Project 4: Reinforcement Learning

Sambo Dutta
University at Buffalo
UBIT Name: sambodut
UB PERSON NUMBER: 50318667
sambodut@buffalo.edu

Abstract

Apart from Supervised learning and Unsupervised learning, Reinforcement learning is the other major area of machine learning which is based on maximization of cumulative reward. There is a basic environmental setup and the goal is to achieve the maximum reward is the process of taking actions.In this project we have implemented the Q-learning, policy determination and the training process of the given environment.

1 Introduction

Reinforcement learning is an area of Machine Learning which is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience.

Reinforcement learning is all about making decisions sequentially. In simple words we can say that the output depends on the state of the current input and the next input depends on the output of the previous input. The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning, an artificial intelligence faces a game-like situation. The computer employs trial and error to come up with a solution to the problem. To get the machine to do what the programmer wants, the artificial intelligence gets either rewards or penalties for the actions it performs. Its goal is to maximize the total reward.

Although the designer sets the reward policy–that is, the rules of the game–he gives the model no hints or suggestions for how to solve the game. It's up to the model to figure out how to perform the task to maximize the reward, starting from totally random trials and finishing with sophisticated tactics and superhuman skills. By leveraging the power of search and many trials, reinforcement learning is currently the most effective way to hint machine's creativity. In contrast to human beings, artificial intelligence can gather experience from thousands of parallel game plays if a reinforcement learning algorithm is run on a sufficiently powerful computer infrastructure.

Reinforcement learning is all about an agent who takes actions; for example, a drone making a delivery, or Super Mario navigating a video game. The algorithm is the agent. Actions is the set of all possible moves the agent can make. A state is a concrete and immediate situation in which the agent finds itself; i.e. a specific place and moment, an instantaneous configuration that puts the agent in relation to other significant things such as tools, obstacles, enemies or prizes. It can the current situation returned by the environment, or any future situation.A feedback or reward is given by the success or failure of an agent's actions in a given state. With successful implementation an agent can outperform humans in reaching the goal state.

2 Algorithm

2.1 Markov Decision Process

The RL process can be viewed as a Markov Decision Process with the following states

1. **A finite set of states**: These are the possible positions of our agent within the environment.
2. **A set of actions available in each state**:This involves all the actions, the agent can take in the environment.
3. **Transitions between states**:The transition probability table.
4. **Rewards associated with each transition**: There might not be rewards in all actions, but with each step we get nearer to the final goal.

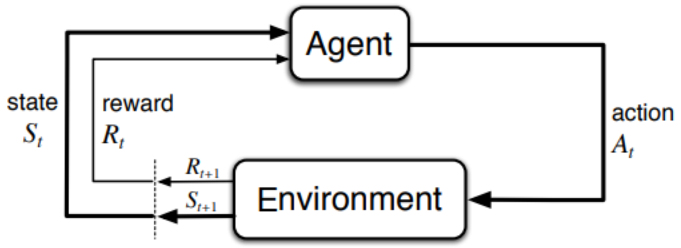

Figure 1: Markov Decision Process

2.2 Environment

The world through which the agent moves, and which responds to the agent. The environment takes the agent's current state and action as input, and returns as output the agent's reward and its next state. If you are the agent, the environment could be the laws of physics and the rules of society that process your actions and determine the consequences of them. Reinforcement learning environments can take on many different forms, including physical simulations, video games, stock market simulations, etc. The reinforcement learning community (and, specifically, OpenAI) has developed a standard of how such environments should be designed, and the library which facilitates this is OpenAI's Gym. The environment we provide is a basic deterministic n*n grid-world environment (the initial state for an 4*4 grid-world is shown in Figure 3) where the agent (shown as the green square) has to reach the goal (shown as the yellow square) in the least amount of time steps possible. The environment's state space will be described as an n*n matrix with real values on the interval [0, 1] to designate different features and their positions. The agent will work within an action space consisting of four actions: up, down, left, right. At each time step, the agent will take one action and move in the direction described by the action. The agent will receive a reward of +1 for moving closer to the goal and -1 for moving away or remaining the same distance from the goal.

