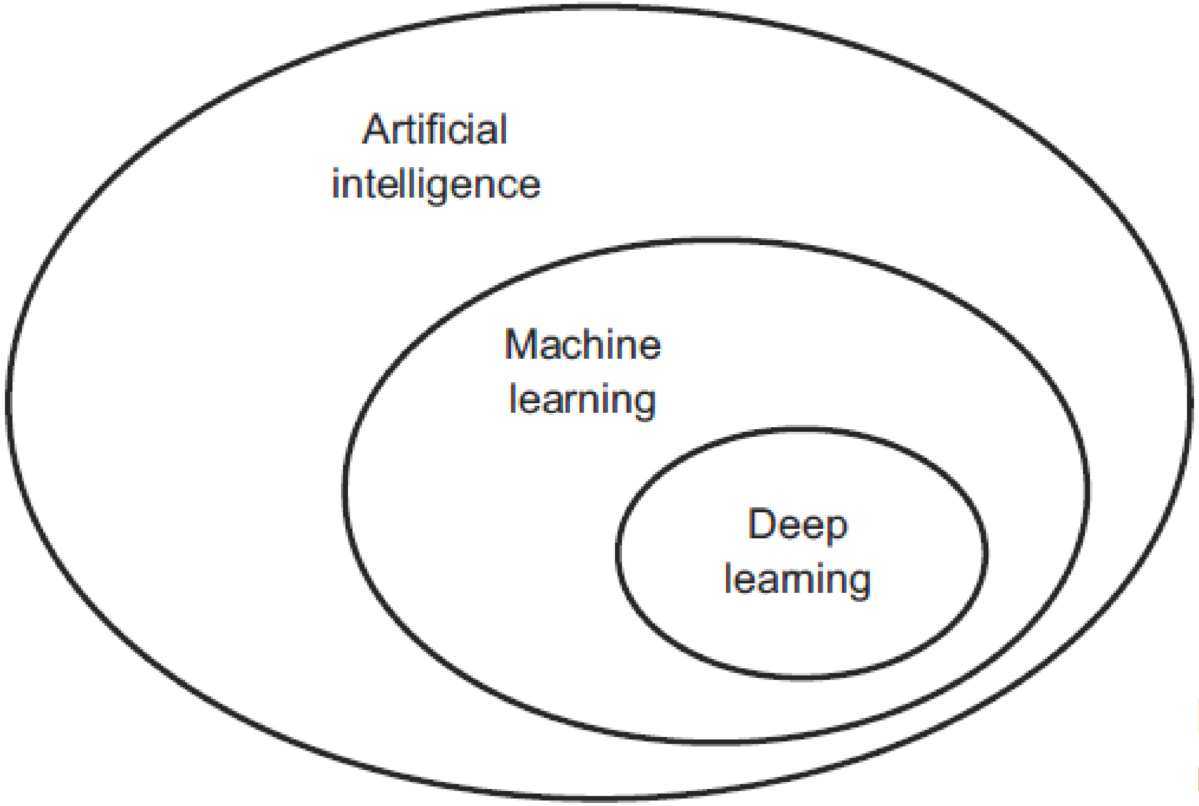
GRASP Laboratory
General Robotics & Autonomous Systems and Processes

Mechatronic Modeling and Design with Applications in Robotics

AI in Modeling and Design

AI, Machine Learning and Deep Learning

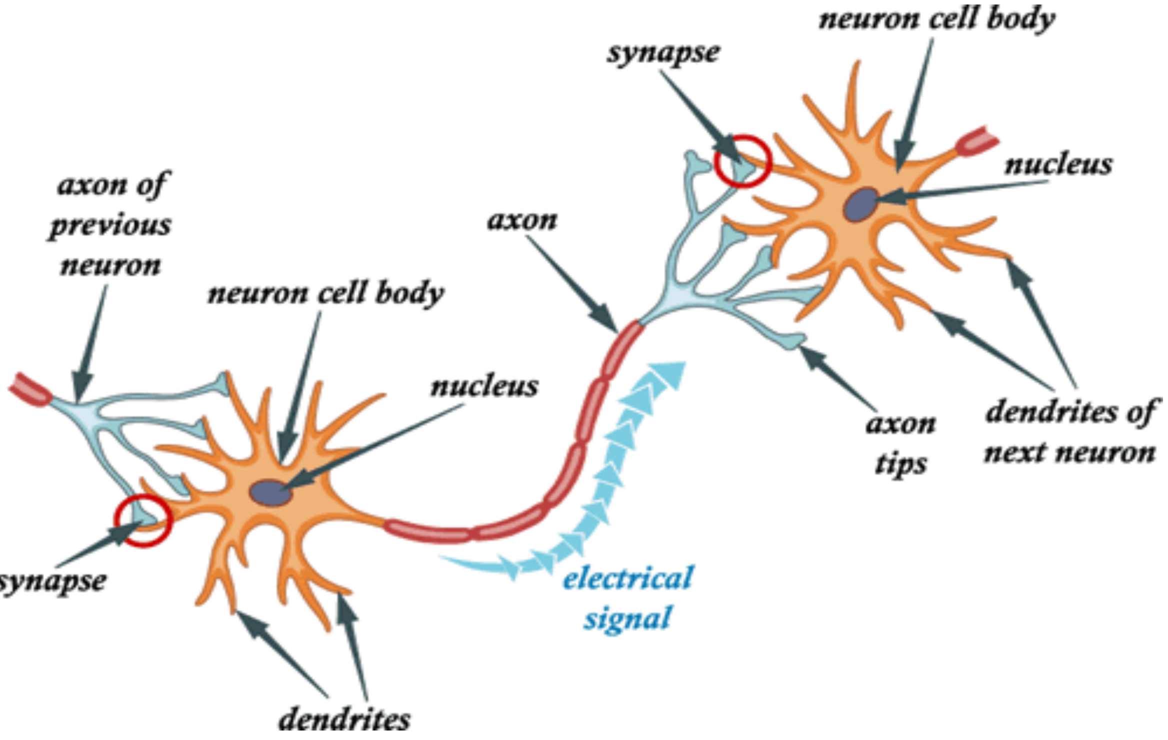
Artificial Intelligence, Machine Learning and Deep Learning



Neural Networks

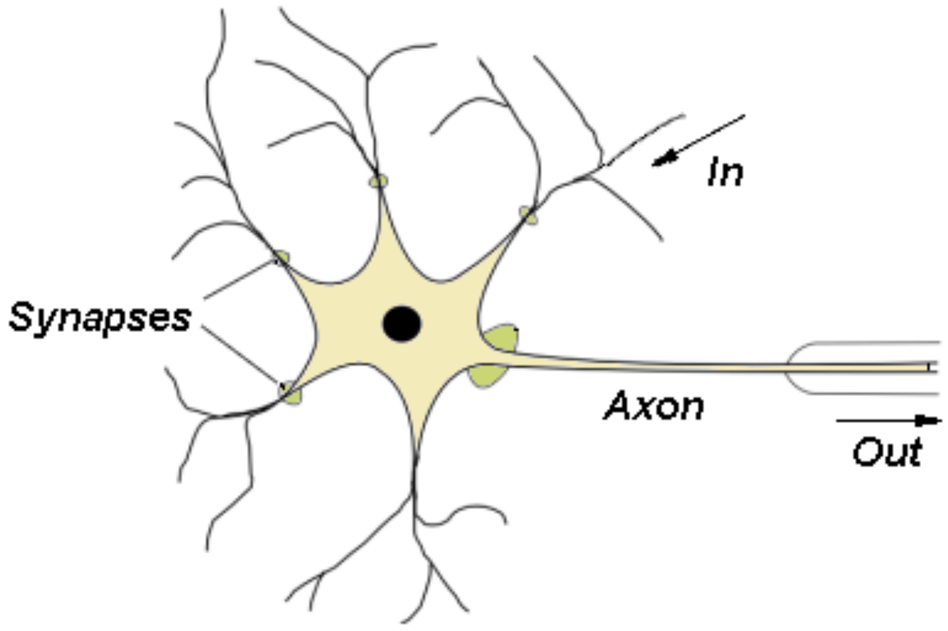
Human and Artificial Neurons

- Neural networks have a long history which goes back to the first attempts to understand how the human and mammal brain works and how/what we call intelligence is formed.

