CS434a/541a: Pattern Recognition
Prof. Olga Veksler

Lecture 12

Announcements

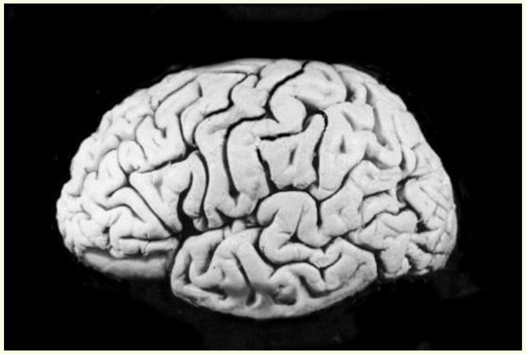
- Assignment 4 posted on the Web, due Dec. 1
- Course evaluations will be conducted Dec. 1 at the end of the lecture

Today

- Multilayer Neural Networks
  - Inspiration from Biology
  - History
  - Perceptron
  - Multilayer perceptron

Brain vs. Computer

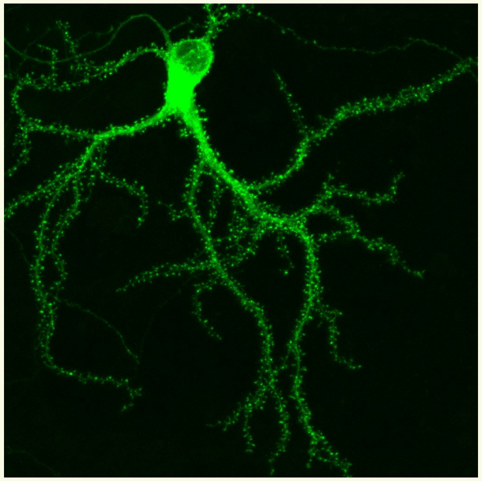
- Designed to solve logic and arithmetic problems
- Can solve a gazillion arithmetic and logic problems in an hour
- absolute precision
- Usually one very fast procesor
- high reliability

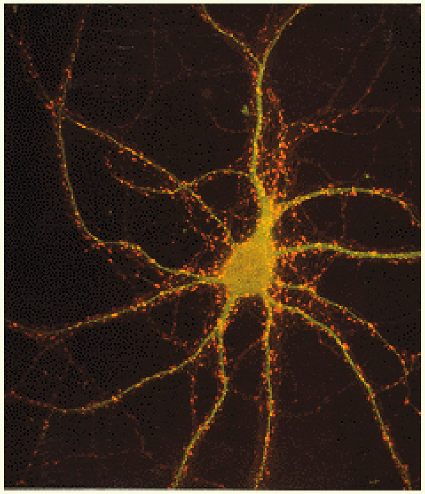


- Evolved (in a large part) for pattern recognition
- Can solve a gazillion of PR problems in an hour
- Huge number of parallel but relatively slow and unreliable processors
- not perfectly precise
- not perfectly reliable

Seek an inspiration from human brain for PR?