2.3 Q-Learning

Q-learning is a technique that evaluates which action to take based on an action-value function that determines the value of being in a certain state and taking a certain action at that state.

We have a function Q that takes as an input one state and one action and returns the expected reward of that action (and all subsequent actions) at that state. Before we explore the environment, Q gives the same (arbitrary) fixed value. But then, as we explore the environment more, Q gives us a better and better approximation of the value of an action a at a state s. We update our function Q as we go.

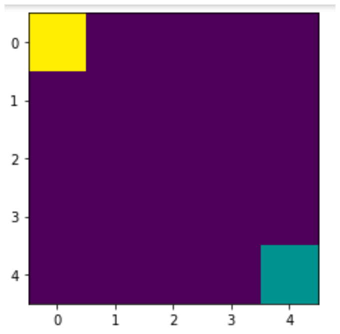

Figure 2: The initial state of the agent in the environment

The equation is given below which shows how we update the value of Q based on the reward we get from our environment:

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

Q shows us the full sum of rewards from choosing action Q and all the optimal actions afterward. When we take action a_t in state s_t, we update our value of Q(s_t,a_t) by adding a term to it. This term contains:
Learning rate alpha: It is how quickly or rather aggressively we want to update the value. When alpha is close to 0, we're not updating very aggressively. When alpha is close to 1, we're simply replacing the old value with the updated value.
The reward is the reward we got by taking action at at state s_t. So we're adding this reward to our old estimate.
We're also adding the estimated future reward, which is the maximum achievable reward Q for all available actions at x_{t+1}.
Finally, we subtract the old value of Q to make sure that we're only incriminating or decrementing by the difference in the estimate (multiplied by alpha of course).
The hyperparameters gamma and alpha and discussed in depth in the following sections.

2.3.1 Hyperparameters

1. **Learning Rate**: This was discussed in thr previous section, here in our experiment we have set the leaning rate to be 0.1.

2. **Discounting factor**: A discount factor between 0 and 1. This quantifies the difference in importance between immediate rewards and future rewards. For example, if is .9, and there's a reward of 5 after 3 steps, the present value of that reward is .9*5.
In our experiment we have set the discounting factor to 0.9.

3. **Epsilon**: This brings up the exploration/exploitation trade-off. One simple strategy for exploration would be for the agent to take the best known action most of the time (say, 80% of the time), but occasionally explore a new, randomly selected direction even though it might be walking away from known reward. This strategy is called the epsilon-greedy strategy, where epsilon is the percent of the time that the agent takes a randomly selected action rather than taking the action that is most likely to maximize reward given what it knows so far. We usually start with a lot of exploration (i.e. a higher value for epsilon). Over time,the agent learns more about the environment and the actions which results in the most long-term rewards.After sometime we can reduce the epsilon steadily it would make sense to steadily reduce epsilon to 10% or even lower as it settles into exploiting what it knows.

In our experiment we have reduced the epsilon each time by a factor of 0.99. However note that on reaching the minimal value of 0.1 it is not reduced further.

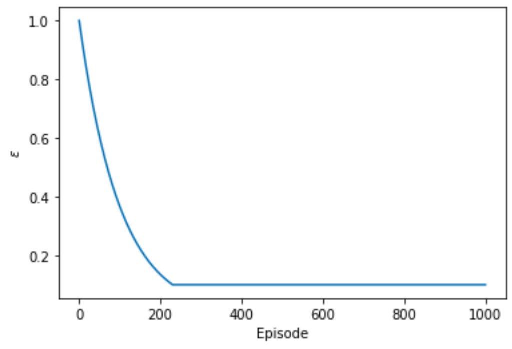

Figure 3: Decay of epsilon

2.4 Policy Learning

In the Q-learning approach, we learned a value function that estimated the value of each state-action pair. Policy learning is a more straightforward alternative in which we learn a policy function, π, which is a direct map from each state to the best corresponding action at that state.
In our experiment we have used the following policy function:

$$\pi(s_t) = \underset{a \in A}{\operatorname{argmax}}\, Q_\theta(s_t, a)$$

i.e. when not deciding the action randomly, the agent will pick the action that will give the highest reward.

