In This Chapter

Why learning?

Supervised (Error based) learning

Gradient descent learning: Least Mean Square, Back Propagation algorithm

Stochastic learning

Unsupervised learning

Hebbian learning algorithm

Competitive learning

Reinforced learning (output based)

Genetic algorithms: operators

What is Learning?

- Learning is the process of acquiring new or modifying existing knowledge, behaviors, skills, values, or preferences.
- Evidence that learning has occurred can be observed through changes in behavior, ranging from simple to complex.

What is Machine Learning?

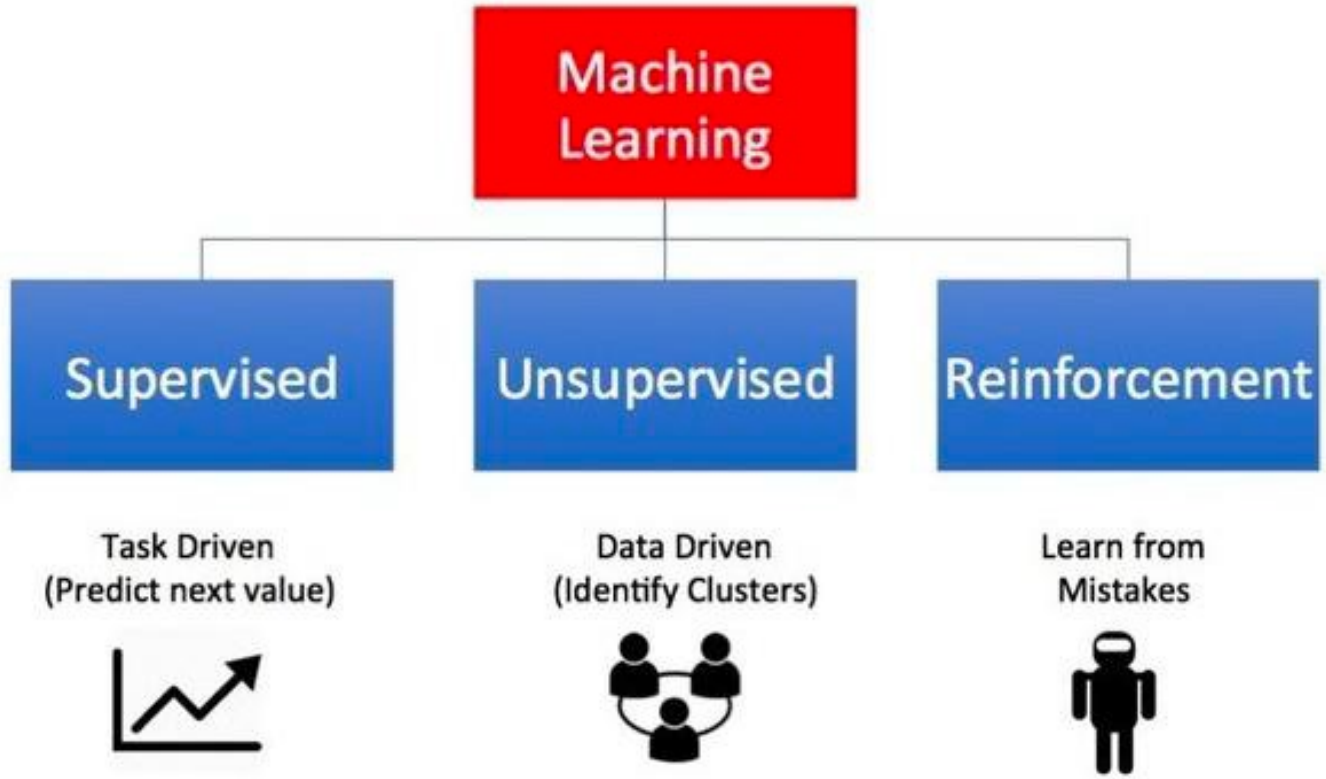
- Machine learning refers to changes in a system that are adaptive, allowing the system to perform the same task more effectively in the future.
- Similar to how humans learn from past experiences, computer systems learn from data, which represents "past experiences" of a specific application domain.
- As a result, machine learning enables computers to learn and improve without being explicitly programmed.

Why Machine Learning?

- To learn a target function (the relationship between input and output) that can predict the values of a discrete class attribute, such as male or female, or high-risk or low-risk.
- To model the underlying structure or distribution in the data, providing deeper insights into the data.
- To learn behaviors through trial-and-error interactions with a dynamic environment.

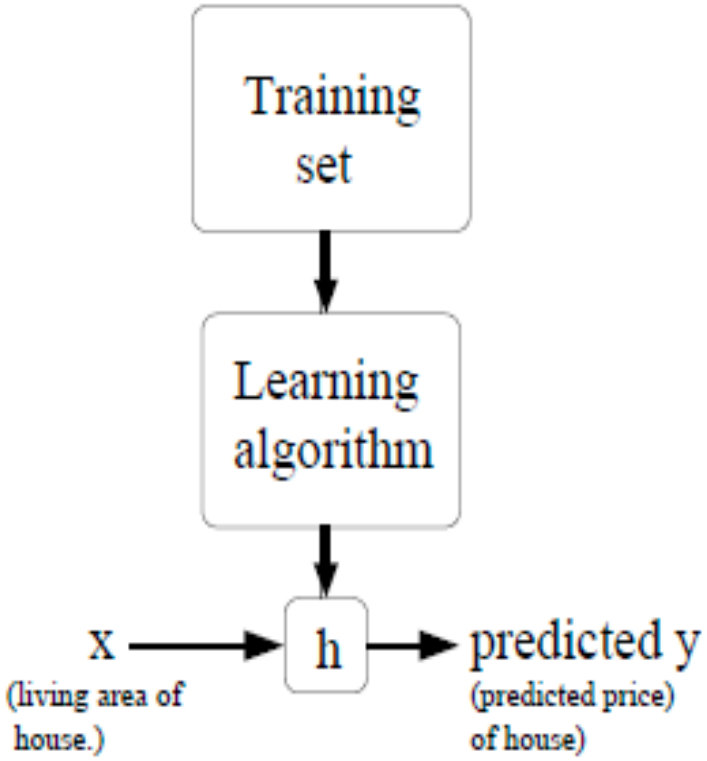
Types of Machine Learning

- Machine learning algorithms are classified into the following three categories based on the training set used:



