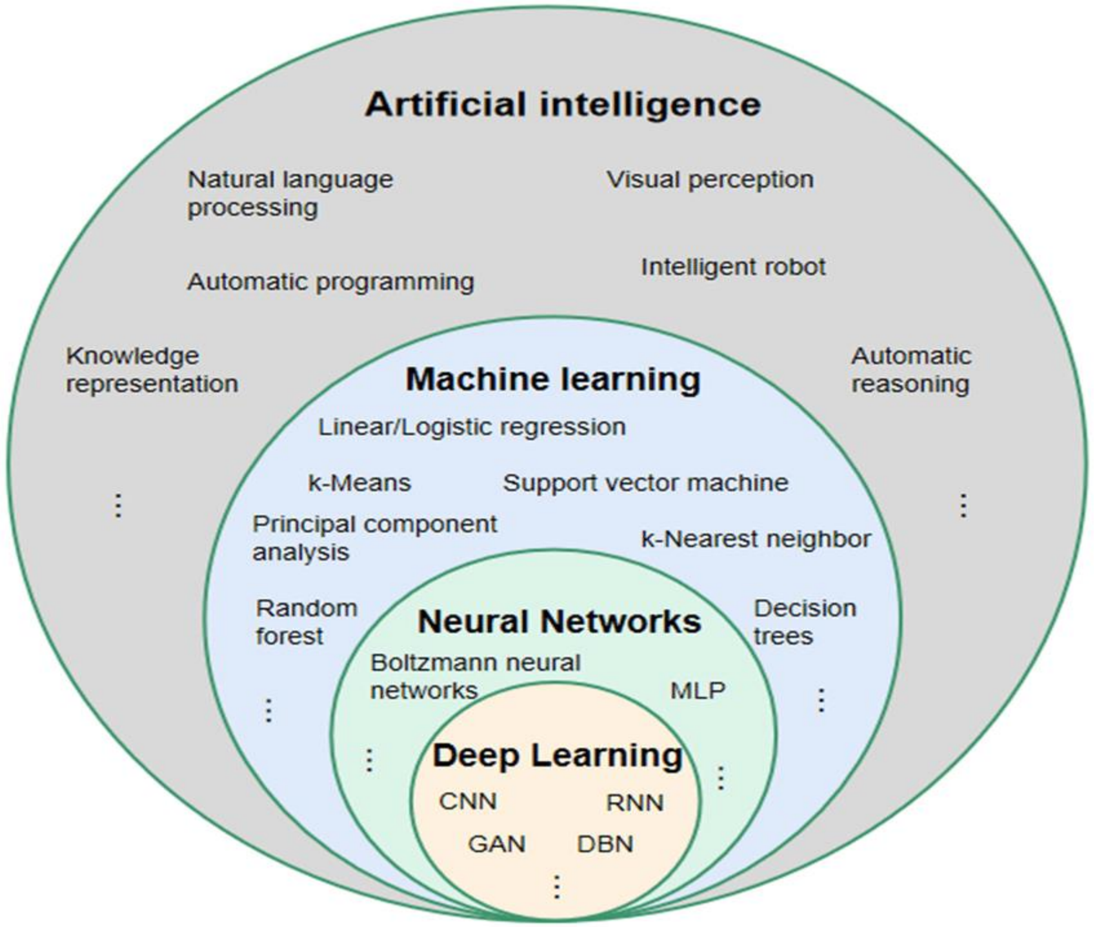
Neural Network, Genetic Algorithm, and Planning

Machine Learning



Machine Learning

- Machine learning is *a subset of AI, which uses algorithms that learn from data to make predictions*.
- Machine learning is a *subfield of artificial intelligence that gives computers the ability to learn* without explicitly being programmed

Categories of Machine Learning:

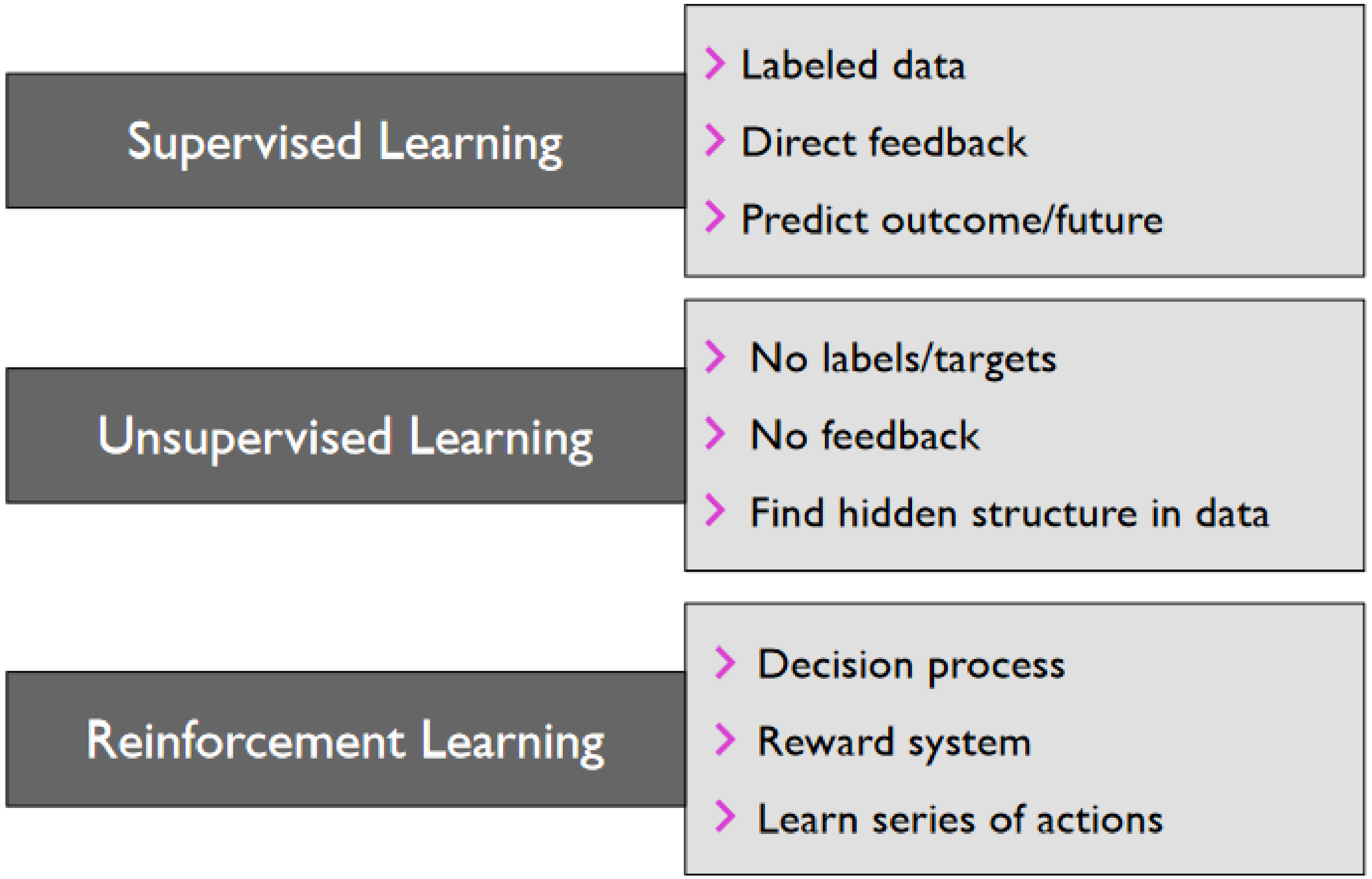
The three broad categories of machine learning are

(i) Supervised learning

(ii)Unsupervised learning

(iii) Reinforcement learning

Machine Learning Categories:



Supervised Learning	Labeled data Direct feedback Predict outcome/future
Unsupervised Learning	No labels/targets No feedback Find hidden structure in data
Reinforcement Learning	Decision process Reward system Learn series of actions

Supervised machine Learning:

Supervised learning is the subcategory of machine learning that focuses on learning a classification or regression model, that is, learning from labeled training data (i.e., inputs that also contain the desired outputs or targets; basically, “examples” of what we want to predict)

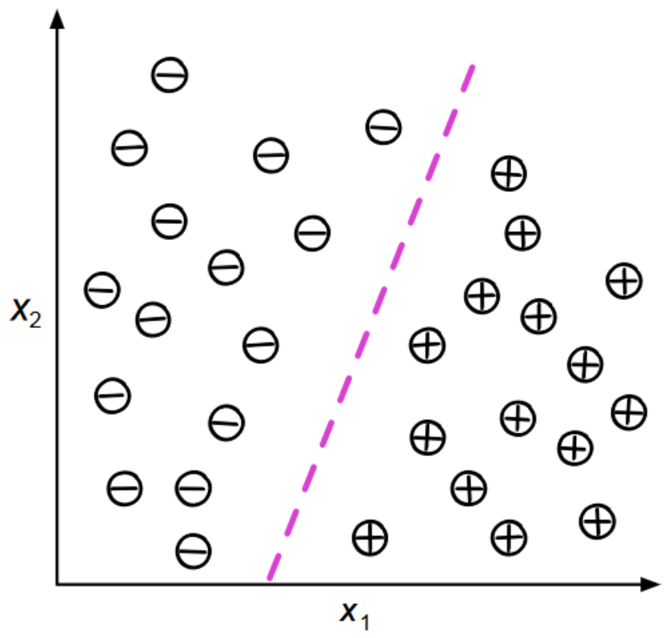


Figure: Illustration of a binary classification problem (plus, minus) and two feature variable (x1 and x2).

Unsupervised machine Learning:

In contrast to supervised learning, unsupervised learning is a branch of machine learning that is concerned with unlabeled data. Common tasks in unsupervised learning are clustering analysis (assigning group memberships) and dimensionality reduction (compressing data onto a lower-dimensional subspace or manifold)

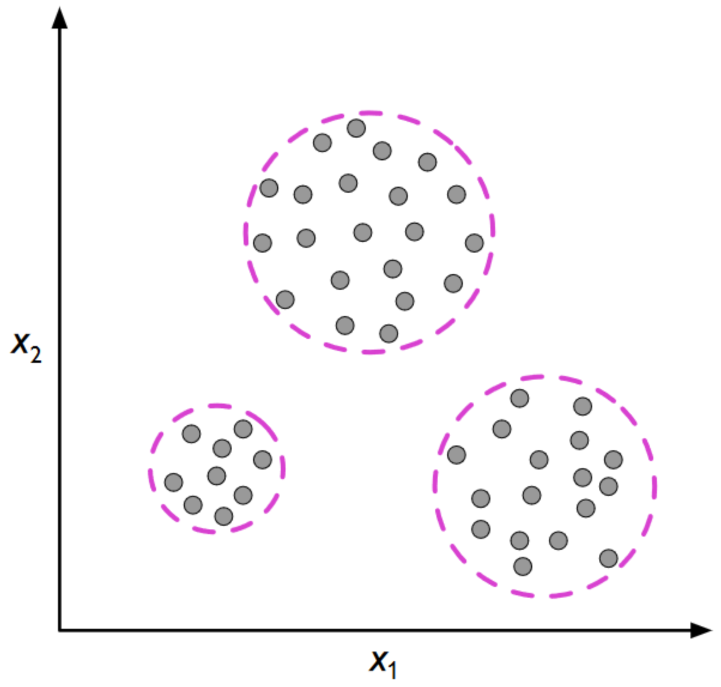


Figure: Illustration of clustering, where the dashed lines indicate potential group membership assignments of unlabeled data points

Machine Learning Examples

Supervised Learning:

- kNN (k Nearest Neighbors)
- Naïve Bayes
- Linear + Logistic Regression
- Support Vector Machines
- Random Forests
- Neural Networks

Unsupervised Learning:

- Clustering
- Matrix Factorization
- HMMs (Hidden Markov Models)
- Neural Networks

Supervised vs Unsupervised ML

Supervised ML	**Unsupervised ML**
• Data has known labels or output • Training data includes both the input and the desired results	• Data does not have known labels or output • Training data does NOT includes the desired results for the input.
Divided into two types: • Classification • Regression	• Clustering/Segmentation • Dimensionality Reduction • Association
• Used for Prediction	• Descriptive results about the dataset • Summary of data distribution
Algorithms: Linear regression, Logistic regression, Support vector machine, Random forest, Decision Tree and ANN	Algorithms: Clustering, K-means clustering, Association and Apriori algorithm

Supervised vs Unsupervised ANN

- Task performed

  Classification

  Pattern Recognition

- NN model :

  Preceptron

  Feed-forward NN

"What is the class of this data point?"

- Task performed

  Clustering

- NN Model :

  Self Organizing Maps

"What groupings exist in this data?"

"How is each data point related to the data set as a whole?"

Types of Neural Networks

- Feedforward Neural Network (FNN)
- Recurrent Neural Network (RNN)
- Convolutional Neural Network(CNN)
- Single Layer or Multilayer Perceptron (MLP)
- Long Short-Term Memory (LSTM)
- Hopfield Network
- Kohonen Self-Organizing Map (SOM):
- Generative Adversarial Network (GAN)
- Deep Belief Networks (DBN)
- Residual Neural Network (ResNet)
- Boltzmann Machine
- Transformers
- Autoencoder

Application of Neural Network

- Facial Recognition
- Aerospace
- Healthcare
- Weather Forecasting
- Automotive
- Defense
- Banking
- Manufacturing
- Financial

Application of Machine Learning

Popular applications of machine learning include the following:

- Email spam detection
- Face detection and matching (e.g., iPhone X)
- Web search (e.g., DuckDuckGo, Bing, Google)
- Sports predictions
- Credit card fraud
- Stock predictions
- Smart assistants (Apple Siri, Amazon Alexa, . . .)
- Product recommendations (e.g., Netflix, Amazon)
- Self-driving cars (e.g., Uber, Tesla)
- Language translation (Google translate)
- Sentiment analysis
- Drug design
- Medical diagnoses

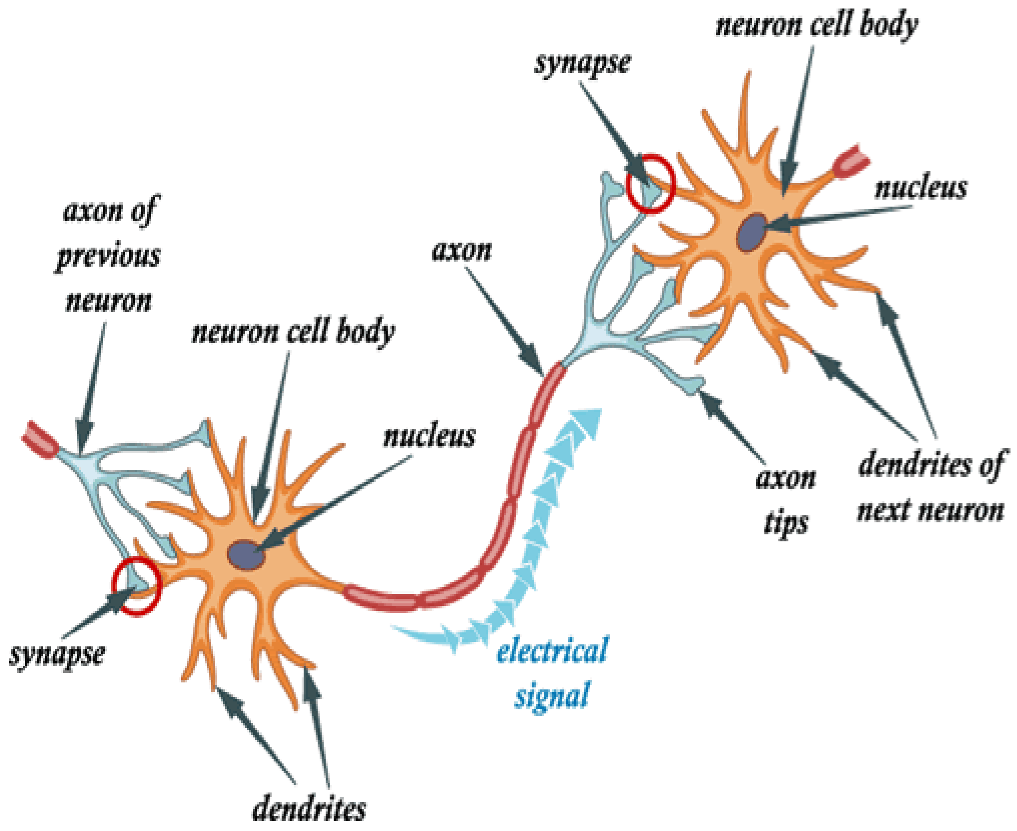
Artificial Neural Networks (NN/ ANN)

