Introduction to Reinforcement Learning

Mr. Gouasmia Zakaria

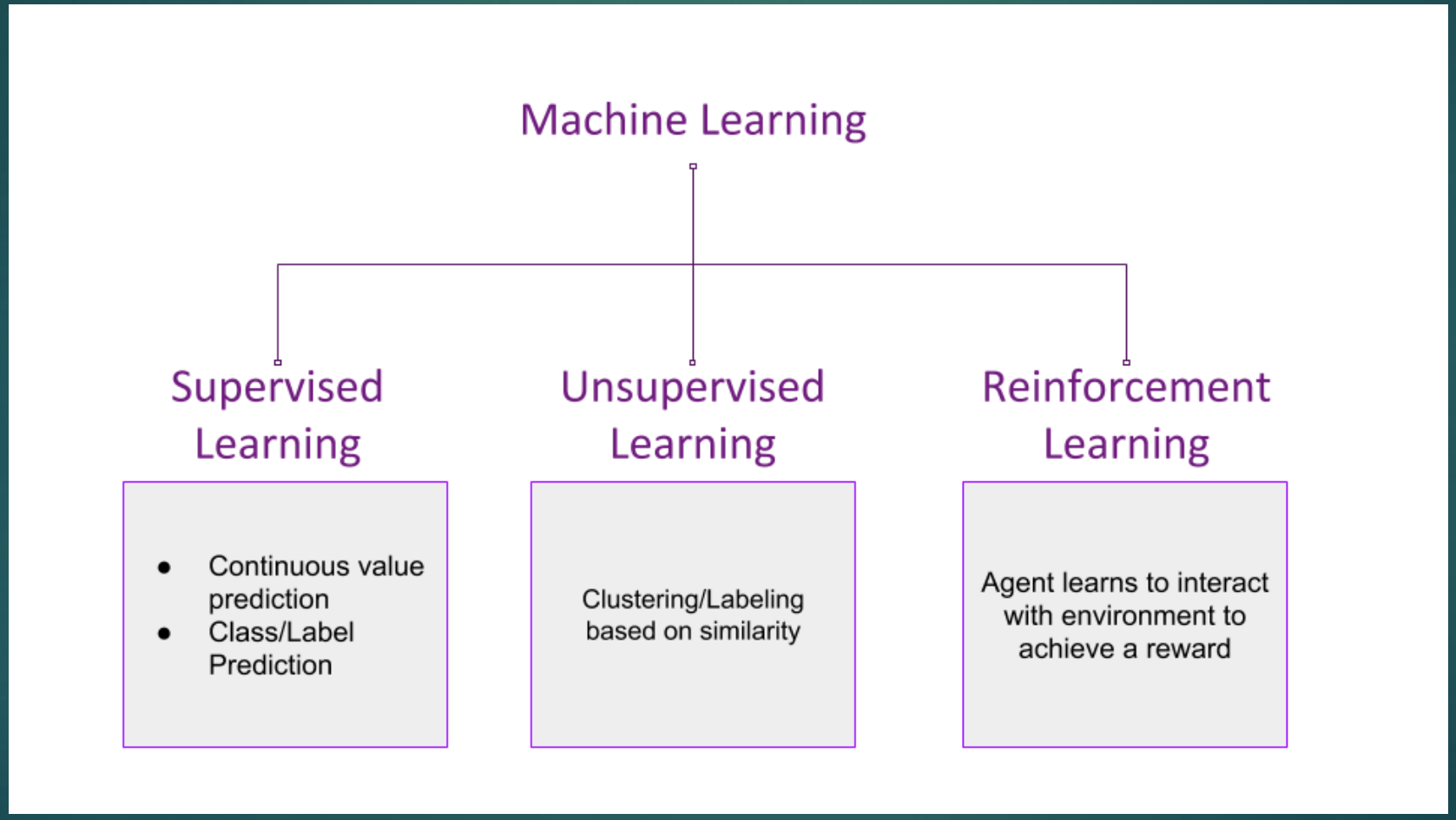
2019/2020

Table of Contents

What is Reinforcement Learning?

