AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

REINFORCEMENT LEARNING

Technical Communication Project by:

Deepthi Sudharsan	CB.EN.U4AIE19022
Isha Indhu S	CB.EN.U4AIE19030
Jayashree O	CB.EN.U4AIE19031
Kavya S Kumar	CB.EN.U4AIE19037
Lakshaya Karthikeyan	CB.EN.U4AIE19039
Meghna B Menon	CB.EN.U4AIE19043

ACKNOWLEDGEMENT

For the completion of this project, we have been guided by our Technical Communication faculty, Mr Saravana Prabhu, who has provided us with sufficient knowledge and guidelines.

We would also like to thank our other professors for supporting this project and Amrita University for this opportunity to hone and display our skills.

DECLARATION

We hereby declare that the project entitled “ **Reinforcement Learning** ” submitted to Amrita Vishwa Vidyapeetham, Coimbatore is the record of original work done by us under the guidance of Mr. Saravana Prabu, Department of English, Amrita Vishwa Vidyapeetham and this project work has not been submitted for any degree, diploma or other similar titles elsewhere. However, extracts of any content which has been used for this project have been duly acknowledged by providing the details in the references.

May 5, 2020

Deepthi Sudharsan (CB.EN.U4AIE19022)

Isha Indhu S (CB.EN.U4AIE19030)

Jayashree O (CB.EN.U4AIE19031)

Kavya S Kumar (CB.EN.U4AIE19037)

Lakshaya Karthikeyan (CB.EN.U4AIE19039)

Meghna B Menon (CB.EN.U4AIE19043)

TABLE OF CONTENTS

Introduction to Artificial Intelligence ………………………………………..……… (5)

Super Intelligence ………………………………………………………………… (6)

General Artificial Intelligence ………………………………………….…….. (6)

Narrow Artificial Intelligence ……………………………………………………….. (7)

Machine Learning ……………………………………………………………...… (7)

Reinforcement Learning ……………………………………………………… (8)

Markov Decision Process ……………………………………………………. (12)

Value Functions ………………………………………………………………. (14)

State Value Function …………………………………………..…………..… (14)

Action Value Function ……………………………………………...………… (15)

Introduction to Various Reinforcement Algorithms …………..………………… (15)

Python Code …...……………………………………….……………..………… (18)

Deep Q Network Algorithm ……………………………………………….…… (18)

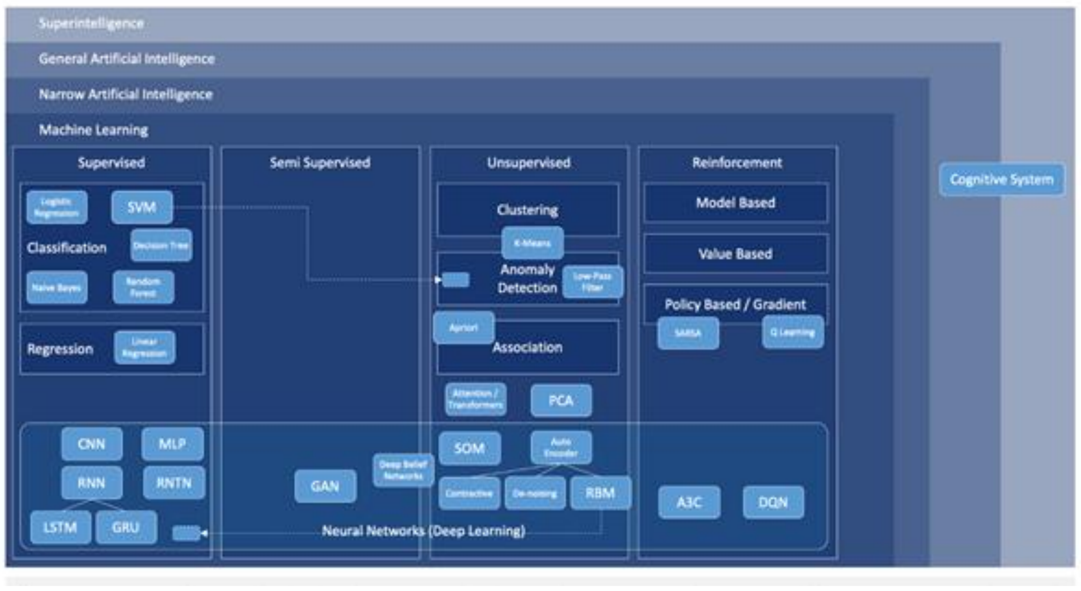
Applications of Reinforcement Learning ….……………………………..… (20)

Future Scope of Reinforcement Learning ..…………………………..………… (22)

References ………………………………...…………………………….………… (23)

