BIOSYSTEMS II: NEUROSCIENCES

2015 Spring Semester

Lecture 30

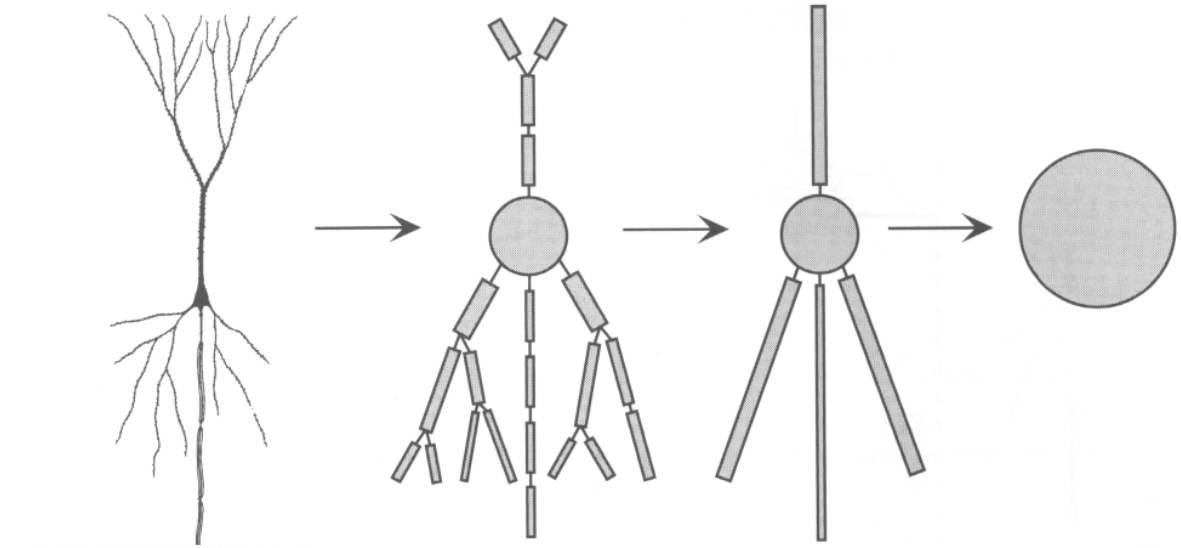
Kechen Zhang

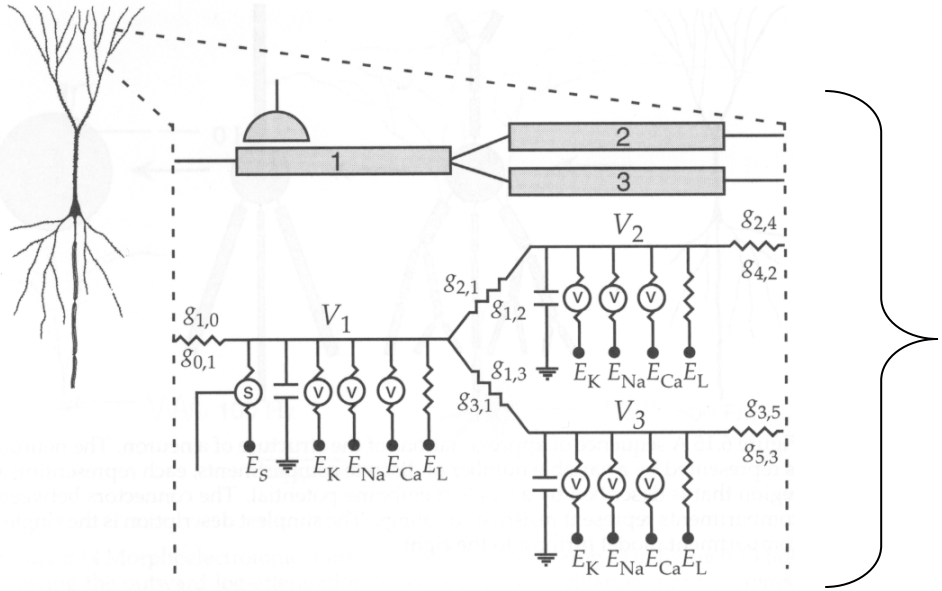
4/10/2015

Commonly Used single Neuron Models

- Compartment models
- Integrate-and-fire models
- Stochastic models
- Firing rate models