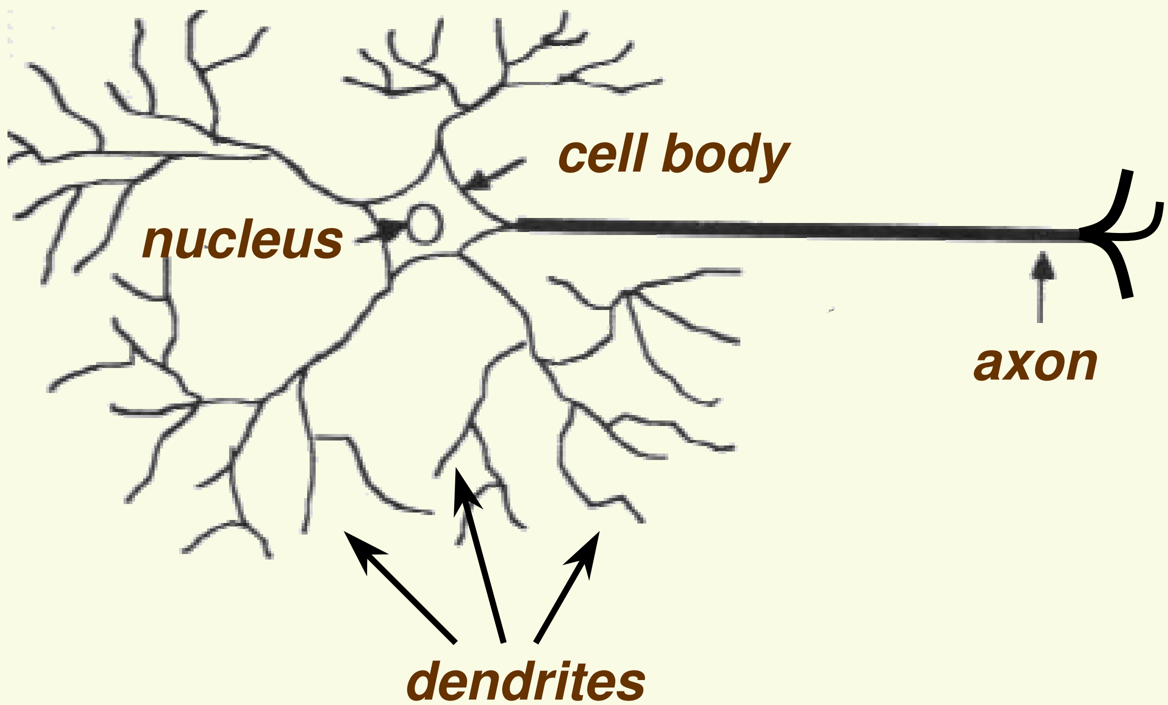
Neuron: Basic Brain Processor





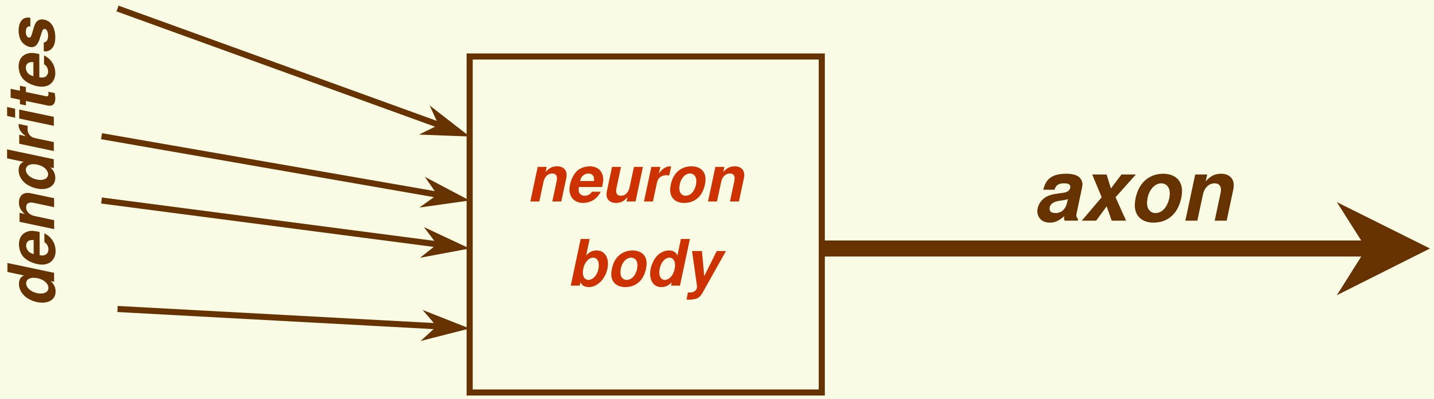
- Neurons are nerve cells that transmit signals to and from brains at the speed of around 200mph
- Each neuron cell communicates to anywhere from 1000 to 10,000 other neurons, muscle cells, glands, so on
- Have around $\boldsymbol{10^{10}}$ neurons in our brain (network of neurons)
- Most neurons a person is ever going to have are already present at birth

