**Reinforcement Learning**

* Supervised learning is simplest and best-studied type of learning
* Another type of learning tasks is learning behaviors when we don’t have a teacher to
* tell us how
* The agent has a task to perform; it takes some actions in the world; at some later

point gets feedback telling it how well it did on performing task

* The agent performs the same task over and over again
* The agent gets carrots for good behavior and sticks for bad behavior
* t’s called reinforcement learning because the agent gets positive reinforcement for

asks done well and negative reinforcement for tasks done poorly

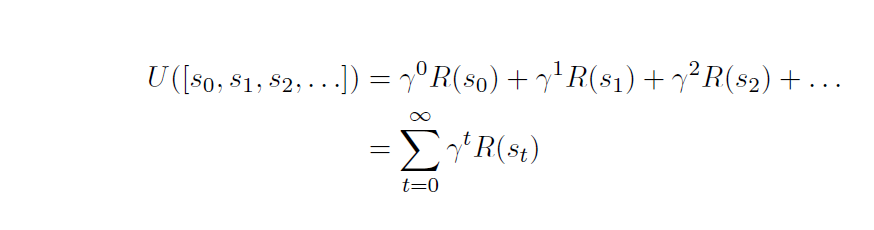
* The problem of getting an agent to act in the world so as to maximize its rewards
* Consider teaching a dog a new trick: you cannot tell it what to do, but you can reward/punish it if it does the right/wrong thing. t has to figure out what it did that made it get the reward/punishment, which is known as the credit assignment problem
* We can use a similar method to train computers to do many tasks, such as playing backgammon or chess, scheduling jobs, and controlling robot limbs

**Markov Decision Process: Policy**

* A policy is a function π: S ->A that specifies what action the agent should take in
* any given state
* Executing a policy can give rise to many action sequences!
* How can we determine the quality of a policy?

Markov Decision Process: Utility

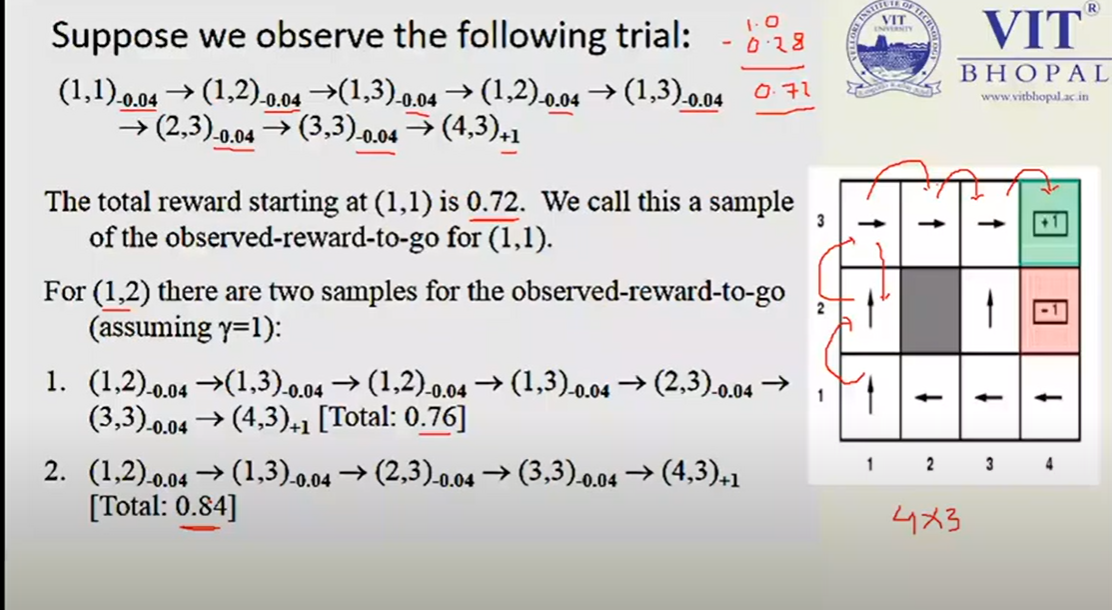
* Utility is an internal measure of an agent’s success
* Agent’s own internal performance measure
* Surrogate for success and happiness
* The utility is a function of the rewards



