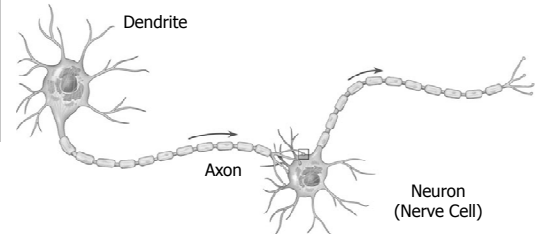


CS522 Advanced Database Systems  
 Classification: Introduction to Artificial Neural Network

Chengyu Sun  
 California State University, Los Angeles

## Biological Neural Network

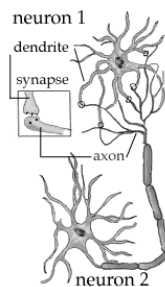


- ◆ Dendrites receives signals from other neurons
- ◆ Axons gives signals to other neurons

Min Chul Jung Introduction to Artificial Neural Networks

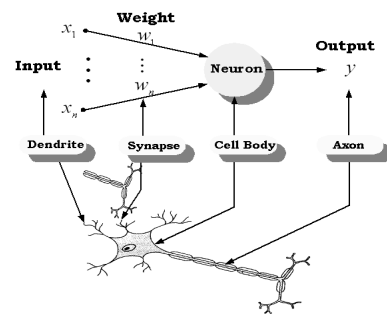
## Synapse

- ◆ *Synapse* is the contact point between a dendrite and an axon
- ◆ Human brain learns by changing the strength of the synaptic connection between neurons upon repeat stimulation by the same impulse



Min Chul Jung Introduction to ANN

## Modeling a Neuron



Min Chul Jung Introduction to Artificial Neural Networks

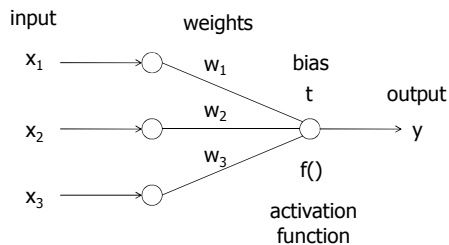
## ... Modeling a Neuron

- ◆ In a Biological Neural Network (BNN), learning results are saved at synapses
- ◆ In a Artificial Neural Network (ANN), learning results are saved in the *weights*

## Example

A1	A2	A3	C
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	0	0	0
0	1	1	?

## Perceptron



## Input, Weights, and Bias

- ◆ Input  $\mathbf{X}=[x_1, x_2, x_3]$ 
  - One per attribute
- ◆ Weights  $\mathbf{W}=[w_1, w_2, w_3]$ 
  - One per input
  - Typically initialized to random values in the range  $[-1.0, 1.0]$  or  $[-0.5, 0.5]$
- ◆ Bias  $t$ 
  - Typically a value in the range of  $[-1.0, 1.0]$

## Output

$$\begin{aligned}
 y &= f(x_1 w_1 + x_2 w_2 + \dots + x_n w_n + t) \\
 &= f(x_1 w_1 + x_2 w_2 + \dots + x_n w_n + x_0 w_0) \\
 &= f(\mathbf{X} \bullet \mathbf{W}) \\
 &= f(\mathbf{XW}^T)
 \end{aligned}$$

$t$  can be written as  $x_0 w_0$  where  $x_0=1$  and  $w_0=t$

## Common Activation Functions

Step Function	Sign Function
$y = \begin{cases} 1 & \mathbf{X} \bullet \mathbf{W} \geq 0 \\ 0 & \mathbf{X} \bullet \mathbf{W} < 0 \end{cases}$	$y = \begin{cases} 1 & \mathbf{X} \bullet \mathbf{W} \geq 0 \\ -1 & \mathbf{X} \bullet \mathbf{W} < 0 \end{cases}$
Linear Function	Sigmoid Function
$y = \mathbf{X} \bullet \mathbf{W}$	$y = \frac{1}{1 + e^{-\mathbf{X} \bullet \mathbf{W}}}$