Supervised Learning

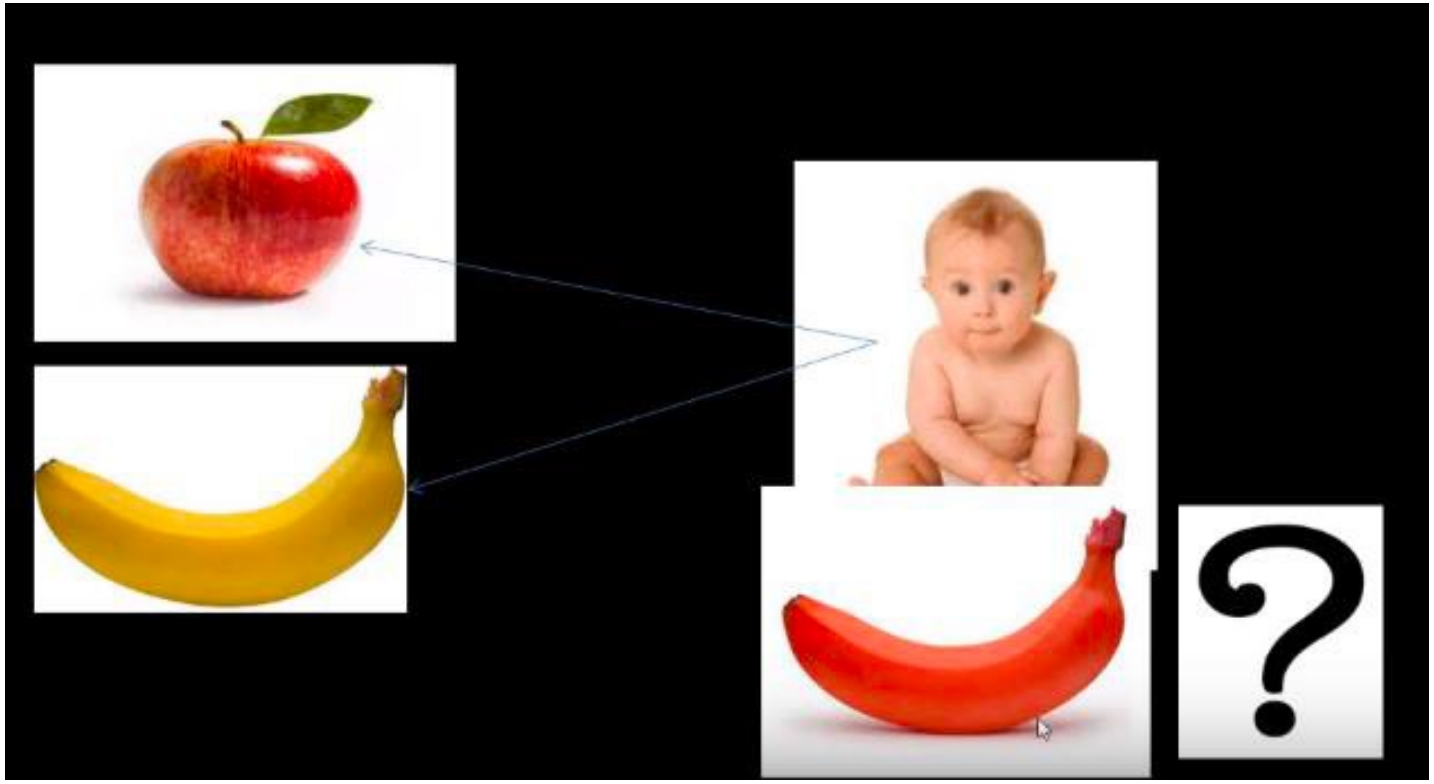
- Supervised learning involves input variables (X) and an output variable (Y), and an algorithm is used to learn the mapping function from the input to the output, expressed as Y = f(X).
- Learning stops when the algorithm achieves an acceptable level of performance.
- This approach allows for the prediction of output variables (Y) when new input data (X) is provided.



Contd..

- It is called supervised learning because the algorithm's learning process from the training dataset resembles a teacher supervising the learning process.
- Since the correct answers are known, the algorithm iteratively makes predictions on the training data and is corrected by the teacher.

Example



Unsupervised Learning

- Unsupervised learning involves only input data (X) without corresponding output variables (targets/labels).
- The goal is to model the underlying structure or distribution in the data to gain a deeper understanding of it.
- It is called unsupervised learning because, unlike supervised learning, there are no correct answers or a teacher guiding the process.
- Algorithms independently discover and present interesting structures within the data.
- Example: Clustering.

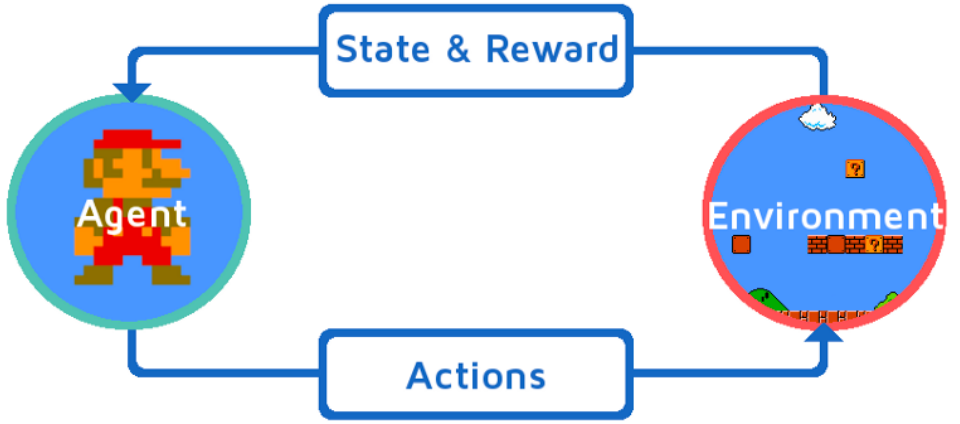
Example

Example


Income
2000
1500
4000
4500
5000
Low Earning
High Earning
Data is not Labelled

Reinforcement Learning

- Reinforcement learning involves learning behavior through trial-and-error interactions with an environment.
- The goal is to learn how to act in order to maximize rewards (encouragements).
- This type of learning emphasizes feedback that evaluates the learner's performance without providing explicit standards of correctness in the form of behavioral targets.
- Examples include learning to ride a bicycle or playing games.



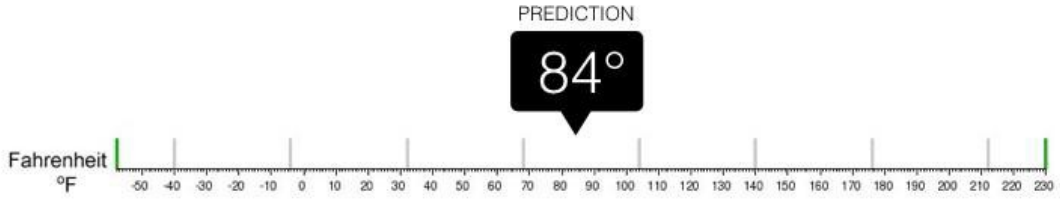
Supervised Learning Algorithms

- **Classification:**
  - Used to predict the outcome of a given sample where the output variable is in the form of categories (discrete). Examples include labels such as male and female, or sick and healthy.
- **Regression:**
  - Used to predict the outcome of a given sample where the output variable is in the form of real values (continuous). Examples include real-valued labels denoting the amount of rainfall or the height of a person.

