CS440/ECE448: Intro to Artificial Intelligence

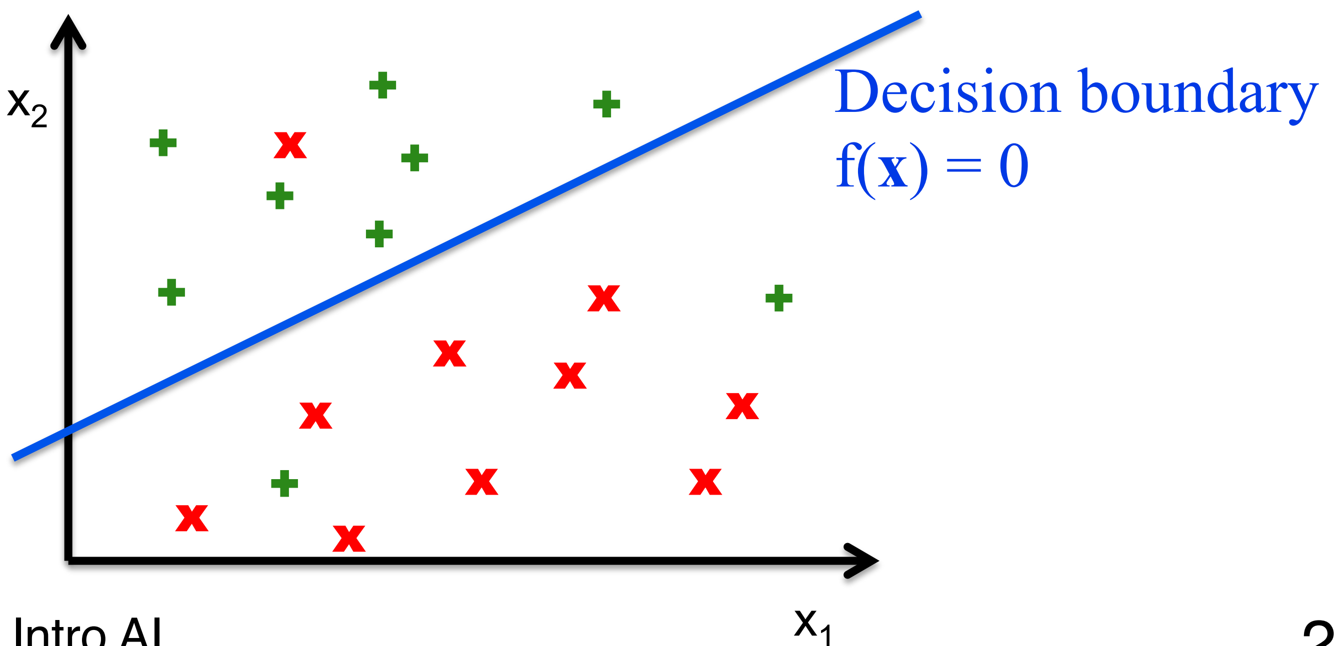
Lecture 25: Perceptrons II

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

Binary classification

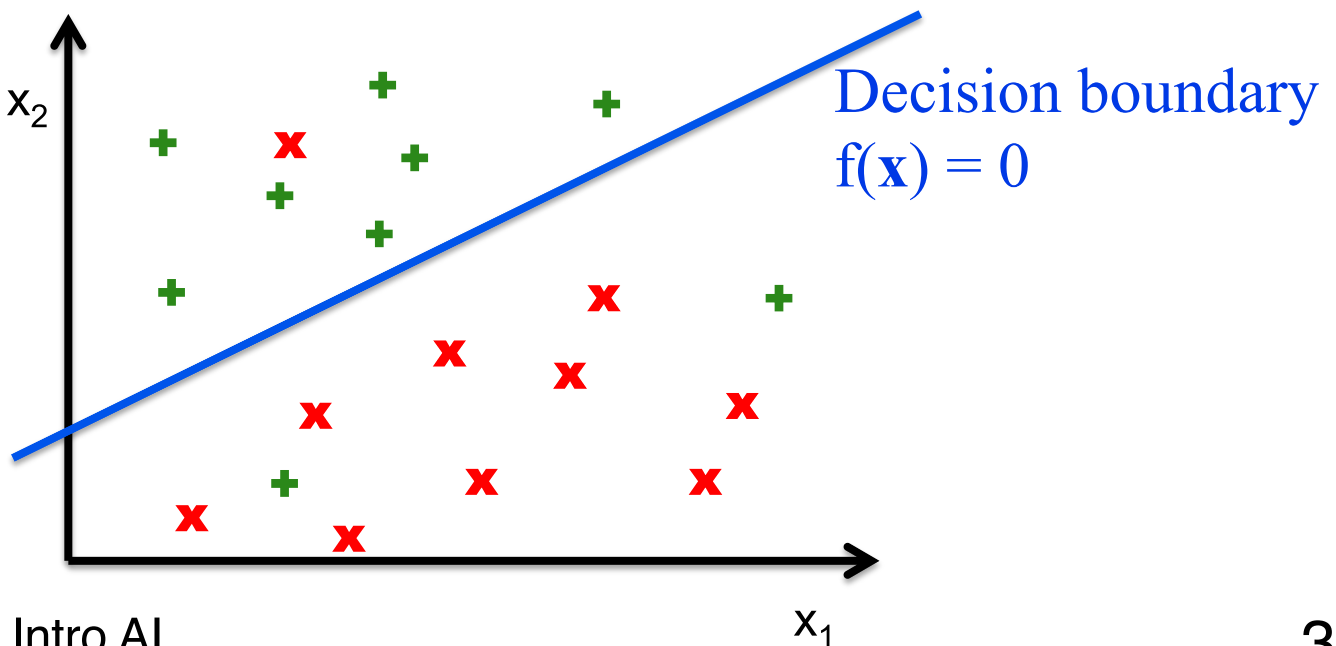
Input: $\mathbf{x} = (x_{1\ldots.}x_d) \in R^d$
Output: return the class predicted by $h_{\mathbf{w}}(\mathbf{x})$
$h_{\mathbf{w}}(\mathbf{x})$: if $f(\mathbf{x}) = \mathbf{wx} > 0$ return $y = 1$, else return $y = 0$



Binary classification: training

Input: $\{(\mathbf{x}^{\mathbf{i}}, y^{i})\}$ with $(x_{1 \ldots} x_d) \in R^d \quad y^i \in \{+1, -1\}$

Task: Find weights $\mathbf{w} = (w_0\ w_{1 \ldots} w_d) \in R^{d+1}$
that define $f(\mathbf{x}) = \mathbf{wx}$



Perceptron algorithm

Given training data $\{(\mathbf{x}^1,y^1),\ldots,(\mathbf{x}^j,y^j),\ldots,(\mathbf{x}^N,y^N)\}$

- Start with initial weight vector **w**
- **Online update:** Update **w** for each $(\mathbf{x}^j,y^j)$
  $w_i := w_i + \alpha\,(y^j - h_{\mathbf{w}}(\mathbf{x}^j))x_i^j$
- **Batch update:** Go through entire data set before updating **w**
  $\Delta w_i = \sum_j (\alpha\,(y^j - h_{\mathbf{w}}(\mathbf{x}))x_i^j)$ $\quad w_i := w_i + \Delta w_i$
- The learning rate α decays over time

