unit-4

Genetic Algorithm

* Genetic algorithms provide a learning method motivated by an analogy to biological evolutions.
* GA's generate successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses.
* A collection of hypotheses called the current population is updated by replacing some fraction of the population by offsprins of the most fit current hypotheses.
* This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offsprins of the next generation

* It widely used in different real-world applications for example image processing, designing electronic circuits, code-breaking, and artificial creativity.

* There are five phases in Genetic Algorithm:-

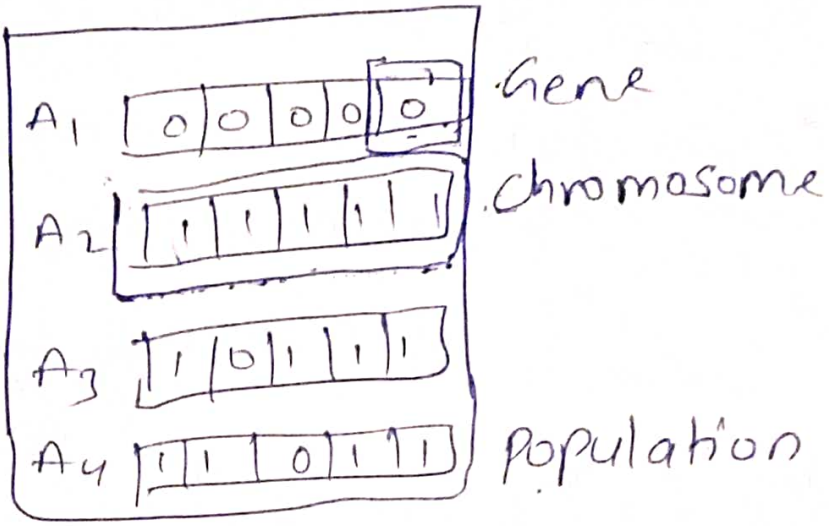
- Initialization.
- Fitness Assignment
- Selection
- crossover (Reproduction)
- Termination.

① Initial population:-

* The process begins with a set of individuals which is called a population.

* Each individual is a ~~show~~ solution to the problem you want to solve known as chromosome.

- An individual is characterized by a set of parameters (variables) known as Genes.
- Genes are joined into a string to form a chromosome (solution).



Fitness function:-

- The fitness function determines how fit an individual is?
- It gives a fitness score to each individual.
- The probability that an individual will be selected for reproduction is based on its fitness score.

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.

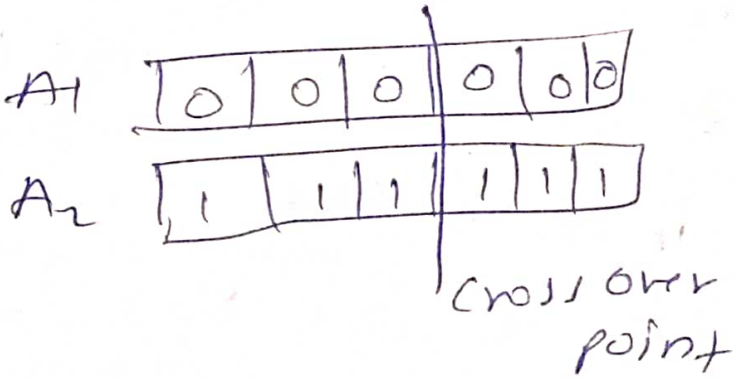
* individuals with high fitness have more chance to be selected for reproduction.

There are three types of selection methods available, which are:-

1. Roulette wheel Selection.
2. Tournament selection.
3. Ranked - based selection.

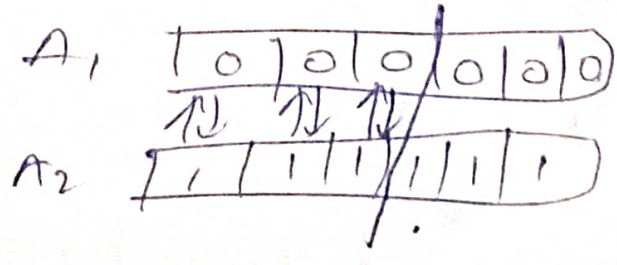
Crossover

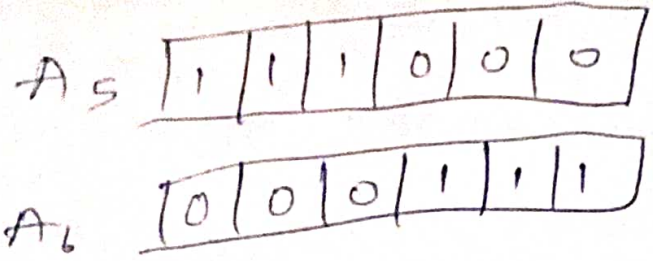
- Crossover is the most significant phase in a genetic algorithm.
- For each pair of parents to be mated, a crossover point is chosen at random from within the genes.
- For example, consider the crossover point to be 3 as shown.