- Reinforcement learning in formal terms, is a method of machine learning wherein the software agent learns to perform certain actions in an environment which lead it to maximum reward. It does so by exploration and exploitation of knowledge it learns by repeated trials of maximizing the reward.

Reinforcement Learning vs. the rest



Intuition to Reinforcement Learning

- Imagine you are supposed to cross an unknown field in the middle of a pitch black night without a torch. There can be pits and stones in the field, the position of those are unfamiliar to you. There's a simple rule - if you fall into a hole or hit a rock, you must start again from your initial point.

Intuition to Reinforcement Learning

- You start walking forward blindly, only counting the number of steps you take. After x steps, you fall into a pit. Your reward was x points since you walked that many steps.

Intuition to Reinforcement Learning

- You start again from your initial position, but after x steps, you take a detour either left/right and again move forward. You hit a stone after y steps. This time your reward was y which is greater than x. You decide to take this path again but with more caution.

Intuition to Reinforcement Learning

- When you start again, you make a detour after x steps, another after y steps and manage to fall into another pit after z steps. This time the reward was z points which was greater than y, and you decide that this is a good path to take again.

Intuition to Reinforcement Learning

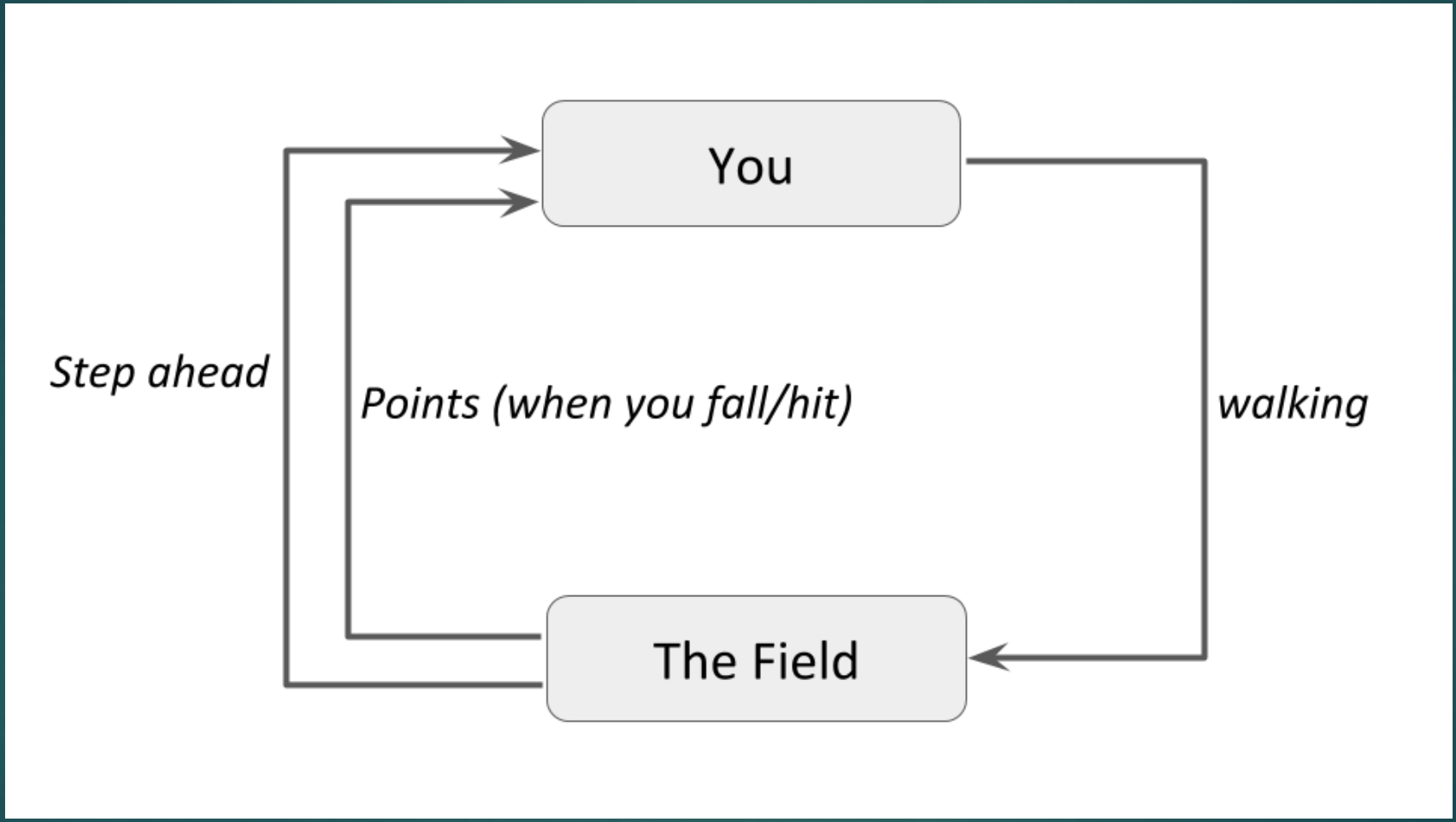
- You restart again, make the detours after x, y and z steps to reach the other side of the field. Thus, you've learned to cross the field without the need of light.