Introduction to Artificial Intelligence

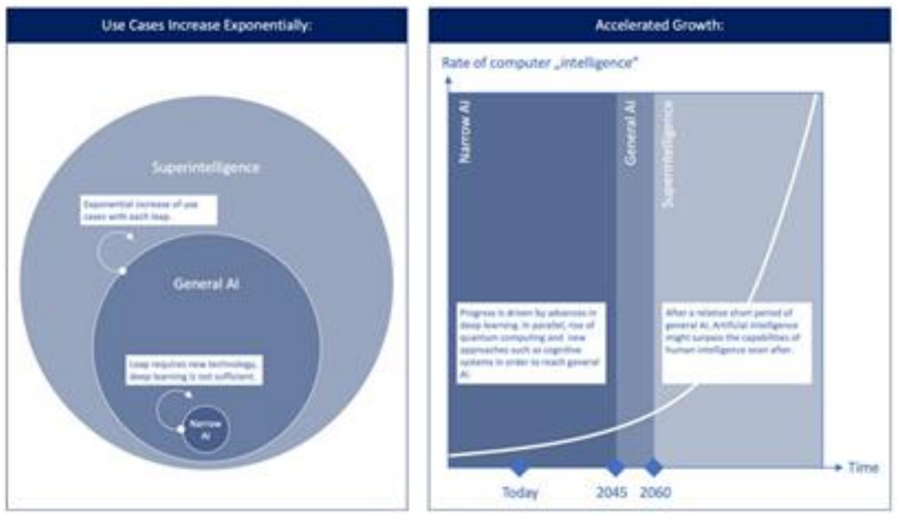
Artificial Intelligence is the new trending technology that has been transforming the world. Despite its widespread unfamiliarity, it has significantly altered all walks of life and has raised important questions for the society, the economy, and the governance. AI is an immensely powerful, wide ranging tool that enables one to efficiently integrate information, analyze data, and use the resulting insights to improve our decision-making. AI can be applied to any field under the sun - finance, national security, health care, criminal justice, transportation, and smart cities, and address issues such as data access problems, algorithmic bias, AI ethics and transparency, and legal liability for AI decisions.



AI is a branch of computer science that deals on how to create machines or software that possess human capabilities such as their intelligence. There are different maturity levels of artificial intelligence – superintelligence, general artificial intelligence, and narrow artificial intelligence.

Superintelligence

Researchers have predicted that at one point in the future, machines will become much more efficient and smarter than humans. They also believe that this might happen between the years of 2050-2100. That state is called the superintelligence where the machine's cognitive capability surpasses that of humans in virtually all domains of interest. The advent of superintelligence is based on the assumption that the rate of progress in an evolutionary learning environment is evolving exponentially.


Use Cases Increase Exponentially:
Superintelligence
Exponential increase of use cases with each leap.
General AI
Leap requires new technology, deep learning is not sufficient.
Narrow AI
Accelerated Growth:
Rate of computer „intelligence"
Narrow AI
General AI
Superintelligence
Progress is driven by advances in deep learning. In parallel, rise of quantum computing and new approaches such as cognitive systems in order to reach general AI.
After a relative short period of general AI, Artificial intelligence might surpass the capabilities of human intelligence soon after.
Time
Today
2045
2060

Superintelligence therefore can either in the best case, lead us to a future of abundance and equity or in the worst case, lead to the extinction of mankind not because superintelligence is evil but just because we stand in its way on achieving its goals.

General Artificial Intelligence

General AI is the state before superintelligence where the machine will have the same capability as a normal human being. It is estimated to be commonplace in 2045. Intelligence highly depends on the ability of an organism to adapt and cope with a changing environment by either changing its behavior or by changing the environment. Some inherent characteristics of general AI are as follows:

- Being able to learn from past experiences

- Store, alter or retrieve from memory
- Draw logical conclusions
- Problem solving ability
- Divergent thinking
- Ability to narrow down a list of possible options
- Recognize and respond to human emotions

Narrow Artificial Intelligence

Since researchers were not that successful on achieving general AI, their focus eventually turned towards narrow AI. As the name suggests, narrow AI was trained for extremely specific purposes. Therefore, it was able to handle only events that it has been trained for. Narrow AI's capability to learn is limited as it is restricted to focus on a particular goal. Some typical narrow AI applications are natural language recognition and processing, autonomous driving, visual image recognition and interpretation, intrusion detection in cyber security and human activity recognition.

Machine Learning

Machine Learning is a subset of the field of Artificial Intelligence. While Artificial Intelligence requires various cognitive capabilities to be able to observe and respond to the external world, machine learning limits itself to the process of learning from patterns in data and predict future patterns. It allows computers to learn with minimal human inputs.

Machine Learning can be broadly subdivided into sub-categories: supervised, semi-supervised, unsupervised learning, reinforcement learning and deep learning.

In **Supervised learning**, the learning dataset includes 'labels' that help the computer infer patterns. We teach the computer via labels, the different groups/clusters/categories in the dataset. For instance, we will use credit card transactions and label which of those are genuine and fraud cases. This is very similar to how we teach children by repetition, what an apple or orange is - they learn by noticing the similarities and differences.

In **Unsupervised learning**, the computer is left to identify patterns and hidden structures without any label inputs. We let the computer decipher what the similarities and differences in the dataset are purely from a statistical perspective. In human terms, imagine if you see a picture of some exotic fruits you have never seen or heard before -

even then, you will be able to say how many ‘types’ of fruits there are in the picture by looking at their size, color, shape etc. In essence, you are performing unsupervised learning. A good real-world example is to let the computer classify user online behavior. This could lead to patterns based on age (young/middle-age and old buyers), on lifestyle (high, low-spenders), on buyer-propensity (curious, serious etc.).