- Biological Inspiration (How the Brain Works)
- Perceptron
- Multi-Layer Perceptron (MLP)
- Feed Forward Network
- Back Propagation algorithm
- Application of Neural Networks

How Brain Works

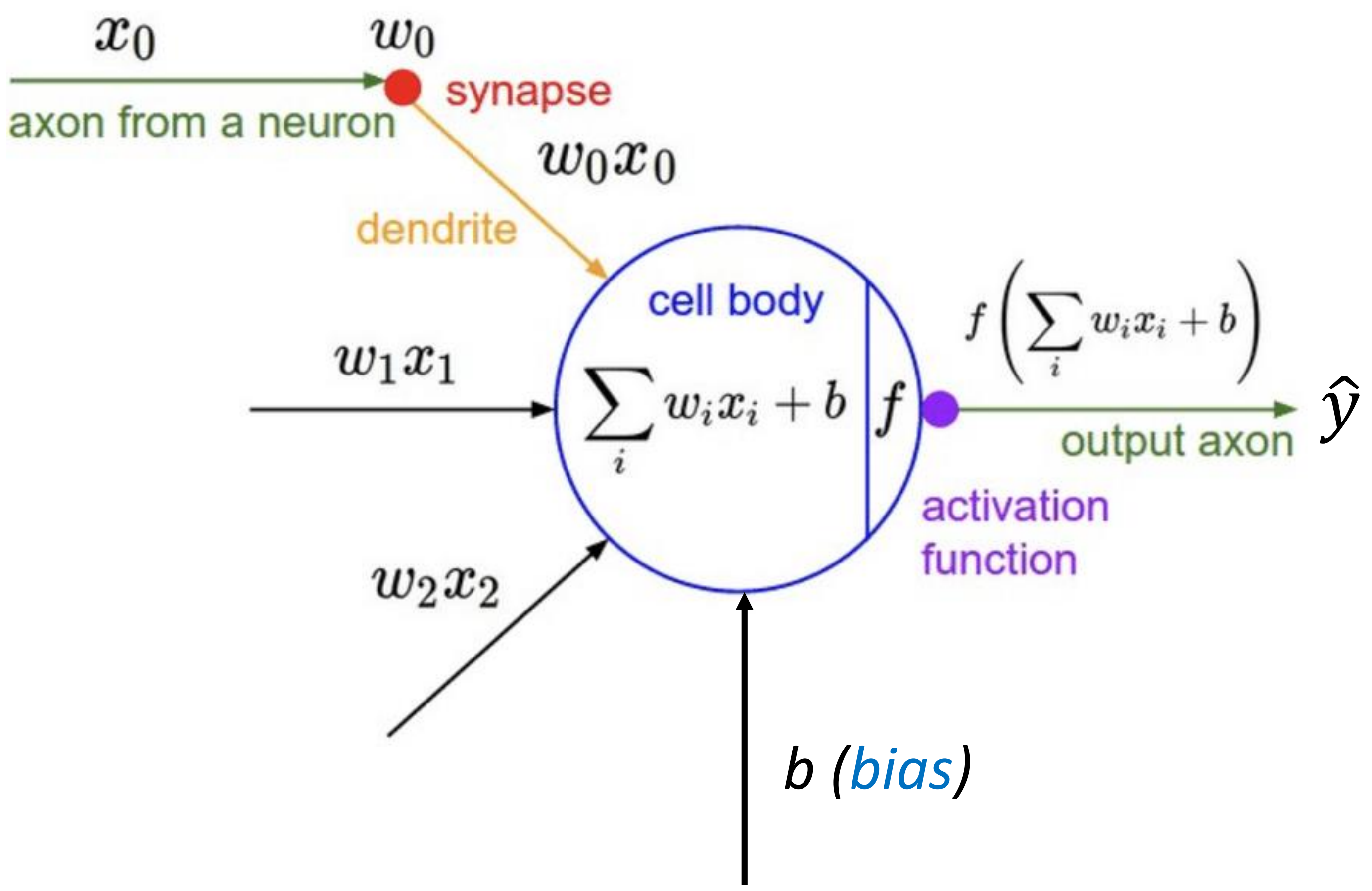
- *Many machine learning methods inspired by biology, e.g., the (human) brain*
- *Our brain has $\sim 10^{11}$ neurons, each of which communicates (is connected)to $\sim 10^{4}$ other neurons.*

Each neuron has many components.
Dendrites: receive inputs from other neurons.
Soma (cell body): controls activity of the neuron.
Axon: sends outputs to other neurons.
Synapse: links neurons, it is the junction between the *axon* of one neuron and the *dendrite* of another, through which the two *neurons* communicate.



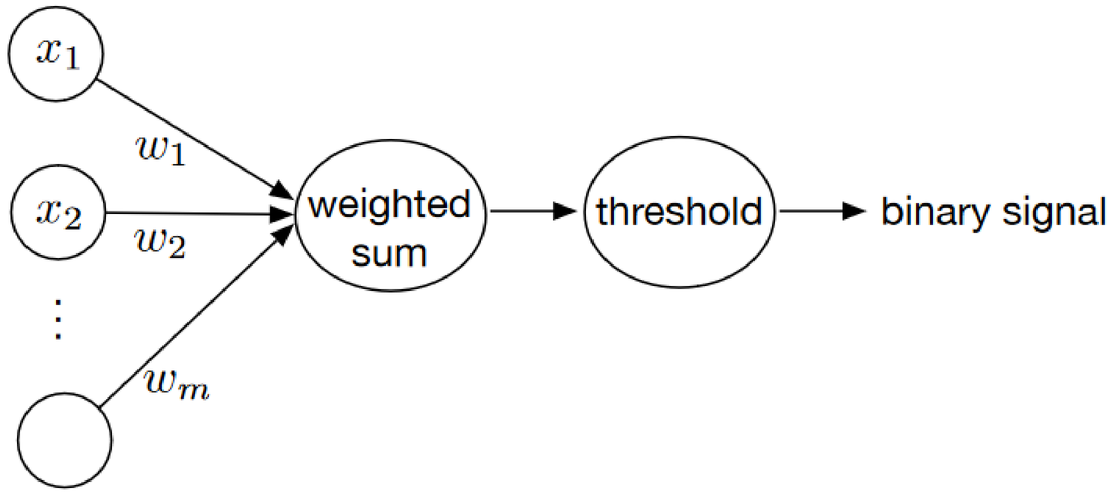
Mathematical Model of a neuron

$$\boxed{\hat{y} = \boldsymbol{\omega x} + b}$$



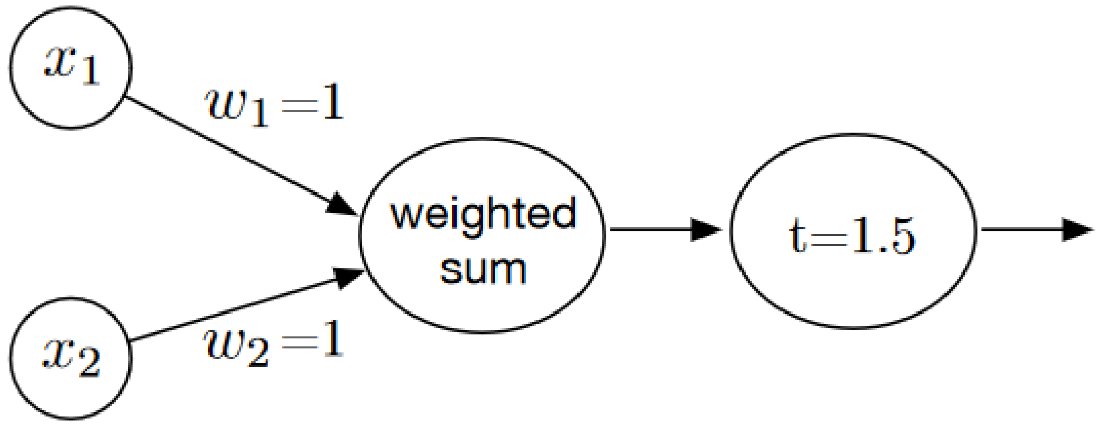
McCulloch & Pitts Neuron Model

- In 1943, Warren McCulloch and Walter Pitts introduced one of the first artificial neurons.
- The main feature of their neuron model is that a *weighted sum of input signals is compared to a threshold to determine the neuron output*.
- When the sum is greater than or equal to the threshold, the output is 1.
- When the sum is less than the threshold, the output is 0.



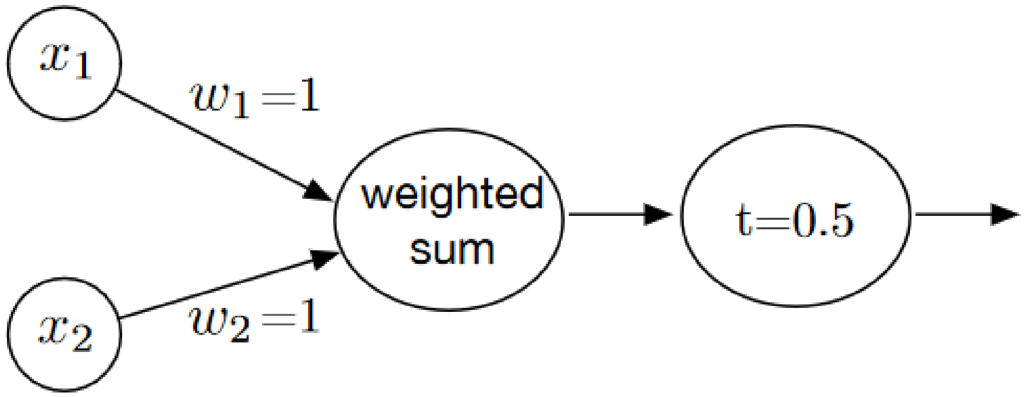
McCulloch & Pitts Neuron Model (Logical AND Gate)

x_1	x_2	Out
0	0	0
0	1	0
1	0	0
1	1	1



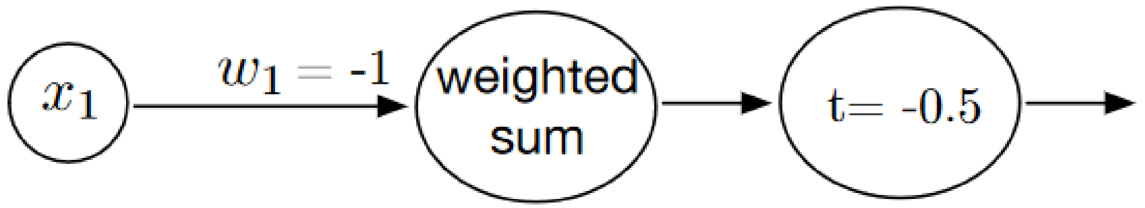
McCulloch & Pitts Neuron Model (Logical OR Gate)

x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	1



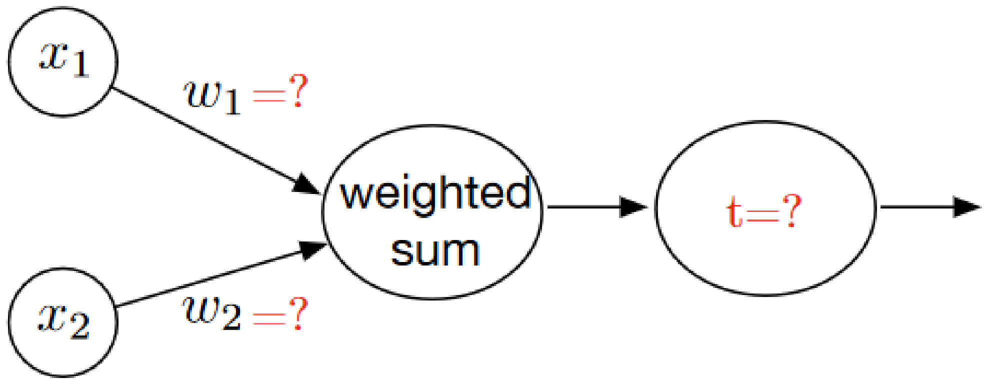
McCulloch & Pitts Neuron Model (Logical NOT Gate)

x_1	Out
0	1
1	0


$$\begin{tikzcd} x_1 \arrow[r, "w_1 = -1"] & \text{weighted sum} \arrow[r] & t = -0.5 \arrow[r] & {} \end{tikzcd}$$

McCulloch & Pitts Neuron Model (Logical XOR Gate)

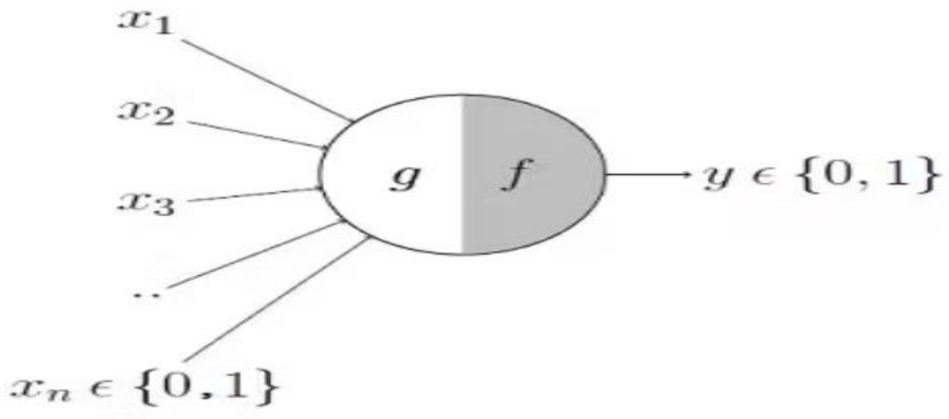
x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	0



McCulloch & Pitts Neuron Model

Key features of the McCulloch-Pitts model:

1. **Binary Output:** The model produces a binary output, which means it can be either active (firing) with a value of 1 or inactive (not firing) with a value of 0.
2. **Inputs and Weights:** It takes multiple binary inputs, each with its associated weight. The inputs can be either 0 or 1, and the weights represent the strength of the connection between inputs and the neuron.
3. **Threshold:** The model includes a threshold value (often denoted as "θ" or "bias"). The neuron will only fire (output 1) if the weighted sum of its inputs exceeds this threshold.