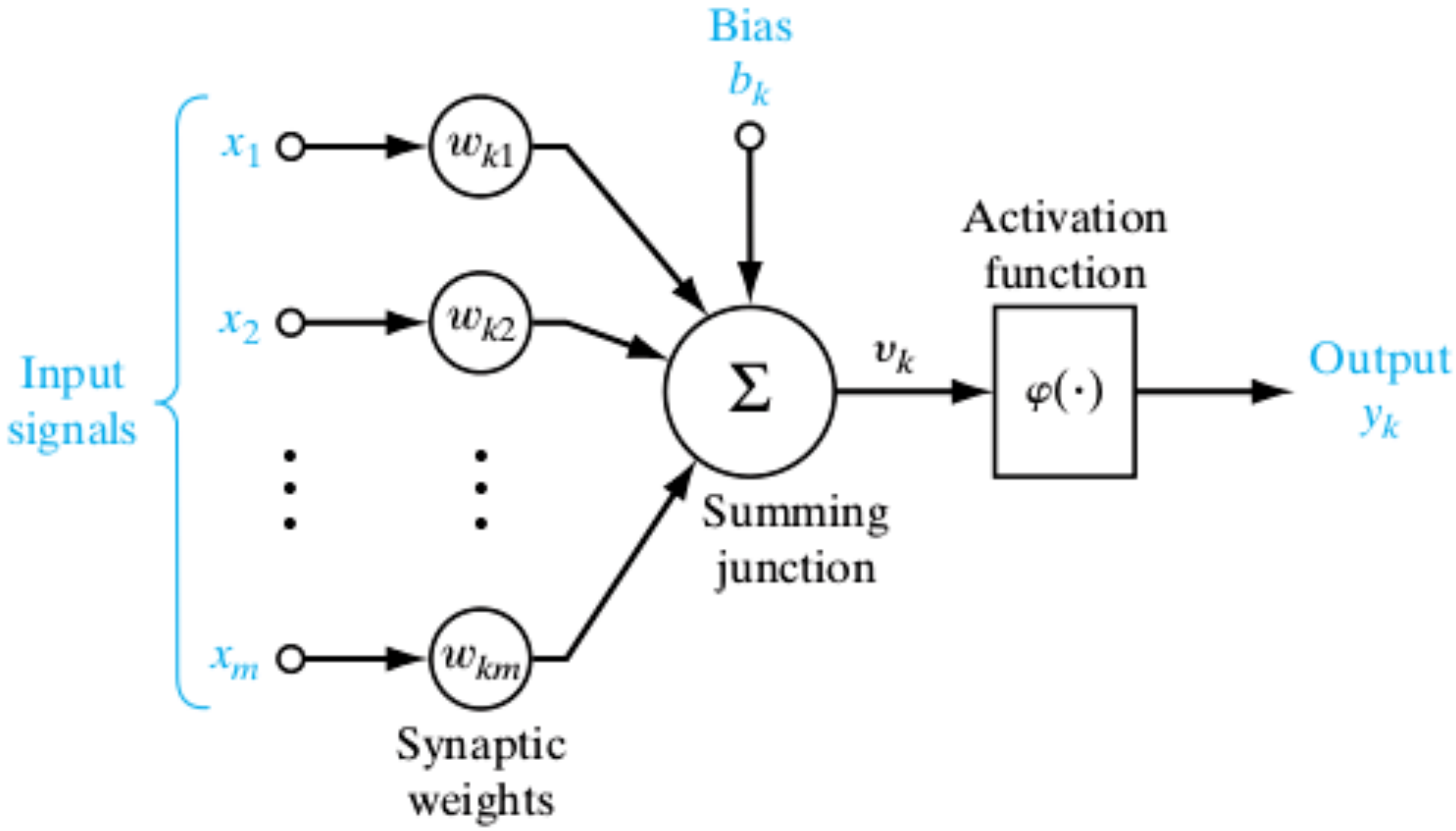


- The human b

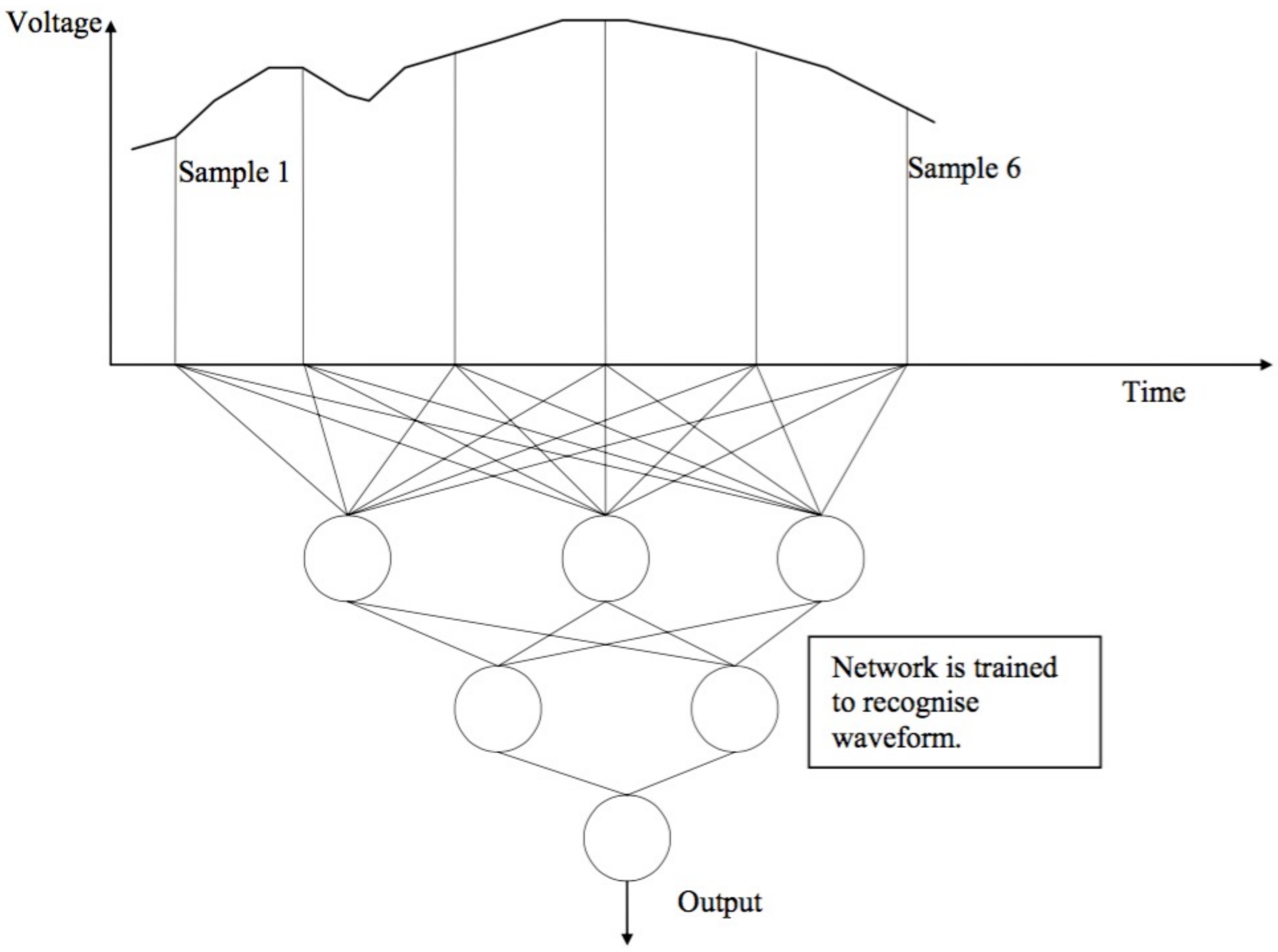
Each neuron is connected with other neurons via elementary structural and functional units/links, known as synapses. It is estimated that there are 50-100 trillions of synapses. These links mediate information between connected neurons.



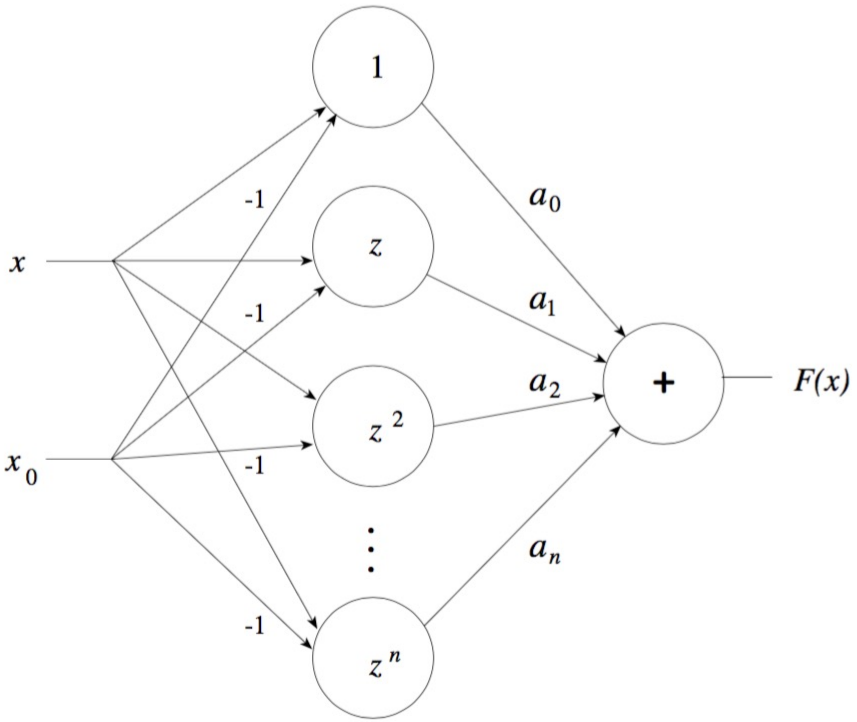
1943: A milestone -- Warren McCulloch and Walter Pitts, developed a computational model for the basic neuron.



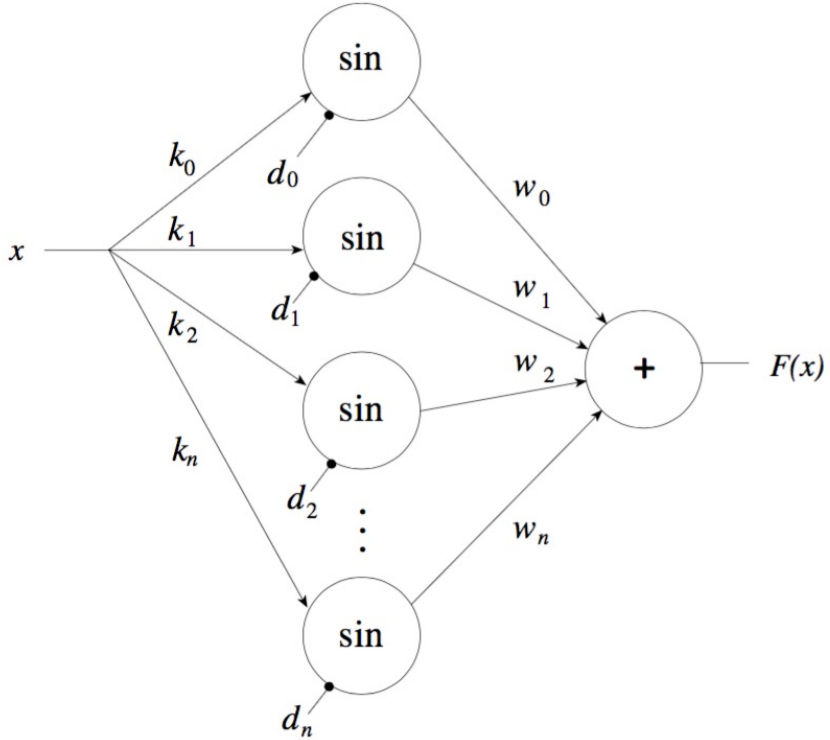
Engineering Applications



Other Applications: Approximation



A Taylor Network



A Fourier Network

$$F(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n + \cdots,$$

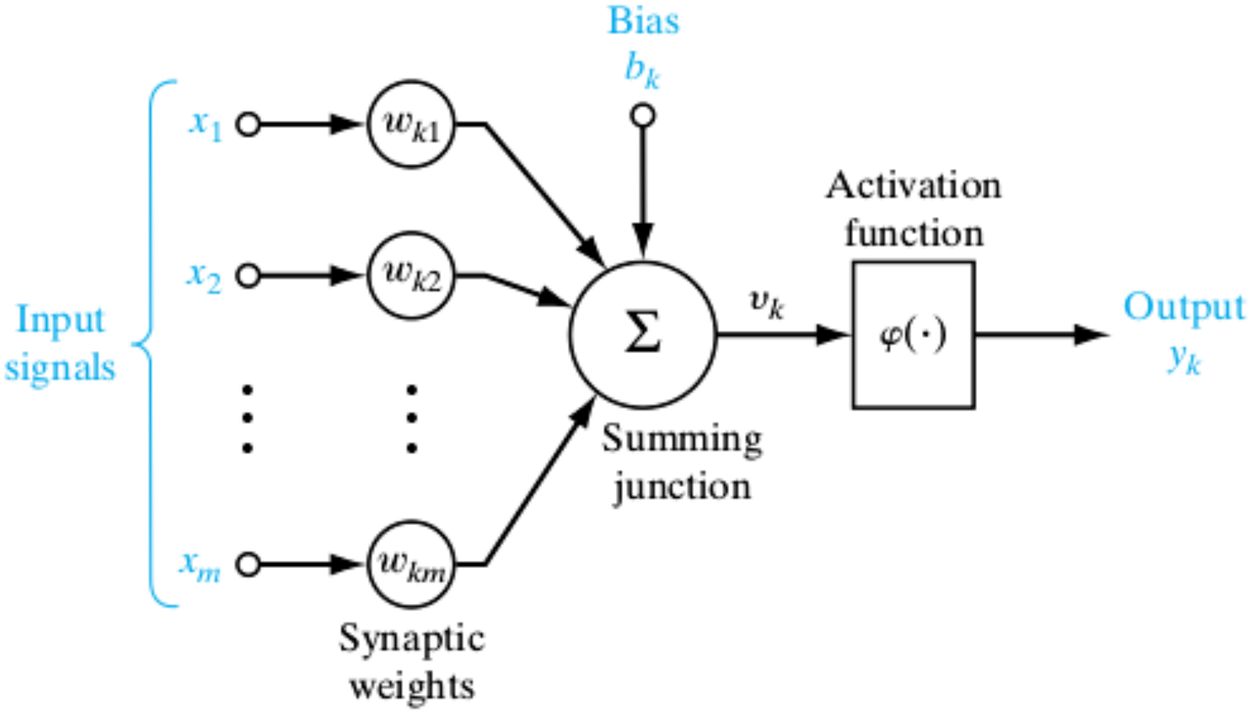
$$F(x) = \sum_{i=0}^{\infty}(a_i \cos(ix) + b_i \sin(ix)).$$

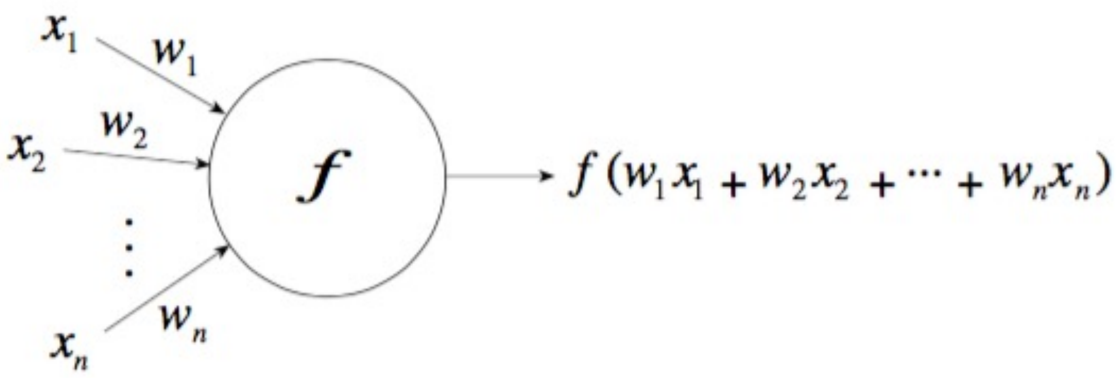
Some Features of Neural Networks

- A universal nonlinear approximator.
- Adaptive learning: An ability to learn how to do tasks based on the trained data.
- Self-Organization: create its own organization or representation of the information during learning time.
- Real Time Operation
- Fault Tolerance
- Application: ANNs are used when the domain of a problem is not entirely known.

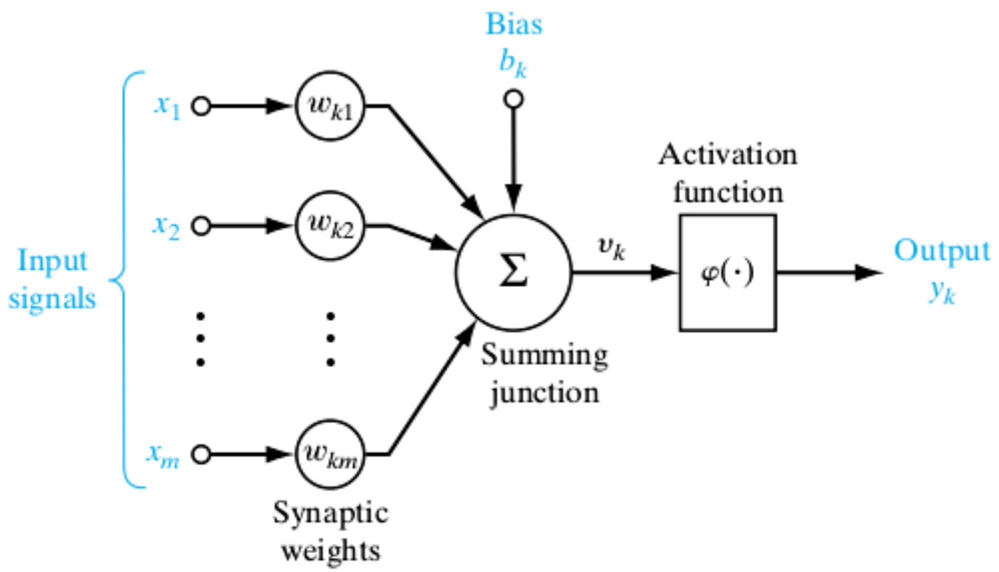
Neural networks do not perform miracles. But if used sensibly they can produce some amazing results.