Neuron: Basic Brain Processor



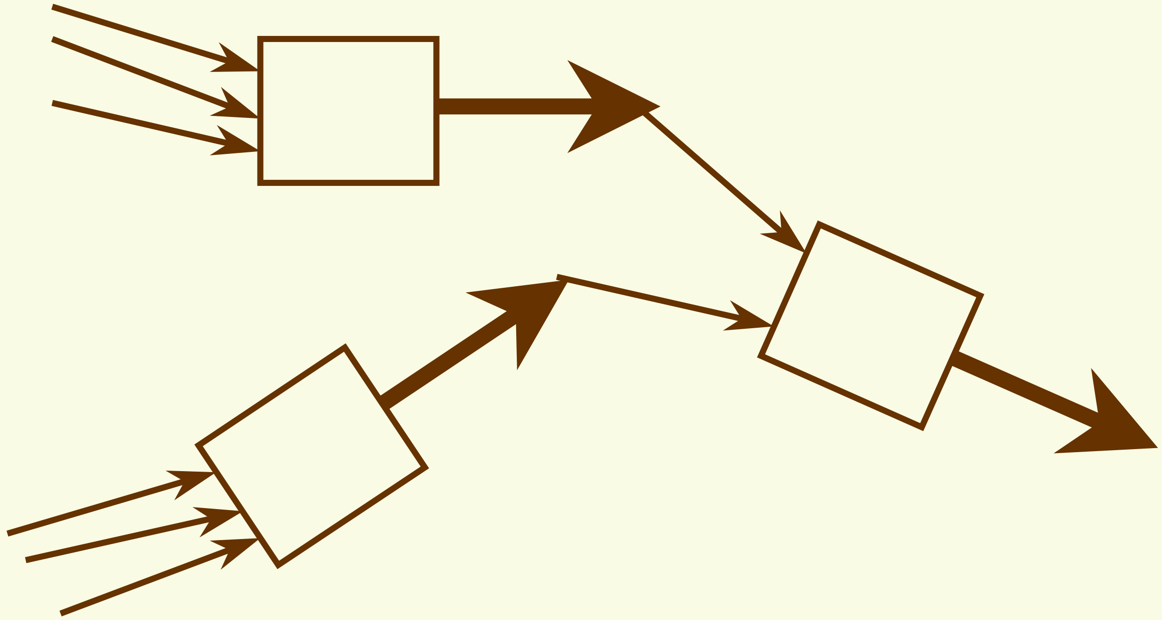
- Main components of a neuron
    - ***Cell body*** which holds DNA information in ***nucleus***
    - ***Dendrites*** may have thousands of dendrites, usually short
    - ***axon*** long structure, which splits in possibly thousands branches at the end. May be up to 1 meter long

Neuron in Action (*simplified*)



- ***Input*** : neuron collects signals from other neurons through dendrites, may have thousands of dendrites
- ***Processor:*** Signals are accumulated and processed by the cell body
- ***Output:*** If the strength of incoming signals is large enough, the cell body sends a signal (a spike of electrical activity) to the axon

Neural Network



ANN History: Birth

- 1943, famous paper by W. McCulloch (neurophysiologist) and W. Pitts (mathematician)
  - Using only math and algorithms, constructed a model of how neural network may work
  - Showed it is possible to construct any computable function with their network
  - Was it possible to make a model of thoughts of a human being?
  - Considered to be the birth of AI
- 1949, D. Hebb, introduced the first (purely pshychological) theory of learning
  - Brain learns at tasks through life, thereby it goes through tremendous changes
  - If two neurons fire together, they strengthen each other's responses and are likely to fire together in the future