With gama as discount factor

Passive RL

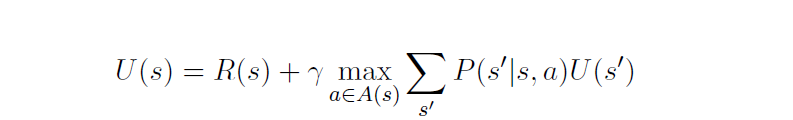
* n passive RL, the agents’ policy π is fixed, it only needs to know how good it is
* Agent runs a number of trials, starting in (1,1) and continuing until it reaches a terminal state
* The utility of a state is the expected total remaining reward (reward-to-go)
* Each trial provides a sample of the reward-to-go for each visited state
* The agent keeps a running average for each state, which will converge to the true value
* This is a direct utility estimation method



The main drawback is that this method makes a wrong assumption that state utilities are independent while in reality they are Markovian. Also, it is slow to converge.

**Problems with Direct Utility Estimation**

* Direct utility fails to exploit the fact that states are dependent as shown in Bellman equations



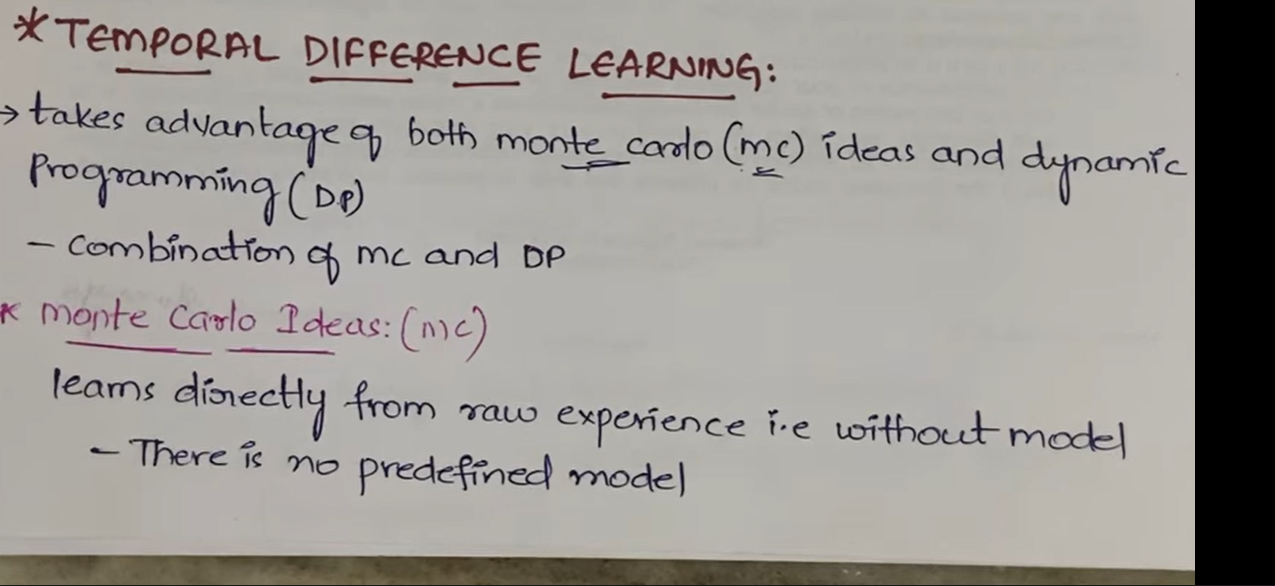
* Learning can be speeded up by using these dependencies
* Direct utility can be seen as inductive learning search in a too large hypothesis space that contains many hypothesis violating Bellman equations

