UNIT- IV

1) Genetic Algorithm :-

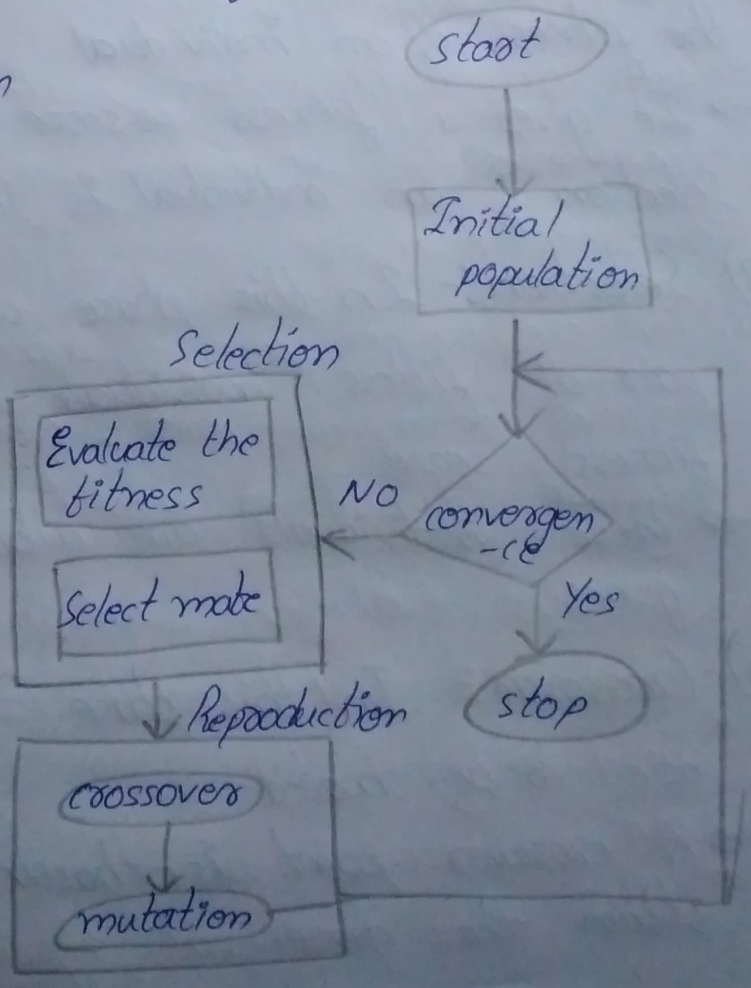
→ Genetic Algorithm is an adaptive heuristic search algorithm which is based on the process of natural selection.

→ It is used to generate high-quality solutions for optimization and search problems.

→ GA is based on the theory of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

→ There are total 5 phases of a GA :-

1) Initial population
2) fitness function
3) Selection
4) Crossover
5) Mutation

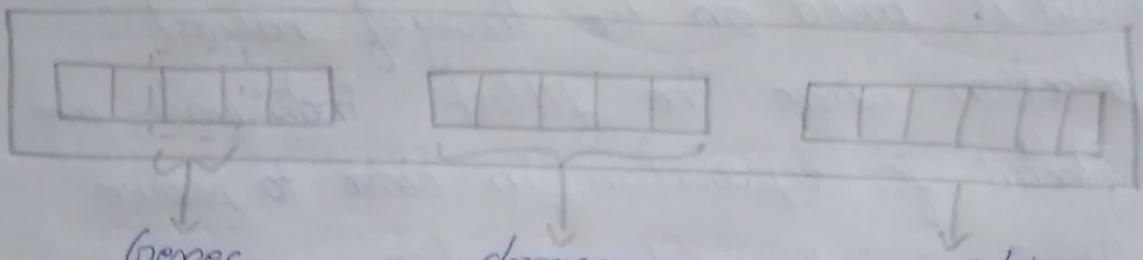


1) Initial population :-

→ The process of a GA begins with a set of individuals called as population.

→ Each individual of this population is a solution to the problem which we are solving.

→ Individual is characterized by a set of parameter -s (variable) called as genes. genes are joined to form a string known as chromosome.



2) Fitness function :- It is used to calculate the fitness of an individual.

→ It gives a fitness score to each individual. Selection of an individual is based on this score.

3) Selection :- In this phase we select Two pair of fittest individuals based on their fitness score.

→ These individuals are allowed to pass their genes.

4) Crossover :- In this phase the selected pairs are to be mated.

→ A crossover point is chosen at random from within the genes

→ Offsprings are generated by exchanging the genes of the parents among themselves till the crossover point is reached.

→ After this phase new ~~population~~ offsprings are added to the population.

Ex:- P_1 :- | 0 | 0 | 0 | 0 | 0 | 0 |

P_2 :- | 1 | 1 | 1 | 1 | 1 | 1 |

→ crossover point

Offsprings:-

O_1 :- | 1 | 1 | 1 | 0 | 0 | 0 |

O_2 :- | 0 | 0 | 0 | 1 | 1 | 1 |

5) Mutation:- In some new offsprings formed, some of their genes can be mutated with a low random probability.

→ After mutation some of the bits in the bit string can be flipped.

Before Mutation:-

| 1 | 1 | 1 | 0 | 0 | 0 |

After mutation:-

| 1 | 1 | 0 | 1 | 1 | 0 |

Termination:- The algorithm terminates if the population has converged (i.e, new offsprings produced are not different from ~~their~~ previous generation).

→ At this stage the algorithm is said to have produced a solution to our problem.

2) Motivation :- GA provide a learning method motivated by an analogy of biological evolution.