Networks of Primitive Functions



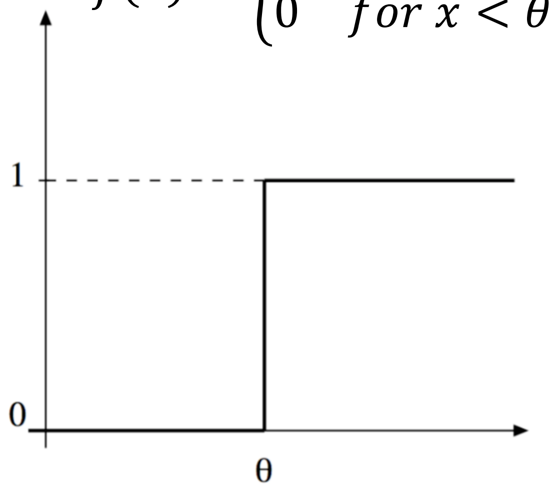


Activation Functions



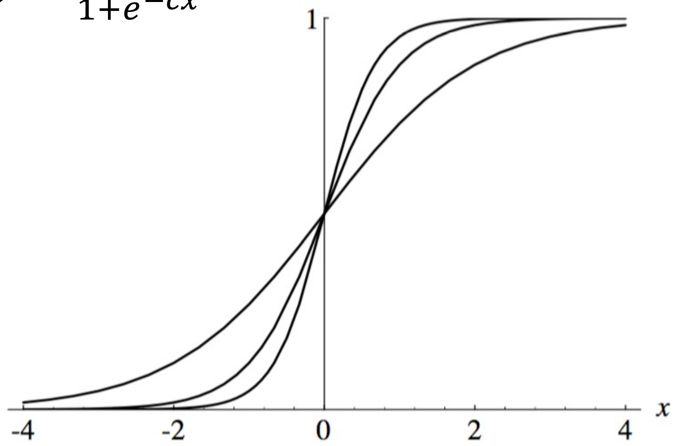
The step function with threshold

$$f(x) = \begin{cases} 1 & for\ x \geq \theta \\ 0 & for\ x < \theta \end{cases}$$