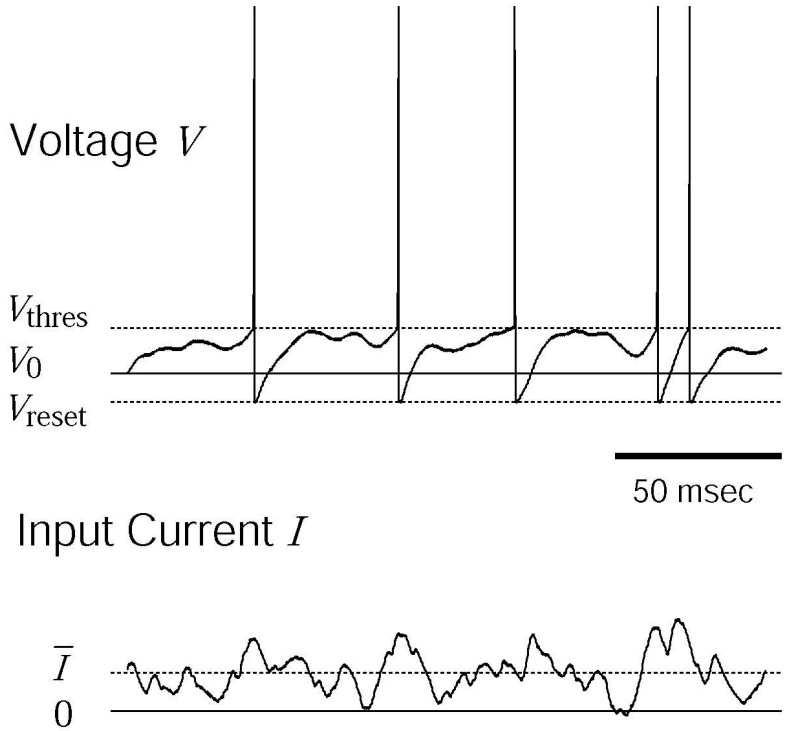
Compartment model of various complexity





Details of 3 compartments

Integrate-and-Fire Model

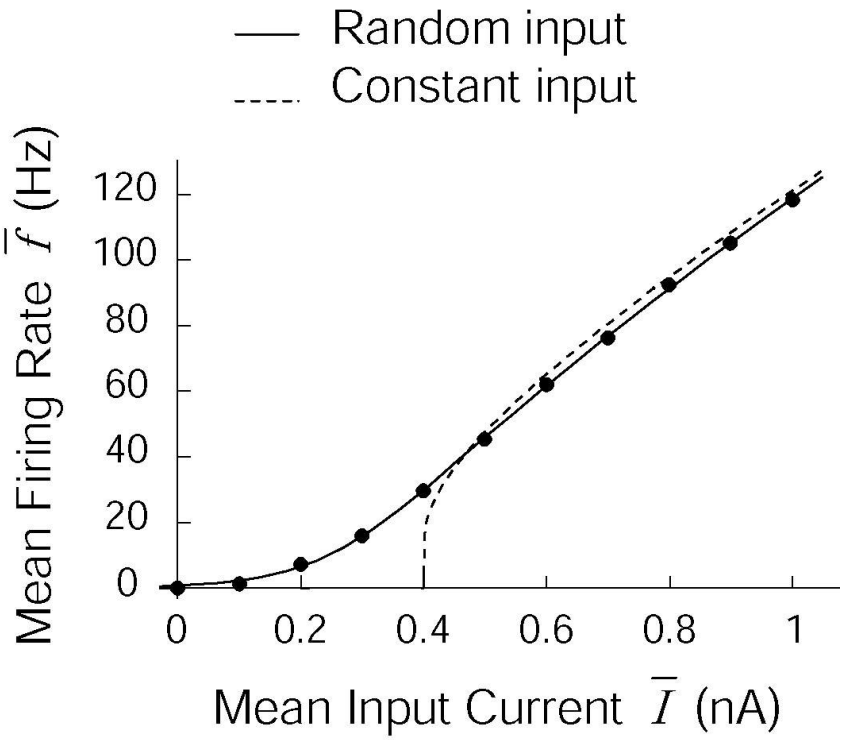


$$C\frac{dV}{dt} = -\frac{(V - V_0)}{R} + I$$

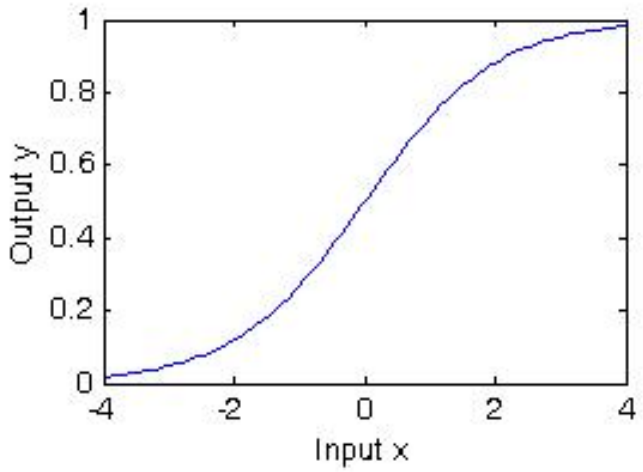
Generate a spike whenever V reaches threshold V_{thres}, then immediately reset it to V_{reset}

V - voltage
I - input current
C - capacitance
R - resistance
V_0 - resting potential

Input-output relation (gain function) of an integrate-and-fire neuron



Examples of gain functions

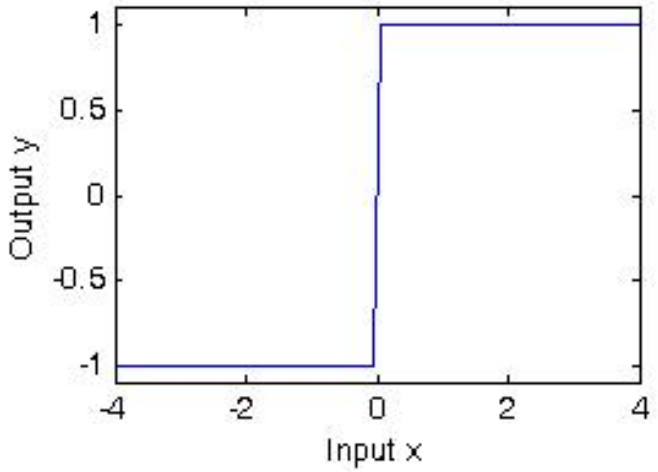


Logistic or sigmoid function:

$$y = \frac{1}{1+\exp(-x)}$$

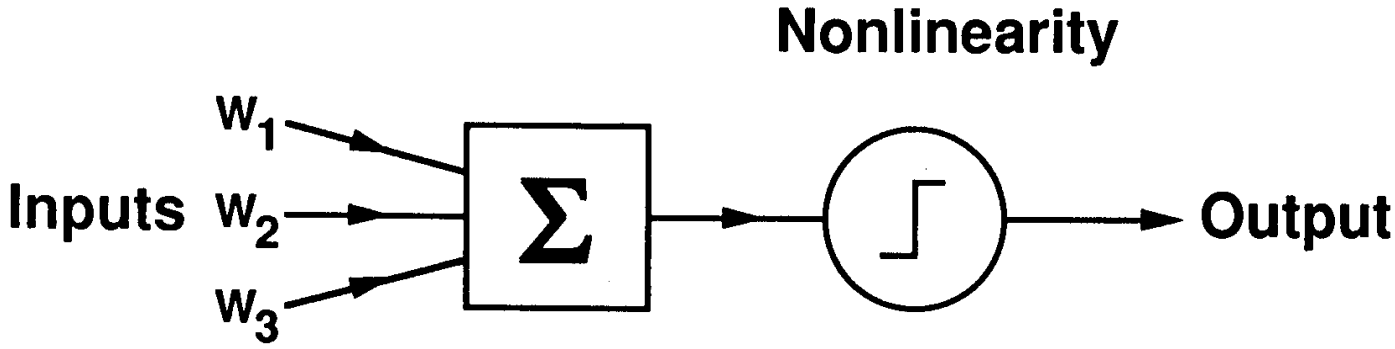
More general form:

$$y = \frac{1}{1+\exp\left(-k\left(x-c\right)\right)}$$



$$y = \text{sign}(x)$$

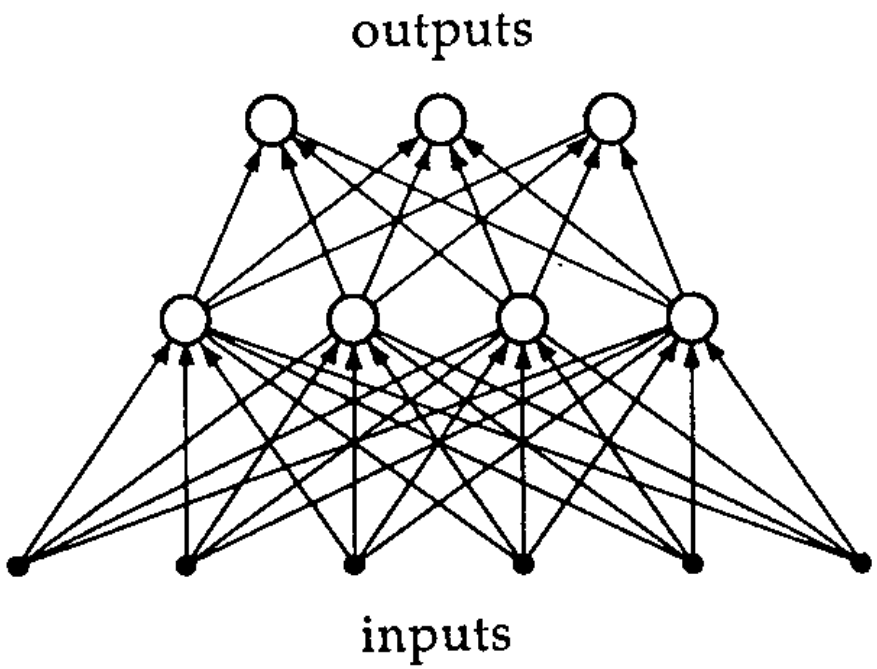
Firing Rate Model of a Neuron



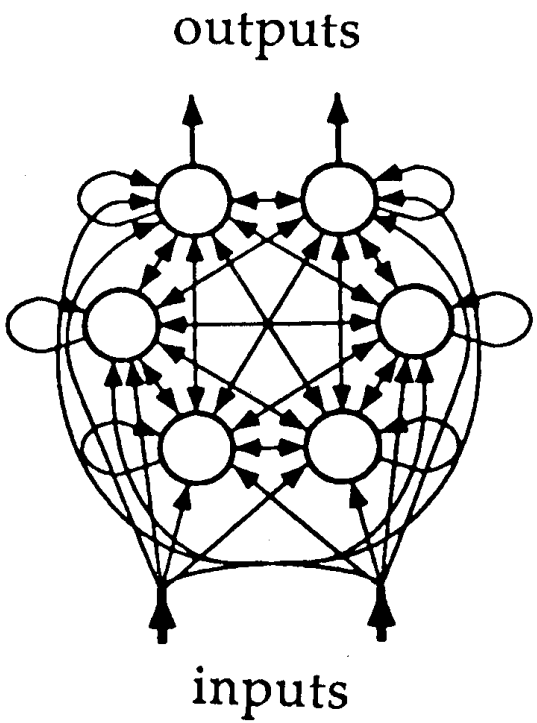
Inputs firing rates from other neurons: $x_1, x_2, x_3, \ldots$
Synaptic weights: $w_1, w_2, w_3, \ldots$
Nonlinear gain function (input-output relation): g
Threshold or bias: h
Output firing rate: $$y = g\left(\sum_i w_i x_i - h \right)$$