## Learning

- ◆ Initialize  $\mathbf{w}$  and  $t$  to random values
- ◆ For each training record  $(\mathbf{x}, y')$ 
  - Compute the predicted output  $y$
  - Update each weight  $w_i$

$$w_i = w_i + \lambda(y' - y)x_i$$

$\lambda$  is the *learning rate*

## About Learning Rate

- ◆ Between 0 and 1
- ◆ Control the speed of adjustment
- ◆ Dynamic learning rate, e.g.  $1/t$  where  $t$  is the number of iterations so far

## Learning Example

$$\mathbf{w} = [0.3, 0.3, 0.3], \quad t = -0.5, \quad \lambda = 0.5$$

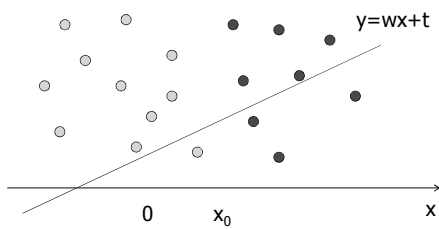
$$y = \begin{cases} 1 & 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.5 \geq 0 \\ 0 & 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.5 < 0 \end{cases}$$

	A1	A2	A3	C
R1	0	1	0	1
R2	1	1	0	0
R3	0	0	1	1
R4	1	0	1	0
R5	0	0	0	0

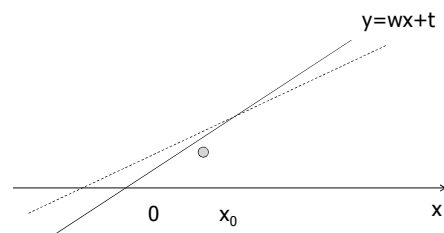
## ... Learning Example

After	$[\mathbf{w}, t]$
	$[0.3, 0.3, 0.3, -0.5]$
R1	$[0.3, 0.8, 0.3, 0]$
R2	$[-0.2, 0.3, 0.3, -0.5]$
R3	$[-0.2, 0.3, 0.8, 0]$
R4	$[-0.7, 0.3, 0.3, -0.5]$
R5	$[-0.7, 0.3, 0.3, -0.5]$

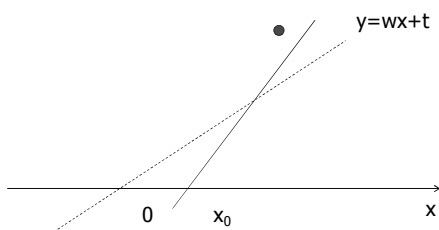
## Intuition Behind Perceptron Learning ...



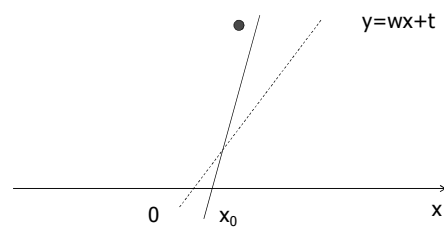
## ... Intuition Behind Perceptron Learning



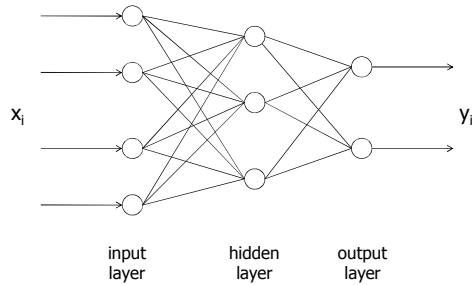
## ... Intuition Behind Perceptron Learning ...



## ... Intuition Behind Perceptron Learning



## Multilayer ANN



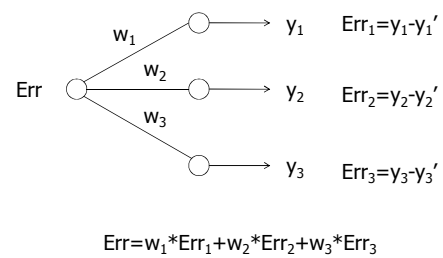
## Terminology

- ◆ Nodes and units
- ◆ Layers and number of layers
- ◆ Feed-forward and recurrent
- ◆ Fully connected

## Learning

- ◆ Each node in a hidden or output layer is a perceptron
- ◆ The output of a node is used as the input for the nodes in the next layer
- ◆ Can we use the same perceptron learning process for multilayer ANN learning??