→ Their are many factors which motivate the usage of genetic algorithm.

(i) GA involves using fitness of a string to perform the search operation. So, algorithm does not need any knowledge about the problem to perform search operation.

(ii) GA can perform parallel search operation on numerous points of the problem

(iii) GA can operate well on problems which have gaps, jumps and noise.

(iv) GA are easily perform parallel operation and take advantage of decreasing costs of powerful computer hardware.

(v) GA can search "spaces of hypothesis" containing complex interacting parts, where overall hypothesis may be difficult to model.

(vi) GA uses evolution which is known to be a successful & robust method for adaptation with in a biological systems.

(vii) GA can produce an optimal solution much faster than other traditional algorithms.

3) Genetic Algorithm :- (Explanation in introduction)

Advantages :-

→ Does not require any derivative information
→ It is faster and more efficient as compared to traditional methods.
→ Has efficient parallel operations performing capabilities.
→ provides a list of good solution
→ useful when search space is large and more parameters are involved.

Limitations :-

→ GA are not suited for all problems.
→ repeated calculation of fitness value is expensive.
→ no ~~guarant~~ guaranty on optimality or quality of solution
→ If not implemented properly, GA will not give a optimal solution.

Example of Genetic Algorithm :-

i) Selection :- Here two pairs of individuals with high fitness score are selected

→ fitness score is decided by using fitness function.

Fitness function :- The fitness function is based on classification accuracy over the training data.

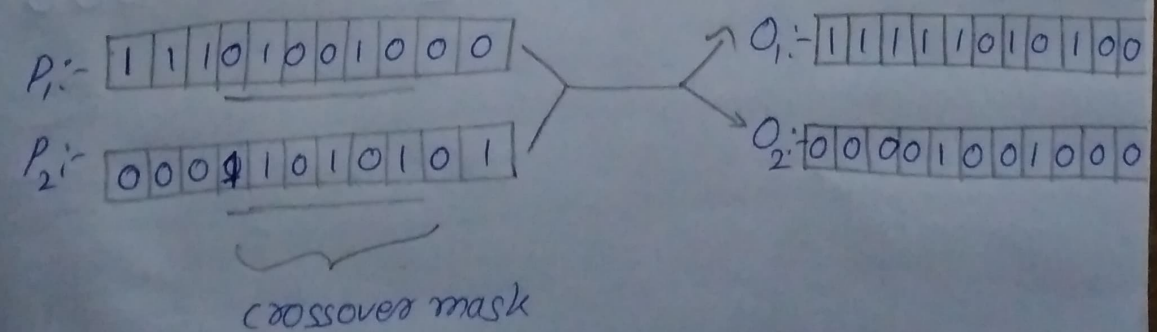
$$fitness(h) = (correct(h))^2$$

→ correct(h) is the percent of all training examples correctly classified by hypothesis h.

ii) Crossover :- This stage creates 2 new offsprings from 2 parent strings by copying selected bits from each parent.

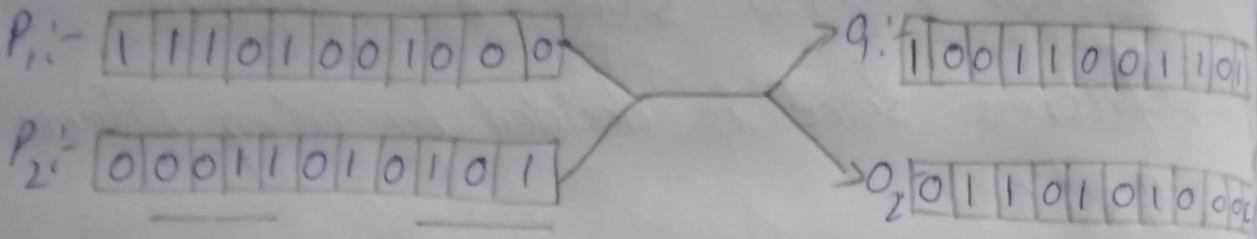
→ A string of n-bits which will be exchanged is called crossover mask

→ Single point cross over :- In single point crossover a string of n contiguous 1's and 0's is selected for crossover.



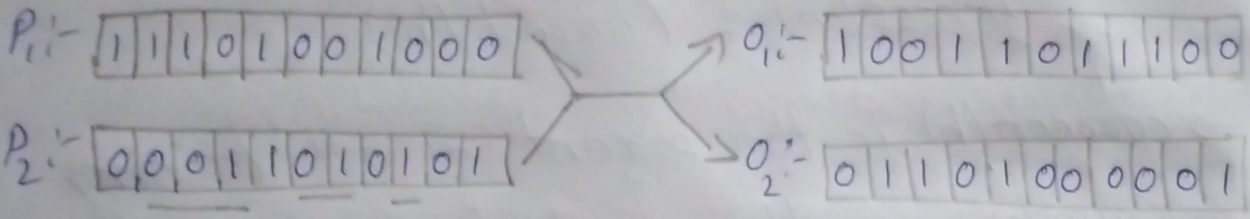
Two-point crossover:- In this type of crossover their are more than 1 crossover masks.

Ex:-



uniform - crossover:- In this type of crossover point is taken randomly.

Ex:-



(iii) Mutation:- It produces small random changes to the bit string by choosing a single bit at random and changing its value

→ It is often applied after crossover has been performed.