ANN History: First Successes

- 1958, F. Rosenblatt,
  - perceptron, oldest neural network still in use today
  - Algorithm to train the perceptron network (training is still the most actively researched area today)
  - Built in hardware
  - Proved convergence in linearly separable case
- 1959, B. Widrow and M. Hoff
  - Madaline
  - First ANN applied to real problem (eliminate echoes in phone lines)
  - Still in commercial use

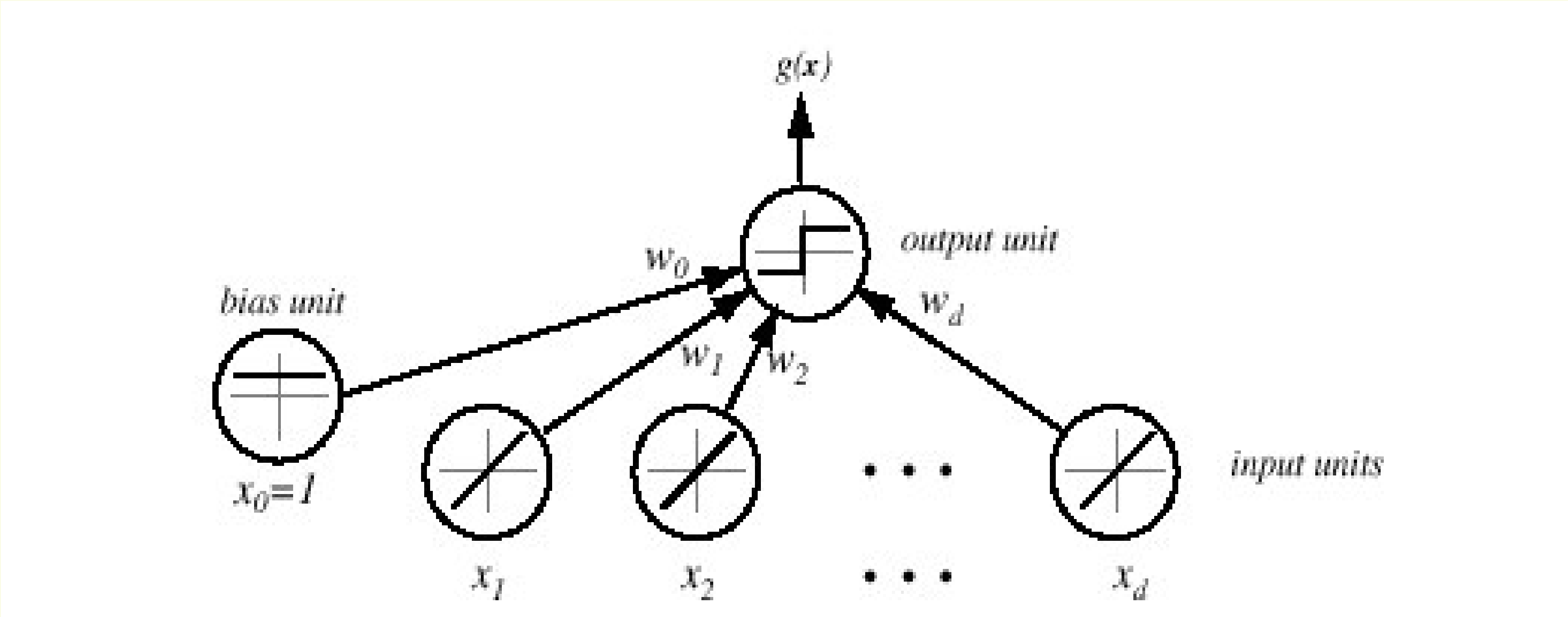
ANN History: Stagnation

- Early success lead to a lot of claims which were not fulfilled
- 1969, M. Minsky and S. Pappert
  - Book "Perceptrons"
  - Proved that perceptrons can learn only linearly separable classes
  - In particular cannot learn very simple XOR function
  - Conjectured that multilayer neural networks also limited by linearly separable functions
- No funding and almost no research (at least in North America) in 1970's as the result of 2 things above