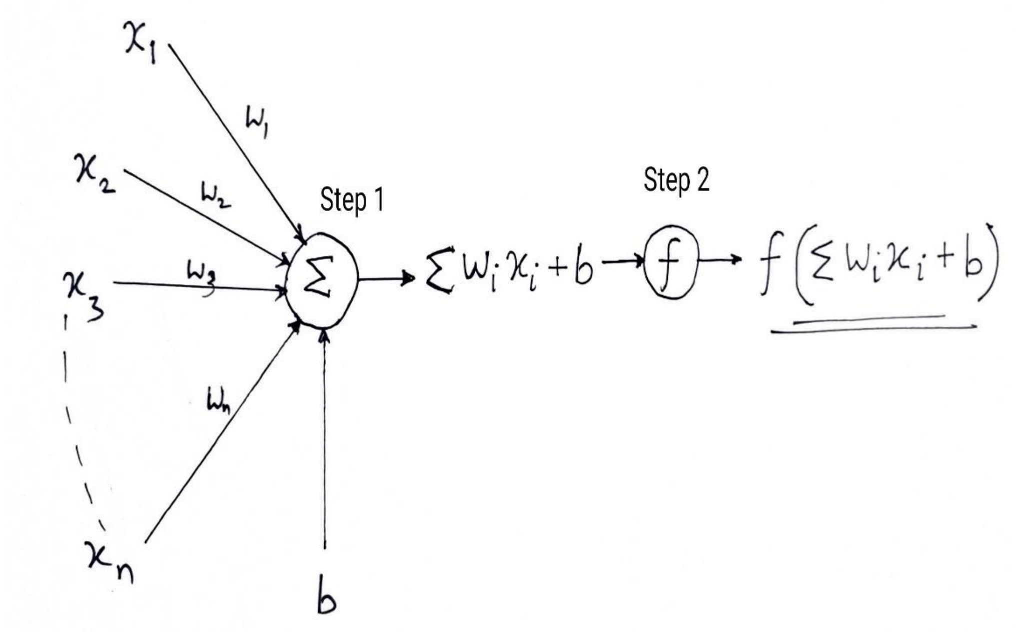
Perceptron

- *Perceptrons* are the *building blocks of neural networks*
- *Perceptron is a binary classifier*

There are two types of Perceptrons:

i) Single layer: Single layer Perceptrons can learn only linearly separable patterns.

ii) Multilayer : Multilayer Perceptrons (MLP) or feedforward neural networks with two or more layers.



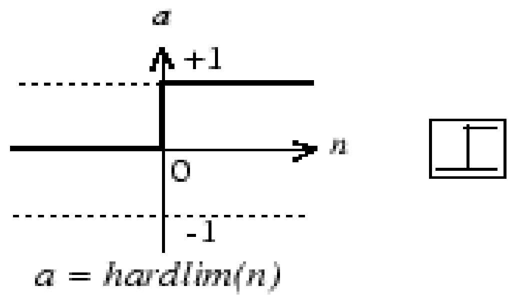
Common activation functions used in perceptrons include the **step function, sigmoid function, and ReLU function**.

Perceptron

$$\boxed{\hat{y} = \boldsymbol{\omega x} + b}$$

The <span>hard-limit transfer function</span> is used to gives a perceptron the ability to classify input vectors by dividing the input space into two regions.

$$\hat{y} = \begin{cases} 1, & if \sum \boldsymbol{\omega x} + b > 0 \\ 0, & if \sum \boldsymbol{\omega x} + b < 0 \end{cases}$$



Hard-Limit Transfer Function

$$\hat{y} = \begin{cases} 1, & if \sum \boldsymbol{\omega x} \prec -\mathrm{b} \\ 0, & if \sum \boldsymbol{\omega x} < -\mathrm{b} \end{cases}$$

$$\boxed{\begin{aligned} \hat{y} &= hardlim\,(\boldsymbol{wx} + b) \\ &= 1, \quad if\ \boldsymbol{wx} + b > 0 \\ &= 0, \quad Otherwise \end{aligned}}$$

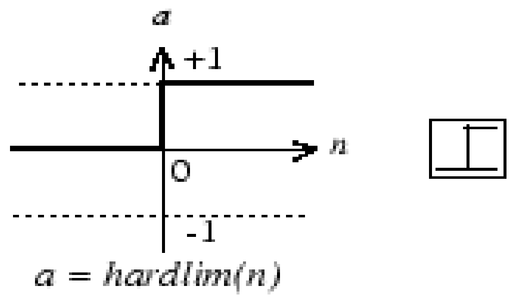
i.e. threshold is the negative of “bias”

Perceptron

$$\boxed{\hat{y} = \boldsymbol{\omega x} + b}$$

The **hard-limit transfer function** is used to gives a perceptron the ability to classify input vectors by dividing the input space into two regions.

$$\hat{y} = \begin{cases} 1, & if \sum \boldsymbol{\omega x} + b > 0 \\ 0, & if \sum \boldsymbol{\omega x} + b < 0 \end{cases}$$



Hard-Limit Transfer Function

$$\hat{y} = \begin{cases} 1, & if \sum \boldsymbol{\omega x} \prec -\mathrm{b} \\ 0, & if \sum \boldsymbol{\omega x} < -\mathrm{b} \end{cases}$$

$$\boxed{\begin{aligned} \hat{y} &= hardlim\,(\boldsymbol{wx} + b) \\ &= 1, \qquad if\ \boldsymbol{wx} + b > 0 \\ &= 0, \qquad Otherwise \end{aligned}}$$

i.e. threshold is the negative of "bias"

Perceptron

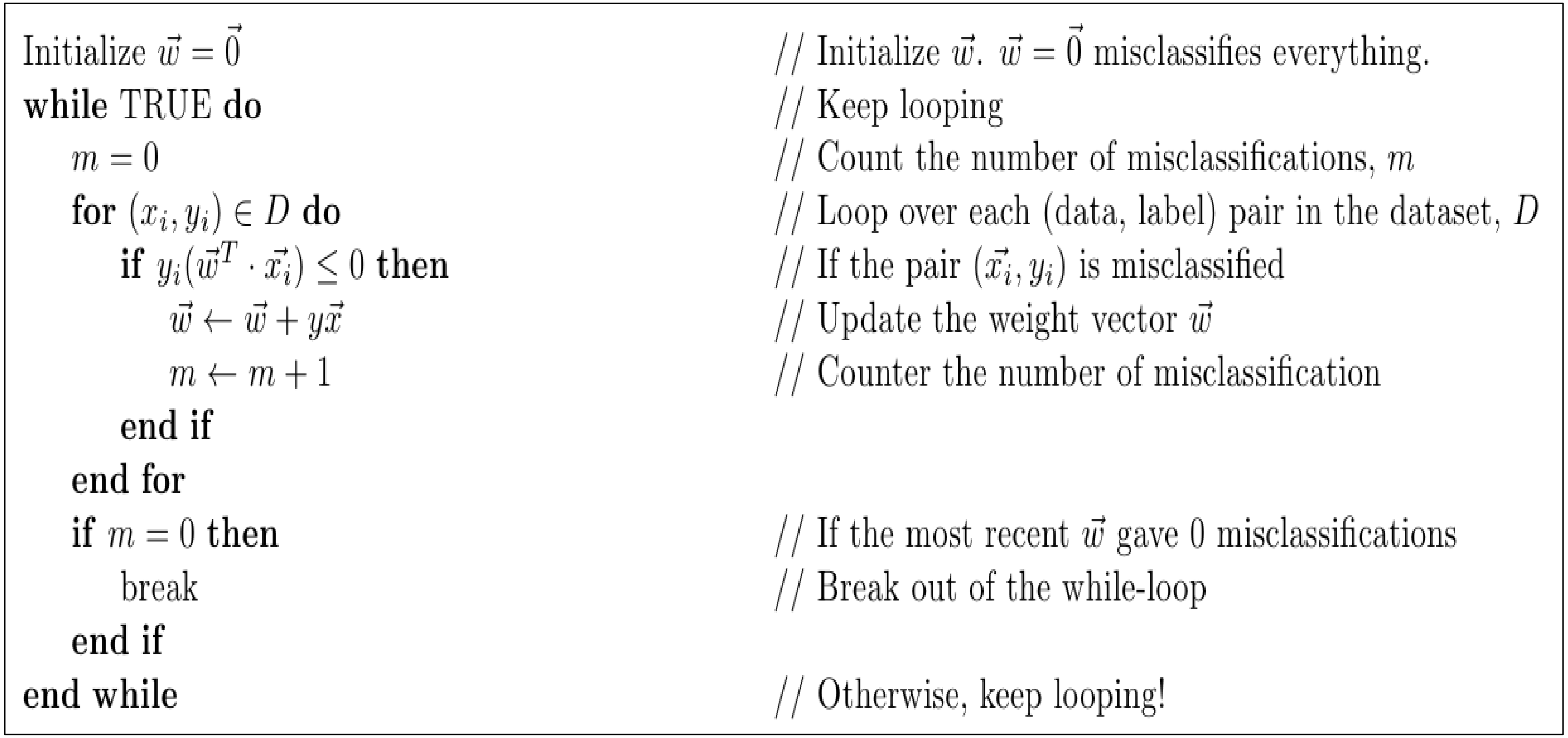
- In the late 1950s, Frank Rosenblatt and several other researchers developed a class of neural networks called perceptrons.
- The neurons in these networks were similar to those of McCulloch and Pitts.
- Rosenblatt's key contribution was the introduction of a learning rule for training perceptron networks to solve pattern recognition problems. He proved that his learning rule will always converge to the correct network weight

Perceptron Learning Rule:

- Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- The input features are then multiplied with these weights to determine if a neuron fires or not.
- The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output.

Perceptron

Perceptron Algorithm:



```
Initialize $\vec{w} = \vec{0}$                      // Initialize $\vec{w}$. $\vec{w} = \vec{0}$ misclassifies everything.
while TRUE do                                       // Keep looping
    $m = 0$                                         // Count the number of misclassifications, $m$
    for $(x_i, y_i) \in D$ do                       // Loop over each (data, label) pair in the dataset, $D$
        if $y_i(\vec{w}^T \cdot \vec{x_i}) \leq 0$ then   // If the pair $(\vec{x_i}, y_i)$ is misclassified
            $\vec{w} \leftarrow \vec{w} + y\vec{x}$     // Update the weight vector $\vec{w}$
            $m \leftarrow m + 1$                    // Counter the number of misclassification
        end if
    end for
    if $m = 0$ then                                 // If the most recent $\vec{w}$ gave 0 misclassifications
        break                                       // Break out of the while-loop
    end if
end while                                           // Otherwise, keep looping!
```

Perceptron

Perceptron Algorithm:

Activation Function / Transfer function

An activation function in neural networks is a function that introduces non-linearity into the model.

- Step function gets triggered above a certain value of the neuron output;

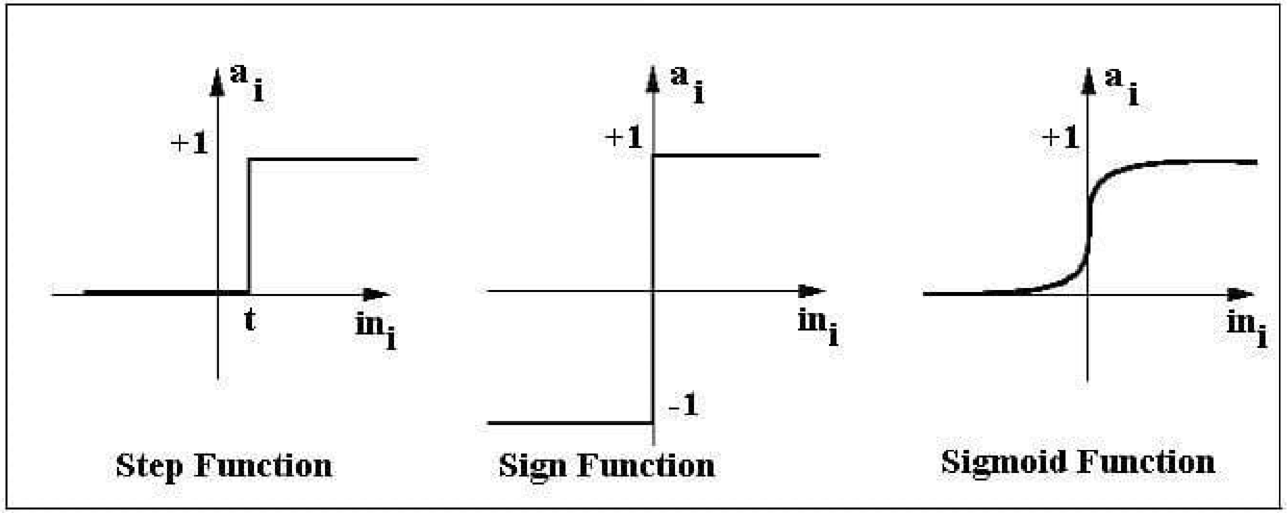
else it outputs zero.

- Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not.
- Sigmoid is the S-curve and outputs a value between 0 and 1.

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

$$f(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases}$$

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



Activation Function / Transfer function

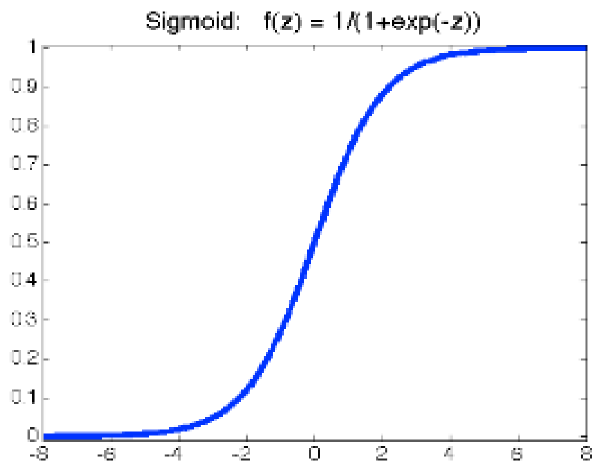
An activation function in neural networks is a function that introduces non-linearity into the model.

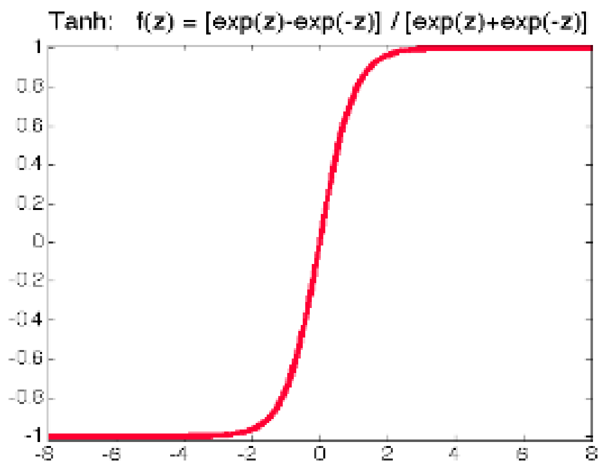
Most commonly used activation functions:

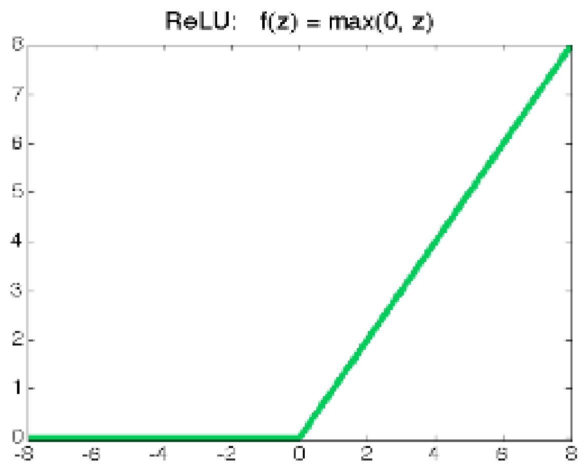
Sigmoid: $\sigma(z) = 1/\ 1+\exp(-z)$

Tanh: $\tanh(z) = \exp(z)-\exp(-z)/\ \exp(z)+\exp(-z)$

ReLU (Rectified Linear Unit): $\mathrm{ReLU}(z) = \max(0, z)$







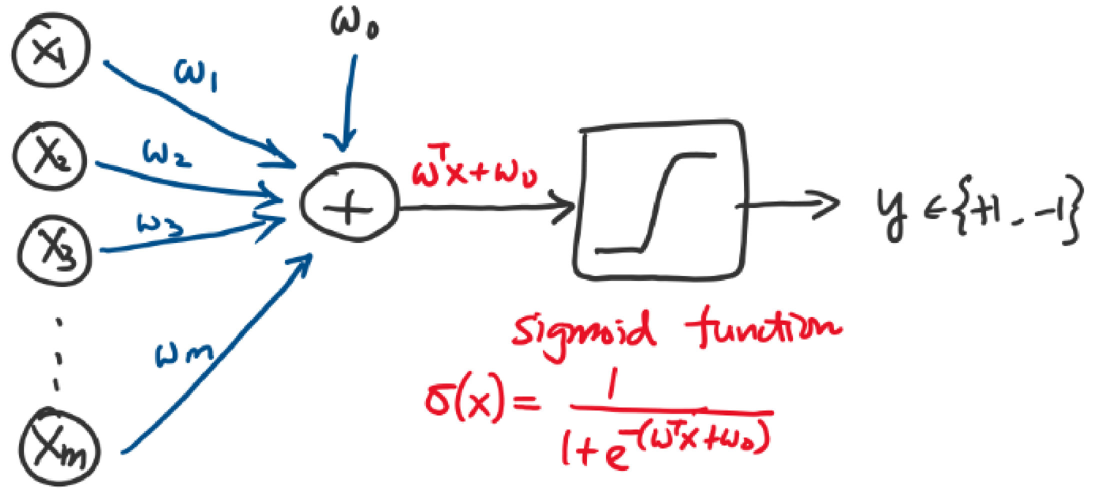
Neural Network (NN) / Artificial Neural Network (ANN)

- Neural networks are sometimes called artificial neural networks (ANNs)
- A neural network is a machine learning model, that makes decisions in a manner similar to the human brain.
- Neural Networks are *computational models* that mimic the function and structure of biological neurons *(or)* we can say it is inspired by the structure and *functions* of the *human brain.)*

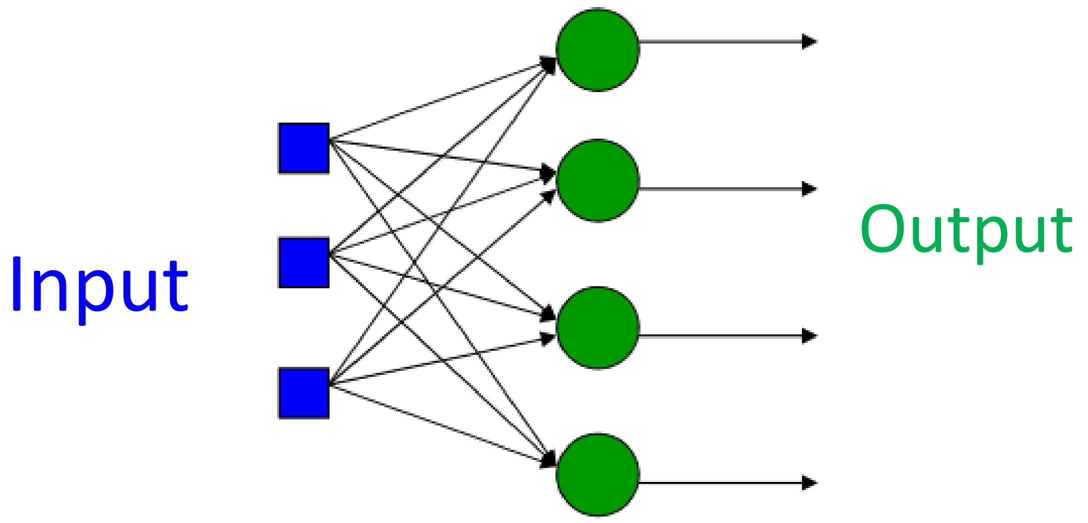
Why are neural networks important?

- Neural networks can learn and model the relationships between input and output data that are non-linear and complex.

Single Layer Perceptron or Single layer Feed-forward Network



- Input neurons $\boldsymbol{x}$
- Weights $\boldsymbol{w}$
- Predicted label $= \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$.

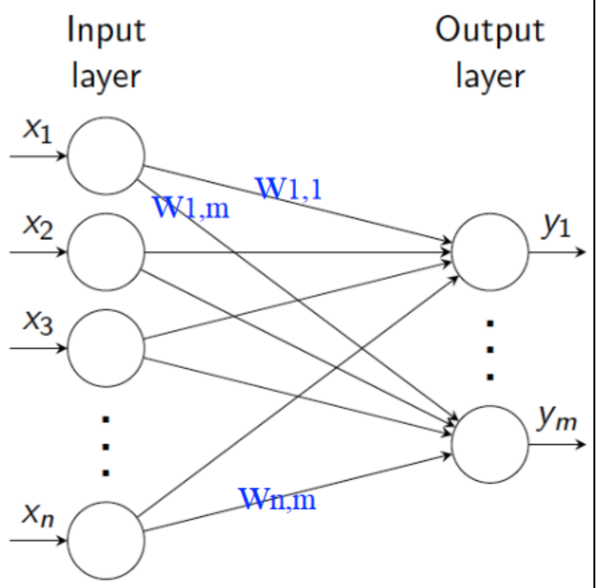


Single Layer Perceptron or Single layer Feed-forward Network

- Single Layer Perceptron has just two layers of input and output. It does not contain Hidden Layers

- The calculation of the single-layer, is done by multiplying the sum of the input vectors of each value by the corresponding elements of the weight vector. The value displayed in the output **is** the input of the activation function.

- Input: column vector x (size n×1)
- Output: column vector y (size m×1)
- Layer parameters:
  weight matrix W (size n×m)
  bias vector b (m×1)
- Units activation: $a = Wx + b$

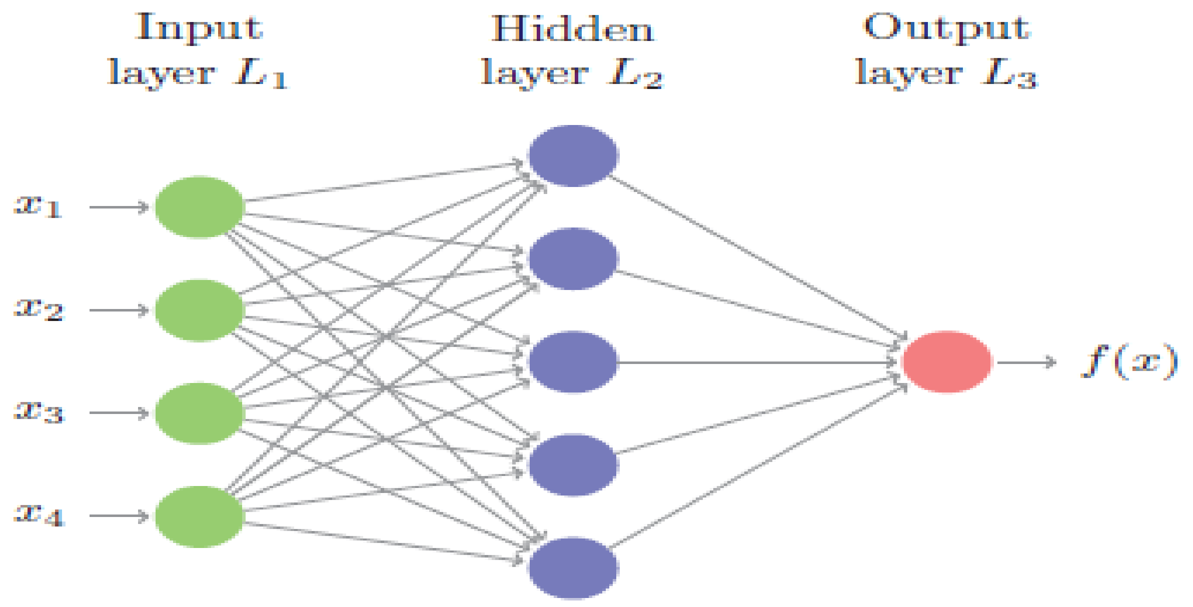


Multi-Layer Perceptron (MLP) or Multilayer Feed forward Network

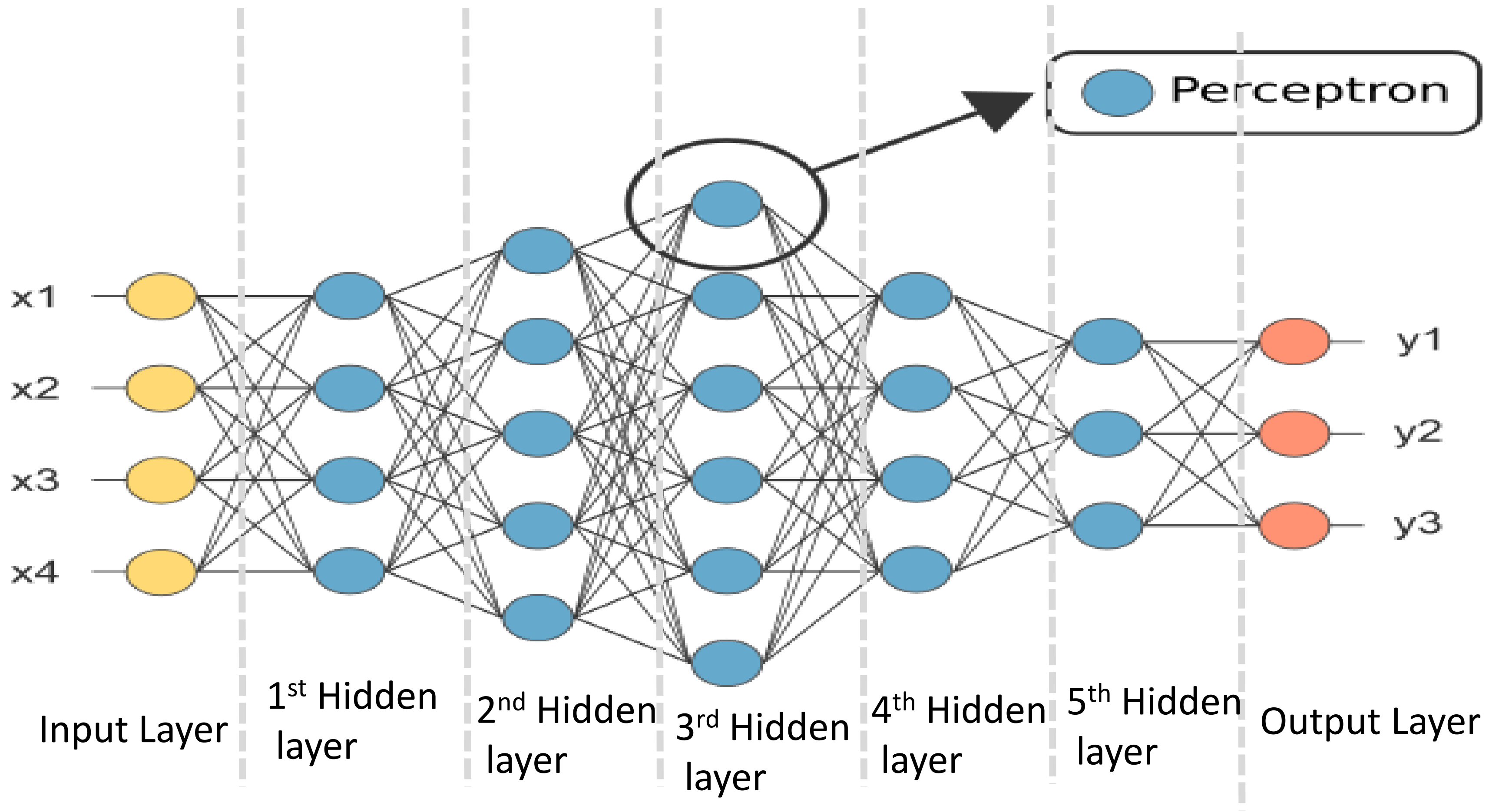
- Feed-forward neural networks are the most popular and most widely used models in many practical applications. They are known by many different names, such as ‘multilayer perceptrons’ (MLP).
- It consists of a number of simple neuron-like processing units, organized in layers and every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, as each connection may have a different weight.
- *Feed-forward Neural networks* consists of an input layer, one or more hidden layers, and an output layer
- Layers between input and output are called hidden.

Multi-Layer Perceptron (MLP) or Multilayer Feed forward Network

- The MLP is one of the most used supervised model: it consists of multiple layers of computational units, usually interconnected in a feed-forward way.
- Each neuron in one layer has direct connections to all the neurons of the subsequent layer



Multi layer feed-forward NN (FFNN)



Multi layer feed-forward NN (FFNN)

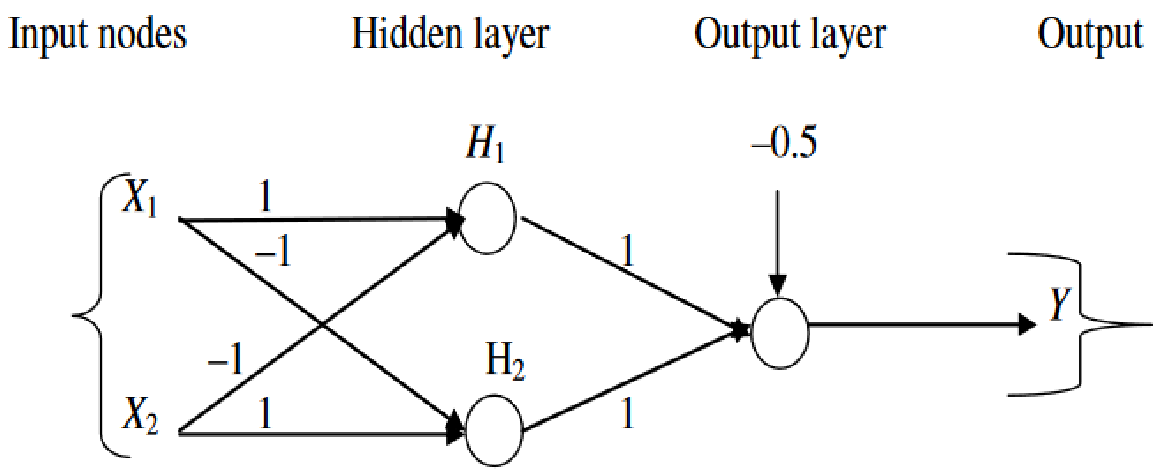
Properties of FFNN:

- No connections within a layer
- No direct connections between input and output layers
- Fully connected between layers
- Often more than 3 layers
- Number of output units need not equal number of input units
- Number of hidden units per layer can be more or less than input or output units

