

# Neural Networks

July 7, 2005  
CS 486/686  
University of Waterloo

## Outline

- Neural networks
  - Perceptron
  - Supervised learning algorithms for neural networks
- Reading: R&N Ch 20.5

CS486/686 Lecture Slides (c) 2005 P. Pasupat

2

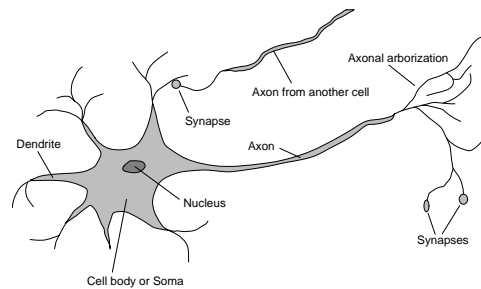
## Brain

- Seat of human intelligence
- Where memory/knowledge resides
- Responsible for thoughts and decisions
- Can learn
- Consists of nerve cells called **neurons**

CS486/686 Lecture Slides (c) 2005 P. Pasupat

3

## Neuron



CS486/686 Lecture Slides (c) 2005 P. Pasupat

4

## Comparison

- Brain
  - Network of neurons
  - Nerve signals propagate in a neural network
  - **Parallel computation**
  - **Robust (neurons die everyday without any impact)**
- Computer
  - Bunch of gates
  - Electrical signals directed by gates
  - **Sequential computation**
  - **Fragile (if a gate stops working, computer crashes)**

CS486/686 Lecture Slides (c) 2005 P. Pasupat

5

## Artificial Neural Networks

- Idea: **mimic the brain to do computation**
- Artificial neural network:
  - Nodes (a.k.a units) correspond to neurons
  - Links correspond to synapses
- Computation:
  - Numerical signal transmitted between nodes corresponds to chemical signals between neurons
  - Nodes modifying numerical signal corresponds to neurons firing rate

CS486/686 Lecture Slides (c) 2005 P. Pasupat

6

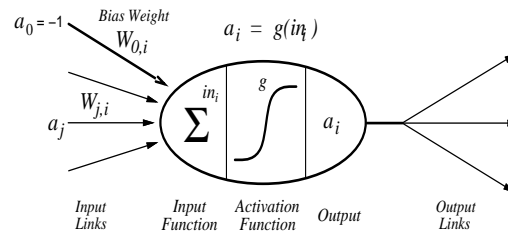
## ANN Unit

- For each unit  $i$ :
- **Weights:  $W_{ji}$** 
  - Strength of the link from unit  $j$  to unit  $i$
  - Input signals  $a_j$  weighted by  $W_{ji}$  and linearly combined:  $in_i = \sum_j W_{ji} a_j$
- **Activation function:  $g$** 
  - Numerical signal produced:  $a_i = g(in_i)$

CS486/686 Lecture Slides (c) 2005 P. Pasupat

7

## ANN Unit



CS486/686 Lecture Slides (c) 2005 P. Pasupat

8

## Activation Function

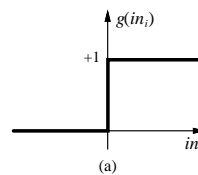
- Should be nonlinear
  - Otherwise network is just a linear function
- Often chosen to mimic firing in neurons
  - Unit should be "active" (output near 1) when fed with the "right" inputs
  - Unit should be "inactive" (output near 0) when fed with the "wrong" inputs

CS486/686 Lecture Slides (c) 2005 P. Pasupat

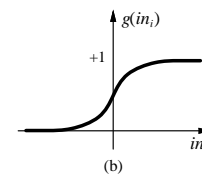
9

## Common Activation Functions

### Threshold



### Sigmoid



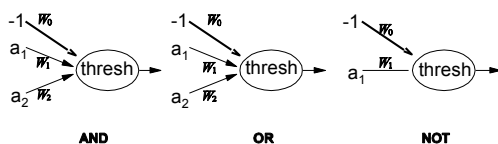
$$g(x) = 1/(1+e^{-x})$$

CS486/686 Lecture Slides (c) 2005 P. Pasupat

10

## Logic Gates

- McCulloch and Pitts (1943)
  - Design ANNs to represent Boolean fns
- **What should be the weights of the following units to code AND, OR, NOT ?**