Ex:-

1 1 1 0 1 0 0 1 0 0 0 → 1 0 1 0 1 0 0 1 0 0 0

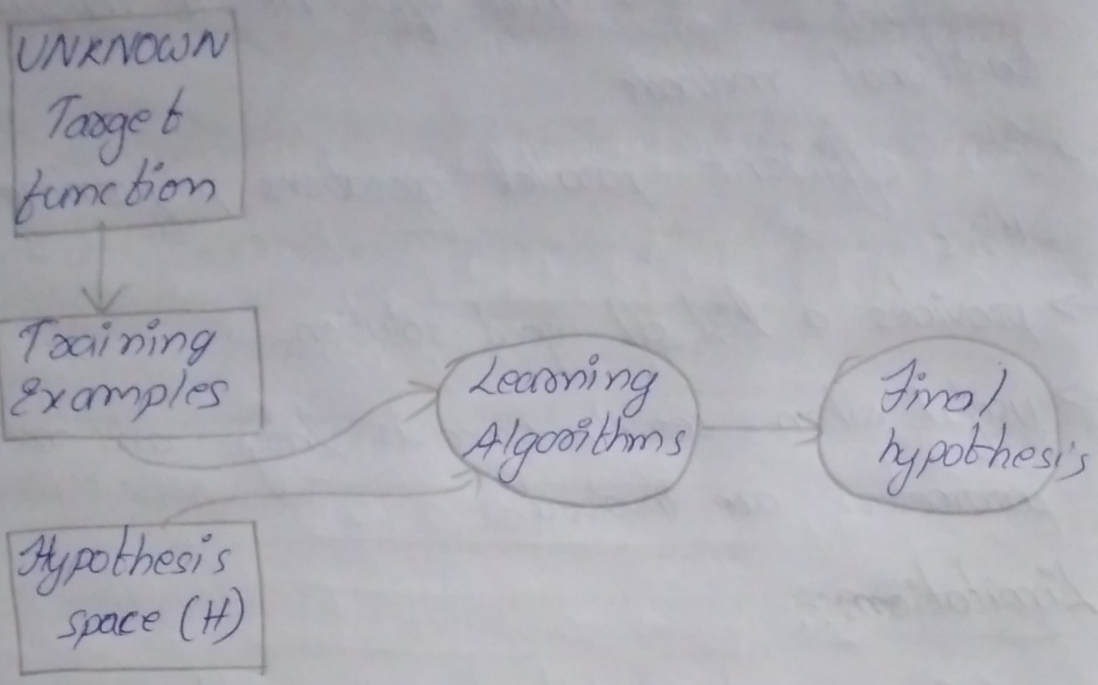
4) Hypothesis space search :-

→ hypothesis can be defined as an explanation for something.

→ A good hypothe fits the evidences and can be used to make predictions about new observations or new situations.

→ Hypothesis space is the set of all possible legal hypothesis.

→ In most supervised ML algorithms the main goal to find out a possible hypothesis from the hypothesis space which can possibly map the inputs to proper outputs.



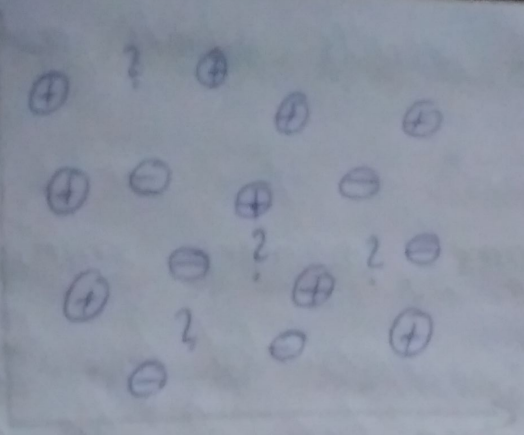
Hypothesis space (H):- From this set the ML algorithm determine the best possible o/p ed (or) hypothesis which would best describe the target function or the o/ps.

Hypothesis (h):- It is a function that describes the target in Supervised ML.

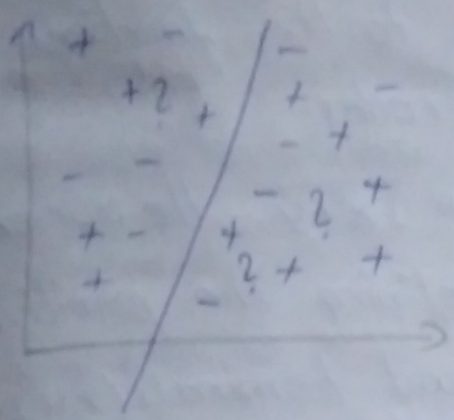
→ The hypothesis given by an algorithm depends on data, restrictions and bias which we imposed on data.

→ For example we have test data for which we have to determine the o/p or results.

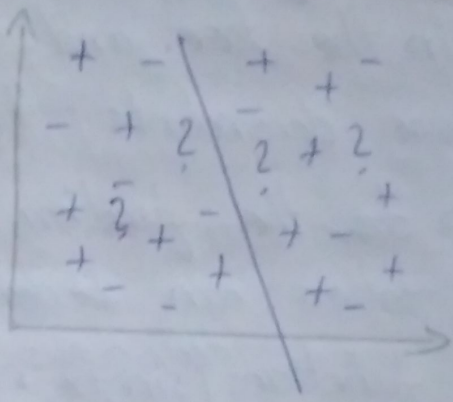
→ Here we are considering a coordinate plain which have some data.



→ we can predict the data by dividing the coordinates as shown below.