Example

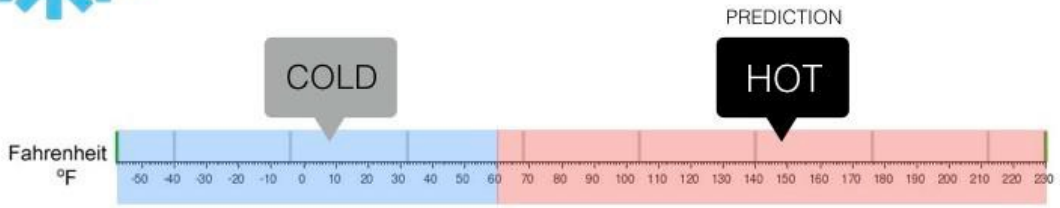
Regression

What is the temperature going to be tomorrow?



Classification

Will it be Cold or Hot tomorrow?



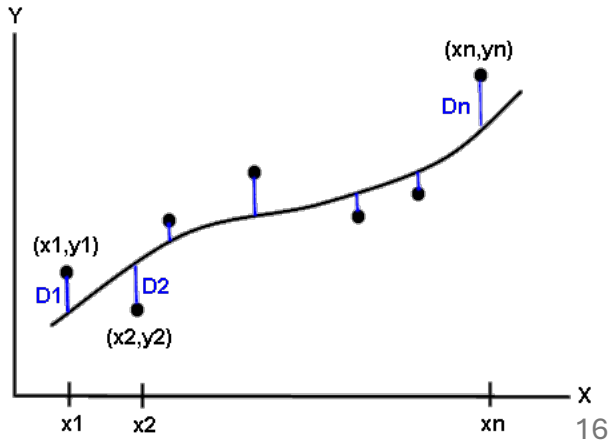
Least Mean square Regression algorithm

To build the predicted output O that closely matches the standard output t (target):

- The model is represented as $f(x) = O = w_0 + w_1 x_1 + \ldots + w_n x_n$
- Train (adjust or choose) the w_i's to minimize the squared error:

$$E[w_1, \ldots, w_n] = \frac{1}{2} \sum_{i=1}^{m} (t_i - o_i)^2$$

- Determine the output for new input based on the line which has the minimum value of the initial squared error.



Gradient Descent Algorithm

The gradient descent algorithm starts with initial guesses for weights (coefficients) W_i and iteratively updates them until convergence:

- Update rule: $W_i = W_i - \eta \nabla E[W]$
  - Where,
    - η is the learning rate, step size, or rate of weight adjustment.
    - Gradient $\nabla E[W]$:

$$\nabla E[W] = \left[\frac{\partial E}{\partial w_0}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{i=1}^{m} (t_i - o_i)^2$$

$$\frac{\partial E}{\partial w_i} = \sum_{i=1}^{m} (t_i - o_i)(-x_i)$$

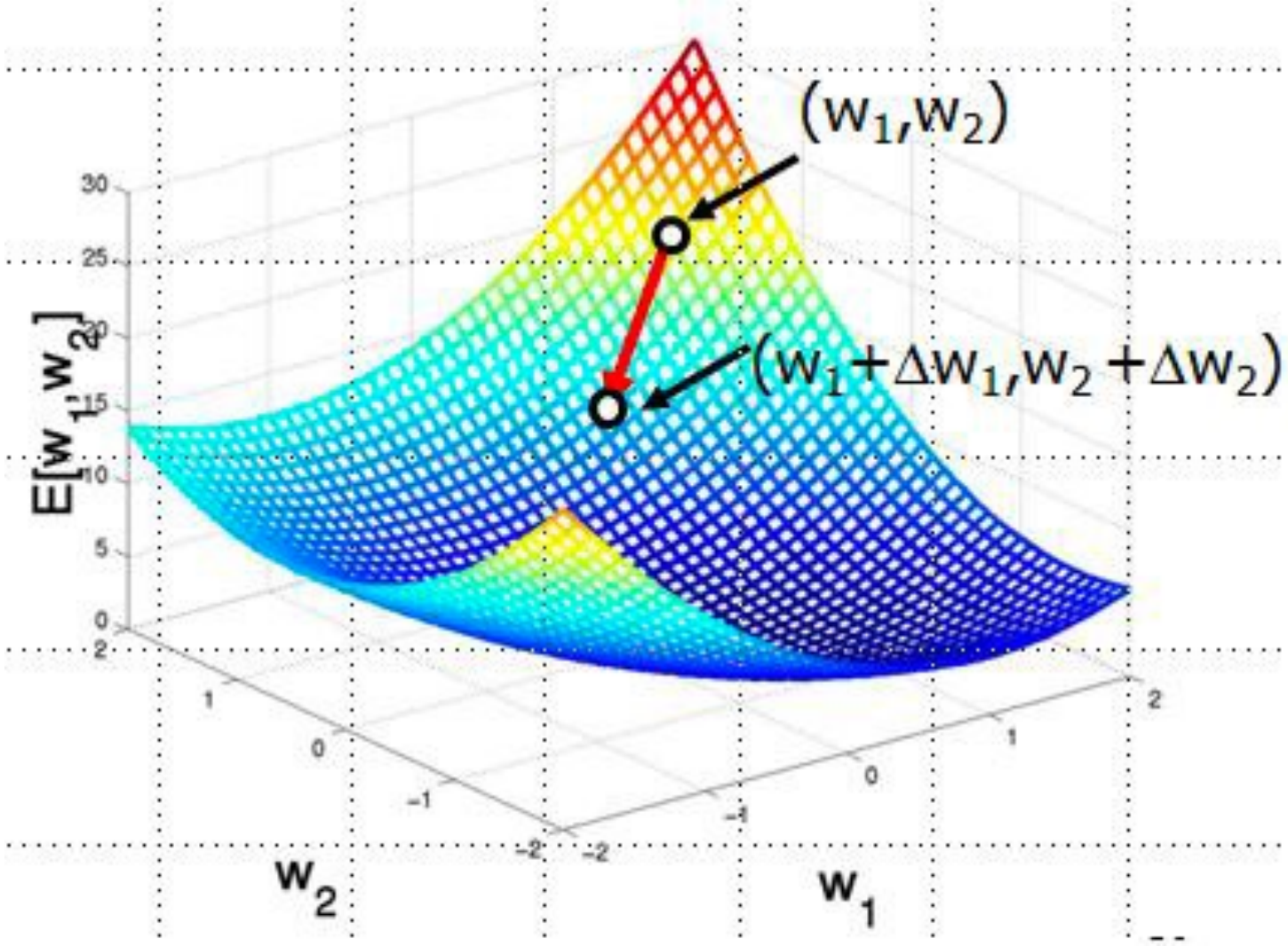
The algorithm adjusts the weights W_i based on the gradient of the error function E with respect to the weights, moving them towards minim↓g the squared error between predicted o_i and target t_i values.