## Understand Backpropagation



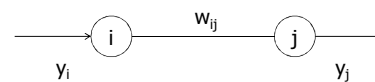
## Adjust Weights

- ◆ To minimize sum of squared error

$$E(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^n (y_i - y_i')^2$$

$$= \frac{1}{2} \sum_{i=1}^n (y_i - f(\mathbf{X} \cdot \mathbf{W}))^2$$

## Delta Rule – Linear Activation Function



$$\Delta w_{ij} = \lambda y_i (y_j' - y_j)$$

## Delta Rule – Nonlinear Activation Function ...

For output nodes:

$$\Delta w_{ij} = \lambda y_i f'_j(\mathbf{X} \bullet \mathbf{W})(y'_j - y_j)$$

$$Err_j = f'_j(\mathbf{X} \bullet \mathbf{W})(y'_j - y_j)$$

See Delta Rule at <http://www.learnartificialneuralnetworks.com/backpropagation.html>

## ... Delta Rule – Nonlinear Activation Function

For hidden nodes:

$$\Delta w_{ij} = \lambda y_i f'_j(\mathbf{X} \bullet \mathbf{W}) \sum_{k=1}^n w_{jk} Err_k$$

$$Err_j = f'_j(\mathbf{X} \bullet \mathbf{W}) \sum_{k=1}^n w_{jk} Err_k$$

See Delta Rule at <http://www.learnartificialneuralnetworks.com/backpropagation.html>

## Sigmoid Activation Function

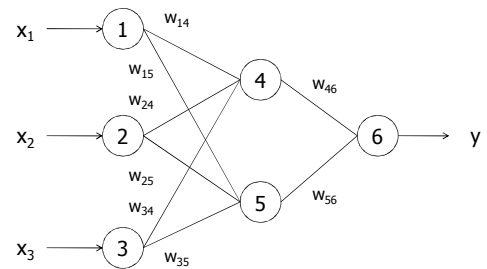
$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = (1 - f(x))f(x)$$

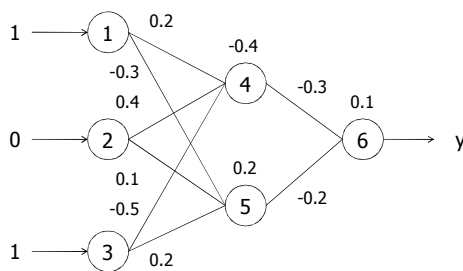


$$f'(\mathbf{X} \bullet \mathbf{W}) = (1 - f(\mathbf{X} \bullet \mathbf{W}))f(\mathbf{X} \bullet \mathbf{W}) = (1 - y)y$$

## Multilayer ANN Example



## Initial Values



## Initial Values

$x_1$	$x_2$	$x_3$	$t_4$	$t_5$	$t_6$		
1	0	1	-0.4	0.2	0.1		
$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2

## Forward Computation

Activation function:  $f(x) = \frac{1}{1 + e^{-x}}$

Node	$\mathbf{X} \cdot \mathbf{W}$	$f(\mathbf{X} \cdot \mathbf{W})$
4	-0.7	0.332
5	0.1	0.525
6	-0.105	0.474

## Backward Propagation ...

Assume  $y=1$

Node	$f'(\mathbf{X} \cdot \mathbf{W})$	Err
6	0.249	0.131
5	0.249	-0.0065
4	0.222	-0.0087

## ... Backward Propagation

$w_{46}$	$w_{56}$	$t_6$	
-0.261	-0.138	0.218	
$w_{15}$	$w_{25}$	$w_{35}$	$t_5$
-0.306	0.1	0.194	0.194
$w_{14}$	$w_{24}$	$w_{34}$	$t_4$
0.192	0.4	-0.508	-0.408

## Input and Output of ANN

### ◆ Input

- One input node for each binary or numeric attribute
- What about categorical attributes??

### ◆ Output

- One output node for a 2-class problem
- K output nodes for a k-class problem

## About ANN

- ◆ Multilayer feed-forward networks can approximate any function
- ◆ Determining the network topology is an empirical process
- ◆ Good at handling redundant features, but sensitive to noise
- ◆ Weight adjustment may converge to local minimum
- ◆ Training can be time consuming

## Readings

- ◆ Textbook Chapter 6.6