**National University of Computer & Emerging Sciences, Karachi** 

**Computer Science Department**

**Spring 2022, Lab Manual – 11**

|  |  |
| --- | --- |
| **Course Code: AI-2002** | **Course : Artificial Intelligence** **Lab** |
| **Instructor(s):** | **Kariz Kamal, Erum Shaheen, Mafaza Mohi, Danish Waseem, Ali Fatmi** |

# Contents:

1. Unsupervised Learning
2. K-Means Clustering
3. Introduction to Re-inforcement Learning.
4. Tasks

# Unsupervised Learning:

**Unsupervised Learning** is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabelled data.

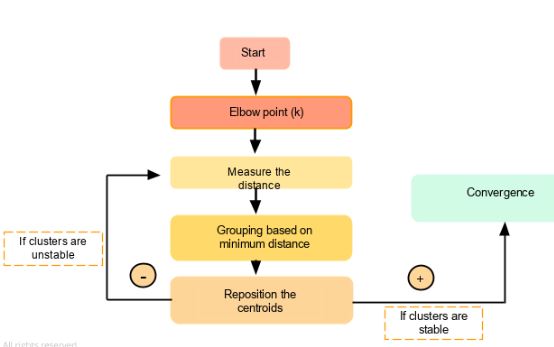
# K-Means Clustering:

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster.

The term ‘K’ is a number. You need to tell the system how many clusters you need to create. For example, K = 2 refers to two clusters. There is a way of finding out what is the best or optimum value of K for a given data.

How Does K-Means Clustering Work:

The flowchart below shows how k-means clustering works:



# The goal of the K-Means algorithm is to find clusters in the given input data. There are a couple of ways to accomplish this. We can use the trial and error method by specifying the value of K (e.g., 3,4, 5). As we progress, we keep changing the value until we get the best clusters.

# Another method is to use the Elbow technique to determine the value of K. Once we get the K's value, the system will assign that many centroids randomly and measure the distance of each of the data points from these centroids. Accordingly, it assigns those points to the corresponding centroid from which the distance is minimum. So each data point will be assigned to the centroid, which is closest to it. Thereby we have a K number of initial clusters.

# For the newly formed clusters, it calculates the new centroid position. The position of the centroid moves compared to the randomly allocated one.

# Once again, the distance of each point is measured from this new centroid point. If required, the data points are relocated to the new centroids, and the mean position or the new centroid is calculated once again.

# If the centroid moves, the iteration continues indicating no convergence. But once the centroid stops moving (which means that the clustering process has converged), it will reflect the result.

# Let's use a visualization example to understand this better.

# We have a data set for a grocery shop, and we want to find out how many clusters this has to be spread across. To find the optimum number of clusters, we break it down into the following steps:

### **Step 1:**

# The Elbow method is the best way to find the number of clusters. The elbow method constitutes running  K-Means clustering on the dataset.

# Next, we use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. Within the sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.

# slide34

# The WSS is measured for each value of K. The value of K, which has the least amount of WSS, is taken as the optimum value.

# Now, we draw a curve between WSS and the number of clusters.

# slide35

# Here, WSS is on the y-axis and number of clusters on the x-axis.

# You can see that there is a very gradual change in the value of WSS as the K value increases from 2.

# So, you can take the elbow point value as the optimal value of K. It should be either two, three, or at most four. But, beyond that, increasing the number of clusters does not dramatically change the value in WSS, it gets stabilized.

### **Step 2:**

# Let's assume that these are our delivery points:

# delivery points

# We can randomly initialize two points called the cluster centroids.

# Here, C1 and C2 are the centroids assigned randomly.

### **Step 3:**

# Now the distance of each location from the centroid is measured, and each data point is assigned to the centroid, which is closest to it.

# This is how the initial grouping is done:

# initial grouping

### **Step 4:**

# Compute the actual centroid of data points for the first group.

### **Step 5:**

# Reposition the random centroid to the actual centroid.

# random centranoid

### **Step 6:**

# Compute the actual centroid of data points for the second group.

### **Step 7:**

# Reposition the random centroid to the actual centroid.

# actual centroid

### **Step 8:**

# Once the cluster becomes static, the k-means algorithm is said to be converged.

# The final cluster with centroids c1 and c2 is as shown below:

# final centroid

# K-Means Clustering Algorithm

Let's say we have x1, x2, x3……… x(n) as our inputs, and we want to split this into K clusters.

The steps to form clusters are:

Step 1: Choose K random points as cluster centers called centroids.

Step 2: Assign each x(i) to the closest cluster by implementing euclidean distance (i.e., calculating its distance to each centroid)

Step 3: Identify new centroids by taking the average of the assigned points.

Step 4: Keep repeating step 2 and step 3 until convergence is achieved

Let's take a detailed look at it at each of these steps.

# Reinforcement Learning:

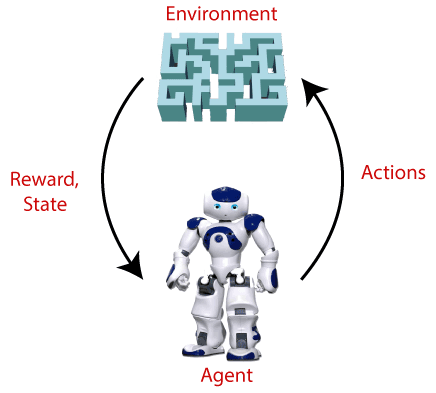
Reinforcement Learning is defined as a Machine Learning method that is concerned with how software agents should take actions in an environment. Reinforcement Learning is a part of the deep learning method that helps you to maximize some portion of the cumulative reward.