Offspring

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





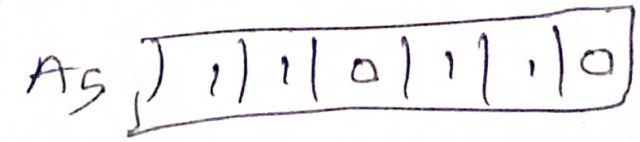
- The new offsprins are added to the population

Mutation

- In certain new offsprins formed some of their genes can be subjected to a mutation with a low random probability.
- This implies that some of the bits in the bit strins can be flipped

Before mutation

After mutation



* Types of mutation styles available,

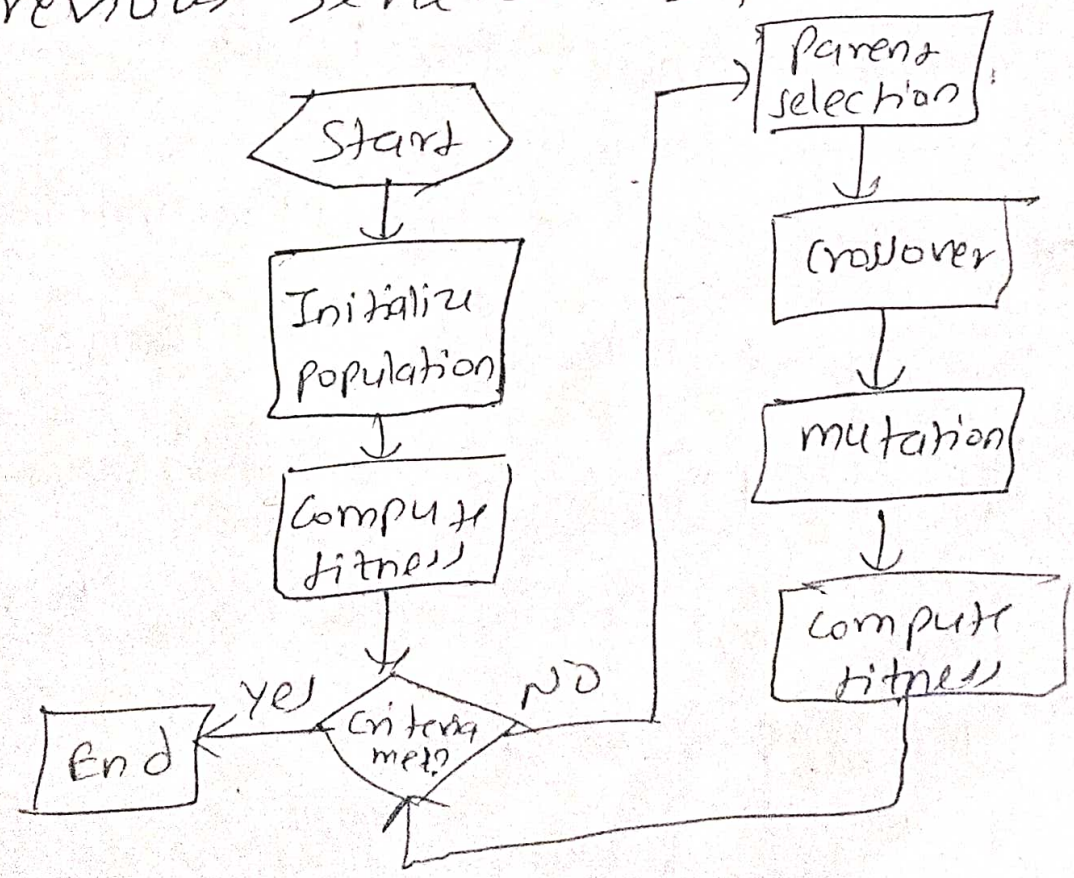
① flip bit mutation

② Gaussian mutation

③ Exchange / swap mutation

Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation).



Example

- Consider the function of maximizing the function

$$f(x) = x^2$$

- where x is permitted to vary between 0 to 31

* Select Encoding Technique

- The minimum value is 0 and maximum value is 31

* Using a five-bit binary integer, numbers between 0(00000) and 31 (11111) can be obtained.

* The objective function here is $f(x) = x^2$, which is to be maximized.

Select Initial Population

- Here initial population of size 4 is chosen, but any number of populations can be selected based on the requirement and application.

