Feed-Forward Neural Networks

Hee-il Hahn

Professor
Department of Information and Communications Engineering
Hankuk University of Foreign Studies
hihahn@hufs.ac.kr

Content

- Introduction
- Single-Layer Perceptron Networks
- Learning Rules for Single-Layer Perceptron Networks
  - Perceptron Learning Rule
  - Adaline Leaning Rule
  - δ-Leaning Rule
- Multilayer Perceptron
- Back Propagation Learning algorithm

Feed-Forward Neural Networks

Introduction

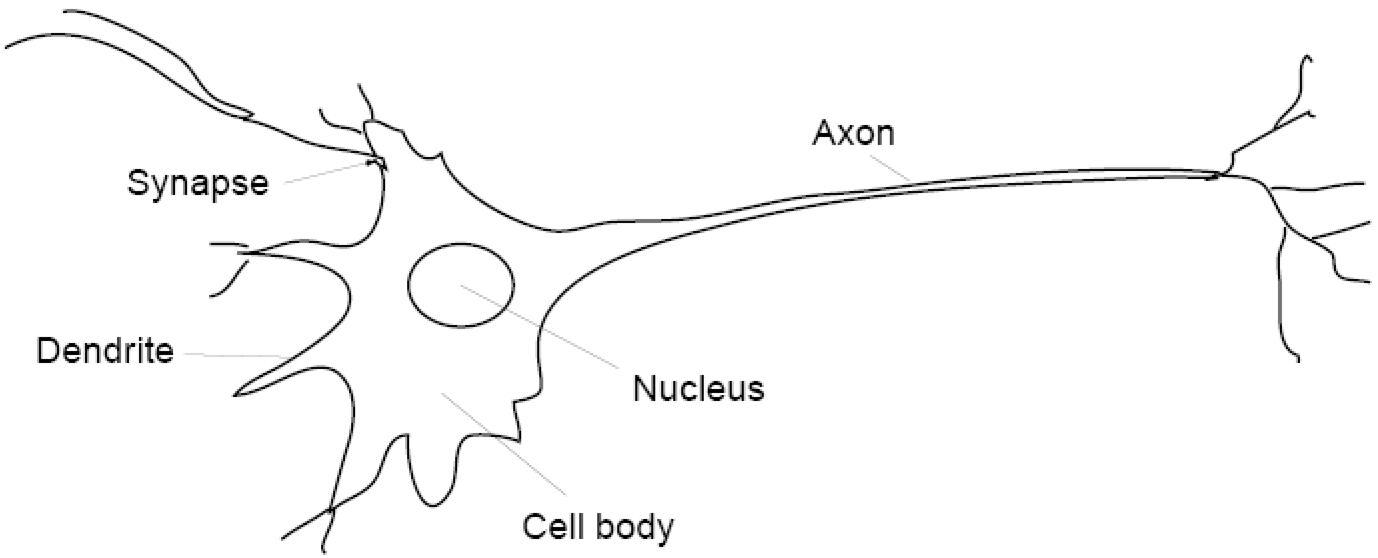
Historical Background

- 1943 McCulloch and Pitts proposed the first computational models of neuron.
- 1949 Hebb proposed the first learning rule.
- 1958 Rosenblatt’s work in perceptrons.
- 1969 Minsky and Papert’s exposed limitation of the theory.
- 1970s Decade of dormancy for neural networks.
- 1980-90s Neural network return (self-organization, back-propagation algorithms, etc.)

