Unit 5: Learning LH 6

Presented By : Tekendra Nath Yogi

Tekendranath@gmail.com

College Of Applied Business And Technology

Contd…

- **Outline:**
  - 5.1 Why learning?
  - 5.2 Supervised (Error based) learning
    - 5.2.1 Gradient descent learning: Least Mean Square, Back Propagation algorithm
    - 5.2.2 Stochastic learning
  - 5.3 Unsupervised learning
    - 5.3.1 Hebbian learning algorithm
    - 5.3.2 Competitive learning
  - 5.4 Reinforced learning (output based)
  - 5.5 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 may be seen in changes in behavior from simple to complex.

What is Machine learning?

- The machine Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task more effectively the next time.
- Like human learning from past experiences, computer system learns from data, which represent some “past experiences” of an application domain.
- Therefore, Machine learning gives computers the ability to learn without being explicitly programmed.

Why machine learning?

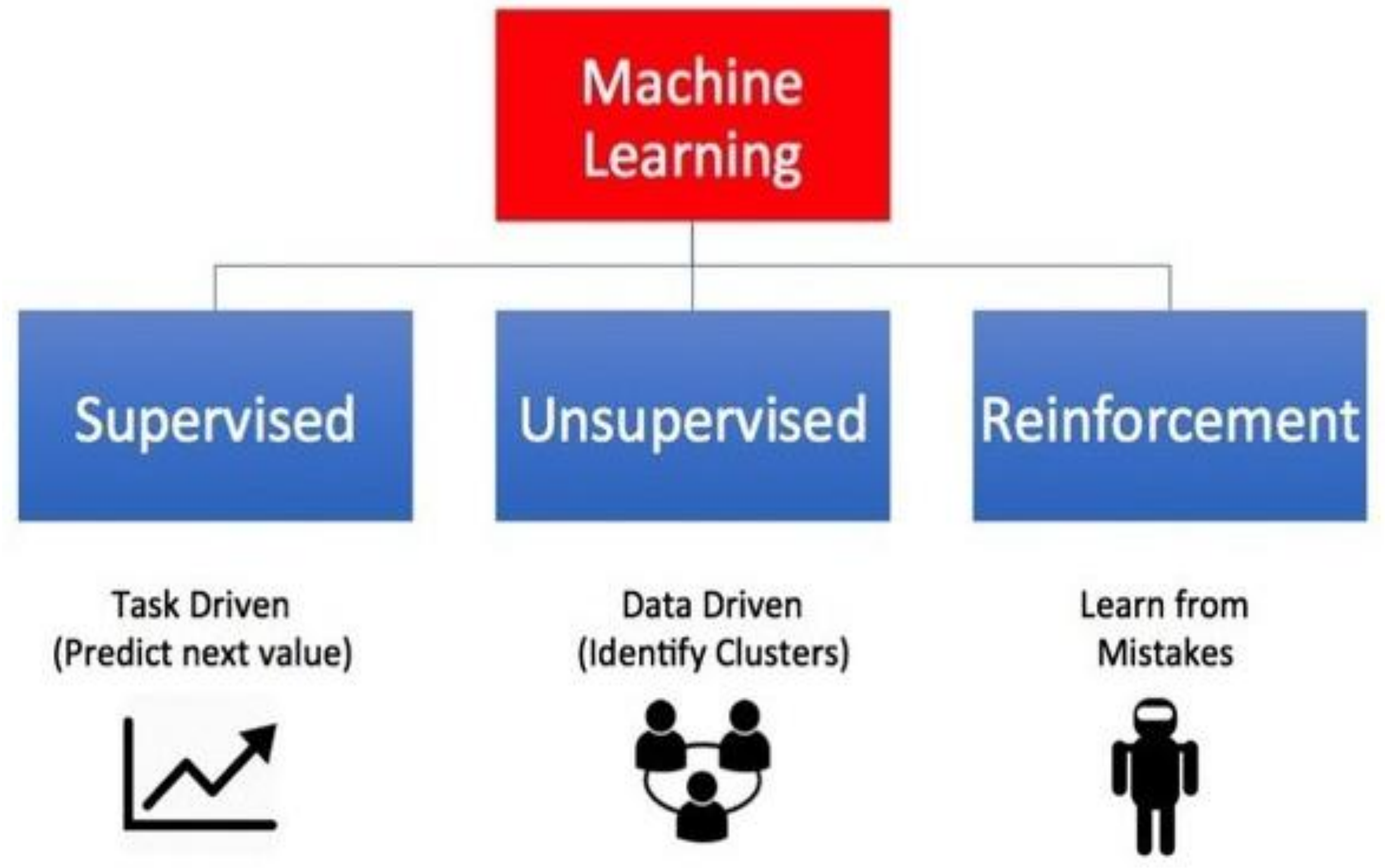
- To learn a target function (relation between input and output)that can be used to predict the values of a discrete class attribute,
  - e.g., male or female, and high-risk or low risk, etc.

- To model the underlying structure or distribution in the data in order to learn more about the data.

- To learns behavior through trial-and-error interactions with a dynamic environment.

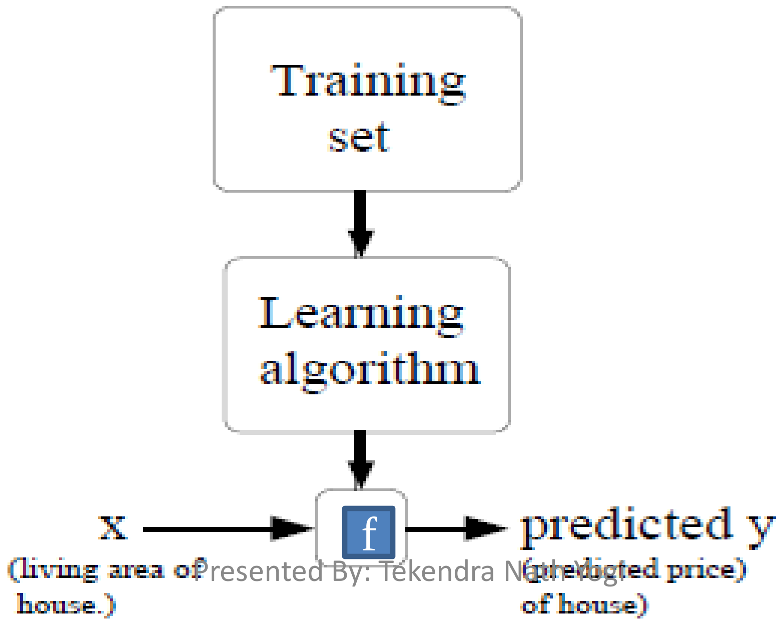
Types of Machine Learning

- Based on training set machine learning algorithms are classified into the following three categories:



Supervised learning

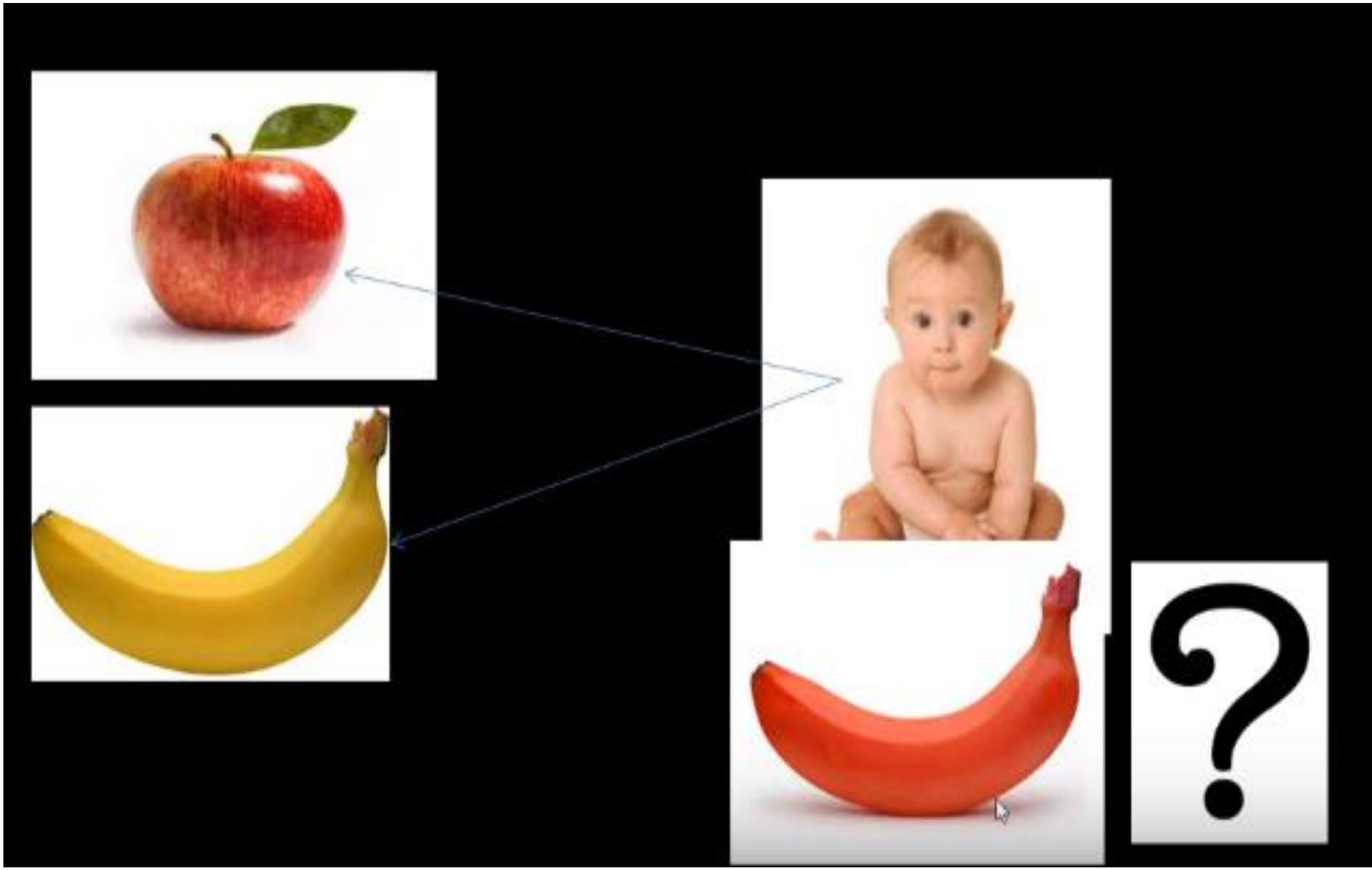
- Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output. Y = f(X)
- Learning stops when the algorithm achieves an acceptable level of performance.
- when you have new input data (x) that you can predict the output variables (Y) for that data.



Supervised learning

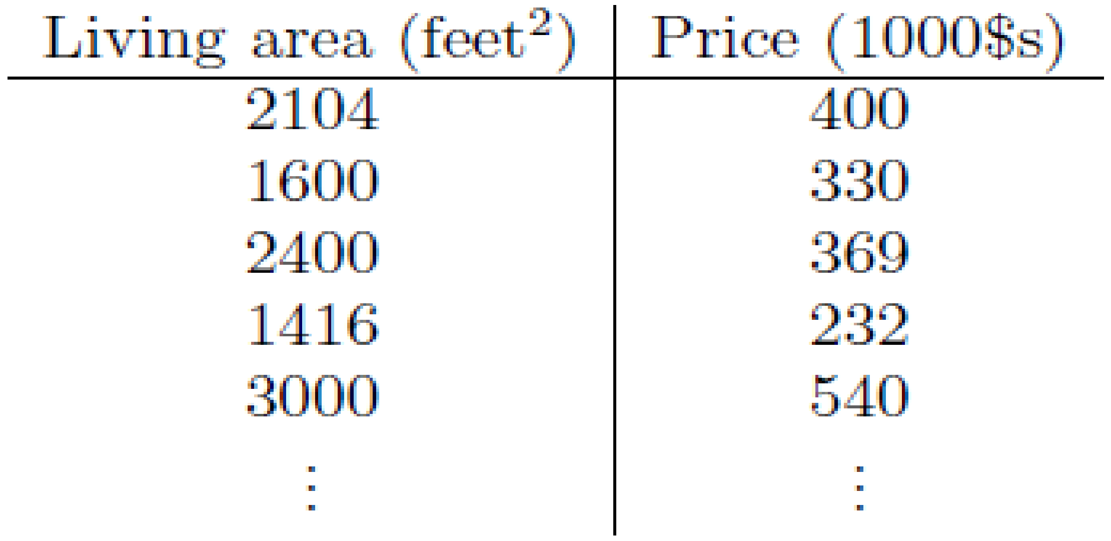
- It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process.
- We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher.

Supervised Learning Example 1



Application Example of Supervised learning

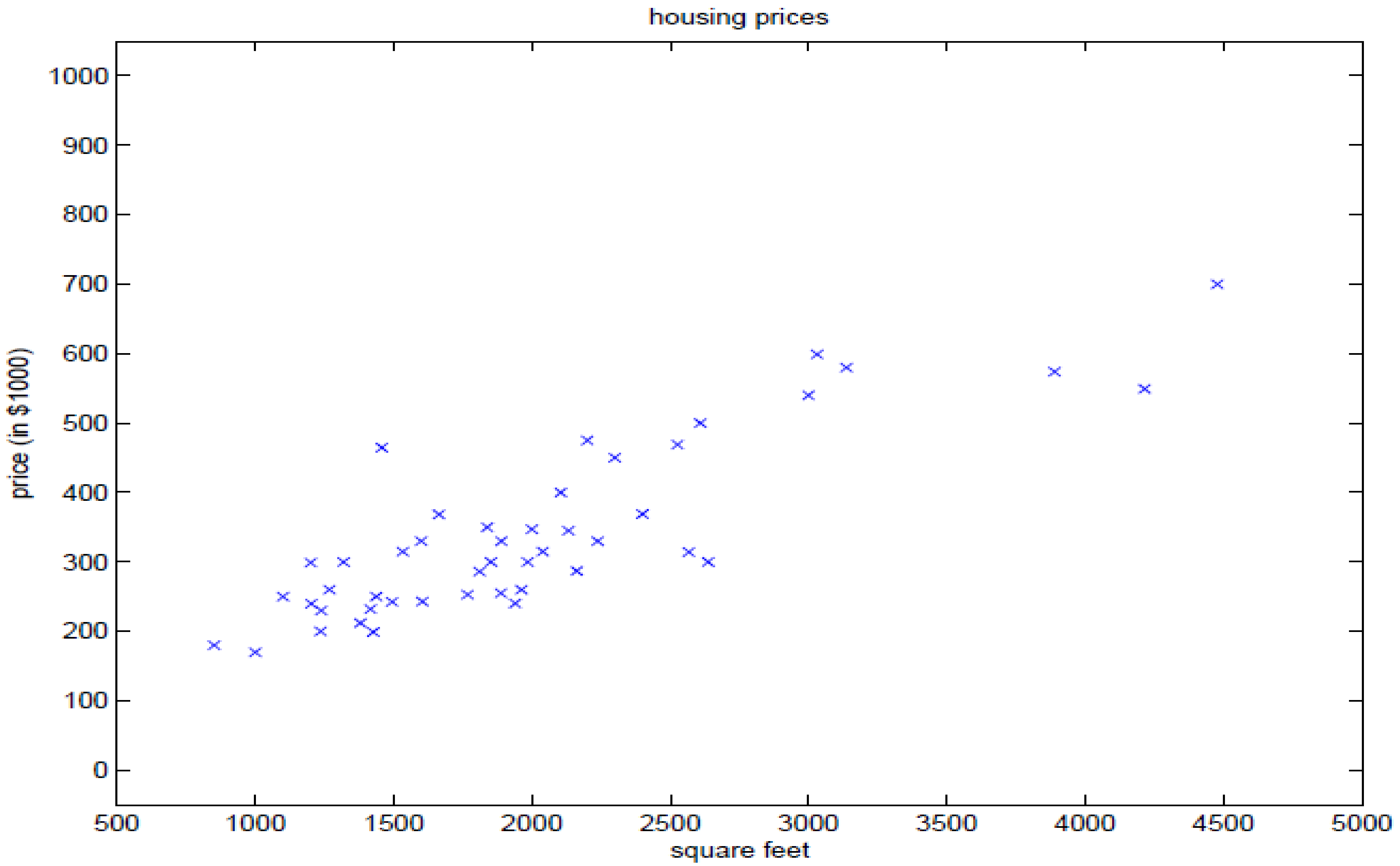
- Suppose we have a dataset giving the living areas and prices of house from KTM:



Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
$\vdots$	$\vdots$

Application Example of Supervised learning

- We can plot this data:



Application Example of Supervised learning

- Given a data like this , we can learn to predict the price of other houses in KTM as a function of the size of their area, such types of learning is known as supervised learning.