In **Reinforcement learning**, we teach the computer how to group/classify/predict something by providing a reward when it guesses something correctly. It is similar to teaching your pet-dog to shake hands or stay-still. If the dog does what is expected, we provide him a reward. The dog soon learns what is expected and starts behaving the same way. A very good example of reinforcement is in game-bots. Some games have large state-space (possible positions) though the game rules might be simple. If we let the computer take an action and rebuke/reward it (using a reward function), it will soon learn what is the right move.

Deep Learning models a problem using artificial neural networks. However, at the core, it uses concepts from supervised and reinforcement learning. It is called “Deep” to highlight the complexity and depth of a neural network.

The **neural network** tries to model a complex equation. At the simplest level, imagine you had two variables (x,y) and you can combine them using + or * operators - you can then create equations of the first-order like ax+by+cxy etc..

By using more layers in between the input and the output, we can model higher-order equations and thus represent complicated equations.

Till the last decade, practical application of Deep Learning has been limited by compute capacity and availability of data. Now, Deep Learning is routinely used for complex problems involving image/video recognition, time-series analysis etc. which were not possible earlier. Google uses Deep Learning to identify and tag people in photos and ability to search your photos using plain text search.

Reinforcement Learning

Reinforcement learning is a machine learning technique that involves training an artificial intelligence agent through repetition of actions and associated rewards. It is modelled to make a sequence of decisions. It employs a trial and error technique to come up with the best possible behavior or path given a specific situation. There are five main components in reinforcement learning:

1) Agent (Decision makers)
2) Environment (Physical world in which the agent operates)
3) States (Current situation of the agent)
4) Actions
5) Rewards (Feedback from the environment)

The agent learns to achieve a goal in an uncertain environment, which is possibly complex. It either gets rewarded or penalized based on the action it performs. The goal of this system is to maximize the total reward. Though the designer states and sets the reward policy, he does not provide any hints or suggestions as to how to obtain the solution. The machine thus starts by completely random trials. As it learns from each trial, it ends up developing tactics to efficiently achieve the goal.

In supervised learning, the training data has an answer key with it, so the model is trained along with the correct answer. Whereas, in reinforcement learning, there is no answer. The reinforcement agent must innovate a way to perform the given task. This is based on the idea that in the absence of training data, the model is bound to learn from its experience. The system has three stages:

1) Input

This is an initial stage from which the model will start.

2) Training

Based upon the input, the model will return a state. Consequently, the user will have to either reward or punish the model based on its output.

3) Output

A myriad of outputs will be available as there are a variety of solutions to each problem.

The best, most optimal solution, is decided based on which output obtained the maximum reward. The reward vs punishment system is termed as positive and negative reinforcement.

The term “reinforcement learning” has been adapted from the concept of reinforcement in psychology. In the psychological sense, the term reinforcement refers to something that increases the likelihood that a particular response/action will occur. This concept of reinforcement is a central idea of the theory of operant conditioning initially proposed by the psychologist B.F. Skinner. Reinforcement is anything that causes the frequency of a given behavior to increase. There are two types of reinforcement: Positive and negative.

In the psychological sense, positive reinforcement is defined as the addition of something to increase a behavior, typically a reward. Negative reinforcement is defined as removing a stimulus to elicit a behavior.

In machine learning, positive reinforcement increases the frequency of a behavior while negative reinforcement decreases the frequency. Positive reinforcement is the most used as it allows models to maximize the performance on a given task. Furthermore, it leads the model to make more sustainable changes which can become consistent patterns and persist for longer periods of time.

On the other hand, negative reinforcement is mainly used for maintaining a minimum performance standard rather than reaching a model’s maximum performance. It is beneficial in ensuring that a model is kept away from undesirable actions, but it isn’t equipped to make a model explore desired actions.

When a reinforcement learning agent is used, there are four different states used in the training: initial state (State 0), new state (State 1), actions and rewards. The initial state of the game is drawn from the environment and based on this given information; the model must then consequently decide on an action.

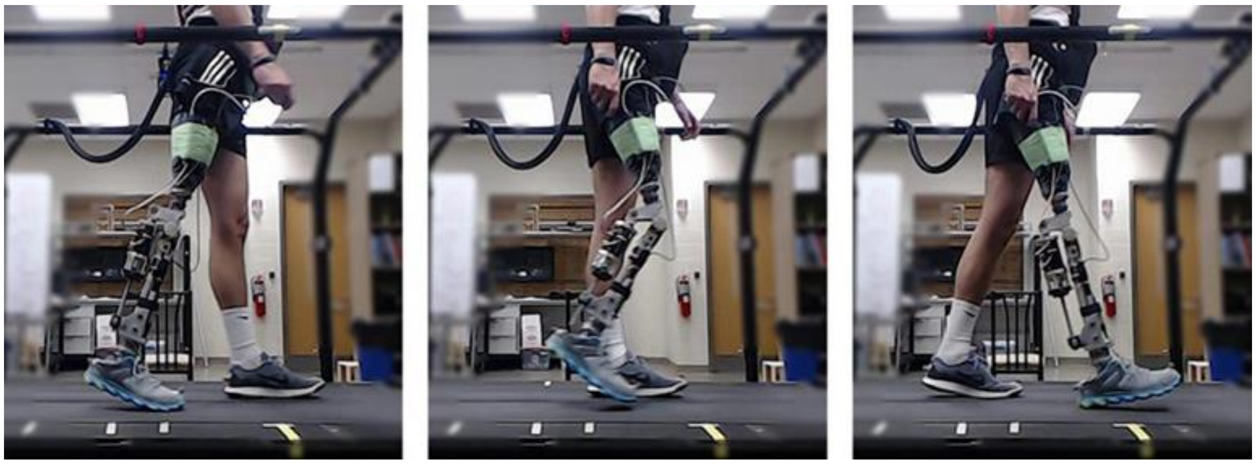
During the initial phases of training, these actions are random. However, after each trial, the model becomes reinforced and certain actions will become more common as the model learns through positive reinforcement. Similarly, certain unsatisfactory actions will become less common as the model receives negative feedback.