Contd..

- The update rule for a single training example is:

$$W_i = W_i + \eta \cdot (t_i - o_i) \cdot (x_i)$$

- This update rule is known as the Least Mean Squares (LMS) update rule.



Gradient-Descent(training_examples, l)

- Each training example is a pair of the form <(x1,…xn),t> where (x1,…,xn) is input values, and t is the target output value, l is the learning rate (e.g. 0.1)
- Initialize each wi to some small random value
- Until the termination condition is meet, Do
  - –Initialize each Δwi to zero
  - –For each <(x1,…xn),t> in training_examplesDo
    - Input the instance (x1,…,xn) to the linear function and compute the output o
    - Calculate change in weight

      Δwi= l* (t-o) xi
- For each weight wi Do
  - wi=wi+Δwi

Contd..

- If we have more than one training example, there are two ways to modify the gradient descent algorithm to handle a training set with multiple examples:
  1. **Batch Gradient Descent**
  2. **Stochastic (Incremental) Gradient Descent**

Batch Mode: Gradient Descent

- This method considers every example in the entire training set at each step, making it a static approach.
- **Algorithm:**
  - Repeat until convergence

    For each weight wi {

    wi = wi + learning_rate * Σ(i=1 to m) (ti - oi) * xi over the entire dataset D

    }

Stochastic (Incremental) Gradient Descent

- This algorithm repeatedly runs through the training set, updating weights dynamically each time it encounters a training example.

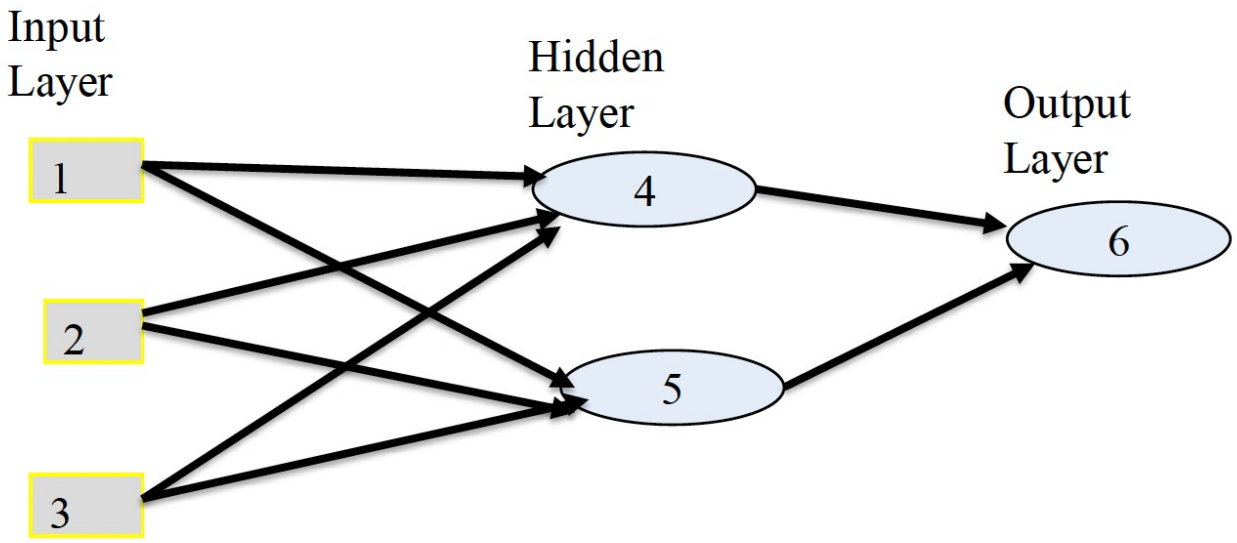
```
For each training example (x1, …, xn, t) in the training set:
    For each weight wi {
      wi = wi + learning_rate * (t - o) * xi
    }
```

Home work

- Comparison Between Batch Gradient Descent and Stochastic Gradient Descent.

Artificial Neural Network

- A neural network is composed of number of nodes or units , connected by links. Each link has a numeric weight associated with it.



- Artificial neural networks are programs design to solve any problem by trying to mimic the structure and the function of our nervous system.

Artificial neural network model:

- Input to the network are represented by mathematical symbol x_n.
- Each of these inputs are multiplied by a connection weight, w_n

$$sum = w_1x_1 + w_2x_2 + \ldots\ldots + w_nx_n$$

- These products are simply summed, fed through the transfer function f() to generate result and output

Backpropagation Algorithm

- Backpropagation is a neural network learning algorithm that adjusts the weights to predict the correct class label of the input.
- **Input:**
  - D: Training dataset and their associated class labels
  - l: Learning rate (typically between 0.0 and 1.0)
- **Output:**
  - A trained neural network

Method:

1. **Initialize** all weights and biases in the network.
2. **Repeat** until the termination condition is satisfied:

   For each training tuple x in D:

   2.1 calculate output:

   For input layer

   $$O_j = I_j$$

   For hidden layer and output layer

   $$I_j = \sum_k O_k * W_{kj} + \theta_j$$

   $$O_j = \frac{1}{1 + e^{-I_j}}$$

Contd..

2.2 Calculate error:

For output layer:

$$err_j = O_j(1-O_j)(\tau_j - O_j)$$

For hidden layer:

$$err_j = O_j(1-O_j)\sum_k (err_k * W_{jk})$$

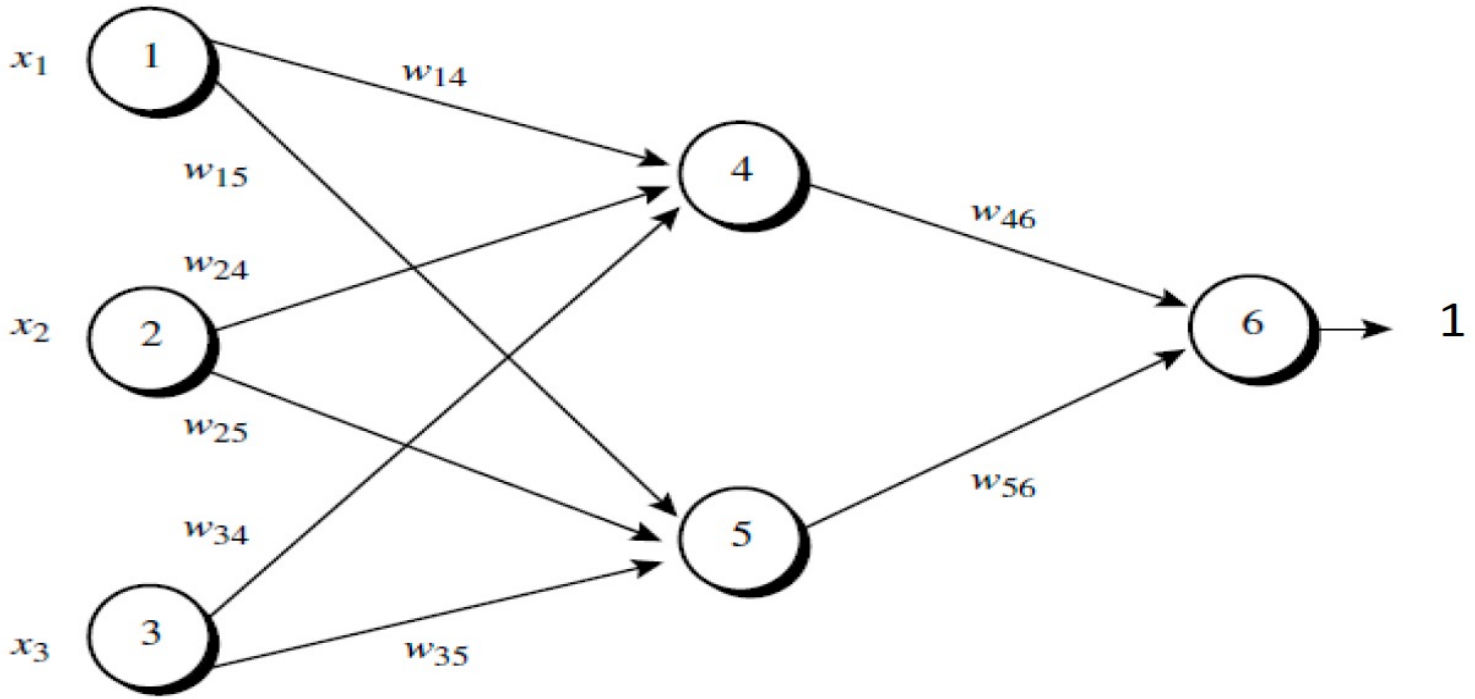
2.3 Update weight

$$W_{ij}(new) = W_{ij}(old) + l * O_i * err_j$$

2.4 Update bias

$$\theta_j = \theta_j + l * err_j$$

- Example: Sample calculations for learning by the back-propagation algorithm.



- Figure above shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table below, along with the first training tuple, *X = (1, 0, 1), whose class label is 1.*

Contd…

- Step1: Initialization
  - Let initial input weight and bias values are:
  - Initial input:

X1	X2	X3
1	0	1

  - Bias values:

θ_4	θ_5	θ_6
-0.4	0.2	0.1

  - Initial weight:

W14	W15	W24	W25	W34	W35	W46	W56
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2

Contd…

- 2. Termination condition: Weight of two successive iteration are nearly equal or user defined number of iterations are reached.
- 2.1For each training tuple X in D, calculate the output for each input layer:
  - For input layer: $O_j = I_j$
    - $O_1=I_1=1$
    - $O_2=I_2=0$
    - $O_3=I_3=1$

Contd…

- For hidden layer and output layer:

1. $$I_4 = O1 * W14 + O2 * W24 + O3 * W34 + \theta 4$$
$$= 1 * 0.2 + 0 * 0.4 + 1 * (-0.5)$$
$$= -0.7$$

$$O_4 = \frac{1}{1+e^{-I_4}} = \frac{1}{1+e^{-(-0.7)}} = \frac{1}{1+(2.7182)^{-(-0.7)}} = 0.332$$

2. $$I_5 = O1 * W15 + O2 * W25 + O3 * W35 + \theta 5$$
$$= 1 * (-0.3) + 0 * 0.1 + 1 * 0.2$$
$$= 0.1$$

$$O_5 = \frac{1}{1+e^{-I_5}} = \frac{1}{1+e^{-(0.1)}} = \frac{1}{1+(2.7182)^{-(0.1)}} = 0.525$$

Contd…

3. $I_6 = \mathrm{O4} * \mathrm{W46} + \mathrm{O5} * \mathrm{W56} + \theta 5$

$= 0.332 * (-0.3) + 0.525 * (-0.2) + 1 * 0.1$

$= -0.105$

$$O_6 = \frac{1}{1 + e^{-I_6}} = \frac{1}{1 + e^{-(-0.105)}} = \frac{1}{1 + (2.7182)^{-(-0.105)}} = 0.474$$

Contd…

- Calculation of error at each node:

- For output layer:

$$err_j = O_j(1-O_j)(\tau_j - O_j)$$

$$err_6 = O_6(1-O_6)(\tau_6 - O_6)$$

$$= 0.474(1-0.474)(1-0.474) = 0.1311$$

- For Hidden layer:

$$err_j = O_j(1-O_j)\sum_k (err_k * W_{jk})$$

$$err_j = O_j(1-O_j)(err_k * W_{jk})$$

$$= \triangle O_5(1-O_5)(err_6 * W_{56})$$

$$= 0.525(1-0.525)(0.1311*(-0.2))$$

$$= -0.0065$$

$$err_j = O_j(1-O_j)(err_k * W_{jk})$$

$$= \triangle O_4(1-O_4)(err_6 * W_{46})$$

$$= 0.332(1-0.332)(0.1311*(-0.3))$$

$$= -0.0087$$

Contd...

- Update the weight: $W_{ij}(new) = W_{ij}(old) + l * O_i * err_j$

1. $W_{46}(new) = W_{46}(old) + l * O_4 * err_6$

$= -0.3 + 0.9 * 0.1311 * 0.332$

$= -0.26$

2. $W_{56}(new) = W_{56}(old) + l * O_5 * err_6$

$= -0.2 + 0.9 * 0.1311 * 0.525$

$= -0.138$

3. $W_{14}(new) = W_{14}(old) + l * O_1 * err_4$

$= 0.2 + 0.9 * -0.0087 * 1$

$= 0.192$

Contd...

4. $$W_{24}(new) = W_{24}(old) + l * O_2 * err_4$$
$$= 0.4 + 0.9 * (-0.0087 * 0$$
$$= -0.4$$

5. $$W_{34}(new) = W_{34}(old) + l * O_3 * err_4$$
$$= -0.5 + 0.9 * (-0.0087 * 1$$
$$= -0.508$$

6. $$W_{15}(new) = W_{15}(old) + l * O_1 * err_5$$
$$= -0.3 + 0.9 * (-0.0065) * 1$$
$$= -0.306$$

Contd…

7. $$W_{25}(new) = W_{25}(old) + l * O_2 * err_5$$
$$= 0.1 + 0.9 * (-0.0065) * 0$$
$$= 0.1$$

8. $$W_{35}(new) = W_{35}(old) + l * O_3 * err_5$$
$$= 0.2 + 0.9 * (-0.0065) * 1$$
$$= 0.194$$

Contd…

- Update the Bias value: $\theta_j = \theta_j + l * err_j$

1. $\theta_6 = \theta_{6(old)} + l * err_6$

   $= 0.1 + 0.9 * 0.1311$

   $= 0.218$

2. $\theta_5 = \theta_{5(old)} + l * err_5$

   $= 0.2 + 0.9 * (-0.0065)$

   $= 0.195$

3. $\theta_6 = \theta_{4(old)} + l * err_4$

   $= (-0.4) + 0.9 * (-0.0087)$

   $= -0.408$

And so on until convergence!

