**Hill Climbing Algorithm in Artificial Intelligence**

* Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
* Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
* It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
* A node of hill climbing algorithm has two components which are state and value.
* Hill Climbing is mostly used when a good heuristic is available.
* In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

**Features of Hill Climbing:**

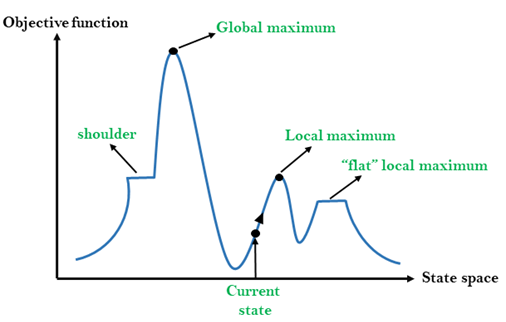
Following are some main features of Hill Climbing Algorithm:

* **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
* **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
* **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

**State-space Diagram for Hill Climbing:**

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.



**Different regions in the state space landscape:**

**Local Maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

**Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

**Current state:** It is a state in a landscape diagram where an agent is currently present.

**Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value.

**Shoulder:** It is a plateau region which has an uphill edge.

AD

**Types of Hill Climbing Algorithm:**

* Simple hill Climbing:
* Steepest-Ascent hill-climbing:
* Stochastic hill Climbing:

1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:

* Less time consuming
* Less optimal solution and the solution is not guaranteed

**Algorithm for Simple Hill Climbing:**

* **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
* **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
* **Step 3:** Select and apply an operator to the current state.
* **Step 4:** Check new state:
  1. If it is goal state, then return success and quit.
  2. Else if it is better than the current state then assign new state as a current state.
  3. Else if not better than the current state, then return to step2.
* **Step 5:** Exit.

2. Steepest-Ascent hill climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

Algorithm for Steepest-Ascent hill climbing:

* **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
* **Step 2:** Loop until a solution is found or the current state does not change.
  1. Let SUCC be a state such that any successor of the current state will be better than it.
  2. For each operator that applies to the current state:
     1. Apply the new operator and generate a new state.
     2. Evaluate the new state.
     3. If it is goal state, then return it and quit, else compare it to the SUCC.
     4. If it is better than SUCC, then set new state as SUCC.
     5. If the SUCC is better than the current state, then set current state to SUCC.
* **Step 5:** Exit.

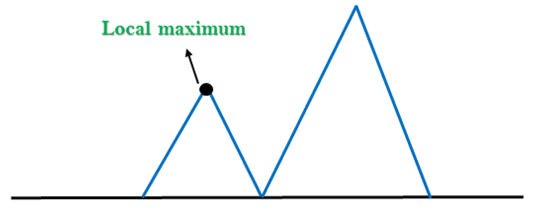
3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Problems in Hill Climbing Algorithm:

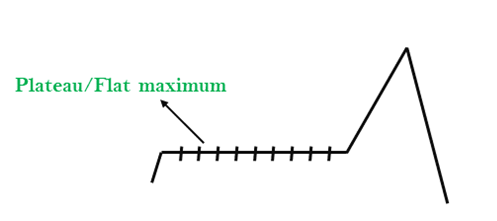
**1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

**Solution:** Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



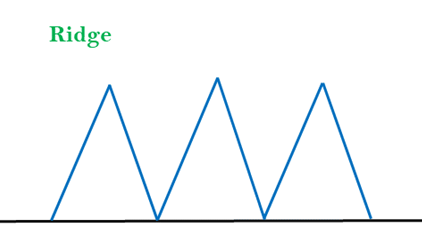
**2. Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

**Solution:** The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



**3. Ridges:** A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

**Solution:** With the use of bidirectional search, or by moving in different directions, we can improve this problem.



Simulated Annealing:

A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum. And if algorithm applies a random walk, by moving a successor, then it may complete but not efficient. **Simulated Annealing** is an algorithm which yields both efficiency and completeness.

In mechanical term **Annealing** is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state. The same process is used in simulated annealing in which the algorithm picks a random move, instead of picking the best move. If the random move improves the state, then it follows the same path. Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path.

**Typical AI assumptions**

Two agents whose actions alternate

Utility values for each agent are the opposite of the other

creates the adversarial situation

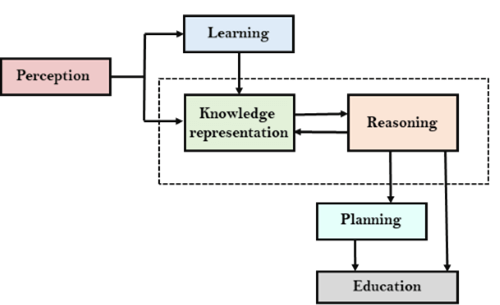
Fully observable environments

In game theory terms: Zero-sum games of perfect information.