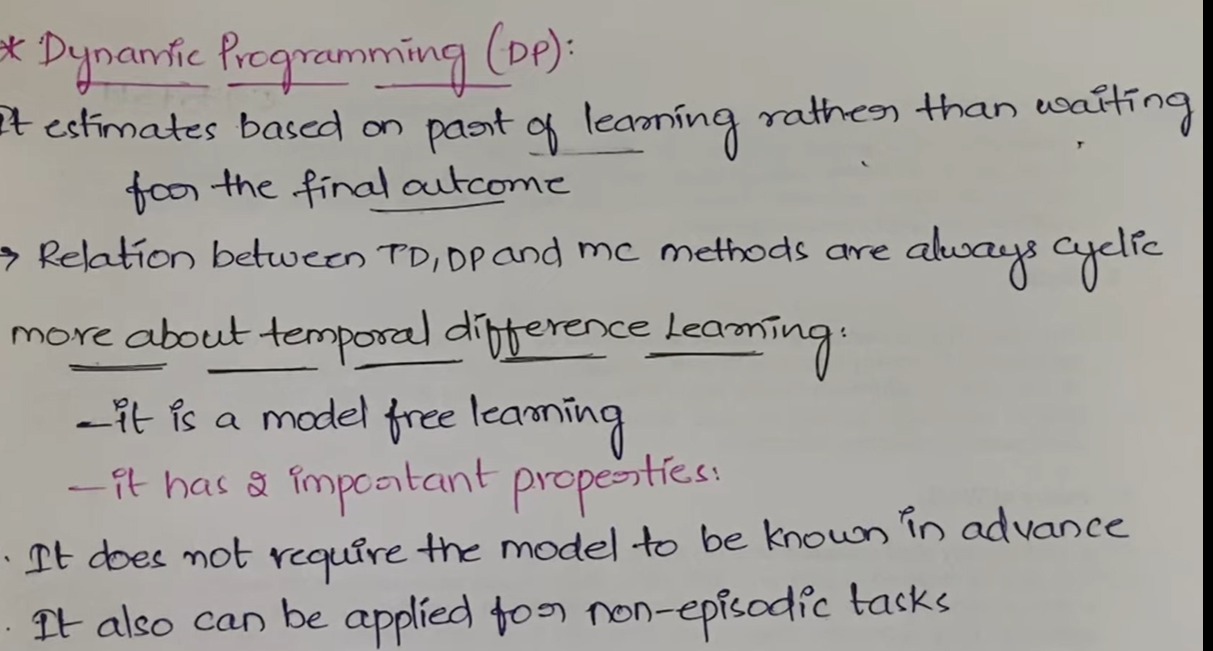
**Adaptive Dynamic Programming**

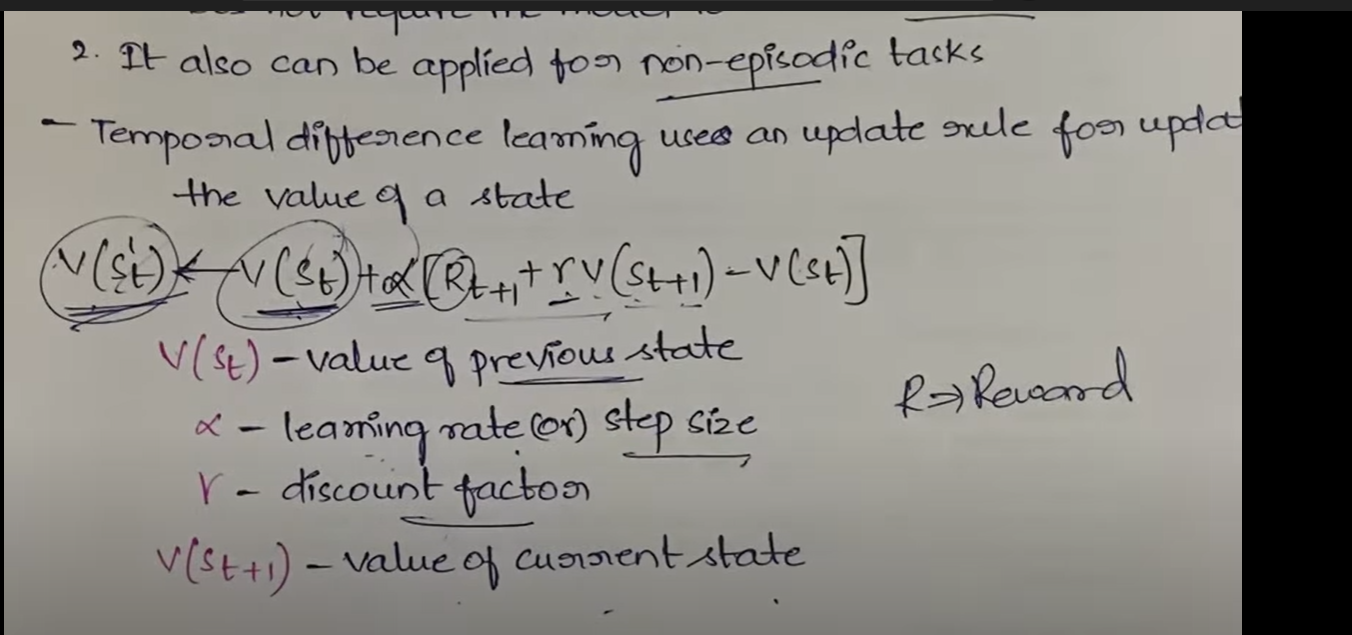
* An ADP agent uses dependencies between states to speed up value estimation
* t follows a policy π and can use observed transitions to incrementally built the
* transition model P(s’|s; π (s))
* t can then plug the learned transition model and observed rewards R(s) into the Bellman equations to U(s)
* The equations are linear because there is no max operator