Basic Concept and Terminology

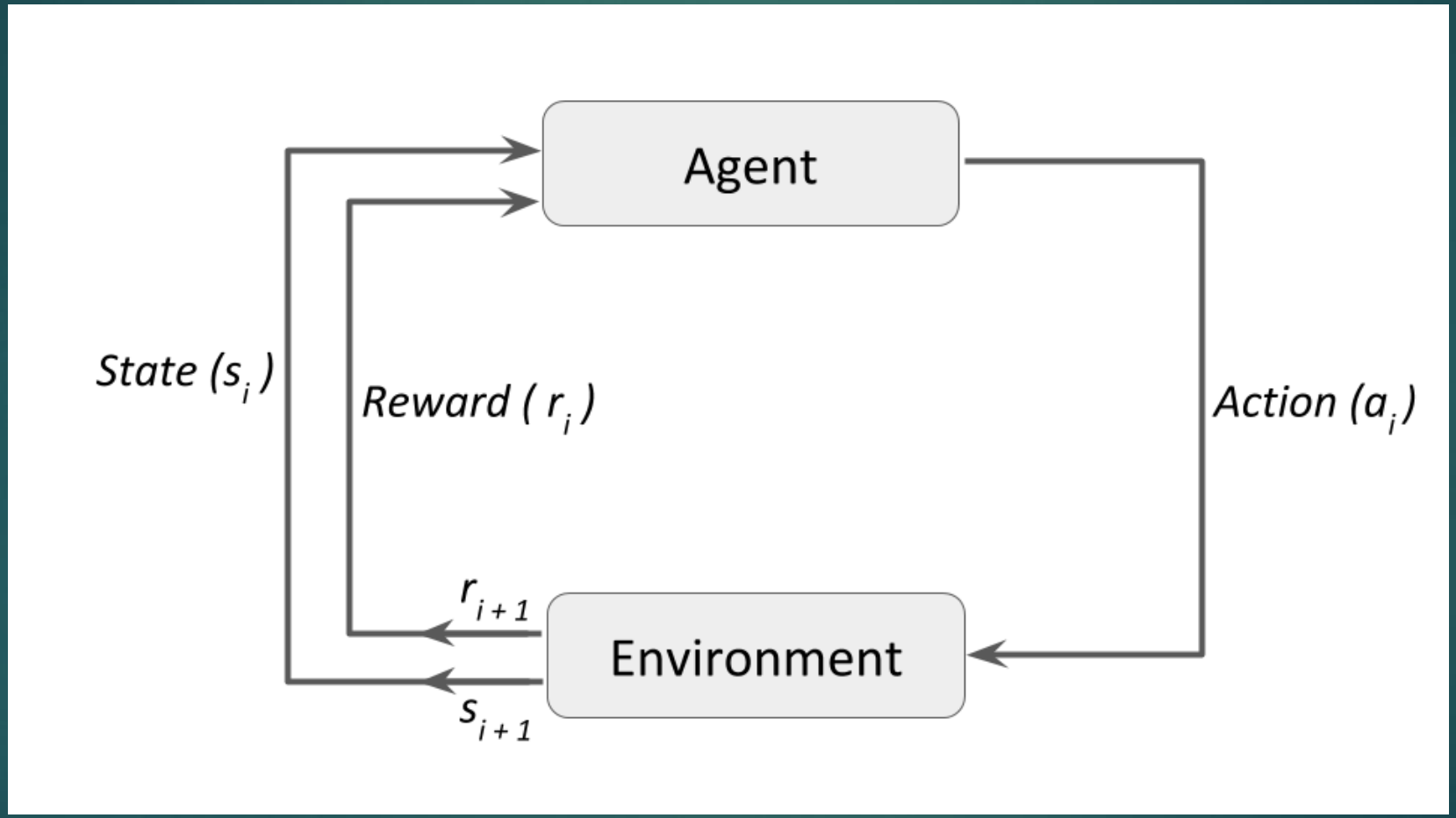
Insight

- In the above example, you are the **agent** who is trying to walk across the field, which is the **environment**. Walking is the **action** the **agent** performs on the **environment**. The distance the **agent** walks acts as the **reward**. The **agent** tries to perform the **action** in such a way that the **reward** maximizes. This is how Reinforcement Learning works in a nutshell.

Basic Concept and Terminology



Basic Concept and Terminology



Terminology

- **Agent:** a hypothetical entity which performs actions in an environment to gain some reward.
- **Action (a):** All the possible moves that the agent can take.
- **Environment (e):** A scenario the agent has to face.
- **State (s):** Current situation returned by the environment.
- **Reward (R):** An immediate return sent back from the environment to evaluate the last action by the agent.
- **Policy (π):** The strategy that the agent employs to determine next action based on the current state.
- **Value (V):** The expected long-term return with discount, as opposed to the short-term reward **R**. **$V_\pi(s)$**, is defined as the expected long-term return of the current state **s** under policy **π**.
- **Q-value or action-value (Q):** Q-value is similar to Value, except that it takes an extra parameter, the current action **a**. **$Q_\pi(s, a)$** refers to the long-term return of the current state **s**, taking action **a** under policy **π**.

How Reinforcement Learning Works

- **Value Based:** in a value-based reinforcement learning method, you try to maximize a value function **V(s)**. As defined in the terminology previously, **Vπ(s)** is the expected long-term return of the current state **s** under policy **π**. Thus, **V(s)** is the value of reward which the agent expects to gain in the future upon starting at that state **s**.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$$

How Reinforcement Learning Works

- **Policy-based:** in a policy-based reinforcement learning method, you try to come up with a policy such that the action performed at each state is optimal to gain maximum reward in the future. Here, no value function is involved. We know that the policy **π** determines the next action **a** at any state **s**. There are two types of policy-based RL methods –
  - **Deterministic:** at any state **s**, the same action **a** is produced by the policy **π**.
  - **Stochastic:** each action has a certain probability, given by the equation below -

$$StochasticPolicy : \pi(a|s) = P[A_t = a|S_t = s]$$

How Reinforcement Learning Works

- **Model-Based:** in this type of reinforcement learning, you create a virtual model for each environment, and the agent learns to perform in that specific environment. Since the model differs for each environment, there is no singular solution or algorithm for this type.