- Price of house = f(area of house)

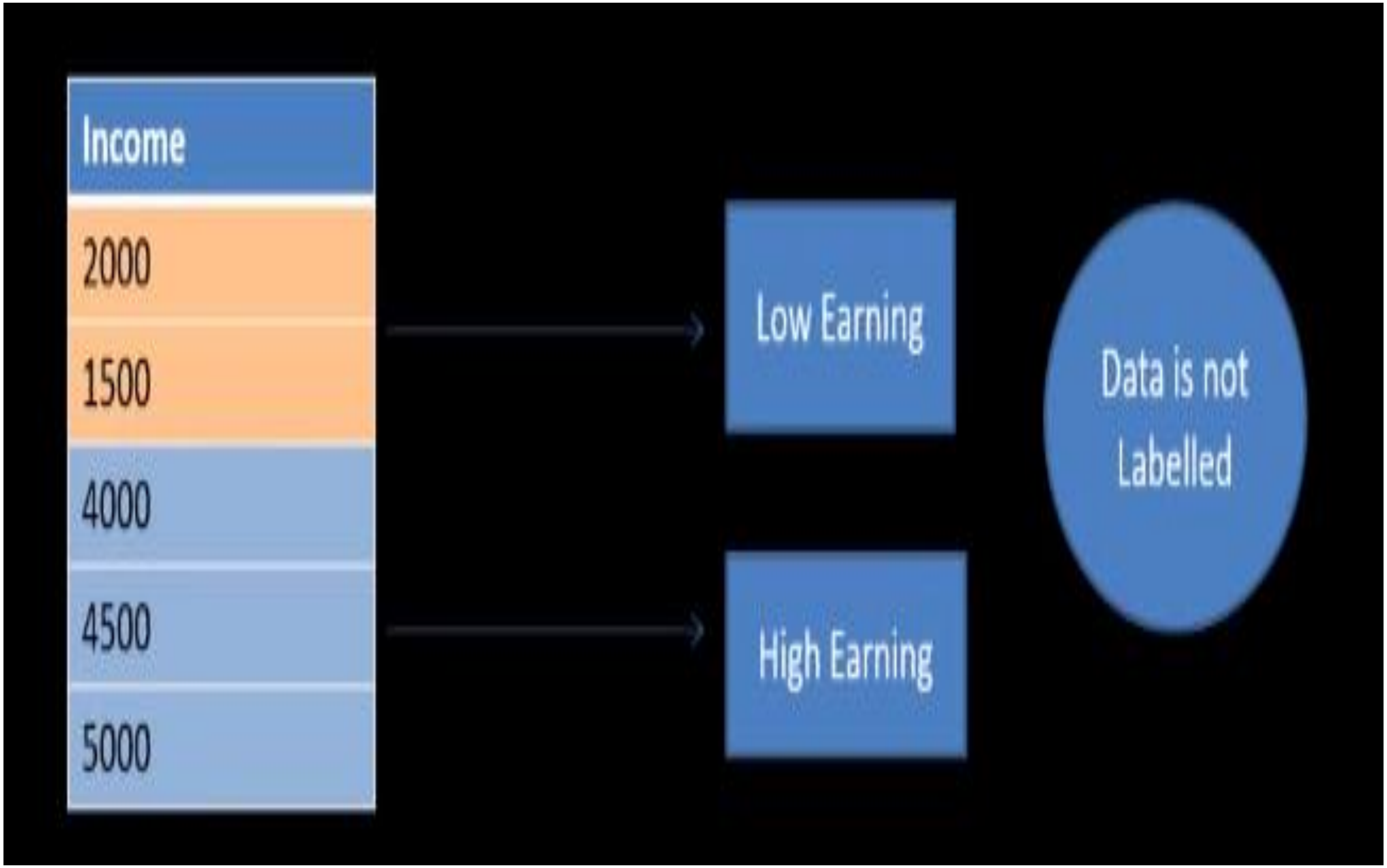
i.e., $y = f(x)$

Unsupervised Learning

- Unsupervised learning is where you only have input data (X) and no corresponding output variables(targets/ labels).
- The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
- These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher.
- Algorithms are left to their own devises to discover and present the interesting structure in the data.
- **E.g., Clustering**