Perceptron Example Space

Input: a vector of *n* components

If input is a vector of Booleans

example space
= n-dimensional Boolean hypercube

If the input is a vector of real numbers

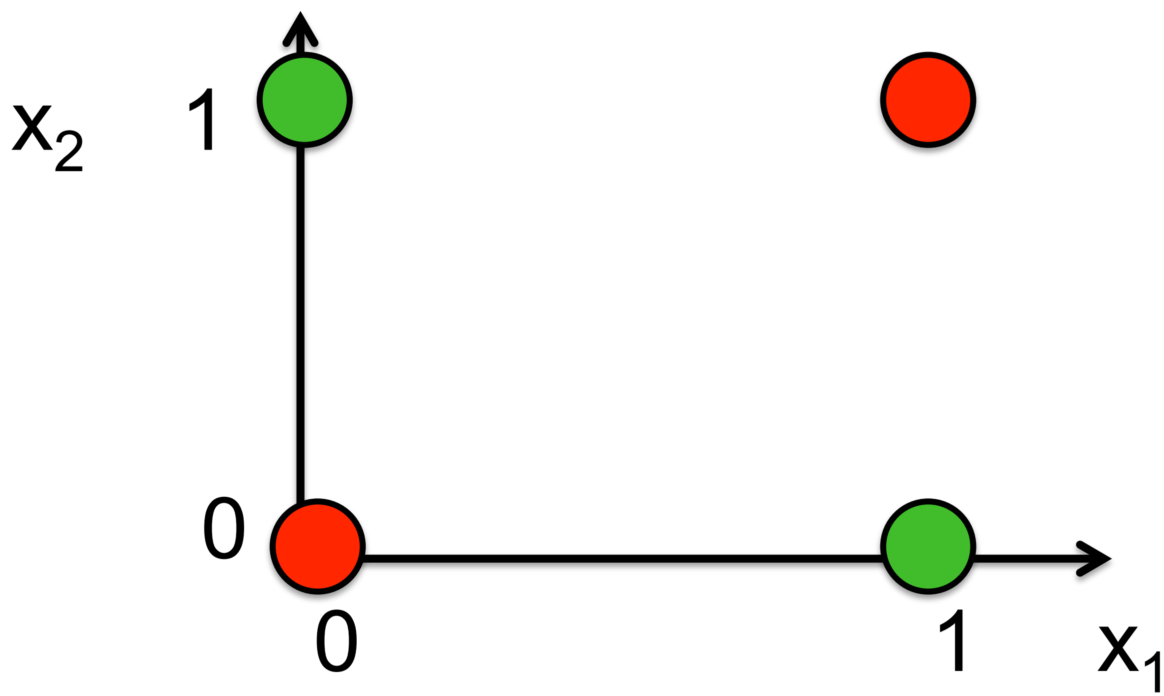
example space
= n-dimensional real space $\Re^n$

Perceptron hypothesis space

Each perceptron defines a hyperplane in the example space.

Not all concepts can be expressed by a hyperplane.

Boolean XOR



XOR is not linearly separable

Does linear separability make sense?

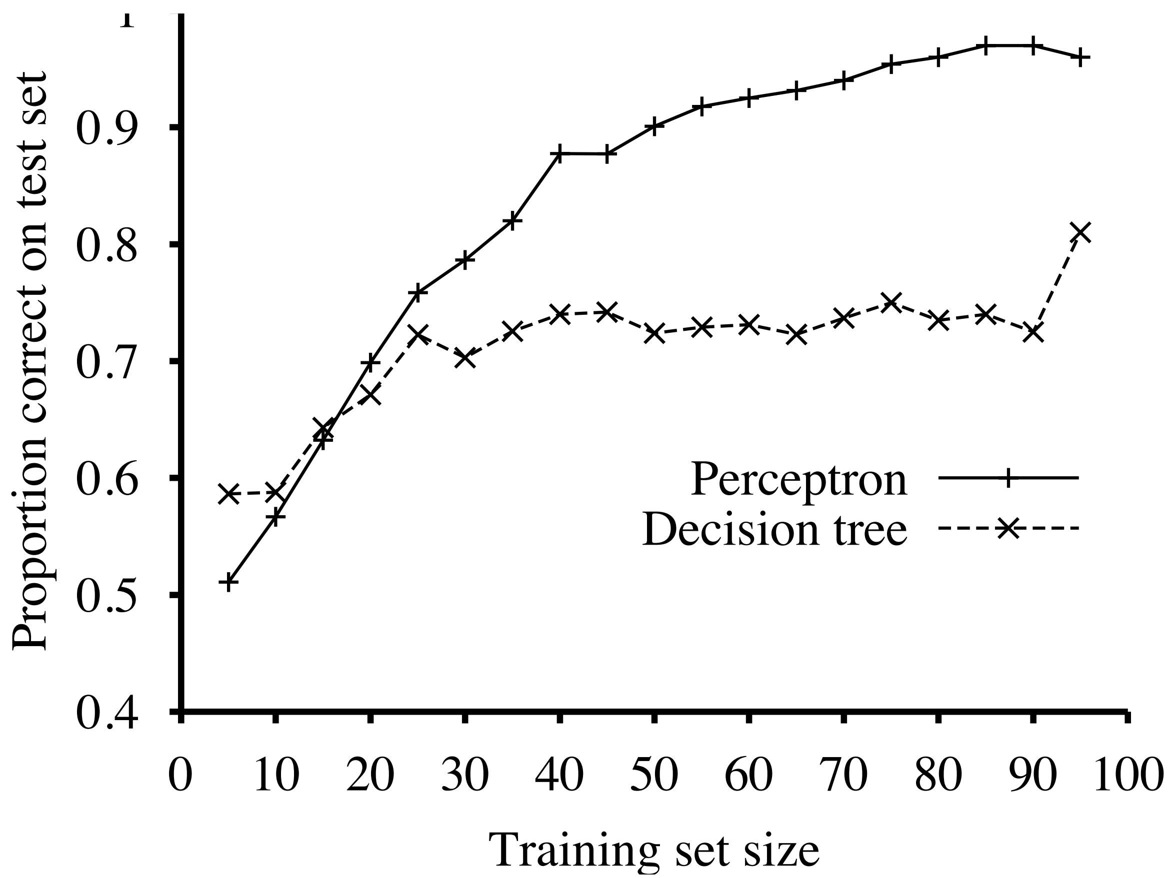
How often is it the case that a data set will be linearly separable?