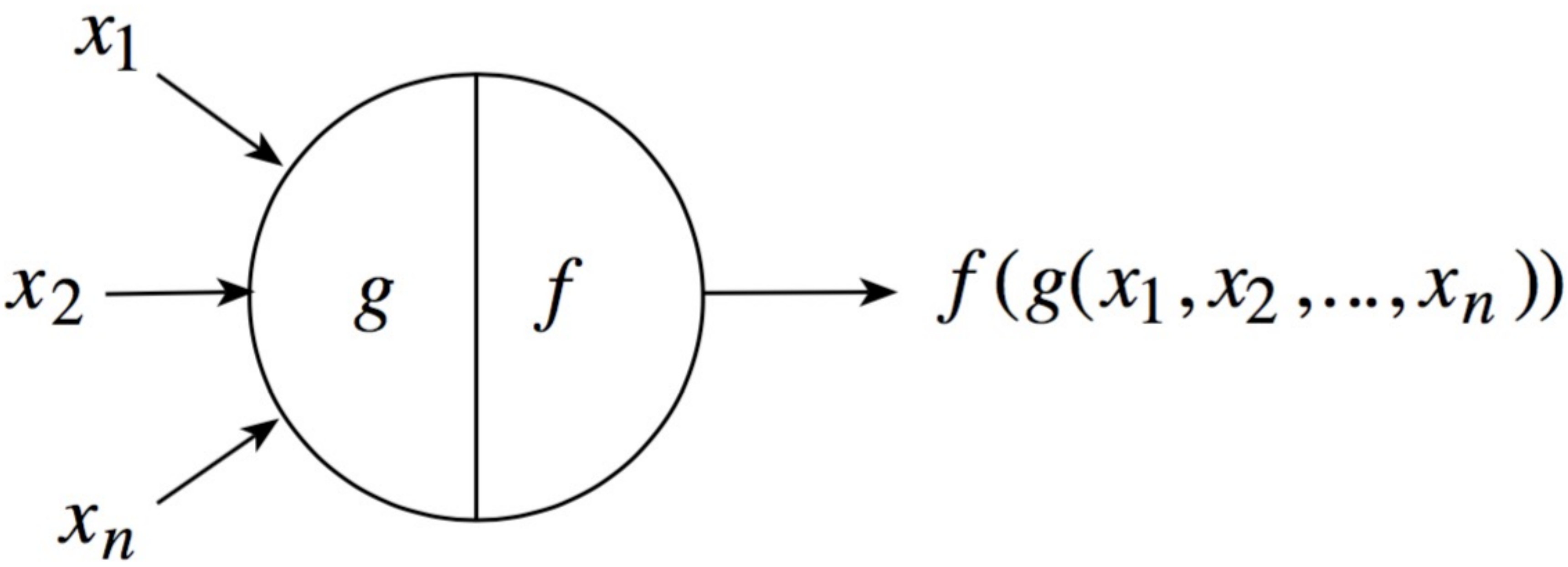


Sigmoid

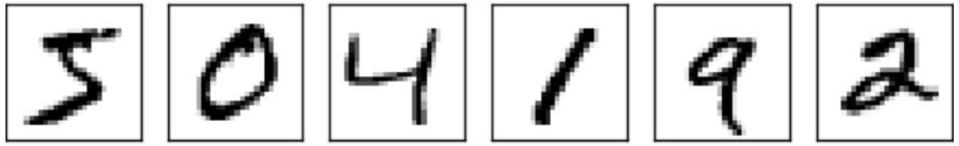
$$f(x) = \frac{1}{1+e^{-cx}}$$

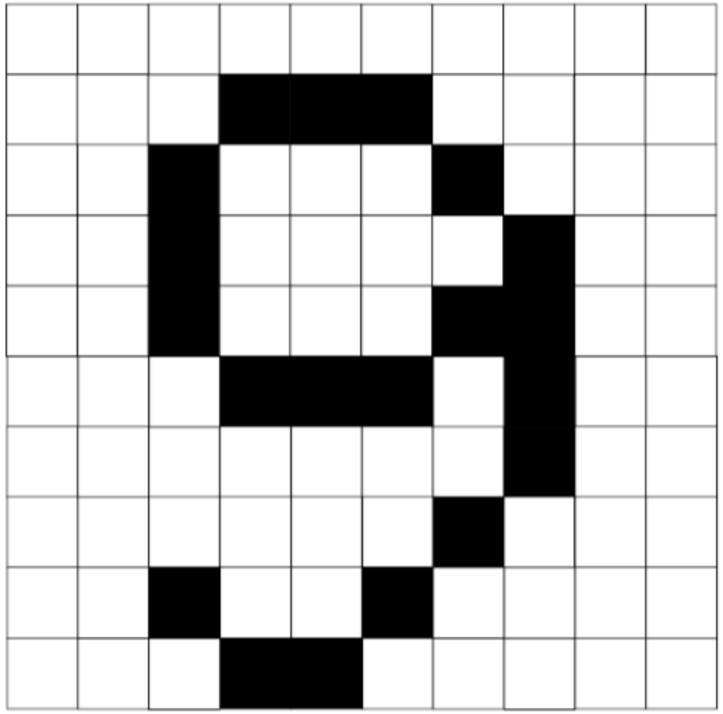


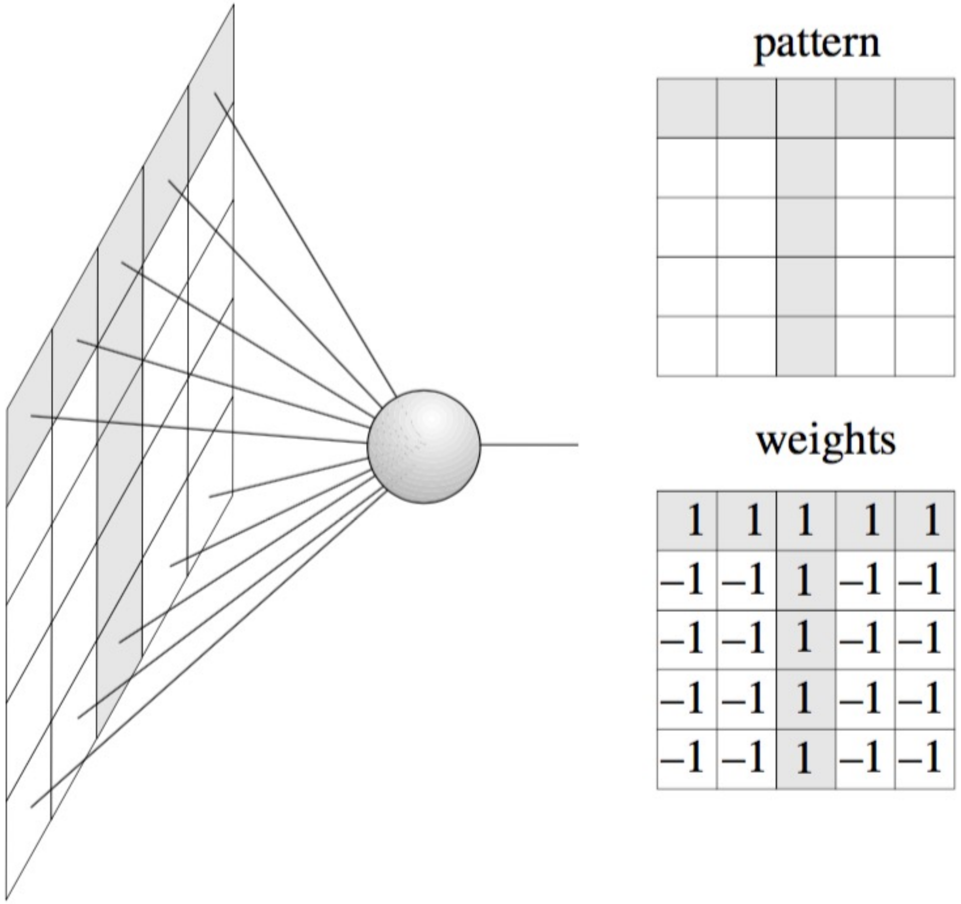
Generic Computing Unit



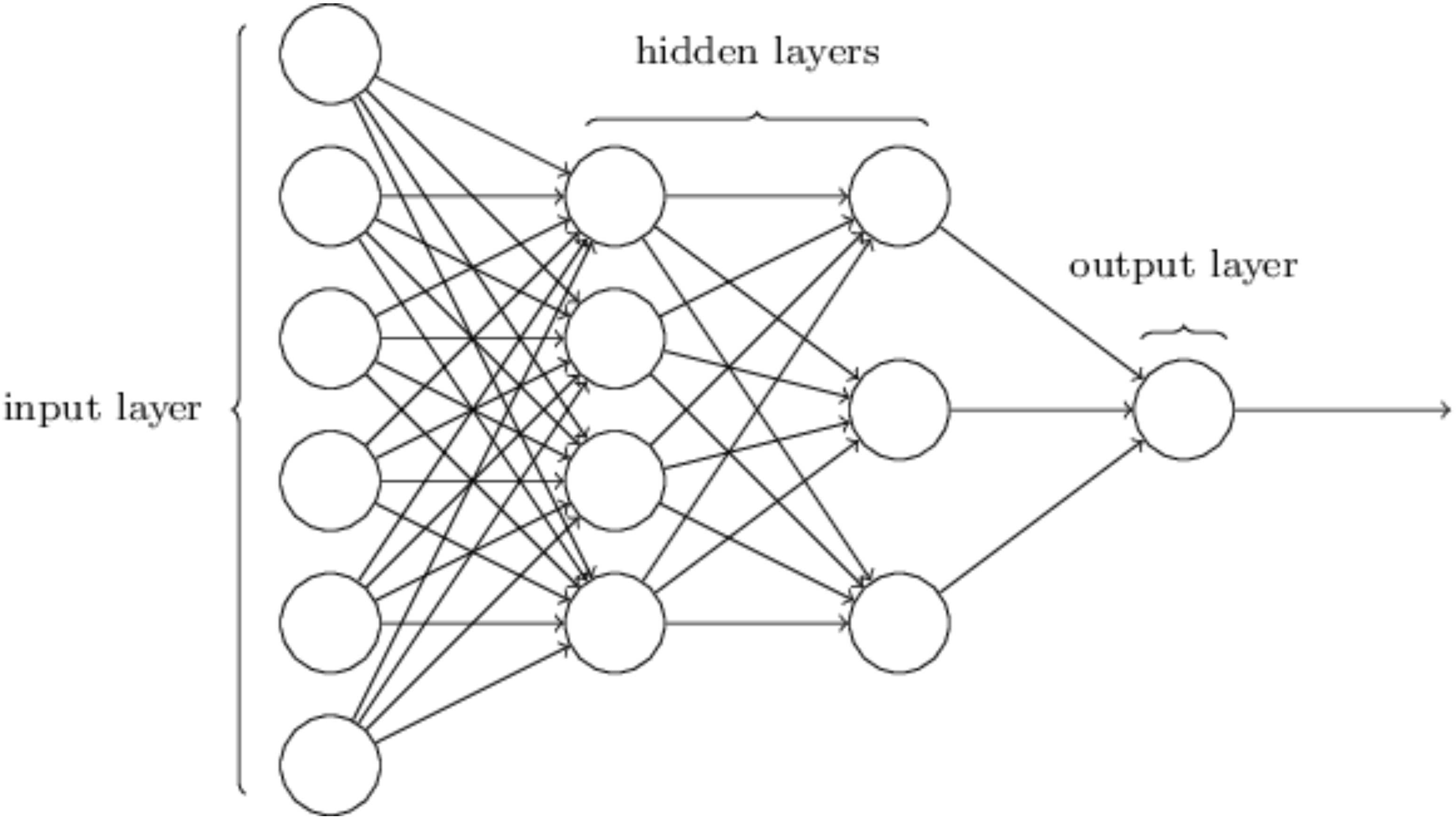
Pre-processing Input Data





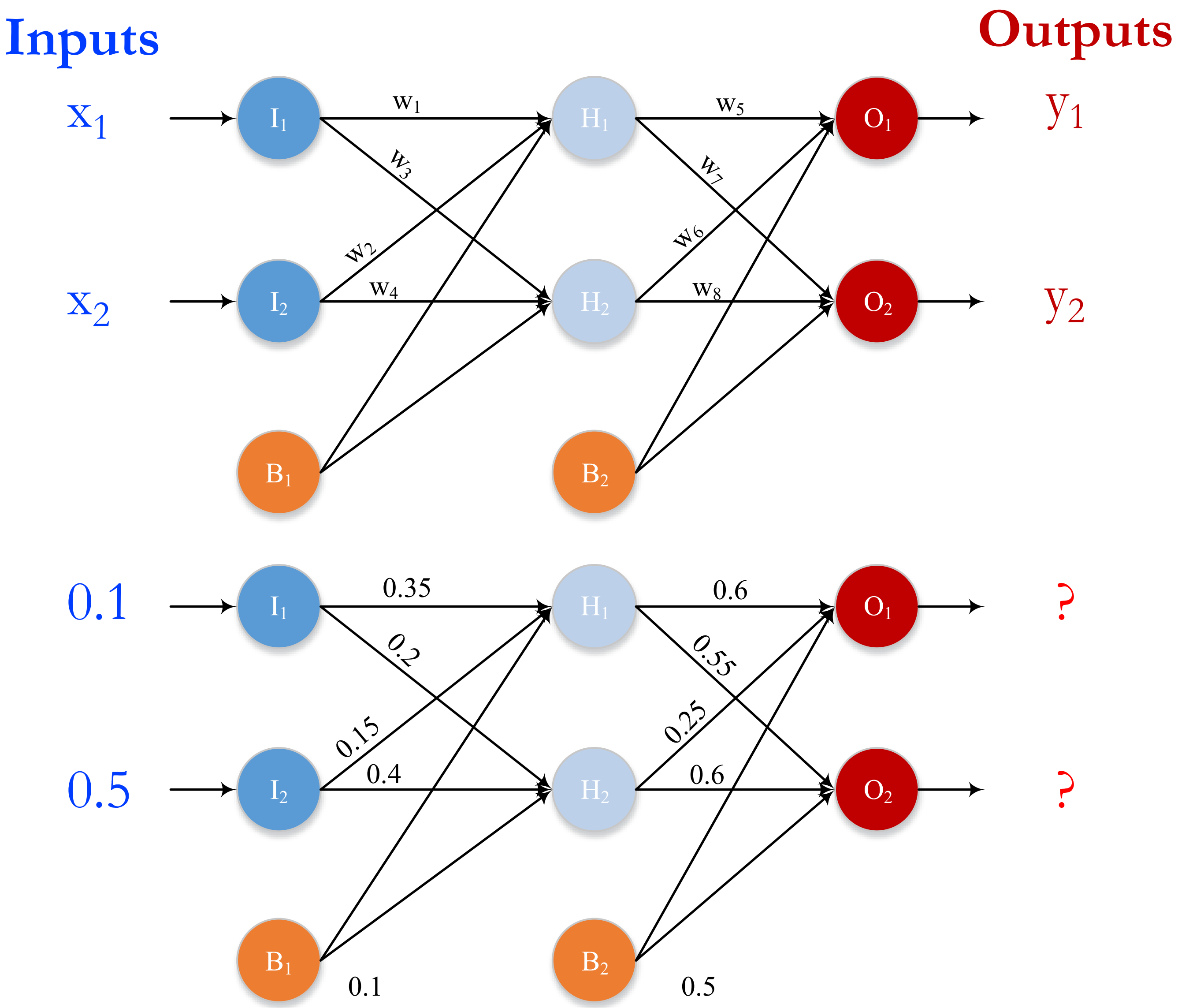


Feed-forward Neural Networks



Feed-forward ANNs allow signals to travel on way only; from input to output. There is no feedback loops(e.g., the output of any layer does not affect the same layer)

An Example of NNs



An Example of NNs: Forward Propagation

- The inputs of the hidden layer are:

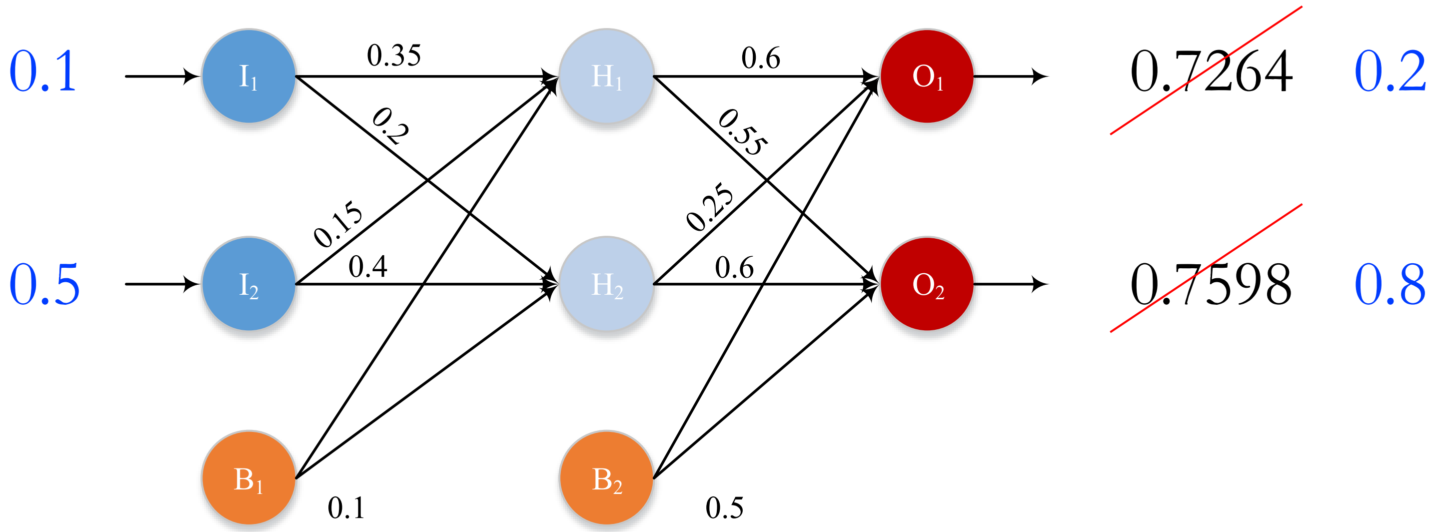
$$in_{H1} = i_1 \times w_1 + i_2 \times w_2 + b_1 = 0.1 \times 0.35 + 0.5 \times 0.15 + 0.1 = 0.21$$
$$in_{H2} = i_1 \times w_3 + i_2 \times w_4 + b_1 = 0.1 \times 0.2 + 0.5 \times 0.4 + 0.1 = 0.32$$

- The output of the hidden layer are:

$$out_{H1} = \frac{1}{1+e^{-in_{H1}}} = \frac{1}{1+e^{-0.21}} = 0.5523$$
$$out_{H2} = \frac{1}{1+e^{-in_{H2}}} = \frac{1}{1+e^{-0.32}} = 0.5793$$

- Repeat the process for finding the inputs of the output layer and the outputs of the output layer:

$$in_{O1} = out_{H1} \times w_5 + out_{H2} \times w_6 + b_2 = 0.5523 \times 0.6 + 0.5793 \times 0.25 + 0.5 = 0.9762$$
$$in_{O2} = out_{H1} \times w_7 + out_{H2} \times w_8 + b_2 = 0.5523 \times 0.55 + 0.5793 \times 0.6 + 0.5 = 1.1514$$
$$out_{O1} = \frac{1}{1+e^{-in_{O1}}} = \frac{1}{1+e^{-0.9762}} = 0.7264$$
$$out_{O2} = \frac{1}{1+e^{-in_{O2}}} = \frac{1}{1+e^{-1.1514}} = 0.7598$$



Backpropagation:

The goal of the backpropagation training is to update the weights so that the neural network can learn and map the given input-output groups.

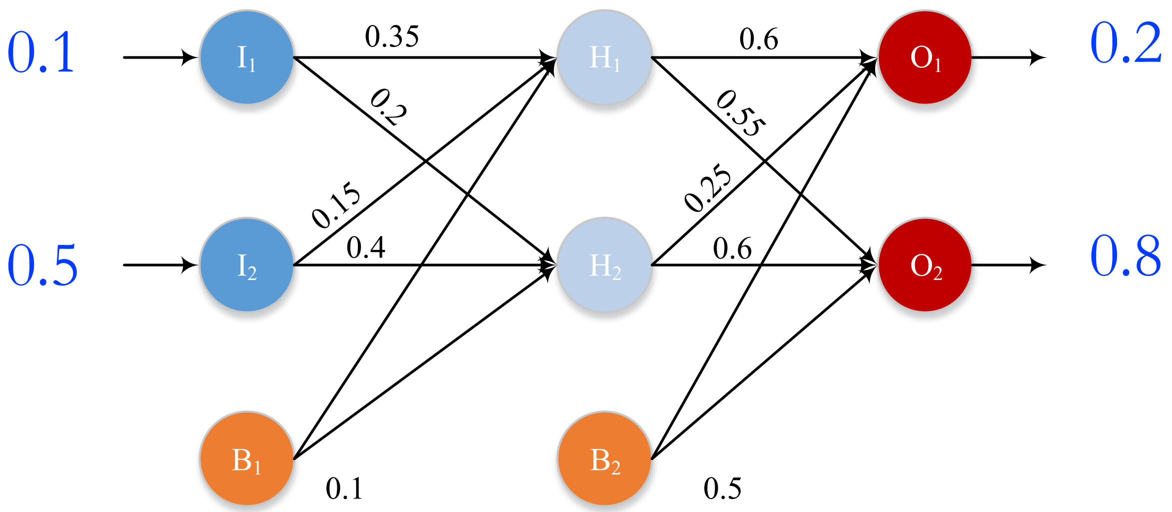
Basic Training Principle/Steps

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.

2. We determine how closely the actual output of the network matches the desired output.

3. We change the weight of each connection so that the network produces a better approximation of the desired output.

A Training Example



Calculating the forward path:

$$out_{O1} = \frac{1}{1+e^{-in_{O1}}} = \frac{1}{1+e^{-0.9762}} = 0.7264$$

$$out_{O2} = \frac{1}{1+e^{-in_{O2}}} = \frac{1}{1+e^{-1.1514}} = 0.7598$$

A Training Example (cont'd)

Calculating the total error:

$$E_{Total} = \sum \frac{1}{2}(target - output)^2 = \frac{1}{2}(target1 - out_{O1})^2 + \frac{1}{2}(target2 - out_{O2})^2$$

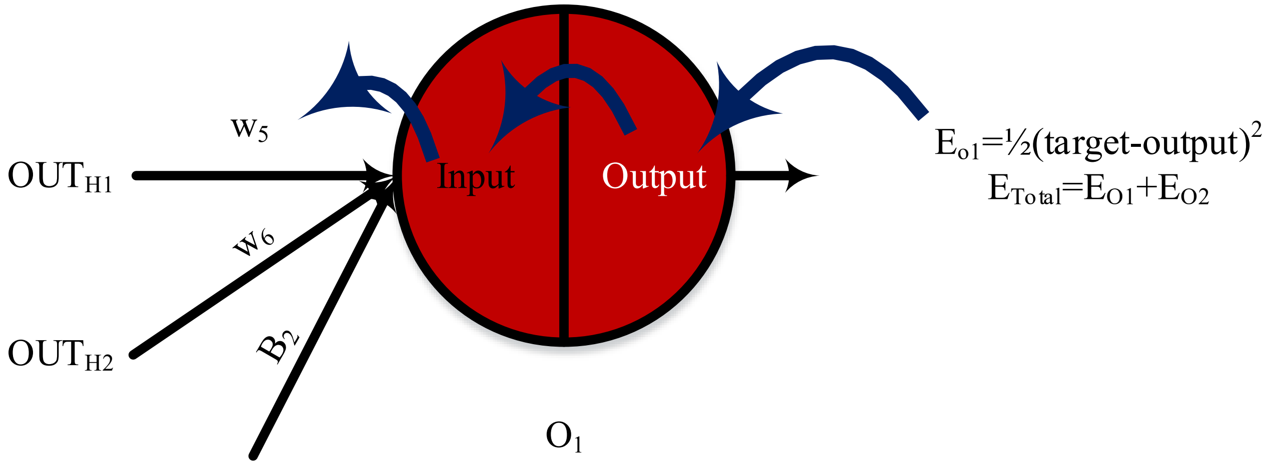
$$E_{Total} = \frac{1}{2}(0.2 - 0.7264)^2 + \frac{1}{2}(0.8 - 0.7598)^2 = 0.1385 + 0.0008 = 0.1393$$

$$E_{O1} = 0.1385$$

$$E_{O2} = 0.0008$$

A Training Example (cont'd)