->easier to solve

* The result is U(s) for the given policy π

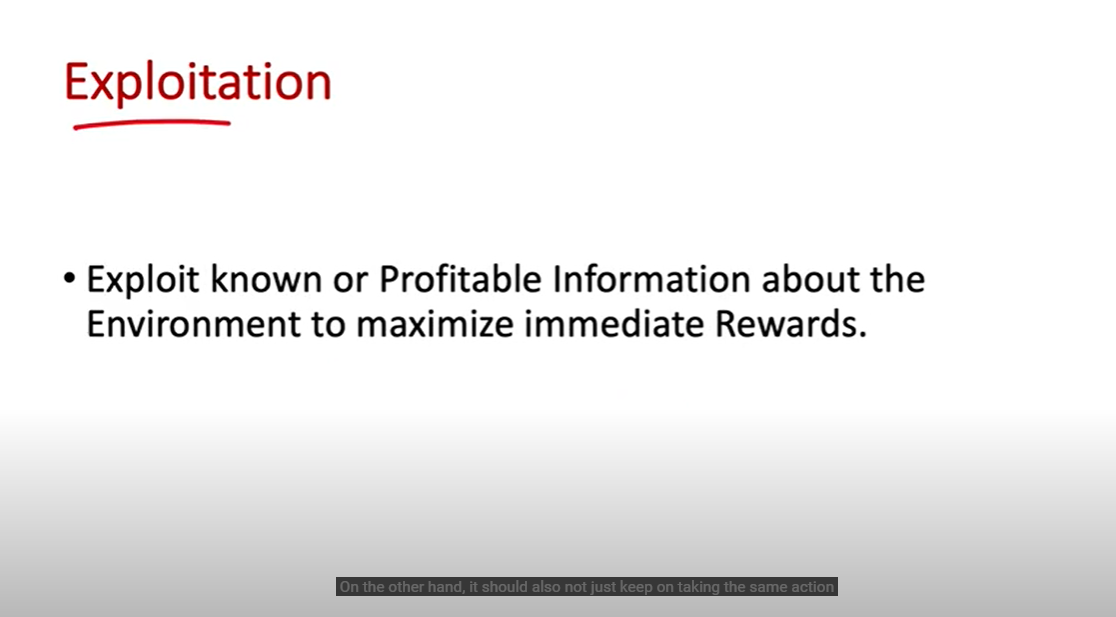
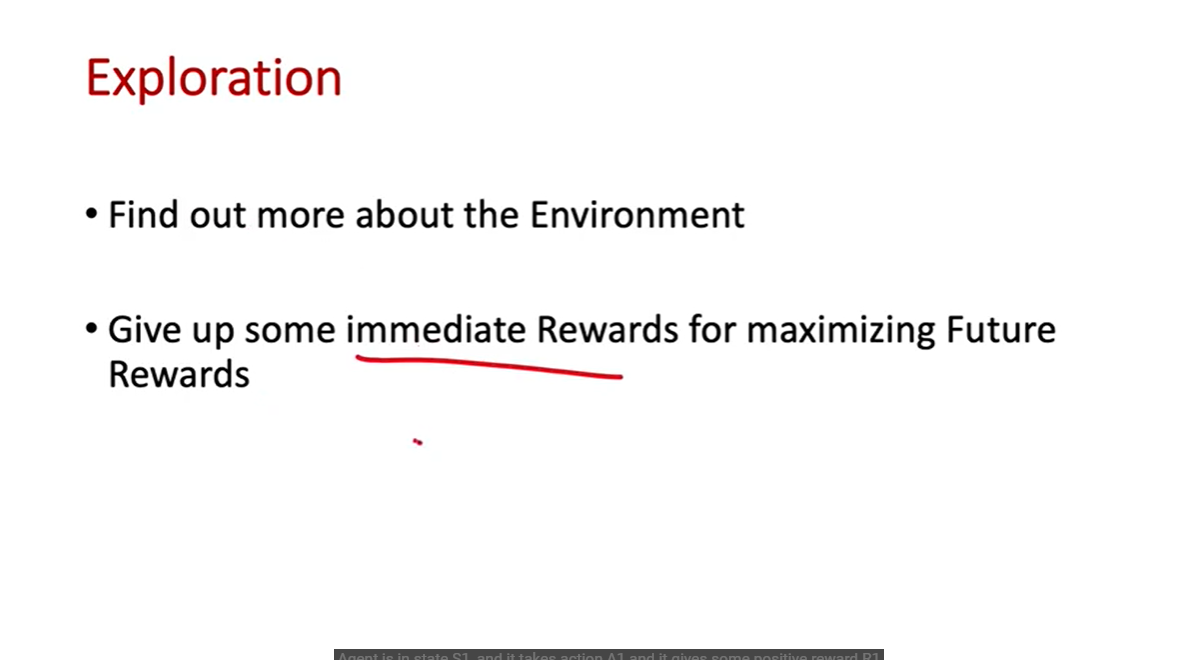