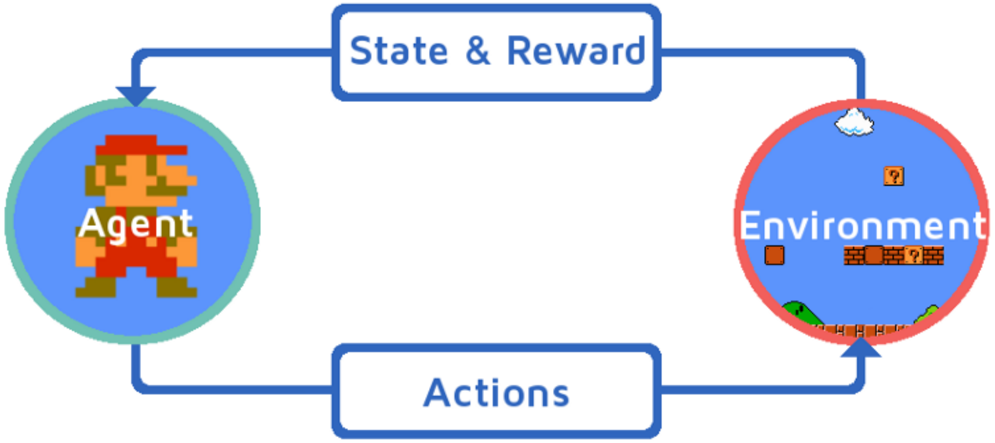
Unsupervised learning Example1

Unsupervised learning example 2

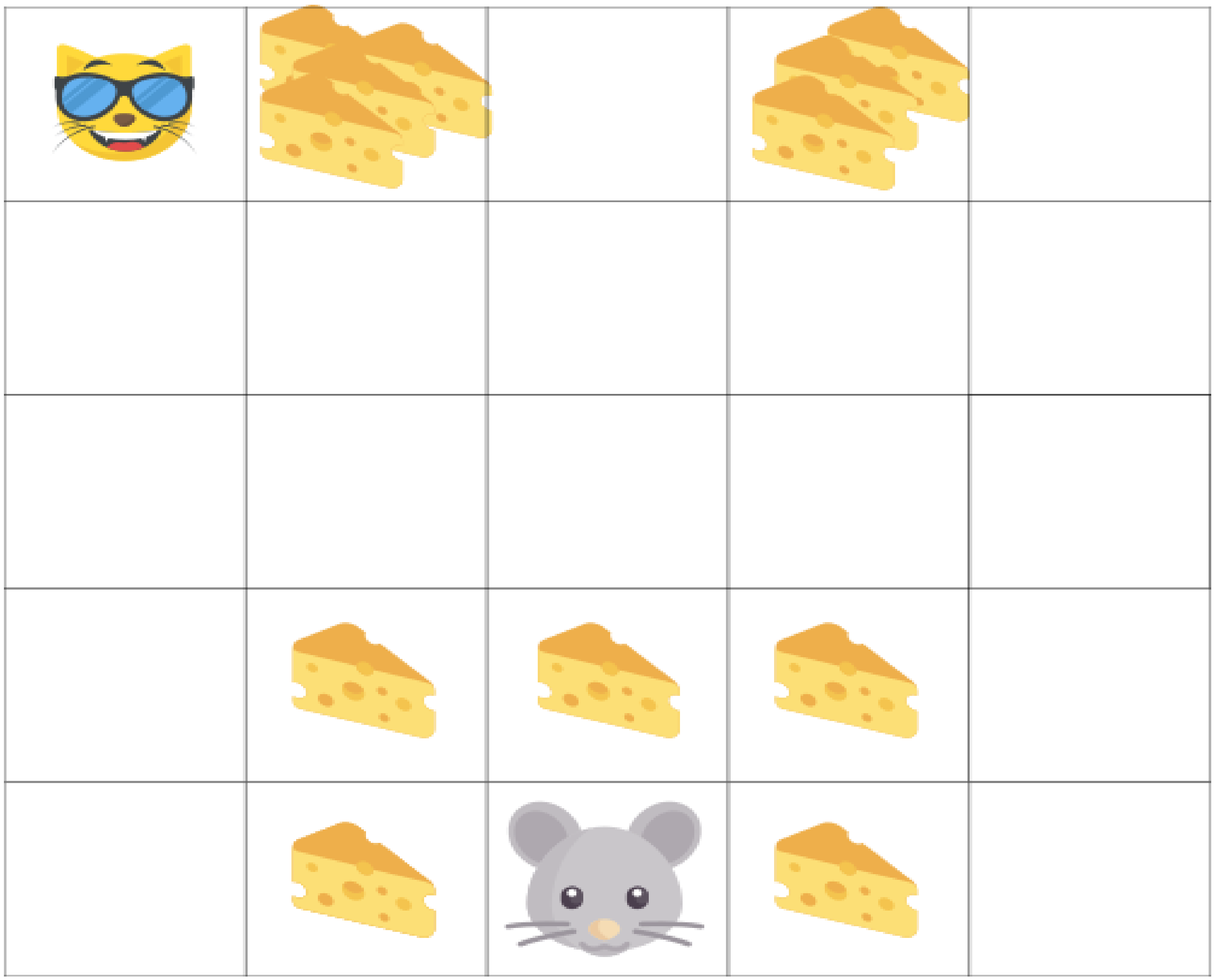


Reinforcement learning

- Is learning behavior through trial-and-error interactions with a environment.
- Is learning how to act in order to maximize a reward (Encouragements).
- Reinforcement learning emphasizes learning feedback that evaluates the learner's performance without providing standards of correctness in the form of behavioral targets.
- Example: Bicycle learning, game playing, etc.



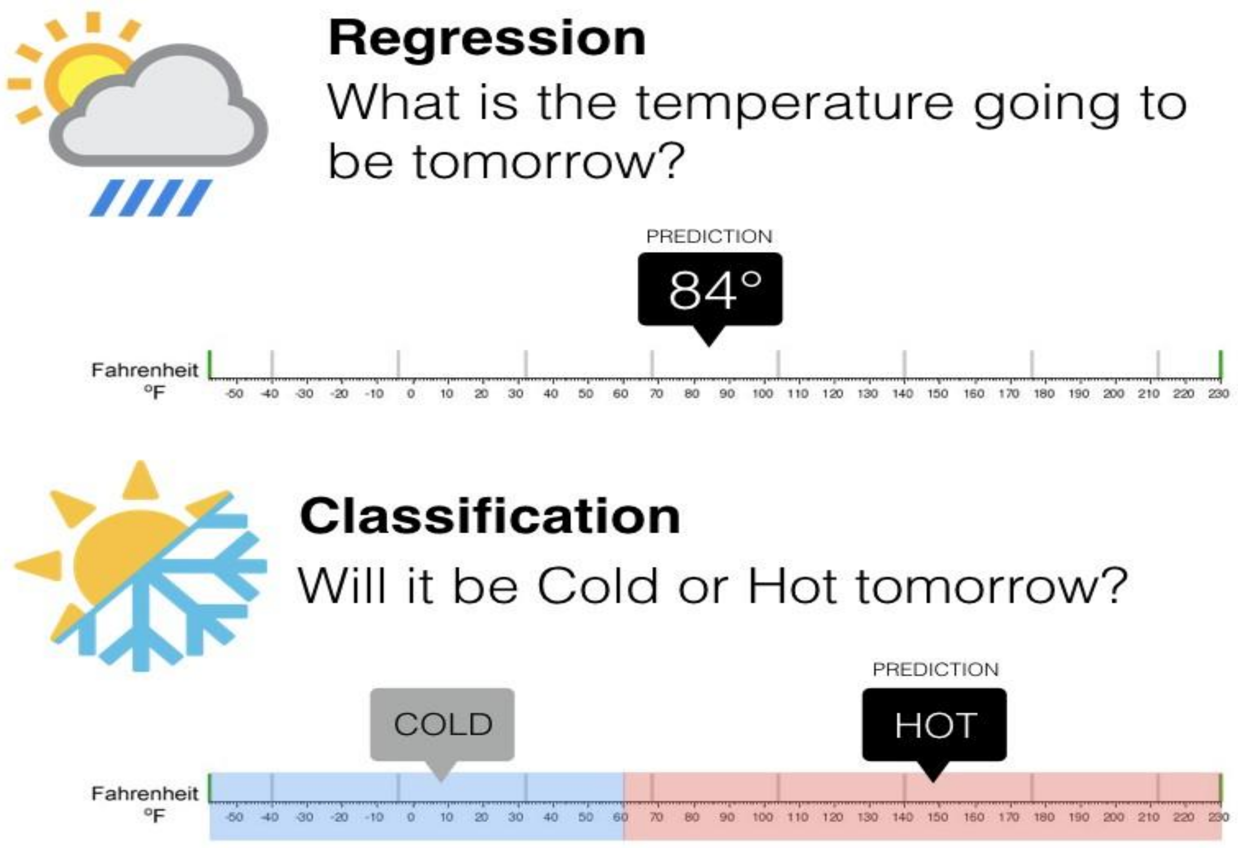
Contd…



Supervised learning Algorithm

- Classification:
  - 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, sick and healthy.
- Regression:
  - 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, the height of a person.

Contd…



Regression

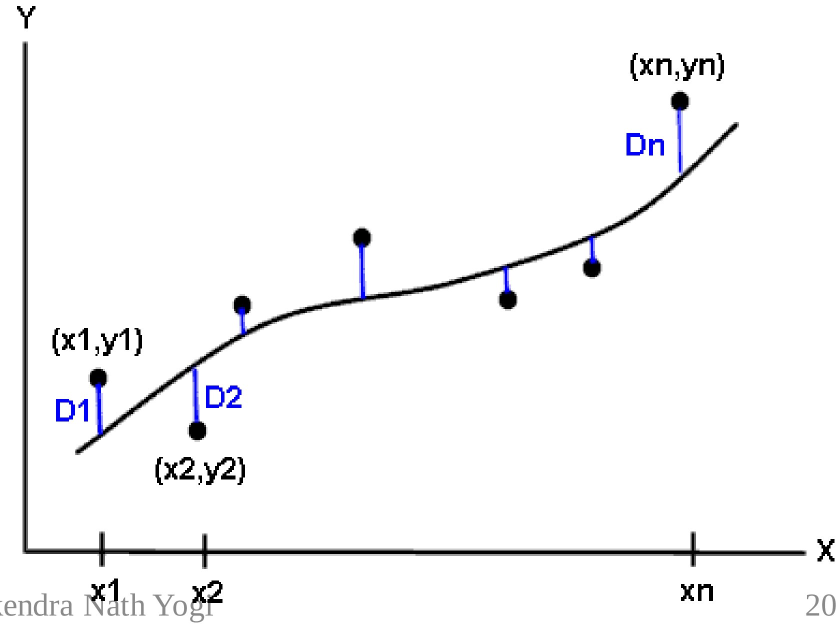
What is the temperature going to be tomorrow?

Classification

Will it be Cold or Hot tomorrow?

Least Mean square Regression algorithm

- For a given data build the predicted output O close to the standard output y i.e., t (target) as:
- $f(x) = O = w_0 + w_1 x_1 + \dots + w_n x_n$
- Train (adjust or choose) the w_i's such that they minimize the squared error
  - $E[w_1,\dots,w_n] = ½ \Sigma_{i=1 \text{ to } m} (t_i - o_i)^2$
- Determine the output for new input based on the lien which has minimum value of initial seared error.