Nervous Systems

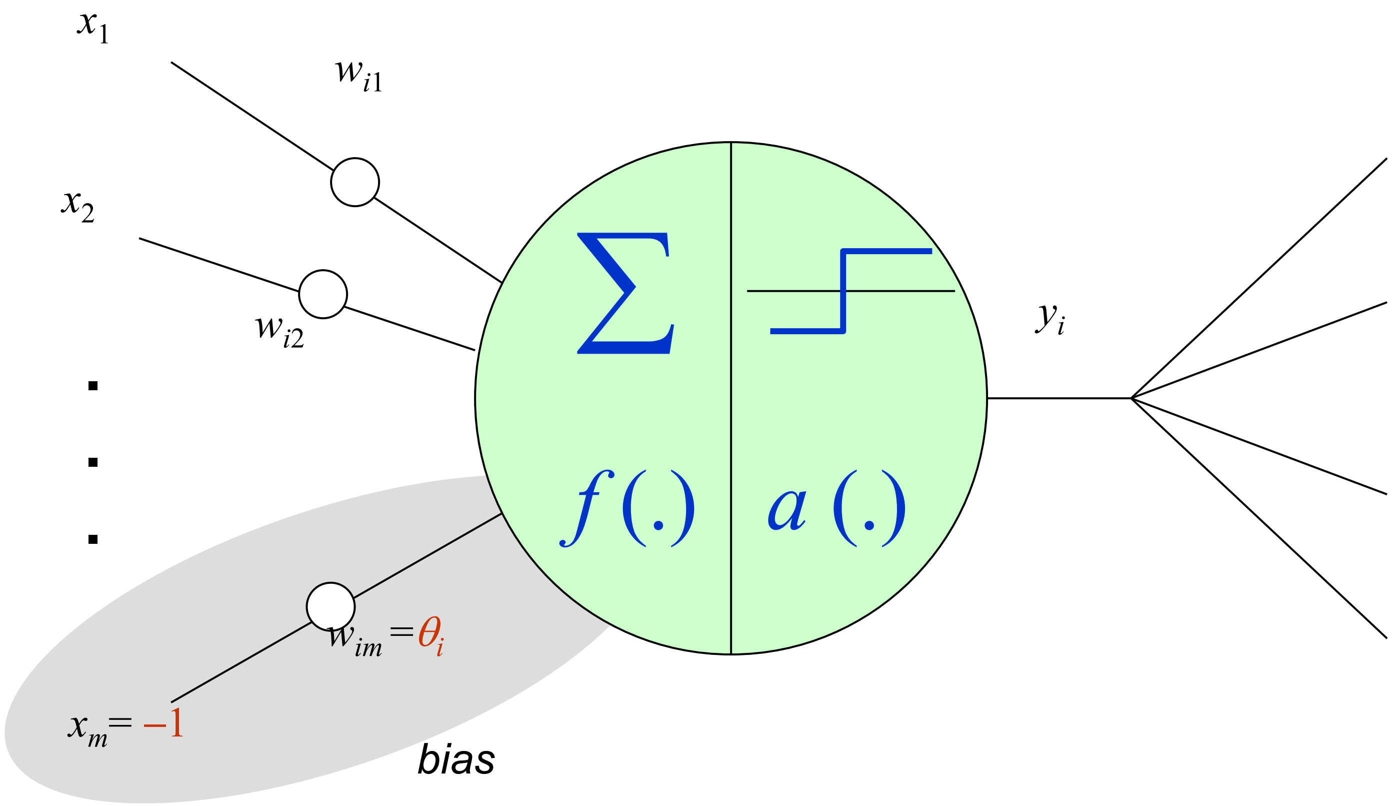
- Human brain contains ~ 10^{11} neurons.
- Each neuron is connected ~ 10^{4} others.
- Some scientists compared the brain with a “complex, nonlinear, parallel computer”.
- The largest modern neural networks achieve the complexity comparable to a nervous system of a fly.