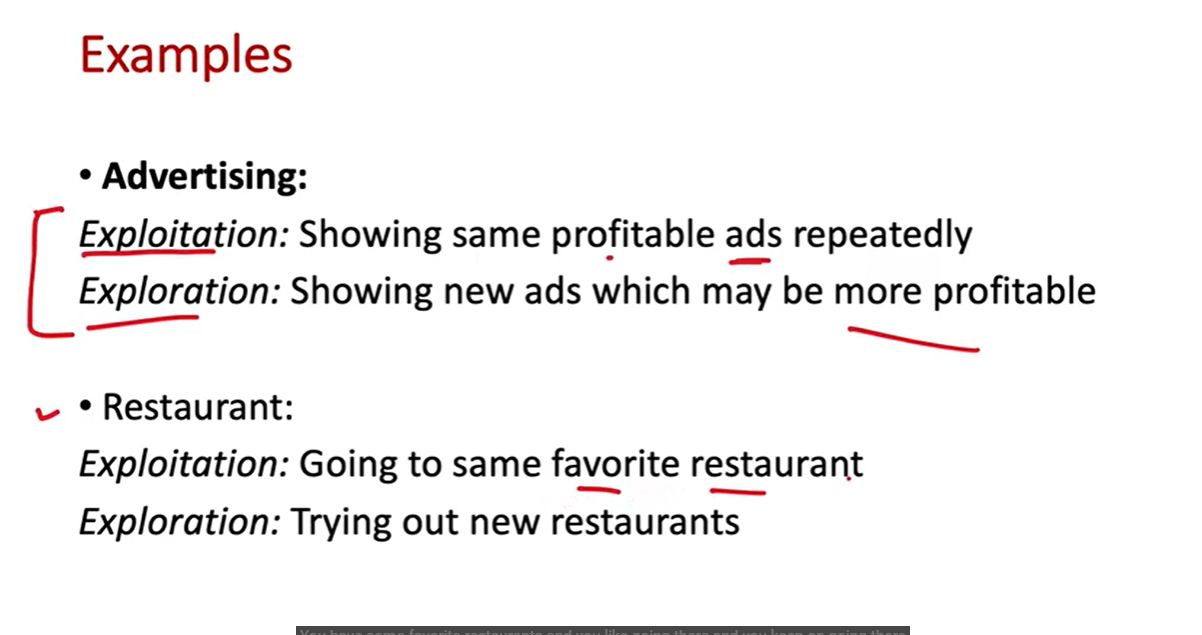


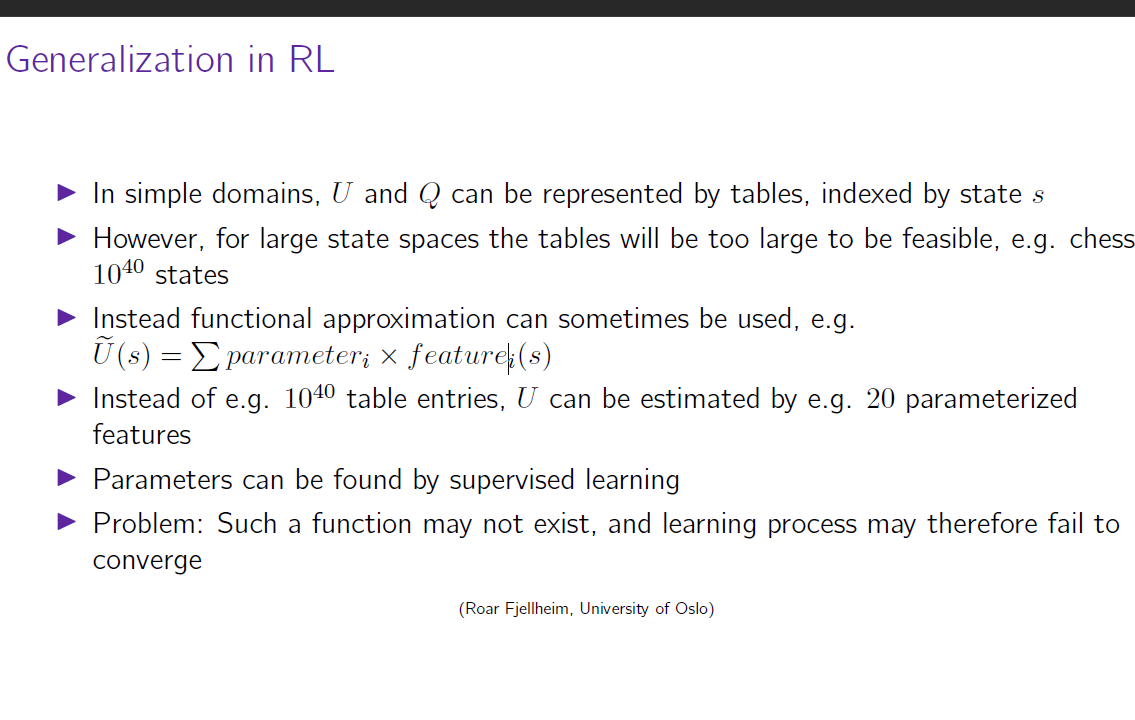


**Active Reinforcement Learning**

* While a passive RL agent executes a fixed policy π, an active RL agent has to decide which actions to take
* An active RL agent is an extension of a passive one, e.g. the passive ADP agent, and adds
* Needs to learn a complete transition model for all actions (not just π), using passive ADP learning
* Utilities need to reflect the optimal policy π\*, as expressed by the Bellman equations
* Equations can be solved by VI or PI methods described before
* Action to be selected as the optimal/maximizing one





**Applications of ML in Robotics**

* **Assistive and medical technologies**

an assistive robot is a device that can brain, process information, and execute actions that can help people with disabilities & seniors.

And smart assistive technologies also exist for ordinary people or users like driver assistance tools. Movement robots give you a therapeutic or diagnostic benefit.

* **Automatic translation**

It is an uncomplicated concept that everyone can easily understand. [Machine learning can be used to translate text into another language](https://www.usmsystems.com/natural-language-processing/) instantaneously.

Apart from the above, it can also be done the same thing with text on images. When it comes to the text, the algorithm can learn about how words in shape together and translate more precisely.

When it comes to images, the neural network identifies letters from the picture, pulls them into text, and then does the translation before placing them back into the image.

* **Computer vision**

Though computer vision is much related to what we are taking, there is some discussion going on is machine/robot vision is the right term when compared to computer vision because robot vision involves more than computer algorithms.

Robot vision so much strongly linked to machine vision that it can be given credit for the emergency of an automatic inspection system and robot guidance.

Now let us see a simple example that is anomaly detection with unsupervised learning like developing systems competent of discovering and assessing faults in silicon wafers with the help of convolutional neural networks like engineered by researchers at the Biomimetic Robotics & machine learning lab.

1. **Imitation learning**