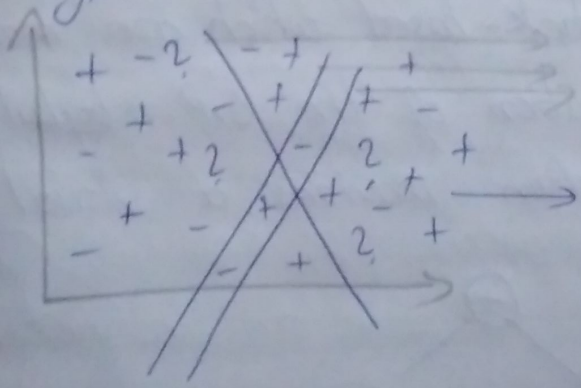


(or)



→ The division of the coordinate plane may depend on data, algorithm and constraints.

→ Each individual possible way to divide is known as the hypothesis



4) Genetic Programming :- GP is an ~~autom~~ method to evolve computer programs

→ It uses ideas of biological evolution to handle complex problems.

→ It is an extension of genetic algorithm, it is used for testing and selecting the best choice

among a set of results which are represent -ed by a string.

Representing programs :- programs manipulated by a GP are typically represented by trees corresponding to the parse tree of the program.

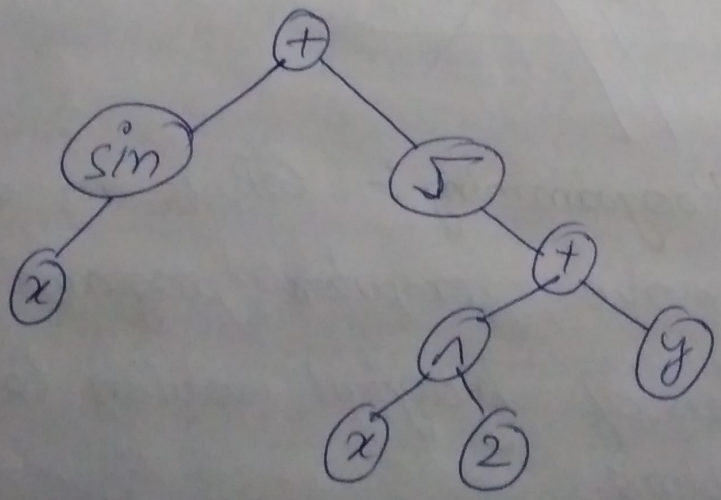
→ Each function call is represented by a node in the tree and argument to the functions are decendents to these nodes.

→ For the function $\sin(x) + \sqrt{x^2+y}$ the represen -tation can be done by defining the primitive functions (sin, +, *, -, /, -) and terminals (x, y, 2).

→ Tree-based GP was the first application of GP.

→ There other types to represent such as linear, cartesian, & stack-based which are more efficient in their execution of the GP. bybut tree-based representation provides a visual representa -tion.

Ex :-



5) Models of Evolution and learning:-

→ model of Evolution & learning are based on a question that "what is the relationship between learning during the life time of a single individual and the skills attained by a species".

→ This question is answered by two scientist.

1) Lamarckian Evolution
2) Baldwin Effect.

1) Lamarckian Evolution:- He proposed that evolution of an organism is influenced by the experience of an individual during his life time.

→ According to him if an organism have learnt to avoid some toxic food.

→ Then the organism could pass on this knowledge genetically to its offsprings. So, that the offspring would not need to learn it again by test and trail.

→ According to this theory it would allow more efficient evolutionary progress than a generate & test process (like GA and GP).

→ But this theory is not accepted because the germ genetics of an individual is unaffected by the lifetime experience of its biological parents.

Baldwin Effect:-

→ According to Baldwin if a species is evolving in a changing environment, there will be an evolutionary pressure on the individuals to ~~ad~~ learn new skills and adapt to the evolution.

→ For example if a new predator appears in the environment then the individuals capable of avoiding the predator will be more sucessfull.

→ Those individuals who are able to learn many skills will depend less strongly on their genetic skills.

→ As a result these individuals can support a more diverse genetic, by relying on individual learing to overcome the gaps in skills provided by genes.

→ Thus a behaviour that was once learned may eventually become an instinctive (spontan-eous) action.

6) Parallelizing Genetic Algorithm:-

→ GA is naturally suited to parallel implementation and there are many approaches to parallelization.

→ "Coarse grain" approach to parallelization subdivide the population into some distinct group of individuals called as demes.

→ Each deme is assigned to a different computational node & a standard GA is performed at each node.

→ Transfer of individuals between demes occurs by a migration process. in which individuals from one deme are transferred to other deme.

→ This process modeled after a process-fertilization b/w physically seperated subpopulations. of a biological species.

→ The approach reduces the crowding problem in nonparallel GAs. where sys fails into a local optimum (local model) due to appearance of a similar genotype which comes to present in the entire population.

7) Introduction of Learning sets of Rules:-

→ Most reliable way of representing the learned hypothesis is by using a set of product rules (if-then rules).

→ In many cases it is useful to learn the target function representation as a set of if-then rules which jointly define the whole function.

→ One of the way to learn these set of rules is to first learn a decision tree and then translate the tree into an equivalent set of rules, where one rule represents a particular node in the tree.

→ Second method is to use a genetic algorithm that converts each rule to a bit-string and uses genetic search operators to explore the hypothesis space.

→ So, rules can be learnt either by using representations or by using learning directly.

→ Important aspect of direct rule learning algorithms is that they can learn set of first-order rules which have more representation-al power that than propositional rules derived from decision trees.

8) Sequential Covering Algorithms

→ A covering algorith, in the context of a propositional learning system, is an algorithm thwhich develops a cover for the set of positive examples.

→ i.e, It developes a cover for a set of all possible hypotheses that account for all the positive examples and excludes the -ve examples.