Reinforcement learning is currently the most effective way to make a machine creative. This methodology forces the machine to seek new, innovative ways to arrive at an optimum solution.

In order to build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its reward at the same time. This problem is termed as the ‘Exploration vs Exploitation trade-off’.

Exploration is the act of collecting more information about the surrounding environment, while exploration is using the information already known about the environment to earn reward points. If an agent only explores and never exploits the environment, the desired actions will never be carried out. However, if the agent only exploits and never explores, their will only learn to carry out one action and won’t discover other possible strategies of earning rewards. Thus, balancing these two metrics is critical when creating a reinforcement learning agent.

A real-life example of reinforcement learning is an automated tuning system that helps train prosthetic legs. Using existing approaches, trained clinicians take up to half a day for this process while the system manages to accomplish it in around ten minutes. The patient simply has to walk with a new prosthetic on a treadmill while guided by a therapist. The system monitors the intentions of the patient and the following movements of the prosthetic. Each patient starts with a randomly selected set of parameters. However, based on the readings, the prosthetic adjusts a dozen different parameters, such as joint stiffness, throughout each step during the training process in real time. This is done through a computer model which compares the patient’s gait to the profile of a normal walking gait in real time, and respectively adapts parameters on the device. The model is able to identify which parameter settings improve and which impairs performance. Using reinforcement learning, the model can quickly identify the set of parameters that allows the patient to walk most comfortably.



To summarize: reinforcement learning is a specialized application of machine and deep learning techniques, which has been designed to solve problems in a certain way so as to maximize the reward received. The key distinguishing factor is how the agent is trained. Instead of inspecting the data provided, as in other methodologies, reinforcement learning interacts with the environment.

Markov Decision Process (MDP):

Markov decision processes help to formulate decision making in a sequential manner. This formalization is the basis for structuring problems that are solved with reinforcement learning. In a Reinforcement learning problem, there are usually five components. The Markov decision process provides mathematical functions that help relate the components.

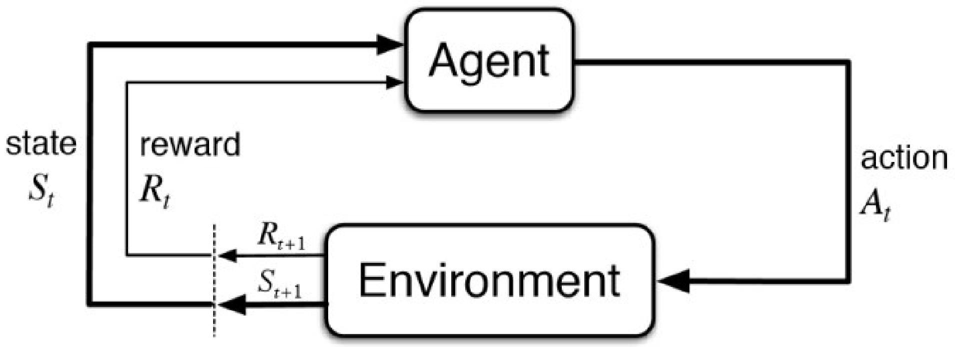
The agent analyses the environment and performs a particular action at a given time step in a given state. This action results in a different state of the agent which then receives a reward for the action that it took. It must be noted that though the term “reward” is used this can be both positive and negative. The agent’s goal in reinforcement learning is to maximize its reward. The agent does not try to simply maximize immediate rewards it tries to maximize the cumulative reward.

For example, in a simple scenario of teaching a dog how to sit. At first, the dog will hardly perform the required command, and the response is to give it a negative reward (punishment). When the dog gets close to what it should do, the response is to give it a positive reward as signs of approval or encouragement. In the scenario that the owner wants the dog to sit until the owner takes a few steps back, if the dog sits for one step back, the owner gives it a small reward. However, if the dog sits patiently through a number of steps it receives a bigger reward. In this scenario the dog can perform in two ways:

- The dog may choose to test a new combination of actions by which it may or may not realize the optimal combination. This may lead to the dog getting a bigger reward. This is known as exploration.
- The dog can stick to known action to obtain immediate rewards. This is known as exploitation.

This example illustrates the point that a cumulative reward may be bigger than immediate rewards. This example can also be extended to real time examples such as in finance this reflects well the risk aversion degree of an economic agent - how willing the agent is to take more risks in search of larger returns, or is willing to take less risk and get a smaller, but safer return.