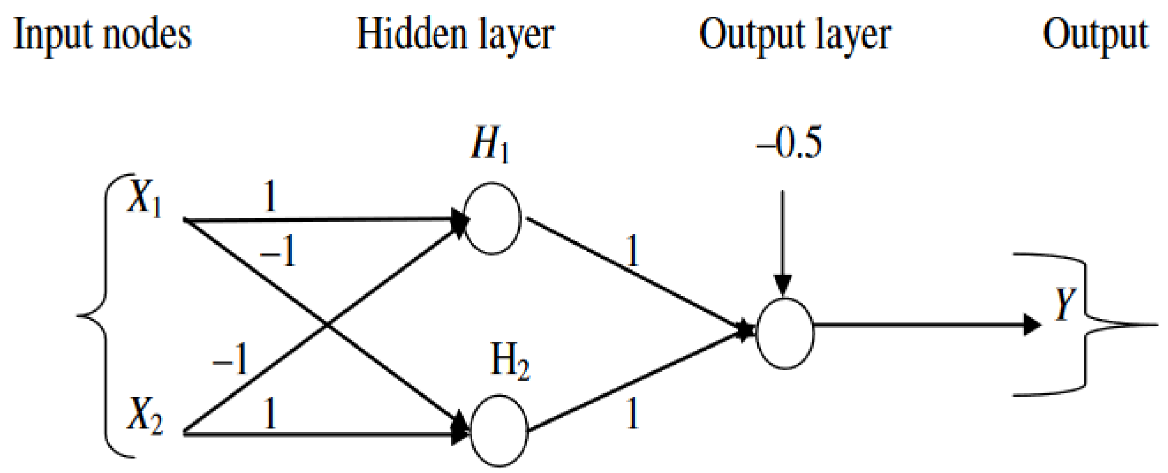
FFNN for XOR

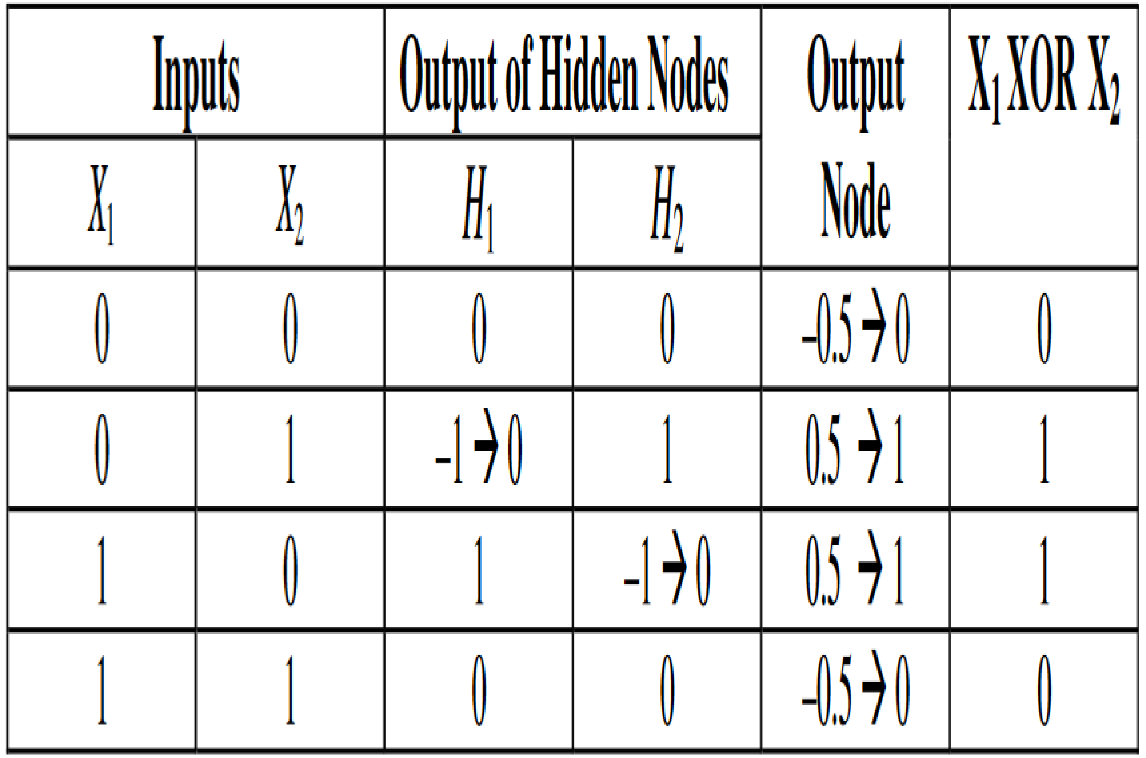
The ANN for XOR has two hidden nodes that realizes this non-linear separation and uses the sign (step) activation function.

- Arrows from input nodes to two hidden nodes indicate the directions of the weight vectors (1,-1) and (-1,1).
- The output node is used to combine the outputs of the two hidden nodes



FFNN for XOR





Inputs		Output of Hidden Nodes		Output Node	X_1 XOR X_2
X_1	X_2	H_1	H_2		
0	0	0	0	$-0.5 \rightarrow 0$	0
0	1	$-1 \rightarrow 0$	1	$0.5 \rightarrow 1$	1
1	0	1	$-1 \rightarrow 0$	$0.5 \rightarrow 1$	1
1	1	0	0	$-0.5 \rightarrow 0$	0

Since we are representing two states by 0 (false) and 1 (true), we will map negative outputs (–1, –0.5) of hidden and output layers to 0 and positive output (0.5) to 1

SLP vs MLP

Single Layer Perceptron (SLP):

- Single layer Perceptrons can learn only linearly separable patterns.
- A single layer perceptron (SLP) is **a feed-forward network based on a threshold transfer function.**
- It consists of a **single layer**, which is the input **layer**, with multiple neurons with their own weights; there are no hidden **layers**

Multi layer Perceptron (MLP):

- Multilayer Perceptron or feedforward neural network with two or more layers have the greater processing power and can process non-linear patterns as well.
- A Multi Layer Perceptron (MLP) one input layer, one output layer and **contains one or more hidden layers**

Backpropagation Algorithm

BP has two phases:

1) Forward pass phase: computes ‘functional signal’, feed forward propagation of input pattern signals through network.

2) Backward pass phase: computes ‘error signal’, propagates the error backwards through network starting at output units (where the error is the difference between actual and desired output values)

Back Propagation:

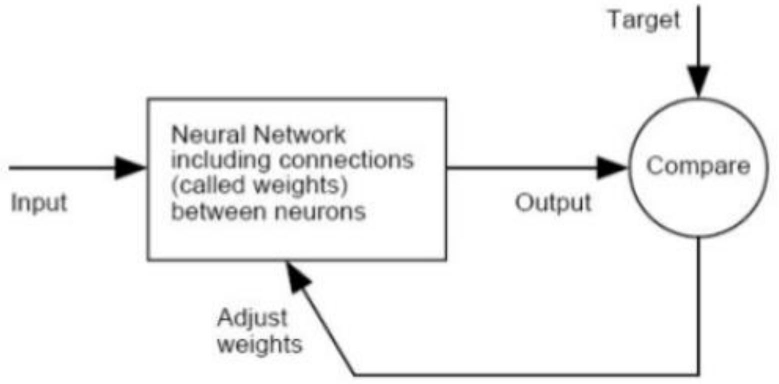
– the output values are compared with the target to compute the value of some predefined error function

– the error is then fed back through the network

– using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function

Backpropagation Algorithm

After repeating this process for a sufficiently large number of training cycles, the network will usually converge.


Target
Input
Neural Network including connections (called weights) between neurons
Output
Compare
Adjust weights

The best number of hidden units depend on:
– number of inputs and outputs
– number of training case
– the amount of noise in the targets
– the complexity of the function to be learned
– the activation function

- Too few hidden units => high training and generalization error, due to under fitting and high statistical bias.
- Too many hidden units => low training error but high generalization error, due to overfitting and high variance

Neural network

- By convention, number of layers is hidden + output (i.e. does not include input). So 3-layer model has 2 hidden layer.

- Single-layer:

$$h = \boldsymbol{w}^T \boldsymbol{x}$$

- Hidden-layer:

$$\boldsymbol{h} = \boldsymbol{W}^T \boldsymbol{x}$$

- Two Hidden Layers:

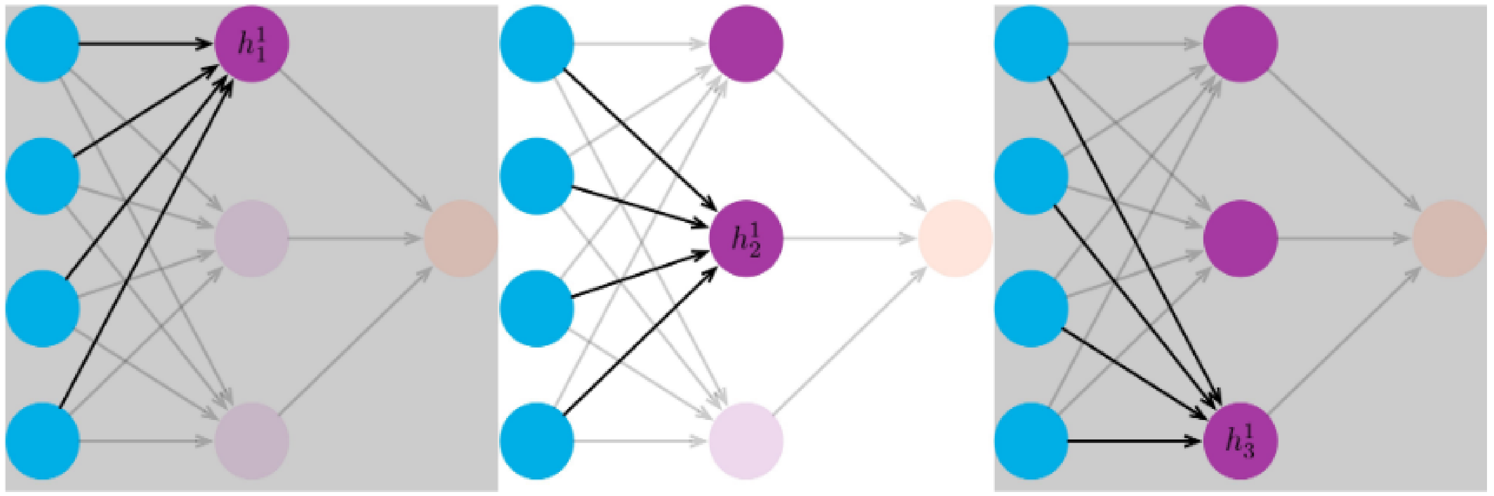
$$\boldsymbol{h} = \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$

- Three Hidden Layers:

$$\boldsymbol{h} = \boldsymbol{W}_3^T \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$

- A LOT of Hidden Layers:

$$\boldsymbol{h} = \boldsymbol{W}_N^T \dots \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$

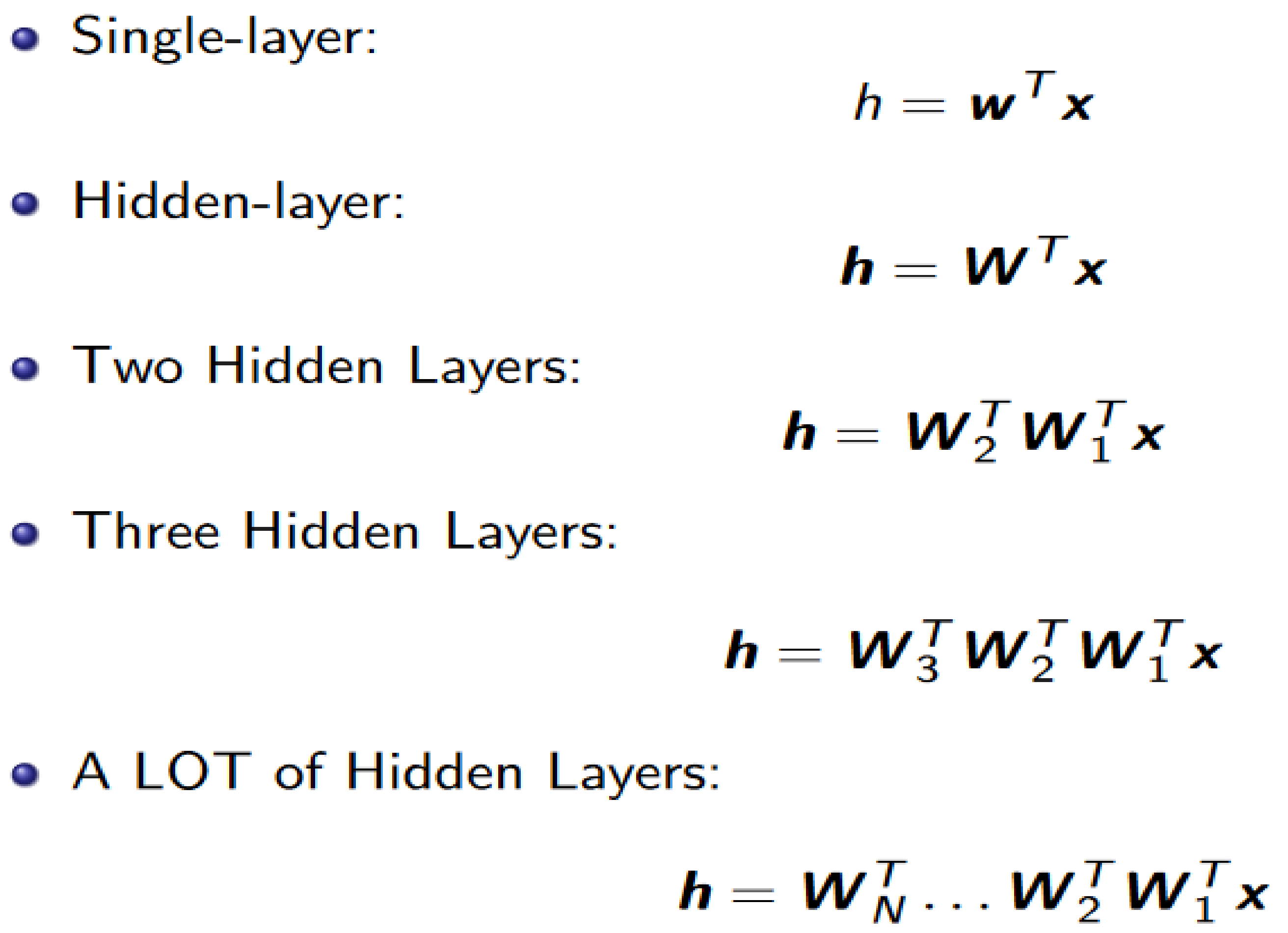


- Each hidden neuron is an **output** of a perceptron
- So you will have

$$\begin{bmatrix} h_1^1 \\ h_2^1 \\ \dots \\ h_n^1 \end{bmatrix} = \begin{bmatrix} w_{11}^1 & w_{12}^1 & \dots & w_{1n}^1 \\ w_{21}^1 & w_{22}^1 & \dots & w_{2n}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}^1 & w_{m2}^1 & \dots & w_{mn}^1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Neural network

- By convention, number of layers is hidden + output (i.e. does not include input). So 3-layer model has 2 hidden layer.



- Single-layer:

$$h = \mathbf{w}^T \mathbf{x}$$

- Hidden-layer:

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

- Two Hidden Layers:

$$\mathbf{h} = \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$

- Three Hidden Layers:

$$\mathbf{h} = \mathbf{W}_3^T \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$

- A LOT of Hidden Layers:

$$\mathbf{h} = \mathbf{W}_N^T \dots \mathbf{W}_2^T \mathbf{W}_1^T \mathbf{x}$$

Genetic Algorithm (GA)

- *Evolutionary algorithms are* a family of optimization *algorithms* based on the principle of Darwinian natural selection.
- A genetic algorithm *is a class of evolutionary algorithm*.
- A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems
- (GA)s are categorized as global search heuristics
- What is Evolutionary Computation?

The study of computational systems that use ideas inspired

from natural evolution, e.g., the principle of survival of the fittest

How evolution works in nature?