Hebbian Learning Algorithm

1. Take number of steps = Number of inputs(X).
2. In each step
   I. Calculate Net input = weight(W) * input(X)
   II. Calculate Net Output (O)= f(net input)
   III. Calculate change in weight = learning rate * O* X
   IV. New weight = old weight + change in weight
3. Repeat step2 until terminating condition is satisfied

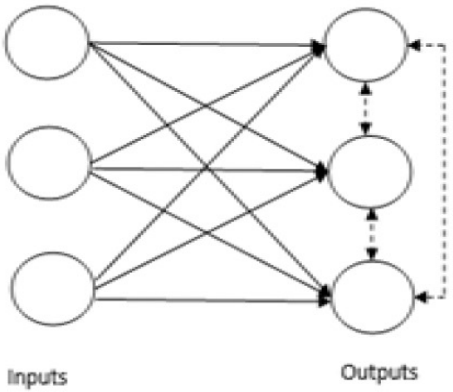
Competitive Learning Rule (Winner-takes-all)

Basic Concept of Competitive Network:

• This network is just like a single layer feed-forward network with feedback connection between outputs.

•The feed- forward connections are excitatory types.

•While the feed-back connections between outputs are inhibitory type, shown by dotted lines, which means the competitors never support themselves.



Basic Concept of Competitive Learning Rule:

- It is unsupervised learning algorithm in which the output nodes try to compete with each other to represent the input pattern.
- Hence, the main concept is that during training, the output unit with the highest Net-input to a given input, will be declared the winner.
- Then only the winning neuron' s weights are updated and the rest of the neurons are left unchanged.
- Therefore , This rule is also called Winner-takes-all

Competitive learning algorithm

- For n number of input and m number of output
- Repeat Until convergence

- Calculate the Net input $net_i = \sum_{j=1}^{n} w_{ij} x_j = \mathbf{w}_i^T \mathbf{x}$

- The outputs of the network are determined by a ``winner-take-all'' competition such that only the output node receiving the maximal net input will output 1 while all others output 0:

$$y_k = \begin{cases} 1 & \text{if } net_k = max\{net_i,\ i = 1, \cdots, m\} \\ 0 & \text{otherwise} \end{cases}$$

- Calculate the change in weight: $\triangle \mathbf{w}_i = \eta(\mathbf{x} - \mathbf{w}_i^{old})$

- Update the weight: $\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + u_i \triangle \mathbf{w}_i$

Where
ui={1 if ith node is winner , 0 else

Genetic Algorithm

GA Introduction

- Search algorithms based on the mechanics of biological evolution.
- Developed by John Holland, University of Michigan
  - To understand the adaptive process of natural systems
  - To design artificial systems software that retains the robustness of natural systems.
- Can be used for finding solutions of:
  - Searching problem
  - Optimization problem

Basic Terminology

Population

Chromosomes

Gene

Allele

Genotype

Phenotype

Encoding & Decoding

Fitness Function

Genetic operators

- a. Selection
- b. Crossover
- c. Mutation

Contd..

Population:

- Subset of solution set
- Solution set – All possible solution member
- Also known as candidate solution
- Collection of chromosomes

Chromosomes:

- Individual solution to the given problem
- Typically encoded as a string or an array of genes.

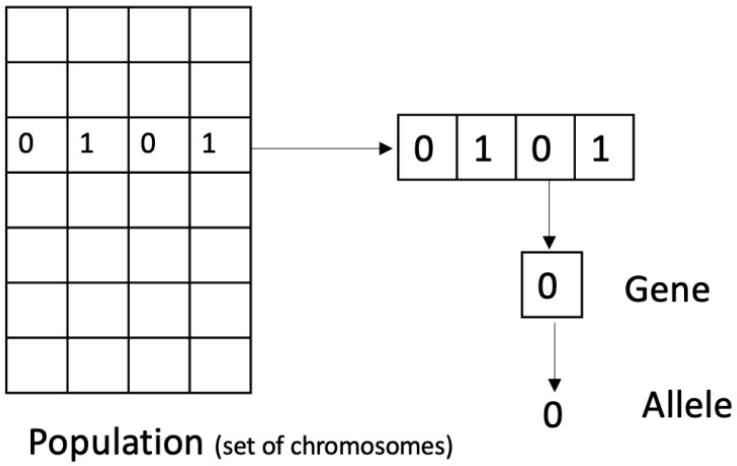
Contd..

Gene:

- One element position of a chromosome.
- Each gene corresponds to a specific parameter or component of the solution.
- The arrangement of genes within a chromosome forms the candidate solution.

Allele:

- The positional value of gene



Contd..

Genotype:

- Mathematical representation format
- Computation space
- Encoding of phenotype solutions

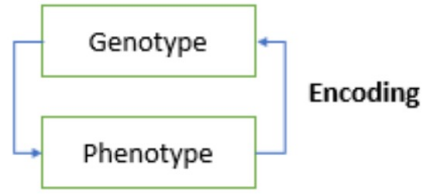
Phenotype:

- How actually it looks like in nature.

Contd..

Encoding:

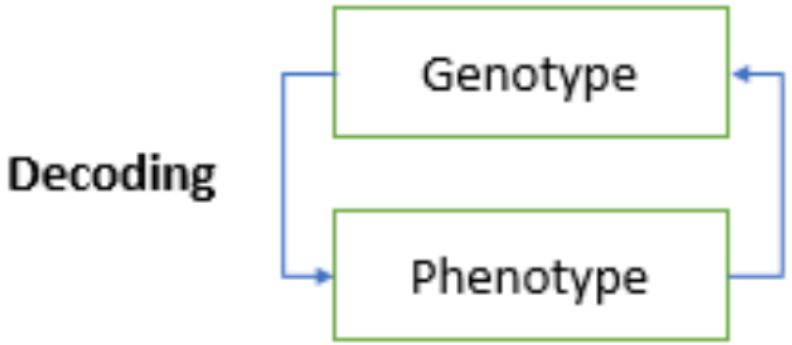
- Conversion of solution set from Phenotype to Genotype.
- It can be: (Genotype Representation)
  - **Binary representation**
    - Boolean 0(absent) or 1(Present) used
  - **Real value representation**
    - Using the actual value of gene (Floating point value)
  - **Integer representation**
    - Integer (whole number) used for representation
  - **Permutation**
    - Positional Significance – e.g. TSP



Decoding:

Contd..

- Reverse of Encoding
- i.e. Conversion from genotype to phenotype.



Contd..

Fitness Function:

- Function used to determine how fit a particular chromosome is.
- How fit the chromosome is toward the optimal solution?