ANN History: Revival

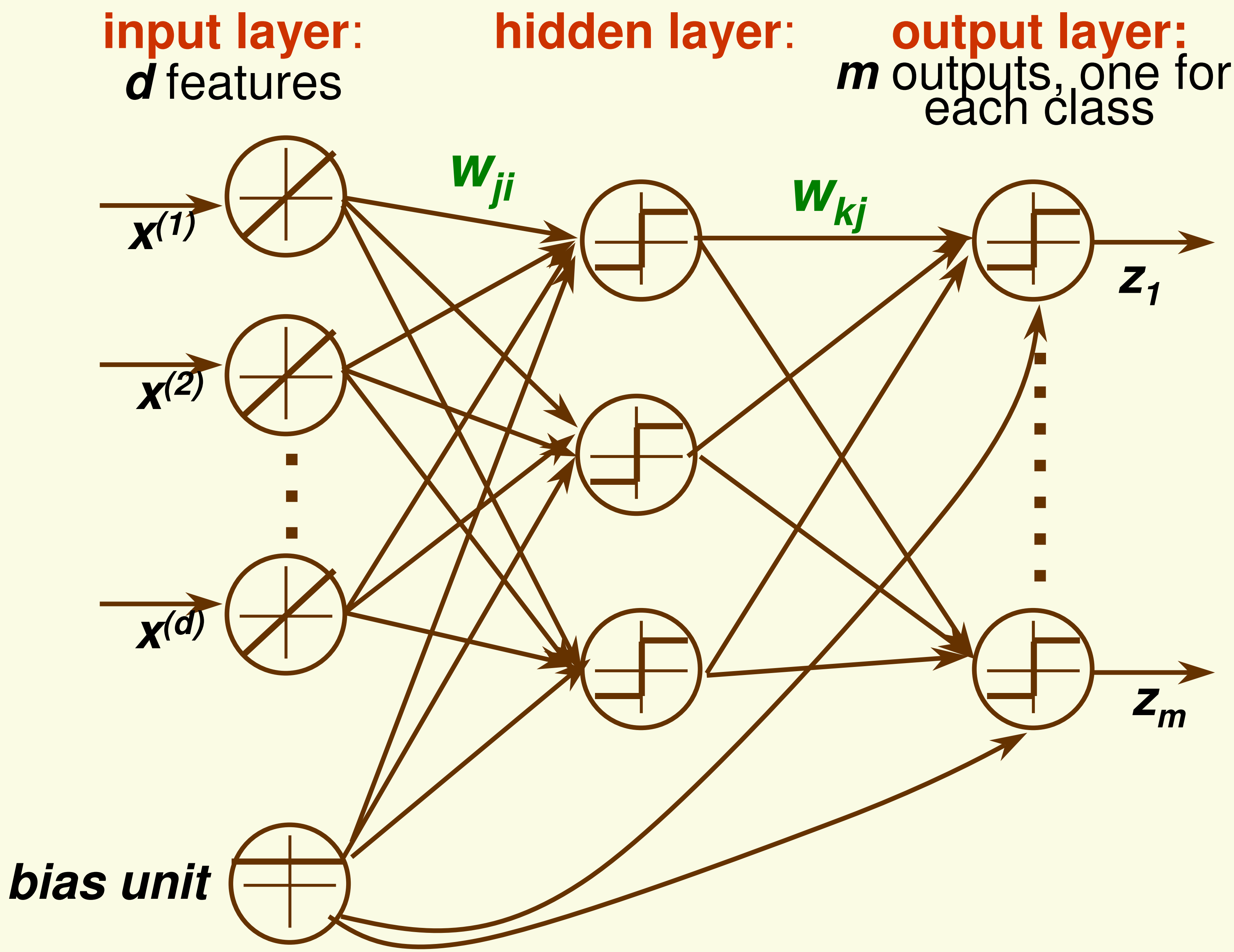
- Revival of ANN in 1980's
- 1982, J. Hopfield
  - New kind of networks (Hopfield's networks)
  - Bidirectional connections between neurons
  - Implements associative memory
- 1982 joint US-Japanese conference on ANN
  - US worries that it will stay behind
- Many examples of mulitlayer NN appear
- 1982, discovery of backpropagation algorithm
  - Allows a network to learn not linearly separable classes
  - Discovered independently by
    1. Y. Lecunn
    2. D. Parker
    3. Rumelhart, Hinton, Williams