Feedforward vs. Recurrent Networks

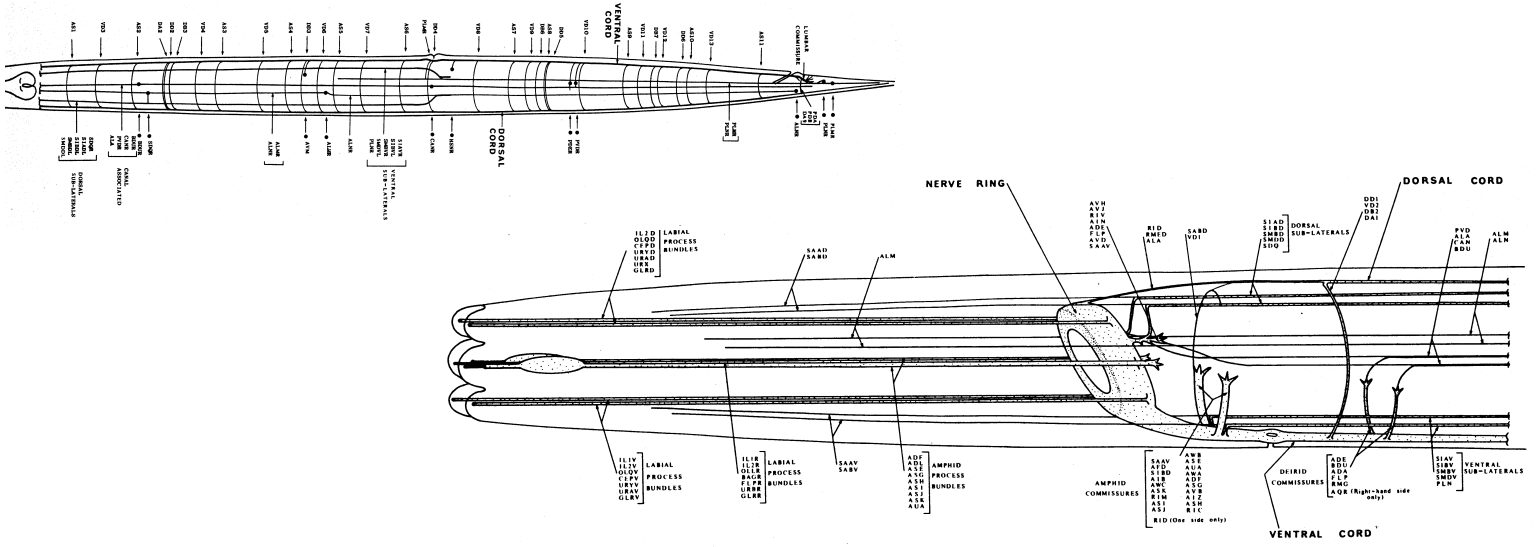
Feedforward

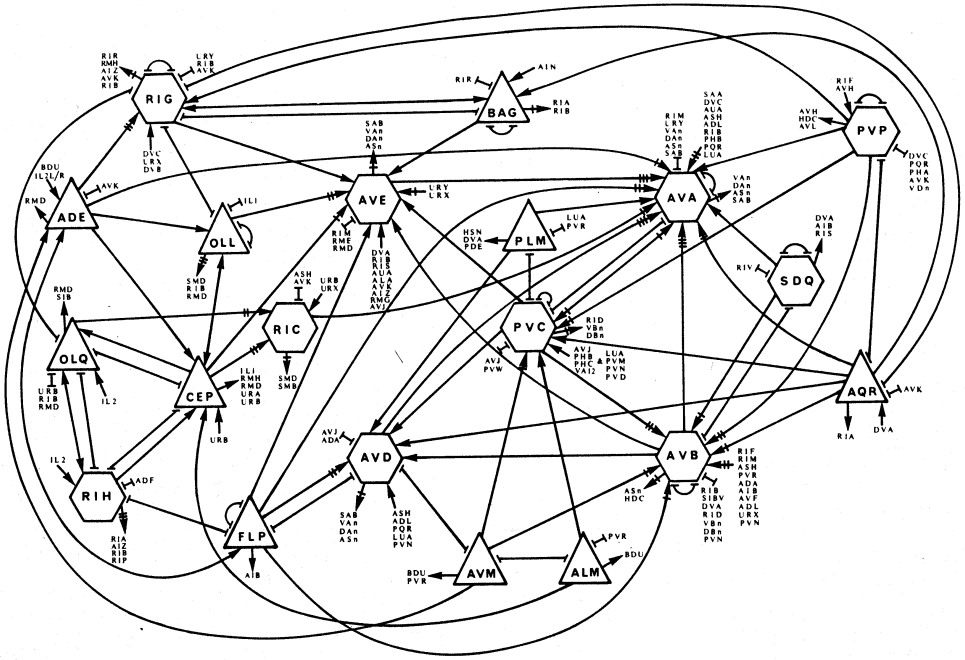


Recurrent



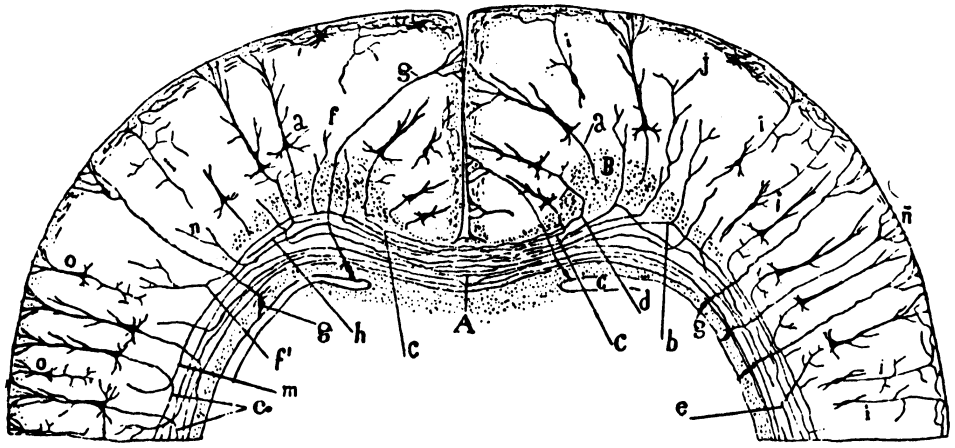
An Example of Biological Neural Networks: C elegans



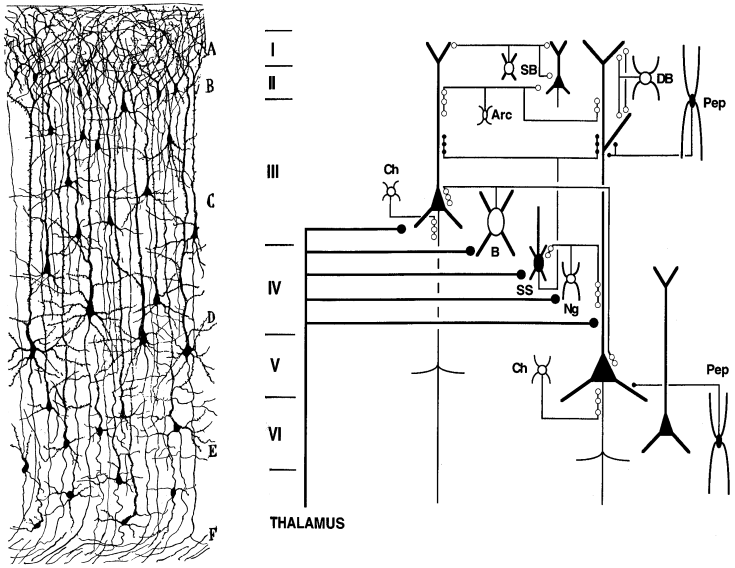


- All synaptic connections among 302 neurons are known and consistent from animal to animal
- A feedforward network can describe only part of a larger recurrent network

An Example of Biological Neural Networks: Mammalian Neocortex



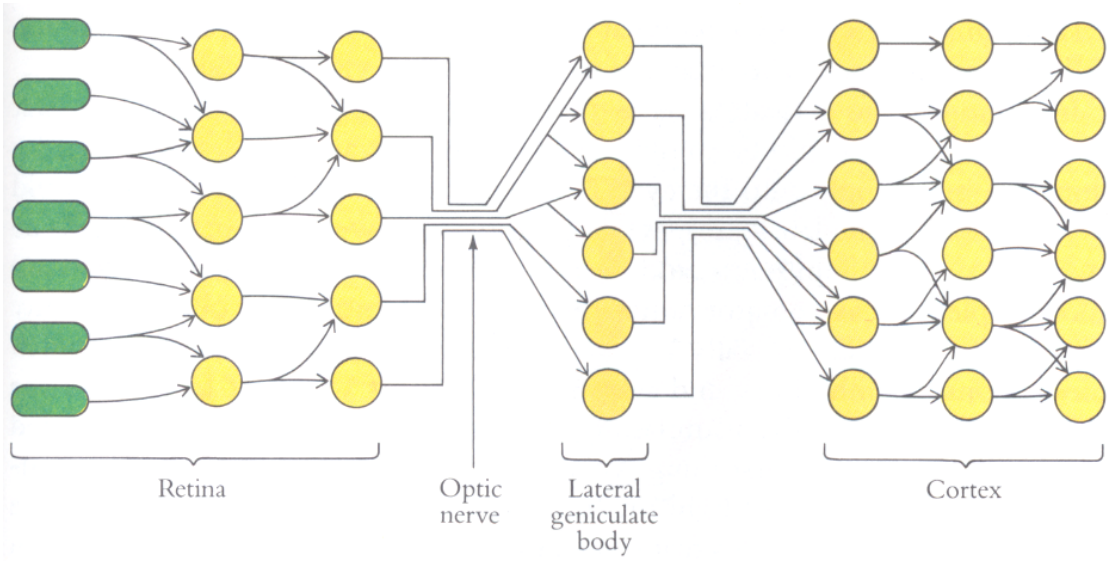
Neocortex of a kitten (Cajal)



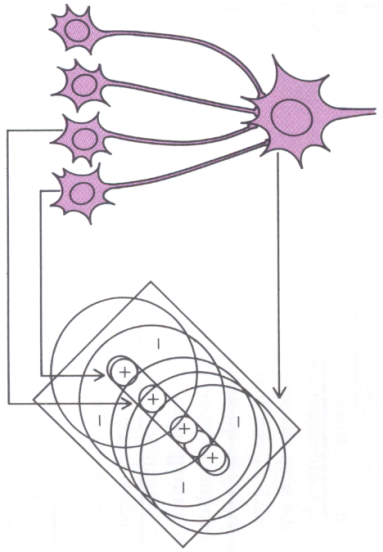
Local architecture

Over 80% of brain volume in humans are occupied by neocortex, and over 98% of axons in white matter interconnect different areas of the neocortex itself rather than connect with other parts of the brain. So the neocortical system is a huge recurrent network.