* Table shows an initial population randomly selected.

Strings no	Initial population	x value	fitness $f(x)=x^2$	Prob	% Prob	Expected count
1	01100	12	144	0.1247	12.47	0.4987
2	11001	25	625	0.5411	54.11	2.1645
3	00101	5	25	0.0216	2.16	0.0866
4	10011	19	361	0.3126	31.26	1.2502
sum			1155	1.0	100	4
Average			288.75	0.25	25	1
Maximum			625	0.5411	54.11	2.1645

Actual count
1
2
0
1
4

$$\text{Prob} = \frac{f(x)}{\Sigma f(x)} \qquad \text{Expected Count} = \frac{f(x_i)}{\text{Avg}(\Sigma f(x))}$$

Actual count = round near value

0 not selected further

1 will be selected one time

2 will be selected two time.

binary to Dec.

Strins no	matingpool	Crossover Point	offsprins after crossover	Xvalue	Fitnes $f(x)=x^2$
1	01101	4	01101	13	169
2	11001		11000	24	576
3	11011	2	11011	27	729
4	10011		10001	17	289
Sum					1763
Avg					440.75
max					729

original → 625

after 1st step/gen → 729

which is better

String no	Offsprins after crossover	mutation Chromosome for flipping	offsprins after mutation	Xvalue	fitness $f(x)=x^2$
1	01101	10000	11101	29	841
2	11000	00000	11000	24	576
3	11011	00000	11011	27	729
4	10001	00101	10100	20	400
Sum					2546
Avg					636.5
Max					841

original → 625
after crossover → 729
after mutation → 841 → which is better

Hypothesis Space Search:-

* GA's employ a randomized beam search method to seek a maximally fit hypothesis. This search is quite different from that of other learning methods.
* The GA search can move much more quickly, replacing a parent hypothesis by an offsprins that may be radically different from the parent.
* Crowding is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of the individual and very similar individuals take over a large fraction of the population

* The negative impact of crowding is that it reduces the diversity of the population, there by slowing further progress by the GA.

* Strategies have been explored for reducing crowding. One approach is to alter the selection function,

* A related strategy is "fitness sharing" in which the measured fitness of an individual is reduced by the presence of other, similar individuals in the population,

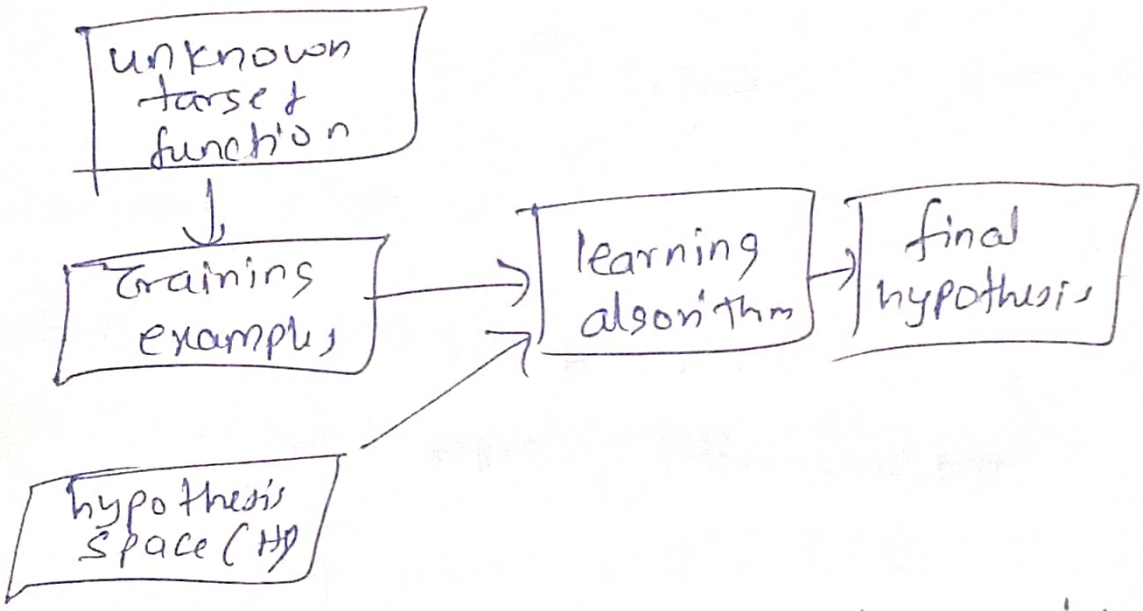
Hypothesis Space Search

* Hypothesis means explanation for something.

* A good hypothesis used to make predictions about new observations.