Gradient Descent Learning Rule

- Gradient descent algorithm starts with some initial guess for weights (coefficients) Wi and repeatedly performs the update until convergence :
- $Wi = Wi - l * \nabla E[w]$
- Where,
  - l is learning rate or step size or rate of weight adjustment.
  - Gradient:

$$\nabla E[w] = [\partial E/\partial w_0, \dots \partial E/\partial w_n]$$

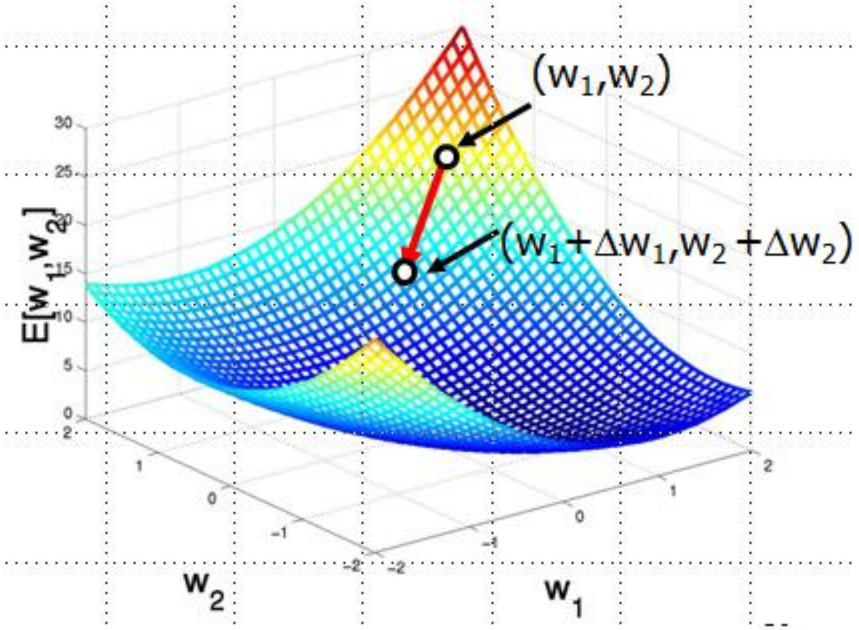
$$\nabla E[wi] = \partial/\partial w_i \ 1/2 \sum_{i=1 \text{ to } m} (t_i - o_i)^2$$

$$= \partial/\partial w_i \ 1/2 \sum_{i=1 \text{ to } m} (t_i - \sum_i w_i x_i)^2$$

$$= \sum_{i=1 \text{ to } m} (t_i - o_i)(-x_i)$$

Contd…

- For a single training example up date rule is:
- Wi= Wi +l*(t_i- o_i)(x_i)
- This is called LMS update rule



Contd…

```
Gradient-Descent(*training_examples*, l)
    Each training example is a pair of the form <($x_1$,…$x_n$),t> where ($x_1$,…,$x_n$) is
    input values, and t is the target output value, l is the learning rate (e.g. 0.1)
• Initialize each $w_i$ to some small random value
• Until the termination condition is met, Do
    – Initialize each $\Delta w_i$ to zero
    – For each <($x_1$,…$x_n$),t> in *training_examples* Do
        • Input the instance ($x_1$,…,$x_n$) to the linear function and compute the
          output o
        • Calculate change in weight
            – $\Delta w_i = l * (t-o) x_i$
    – For each weight wi Do
        • $w_i = w_i + \Delta w_i$
```

Contd…

- If we have more than one training examples:
- Two way to modify this gradient descent algorithm for a training set of more than one example.
  - Batch gradient descent
  - Stochastic(Incremental) gradient descent

Contd…

- Batch mode : gradient descent
  - This method looks at every example in the entire training set on every step. So it is static.
  - Algorithm

```
Repeat until convergene
{
    $w_i=w_i + l* \Sigma_{i=1 \text{ to } m} (t_i- o_i)x_i$  over the entire data D and for each weight
}
```

Contd…

- Stochastic (Incremental mode:)gradient descent
  - This algorithm repeatedly run through the training set and each time encounter a training example. So it is dynamic.
  - Algorithm:

```
for(i=1 to m)
{
    $w_i=w_i +l* (t-o) x_i$          for each weight
}
```

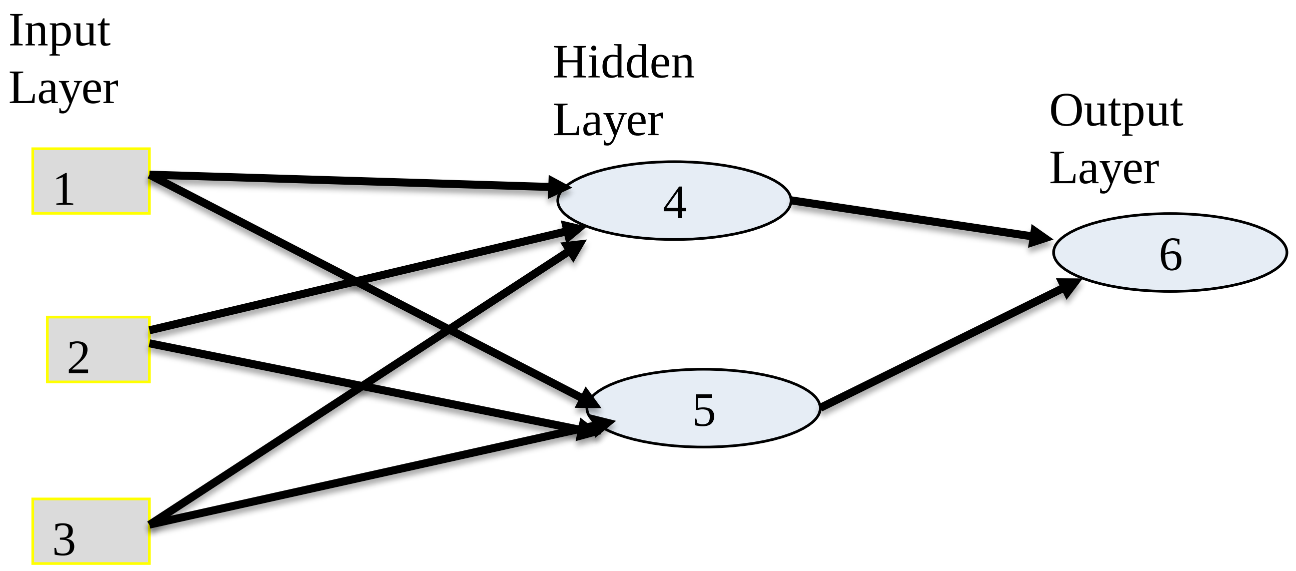
Contd…

- Comparison between batch gradient descent and stochastic gradient descent:

BGD	SGD
For a single step take entire training set	For a single step take only single training example
So a costly operation	It does not matter how large m is
Slower	Faster
Not preferred	Used when m is large

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

$$sum = w_1x_1 + w_2x_2 + ... + w_nx_n$$

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

Back propagation algorithm

- Back propagation is a neural network learning algorithm. Learns by adjusting the weight so as to be able to predict the correct class label of the input.

Input: D training data set and their associated class label

l= learning rate(normally 0.0-1.0)

Output: a trained neural network.

Method:

```
Step1: initialize all weights and bias in network.
Step2: while termination condition is not satisfied .
           For each training tuple x in D
```

Contd…

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

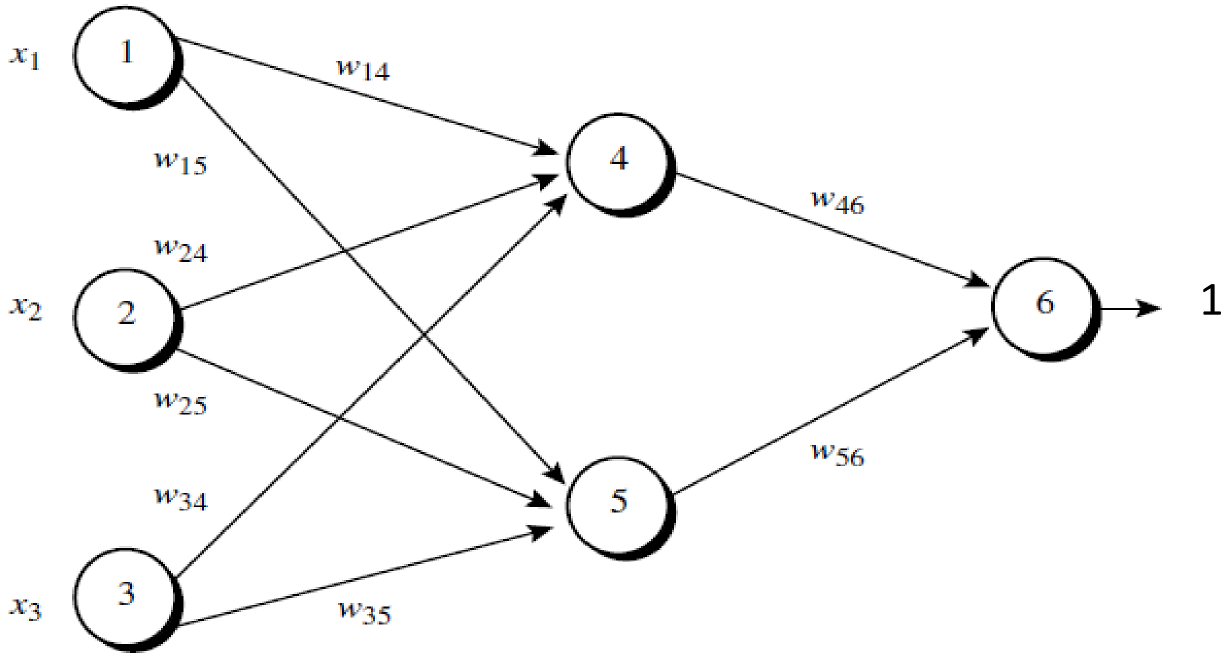
$$W_{ij}(new) = W_{ij}(old) + l * O_i * err_j$$

2.4 Update bias

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

Contd…

- 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. $I_6 = O4 * W46 + O5 * W56 + \theta 5$

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

$= -0.105$

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

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:

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

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

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

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

$$= -0.0065$$

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

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

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

$$= -0.0087$$

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!