A Simple Implementation

- Reinforcement Learning comes with its own classic example - the Multi-Armed Bandit problem. Never heard? No worries! Here's what it is - assume you're at a casino and in a section with some slot machines. Let's say you're at a section with 10 slot machines in a row and it says "Play for free! Max payout is 10 dollars" Each slot machine is guaranteed to give you a reward between 0 and 10 dollars. Each slot machine has a different average payout, and you have to figure out which one gives the most average reward so that you can maximize your reward in the shortest time possible.

A Simple Implementation

ϵ (epsilon)-greedy algorithm

- One very famous approach to solving reinforcement learning problems is the **ϵ (epsilon)-greedy** algorithm, such that, with a probability **ϵ**, you will choose an action **a** at random (**exploration**), and the rest of the time (probability **1−ϵ**) you will select the best lever based on what you currently know from past plays (**exploitation**). So most of the time you play greedy, but sometimes you take some risks and choose a random lever and see what happens.

ϵ (epsilon)-greedy algorithm

```
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(5)
```

ϵ (epsilon)-greedy algorithm

```
n = 10

arms = np.random.rand(n)

eps = 0.1 #probability of exploration action
```

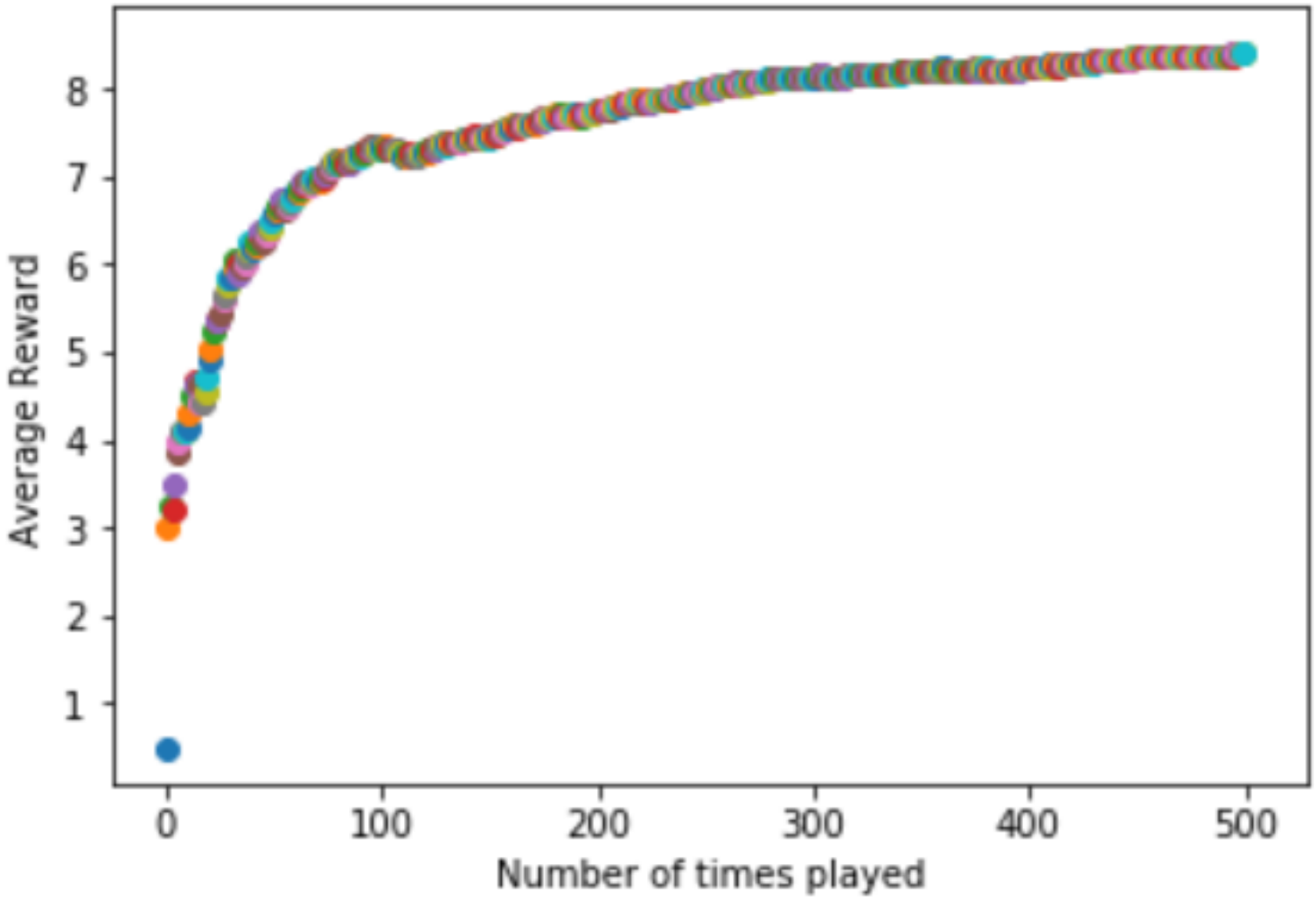
```
def reward(prob):