* Hypothesis Space is the set of all possible legal hypotheses.

* The main goal is to find a possible hypothesis space which can possibly map the inputs to proper outputs.



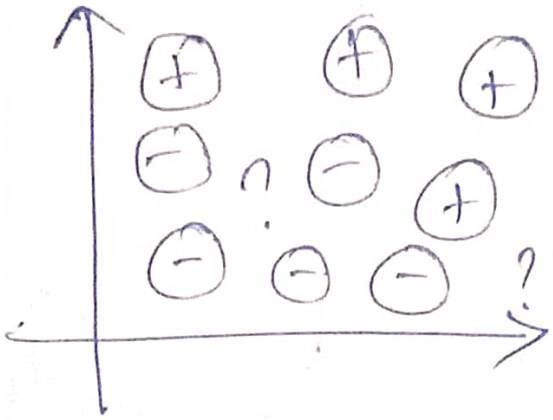
Hypothesis Space :- set of all possible legal Hypothesis Hence it is known as Hypothesis set.

* It used by supervised Machine learning algorithm to determine target function.

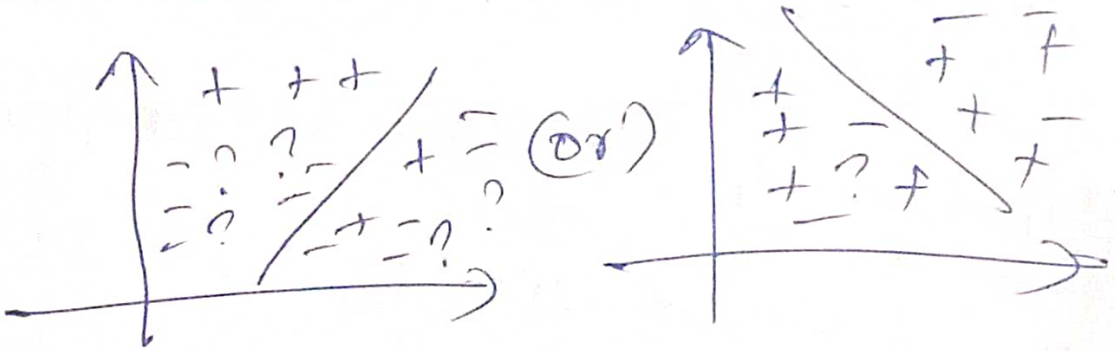
Hypothesis :- It is a function that describes the target of supervised ML.

→ It depends on data, restrictions, and bias which we imposed on data.

→ Ex:- consider a coordinate plan which have some data



* we can predict the data by dividing the coordinates



* The division of alsorithm depends on the data, alsorithm.
* Each individual way to divide is known as Hypothesis.

Genetic Programming

* It is Extension of genetic algorithm.
* Main-idea is to represent a computer program as tree.
* used when an exact solution is not known in advance.
* It uses ideas of biological Evolution for computing problems.

Representing Programs

* Programs manupulated by a Genetic Programming are typically represented by trees corresponding to parse tree of program.
* Each function is represented by nodes in a tree.
* For the function $\sin(x) + \sqrt{x^2+y}$ the representation can be done by defining the primitive function $(\sin, +, \sqrt{}, *)$ and terminals $(x, y, 2)$

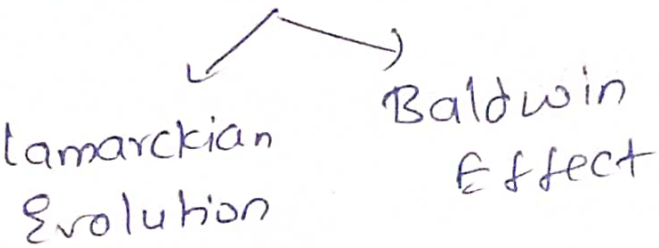
$\sin(x) + \frac{1}{x^2+y}$

crossover operation applied to these parent programs would be like as shown

Models of evolution and learning

Based on a question what is the relationship between learning during the lifetime of a single individual and the skills obtained by a species

there are 2 models



Lamarckian evolution

* He belived that evolution of an organism is influnced by the experience of an individual during lifetime.
* for ex:- if an organism have learnt to avoid some toxic food.
* The organism could pass on this knowledge to its offsprings so that offspring need not learn from test and fail.

* It would allow more efficient Evolutionary progress than GA & GP.

* But this theory is not accepted because the genetics of an individual is unaffected by the lifetime experience.