Example: A feedforward network in visual pathway

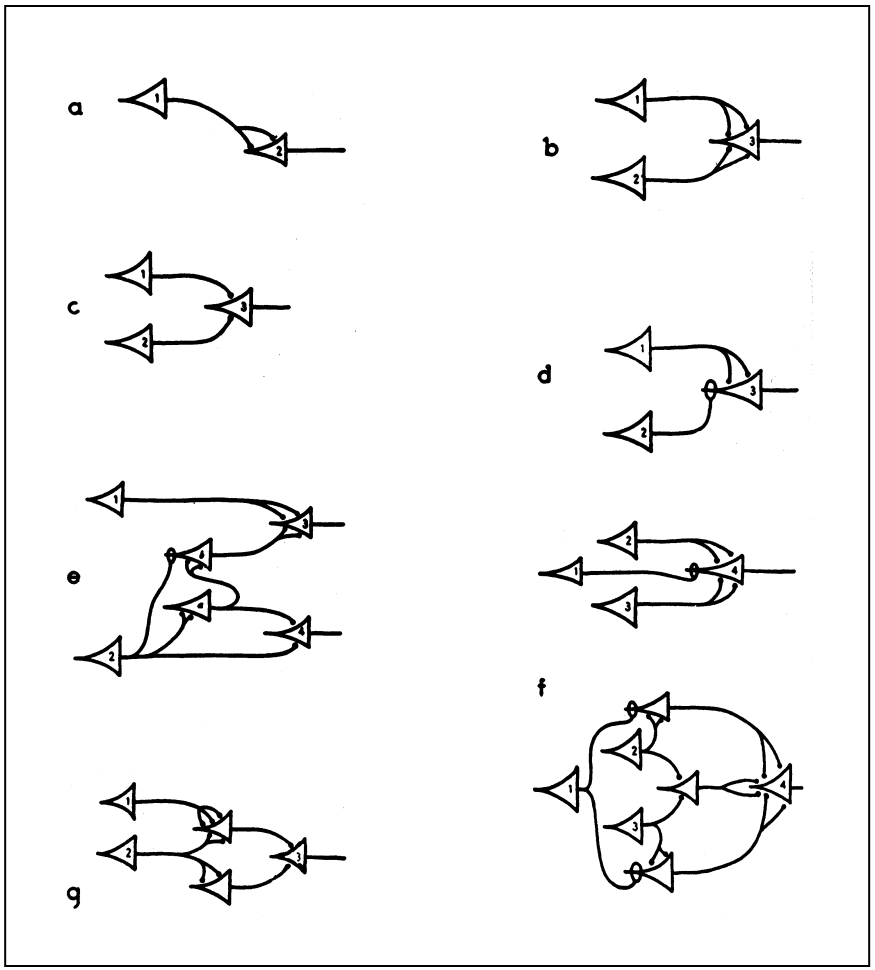


Oriented receptive field of a simple cell in visual cortex can be derived by linearly combining inputs from many smaller circular receptive fields in lateral geniculate body.



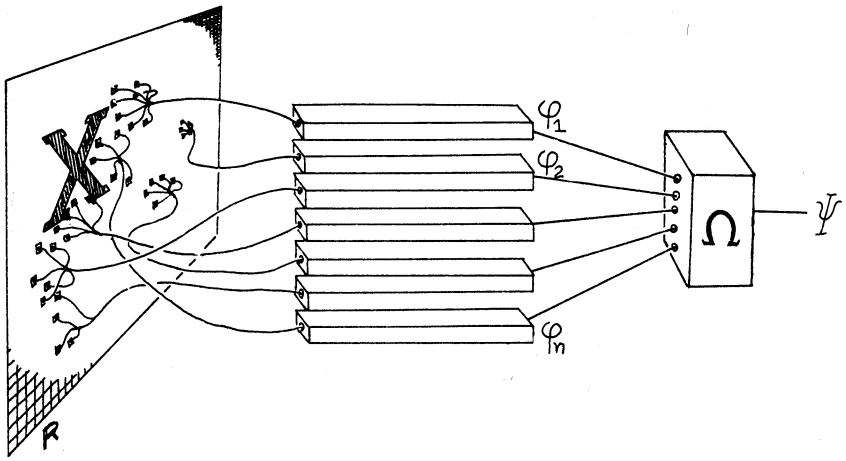
(Hubel)

McCulloch and Pitts Model (1943)



- Binary neurons with thresholds
- Synchronized update
- Memory as reverberant activity in a circle
- A network is powerful enough to do arbitrary logical calculations
- But, brain is not a digital computer

Perceptron



- Invented by Rosenblatt (1950's)
- Output is a weighted linear combination of the inputs
- Supervised learning: minimize error for each example by updating the weights
- Guaranteed convergence to a solution in finite steps if the classification problem is linearly separable

Perceptron

Output: $$y = \sum_i w_i x_i - h$$

where

input pattern: $x_1, x_2, x_3, \ldots$
weights: $w_1, w_2, w_3, \ldots$
threshold: h

Each input pattern is classified into one of two classes depending on whether $y > 0$ or $y < 0$.

Learning rule: $$\Delta w_i = k(Y - y)x_i$$

where

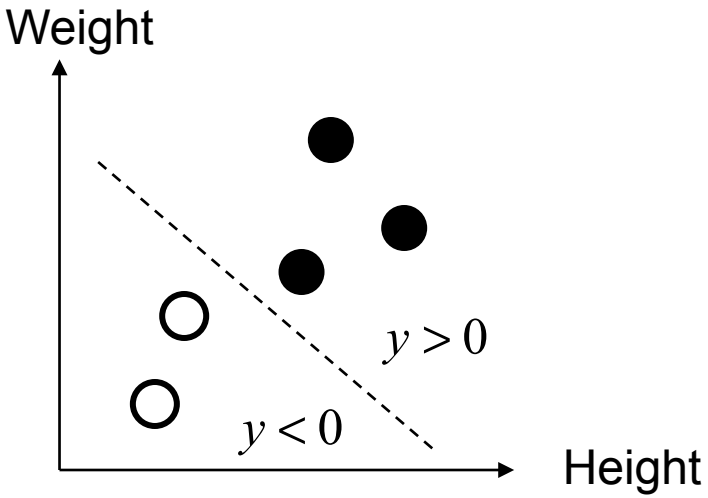
desired output: Y
learning rate: $k > 0$