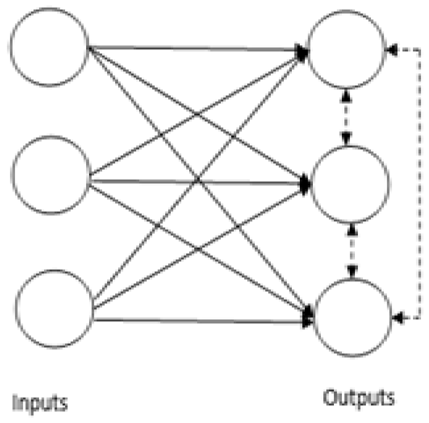
Hebbian Learning Algorithm

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: $\Delta \mathbf{W}_i = \eta(\mathbf{X} - \mathbf{W}_i^{old})$
- Update the weight: $\mathbf{W}_i^{new} = \mathbf{W}_i^{old} + u_i \Delta \mathbf{W}_i$

Where,

$$u_i = \begin{cases} 1 & \text{if ith node is winner} \\ 0 & \text{else} \end{cases}$$

Unit: 5 Genetic Algorithm

Presented By : Tekendra Nath Yogi

Tekendranath@gmail.com

College Of Applied Business And Technology

Copying ideas of Nature

Introduction to Genetic Algorithms

- Nature has always been a great source of inspiration to all mankind.
- A genetic algorithm is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution.

“Select The Best, Discard The Rest”

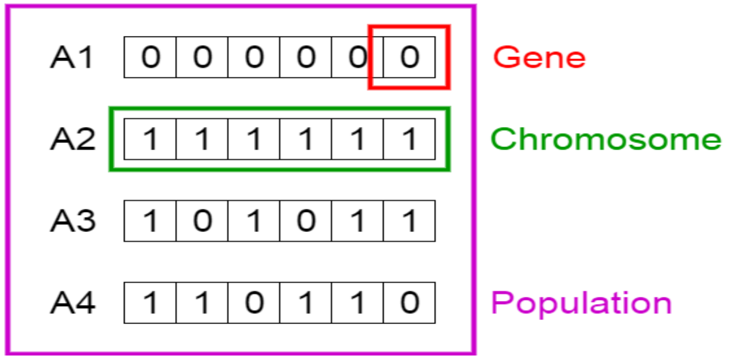
- i.e., the fittest individuals are selected for reproduction in order to produce offspring of the next generation.
- GAs were developed by John Holland and his students and colleagues at the University of Michigan

Notion of Natural Selection

- The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation.
- If parents have better fitness, their offspring will be better than parents and have a better chance at surviving.
- This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
- This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Basic Terminology

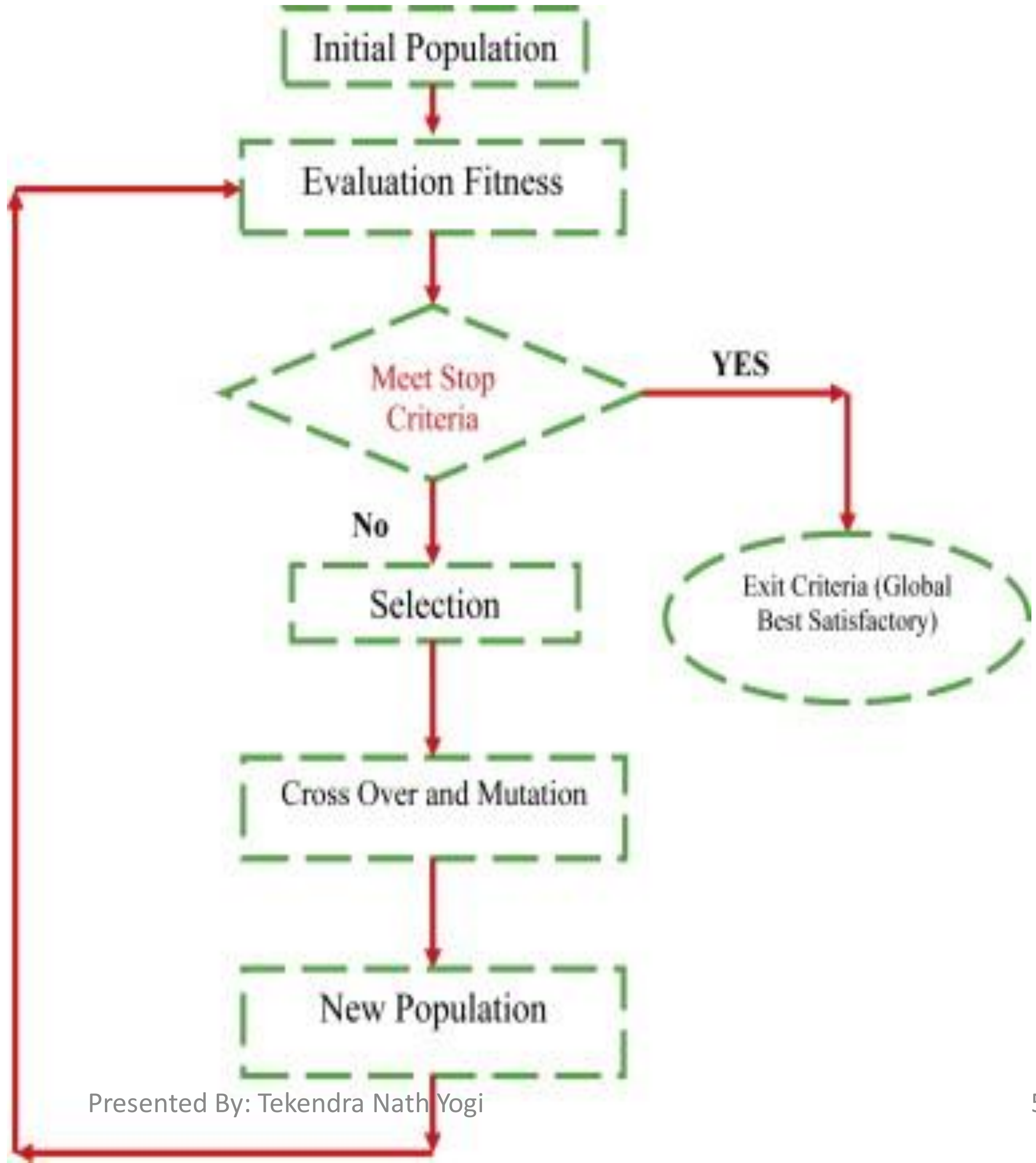
- **Population**: set of individuals each representing a possible solution to a given problem.
- **Gene**: a solution to problem represented as a set of parameters ,these parameters known as genes.
- **Chromosome**: genes joined together to form a string of values called chromosome.
- **Fitness Function:** measures the quality of solution. It is problem dependent.



Nature to Computer Mapping

Nature	Computer
Population	Set of solutions.
Individual	Solution to a problem.
Fitness	Quality of a solution.
Chromosome	Encoding for a Solution.
Gene	Part of the encoding of a solution.
Reproduction	Crossover

Five phases are considered in a genetic algorithm.