Neurons

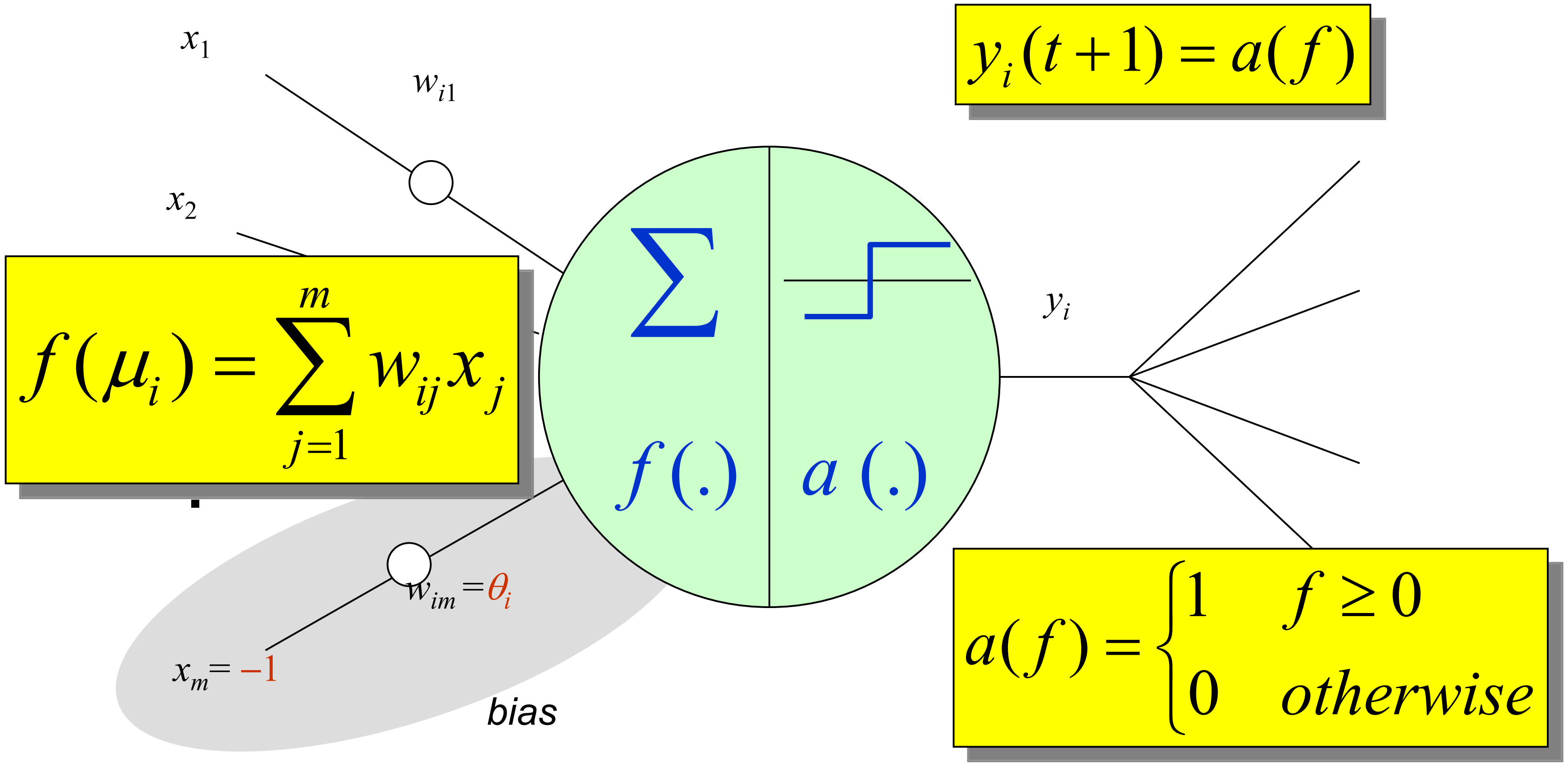


- The main purpose of neurons is to receive, analyze and transmit further the information in a form of signals (electric pulses).
- When a neuron sends the information we say that a neuron “fires”.

A Model of Artificial Neuron

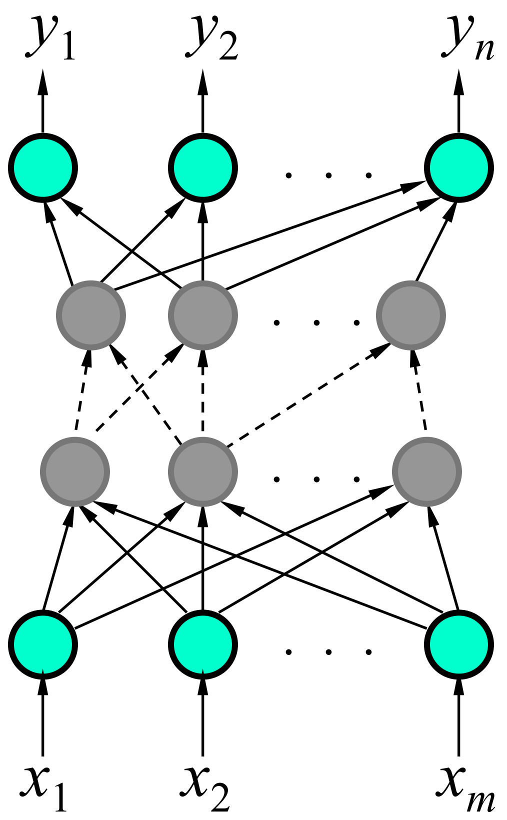


A Model of Artificial Neuron



Feed-Forward Neural Networks

- Graph representation:
  - nodes: neurons
  - arrows: signal flow directions
- A neural network that does not contain cycles (feedback loops) is called a feed–forward network (or perceptron).