Extending this to a mathematical model. Consider S to be a set of states. A, a set of actions and R to be a set of rewards. For each time step t = 0,1,2,3… the agent will receive information about the environment called state St that belongs to the state set S. (i.e.) St ϵ S. Based on the particular state the agent takes a particular action At. Thus, this forms a state-action pair, (St, At). The time then goes to the next step, the environment changes to a new state St+1 ϵ S which makes the agents take an appropriate action At+1 making a new state-action pair, (St+1, At+1). The action (At) taken in the previous state(St) enables the agent to receive a reward Rt+1 ϵ R.



This can be represented mathematically as,

$$f(S_t, A_t) = R_{t+1}$$

This indicates that performing a given action at a particular state leads to the agent receiving a particular reward.

$$G_t = R_{t+1} + R_{t+2} + \dots \qquad R_{t+1}, R_{t+2} \dots \in R$$

The cumulative reward that the agent must maximize can be represented as, this is called as “return”

Since the sets S and R are finite, the random variables Rt and St have well defined probability distributions. In other words, all the possible values that can be assigned to Rt and St have some associated probability. These distributions depend on the preceding state and action that occurred in the previous time step t−1.

For example, suppose s′ ∈ S and r ∈ R. Then there is some probability that St=s′ and Rt=r. This probability is determined by the particular values of the preceding state, s ∈ S and action a ∈ A(s). Note that A(s) is the set of actions that can be taken from state s.

Defining a probability for above, for all s′ ∈ S, s ∈ S, r ∈ R, and a ∈ A(s), we define the probability of the transition to state s′ with reward r from taking action a in state s as:

$$p(s',r|s,a)=Pr\{St=s',Rt=r|St-1=s,At-1=a\}$$

Value Functions

A value function in reinforcement learning is a function that measures the benefits of the agent being in a particular state or for the agent to perform certain actions in a given state. The value function can be of two types,

1. State Value Function:

   The state value function for a given policy π indicates the benefits of a state when the agent follows a policy π. It is the expected return (Gt) for starting from a state ‘s’ at a time ‘t’ and thereafter following the policy π.

$$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$
$$G_t = R_{t+1} + R_{t+2} + \dots$$
$$R_{t+1}, R_{t+2} \dots \in R$$

2. Action Value Function:

   The action value function for a given policy π indicates the benefits of the agent performing a given action after being in a particular state. It is the expected return (Gt) after starting at a state ‘s’ at a time ‘t’ taking the action ‘a’ and thereafter following the policy π.

$$q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a]$$
$$G_t = R_{t+1} + R_{t+2} + \dots$$
$$R_{t+1}, R_{t+2} \dots \in R$$

   Generally, the action value function is called a “Q-Function” and the result of the function is called a “Q-Value”. Here, ‘Q’ stands for quality.

Introduction to Various Reinforcement Learning Algorithms

There are mainly three different approaches to advance towards the implementation of a Reinforcement Learning Algorithm.

The first approach is a ‘Value-Based’ reinforcement learning method where we are expected to maximize a value function V(s). In reinforcement learning, the agent is the one who makes decisions based on the rewards and punishments. At each time ‘t’, the agent will receive an observation. Thus, in this approach, the agent is has been awaiting a long term response of the current states under the policy.

The next approach is a ‘Policy-Based’ RL method in which we try to come up with such a policy so that the action that has been performed in every state helps us to gain maximum reward in the future.

The last approach is a ‘Model-Based’ RL method in which we need to create a virtual model for every environment. The agent shall then learn to perform in that specific environment.

Q-Algorithm

Q-learning is a popular reinforcement learning algorithm that seeks to discover the most efficient action to take when given the current state. The ‘q’ stands for quality which represents the usefulness of a given action in gaining a future reward. It is considered an off-policy algorithm as the function learns from actions that are outside the current policy. This is usually done by taking random actions, and thus a policy is not required.

The main purpose of this algorithm is to learn a policy that maximizes the total reward. It is commonly used to solve for optimal policy in a Markov Decision Process which gives the best policy by suggesting the maximum expected value using value iteration method.

The mathematical theory behind the Q-algorithm is the Bellman Equation.

$$v(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1}) | S_t = s]$$

E in the above equation refers to the expectation and ƛ refers to discount factor. The above equation is rewritten in terms of Q-value as

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \dots | s, a] \\ &= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a') | s, a] \end{aligned}$$

The optimal Q-value (denoted as Q*) can be expressed as:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \lambda \max_{a'} Q^*(s', a') | s, a]$$

Where **s** is the current situation returned by the situation, **a** refers to all the possible moves that the agent can take.

During q-learning, a q-matrix that follows the shape of [state, action] is created. The values are initialized to zero. After each episode, the q-values are updated and stored. This q-matrix becomes a reference for the agent to select the best action based on the q-value.

Algorithm

1. All the values in Q-matrix are initialized to zero. The lambda value is set, and the reward matrix is filled accordingly.

2. For each episode, a random starting state is selected.

3. The current state (S) is selected among all possible actions.

4. The next state (S’) should be selected according to the action (a).

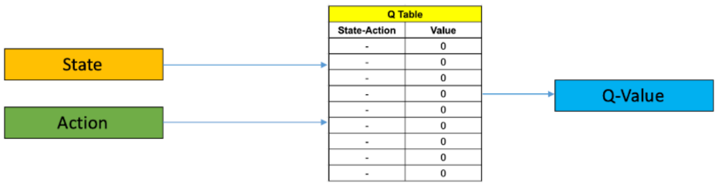
5. For all possible actions from the state (S’), the one with the highest Q value is

   selected.