- *Evolution: change in the inherited characteristics of biological populations over successive generations.*
  - *Heritable characteristics or heritable traits, e.g., the colour of your eyes are passed from one generation to the next via DNA*
  - *DNA: Deoxyribonucleic acid, a molecule that encodes genetic information*
  - *Change or genetic variation comes from:*
    - *Mutations: changes in the DNA sequence,*
    - *Crossover: reshuffling of genes through sexual reproduction and migration between populations*
- *Evolution is driven by natural selection - survival of the fittest*
- *Genetic variations that enhance survival and reproduction become and remain more common in successive generations of a population.*

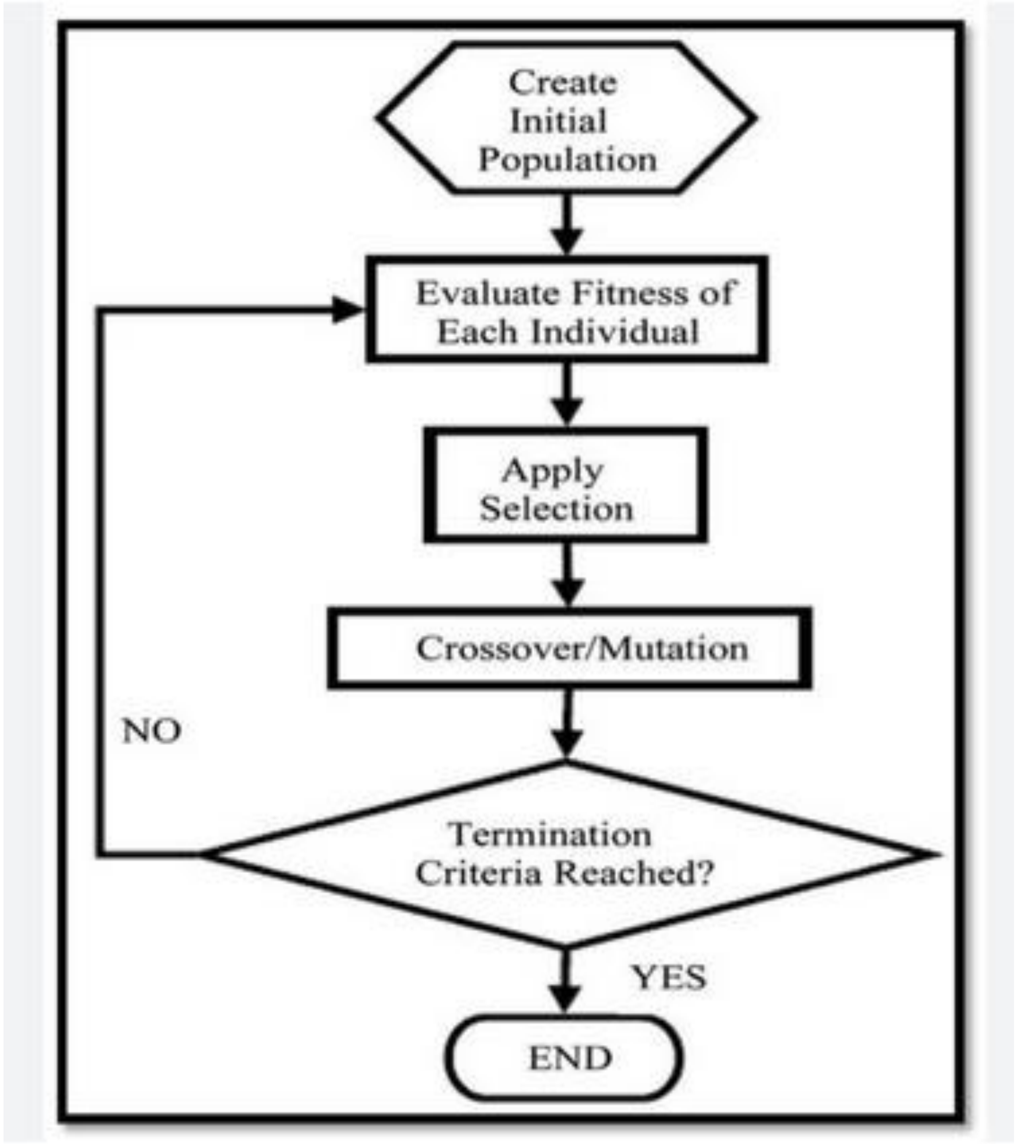
Genetic Algorithm

- Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified to form a new population
- Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.
- This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Genetic Algorithm

```
function sga ()
{
Initialize population;
Calculate fitness function;
    While(fitness value != termination criteria)
        {
            Selection;
            Crossover;
            Mutation;
            Calculate fitness function;
        }
}
```

The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population


Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

Genetic Algorithm

1.[Start] Generate random population of *n* chromosomes (suitable solutions for the problem)
2.[Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population
3.[New population] Create a new population by repeating following steps until the new population is complete

1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
2. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
4. **[Accepting]** Place new offspring in the new population

4.[Replace] Use new generated population for a further run of the algorithm
5.[Test] If the end condition is satisfied, **stop**, and return the best solution in current population
6.[Loop] Go to step **2**

- *Fitness function* is defined as an objective function that quantifies the optimality of a solution (chromosome) to the target problem.
- The actual definition of a fitness function is problem dependent.

Genetic Algorithm: Encoding

Encoding of a Chromosome:

The chromosome should in some way contain information about solution which it represents. The most used way of encoding is a binary string. The chromosome then could look like this:

Chromosome 1	**1101100100110110**
Chromosome 2	**1101111000011110**

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution

There are many ways of encoding:

1. Binary encoding: Representing a gene in terms of bits (0s and 1s).
2. Real value encoding: Representing a gene in terms of values or symbols or string.
3. Permutation (or Order) encoding: Representing a sequence of elements)
4. Tree encoding: Representing in the form of a tree of objects

Genetic Operators: Selection

The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.

The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.

Selection is the process for creating the population for next generation from the current generation

"Selects the best, discards the rest"

Genetic Operators: Selection

- Identify the good solutions in a population
- Make multiple copies of the good solutions

Now how to identify the good solutions?

A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions

A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

Selection method is procedure used to rate the fitness of individual solutions and select the fitter ones.

Genetic Algorithms: Selection of the Fittest

- Selection methods:
  - ***Fitness Proportionate (Roulette Wheel) Selection***:

    Evaluate the fitness f of all individuals h_i, $i = [1..n]$, select k of them with the likelihood $Pr(h_i)$:

    $$Pr(h_i) = f(h_i) \, / \sum_j f(h_j).$$

  - ***Tournament Selection***:

    Choose k individuals to compete, evaluate their fitness, rank them, and assign probability $Pr(h_i)$ to each h_i :

    h_i with rank 1 $\rightarrow Pr(h_i) = p$,

    h_i with rank 2 $\rightarrow Pr(h_i) = p \cdot (1\text{-}p)$,

    h_i with rank 3 $\rightarrow Pr(h_i) = p \cdot (1\text{-}p)^2$, …

  - ***Rank Selection***:

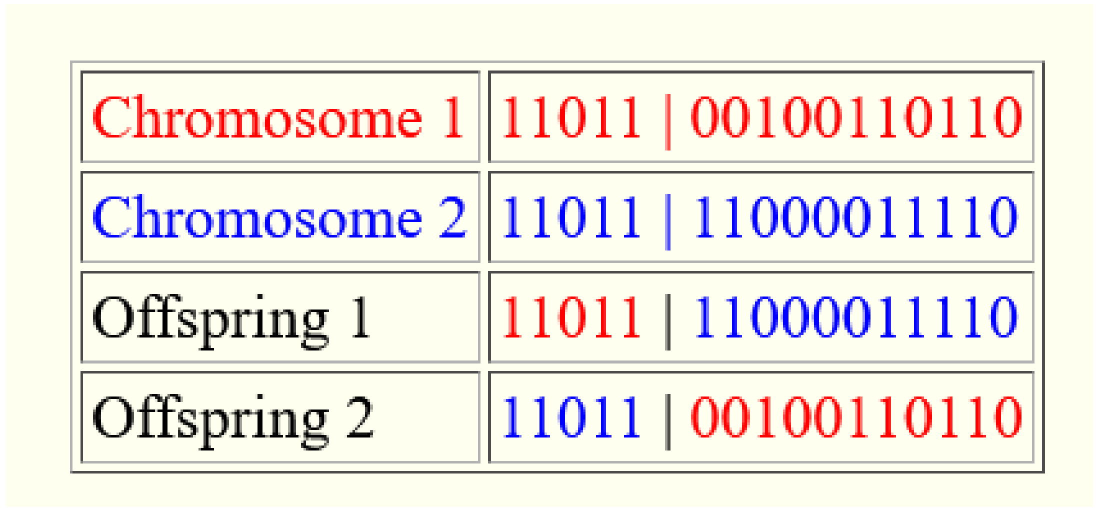
    Evaluate the fitness f of all individuals h_i, $i = [1..n]$, rank them, selects k of them having the largest fitness $f(h_i)$.

Genetic Operators: Crossover

Crossover:

After we have decided what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring.

The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent.



Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Genetic Operators: Crossover

Crossover:

Randomly select two parents with probability $p_c \in [0, 1]$ for crossover

1-point crossover: select a single crossover point on two strings, swap the data beyond that point in both strings.

n-point crossover:

- Select multiple crossover points on two strings,
- Split strings into parts using those points
- Alternating between the two parents and then glue parts

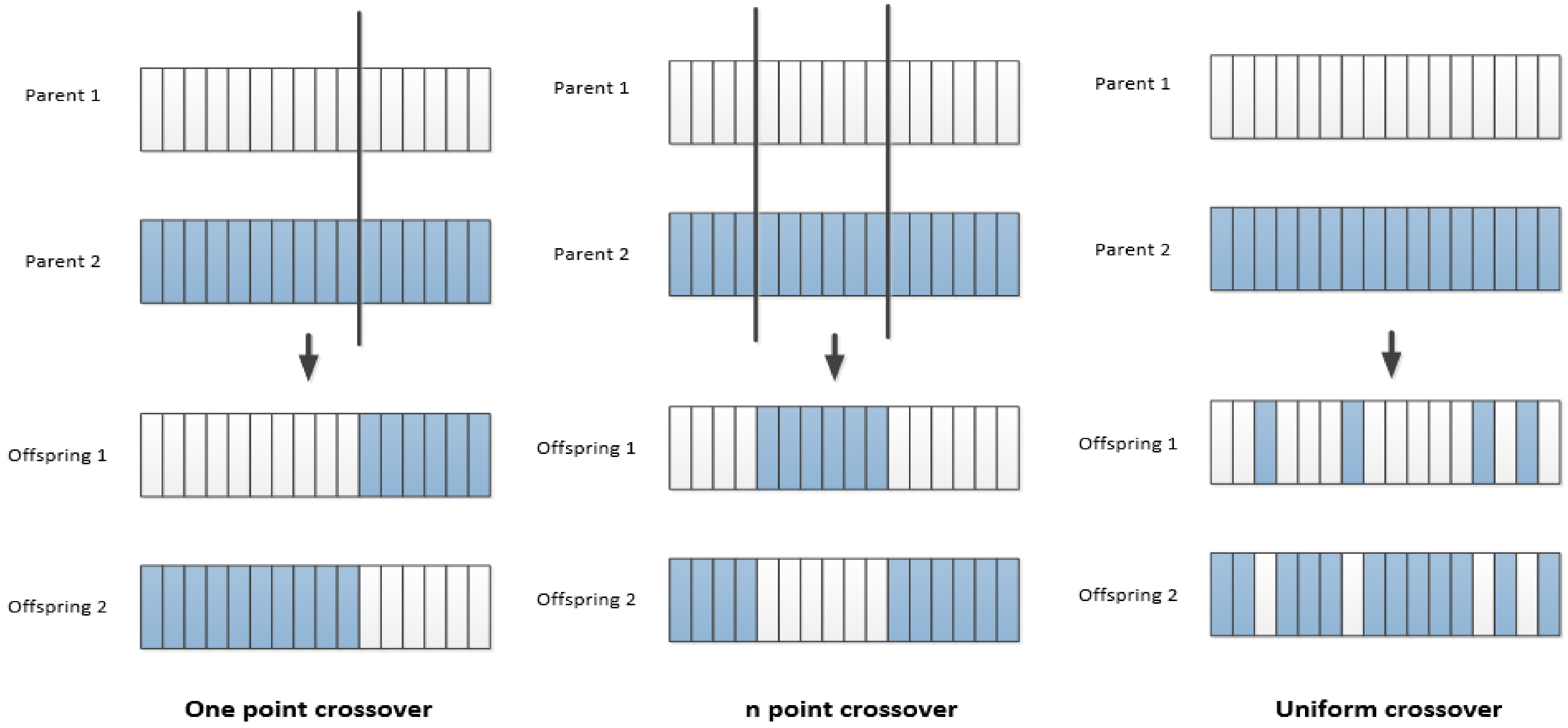
Uniform crossover: For each $i \in \{1, \cdots, L\}$: toss a coin

If 'head': copy bit i from parent 1 to offspring 1, parent 2 to offspring 2

If 'tail': copy bit i from parent 1 to offspring 2, parent 2 to offspring 1

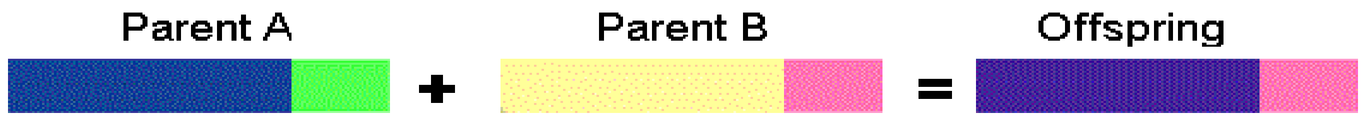
Genetic Operators: Crossover

Crossover:



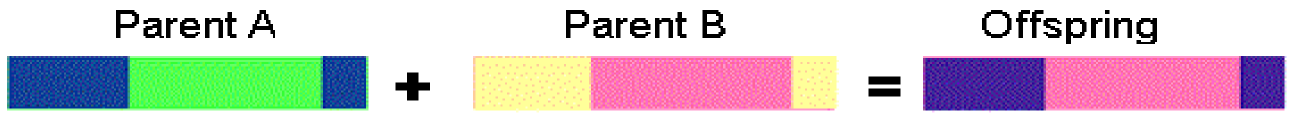
Genetic Operators : Crossover

Single point crossover - one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent



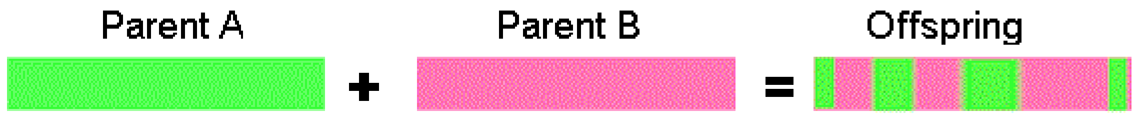
11001011+11011**111** = **11001111**

Two point crossover - two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent



110010**11** + 11**0111**11 = **11011111**

Uniform crossover - bits are randomly copied from the first or from the second parent



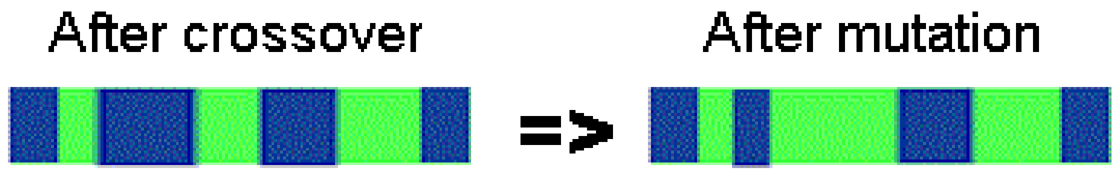
1**1**00**1**0**11** + **1**1**011**1**0**1 = 11011111

Genetic Operators : Mutation

Mutation:

Randomly change one or more digits in the string representing an individual. For example, the individual 1-2-3 may be changed to 1-3-3 or 3-2-3, giving two new offspring. How often to do mutation, how many digits to change, and how big a change to make are adjustable parameters.