Story

Lamarck was a scientist who, in the late nineteenth century, proposed that experiences of single organism directly affected the genetic makeup of their offsprings : If an individual learned during its lifetime to avoid some toxic food, it could pass this trait on genetically to its offsprings, which therefore would not need to learn the trait. This is an attractive

conjecture, because it would presumably allow for more effective evolutionary progress than a generate-and-test process (like GAs and GPs) that ignores the experience gained during an individual's lifetime.

Baldwin Effect:-

Baldwin explained about the learning behaviour i.e, if a species is evolving in a changing environment, there will be an evolutionary pressure on the individuals to learn new skills and adopt to the evolution. He considered two things

1. Genotype → genetic code
   → global search.
2. phenotype → characteristics (Behaviour)
   → local search.

* measure cost of learning not in terms of money but in terms of time and energy.

Parallelizing genetic algorithms

It uses multiple algorithms to solve a single task

There are 2 - categories

Fine Grained Coarse Grained.

Fine Grained :-

detailed description which deals with much smaller component.

Coarse Grained :-

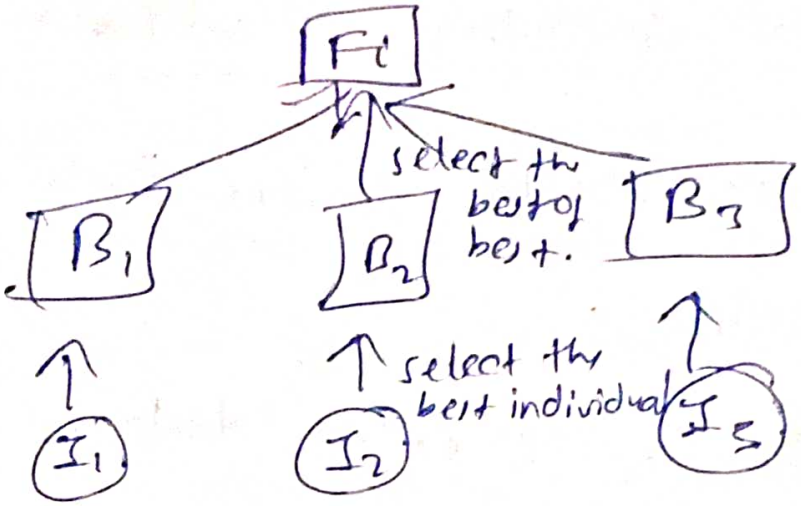
divides into fewer components

Size of component is more than that of fine grained.

→ all the algorithm solve the same task and once they

obtain solution the best one it selected.

* This approach reduces the crowding problem



Learning Set of rules :-

* most reliable way of representing the learned hypothesis is by using a set of product rules

* 1 way - first learn decision tree and translate that tree into rules, one rule for each leaf node.

* 2 way - use a genetic algorithm that encodes each rule as a bit string and uses genetic search operators to [illegible] space.

2 types of algorithms
- First order rules
- Sequential Covering algorithm.

Sequential Covering Algorithm:-

→ It sequentially Covers Each and Every rule.

→ Extracts rules from dataset directly

→ It is mainly based on one of the Evalution measure
- accuracy High
- converge low.