    reward = 0
    for i in range(10):
        if random.random() < prob:
            reward += 1
    return reward
```

```
def reward(prob):
    reward = 0
    for i in range(10):
        if random.random() < prob:
            reward += 1
    return reward
```

```
#initialize memory array; has 1 row defaulted to random action index
av = np.array([np.random.randint(0,(n+1)), 0]).reshape(1,2) #av = action-value
```

```
#greedy method to select best arm based on memory array
def bestArm(a):
    bestArm = 0 #default to 0
    bestMean = 0
    for u in a:
        avg = np.mean(a[np.where(a[:,0] == u[0])][:, 1]) #calculate mean reward for each
        if bestMean < avg:
            bestMean = avg
            bestArm = u[0]
    return bestArm
```

```
plt.xlabel("Number of times played")
plt.ylabel("Average Reward")
for i in range(500):
    if random.random() > eps: #greedy exploitation action
        choice = bestArm(av)
        thisAV = np.array([[choice, reward(arms[choice])]])
        av = np.concatenate((av, thisAV), axis=0)
    else: #exploration action
        choice = np.where(arms == np.random.choice(arms))[0][0]
        thisAV = np.array([[choice, reward(arms[choice])]]) #choice, reward
        av = np.concatenate((av, thisAV), axis=0) #add to our action-value memory array
    #calculate the mean reward
    runningMean = np.mean(av[:,1])
    plt.scatter(i, runningMean)
```