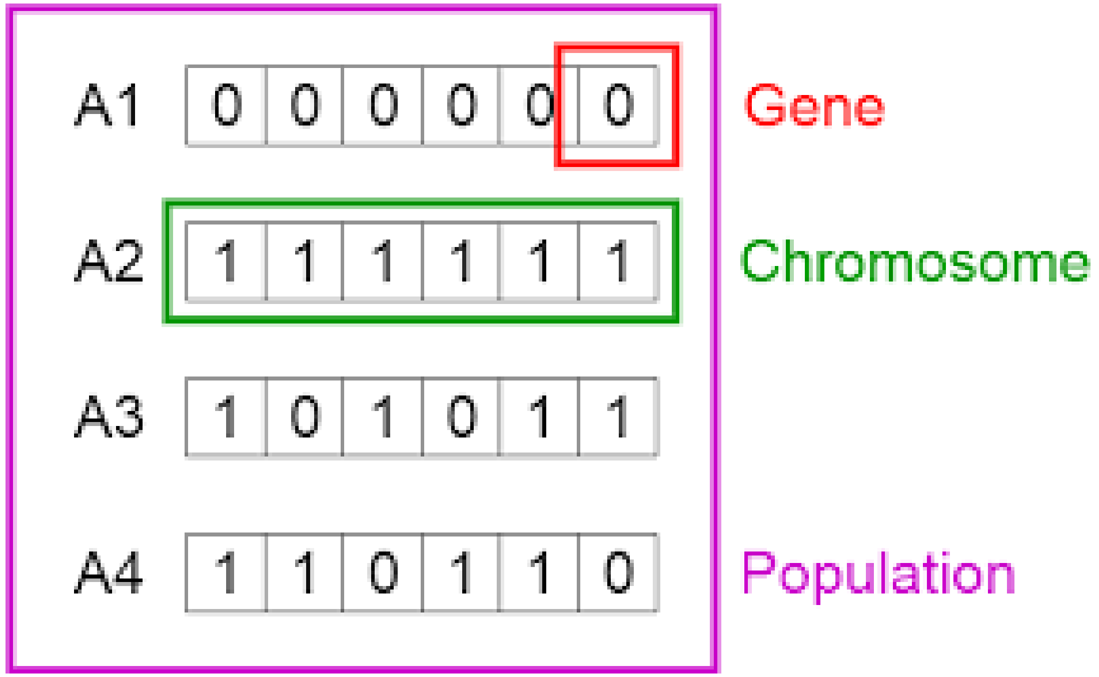
Bit inversion - selected bits are inverted



1**1**001001 => 1**0**001001

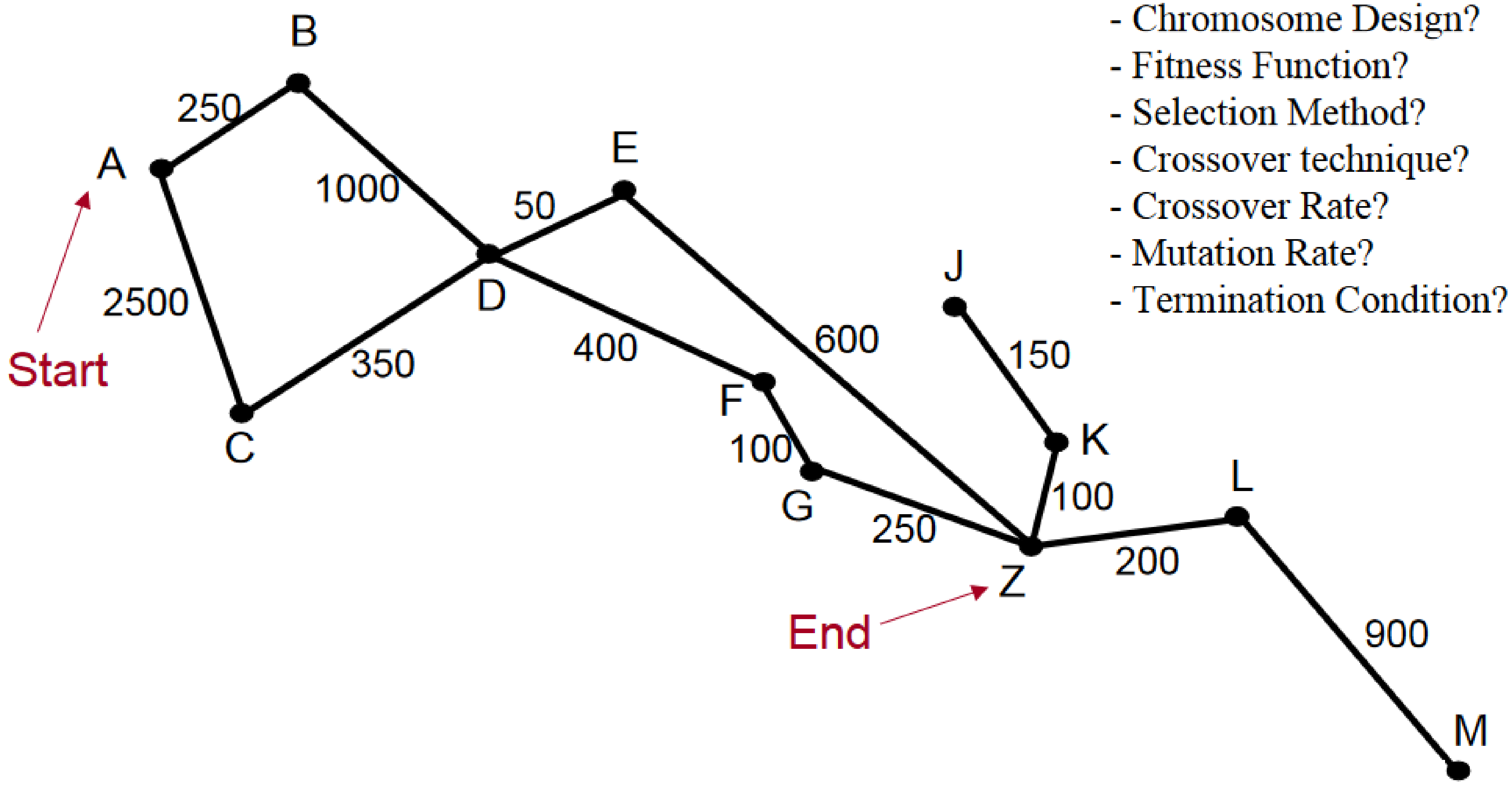
Genetic Algorithms:

- Two basic methods of reproduction, called mutation and crossover



Genetic Algorithms: Example

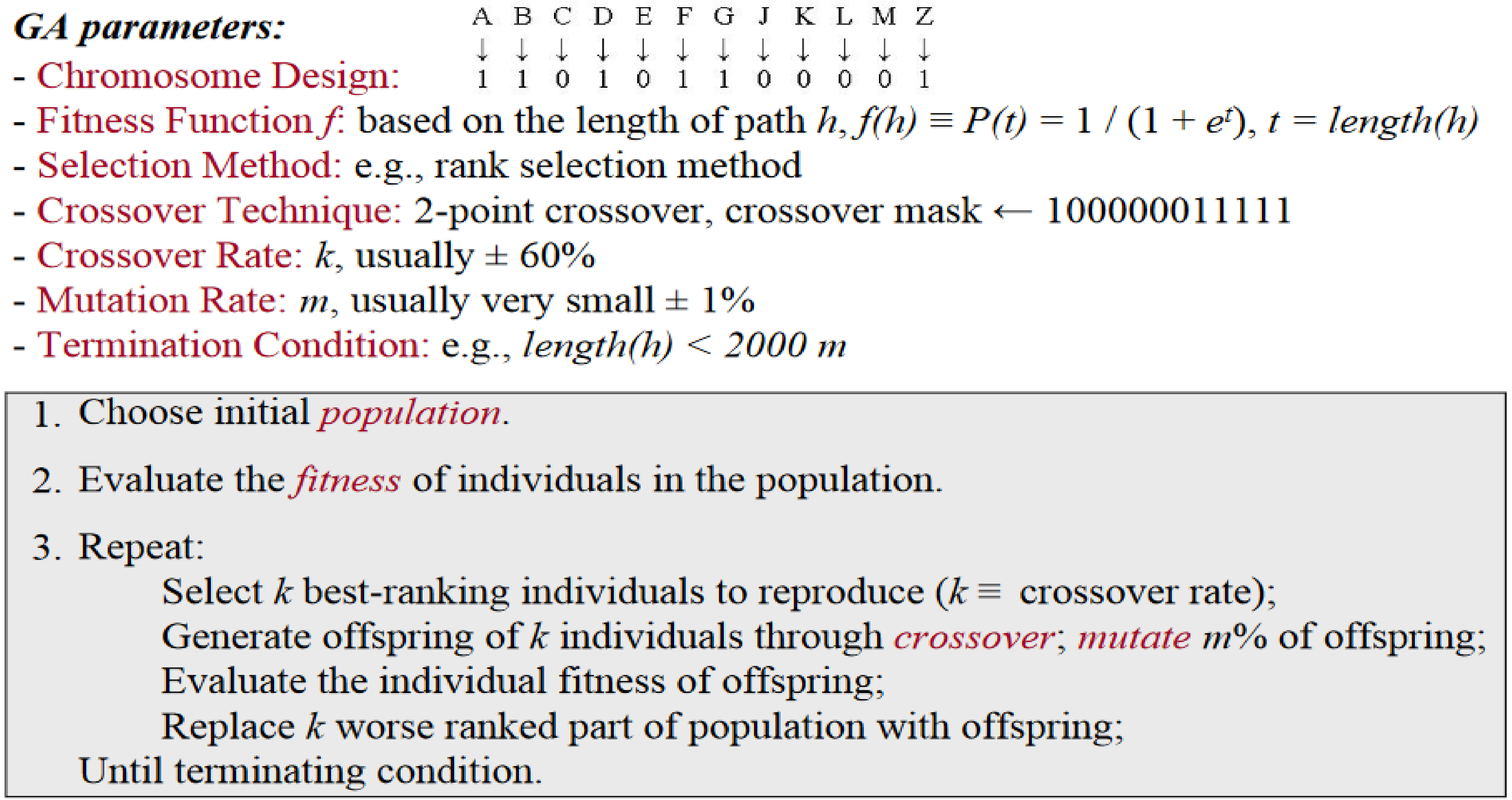
Traveling Salesman Problem A → Z:



GA parameters:

- Chromosome Design?
- Fitness Function?
- Selection Method?
- Crossover technique?
- Crossover Rate?
- Mutation Rate?
- Termination Condition?

Genetic Algorithms: Example



GA parameters:

- Chromosome Design:

A	B	C	D	E	F	G	J	K	L	M	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	1	0	1	0	1	1	0	0	0	0	1

- Fitness Function f: based on the length of path h, $f(h) \equiv P(t) = 1 / (1 + e^t)$, $t = length(h)$
- Selection Method: e.g., rank selection method
- Crossover Technique: 2-point crossover, crossover mask $\leftarrow$ 100000011111
- Crossover Rate: k, usually $\pm$ 60%
- Mutation Rate: m, usually very small $\pm$ 1%
- Termination Condition: e.g., $length(h) < 2000\ m$

```
1. Choose initial population.
2. Evaluate the fitness of individuals in the population.
3. Repeat:
           Select $k$ best-ranking individuals to reproduce ($k \equiv$ crossover rate);
           Generate offspring of $k$ individuals through crossover; mutate $m$% of offspring;
           Evaluate the individual fitness of offspring;
           Replace $k$ worse ranked part of population with offspring;
   Until terminating condition.
```

Questions

Q1. Name and describe the main features of Genetic Algorithms (GA).

Encoding:

Fitness function:

Selection:

Crossover:

Mutation:

Questions :GA

Q1. Name and describe the main features of Genetic Algorithms (GA).

Answer: Genetic Algorithms (GA) use principles of natural evolution. There are five important features of GA

Fitness Function represents the main requirements of the desired solution of a problem (i.e. cheapest price, shortest route, most compact arrangement, etc). This function calculates and returns the fitness of an individual solution.

Encoding possible solutions of a problem are considered as individuals in a population. If the solutions can be divided into a series of small steps (building blocks), then these steps are represented by genes and a series of genes (a chromosome) will encode the whole solution. This way different solutions of a problem are represented in GA as chromosomes of individuals.

Selection operator defines the way individuals in the current population are selected for reproduction. There are many strategies for that (e.g. roulette–wheel, ranked, tournament selection, etc), but usually the individuals which are more fit are selected.

Crossover operator defines how chromosomes of parents are mixed in order to obtain genetic codes of their offspring (e.g. one–point, two–point, uniform crossover, etc). This operator implements the inheritance property (offspring inherit genes of their parents)

Mutation operator creates random changes in genetic codes of the off-spring. This operator is needed to bring some random diversity into the genetic code. In some cases GA cannot find the optimal solution without mutation operator (local maximum problem).

Planning

- **Planning** is the task of determining a sequence of actions that will achieve a goal.
- Plans (aka solutions) are sequences of moves that transform the initial state into the goal state

Domain independent heuristics and strategies must be based on a domain independent representation

- Initial state, set of actions and goal state
- STRIPS, ADL, PDDL are languages based on propositional or first-order logic

Classical planning environment

- fully observable, deterministic, finite, static and discrete

Planning: Problem Representation

We usually specify an initial state and a goal state, and then try to find (a short!) plan that transform the initial state in the goal state

Problem Representation:

State

- What is true about the (hypothesized) world?

Goal

- What must be true in the final state of the world?

Actions

- What can be done to change the world?
- Preconditions and effects

Planning: Example **Making Tea**

Suppose you have a robot that can serve tea

Think about what the robot must do to make tea, e.g.:

- it must put water in the kettle
- heat the kettle
- get a cup
- pour hot water into the cup (after the water is hot enough)
- get a tea bag
- leave the tea bag in the water for enough time
- remove the tea bag
- add milk
- add sugar
- mix the tea
- serve the tea

There are many, many actions to consider! most actions consist of many smaller sub-actions and in some situations very long sub-plans might be triggered

Example Planning Problem: Getting Dressed

Suppose you want to put your socks and shoes on

suppose you have:

- a left sock
- a right sock
- a left shoe
- a right shoe

one possible plan for putting these on is:

- put left sock on left foot
- put right sock on right foot
- put left shoe on left foot
- put right shoe on right foot

another possible plan is:

- put left sock on left foot
- put left shoe on left foot
- put right sock on right foot
- put right shoe on right foot

For this planning problem, what matters is that the left sock be put on before the left shoe, and the right sock be put on before the right shoe

Example Planning Problem: Blocks World

Suppose you have three cubical blocks, A, B, and C, and a robot arm that can pick up and move one block at a time

Suppose they are initially arranged on a table like this:

```
     C
  A  B            initial state
--------------
```

A and B are on the table, and C is on top of A. Suppose that we want to re-arrange them into this state:

```
  A
  B
  C             Goal state
--------------
```

we can only pick up a block if there is nothing on top if it

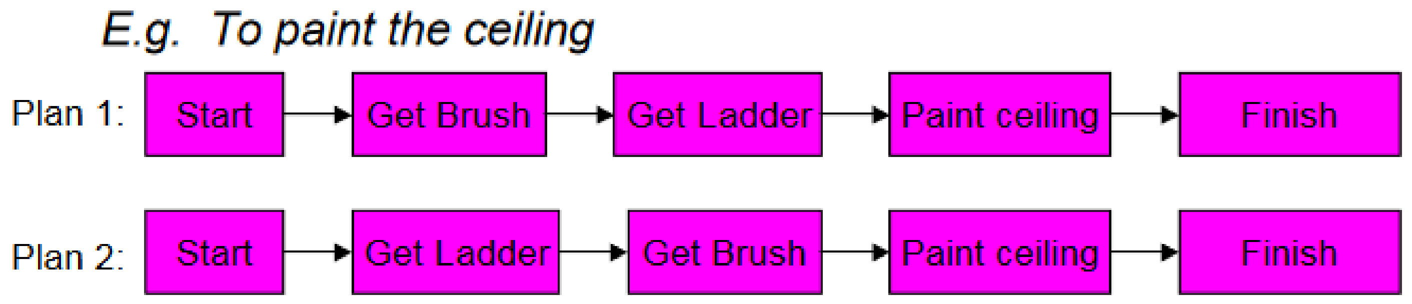
- move C to the table
- move B on top of C
- move A on top of B

State-Space Planners

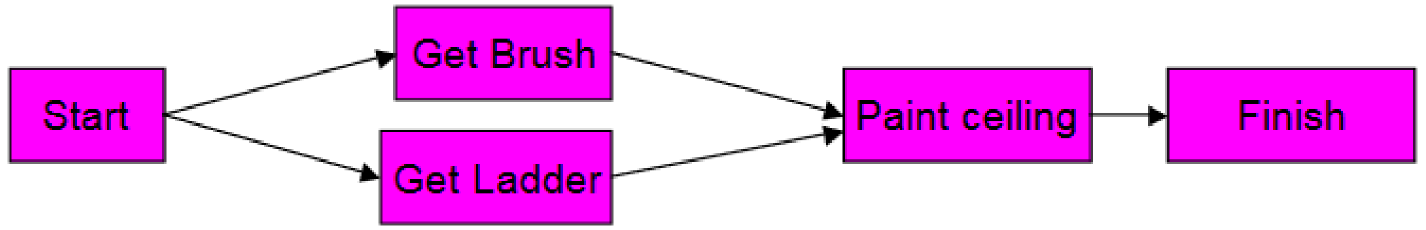
- Progression planners (forward planner) starts at the *initial* state, and applies actions in an effort to find a path to the goal state
- Regression planners (Backward planner) starts at the *goal* state, and works backwards from that to try to find a path to the goal state
- In both cases, the planners work with sets of states instead of using individual states, like in straightforward search

Total vs. Partial Order plan

- Total order: Plan is always a strict sequence of actions



- Partial order: Plan steps may be unordered



Questions: GA

Q1: Consider a genetic algorithm in which individuals are represented using a 5-bit string of the form b1b2b3b4b5. An example of an individual is 001001 for which b1=0, b2=0, b3=1, b4=0, b5=1.