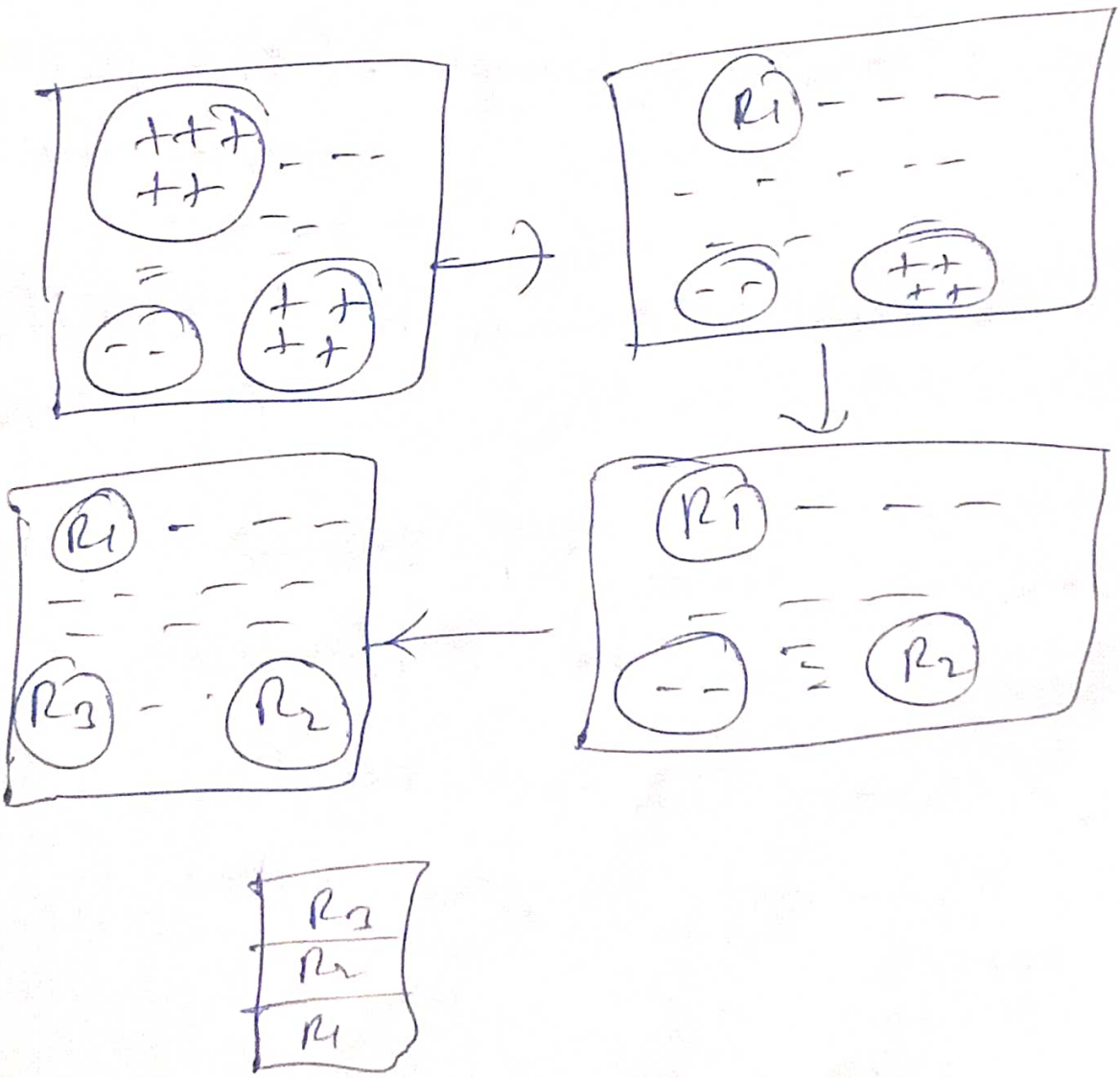
Algorithm

* Create an Empty dataset of decisiontree(R).
* A function called 'learn oneRule' is used
  it expands the best rule for class y
* If all training records of class y = '+'

If all train records of class y are

* Get desirable values (only +ve)
* Eliminate records (i.e the desirable ones)
* New rules is added to the bottom of R.

Ex:- dataset with + and -ve



Sequential Covering (target_attribute, Attributes, Examples, Threshold)

- learned_rules ← { }
- Rule ← learn-one-Rule (target_Attr, Attr, Examples)
- while performance (Rule, Examples) > threshold, do
  - learned_rules ← learned rules
  - Examples ← Examples − [Examples correctly classified by rule]
  - Rule ← learn-one rule (target Attr, Attr, Examples)
- learned_rules ← Sort learned rules according to Performance
- return learned_rules

Learning first order Rules

→ It contain variable.

→ first order → Horn clause :-

Ex:

* Task is to learn simple target concept

Daughter $(x, y) \rightarrow x$ is a daughter of y

→ True if yes

False otherwise.

* Each training example consists of description of two people.

$\langle name_1 = Shoba, mother_1 = Bob, father_1 = Bob,$

$male = False, female = True \rangle$

$\langle name_2 = Vanu, mother_2 = sita, father_2 = Rama,$

$male = ~~false~~ True, female = False \rangle$

If we collect the no of training examples for the target concept Daughter$_{1,2}$ and provide them to a Propositional rule learner, the result would be

If $(father_1 = Bob) \wedge (name_2 = Bob) \wedge$

$(female_1 = True) \Rightarrow Daughter_{1,2} = True$

Now, the problem here is that the propositional representations offer no general way to describe Essential relations so we use variable.

- In ~~fact~~ first order representation it could learn following general rule.

If father(x,y) ∧ female(y)
then Daughter(x,y)

Here x,y are variable that can be bound to any persons.

- First order ~~the~~ Horn clauses may also refer to the variables in the preconditions that do not occur in post conditions.

Eg:- If father(y,z) ∧ mother(z,x) ∧ female(y) then Granddaughter (x,y)

z is present in precondition not in post condition.

Predicates

Predicates takes a value true or false

→ use lower cases

→ represent varables

Functions

function can take any constant as their value

→ use captials

→ Represent Constant.

Learning Sets of first-order Rules:

FOIL (first-order Inductive learning)

* FOIL is a program / Algorithm which is very similar to sequential converting and learn one Rule algorithm.

