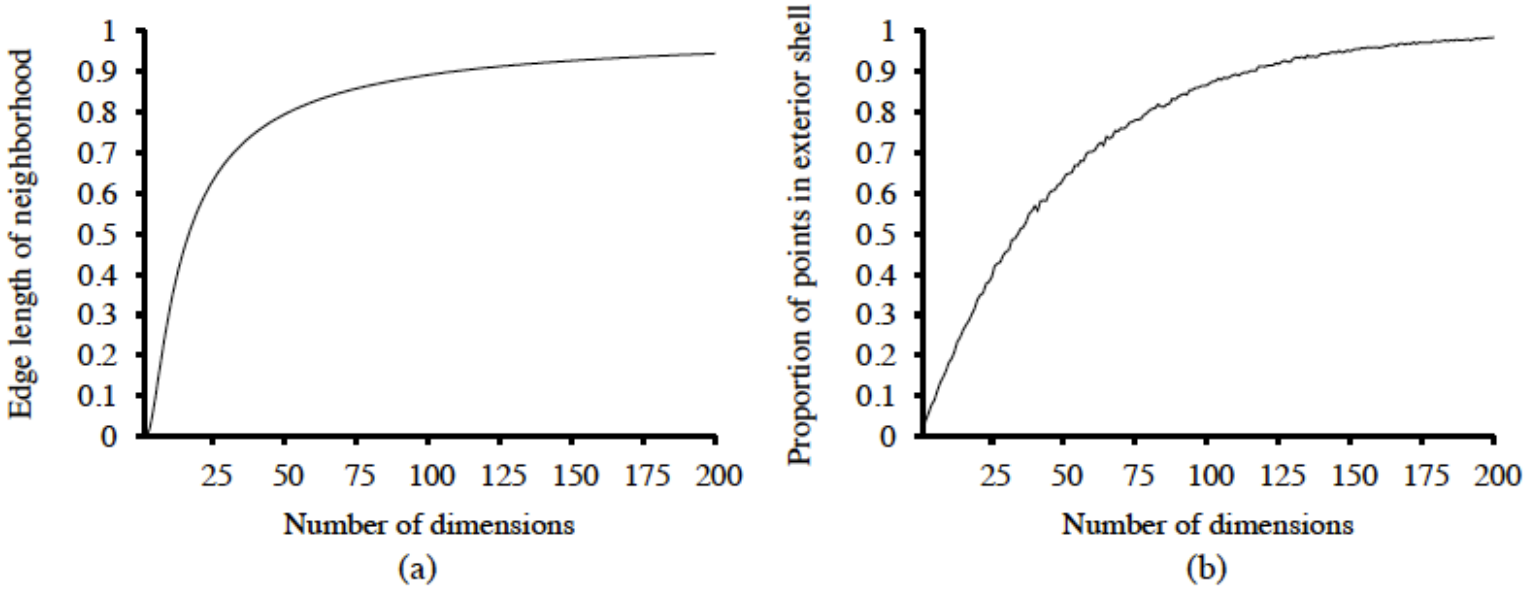


Figure 18.27 **FILES: .** The curse of dimensionality: (a) The length of the average neighborhood for 10-nearest-neighbors in a unit hypercube with 1,000,000 points, as a function of the number of dimensions. (b) The proportion of points that fall within a thin shell consisting of the outer 1% of the hypercube, as a function of the number of dimensions. Sampled from 10,000 randomly distributed points.