Calculating the backward pass and update weights:



How much a change in w_5 affects the total error:

$$\frac{\partial E_{Total}}{\partial_{w5}} = \frac{\partial E_{Total}}{\partial OUT_{O_1}} \times \frac{\partial OUT_{O_1}}{\partial IN_{O1}} \times \frac{\partial IN_{O1}}{\partial w_5}$$ Gradient Descent

A Training Example (cont'd)

$$E_{Total} = \frac{1}{2}(target_{O1} - OUT_{O1})^2 + \frac{1}{2}(tatget_{O2} - OUT_{O2})^2$$

$$\frac{\partial E_{Total}}{\partial OUT_{O1}} = 2 \times \frac{1}{2}(target_{O1} - OUT_{O1}) \times (-1)$$

$\frac{\partial OUT_{O1}}{\partial IN_{O1}} = OUT_{O1}(1 - OUT_{O1})$ because $OUT_{O1} = \frac{1}{1+e^{-IN_{O1}}}$

Finally, $IN_{O1} = w_5 \times OUT_{H1} + w_6 \times OUT_{H2} + B_2$

$$\frac{\partial IN_{O1}}{\partial w_5} = OUT_{H1}$$

Putting them all together:

$$\frac{\partial E_{Total}}{\partial w_5} = -(target_{O1} - OUT_{O1}) \times OUT_{01}(1 - OUT_{O1}) \times OUT_{H1}$$

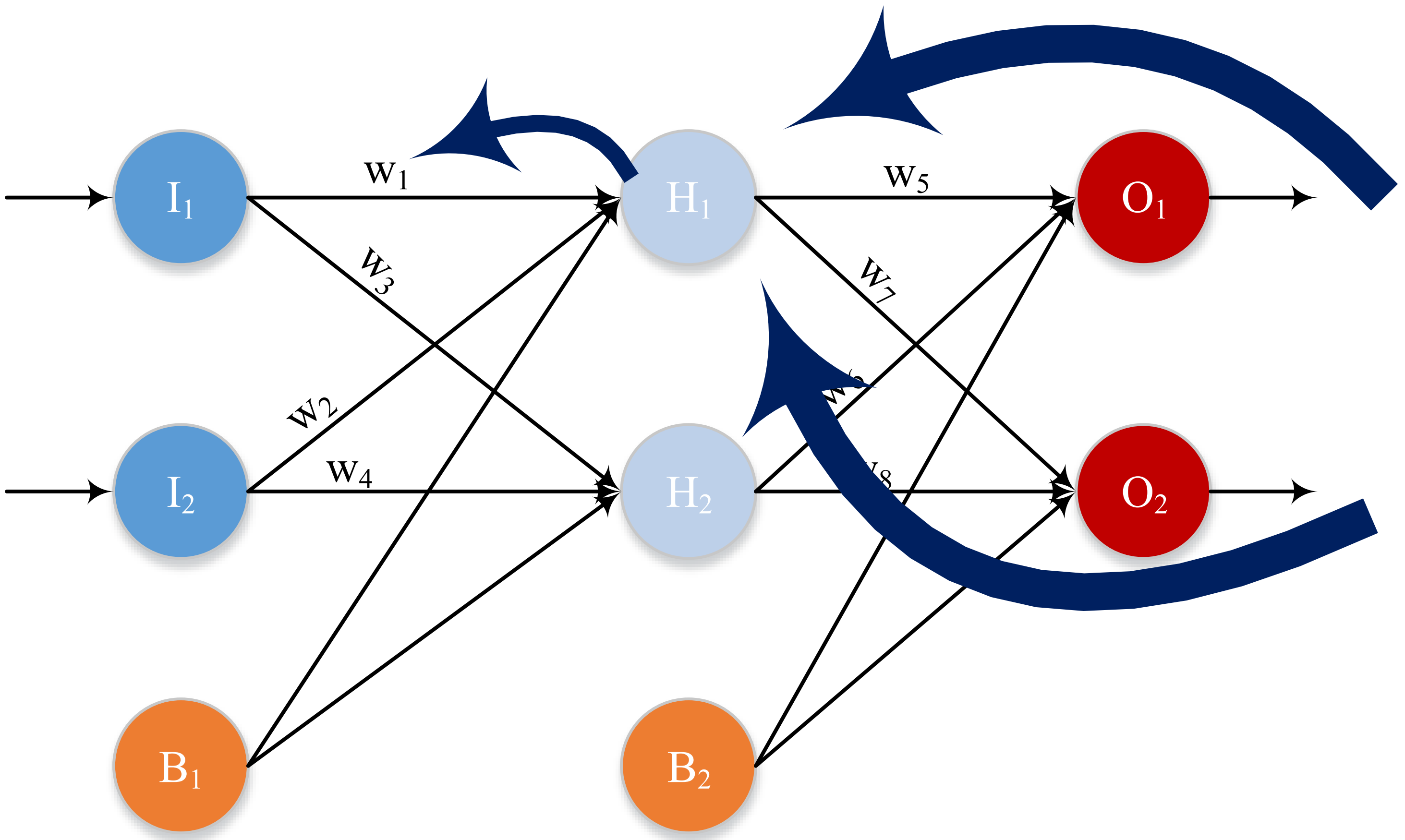
A Training Example (cont'd)

Alternatively, we have $\frac{\partial E_{Total}}{\partial OUT_{O1}}$ and $\frac{\partial OUT_{O1}}{\partial IN_{O1}}$ which can be written as $\frac{\partial E_{Total}}{\partial IN_{O1}}$, aka δ_{O1}

To decrease the error, $w_5^* = w_5 - \eta \times \frac{\partial Etotal}{\partial w_5}$

$$w_5^* = w_5 - \eta \times \frac{\partial E_{Total}}{\partial w_5} = 0.6 - 0.5 \times 0.578 = 0.5711$$
$$w_6^* = 0.2197$$
$$w_7^* = 0.5520$$
$$w_8^* = 0.6021$$

A Training Example (cont'd)



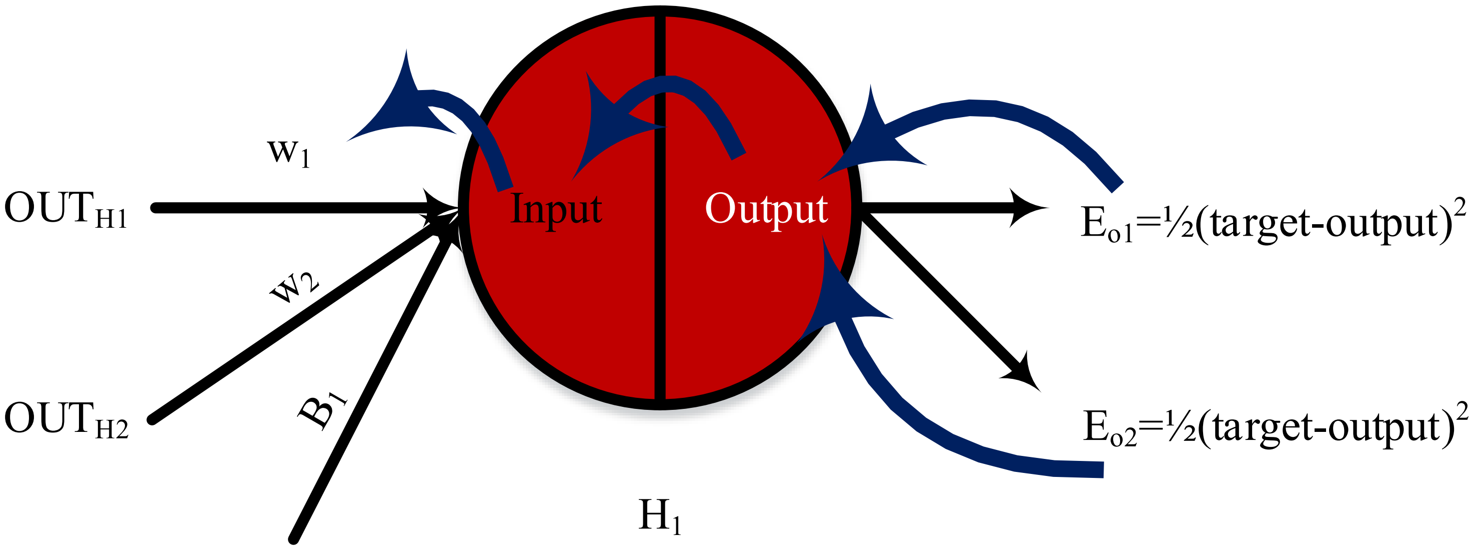
$E_{o1}=½(target-output)^2$

$E_{Total}=E_{O1}+E_{O2}$

$E_{o2}=½(target-output)^2$

$E_{Total}=E_{O1}+E_{O2}$

A Training Example (cont'd)



$$\frac{\partial E_{Total}}{\partial w_1} = \frac{\partial E_{Total}}{\partial OUT_{H1}} \times \frac{\partial OUT_{H1}}{\partial IN_{H1}} \times \frac{\partial IN_{H1}}{\partial w_1}$$

$$\frac{\partial E_{Total}}{\partial OUT_{H1}} = \frac{\partial E_{O1}}{\partial OUT_{H1}} + \frac{\partial E_{O2}}{\partial OUT_{H1}}$$

$$\frac{\partial E_{Total}}{\partial w_1} = \left(\frac{\partial E_{O1}}{\partial OUT_{H1}} + \frac{\partial E_{O2}}{\partial OUT_{H1}}\right) \times \frac{\partial OUT_{H1}}{\partial IN_{H1}} \times \frac{\partial IN_{H1}}{\partial w_1}$$

A Training Example (cont'd)

$$\frac{\partial E_{Total}}{\partial w_1} = \left(\frac{\partial E_{O1}}{\partial OUT_{O1}} \times \frac{\partial OUT_{O1}}{\partial IN_{O1}} \times \frac{\partial IN_{O1}}{\partial OUT_{H1}} + \frac{\partial E_{O2}}{\partial OUT_{O2}} \times \frac{\partial OUT_{O2}}{\partial IN_{O2}} \times \frac{\partial IN_{O2}}{\partial OUT_{H1}}\right) \times \frac{\partial OUT_{H1}}{\partial IN_{H1}} \times \frac{\partial IN_{H1}}{\partial \omega_1}$$

$$\frac{\partial E_{Total}}{\partial w_1} = \left(\sum(\delta_O \times w_{ho})\right) \times \frac{\partial OUT_{H1}}{\partial IN_{H1}} \times \frac{\partial IN_{H1}}{\partial w_1}$$

$$w_1^* = 0.3493$$
$$w_2^* = 0.1464$$
$$w_3^* = 0.1997$$
$$w_4^* = 0.3987$$

***Check the matlab code for detail steps and calculations!!**

Delta Rule

In machine learning, the delta rule is a gradient descent learning rule for updating the weight of the inputs to artificial neurons in a single-layer neural network.