ANN: Perceptron



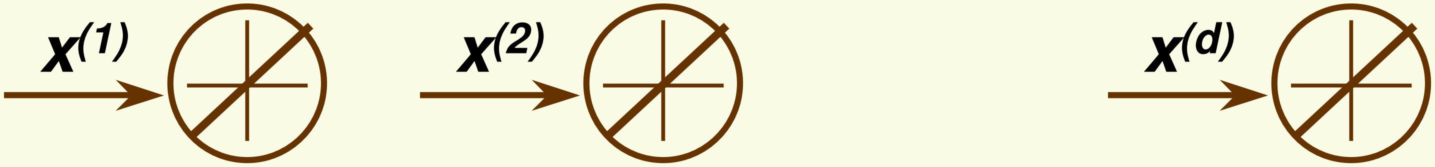
- Input and output layers
- $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{w_0}$
- Limitation: can learn only linearly separable classes

Multilayer Perceptron



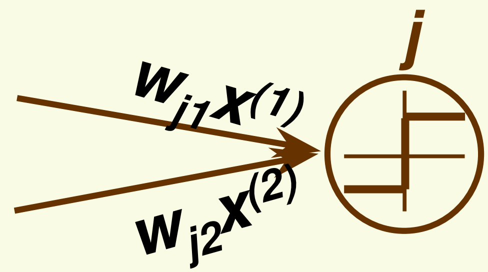
FeedForward Operation

1. Each sample is presented to the input layer



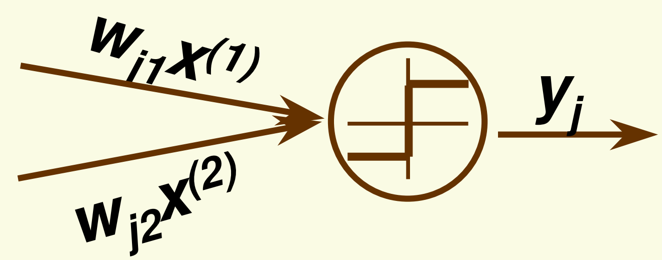
2. Each hidden unit ***j*** computes its net activation
   - dot product of input with incoming weights

$$net_j = \sum_{i=1}^{d} x^{(k)} w_{ji} + w_{j0}$$



3. Each hidden unit ***j*** emits a nonlinear function of its activation

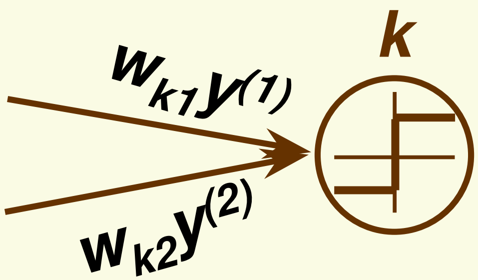
$$y_j = f(net_j) = \begin{cases} 1 & \text{if } net_j \geq 0 \\ -1 & \text{if } net_j < 0 \end{cases}$$