Example of Linear Separability

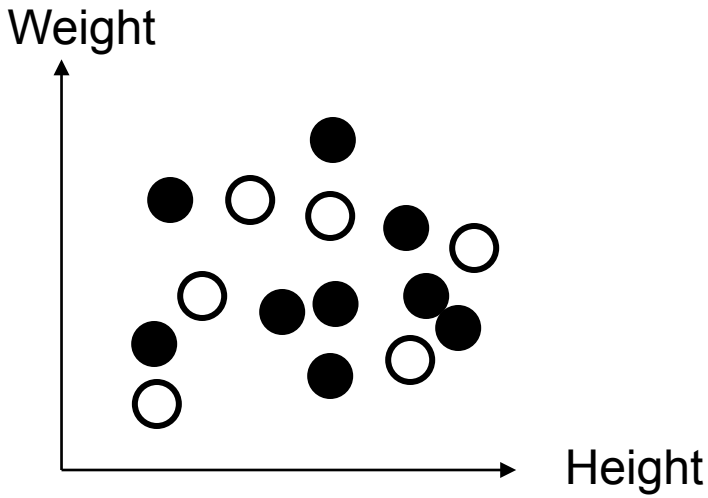
Name	Height (input x_1)	Weight (input x_2)	Gender (desired output Y)
John	5’ 10”	140	1
Mary	5’ 7”	110	0
Tom	6’ 2”	190	1

Ouput $y = w_1 x_1 + w_2 x_2 - h$ where $h = 0.5$ is threshold

Separable Case



Inseparable Case



Perceptron learning rule as gradient descent

Consider the square error for the desired output Y and the actual output y:

$$E = \frac{1}{2}(y - Y)^2$$

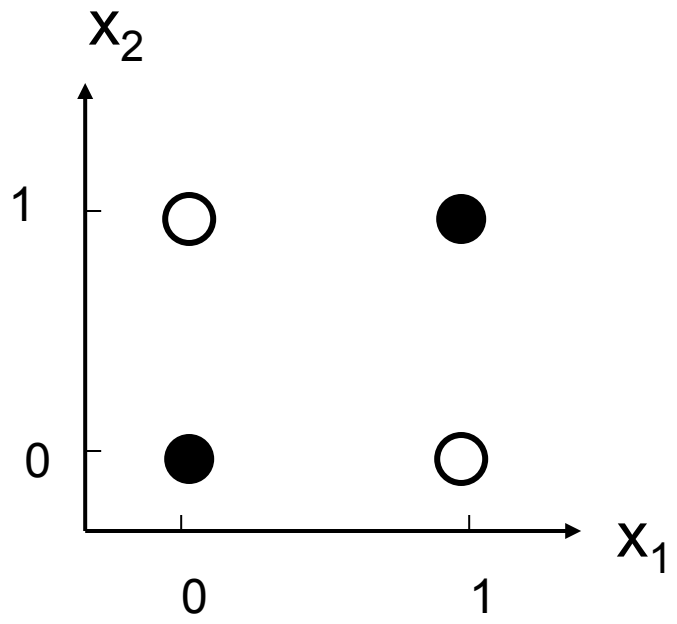
To minimize the error, we change the weights slightly along the direction of the steepest descent (gradient descent). This gives:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = -\eta (y - Y) \frac{\partial y}{\partial w_i} = -\eta (y - Y) x_i$$

which is exactly the perceptron learning rule.

Exclusive-OR (XOR) Problem

Input x_1	Input x_2	Output y
0	0	-1
0	1	1
1	0	1
1	1	-1



XOR: either A or B but not both.
This problem is linearly inseparable and cannot be learned by a perceptron.
In general, the decision boundary of a perceptron is given by $\sum_i w_i x_i = const$
which corresponds to a hyperplane in the input space.

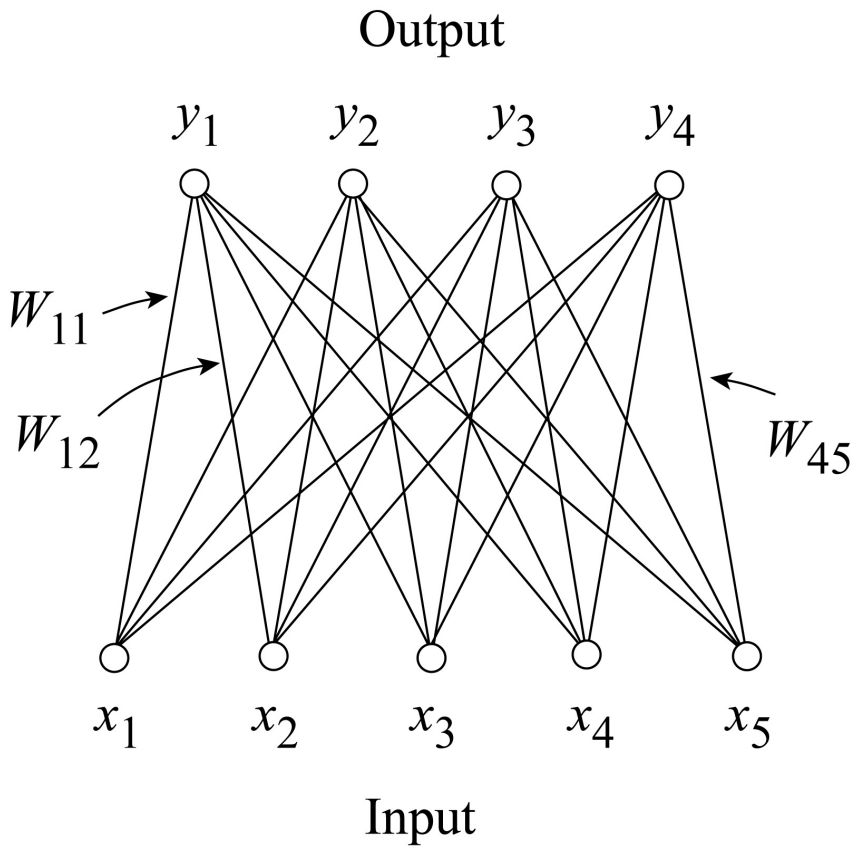
Linear perceptron with multiple inputs and outputs

Linear mapping:

$$y_i = \sum_j W_{ij} x_j$$

Vector-matrix form:

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$



Optimal linear mapping

Suppose multiple inputs are mapped linearlly to multiple outputs:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \mathbf{W} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$ or $\mathbf{y} = \mathbf{W}\mathbf{x}$, where $\mathbf{W}$ is an $m \times n$ weight matrix

Given all the inputs and corresponding desired outputs:

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots \\ \vdots & \vdots & \cdots \\ x_n^{(1)} & x_n^{(2)} & \cdots \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} y_1^{(1)} & y_1^{(2)} & \cdots \\ \vdots & \vdots & \cdots \\ y_m^{(1)} & y_m^{(2)} & \cdots \end{pmatrix},$$ find $\mathbf{W}$ such that $\mathbf{Y} = \mathbf{W}\mathbf{X}$.