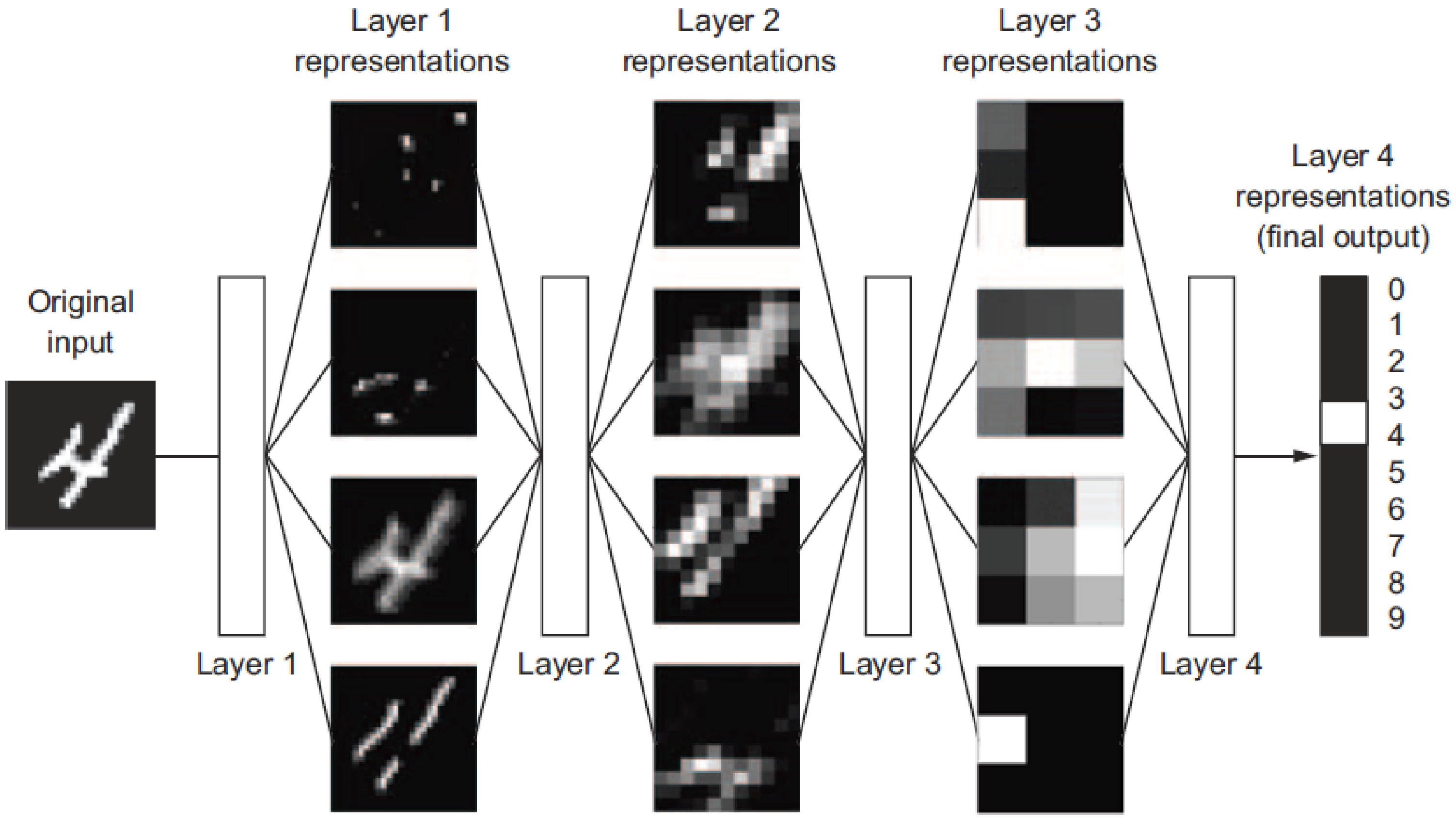
$$\Delta w_{ij} = \alpha(t_j - y_i)g'(h_j)x_i$$

where

α is a small constant called learning rate
$g(x)$ is the neuraon's activiation function
t_j is the target output
h_j is the weighted sum of the neuron's inputs
y_j is the catual output
x_i is the *i*th input

Deep Learning

- Deep learning: a neural network has more than two hidden layers.
- A multistage information-distillation operation.



Machine Learning

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evolution

- Accuracy
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

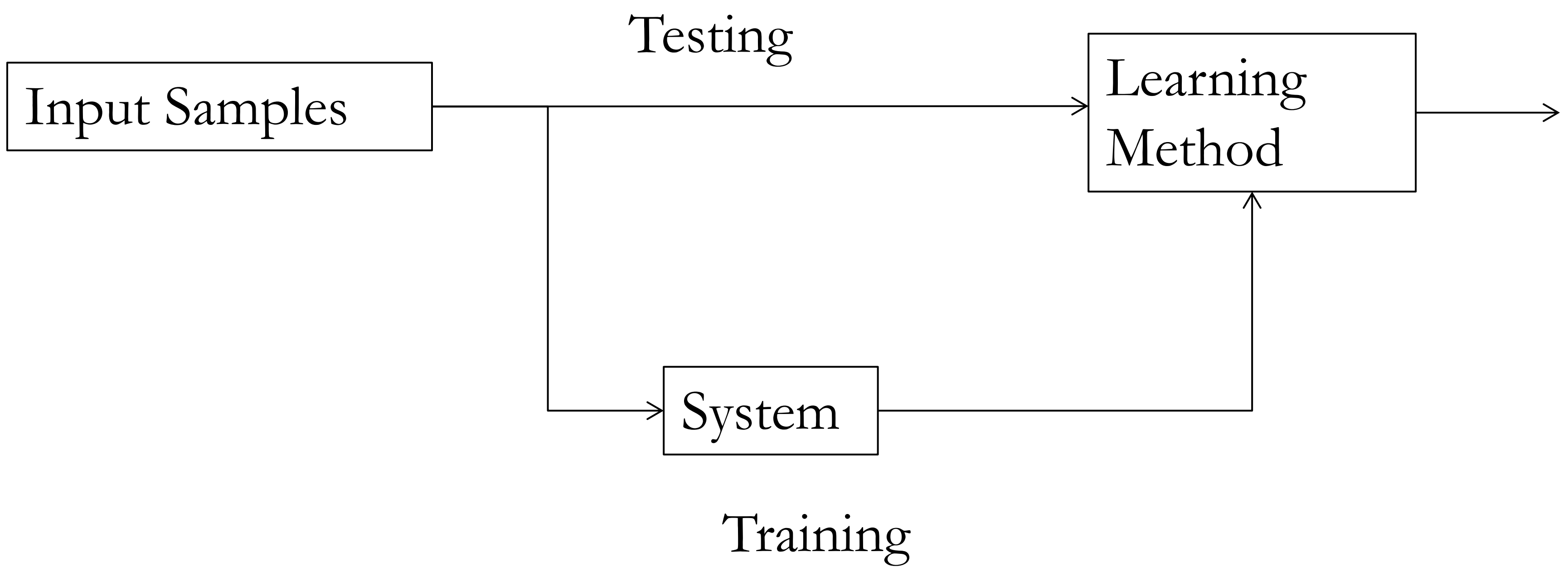
Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Algorithms

- **Supervised learning** ($\{x_n \in R^d, y_n \in R\}_{n=1}^N$
  - Prediction
  - Classification (discrete labels), Regression (real values)
- **Unsupervised learning** ($\{x_n \in R^d\}_{n=1}^N$)
  - Clustering
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction
- **Semi-supervised learning**
- **Reinforcement learning**
  - Decision making (robot, chess machine)

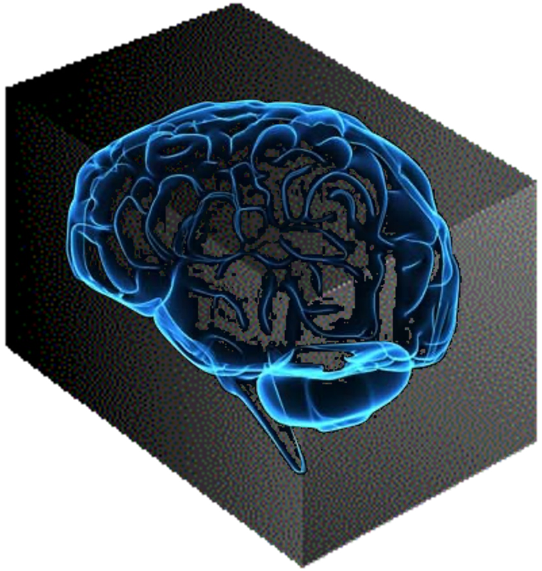
Learning System Model



Genetic Algorithm

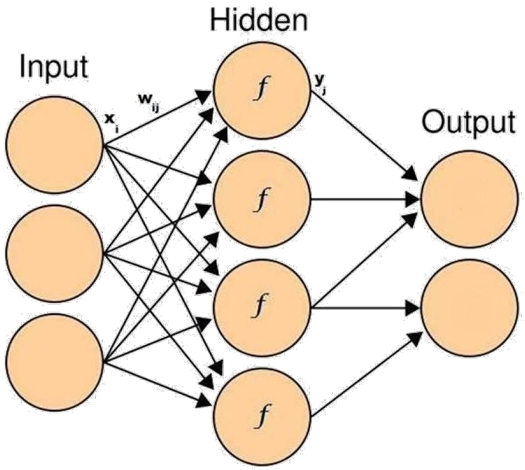
Introduction

Inputs



Outputs

Reverse Engineering



$$f(\mathbf{x}) = y_j = \frac{1}{1 + \exp(-\sum_{i=1}^{n} w_{ij} x_i)}$$

Artificial Intelligence:

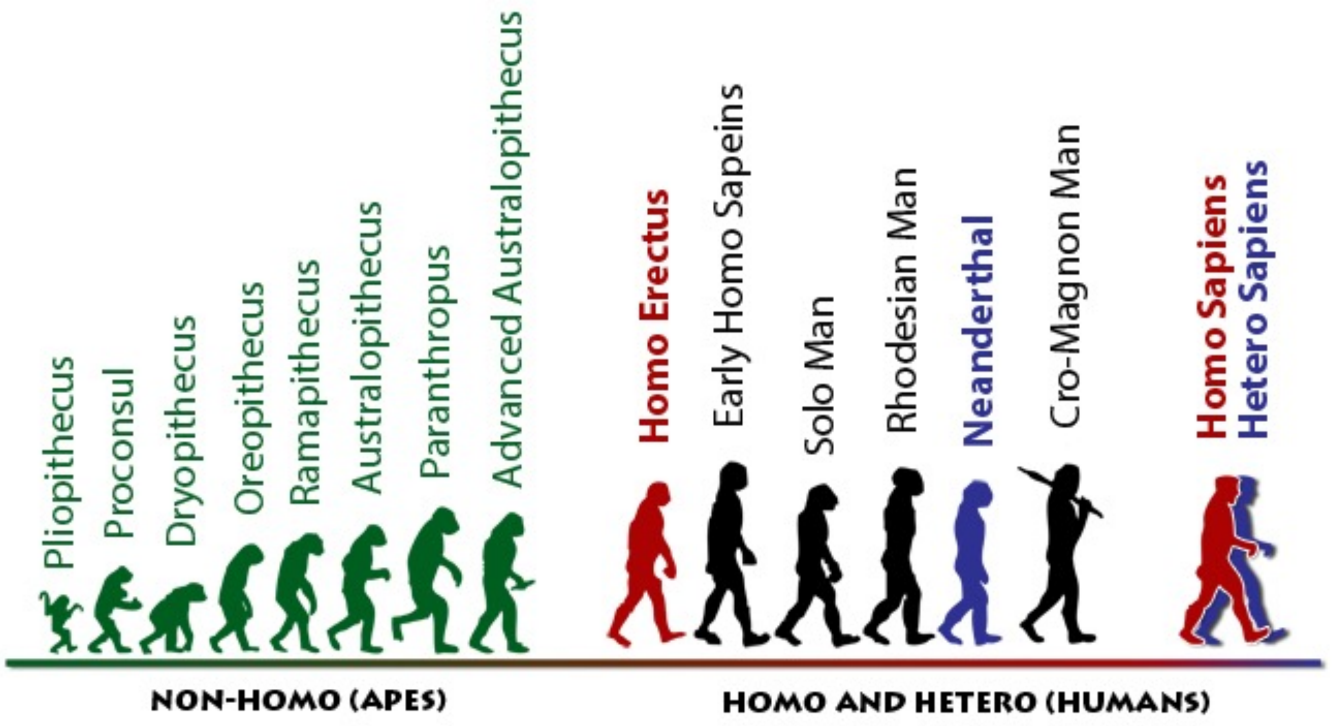
Capable of human-like qualities

Evolutionary Algorithm (EA)

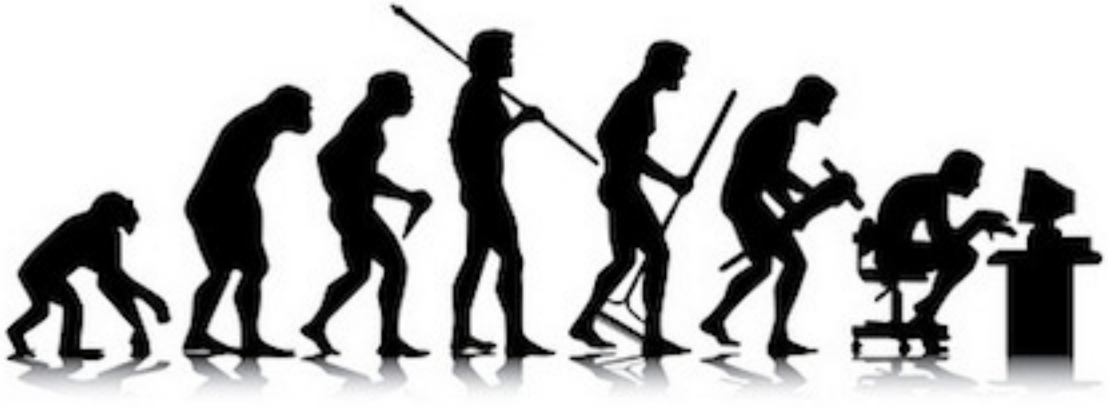
EA attempts to simulate the process of evolution.