→ This alg is called as a sequential covering alg because it learn one rule at a time and request repeat this process to cover the full set of positive examples.

→ This algorithm is based on the strategy of "Learning one rule" and remove the data it covers.

Algorithm:-

The algorithm - given a set of examples.

1. Start with an empty Cover.
2. Using Learn-one-Rule to find the best hypothesis
3. If the Just-Learnt-Rule satisfies the threshold than
   - Put Just-Learnt-Rule to to cover
   - Remove examples covered by Just-Learnt-Rule

- Go to step-2.

4. Sort the cover according to its performance over examples.

5. Return: Cover.

Ex:-

An example of set of propositional examples in this sys is as follows.

Properties:-

1) One character of this algorithm is it requires high accuracy, i.e, the prediction ~~should~~ will be correct with high probability.

2) It may possibly give low coverage, i.e, it → may not ~~cover~~ make prediction for all examples. Their can be some examples which cannot be classified by the algorithm.

9) Learning Rule sets: Summary.

I^st dimension:-

→ A sequential covering algorithm like CN_2 learns one rule at a time and removes the covered example and repeats the process.

→ ~~Simultaneous covering~~

→ Decision tree algorithms such as ID_3 learn the entire set of disjuncts simultaneously.

→ CN2 chooses among alternative attribute-value pairs by comparing the subsets of data they cover.

→ ID3 chooses attributes by comparing the partitions of the data they generate.

→ Thus, CN2 makes a larger no. of independent choices. So it is better if there is plenty of data.

II nd dimension :-

→ In this dimension the approaches vary in terms of search.

→ In "LEARN - ONE - RULE" the search is from general to specific hypotheses.

→ In EG Find-S searches from specific to general.

→ In general to specific search there is a single maximally general hypothesis. ~~fr~~om which we begin the search.

→ In specific to general search there are many maximally specific hypothese, so it is unclear which to select as starting point.

→ To address this we choose several positive examples at~~e~~ random to initialize and to guide the search.

→ The best hypothesis obtained through

III 2nd Dimension:-

→ This dimension is about wether the "LEARN-ONE-RULE" search is a "generate then test" search or it is a example-

→ "Learn-One-Rule" is a a "general-then-test" search and it could be "example-driven"

→ In "Generate-then-test" each choice in the search is based on the hypothesis performance over many examples.

→ so, the impact of imperfect data is minimi-zed.

→ In example-driven individual training examples constraint the generation of hypothesis.

→ In example-driven search imperfect data can have a large impact.

8) Learn-One-Rule algorithm (continuation)

Algorithm:-

LEARN-ONE-RULE (Target_attribute, Attributes, Examples, K)

// Returns a single rule that covers some of the examples, conducts a general to specific ~~gee~~ greedy beam search for the ~~bet~~ best rule, guided by the PERFORMANCE metric //

- Initilize Best_hypothesis to the most general hypothesis ∅.
- Initilize Candidate_hypothesis to the set {Best_hypothesis ∅}
- While Candidate_hypothesis is not empty, DO.

1. Generate the next more specific candidate_hypothesis.

New_candidate_hypothesis ← new generated and specialized candidates

2. Update Best_hypothesis.

Best hypothesis ← h with best PERFORMANCE.

3. Update Candidate_hypothesis.

candidate_hypothesis ← the k best members of New_Candidate_hypotheses.

- Return a rule of the form.

"If Best_hypothesis THEN prediction"

where prediction is the most frequent value of Target_attribute among those.

Examples that match Best_hypothesis.

10) Learning First-Order rules:-

→ First order logic allows the expression of propositions and their truth functional

Propositional logic:- propositional logic allows the expression of individual propositions and their truth functional combinations.

* Eg:- propositions like "Tom is a men" or "All men are mortal" may be represented by single proposition letters such as P or Q.

* Truth functional combinations are built by using connectives, such as $\wedge, \vee, \neg, \rightarrow$,-
Ex:- $P \wedge Q$.

* Inference rules can also be defined over propositional forms.

Ex:-
$$\frac{P \rightarrow Q \quad P}{Q}$$

First Order logic:- First order logic allows the expression of propositions and their truth functional combinations.

→ It also allows us to represent propositions as assertions of predicates about individuals or sets of individuals.

* E.g:- propositions like "Tom is a man" or "All men are mortal" may be represented by predicate-argument representation such as

man(tom) or $\forall x$ (man(x) $\rightarrow$ mortal(x))

* Inference rules permit conclusions to be drawn about set / Individuals - e.g:- mortal(tom)

→ First order logic is ~~xxx~~ much more expressive than propositional logic because it allows ~~greater~~ better specification and reasoning when representing knowledge.

→ First order learners can generalise over relational concepts and can also acquire recursiv-e rules.

E.g:-

IF Parents (x, y) THEN Ancestor (x, y)

IF Parent (x, z) $\wedge$ Ancestor (z, y)

THEN Ancestor (x, y).

Terminology in First-Order Logic:-

→ All expressions in first-order logic are composed of the following attributes:-

1. Constants :- tyler, 23, a
2. variables :- A, B, C
3. Predicate symbols :- male, father (True or False

values only).

4. Function symbols - age (can take on any constant as a value)

5. Connectives - $\wedge, \vee, \neg, \rightarrow, \leftarrow$

6. quantifiers - $\forall, \exists$

Term :- It can be defined as any constant, variable or function applied to any term.

E.g:- age (bod), 23, A

Literals :- It can be defined as any predicate or negated predicate applied to any term.

1) Ground Literal :- a literal that contains no variables. e.g:- female (getha)

2) Positive Literal :- a literal that does not contain a negated predicate.