This neural network learning method helps you to learn how to attain a complex objective or maximize a specific dimension over many steps.

Here are some important terms used in Reinforcement AI:

* **Agent:**It is an assumed entity which performs actions in an environment to gain some reward.
* **Environment (e):**A scenario that an agent has to face.
* **Reward (R):**An immediate return given to an agent when he or she performs specific action or task.
* **State (s):**State refers to the current situation returned by the environment.
* **Policy (π):**It is a strategy which applies by the agent to decide the next action based on the current state.
* **Value (V):**It is expected long-term return with discount, as compared to the short-term reward.
* **Value Function:**Itspecifies the value of a state that is the total amount of reward. It is an agent which should be expected beginning from that state.
* **Model of the environment:**This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.
* **Model based methods:** It is a method for solving reinforcement learning problems which use model-based methods.
* **Q value or action value (Q):**Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action.

RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.



# Types Of Reinforcement Learning:

There are mainly two types of reinforcement learning, which are:

* **Positive Reinforcement**
* **Negative Reinforcement**

**Positive Reinforcement:**

The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.

This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

**Negative Reinforcement:**

The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.

## **Reinforcement Learning Algorithms**

Reinforcement learning algorithms are mainly used in AI applications and gaming applications. The main used algorithms are:

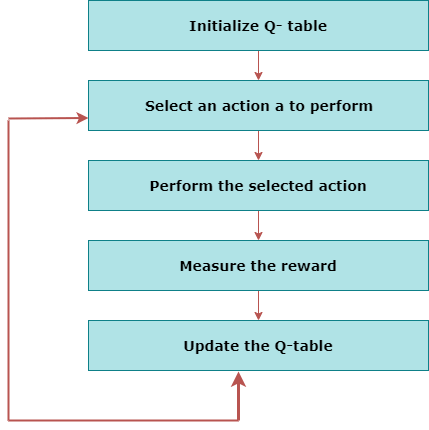
**Q-Learning:**

* + Q-learning is an **Off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
  + It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."

**Q-Table:**

The Q-table is a matrix where we have a row for every state (2000) and a column for every action (4). It's first initialized to 0, and then values are updated after training. Note that the Q-table has the same dimensions as the reward table, but it has a completely different purpose.

The below flowchart explains the working of Q- learning



**Note:Refer RL Notebook for Q-Learning Implementation.**

**SARSA-Learning:**

SARSA algorithm is a slight variation of the popular Q-Learning algorithm. For a learning agent in any Reinforcement Learning algorithm it’s policy can be of two types:- 

1. **On Policy:** In this, the learning agent learns the value function according to the current action derived from the policy currently being used.
2. **Off Policy:** In this, the learning agent learns the value function according to the action derived from another policy.

Q-Learning technique is an **Off Policy** technique and uses the greedy approach to learn the Q-value. SARSA technique, on the other hand, is an **On Policy** and uses the action performed by the current policy to learn the Q-value.

* In Q-learning, this is done by choosing the **greedy action**agag, i.e the action that maximize the Q-value function at the new state Q(s’, a):

Q(s,a)=Q(s,a)+α[r+γQ(s′,ag)−Q(s,a))]

Q-learning is called off-policy learning because the new action agag is taken as greedy, not using the current policy.

* n SARSA, this is done by choosing another action a′a′ following the **same current policy** above and using r+γQ(s′,a′)r+γQ(s′,a′) as target.

Q(s,a)=Q(s,a)+α[r+γQ(s′,a′)−Q(s,a))]

SARSA is called on-policy learning because new action a′a′ is chosen using the same epsilonepsilon-greedy policy as the action aa, the one that generated s′s′.

Tasks:

1. Load Gapminder dataset:

**Input data set is gapminder**

**○ !pip install gapminder**

**○ install.packages("gapminder")**

 2)Apply K-mean clustering with and without elbow method for

a. GDP and continent .

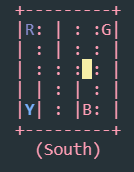
b. Life expectancy and continent.

c. Plot the clusters formed by k-mean .

3) Run the SARSA notebook with large number of Timestamps

4) Q-learning Scenario:

* Starting at a random state, our job is to get the taxi to the passenger’s location, pick up the passenger and drive to the destination, drop the customer, and then the episode ends.
* There are 4 designated locations in the grid indicated by **Red — 0 , Green — 1, Yellow — 2, and Blue — 3**, the blue letter correspond to pick up location and purple letter indicate the drop off location. The solid lines indicate walls that the taxi cannot pass, whereas the filled rectangle is the taxi, when it is yellow it is empty and when it is green it is carrying a passenger.



* **Each state is defined by a 4 entries tuple: （taxi\_row, taxi\_col, passenger\_location, destination).**For example, the image shows state (2,3,2,0), which means we are at position row index 2 (note that python index start at 0 so this means row 3), and column index 3, the passenger is at Yellow, encoded by 2 and our destination is red, encoded by 0.
* State Space: We can see that our state space consist of **500 possible states**, with 25 possible taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations
* Action space: There are **6 discrete deterministic actions:** 0 — move south, 1 — move north, 2 — move east, 3 — move west, 4 — pickup passenger , 5 — drop off passenger
* Rewards: Except for delivering the passenger with gets a reward of +20, each extra step has a penalty of R=-1, executing “pickup” and “drop-off” actions illegally results in R=-10