Simple Genetic Algorithm

```
Simple_Genetic_Algorithm()
        {
        Initialize the Population;
        Calculate Fitness Function;
        While(Fitness Value != Optimal Value)
                {
                Selection;//Natural Selection, Survival Of Fittest
                Crossover;//Reproduction, Propagate favorable characteristics
                Mutation;//Mutation
                Calculate Fitness Function;
                }
        }
```

Initialization of Population

- There are two primary methods to initialize a population in a GA. They are −
- **Random Initialization**
- **Heuristic initialization**
- There are several things to be kept in mind when dealing with GA population −
  - The diversity of the population should be maintained otherwise it might lead to premature convergence.
  - The population size should not be kept very large as it can cause a GA to slow down, while a smaller population might not be enough for a good mating pool. Therefore, an optimal population size needs to be decided by trial and error.

Fitness Function

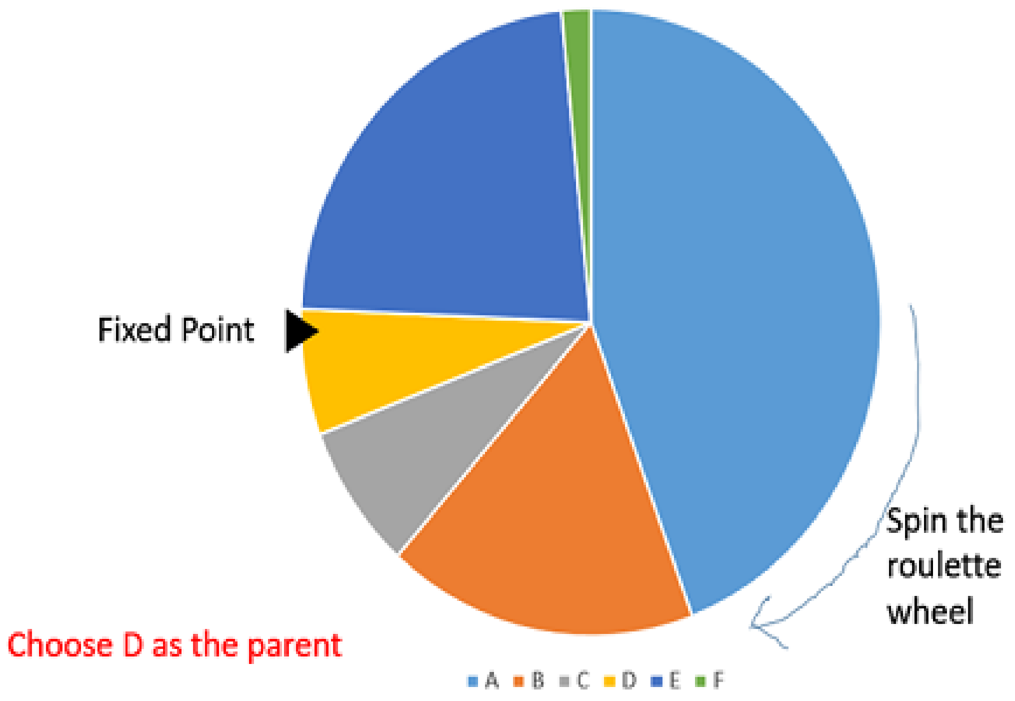
- The fitness function which takes a candidate solution to the problem as input and produces as output how fit an individual is (the ability of an individual to compete with other individuals).
- A fitness function should possess the following characteristics −
  - The fitness function should be sufficiently fast to compute.
  - It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

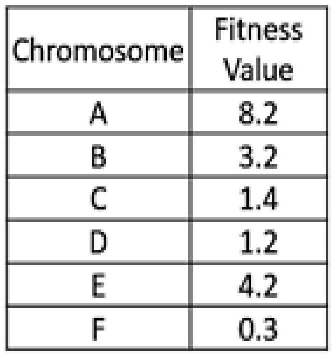
Selection

- The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.
- Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.
- Selection of the parents can be done by using any one of the following strategy:
  - Fitness Proportionate Selection(Roulette Wheel Selection)
  - Tournament Selection
  - Rank Selection

Contd..

- **Fitness Proportionate Selection(Roulette Wheel Selection):**
  - Chance of selection is directly propositional to the fitness value. This is chance not fixed so, any one can become parent.