The optimal weight matrix $\mathbf{W}$ that mininizes the square error of between the actual outputs $\mathbf{WX}$ and the desired outputs $\mathbf{Y}$ is:

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^{+}$$

where is $\mathbf{X}^{+}$ the pseudoinverse of $\mathbf{X}$.

What is a pseudoinverse?
(Moore-Penrose generalized inverse)

Let $\mathbf{A}$ be a full-rank $m \times n$ matrix.

Its pseudoinverse $\mathbf{A}^{\dagger}$ is a full-rank $n \times m$ matrix.

When matrix $\mathbf{A}$ has more rows than columns ($m > n$),

$$\mathbf{A}^{\dagger} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$$

so that $\mathbf{A}^{\dagger}\mathbf{A} = \mathbf{I}_n$ (identity).

When matrix $\mathbf{A}$ has more columns than rows ($n > m$),

$$\mathbf{A}^{\dagger} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}$$

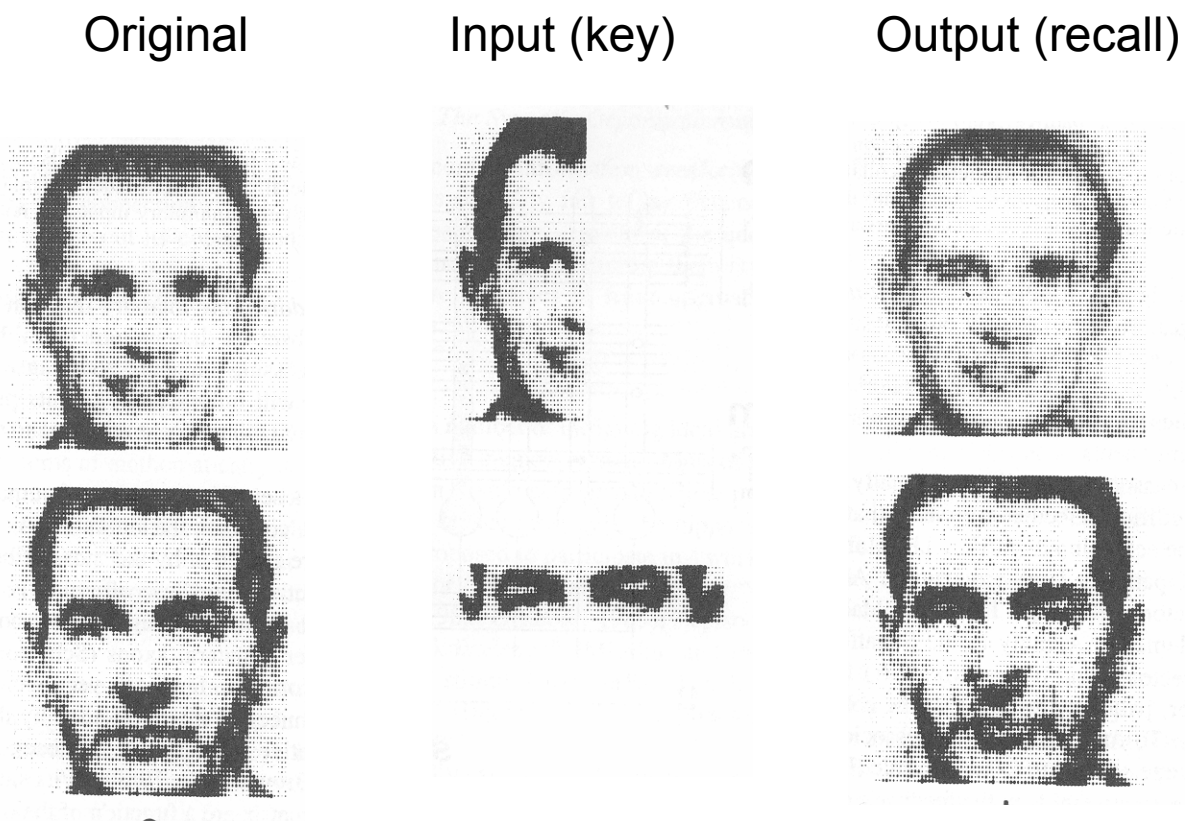
so that $\mathbf{A}\mathbf{A}^{\dagger} = \mathbf{I}_m$ (identity).