E.g:- female (getha)

3) Negative Literal :- a literal that contain a negated predicate.

E.g:- $\neg$ father (x, y)

clause :- It can be defined as any disjunction of literals whose variables are universally quantified.

E.g:- $M_1 \vee - - - - \vee M_n$

→ Here $M_1, M_2, \ldots M_n$ are literals with variables universally quantified along with a disjunction operator.

Horn clause :- It can be defined as any clause containing exactly one positive literals.

E.g.:- $H \vee \neg L_1 \vee \ldots \vee \neg L_n$

⇒ Since.

$$\neg L_1 \vee \neg L_2 \vee \neg L_3 \vee \ldots \vee \neg L_n = \neg (L_1 \wedge \ldots \wedge L_n)$$

→ According to inference rules.

$$(A \vee \neg B) \equiv (A \leftarrow B)$$

$$\Rightarrow (H \vee \neg (L_1 \wedge L_2 \wedge \ldots \wedge L_n)) \equiv$$

$$\Rightarrow \quad H \leftarrow (L_1 \wedge \ldots \wedge L_n).$$

II) Learning sets of First-Order rules ; FOIL :-

→ First Order Inductive Learning is a rule-based learning algorithm.

→ It is a natural extension of Sequential-Covering and Learn-One-Rule algorithms. it follows a greedy approach.

→ Like Sequential-Covering, FOIL also learns one rule at a time and removes +ve examples covered by the learned rule before attempting

to learn a further rule.

→ Unlike Sequential-Covering & Learn-One-Rule, FOIL

(i) only tries to learn rules that predict when the target literal is true.

(ii) performs a a simple hill-climbing search.

→ FOIL searches its hypothesis space via two nested loops:

(i) The outer loop at each :- It adds a new rule to the disjunctive hypothesis at each iteration.

→ It uses specific-to-general search & starts with an empty disjunctive hypothesis that does not contain any +ve instances at initial stage and stops when the hypothesis covers all +ve examples.

(ii) Inner loop:- It add a conjuctive constraint to the rule precondition on each iteration.

→ It uses general-to specific search & starts with the most general precondition (empty) and stops when the hypothesis excludes all negative examples.

Algorithm:-

FOIL (Target_predicate, Predicate, Examples)

- POS ← positive Examples
- Neg ← negative Examples
- Learned_rules ← { }
- while Pos, do

Learn a New Rule

NewRule ← most general rule possible

NewRuleNeg ← Neg

while NewRuleNeg, do

Add a new literal to specialize NewRule

1. Candidate_literals ← generate candidates based on predicates
2. Best_literals.

$$\underset{L \in}{\operatorname{argmax}} \text{ Candidate\_literals } FOIL\_Gain(L, NewRule)$$

3. Add Best_literal to NewRule preconditions.
4. NewRuleNeg ← subset of NewRuleNeg that satisfies NewRule preconditions.

- Learned rules ← Learned_rules + NewRule
- Pos ← Pos − {members of pos covered by Newrule}.

- Return Learned_rules.

Disadvantages of FOIL:-

1) Requires large extensional background definitions
2) Requires complete examples to learn recursive

definations.

3. Requires a large set of closed-world negatives
4. Inability to handle logical functions.
5. Hill-climbing search gets stuck at local optima.

12) Induction as inverted deduction:-

→ A second approach for inductive logical programming is based on the observation that induction is the inverse of deduction.

→ In fact, induction is the inverse operation of deduction & cannot be convinced to exist without corresponding operation. So, the

→ So, the question of relative importance b/w Induction & deduction cannot arise. They are similar to addition & substraction which are inverse of one another.

→ Induction is finding the hypothesis h such that

$$(\forall \langle x_i, f(x_i) \rangle \in D)\ B \wedge h \wedge x_i \vdash f(x_i)$$

where

- D is training data
- x_i is i^{th} training instance.
- $f(x_i)$ is the target function value for x_i.
- B is other background knowledge

→ The equation means, for some given data "D" and some partial background knowledge "B", learning can be described as generating a hypothesis "h" that, together with "B", explains "D".

→ If the training data is a set of examples of the form $\langle x_i, f(x_i) \rangle$ where x_i denotes the i^{th} training example and $f(x_i)$ denotes its target value.

13) Inverting Resolution:-

→ The resolution rule is a sound and complete rule for deductive inference in first-order logic.

→ Let L be an arbitary propositional literal & P & and $P \vee R$ be arbitary propositional clauses.

→ The resolution rule is

$$\frac{P \vee L \qquad \neg L \vee R}{P \vee R}$$

→ In the given two assertions $P \vee L$ & $P \neg L \vee R$ it is obvious that either L or $\neg L$ must be false.

→ Therefore either P or R must be true. Thus the conclusion P ∨ R of the resolution rule is satisfied.

→ This operation is used in cigol.

→ Let $C = A \vee B$ & $C_2 = B \vee D$

→ Any literal present in C but not in C_1 mus be present in C_2.

→ $C_2 = A \vee \neg D$ (or) $C_2 = A \vee \neg D \vee B$.

→ The literal that occurs in C_1 but not in C must be removed by the resolution rule and its negation must occur in C_2.