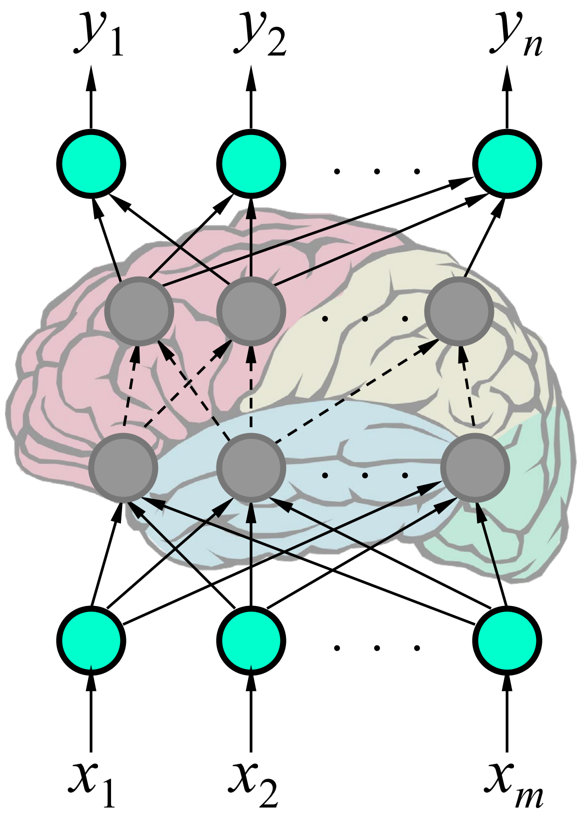


Layered Structure

Output Layer

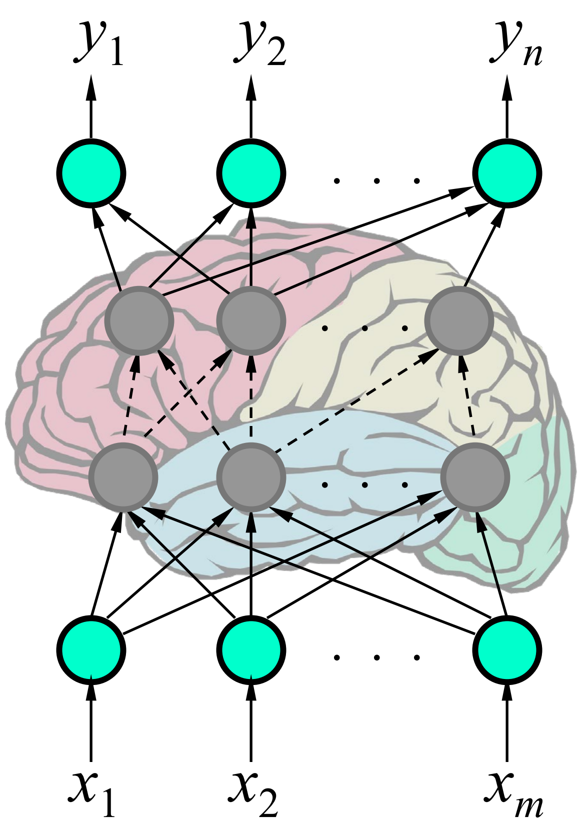
Hidden Layer(s)