Imitation learning is something that is very much similar to observational learning. It is the behavior exhibited by humans do as infants and toddlers, and it comes under the category of reinforcement learning.

, Imitation learning became an integral part of field robotics industries such as agriculture, construction, military, search & security, and many more

* **Multi-agent Learning**

Negotiation and coordination are the significant components of multi-agent learning that involve machine learning-based robots that can acclimatize to a changing landscape of other agents/robots and find equilibrium strategies.

Examples of multi-agent learning :

No-regret learning tools.

Market-based distributed control systems

**How Machine Learning is useful in gaming**

Game engines have embraced the idea of incorporating ML into all aspects of its product and not just for use as a game AI. While most developers may try to use ML for gaming, it certainly helps game development in the following areas:

**Map/Level Generation**: There are already plenty of examples where developers have used ML to auto-generate everything from dungeons to the realistic terrain. Getting this right can provide a game with endless replayability, but it can be some of the most challenging ML to develop.

**Texture/Shader Generation**: Another area that is getting the attention of ML is texture and shader generation. These technologies are getting a boost brought on by the attention of advanced generative adversarial networks, or GAN. There are plenty of great and fun examples of this tech in action; just do a search for DEEP FAKES in your favorite search engine.

**Model Generation**: There are a few projects coming to fruition in this area that could greatly simplify 3D object construction through enhanced scanning and/or auto-generation. Imagine being able to textually describe a simple model and having ML build it for you, in real-time, in a game or other AR/VR/MR app, for example.

**Audio Generation:** Being able to generate audio sound effects or music on the fly is already being worked on for other areas, not just games. Yet, just imagine being able to have a custom designed soundtrack for your game developed by ML.

**Artificial Players:** This encompasses many uses from the gamer themselves using ML to play the game on their behalf to the developer using artificial players as enhanced test agents or as a way to engage players during low activity. If your game is simple enough, this could also be a way of auto testing levels.