CS486/686 Lecture Slides (c) 2005 P. Pasupat

11

## Network Structures

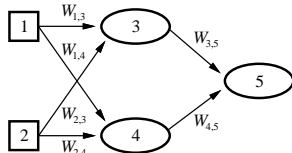
- **Feed-forward network**
  - Directed **acyclic** graph
  - No internal state
  - Simply computes outputs from inputs
- **Recurrent network**
  - Directed **cyclic** graph
  - Dynamical system with internal states
  - Can memorize information

CS486/686 Lecture Slides (c) 2005 P. Pasupat

12

## Feed-forward network

- Simple network with two inputs, one hidden layer of two units, one output unit



$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4)$$

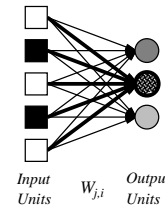
$$= g(W_{3,5}g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2))$$

CS486/686 Lecture Slides (c) 2005 P. Pasupat

13

## Perceptron

- Single layer feed-forward network



CS486/686 Lecture Slides (c) 2005 P. Pasupat

14

## Supervised Learning

- Given list of <input,output> pairs
- Train feed-forward ANN
  - To compute proper outputs when fed with inputs
  - Consists of adjusting weights  $W_{ji}$
- Simple learning algorithm for threshold perceptrons

CS486/686 Lecture Slides (c) 2005 P. Pasupat

15

## Threshold Perceptron Learning

- Learning is done separately for each unit
  - Since units do not share weights
- Perceptron learning for unit  $i$ :
  - For each <inputs,output> pair do:
    - Case 1: correct output produced
      - $\forall_j W_{ji} \leftarrow W_{ji}$
    - Case 2: output produced is 0 instead of 1
      - $\forall_j W_{ji} \leftarrow W_{ji} + a_j$
    - Case 3: output produced is 1 instead of 0
      - $\forall_j W_{ji} \leftarrow W_{ji} - a_j$
  - Until correct output for all training instances

CS486/686 Lecture Slides (c) 2005 P. Pasupat

16

## Threshold Perceptron Learning

- Dot products:  $a \bullet a \geq 0$  and  $-a \bullet a \leq 0$
- Perceptron computes
  - 1 when  $a \bullet W = \sum_j a_j W_{ji} > 0$
  - 0 when  $a \bullet W = \sum_j a_j W_{ji} < 0$
- If output should be 1 instead of 0 then
  - $W \leftarrow W + a$  since  $a \bullet (W + a) \geq a \bullet W$
- If output should be 0 instead of 1 then
  - $W \leftarrow W - a$  since  $a \bullet (W - a) \leq a \bullet W$

CS486/686 Lecture Slides (c) 2005 P. Pasupat

17

## Threshold Perceptron Hypothesis Space

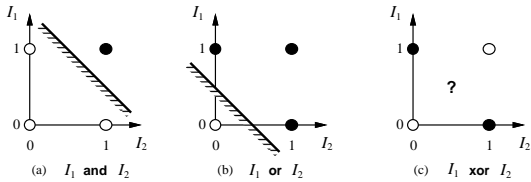
- Hypothesis space  $h_W$ :
  - All binary classifications with param.  $W$  s.t.
    - $a \bullet W > 0 \rightarrow 1$
    - $a \bullet W < 0 \rightarrow 0$
- Since  $a \bullet W$  is linear in  $W$ , perceptron is called a **linear separator**

CS486/686 Lecture Slides (c) 2005 P. Pasupat

18

## Threshold Perceptron Hypothesis Space

- Are all Boolean gates linearly separable?