3 Results

For our experiment we have compared three variants of the agents. First we have taken a random agent which moves in the environment randomly and has no guarantee whether it will reach the goal or not. Then we compared with the heuristic agent, here we have specif ed the path that the agent has to take. to reach the goal. There is no randomness in the behaviour of the heuristic agent. And at last we have used the Q learning agent which is combination of the above two agents. It has both the natures of the random agent and the heuristic agent. Unlike the heuristic agent the Q-learning agent shows a random behaviour and there is no need to specify the path. Additionally this Q-learning agent though has the random nature of the random learning agent,it has guarantee that it would reach the ultimate goal after a few episodes. This whole process of Q-agent playing game is depicted in Figure 5.

In the below Figure 4 we can see how the Q-learning agent learns with every episode and updates its Q-table, achieving more reward every time. The maximum value reached is however 8 because there are 8 moves the agent makes from the initial setup to the goal.

4 Conclusion

From the experiments conducted above we see that Q-Learning agent which both explores and exploits the knowledge gained from experience performs the best. It manages to reach the ultimate goal state after a few episodes and combines both the positive natures of the two previous agents-randomness and reaching the final goal.

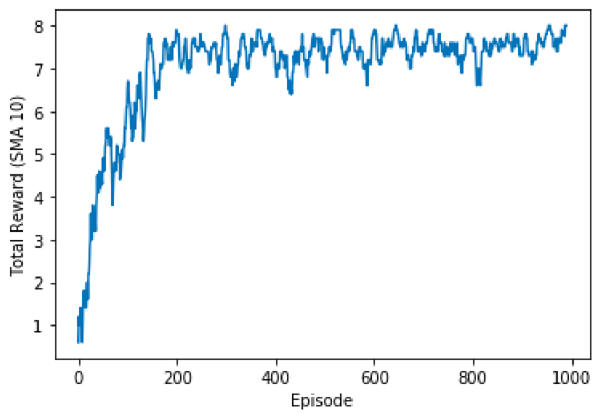

Figure 4: Rewards vs Episodes

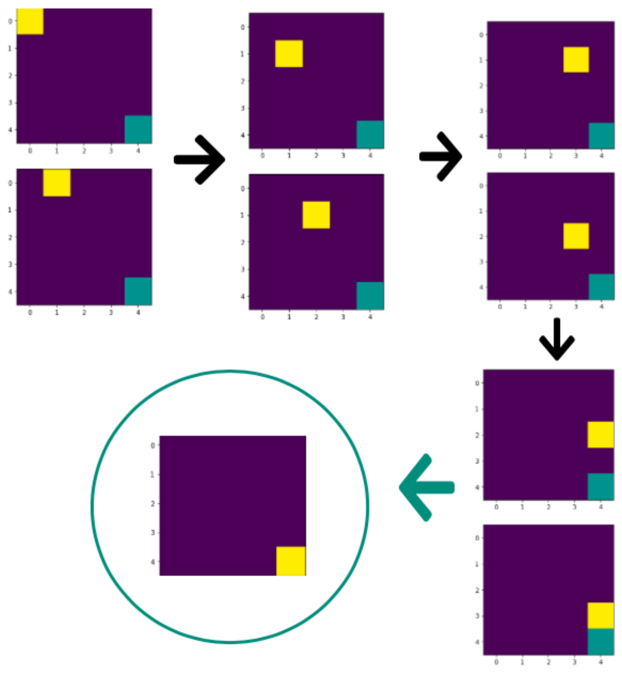

Figure 5: Q-Agent

5 Reference

[1] https://skymind.ai/wiki/deep-reinforcement-learning
[2]https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265