- **Genetic Operators(3 Types):**
  1. Reproduction
  2. Crossover
  3. Mutation

Five phases are considered in a genetic algorithm.

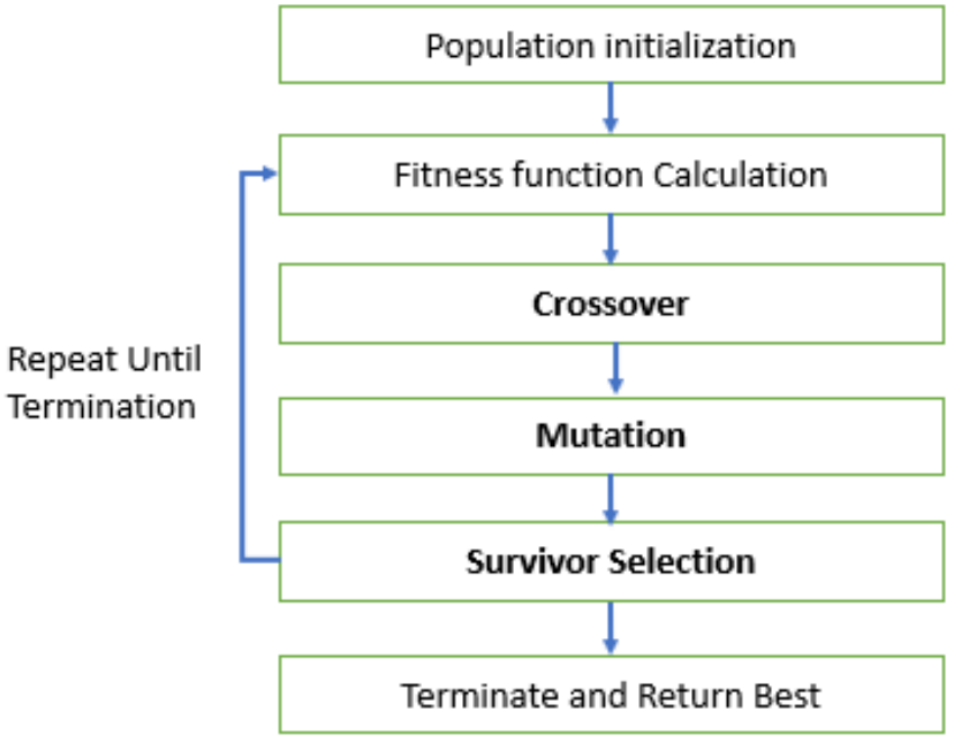


Fig: Genetic Operators in GA

Initialization of Population

- There are two primary methods to initialize a population in a Genetic Algorithm (GA):
  1. **Random Initialization**
  2. **Heuristic Initialization**
- When dealing with a GA population, several factors should be considered:
- **Diversity**: The population's diversity should be maintained to avoid premature convergence.
- **Population Size**: A very large population can slow down the GA, while a smaller population might not provide a sufficient mating pool. Therefore, an optimal population size should be determined through trial and error.

Fitness Function

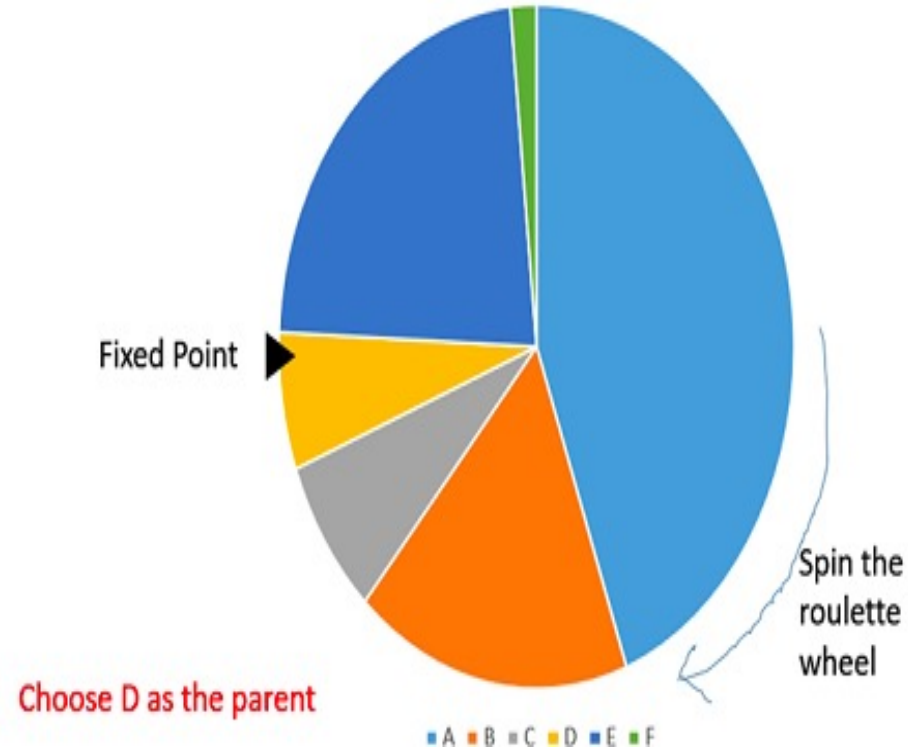
- The fitness function evaluates a candidate solution to the problem, producing an output that indicates how fit the individual is (i.e., their ability to compete with other individuals).
- A fitness function should have the following characteristics:
  - **Efficiency**: It should be sufficiently fast to compute.
  - **Quantitative Measurement**: It must quantitatively assess how fit a given solution is or how fit individuals can be produced from the given solution.

Selection

- The purpose of the selection phase is to choose the fittest individuals and allow them to pass their genes to the next generation.
- Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with higher fitness have a greater chance of being selected for reproduction.
- Selection of parents can be done using one of the following strategies:
  - **Roulette Wheel Selection**
  - **Tournament Selection**
  - **Rank Selection**