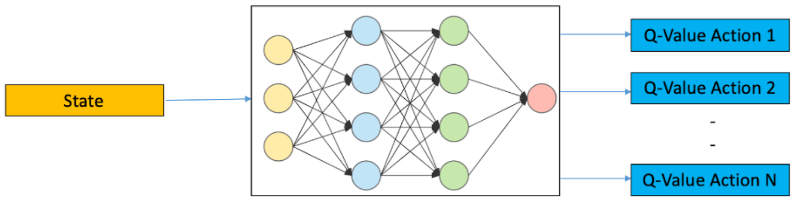
6. Q-table is updated using:

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \lambda \max_{a'} Q^*(s',a') | s,a\right]$$

7. The next state is now the current state for next iteration. This process is continued till the desired outcome is reached.

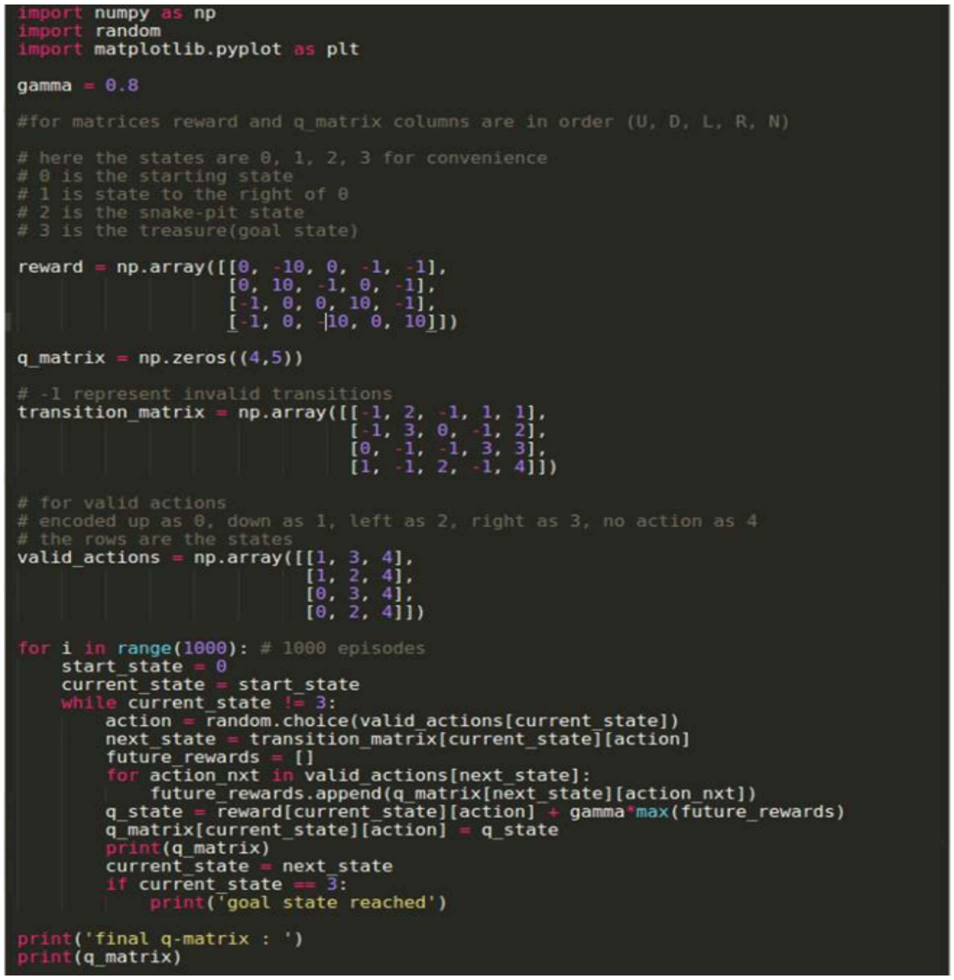


Q Learning



Deep Q Learning

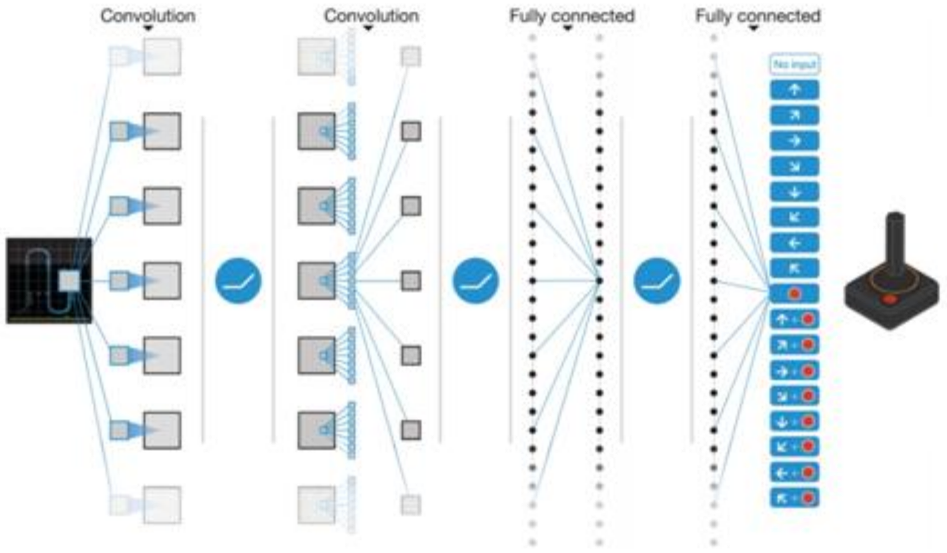
Python code



```python
import numpy as np
import random
import matplotlib.pyplot as plt

gamma = 0.8

#for matrices reward and q_matrix columns are in order (U, D, L, R, N)

# here the states are 0, 1, 2, 3 for convenience
# 0 is the starting state
# 1 is state to the right of 0
# 2 is the snake-pit state
# 3 is the treasure(goal state)

reward = np.array([[0, -10, 0, -1, -1],
                   [0, 10, -1, 0, -1],
                   [-1, 0, 0, 10, -1],
                   [-1, 0, -10, 0, 10]])

q_matrix = np.zeros((4,5))

# -1 represent invalid transitions
transition_matrix = np.array([[-1, 2, -1, 1, 1],
                              [-1, 3, 0, -1, 2],
                              [0, -1, -1, 3, 3],
                              [1, -1, 2, -1, 4]])

# for valid actions
# encoded up as 0, down as 1, left as 2, right as 3, no action as 4
# the rows are the states
valid_actions = np.array([[1, 3, 4],
                          [1, 2, 4],
                          [0, 3, 4],
                          [0, 2, 4]])

for i in range(1000): # 1000 episodes
    start_state = 0
    current_state = start_state
    while current_state != 3:
        action = random.choice(valid_actions[current_state])
        next_state = transition_matrix[current_state][action]
        future_rewards = []
        for action_nxt in valid_actions[next_state]:
            future_rewards.append(q_matrix[next_state][action_nxt])
        q_state = reward[current_state][action] + gamma*max(future_rewards)
        q_matrix[current_state][action] = q_state
        print(q_matrix)
        current_state = next_state
        if current_state == 3:
            print('goal state reached')

print('final q-matrix : ')
print(q_matrix)
```

Deep Q Network (DQN) Algorithm

One of the biggest disadvantages of Q-Learning algorithm is that it lacks generality. DQN uses a neural network to estimate the Q-value function. A neural network is nothing but a sort of computer software which uses biological neurons. The current to this network is the input that we are expected to give whereas the output is the corresponding Q-value for each of the actions.



A couple of years ago, DeepMind applied DQN to a game by ‘Atari’ wherein they gave the raw image of the current situation as the input. The input went through many layers including the convolutional layer and the fully connected layer too.

```
Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, $M$ do
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T do
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the