* used to learn Horn clauses.

* The FOIL program in the natural extension of these earlier algorithms to first-order representa-

* The specific method for generating candidate literals and definition of foil - have am given in the text algorithm can be modified to better accomidate noisy data.

FOIL (target_predicate, predicates, examples)

- pos ← those examples for which the target_ Attribute is TRUE
- neg ← those examples for which the target_predicate is false
- learned rules ← { }
- while pos, do
  - learn new Rule
    - new rule ← the rule that predicts target_predicate with no pre condition
    - new rule neg ← neg
    - while newRule is neg, do

Add a new literal to specialize
new Rule

- Candidate-literals ← Generate
  Candidate new literals for
  new rule, besides Predicate
- Best_literal ← argmax FOIL_
  Gain(L, NewRule)
- add Best_literal to preconditions
  of new Rule.
- NewRule Neg ← subset of new
  Rules that satisfy new Rule
  Neg
- learned_rules ← learned_rules +
  NewRule.
- Pos ← Pos - {members of Pos
  by new Rule}

return learned Rule.

→ It is extension of Sequential covering.

→ Similarity → It will learn 1 rule at a time and removes the +ve samples

→ difference → It will try to learn that the rule predict when the target literal is true.

Foil searches its hyp by using 2 nested loops

1. Outer loop: disjunctive (v)
2. Inner loop: conjuction (n) adding a rule (from most general to more specific)

Performance of foil Algs

$$FOIL\text{-}Gain(L,R) = t\left(\log_2 \frac{P_1}{P_1+n_1} - \log_2 \frac{P_0}{P_0+n_0}\right)$$

L = candidate literal that is to be added to R

P_0 = no. of the +ve bindings of R

n_0 = no of -ve binding to R

P_1 = no of +ve bindings to (R+L)

n_1 = no of -ve bindings to (R+L)

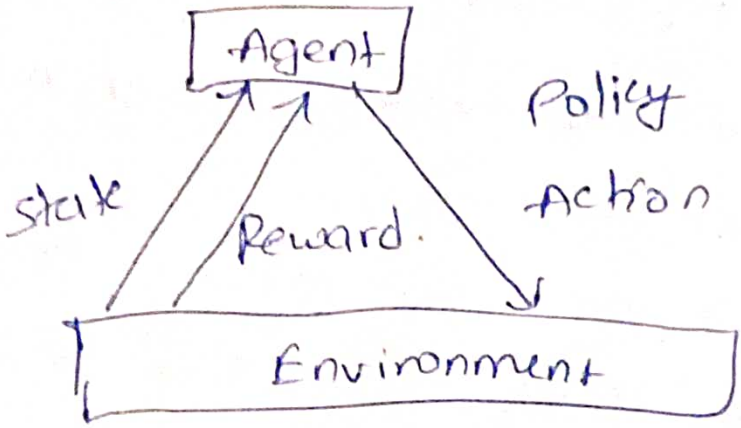
t = no of the bindings to R also covered by (R+L)

Reinforcement learning:-

* It is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.
* It is an area of ML. It is about taking suitable action to maximize record in a particular situation.
* Reinforcement learning emphasizes learning feedback that evaluate the learners perform without providing standards of correct in the form of behavioral learning.
  Ex:- Bicycle learning.
* Trial and Error is an ~~Info~~ important feature of Reinforcement learning.

* An agent can improve its performance by using feedback from Environment is called record signal.

* The Agent may also get a penalty as a negative reward.



1. Agent() - Entity that can explore the Environment & get upon it.

2. Environment : External condition.

3. Action :- That taken by an agent within Environment

4. State :- Situation retained by the Environment after each action taken by the agent.

5. policy :- Defines agent behaviour at an given time.

1) Robotics, Business Planning.
2) machine learning.

Elements of reinforcement learning

Reward signal:-

* Reward is a numerical value which Environment sends back to the agent Every time when it Performs an action.
* The goal of RL agent is to maximize the total Reward iterations in long ~~term~~ run.

Value function:-

* It specifies what is the good in long run.
* The value of a state is the total amount of reward an agent can Expect to accumulate over the future, Starting from the present

Model of Environment :-

* used for planning.
* Make predictions how the Environment will behave.

Q-Learning :-

* Q+ is a basic form of Reinforcement learning which uses Q-values to improve the behaviour of the learning agent.
* Q in Q-learning refers to quality.
* Q+ is the best action to take the given current state.
* q-learning seeks to learn a policy that maximize the total reward.

Terms required for Q-learning.

1. Policy: $\Phi(s_t, a_t)$

* It is the combination of state and action at the time.
* policy is undertaken by agent.
* agent observes the given state and selects best possible action.