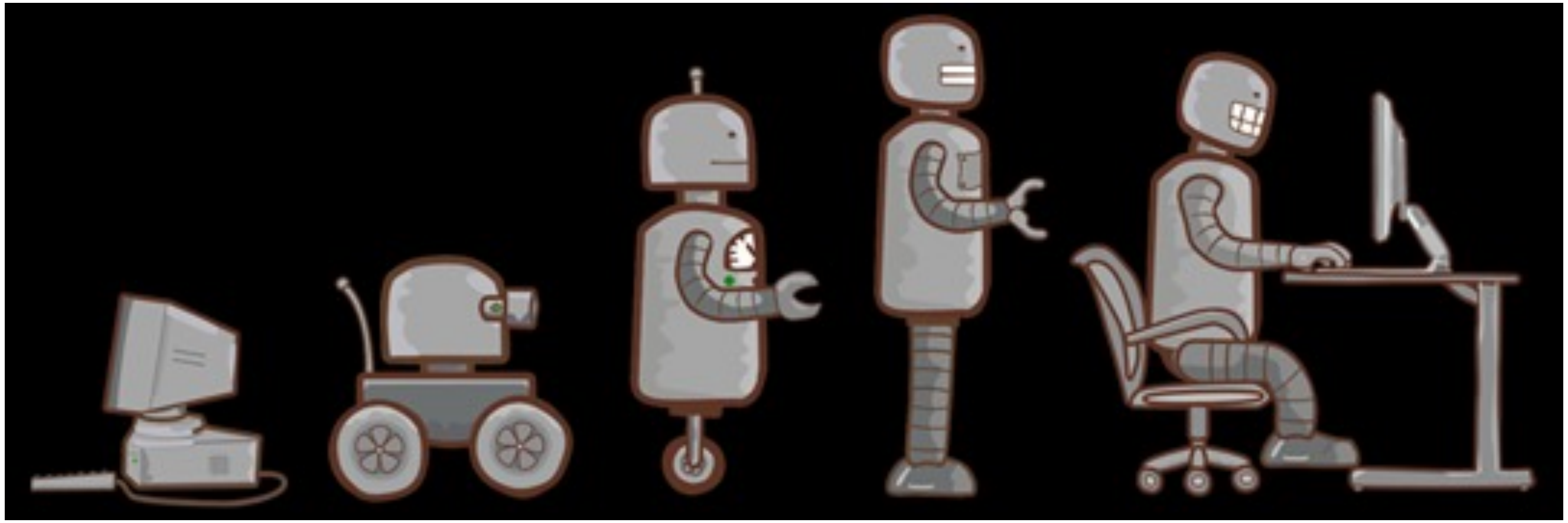
Understanding the system (e.g., human brain)

The driving force behind the creation/evolution



Evolution of Human and AI





History (Inheritance)

Charles Darwin: *Natural Selection*

- Huge lizards, strange birds and Giant Tortoises Galapagos Island.

The Discovery of Inheritance

Original (true) Generation (F0)

Parent *a*	Parent *b*
TT	**ss**
(tall)	(small)

First Filial generation (F1)

Child *a*	Child *b*
Ts	**Ts**
(tall)	(tall)

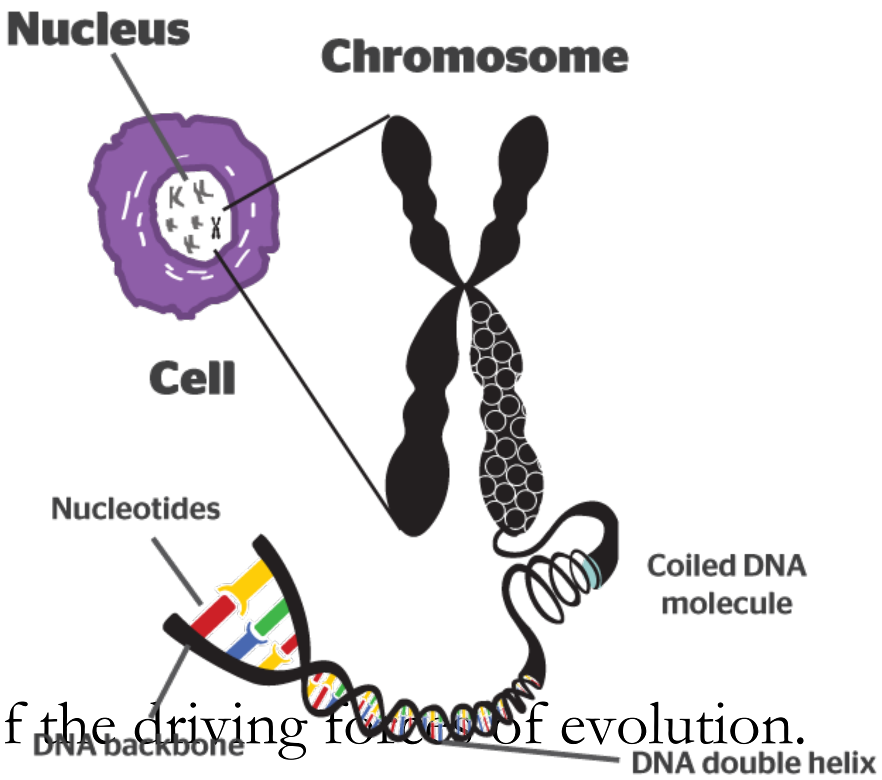
Second Filial generation (F2)

Grandchild *a*	Grandchild *b*	Grandchild *c*	Grandchild *d*
TT	**Ts**	**sT**	**ss**
(tall)	(tall)	(tall)	(small)

Factors were inherited in pairs, one from each parent plant.

T is dominant and S is regressive

The Discovery of Chromosomes and DNA (1903)



Mutation (1920's): one of the driving forces of evolution.

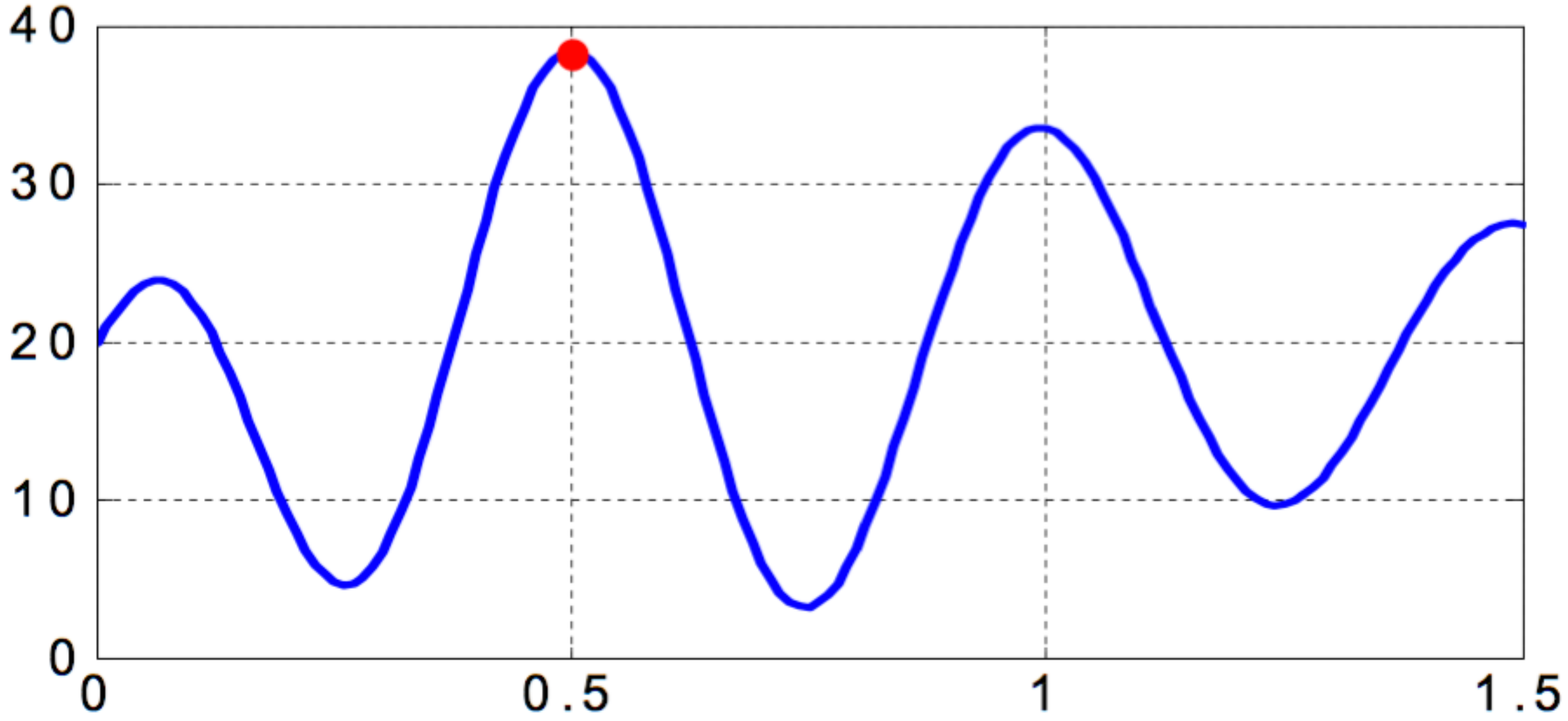
Genetic Algorithm (GA)

GA: Artificial version of Biological Evolution, allowing the fittest to survive while killing off the weakest.

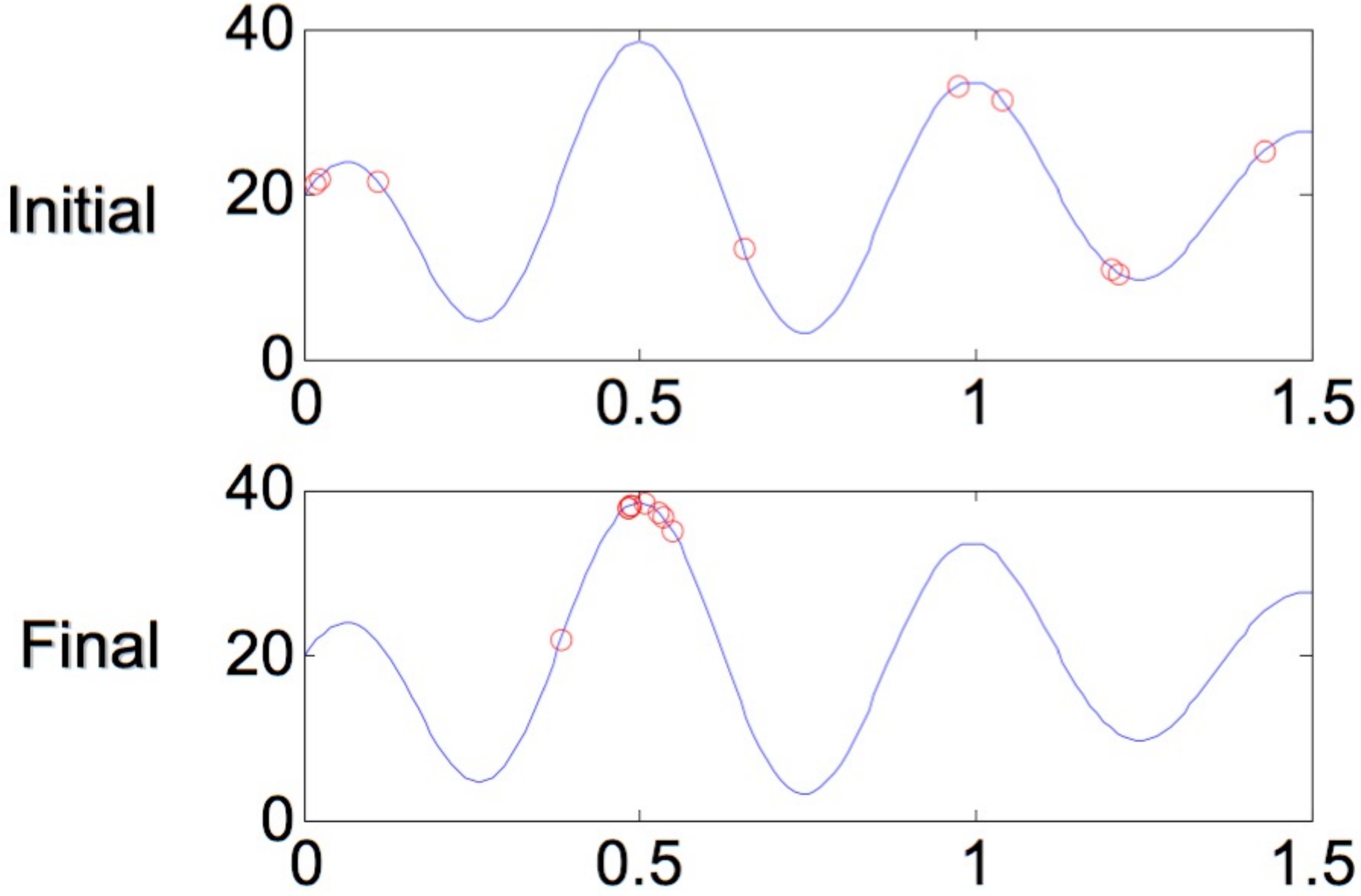
- Stochastic optimization technique
- Ability to escape from local optimal solutions (Gradient methods do not have this property.)
- The algorithm consists of:
  a) coding the problem
  b) generating an initial population
  c) evaluating fitness
  d) crossover (breeding) and
  e) mutation