**AI knowledge cycle:**

An Artificial intelligence system has the following components for displaying intelligent behavior:

* Perception
* Learning
* Knowledge Representation and Reasoning
* Planning
* Execution



The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

**Example: Write the negation for each of the following. Determine whether the resulting statement is true or false. Assume U = R.**

1.∀ x ∃ m(x2<m)

**Sol:** Negation of ∀ x ∃ m(x2<m) is ∃ x ∀ m (x2≥m). The meaning of ∃ x ∀ m (x2≥m) is that there exists for some x such that x2≥m, for every m. The statement is true as there is some greater x such that x2≥m, for every m.

2. ∃ m∀ x(x2<m)

**Sol:** Negation of ∃ m ∀ x (x2<m) is ∀ m∃x (x2≥m). The meaning of ∀ m∃x (x2≥m) is that for every m, there exists for some x such that x2≥m. The statement is true as for every m, there exists for some greater x such that x2≥m.

# Probabilistic reasoning in Artificial intelligence

## Uncertainty:

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

## Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors
3. Equipment fault
4. Temperature variation
5. Climate change.

## Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

**Need of probabilistic reasoning in AI:**

* When there are unpredictable outcomes.
* When specifications or possibilities of predicates becomes too large to handle.
* When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

* **Bayes' rule**
* **Bayesian Statistics**

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

**Probability:** Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

1. 0 ≤ P(A) ≤ 1,   where P(A) is the probability of an event A.
2. P(A) = 0,  indicates total uncertainty in an event A.
3. P(A) =1, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

Probabilistic reasoning in Artificial intelligence

* P(¬A) = probability of a not happening event.
* P(¬A) + P(A) = 1.

**Event:** Each possible outcome of a variable is called an event.

**Sample space:** The collection of all possible events is called sample space.

**Random variables:** Random variables are used to represent the events and objects in the real world.

**Prior probability:** The prior probability of an event is probability computed before observing new information.

**Posterior Probability:** The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

AD**Conditional probability:**

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

Probabilistic reasoning in Artificial intelligence

**Where P(A⋀B)= Joint probability of a and B**

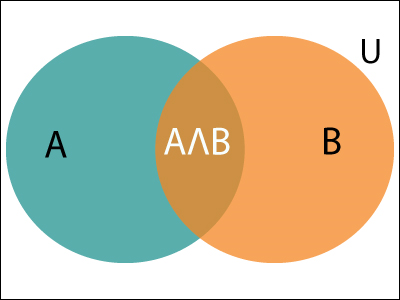
**P(B)= Marginal probability of B.**

If the probability of A is given and we need to find the probability of B, then it will be given as:

Probabilistic reasoning in Artificial intelligence

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A⋀B) by P( B )**.

AD



**Example:**

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

**Solution:**

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

Probabilistic reasoning in Artificial intelligence

**Hence, 57% are the students who like English also like Mathematics.**

Why is NLP important?

**Large volumes of textual data**

Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.

Today’s machines can analyze more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that’s generated every day, from medical records to social media, automation will be critical to fully analyze text and speech data efficiently.

**Structuring a highly unstructured data source**

Human language is astoundingly complex and diverse. We express ourselves in infinite ways, both verbally and in writing. Not only are there hundreds of languages and dialects, but within each language is a unique set of grammar and syntax rules, terms and slang. When we write, we often misspell or abbreviate words, or omit punctuation. When we speak, we have regional accents, and we mumble, stutter and borrow terms from other languages.

While supervised and unsupervised learning, and specifically deep learning, are now widely used for modeling human language, there’s also a need for syntactic and semantic understanding and domain expertise that are not necessarily present in these machine learning approaches. NLP is important because it helps resolve ambiguity in language and adds useful numeric structure to the data for many downstream applications, such as speech recognition or text analytics.