Input Layer



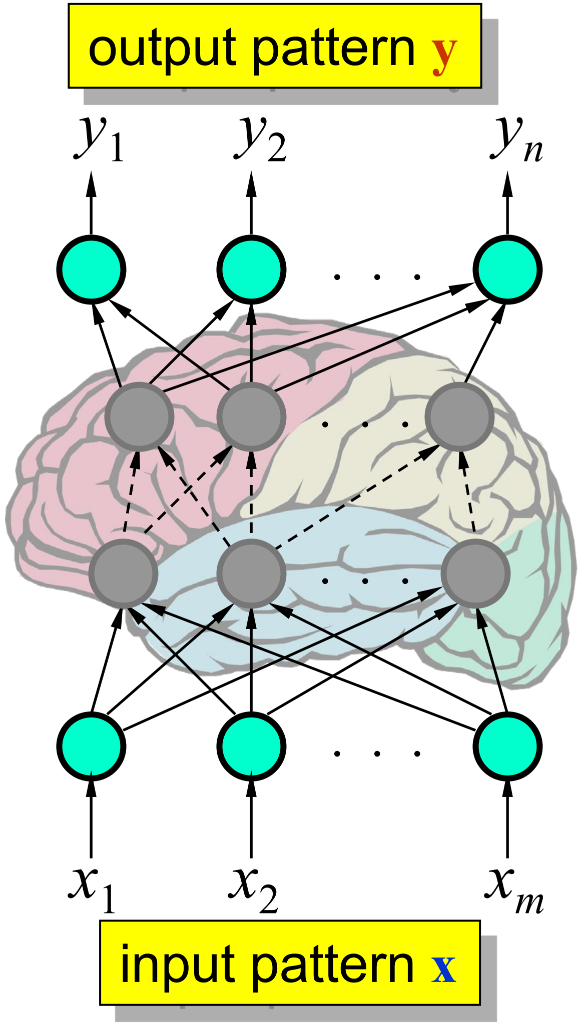
Knowledge and Memory

- The output behavior of a network is determined by the weights.
- Weights – the memory of an NN.
- Knowledge – distributed across the network.
- Large number of nodes
  - increases the storage "capacity";
  - ensures that the knowledge is robust;
  - fault tolerance.
- Store new information by changing weights.



Pattern Classification

- Function: x → y
- The NN’s output is used to distinguish between and recognize different input patterns.
- Different output patterns correspond to particular classes of input patterns.
- Networks with hidden layers can be used for solving more complex problems than just a linear pattern classification.



Training

Training Set

$$\mathbf{T} = \left\{(\mathbf{x}^{(1)}, \mathbf{d}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{d}^{(2)}), \ldots, (\mathbf{x}^{(k)}, \mathbf{d}^{(k)}), \ldots\right\}$$

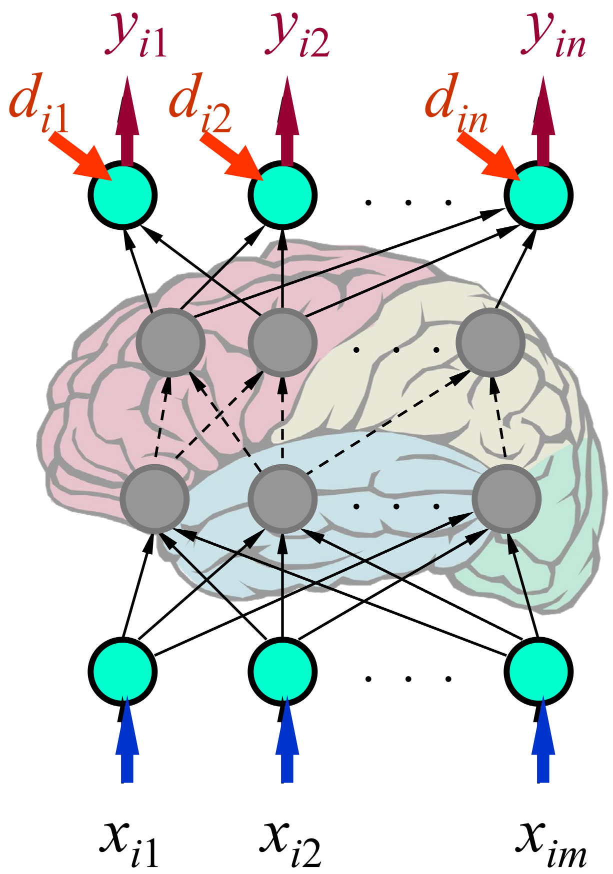
$$\mathbf{x}^{(i)} = (x_{i1}, x_{i2}, \ldots, x_{im})$$

$$d^{(i)} = (d_{i1}, d_{i2}, \ldots, d_{in})$$

Goal:

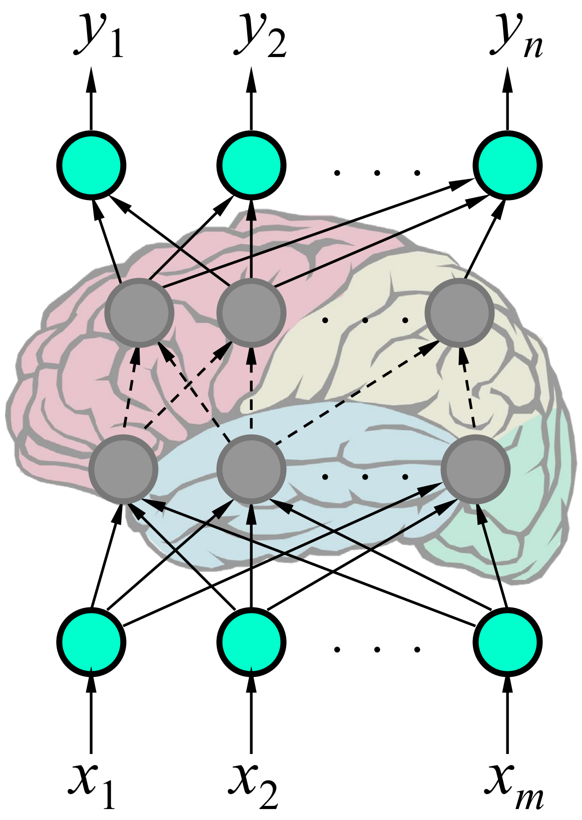
$$\text{Min } E = \sum_i error(\mathbf{y}^{(i)} - \mathbf{d}^{(i)})$$