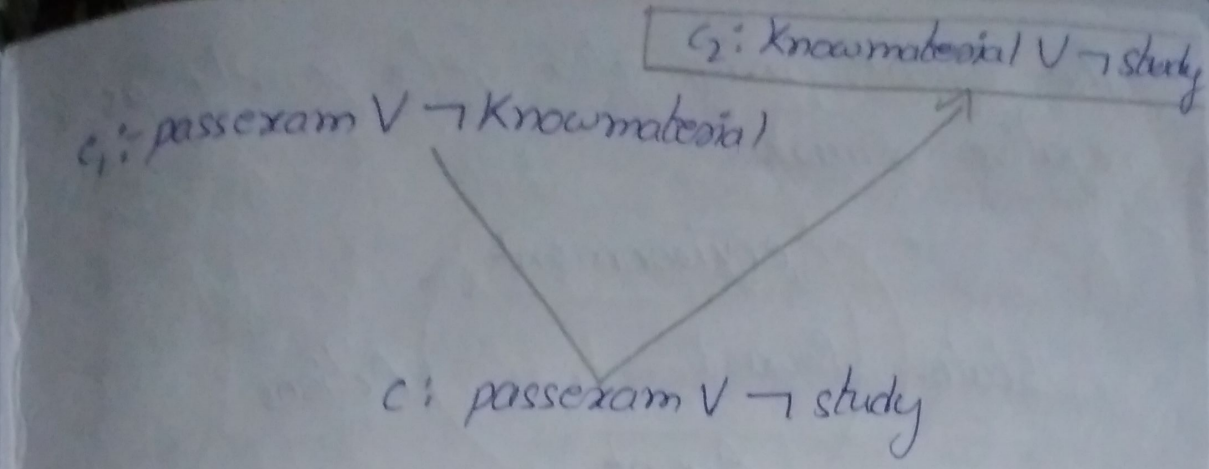
→ Cigol uses inverse resolution with Sequential covering but with first order representation.

Ex:-

C_2: Know material $\vee \neg$ study

C_1: PassExam $\vee \neg$ Know material

C: passExam $\vee \neg$ study



14) Reinforce Learning:- It address how an autonomous agent which senses and acts in its environment can learn to choose optimal actions to achieve its goals.

→ In supervised learning we get immediate feedback, in unsupervised we don't get a feed-back. But in Reinforced learning we get delayed scalar feedback.

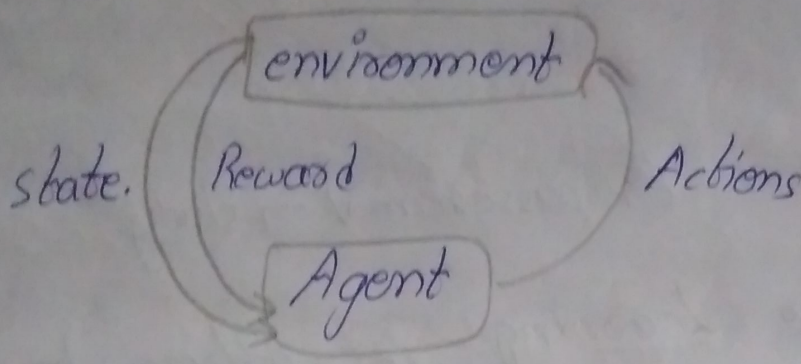
→ Reinforcement learning:- itis learning what to do and how to map situations to actions.

→ Reinforced learning:- it deals with agents that must sense and act upon their environment. It combines classical Artifical intelligence and machine learning techniques.

→ Trail & Error is an important feature of Reinforcement learning.

→ An agent can improve its performance by using feedback from environment. This feedback from environment is called as reward signal.

-> The agent may also get a penalty as a negative remark.



Terms used in Reinforcement Learning.

1) Agent ():- entity that can explore the environment and act upon it.

2) Environment ():- The situation in which an agent is present (or) surroundings of an agent.

3) Action ():- actions taken by an agent within the environment.

4) State ():- It is a situation returned by the environment after each action taken by the agent.

5) Reward ():- A feedback from environment to agent, to evaluate the actions of agent.

6) Policy ():- strategy applied by the agent for next actions based on current state.

7) value ():- It is expected long-term returned with discount factor & it is opposite to short-term reward.

8) Q-value ():- Similar to value, but takes an additional parameter current action (a).

Elements of Reinforcement Learning:-

1) Policy:- It defines the learning agent behaviour for given time period.

→ It is a mapping from perceived (recogniz-ed) states of the environment to actions to be taken when in those states.

2) Reward function:- It is used to determine a goal in reinforcement learning problems.

→ It also maps perceived state of the environment to a single number.

3) value function:- It specifies what is good in the 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 state.

4) Model of the environment:- Model are used for planning

→ with help of model, one can make predections about how the environment will behave.

→ If a state & action are given, then a model can predict the next state and reward.

5) Q-learning:- It is a basic form of Reinforcement Learning which uses Q-values to iteratively improve the behaviour of the learning agent.

→ "Q" in Q-learning stands for quality which represents how useful a particular action is for gaining some future rewards.

→ Q-learning is an off policy reinforcement learning algorithm which seeks to find a best action to take in at the given current state.

→ It is considered off-policy because the q-learning function learns from the actions that are present in the current policy.

→ q-learning seeks to learn a policy that maximizes the total reward.

Q-values (or) Action-values:- Q-values are defined for states and actions. Q(S,A) is an estimation of how good is it to take the action A at the state S.

$$S \xrightarrow{a} \text{(best action)}$$

(given state)

→ Agent only knows about possible states and actions. agent does not know about rewards and transitions between states.

→ So, the agent has to actively learn through what are good and bad actions by trial & error.

→ A Q-table is created while performing the Q-learning. It helps us to find the best action for each state. ~~It to~~ It maximizes the expected reward by selecting the best of all possible actions.