Roulette Wheel Selection

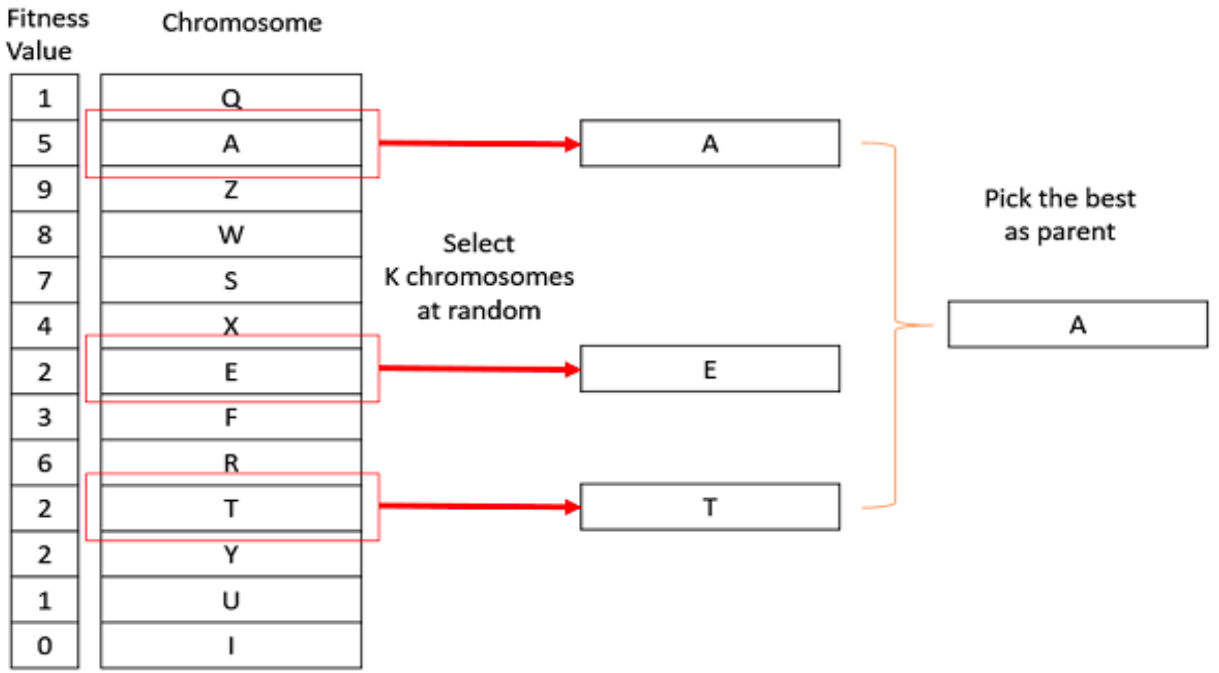
- In Fitness Proportionate Selection, also known as Roulette Wheel Selection, the chance of selection is directly proportional to the fitness value. This means that individuals with higher fitness values have a greater probability of becoming parents, but it's not fixed, allowing any individual to potentially be selected.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

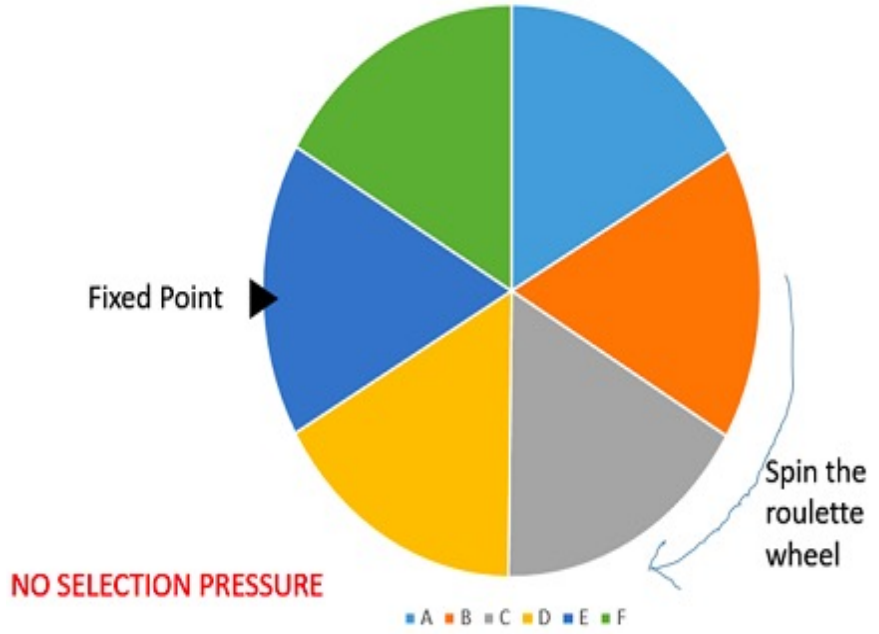
Tournament Selection

- Tournament Selection involves randomly selecting k individuals and then choosing only one among them to become a parent of the next generation.



Rank Selection

- Rank Selection is often used when individuals in the population have very similar fitness values. This results in each individual having nearly equal shares of the selection pie chart, meaning that each individual, regardless of its relative fitness compared to others, has approximately the same probability of being selected as a parent. This can reduce the selection pressure towards fitter individuals, potentially leading to poorer parent selections in such scenarios.



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

Contd..

- In Rank Selection, every individual in the population is ranked based on their fitness. Individuals with higher ranks are preferred more than those with lower ranks when selecting parents for the next generation.

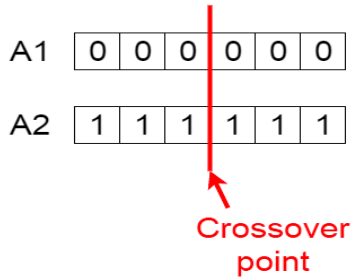
Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

Crossover

- Crossover is a crucial phase in a genetic algorithm. For each pair of parents selected to mate, a crossover point is randomly chosen within their genes.
- Types
  - One - point Crossover
  - Two - point Crossover
  - Uniform Crossover
  - Whole Arithmetic Recombination
  - Davis' Order Crossover (OX1)
  - Cyclic Crossover (CX)
  - Partially Matched Crossover (PMX)

Contd..

- Consider an example, crossover point to be 3 as shown in figure below.



- Offspring are generated by exchanging genes between parents until the crossover point is reached.

Mutation

- In biological terms, offspring do not inherit identical traits from their parents. As they develop, changes occur in their genes, distinguishing them from their parents.
- This phenomenon is called mutation, characterized by random alterations in the chromosome. In practical terms, this means that some bits in the bit string may be flipped.
- Types:
  - ➢ Bit flip mutation
  - ➢ Random resetting
  - ➢ Swap mutation
  - ➢ Inversion mutation

Termination

- The algorithm terminates under the following conditions:
  - When the population has converged, meaning it no longer produces offspring significantly different from the previous generation. At this point, the genetic algorithm is said to have provided a set of solutions to the problem.
  - OR when the predefined number of generations is reached.

Challenges for GA Practitioners

- **Choosing Basic Implementation Issues:**
  - Representation
  - Population size, mutation rate, etc.
  - Selection and deletion policies
  - Crossover and mutation operators
- **Termination Criteria**

Advantages of Genetic Algorithms

- Concept is straightforward and easy to understand.
- Supports multi-objective optimization.
- Effective in noisy environments.
- Always provides a solution; solution quality improves over time.
- Facilitates leveraging previous or alternate solutions.
- Flexible building blocks for hybrid applications.
- Established history and wide range of applications.

When to Use a Genetic Algorithm

- Alternative solutions are too slow or overly complicated.
- Need an exploratory tool to investigate new approaches.
- Problem is similar to one previously solved successfully using a GA.
- Desire to hybridize with an existing solution.
- Benefits of GA technology align with key problem requirements.

- Thank You