Given N random data points in d dimensions.
Assume we randomly assign classes C1, C2 to these data points.
(C1 and C2 have equal probability)

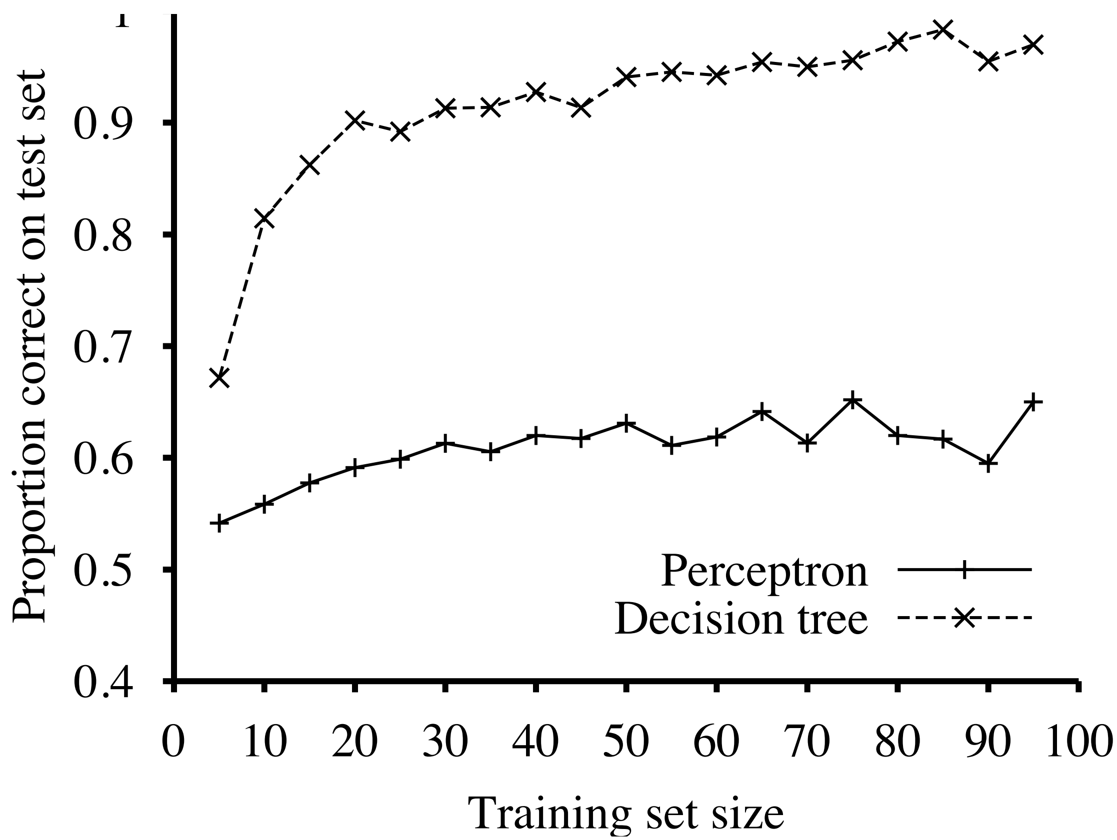
Each assignment of classes = one concept.
There are 2^N concepts.
How many concepts are linearly separable?