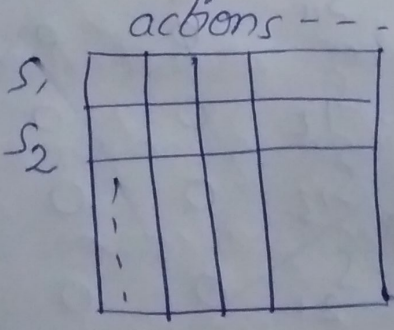
→ In a Q-table.

(i) Nom. of rows = no of states.

(ii) No of colums = no of possible actions.

→ Q-table is initalised with zero values.

→ Q(S, A) => Q (state, action) returns the expected future reward of that function on that state.



→ when Q-table is redy, the agent will start to exploit the environment & start taking better actions.

Q-Learning Algorithm:-

⇒ For each state action pair (S,a) initialise the table entry zero.

observe the current state s.

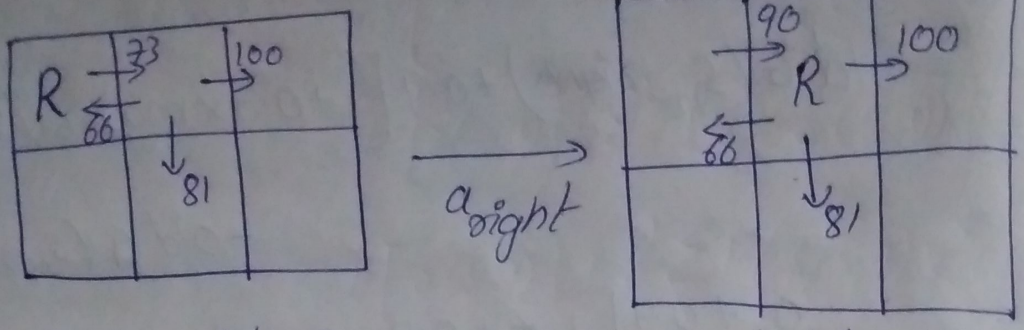
Do forever:

- 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

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

- $s \leftarrow s'$

Ex:-



Initial state: S_1 Next state: S_2

$$\hat{Q}(S_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(S_2, a')$$

Let reward $r=0$ & $\gamma = 0.9$

$$\hat{Q}(S_1, a_{right}) \leftarrow 0 + 0.9 \times \max[66, 81, 100]$$

$$\leftarrow 0 + 0.9 \times [100]$$

$$\leftarrow 90$$

16) Non-deterministic rewards & actions:-

→ In Q-learning we considered ~~the~~ studied if in deterministic environment.

→ when we consider Q-learning in non-deter-ministic case the reward function $r(S,a)$ and action transition function $\delta(S,a)$ may lead to probabilistic outcomes.

→ For example, in robot problems with noisy sensors & effectors it is often appropriate to model actions and rewards as non-deterministic.

→ In such cases the functions $\delta(s,a)$ and $r(s,a)$ can be viewed as first producing a probability distribution over the outcomes based on s and a. and then choosing an outcome at random according to this distribution

→ when this probability distribution depend solely on a and s, then we call the system a non-deterministic markov decision process.

→ In non-deterministic case, we must redefine ~~change~~ the objective of the learner to take into account ~~that~~ the fact that the outcomes of actions are no longer deterministic.

→ we should also ~~generalize~~ redefine the value V^{π} of a policy π to be the expected value of the discounted cumulative reward received by applying this policy.

$$V^{\pi}(s_1) = E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right]$$

→ we defined optimal policy π^* to the policy π that maximizes $V^{\pi}(s)$ for all states s.

→ Now we generalize the definition of Q, by taking its expected value.

$$Q(s,a) \equiv E\left[r(s,a) + \gamma V^*(\delta(s,a))\right]$$

$$\equiv E\left[r(s,a)\right] + \gamma E\left[V^*(\delta(s,a))\right]$$

$$\equiv E\left[r(s,a)\right] + \gamma \sum_{s'} P(s'/s,a) V^*(s')$$

17) Temporal Difference Learning :-

→ Temporal - difference (TD) learning is a combination of Monte carlo ideas and dynamic programming ideas.

→ TD method updates the estimates based on other learned ~~examples~~ estimates, without waiting for the final outcome.

→ In Q-learning algorithm we learn by iteratively reducing the ~~discrepency~~ b differences between Q value estimates of adjacent states.

→ So, Q-learning is also a special type of temporal difference algorithms. which learns by reducing the difference b/w estimates made by agents.

→ TD learning is a model-free learning & it has 2 properties:-

(i) don't require model- dynamics to be known in advance.

(ii) can be applied on non-episodic tasks also

→ TD learning uses TD update rule for updating the value of a state:-

$$V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

$V(S_t)$ = value of previous state

α = Learning rate

R_{t+1} = Reward

γ = discount factor

$V(S_{t+1})$ = value of current state

$V(S_t)$ = value of previous state.

18) Relationship to Dynamic Programming:-

→ Reinforcement learning methods like Q-learning are related to the research on dynamic programming approaches for solving Markov decision processes

→ Dynamic programming is a technique available to solve self-learning problems.

→ The term dynamic programming refers to a collection of algorithms which can be used to compute optimal policies for a given model of the environment.

→ In Q-learning we assume that initially the agent doesnot have any knowledge about $\delta(s,a)$ and $r(s,a)$

→ when the agent is exposed to real world. The agent must learn to prepare a acceptable policy.

→ online systems:- The system that learn by moving in the real world environment & observing the results is called as online system

→ offline system:- Systems that learn by considering actions within an internal model is called as offline system.