Example of linear mapping: autoassociative memory

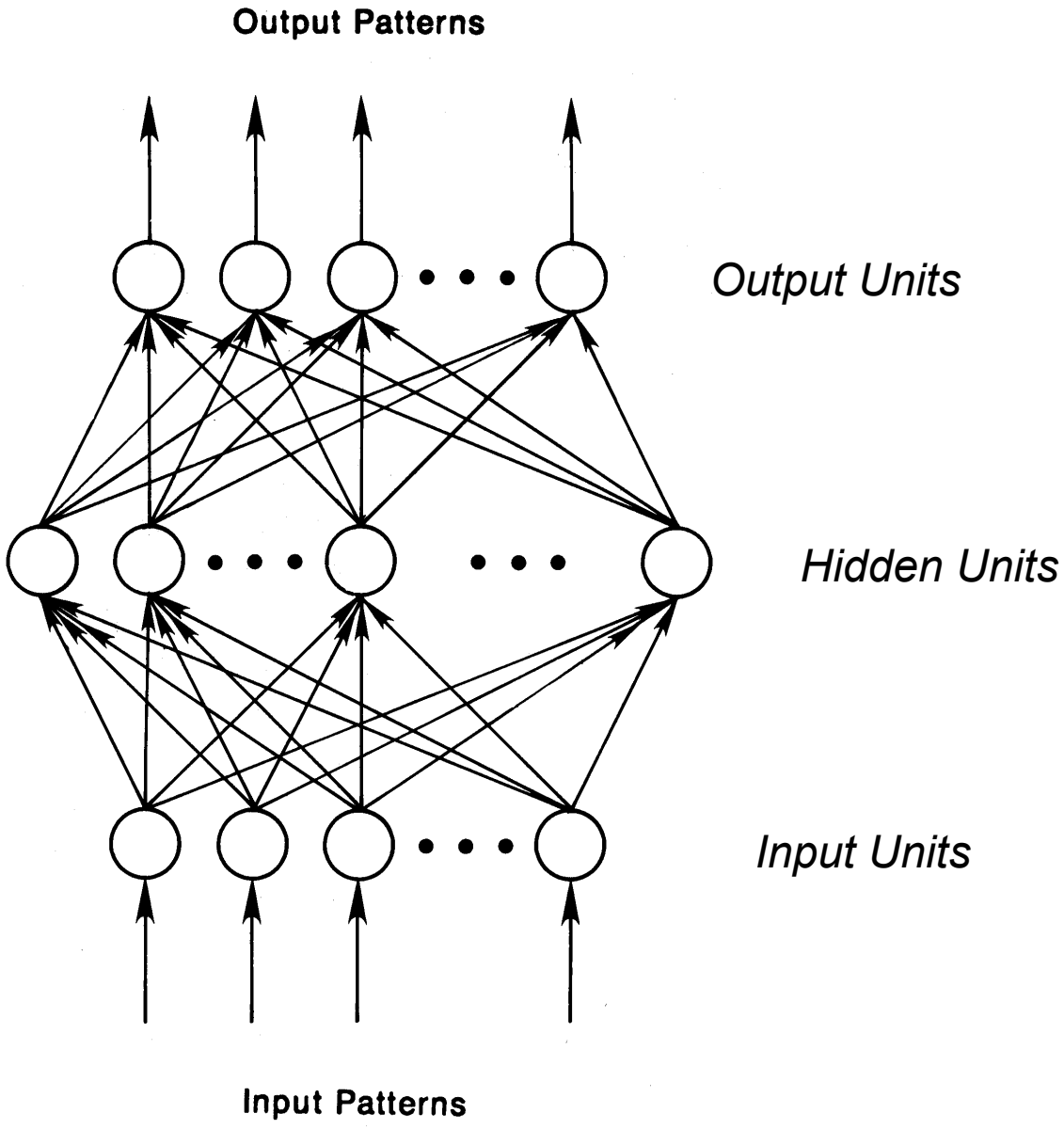
Image Data: $\mathbf{X}$

Desired autoassociation: $\mathbf{X} = \mathbf{WX}$

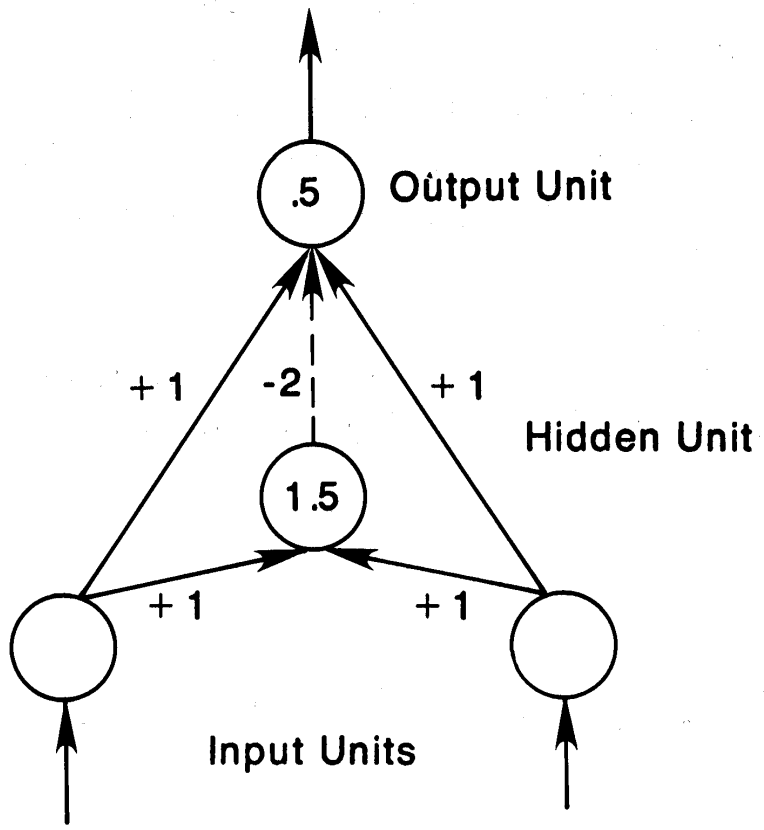
Solution: $\mathbf{W} = \mathbf{XX}^{\dagger}$



Multilayer Perceptron

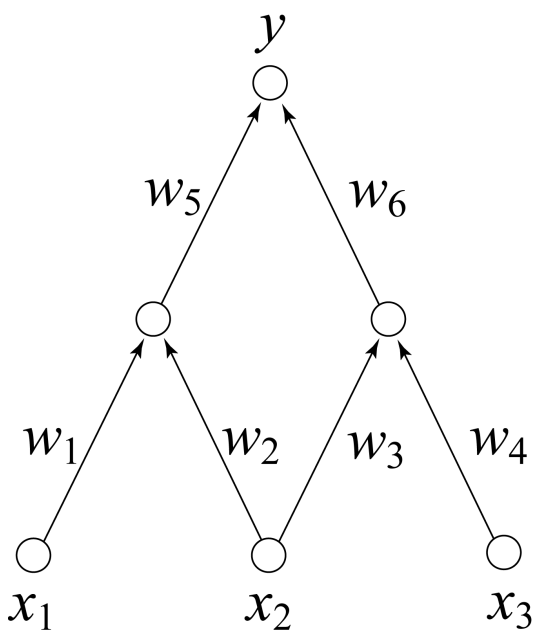


A Solution to the XOR Problem by Multilayer Perceptron



Multiple solutions can be found by training a network. Here is one solution. A number inside a circle is a threshold and a number by an arrow is a synaptic weight.

Example:
Learning in
multilayer perceptron



The output y of the multilayer perceptron depends on the inputs x_1, x_2, x_3 as well as the synaptic weights $w_1, w_2, \cdots, w_6$. Assuming that all other parameters such as the threshold and slope of the gain function are already fixed, we write the output as

$$y = F\left(x_1, x_2, x_3, w_1, w_2, w_3, w_4, w_5, w_6\right).$$

A learning rule for this perceptron can be derived using the gradient descent method to minimize the error function:

$$E = \frac{1}{2}\left(y - Y\right)^2,$$

where Y is the desired output. The result is

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta\left(Y - y\right)\frac{\partial F}{\partial w_i}.$$

where $\eta > 0$ is the learning rate.

Multilayer Perceptron Learning

- A multilayer feedforward network can approximate any given input-output relationship to arbitrary precision if there are enough hidden units.
- All the parameters (weights and biases) ofthe network can be learned by minimizing the error between the actual outputs of the network and the desired outputs (data of input-output pairs). The first algorithm for error minimization, the back-propagation algorithm, is essentially a gradient descent method (Rumelhart, Hinton and McClelland 1980's).
- Unlike in a single layer perceptron, the learning rule here is not local in the sense that modification of a synaptic weight depends not only on the activities of the pre- and post-synaptic neurons but also on the activities of all other neurons in the entire network.