    network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  End For
End For
```

Pseudo Code for DQN

There are some other techniques too which are essential for training DQN. One of which is ‘Experience Replay’ wherein the sample transitions are stored which will later be randomly selected from the transition pool to update the knowledge. The other is the ‘Separate Target Network’. According to the above pseudo code, the target network is reset to another one, every C steps. Therefore, the fluctuation becomes less severe, resulting in more stable training.

Applications of Reinforcement Learning

There are a variety of applications for Reinforcement learning in the physical world. A few examples would be finance, robotics, gaming, and NLP.

1) Robotics

   For a robot to successfully operate in a particular environment it must analyze the environment and somehow, plan its actions and execute those plans, while using feedback to make sure everything proceeds according to plan. In order for a robot implemented using reinforcement learning to provide good results the following principles should be taken into account:

   - Effective representations
   - Approximate models
   - Prior knowledge or information

   Furthermore, we can use Reinforcement Learning to train the robot to perform activities like walking, picking, or moving objects etc. by using a reward function based on the position of joints, physical location of the robot etc.

2) Finance and Trade

   In finance the goal has always been to create an algorithm that learns how to dominate in the markets on its own. As mentioned above, reinforcement learning involves maximizing the cumulative reward. This can be very well applied in finance which has the same final goal - maximizing returns. Moreover, we can use Reinforcement Learning to perform portfolio optimization where the reward function could be a function of profit/loss, transaction cost, portfolio risk score etc.

3) Healthcare

   There have been several papers that suggest the use of reinforcement techniques in healthcare. For example, the paper “Reinforcement Learning with

Action-Derived Rewards for Chemotherapy and Clinical Trial Dosing Regimen Selection" by Gregory Yauney and Pratik Shah, talks about a method to find the optimal policy for treating patients with chemo via reinforcement learning. The paper also uses Q-Learning as the underlying model. For an action space, they formulate a quantity of doses for a given duration that an agent is able to choose from. Dose cycles are only initiated with a frequency determined by experts. Transitions states are computed at the end of each cycle. The reward function is defined as the mean reduction in tumor diameter. The evaluation is done using simulated clinical trials. It is unclear exactly how these simulations were set up but apparently simulations of this type generally incorporate both pathological and statistical data.