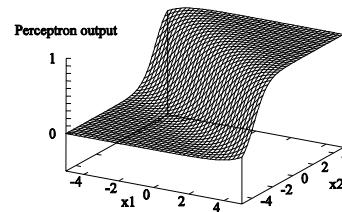


CS486/686 Lecture Slides (c) 2005 P. Pasupat

19

## Sigmoid Perceptron

- Represent "soft" linear separators



CS486/686 Lecture Slides (c) 2005 P. Pasupat

20

## Sigmoid Perceptron Learning

- Formulate learning as an optimization search in weight space
  - Since  $g$  differentiable, use gradient descent
- Minimize squared error:
  - $E = 0.5 \text{Err}^2 = 0.5 (y - h_W(\mathbf{x}))^2$ 
    - $\mathbf{x}$ : input
    - $y$ : target output
    - $h_W(\mathbf{x})$ : computed output

CS486/686 Lecture Slides (c) 2005 P. Pasupat

21

## Perceptron Error Gradient

- $E = 0.5 \text{Err}^2 = 0.5 (y - h_W(\mathbf{x}))^2$
- $\partial E / \partial W_j = \text{Err} \times \partial \text{Err} / \partial W_j$ 

$$= \text{Err} \times \partial (y - g(\sum_j W_j x_j))$$

$$= -\text{Err} \times g'(\sum_j W_j x_j) \times x_j$$
- When  $g$  is sigmoid fn, then  $g' = g(1-g)$

CS486/686 Lecture Slides (c) 2005 P. Pasupat

22

## Perceptron Learning Algorithm

- Perceptron-Learning(examples, network)
  - Repeat
    - For each  $e$  in examples do
      - $\text{in} \leftarrow \sum_j W_j x_j[e]$
      - $\text{Err} \leftarrow y[e] - g(\text{in})$
      - $W_j \leftarrow W_j + \alpha \times \text{Err} \times g'(\text{in}) \times x_j[e]$
  - Until some stopping criteria satisfied
  - Return learnt network
- N.B.  $\alpha$  is a learning rate corresponding to the step size in gradient descent

CS486/686 Lecture Slides (c) 2005 P. Pasupat

23

## Multilayer Feed-forward Neural Networks

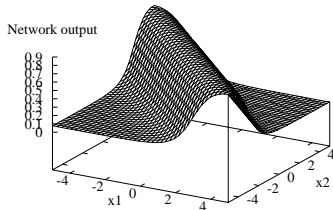
- Perceptron can only represent (soft) linear separators
  - Because single layer
- With multiple layers, what fns can be represented?
  - Virtually any function!

CS486/686 Lecture Slides (c) 2005 P. Pasupat

24

## Multilayer Networks

- Adding two sigmoid units with parallel but opposite “cliffs” produces a ridge

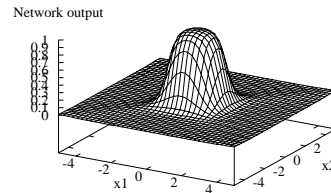


CS486/686 Lecture Slides (c) 2005 P. Prasant

25

## Multilayer Networks

- Adding two intersecting ridges (and thresholding) produces a bump



CS486/686 Lecture Slides (c) 2005 P. Prasant

26

## Multilayer Networks

- By tiling bumps of various heights together, we can approximate any function
- Training algorithm:
  - Back-propagation
  - Essentially gradient performed by propagating errors backward into the network
  - See textbook for derivation

CS486/686 Lecture Slides (c) 2005 P. Prasant

27

## Neural Net Applications

- Neural nets can approximate any function, hence 1000's of applications
  - NETtalk for pronouncing English text
  - Character recognition
  - Paint-quality inspection
  - Vision-based autonomous driving
  - Etc.

CS486/686 Lecture Slides (c) 2005 P. Prasant

28

## Neural Net Drawbacks

- Common problems:
  - How should we interpret units?
  - How many layers and units should a network have?
  - How to avoid local optimum while training with gradient descent?

CS486/686 Lecture Slides (c) 2005 P. Prasant

29

## Next Class

- Next Class:
  - Ensemble learning
  - Russell and Norvig Sect. 18.4

CS486/686 Lecture Slides (c) 2005 P. Prasant

30