FeedForward Operation

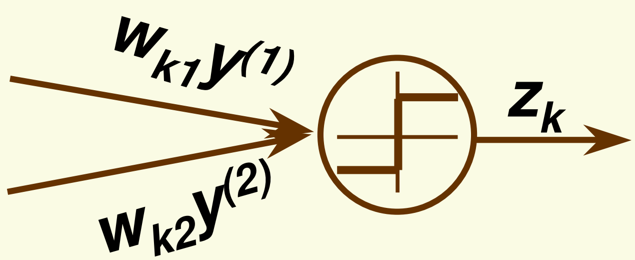
4. Each output unit ***k*** computes its net activation based on the hidden units
   - dot product of the hidden units with weights at this output unit

$$net_k = \sum_{j=1}^{N_h} y_j w_{kj} + w_{k0}$$



5. Each output unit ***k*** emits a nonlinear function of its activation

$$z_k = f(net_k) = \begin{cases} 1 & \text{if } net_k \geq 0 \\ -1 & \text{if } net_k < 0 \end{cases}$$



Discriminant Function

- We can gather all the terms in previous slides in the discriminant function for class ***k*** (the output of the ***k***th output unit)

$$g_k(x) = z_k$$

$$= f\left(\underbrace{\sum_{j=1}^{N_H} w_{kj} f\left(\overbrace{\sum_{i=1}^{d} w_{ji} x_i + w_{j0}}^{\text{activation at jth hidden unit}}\right) + w_{k0}}_{\text{activation at kth output unit}}\right)$$

Discriminant Function

$$g_k(\boldsymbol{x}) = f\left(\sum_{j=1}^{N_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

- Given samples $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ each of one of the $\boldsymbol{m}$ classes
- Suppose for each sample $\boldsymbol{x}$, we wish

$$g_k(\boldsymbol{x}) = \begin{cases} 1 & \text{if } x \text{ is of class } k \\ 0 & \text{otherwise} \end{cases}$$