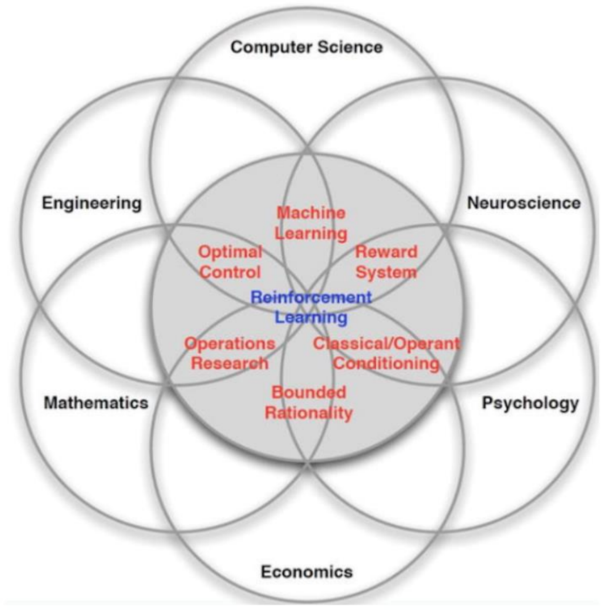
KENSCI
KenSci, a company built by doctors and data scientists, helps providers and payers intervene earlier, at lowered costs. KenSci's risk prediction platforms use existing sources such as Electronic Medical Records (EMR), Claims and Financial data to help uncover clinical, operational, and financial risks. Reinforcement Learning is used by KenSci to predict ailments and treatments to help medical practitioners and patients detect and diagnose at earlier stages. Moreover, it helps in the prediction of population health threats through pinpointing patterns, growing precarious markers, model disease advancement, among others.

Drug discovery is a great area to apply Reinforcement Learning, where we can virtually try various formulations for effectiveness.

4) Gaming

Reinforcement learning has exceeded human level performance in games with the help of deep learning methods too. The games use both the techniques to create several agents with different algorithms that successfully learn to play them. Atari Learning Environment (ALE) is a popular testbed for Deep Reinforcement Learning methods. In 2013, a company called "DeepMind" implemented a system using Reinforcement Learning that could learn how to play Atari games with human and sometimes better than human performance. The paper "Playing Atari with Deep Reinforcement Learning" applied DRL to seven different Atari games from ALE. Their method outperforms human experts in three of the seven games.

Another example of Reinforcement Learning in gaming would be DeepMind's AlphaGo Zero. The system started off with a neural network which had no knowledge on the game of Go. It then played games against itself, by combining the neural network with a powerful search algorithm. In each iteration, the network is tuned, updated and the performance of the system improves.



Future Scope of Reinforcement Learning

Though RL is in its infancy and has a lot of potential, there are a few challenges. To name a few:

1) Primary among the challenges, is the number of attempts a computer might need to learn effectively. Typically, a complicated task like playing a game needs to be done millions of times to be effective.
2) Many real-world problems where the risks of wrong results are high - like in Healthcare, Finance - ability to RL is limited.
3) The ability to provide a suitable reward function is sometimes very hard and takes trial and error.
4) There are numerous future applications that use reinforcement learning like smart traffic signals, advanced self-driving cars, autonomous robots, prosthetic limbs, automated factories etc.
5) Using a reinforcement learning approach, generating molecules which exhibit inhibitory against various diseases has been made much easier. Drug designing can now have user designed properties due to this kind of an approach.

Though there is room for improvement to overcome these challenges, reinforcement learning (RL) technology is surely going to do great things in the near future.

REFERENCES

- https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/
- https://www.geeksforgeeks.org/what-is-reinforcement-learning/
- https://www.brookings.edu/research/how-artificial-intelligence-is-transforming-the-world/
- https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html#comments
- https://www.medgadget.com/2019/01/reinforcement-learning-system-automatically-trains-prosthetic-legs.html
- https://www.daisyintelligence.com/reinforcement-learning-ai/
- https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56
- https://towardsdatascience.com/artificial-intelligence-framework-a-visual-introduction-to-machine-learning-and-ai-d7e36b304f87
- https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287
- https://www.guru99.com/reinforcement-learning-tutorial.html
- https://www.techopedia.com/definition/34032/deep-q-networks
- https://chatbotsmagazine.com/the-future-with-reinforcement-learning-part-1-762cfcce3638
- https://www.crunchbase.com/organization/kensci#section-overview
- https://www.mygreatlearning.com/blog/reinforcement-learning-in-healthcare/
- https://www.researchgate.net/publication/334388608_Deep_Reinforcement_Learning_in_Financial_Markets
- https://vmayoral.github.io/robots,/ai,/deep/learning,/rl,/reinforcement/learning/2016/07/06/rl-intro/
- https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da
- https://deeplizard.com/learn/video/eMxOGwbdqKY
- https://deeplizard.com/learn/video/my207WNoeyA
- https://emerj.com/ai-sector-overviews/machine-learning-in-pharma-medicine/
- https://towardsdatascience.com/reinforcement-learning-for-real-world-robotics-148c81dbdcff
- https://medium.com/analytics-vidhya/take-a-peek-at-deep-reinforcement-learning-for-nlp-a8e37fbd7640
- https://arxiv.org/abs/1908.08796
- https://emerj.com/ai-sector-overviews/machine-learning-in-pharma-medicine/