If $N \leq d+1$: all concepts are linearly separable
If $N = 2(d+1)$: half of the concepts are linearly separable

Perceptron and decision tree: majority function

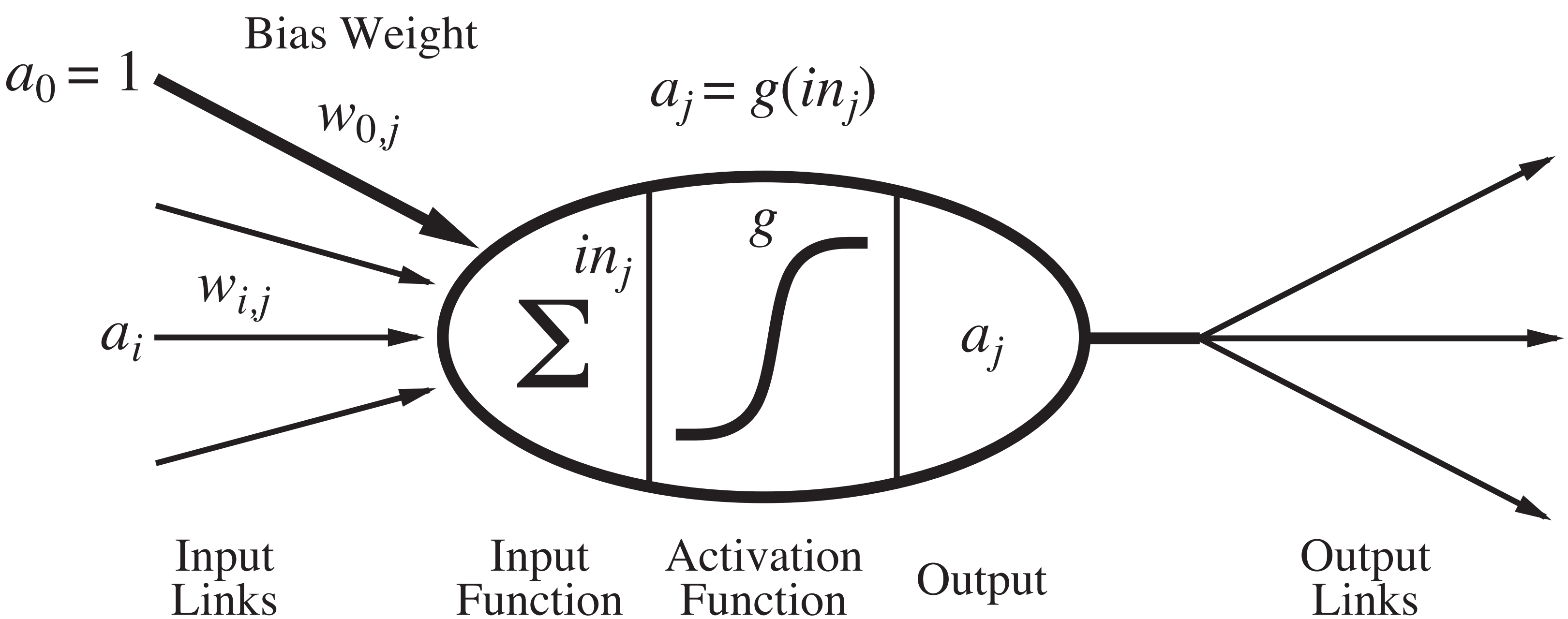


Perceptron and decision tree: non-linearly separable problem



From perceptrons to neural networks

We can think of a single perceptron as one neuron



From perceptrons to neural networks

A neural network consists of nodes connected by directed links.

Each node has an activation a_i
Links a_{ij} propagate the activation a_i from i to j.

Each link has a weight w_{ij} that determines the strength and sign of the connection

From perceptrons to neural networks

Each unit computes a weighted sum of its inputs: $in_j = \sum_j w_{ij}\, a_{ij}$

and applies an **activation function** to this input

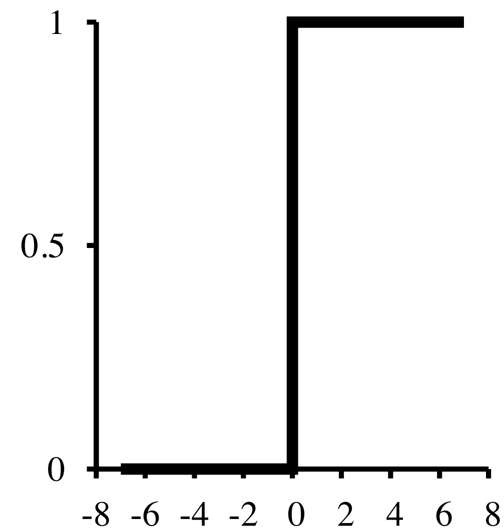
$$a_j = g(in_j) = g(\textstyle\sum_j w_{ij}\, a_{ij})$$

activation function: linear threshold or sigmoid threshold

The perceptron threshold

The perceptron uses a hard threshold function:
$h_{\mathbf{w}}(\mathbf{x})$: if $f(\mathbf{x}) = \mathbf{wx} > 0$ return $y = 1$, else return $y = 0$