Nonparametric Regression

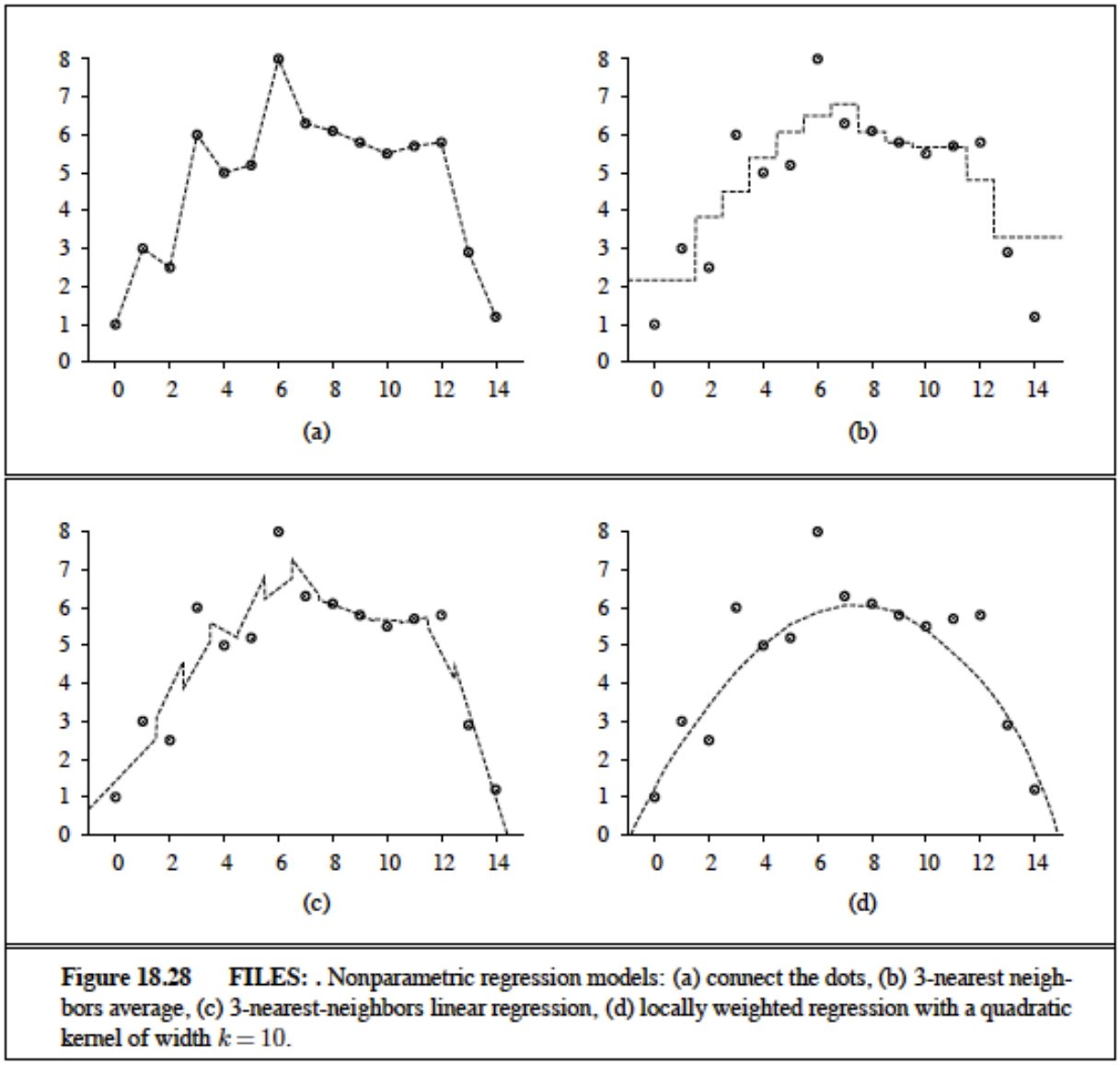


Figure 18.28 **FILES:** . Nonparametric regression models: (a) connect the dots, (b) 3-nearest neighbors average, (c) 3-nearest-neighbors linear regression, (d) locally weighted regression with a quadratic kernel of width $k = 10$.

Locally Weighted Regression

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j K(Distance(\mathbf{x}_q, \mathbf{x}_j))(y_j - \mathbf{w} \cdot \mathbf{x}_j)^2$$

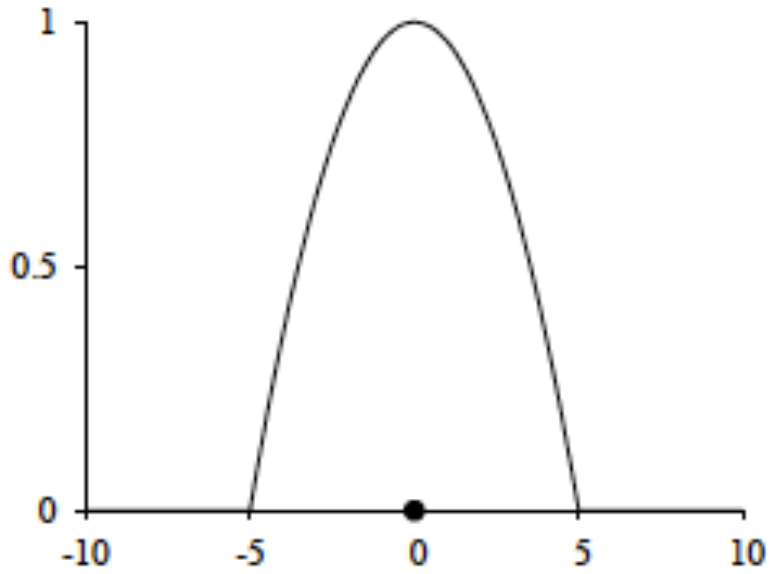


Figure 18.29 **FILES: .** A quadratic kernel, $K(d) = \max(0, 1 - (2|x|/k)^2)$, with kernel width $k = 10$, centered on the query point $x = 0$.

Summary

- How the brain works (!)
- Artificial neural networks
- Perceptrons
- Multilayer feed-forward networks
  - Error backpropagation algorithm
- Nonparametric methods
  - Nearest neighbor models
  - Nonparametric regression

Next Class

- Learning Probabilistic Models
- Secs. 20.1 – 20.4