**Why use Decision Trees?**

As a budding data professional, you’ll have plenty of responsibility at your future position, therefore, it’s important to know which techniques are most beneficial to you. There are many advantages to using decision trees that can help you improve your skills and advance in your data science journey, such as:

* **Decision trees are easy to understand.** Because of their structure, which follows the natural flow of human thought, most people will have little trouble interpreting them. In addition, visualizing the model is effortless and allows you to see exactly what decisions are being made.
* **There is little to no need for**[**data preprocessing**](https://365datascience.com/courses/data-cleaning-preprocessing-pandas/)**.** Unlike other algorithms, decision trees take less time to model as they require less coding, analysis, or even [dummy variables](https://365datascience.com/tutorials/statistics-tutorials/dummy-variable/). The reason is that the technique looks at each data point individually instead of the set as a whole.
* **Versatile when it comes to data.** In other words, [standardizing](https://365datascience.com/tutorials/statistics-tutorials/standardization/) the collected data is not a necessity. You can imbue both [numerical and categorical data](https://365datascience.com/tutorials/statistics-tutorials/numerical-categorical-data/) into the model as it’s able to work with features of both types.

All of these make decision trees ideal for [communicating with business stakeholders](https://365datascience.com/career-advice/career-guides/5-business-basics-data-scientists/) as they’ll be able to follow along without any specialized knowledge required.

What Are the Disadvantages of Decision Trees?

Of course, where there are benefits, there are also limitations. This is true even for an intuitive analysis method such as a decision tree. Some of the disadvantages include:

* **There is a tendency to overfit.** Essentially, the model performs so well on the training data that it compromises the decision-making process. You can prevent this by either stopping the decision tree before it has a chance to do so or, alternatively, letting it grow and then pruning the decision tree after [overfitting](https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/) occurs.
* **Mathematical equations are more costly.** Not only does the decision tree require more time to calculate, but it also consumes more memory. This is not ideal as sometimes you will have to work with substantial amounts of data and stricter deadlines – efficiency is of the essence.
* **Decision trees can be unstable.** For example, a minor modification of the data can lead to significant changes – perhaps even generating a new tree with contrary results. Another instance is the model producing biased decisions if some of the classes dominate over the rest.

Don’t be discouraged, however, as these disadvantages can be easily overcome with the right techniques. You just have to be conscious of how you approach them and prepare appropriately.

**Capabilities of Expert Systems**

The expert systems are capable of −

* Advising
* Instructing and assisting human in decision making
* Demonstrating
* Deriving a solution
* Diagnosing
* Explaining
* Interpreting input
* Predicting results
* Justifying the conclusion
* Suggesting alternative options to a problem

They are incapable of −

* Substituting human decision makers
* Possessing human capabilities
* Producing accurate output for inadequate knowledge base
* Refining their own knowledge

**How the graph will traverse to its final stage using DLS method? Draw the steps.**

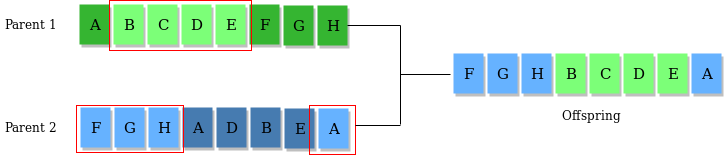
**Programming Without and With AI**

The programming without and with AI is different in following ways −

|  |  |
| --- | --- |
| **Programming Without AI** | **Programming With AI** |
| A computer program without AI can answer the **specific** questions it is meant to solve. | A computer program with AI can answer the **generic** questions it is meant to solve. |
| Modification in the program leads to change in its structure. | AI programs can absorb new modifications by putting highly independent pieces of information together. Hence you can modify even a minute piece of information of program without affecting its structure. |
| Modification is not quick and easy. It may lead to affecting the program adversely. | Quick and Easy program modification. |

**Operators of Genetic Algorithms**

Once the initial generation is created, the algorithm evolves the generation using following operators –   
**1) Selection Operator:** The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.   
**2) Crossover Operator:** This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring). For example –



**3) Mutation Operator:** The key idea is to insert random genes in offspring to maintain the diversity in the population to avoid premature convergence. For example – 



The whole algorithm can be summarized as –

1) Randomly initialize populations p

2) Determine fitness of population

3) Until convergence repeat:

a) Select parents from population

b) Crossover and generate new population

c) Perform mutation on new population

d) Calculate fitness for new population

**Example problem and solution using Genetic Algorithms**

Given a target string, the goal is to produce target string starting from a random string of the same length. In the following implementation, following analogies are made –

* Characters A-Z, a-z, 0-9, and other special symbols are considered as genes
* A string generated by these characters is considered as chromosome/solution/Individual

**Fitness score** is the number of characters which differ from characters in target string at a particular index. So individual having lower fitness value is given more preference.

**Properties of Mini-Max algorithm:**

* **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
* **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
* **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **O(bm)**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
* **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

AD