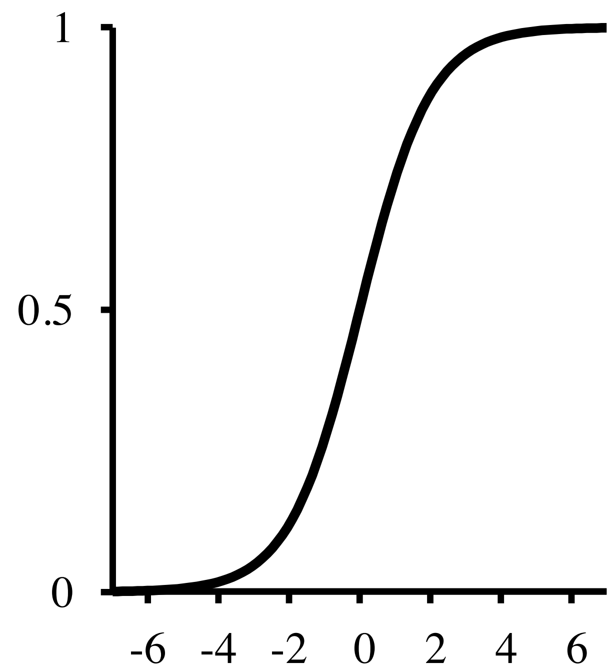
This is a non-differentiable function, so we cannot use gradient descent.



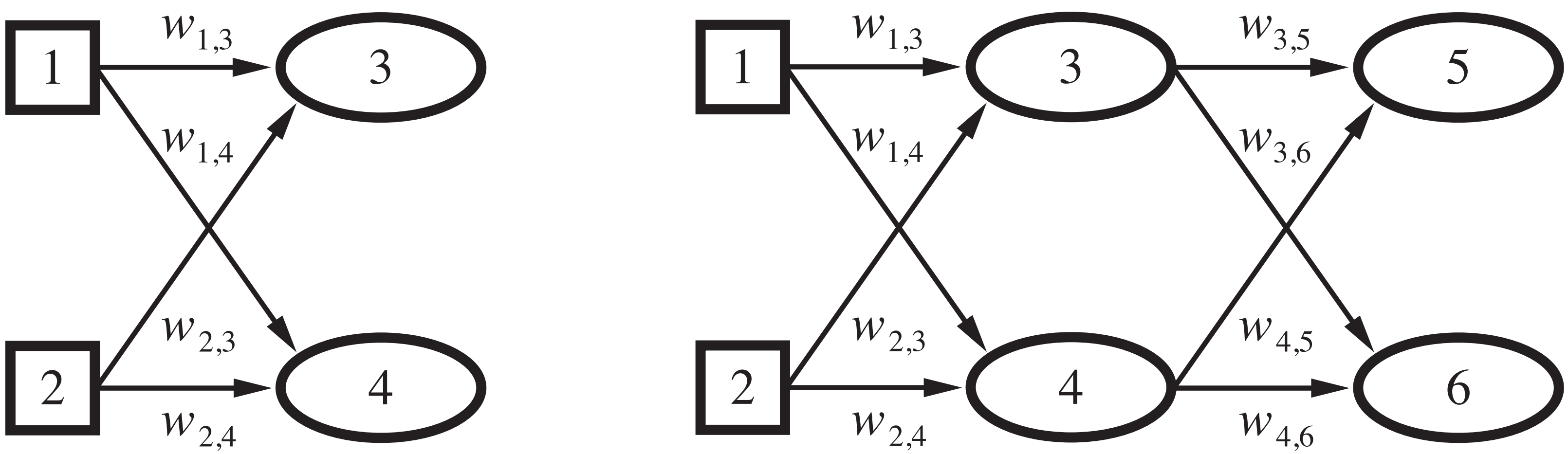
The sigmoid threshold

The logistic (sigmoid) function is differentiable.
We can also think of it as a probability

$$h_{\mathbf{w}}(x) = \frac{1}{1 + e^{-\mathbf{wx}}}$$



Perceptron networks



(a) (b)