Reinforcement Learning Assignment-5
Function Approximation, Policy Gradients and Deep Reinforcement Learning

Utkarsh Prakash
180030042

April 14, 2022

1 OpenAI Gym Environments Considered

1.1 Mountain Car

A car is on a one-dimensional track, positioned between two "mountains". The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum.

The state of the car is decided by the position of the car along the horizontal axis and the velocity. At the beginning of the episode, the car starts from the bottom of the hills (valley). The episode ends when the car reaches the goal or after 200 steps (which ever is earlier). The action space consists of 3 actions: {push left, push right, do nothing}. A negative reward of -1 is applied at each timestep. The objective of the car is to reach the top of the hill on the right as early as possible, because at each timestep it will be rewarded negatively.
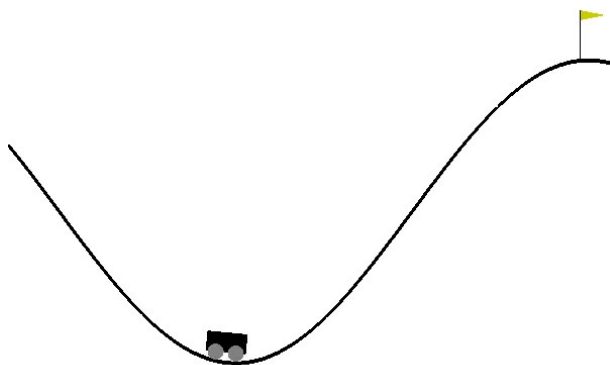

Figure 1: MountainCar. For more details click here

1.2 CartPole

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the

pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center or the length of the episode is atleast 200.
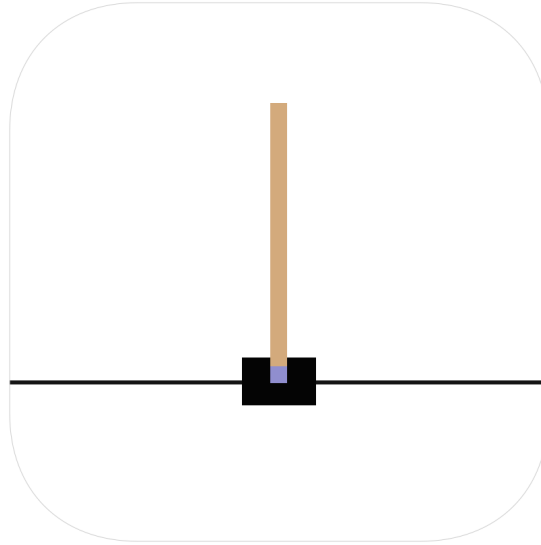

Figure 2: CartPole. For more details click here

The state of the cart is decided by it's position (x), velocity (x_dot), pole angle (theta) and pole angular velocity (theta_dot). The action space consists of 2 actions: {push left, push right}.

2 Linear Function Approximation

2.1 Mountain Car

2.1.1 Tile Coding

We consider a grid of 14 x 14 tiling i.e. each tiling consists of 196 tiles and we have $n = 7$ such tilings. The offsets for these tilings were: [(0, 0), (0.018, 0.004), (0.036, 0.008), (0.055, 0.013), (0.073, 0.017), (0.092, 0.021), (0.110, 0.025)], where the first and second dimension represents the position and velocity of the car respectively. We run Q-Learning and SARSA algorithm for $5,000$ episodes and average the results over 2 runs. The ϵ or the exploration factor in the ϵ-greedy policy is chosen to be 0.8 initially and is decreased by a factor of 0.99 after every epsiode. This ensures that each of the state is explored infinitely often. The learning rate or the step-size (α) and discount factor (γ) are chosen to $0.3/n$, where n is the number of tiling and 0.99 respectively. We use a constant step-size. The comparison between the average episodic rewards for both the algorithms is shown in Fig. 3a.

2.1.2 Radial Basis Function Coding

We represent the state space using 22 x 22 equally spaced RBF centers with a width of 0.000486 which is the square of half of the distance between two adjacent centers. We run Q-Learning and

SARSA algorithms for 2,000 episodes and average the results over 2 runs. The learning rate or the step-size (α) is set to 0.05. Rest of the setting is similar to tile coding.

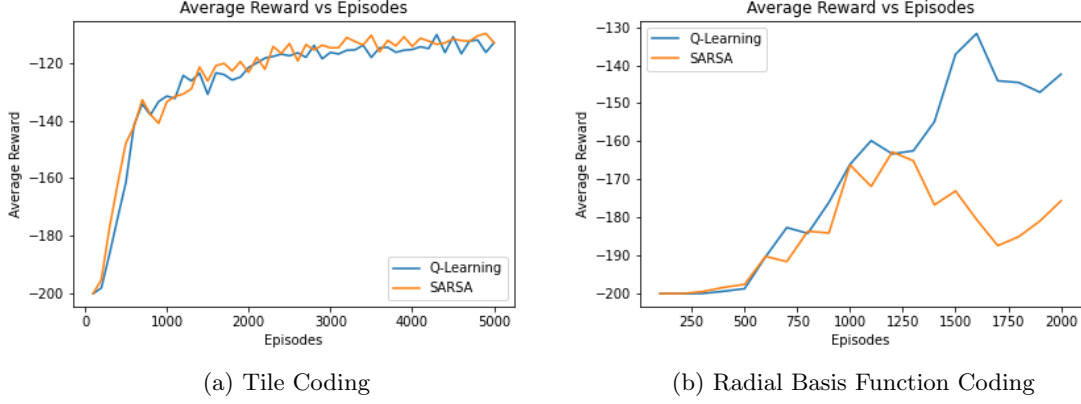(a) Tile Coding   (b) Radial Basis Function Coding

Figure 3: Average episodic reward as a function of episodes for Mountain Car.