**NPCs or Game AI**: Currently, there are better patterns out there to model basic behavioral intelligence in the form of Behavior Trees. While it’s unlikely that BTs or other similar patterns will go away any time soon, imagine being able to model an NPC that may actually do an unpredictable, but rather cool behavior. This opens all sorts of possibilities that excite not only developers but players as well.

**Decision Tree**

* Decision Tree is a non-parametric supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems
* In a decision tree, there are two types of nodes: decision node and leaf node. Decision nodes are used to make any decision and have multiple branches, branches represent the decision rules and each leaf node represents the outcome
* It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure
* A decision tree simply asks a question, and based on the answer

(Yes/No), it further split the tree into subtrees

* In order to build a tree, we use the CART algorithm, which stands for

Classification and Regression Tree algorithm

**Making of decision tree**

* Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
* Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
* Step-3: Divide the S into subsets that contains possible values for the best attributes.
* Step-4: Generate the decision tree node, which contains the best attribute.
* Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example:

Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:

Information Gain:

* Information gain is the measurement of changes in entropy after the

segmentation of a dataset based on an attribute

* It calculates how much information a feature provides us about a class
* According to the value of information gain, we split the node and

build the decision tree

* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Advantages of the Decision Tree

* It is simple to understand as it follows the same process which a human follow while making any decision in real-life
* It can be very useful for solving decision-related problems
* It helps to think about all the possible outcomes for a problem
* There is less requirement of data cleaning compared to other algorithms

Disadvantages of the Decision Tree

* The decision tree contains lots of layers, which makes it complex
* It may have an overfitting issue, which can be resolved using the Random Forest algorithm
* For more class labels, the computational complexity of the decision tree may increase

**Probability Estimation Tree**

* Probability estimation tree (PET) generalizes classification tree in that they assign class probability distributions instead of class labels to examples that are to be classified. This property has been demonstrated to allow PET to outperform classification trees with respect to ranking performance, as measured by the area under the ROC curve (AUC)
* It has further been shown that the use of probability correction improves the performance of PET. This has led to the use of probability correction also in forests of PETs. However, it was recently observed that probability correction may in fact deteriorate performance of forests of PETs
* Decision trees that yield class probabilities for different class labels

are called probability estimation tree (PET)

* Decision Tree is turned into probability estimation tree by storing a probability for each possible class at the terminal nodes instead of a

single result class

**Regression Tree**

* The general regression tree building methodology allows input variables to be a mixture of continuous and categorical variables
* A regression tree may be considered as a variant of decision trees, designed to approximate real-valued functions, instead of being used for classification methods
* A regression tree is built through a process known as binary recursive partitioning, which is an iterative process that splits the data into partitions or branches, and then continues splitting each partition into smaller groups as the method moves up each branch
* Initially, all records in the training set (pre-classified records that are used to determine the structure of the tree) are grouped into the same partition
* The algorithm then begins allocating the data into the first two partitions or branches, using every possible binary split on every field

**Clustering Tree**

* The decision tree is a flexible and useful classification tool. But on the data with high dimensionality, it meets problems
* For most of current decision tree algorithms, when splitting a node of a tree, only the “best” one feature is selected and used. Since more features are ignored, the classification accuracy is not high. To solve the problem, cluster tree is used for text categorization
* Unlike decision trees (e.g. CART, C4.5), clustering results are used as

the splitting rule and more features are considered

* Obviously, the used clustering algorithm is a very important to the cluster tree
* For better performance, a text clustering algorithm is proposed to

enhance the cluster tree.

* cluster tree solves the high-dimensionality problem and outperforms CART on text data
* The cluster tree is a flow-cluster-like tree structure (Fig.1). The nodes are clusters or subsets of the training set
* In the internal nodes, the class-purity is not high. But in the leaf nodes, the class-purity is high generally
* Hence, if a new object drops in a leaf cluster, its class label can be estimated by the information of the cluster
* For the cluster tree algorithm, it includes two steps, generating tree and classification using tree
* In generating a cluster tree, the class-purity and the sample-capacity are two important parameters