An Example

Solve: $\max f(x), f(x) = 20 + 100x\cos(4\pi x)e^{-2x} \quad x \in [0,1.5]$



Distributions of Chromosomes



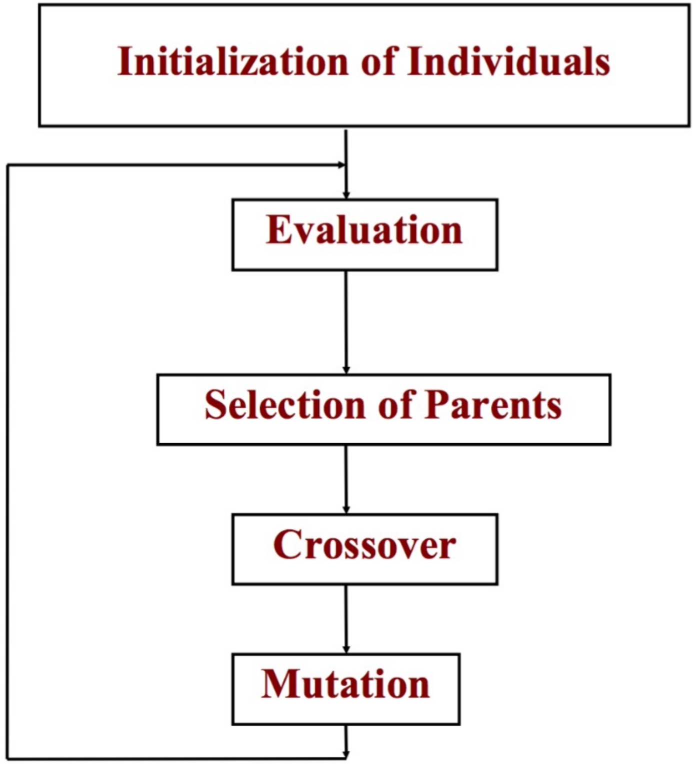
Procedure of GA

- **Selection:** Selection of fit individuals for reproduction
- **Crossover:** Mating of selected individuals for reproduction
- **Mutation:** Introduction of new alleles into chromosomes in the population, to create completely new solutions

i. New population is produced by mating the best individuals

ii. Over generations, desirable characteristics are spread throughout population

iii. Mutation is used to escape from a local minimum



Coding and Generating an Initial Population

Generate randomly initial population of N (=10) chromosomes. (required precision: 3 decimal points)
Note: Population size 2^n; n=number of don't care genes

decimal	0.000	$\leftrightarrow$	1.500
binary	00000000000	$\leftrightarrow$	11111111111

chromosome	binary encoding	decimal value
x_1	00010010100	0.1085
x_2	11001100111	1.2010
x_3	11001101001	1.2025
x_4	10100110001	0.9739
x_5	11001110111	1.2128
x_6	01101111101	0.6544
x_7	00000010110	0.0161
x_8	11110100000	1.4304
x_9	10110001011	1.0398
x_{10}	00000011110	0.0220

Evaluation Fitness

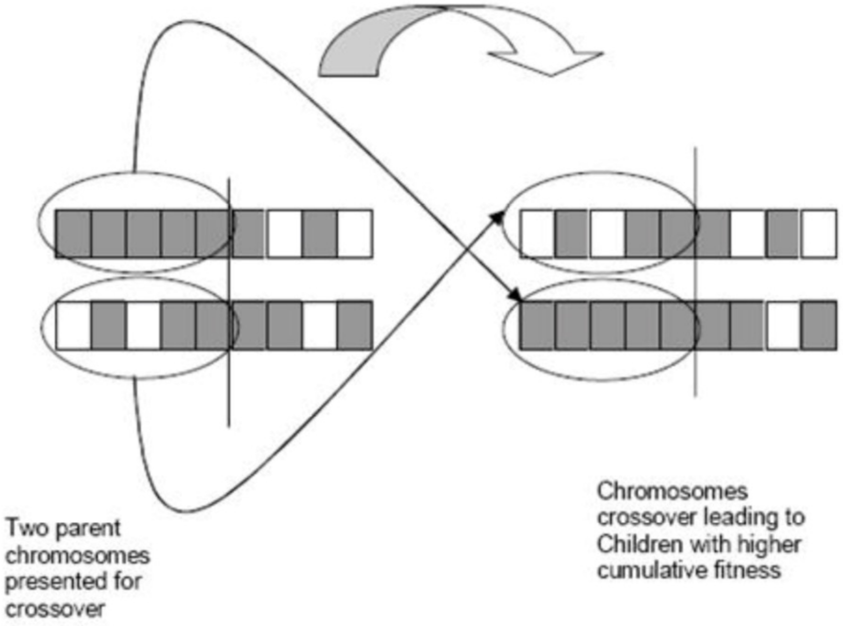
Calculate fitness values for the chromosomes:

$$f(x) = 20 + 100x\cos(4\pi x)e^{-2x}, \qquad i = 1,2,\dots 10$$

chromosome	binary encoding	decimal value	fitness value
x_1	00010010100	0.1085	21.8025
x_2	11001100111	1.2010	11.1218
x_3	11001101001	1.2025	11.0231
x_4	10100110001	0.9739	33.1448
x_5	11001110111	1.2128	10.4288
x_6	01101111101	0.6544	13.6220
x_7	00000010110	0.0161	21.5290
x_8	11110100000	1.4304	25.2480
x_9	10110001011	1.0398	31.4024
x_{10}	00000011110	0.0220	22.0240

Crossover

- **Crossover:** Exchanges some genes of the two parents to create the genotypes of the offspring
- **Method:** Select points along parents' chromosomes (randomly) and exchange genes between these points
  Note: In Simple Crossover , only one point is chosen. See Figure

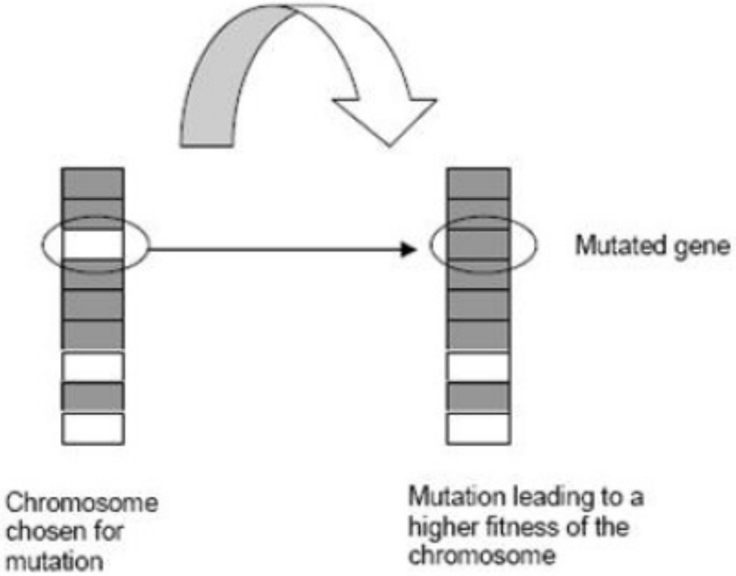


Mutation

Introduces completely new alleles into a population of chromosomes Creates completely new solutions (avoids stagnation)

Method: Select one or more genes in an individual at random and change their alleles

Note: Allele change itself can be random or deterministic fashion



Selection

Select N chromosomes in the next generation (t+1) from N chromosomes in the current generation (t). Compute the probability of x_i being selected: $P(x_i) := \frac{f(x_i)}{\sum_{j=1}^{10} f(x_i)}$

chromosome	binary encoding	decimal value	fitness value	$p(x_i)$
x_1	00010010100	0.1085	21.8025	0.1083
x_2	11001100111	1.2010	11.1218	0.0552
x_3	11001101001	1.2025	11.0231	0.0547
x_4	10100110001	0.9739	33.1448	0.1646
x_5	11001110111	1.2128	10.4288	0.0518
x_6	01101111101	0.6544	13.6220	0.0677
x_7	00000010110	0.0161	21.5290	0.1069
x_8	11110100000	1.4304	25.2480	0.1254
x_9	10110001011	1.0398	31.4024	0.1560
x_{10}	00000011110	0.0220	22.0240	0.1094

After Selection

chromosome	binary encoding	decimal value	fitness value
x_1	00000011110	0.0220	22.0240
x_2	00000011110	0.0220	22.0240
x_3	11001101001	1.2025	11.0231
x_4	10100110001	0.9739	33.1448
x_5	10110001011	1.0398	31.4024
x_6	10110001011	1.0398	31.4024
x_7	10100110001	0.9739	33.1448
x_8	10110001011	1.0398	31.4024
x_9	10100110001	0.9739	33.1448
x_{10}	00000010110	0.0161	21.5290

Generic Operation

Crossover: Exchange some genes of two parents

```
parent 1   1010011011
parent 2   0101110010
                ⇓
 child 1   1010010010
 child 2   0101111011
```

Mutation: Exchange some

```
before   1010010010
            ↓    ⇓
 after   1110010010
```

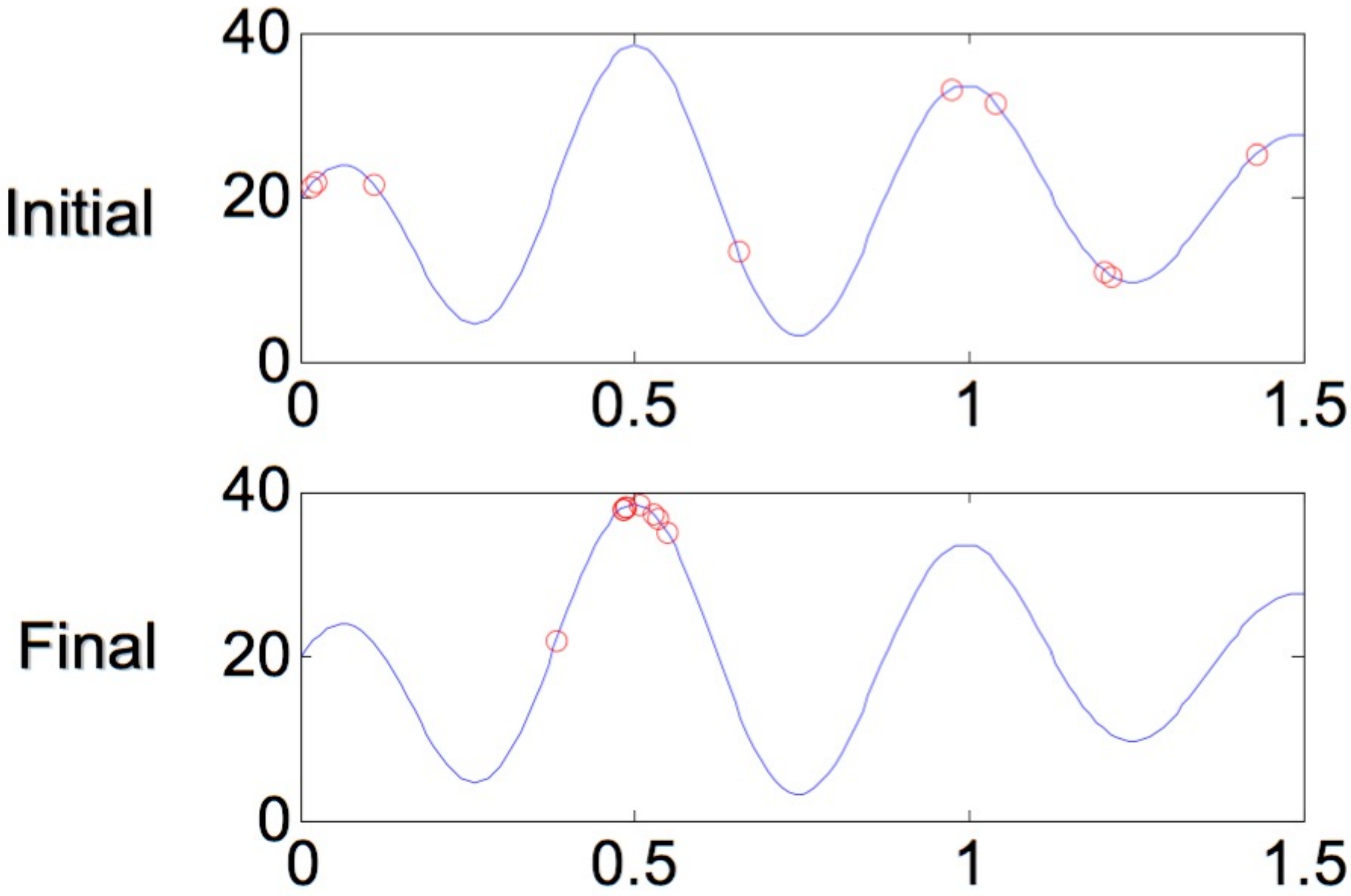
After Genetic Operation

chromosome	binary encoding	decimal value	fitness value
x_1	00000001011	0.0081	20.7891
x_2	00000011110	0.0220	22.0240
x_3	11001101001	1.2025	11.0231
x_4	10100110001	0.9739	33.1448
x_5	10110011110	1.0537	29.9967
x_6	10110001011	1.0398	31.4024
x_7	10100110001	0.9739	33.1448
x_8	10110001011	1.0398	31.4024
x_9	10100110001	0.9739	33.1448
x_{10}	00000110110	0.0396	23.2132

After 16 Generations

chromosome	binary encoding	decimal value	fitness value
x_1	01010010001	0.4814	37.8831
x_2	01011101011	0.5474	35.1638
x_3	01010010001	0.4814	37.8831
x_4	01010010001	0.4814	37.8831
x_5	01011011001	0.5342	36.6842
x_6	01011011001	0.4873	38.1542
x_7	01010101011	0.5050	38.3936
x_8	01011010001	0.5283	37.2136
x_9	01000011001	0.3835	24.1274
x_{10}	01010011001	0.4873	38.1542

Distributions of Chromosomes



The End!!