The **fitness function** is defined over these individuals as follows:
f(b1b2b3b4b5) = b1 + b2 + b3 + b4 + b5 + AND(b1,b2,b3,b4,b5),

Calculate the probability of selecting of individuals from the population :

00101, 11101, 00000, 10010, 11111

Q2: Crossover Suppose that a single crossover point will be used for crossover. This point has been chosen as the point between the 2nd and the 3rd bits (i.e. between b2 and b3). Show the two offspring that will result from crossing over the following two individuals:

First parent: 00101, Second parent: 10111

Questions: GA

Q1: Solution

Individual	fitness Value (f_i)	Probability of being selected ($f_i/\Sigma f_i$)
00101	0+0+1+0+1+0 = 2	2/14 = 0.14
11101	1+1+1+0+1+0 = 4	4/14 = 0.29
00000	0+0+0+0+0+0 = 0	0/14 = 0
10010	1+0+0+1+0+0 = 2	2/14 = 0.14
11111	1+1+1+1+1+1 = 6	6/14 = 0.43

Q2: Crossover Suppose that a single crossover point will be used for crossover. This point has been chosen as the point between the 2nd and the 3rd bits (i.e. between b2 and b3). Show the two offspring that will result from crossing over the following two individuals:

First parent: 00101, Second parent: 10111

Solution : Offspring 1: 00111, Offspring 2: 10101

GA: Crossover

Q1: Find out the two offspring after one point crossover of x_1 and x_2

x_1 = 8 7 1 2 6 6 1 and x_2 = 6 5 4 1 3 5 3, Crossover point is in between 4th and 5th position

Q2: Find out the two offspring after one point crossover of x1 and x2

x1= 6 5 4 1 3 5 3 2 and x2 = 2 3 9 2 1 2 8 5

FOL

Write the following in FOL form

1. Every investor bought stocks.
2. If the SENSEX crashes, then all stocks fall.
3. Alex is an investor.
4. If the SENSEX crashes then Alex is not happy.

Find out the flowing from above

constants symbols ?

relation symbols ?

variable symbols?

FOL

Write the following in FOL form

1. Every investor bought stocks.

forall x [investor(x) -> bought(x,stocks)]

2. If the SENSEX crashes, then all stocks fall.

crashes(SENSEX) -> fall(stocks)

3. Alex is an investor.

investor(Alex)

4. If the SENSEX crashes then Alex is not happy.

crashes(SENSEX) -> ~happy(Alex)

Find out the flowing from above

constants symbols : SENSEX, stocks, Alex.

relation symbols : investor, bought, crashes, fall, happy

variable symbols: x

FOL

Translate the following sentences in first-order logic.

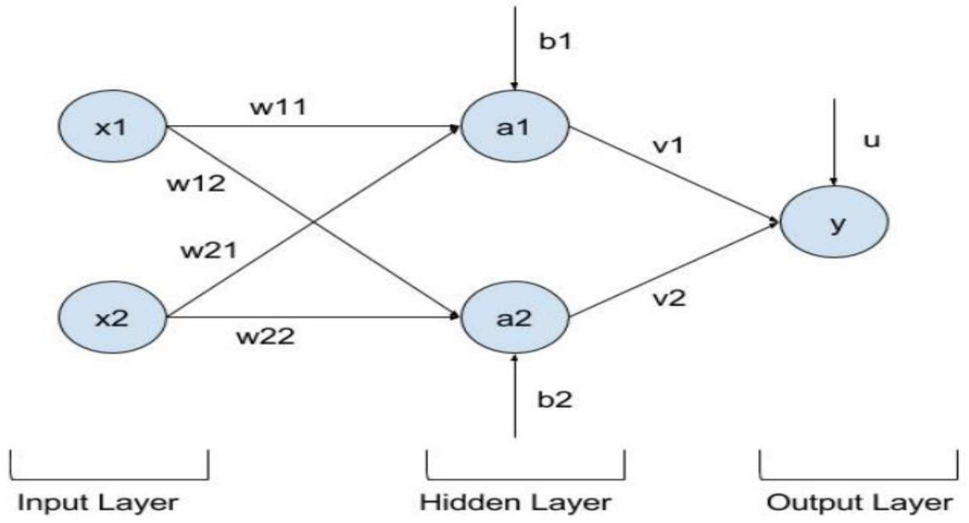
i. Star Trek, Star Wars and The Matrix are science fiction movies.
ii. Every AI student loves Star Trek or Star Wars.
iii. Some AI students do not love Star Trek.
iv. All AI students who love Star Trek also love The Matrix.
v. Every AI student loves some science fiction movie.
vi. No science fiction movie is loved by all AI students.
vii. There is an AI student who loves all science fiction movies.

FOL

Translate the following sentences in first-order logic.

- i. Star Trek, Star Wars and The Matrix are science fiction movies.
- Answer: SciFi(Star Trek) ∧ SciFi(Start Wars) ∧ SciFi(Matrix)
- ii. Every AI student loves Star Trek or Star Wars.
- Answer: ∀x AI Student(x) → Loves(x, Star Trek) ∨ Loves(x, Start Wars)
- iii. Some AI students do not love Star Trek.
- Answer: ∃x AI Student(x) ∧ ¬Loves(x, Star Trek)
- iv. All AI students who love Star Trek also love The Matrix.
- Answer: ∀x AI Student(x) ∧ Loves(x, Star Trek) → Loves(x, Matrix)
- v. Every AI student loves some science fiction movie.
- Answer: ∀x AI Student(x) → (∃y SciFi(y) ∧ Loves(x, y)
- vi. No science fiction movie is loved by all AI students.
- Answer: ¬(∃y SciFi(y) ∧ (∀x AI Student(x) → Loves(x, y)))
- vii. There is an AI student who loves all science fiction movies.
- Answer: ∃x AI Student(x) ∧ (∀ySciFi(y) → Loves(x, y))

ANN: Questions



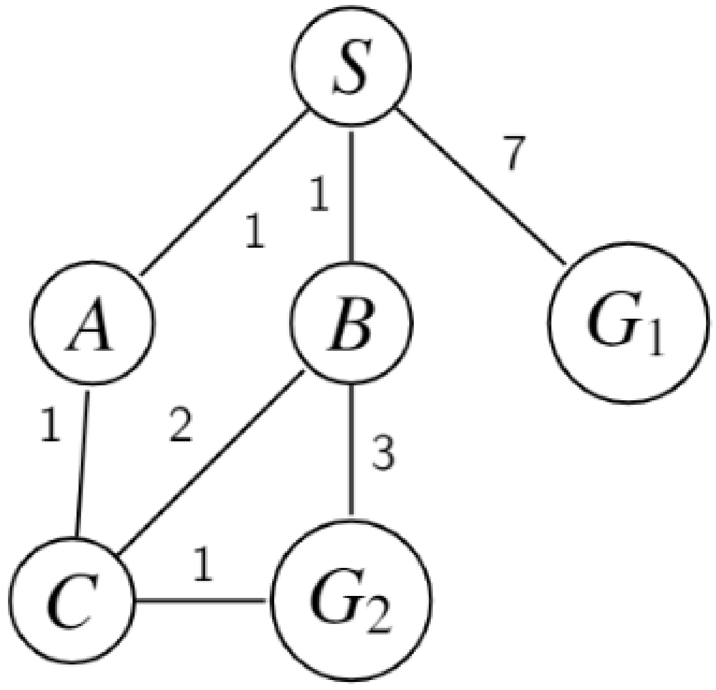
$W_{11}=1$
$W_{12}=0.5$
$W_{21}=0.5$
$W_{22}=1$
$V_1=1$
$V_2=1$

x1	x2	a1	a2	y
1	2			

Calculate output y

Searching

Which goal is reached and what is the total cost of the solution found for the following state-space graph when using Breadth-First Search and Uniform-Cost Search (S is the start state, G1 and G2 are the goal states, arcs are bidirectional, no repeated state checking, break any ties alphabetically)?



Graph Search

- Given the graph above, show the order in which the states are visited by the search algorithms listed below. ***Please do graph search with duplication check, not tree search.***

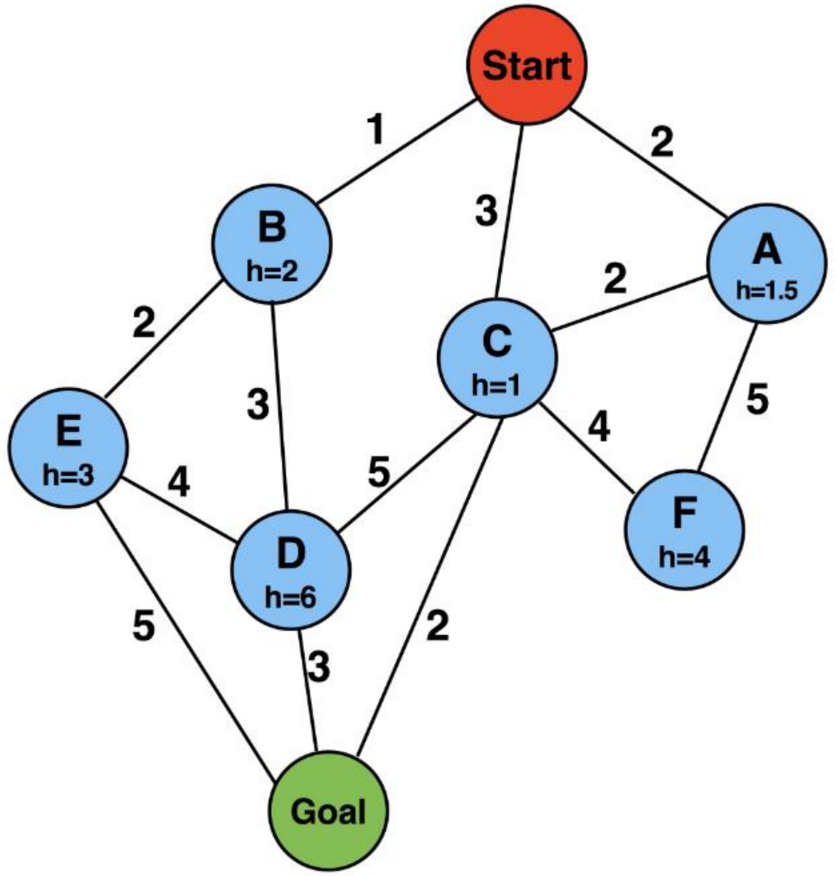
i. ***DFS***

ii. ***BFS***

iii. ***UCS***

iv. ***Greedy BFS***

v. ***A* Search***

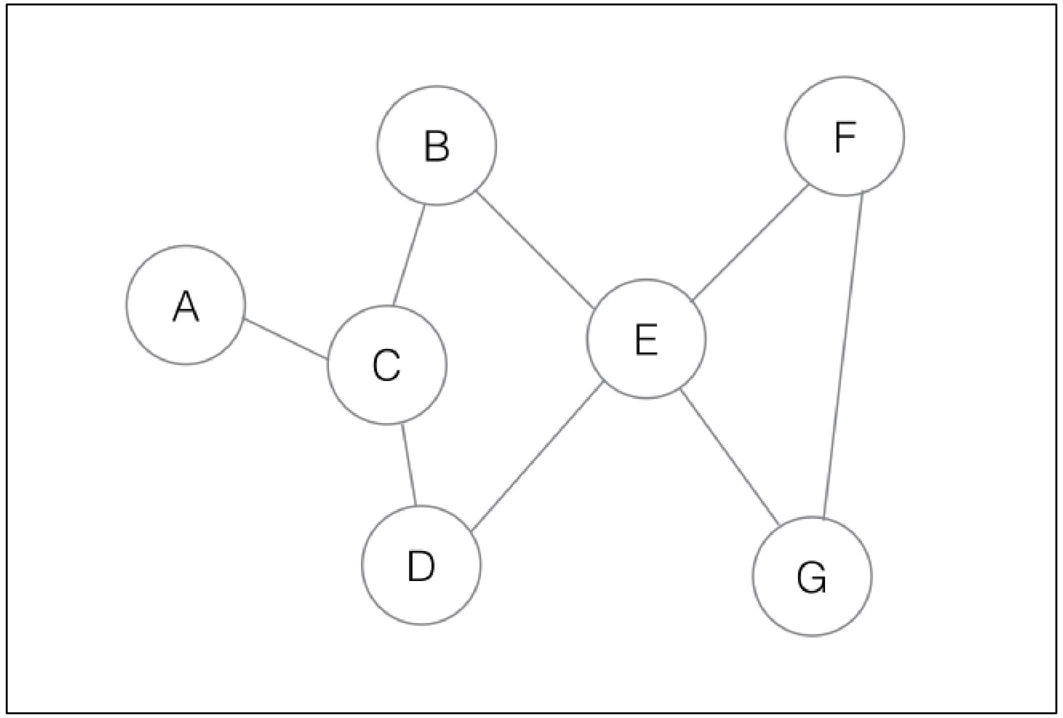


Constraint Satisfaction Problems - CSP

- For the graph below, we would like to color each node with either Red, Green, or Blue, such that no two adjacent nodes have the same color

(a) Define variables, Domains and Constrains for the below node coloring problem

(b) According to the degree heuristic, which node would you consider first? Why?



Graph Search

- Why is graph-search DFS complete (for problems with a finite set of states), but tree-search DFS is not?
- How do the two heuristic search algorithms Uniform-Cost Search and A* search differ from each other?
- When does regular hill-climbing search fail to find the optimal solution?
- Is $p \lor \neg p$ valid?
- In which models is $\neg p \rightarrow q$ valid?
- Show that $\neg[p \lor \neg(\neg q \lor \neg r)]$ is logically equivalent to $(p \lor q) \rightarrow \neg(p \lor r)$
- Define Modus Ponens in propositional logic.