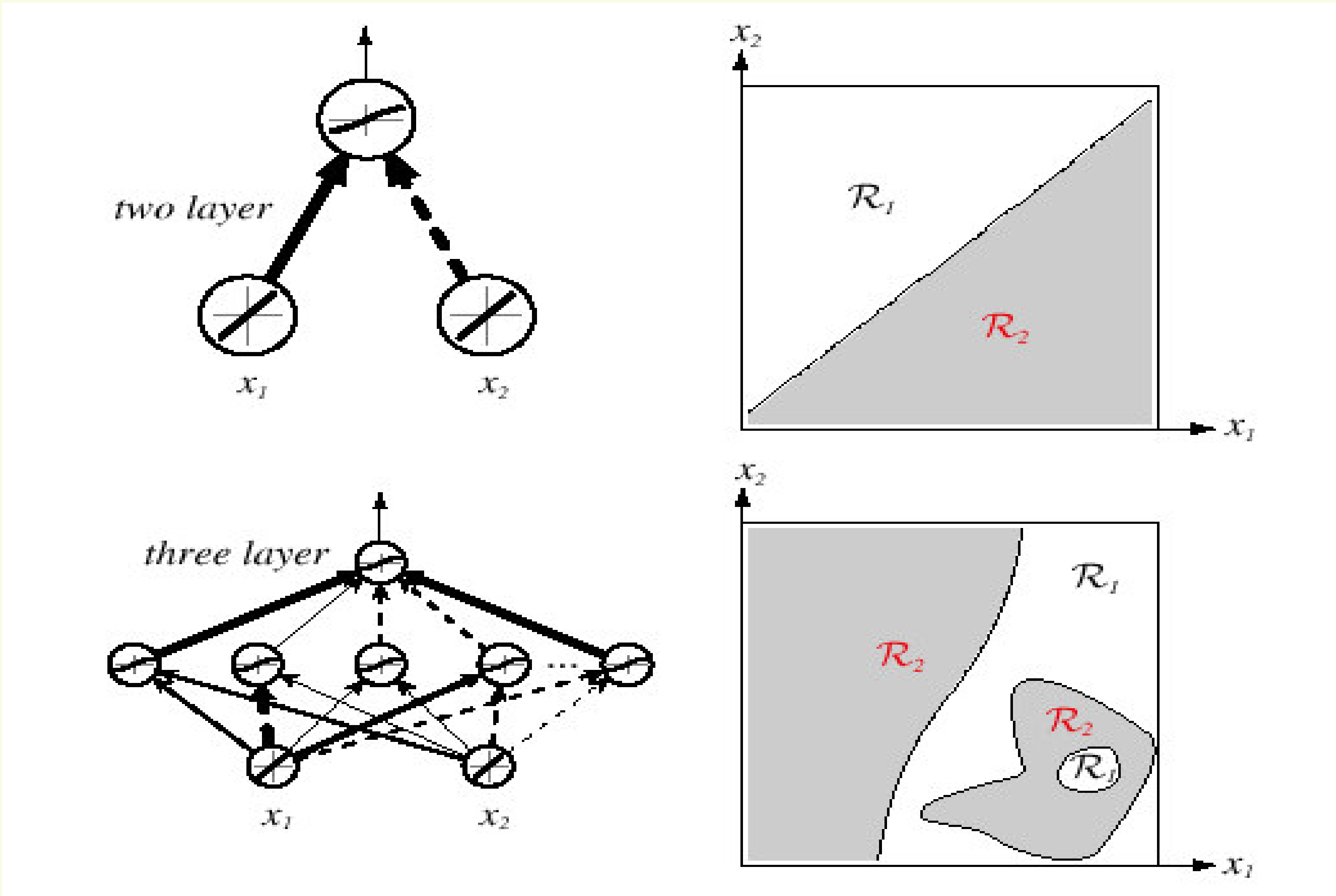
- The goal is to learn (to adjust) weights $\boldsymbol{w_{kj}}$ and $\boldsymbol{w_{ji}}$ to achieve the desired $\boldsymbol{g_k}(\boldsymbol{x})$ for all $\boldsymbol{k}$

Expressive Power of MNN

- It can be shown that every **continuous** function from input to output can be implemented with enough hidden units, 1 hidden layer, and proper nonlinear activation functions

- This is more of theoretical than practical interest
  - The proof is not constructive (does not tell us exactly how to construct the MNN)
  - Even if it were constructive, would be of no use since we do not know the desired function anyway, our goal is to learn it through the samples
  - But this result does give us confidence that we are on the right track
    - MNN is general enough to construct the correct decision boundaries, unlike the Perceptron

Discriminant Function



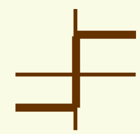
MNN

- Can vary
  - number of hidden layers
  - Nonlinear activation function
    - Can use different function for hidden and output layers
    - Can use different function at each hidden and output node

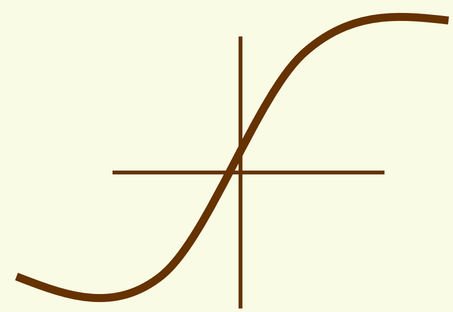
MNN Activation function

- Must be nonlinear for expressive power larger than that of perceptron
  - If use linear activation function, can only deal with linearly separable classes
- In previous example, used discontinuous activation function

$$f(net_k) = \begin{cases} 1 & \text{if } net_k \geq 0 \\ -1 & \text{if } net_k < 0 \end{cases}$$



- We will use gradient descent for learning, so we need to use continuous activation function



Next Time

- We will learn how to train a MNN using back propagation algorithm