$$= \sum_i \left\|\mathbf{y}^{(i)} - \mathbf{d}^{(i)}\right\|^2$$



Generalization

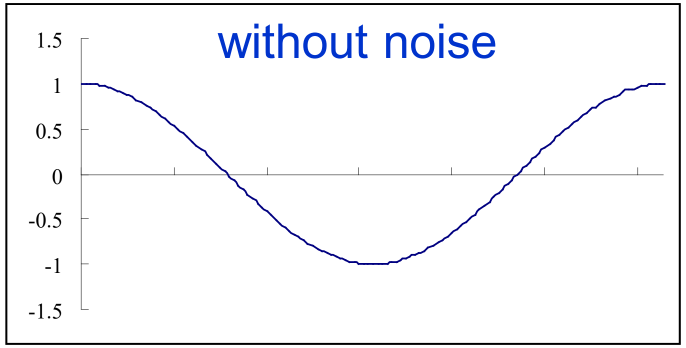
- Properly training a neural network may produce reasonable answers for input patterns not seen during training (generalization).
- Generalization is particularly useful for the analysis of a "noisy" data (e.g. time–series).

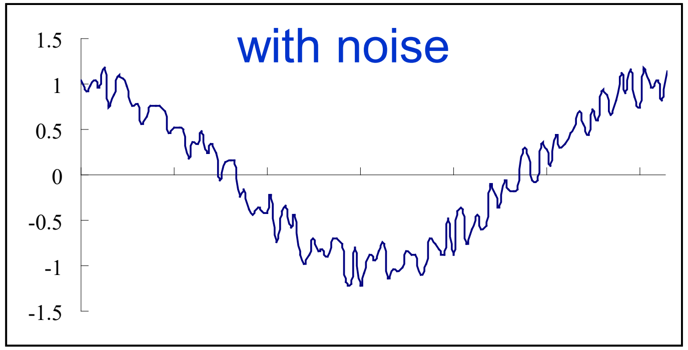


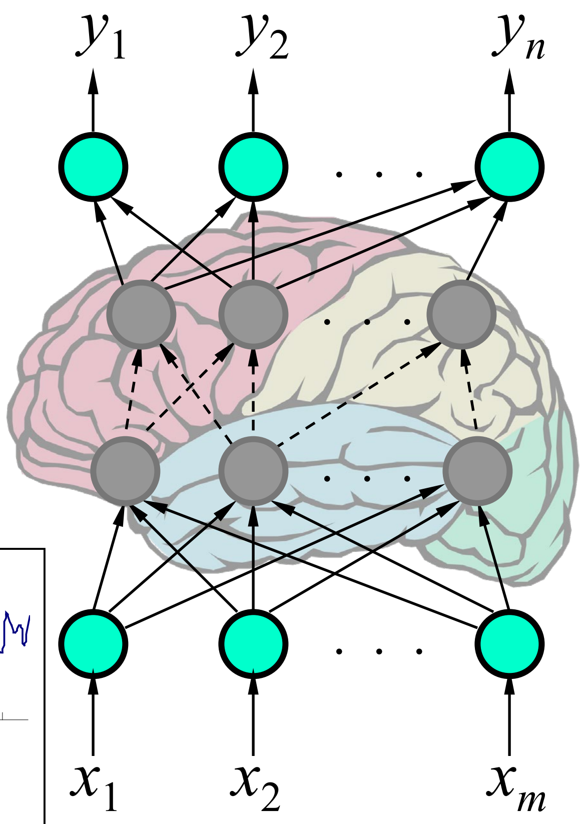
Generalization

- Properly training a neural network may produce reasonable answers for input patterns not seen during training (generalization).

- Generalization is particularly useful for the analysis of a "noisy" data (e.g. time–series).







Applications

- Pattern classification
- Object recognition
- Function approximation
- Data compression
- Time series analysis and forecast
- . . .