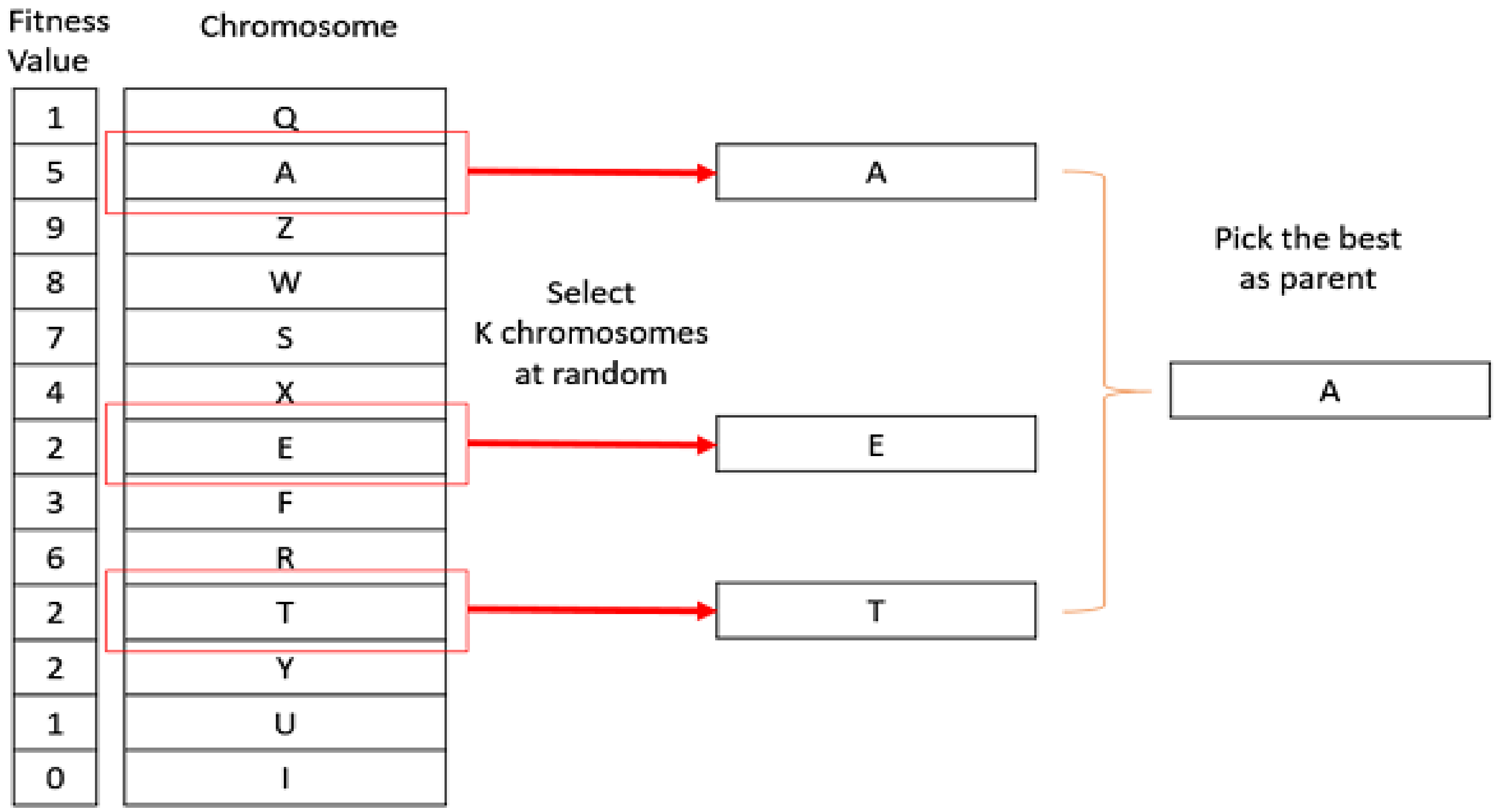




Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

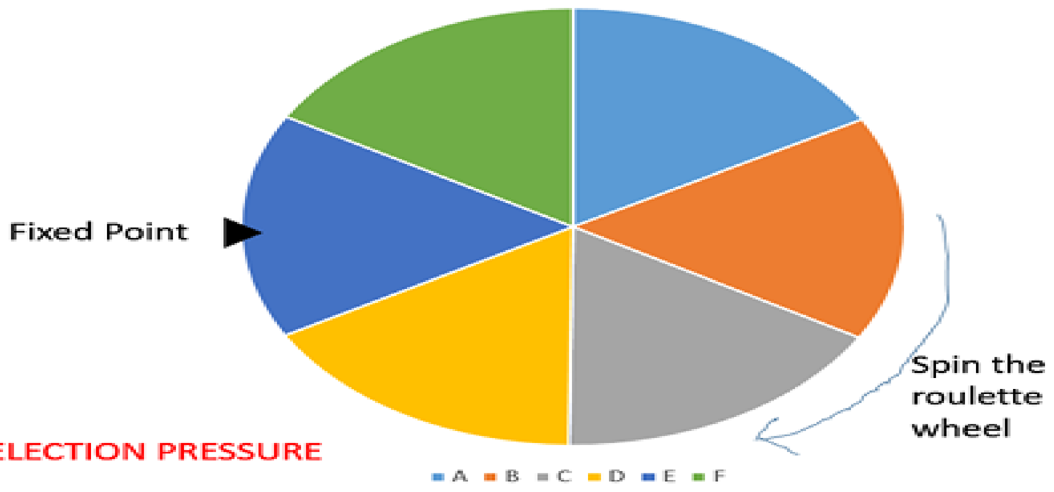
Contd..

- **Tournament Selection:** Take randomly k individuals and select only one among k to make the parent of the next generation.



Contd..

- **Rank Selection:** mostly used when the individuals in the population have very close fitness values . This leads to each individual having an almost equal share of the pie chart. and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.

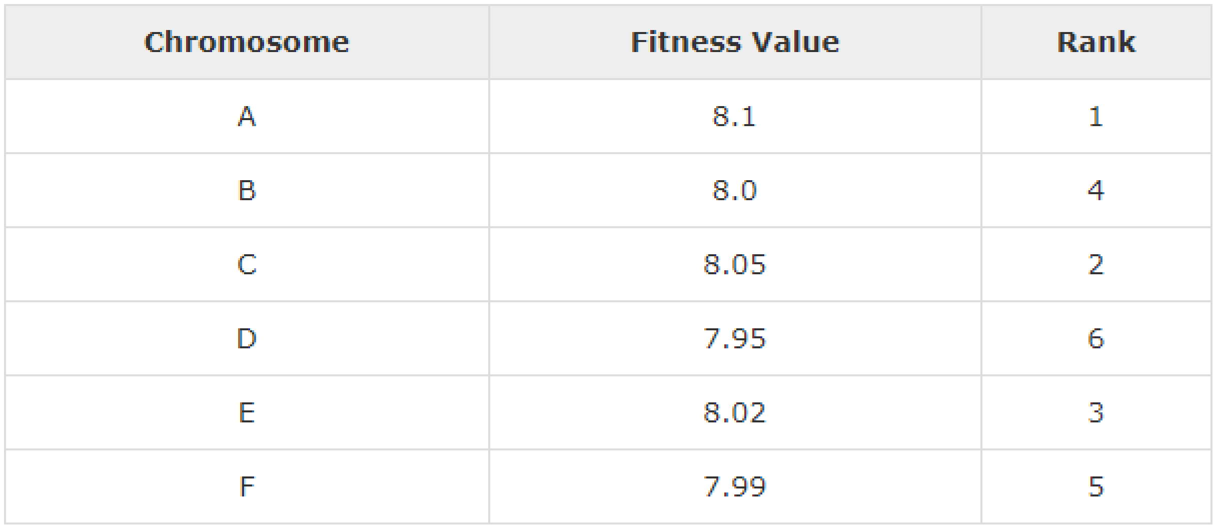


Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

NO SELECTION PRESSURE

Contd..

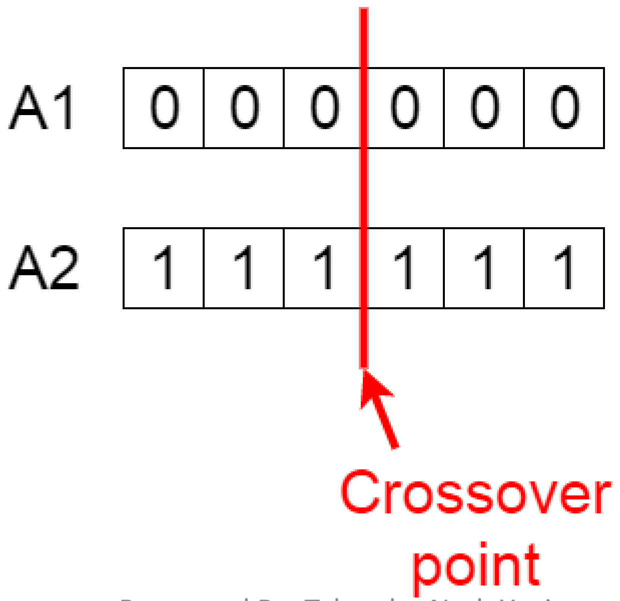
- **Rank Selection:** every individual in the population is ranked according to their fitness. The higher ranked individuals are preferred more than the lower ranked ones.



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

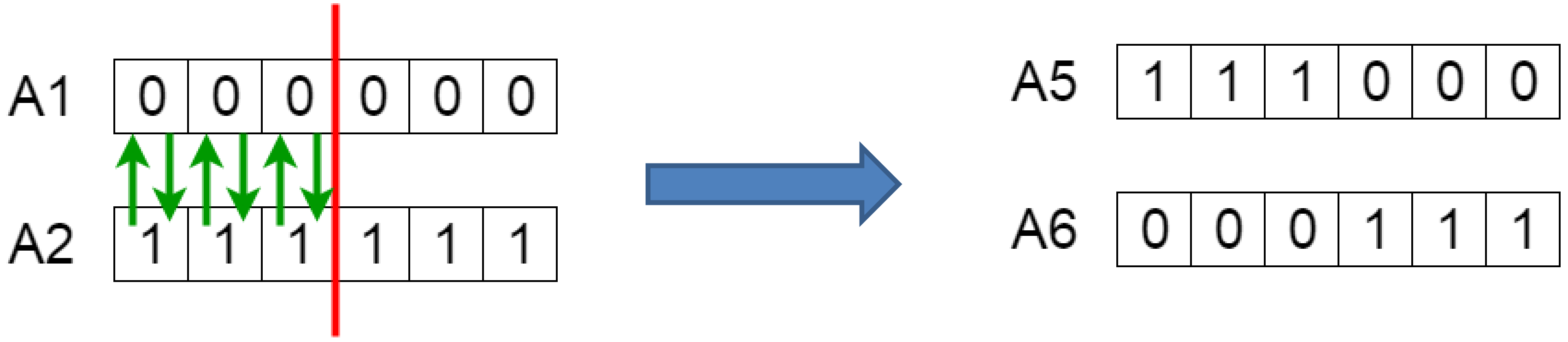
Crossover

- Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.
- For example, consider the crossover point to be 3 as shown below.



Contd..

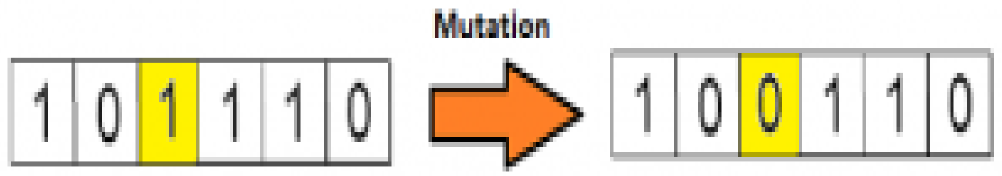
- Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.



- The new offspring are added to the population

Mutation

- Now if you think in the biological sense, are the children produced have the same traits as their parents? The answer is NO. During their growth, there is some change in the genes of children which makes them different from its parents.
- This process is known as mutation, which may be defined as a random tweak in the chromosome.
- This implies that some of the bits in the bit string can be flipped.



- Mutation occurs to maintain diversity within the population and prevent from the premature convergence.

Termination

- The algorithm terminates:
  - If the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.
  - OR If the predefined number of generation is reached.

Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria

Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for “noisy” environments
- Always an answer; answer gets better with time
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use

When to Use a GA

- Alternate solutions are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements

Thank You !