Average Reward
8
7
6
5
4
3
2
1
0
100
200
300
400
500
Number of times played

Conclusion

- Game Theory and Multi-Agent Interaction - reinforcement learning has been used extensively to enable game playing by software. A recent example would be Google's DeepMind which was able to defeat the world's highest ranked Go player and later, the highest rated Chess program Komodo.

- Robotics - robots have often relied upon reinforcement learning to perform better in the environment they are presented with. Reinforcement learning comes with the benefit of being a play and forget solution for robots which may have to face unknown or continually changing environments. One well-known example is the Learning Robots by Google X project.

Conclusion

- Vehicle navigation - vehicles learn to navigate the track better as they make re-runs on the track. A proof of concept is presented in this video while a real-life world example was presented at the O'Reilly AI Conference in this video.
- Industrial Logistics - industry tasks are often automated with the help of reinforcement learning. The software agent facilitating it gets better at its task as time passes. BonsAI is a startup working to bring such AI to the industries.

References

- Sutton, Richard S. and Barto, Andrew G., Reinforcement Learning: An Introduction, MIT Press, 1998
- The Reinforcement Learning Repository, University of Massachusetts, Amherst
- Wikipedia article on Reinforcement Learning
- A Beginners Guide to Deep Reinforcement Learning

Links

- A Glossary of terms in Reinforcement Learning
- Bibliography on Reinforcement Learning
- David J. Finton's Reinforcement Learning Page
- Stanford University Andrew Ng Lecture on Reinforcement Learning