Ex:- Empty Road - Speed driving.
Crowdy Road - slow driving.

2. Reward :- Scalar quantity

Example :- Maze Game
correct action → reward.

3. penality :- commonly called as -ve reward.
wrong action.
Example :- Parking in no parking zone.

* Q-learning mainly depends on 2 factors :-

1 - Q function.
2 - Q. table.

Q- function :-

$$Q(s_t, a_t) = E(R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n R_{t+n} / s_t, a_t)$$

↓

Bellman Equation

R - Reward.

γ - discount function.

Q → is the evaluation function the agent will learn.

Q- table :-

Combination of actions and state

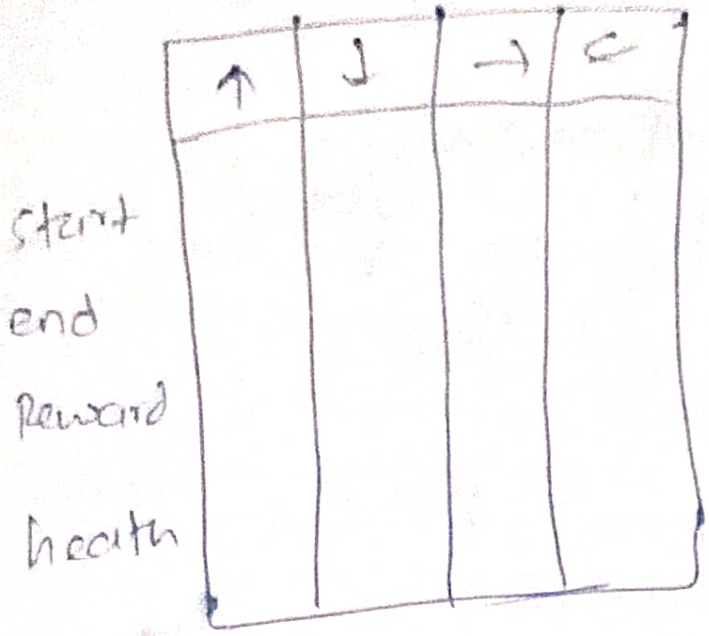
Ex :- In a game

Action :- up, down, left, Right.

State :- start, end, reward, health etc

Steps :-

1. Exploration :- Explore all possible path
2. Exploitation :- best possible path is identified.



Initially all values = 0

Initialize q table

Choose Action

Perform Action

measure Action

update Action.

q-learning Algorithm

q learning algorithm

For each s, a initialize the table entry $\hat{q}(s,a)$ to zero.

Observe the current state s.

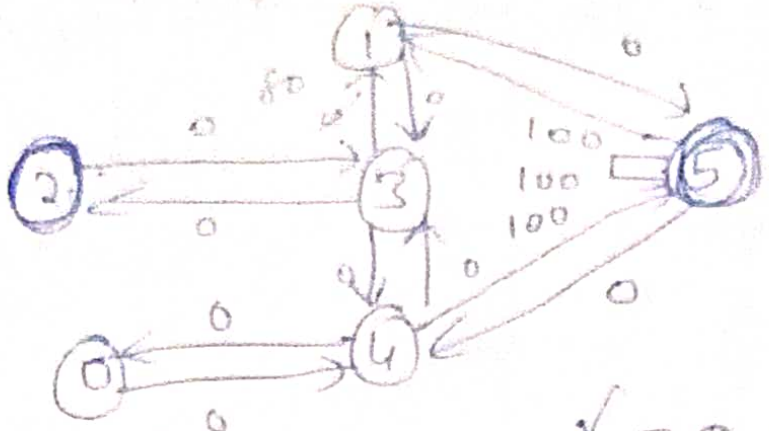
Do forever:

- select an action a and execute it.
- Receive immediate reward r
- observe the new state s'
- update the table entry for $\hat{q}(s,a)$ as follows:

$$\hat{q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{q}(s',a')$$

- $s \leftarrow s'$

Ex:-



$\gamma = 0.8$ Assume

current state

$s = 3$

$a = 2, (1), 4$ → action.

$r = 0$

$s' = 1$

$\hat{\varphi}(3,1) = 0 + 0.8 \max(0, 100) = 80$

$\hat{\varphi}(3,1) = 80$

current state, action

$s = 3, 1$

$a = 2, (1), 4, 3, (5)$

$r = 0, 100$

$s' = 1, 5$

$\hat{\varphi}(1,5) = 100 + 0.8 \max(0,0) = 100$

$s = 3, 1, 5$

The steps must be continued unless the all the zero are removed and change the values.