Conclusion:

Tile coding seems to be performing better than the Radial Basis Function coding as the former achieves a higher average episodic rewards. Moreover, the learning is also much stable while using Tile Coding than the RBF coding. The computational cost of RBF coding is significantly higher than that of Tile coding as the former requires computation of the costly RBF function as opposed to one-hot calculation in the latter.

2.2   CartPole

2.2.1   Tile Coding

We consider a hyper-grid of 22 x 22 x 22 x 22 tiling i.e. each tiling consists of 22^4 tiles and we have $n = 4$ such tilings. The offsets for these tilings were: [(0, 0, 0, 0), (0.07, 0.23, 0.03, 0.55), (0.14, 0.48, 0.06, 1.11), (0.20, 0.71, 0.08, 1.67)], where the dimensions represent position (x), velocity (x_dot), pole angle (theta) and pole angular velocity (theta_dot) of the cart respectively. We run Q-Learning and SARSA algorithm for 1,000 episodes and average the results over 2 runs. The ϵ or the exploration factor in the ϵ-greedy policy is chosen to be 0.8 initially and is decreased by a factor of 0.99 after every epsiode. This ensures that each of the state is explored infinitely often. The learning rate or the step-size (α) and discount factor (γ) are chosen to 0.1/n, where n is the number of tiling and 0.99 respectively. We use a constant step-size. The comparison between the average episodic rewards for both the algorithms is shown in Fig. 4a.

3

2.2.2 Radial Basis Function Coding
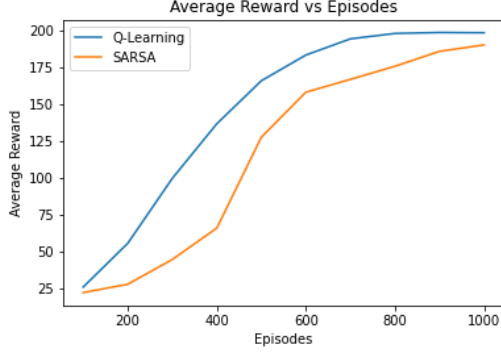
(a) Tile Coding

(b) Radial Basis Function Coding

Figure 4: Average episodic reward as a function of episodes for CartPole.

3 REINFORCE Algorithm

In this section we compare the performance of different versions of the REINFORCE algorithms which differ in their policy gradient calculations, as described below:

$$\nabla \rho(\theta) = \mathbb{E}_\tau \left[\left(\sum_{t=0}^{T} \nabla_\theta log\pi(a_t|s_t) \right) \left(\sum_{t=0}^{T} \gamma^t R_t \right) \right] \tag{1}$$

$$\nabla \rho(\theta) = \mathbb{E}_\tau \left[\left(\sum_{t=0}^{T} \nabla_\theta log\pi(a_t|s_t) \left(\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \right) \right) \right] \tag{2}$$

Let's call the version of the algorithm described by 1 and 2 as **REINFORCE** and **REIN-FORCE_var** respectively. We benchmark the performance on standard Mountain Car and Cart-Pole tasks.

3.1 Mountain Car

3.2 CartPole

We parameterize the policy using a neural network consisting of 4, 16 and 2 units in the input, hidden and output layer. We use ReLU and softmax activation for the first and second layer respectively. We run each of the algorithm for $5,000$ episodes and average the results over 2 runs. The learning rate or the step-size (α) used is fixed and equals 0.01. The discount factor (γ) is chosen to be 0.99.
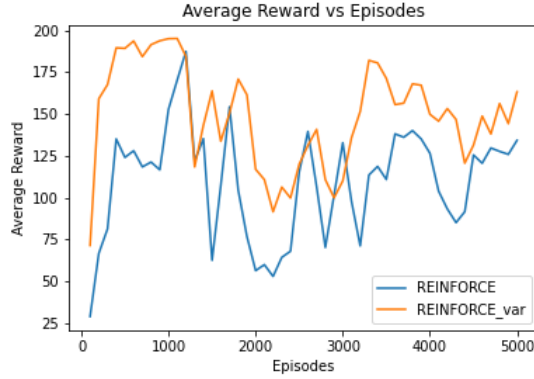
Figure 5: Average episodic reward as a function of episodes for CartPole problem using different versions of REINFORCE algorithm.

Conclusion:
The REINFORCE_var version seems to be performing better than the vanilla version of the RE-INFORCE algorithm. This can be attributed to the fact that the variance of the policy gradient estimate is lower in the REINFORCE_var version than the vanilla one, which makes the learning more stable. If we would averaged over a lot of multiple runs, the effect of lower variance would have been more pronounced.

4 Deep Reinforcement Learning

In this section we compare the performance of Advantage Actor Critic method (A2C) and Deep Q-Network (DQN) on the standard CartPole task.

4.1 Advantage Actor Critic (A2C)

The Actor network comprises of 1 input, 2 hidden and 1 output layers. The input, first hidden, second hidden and output layer comprises of 4, 64, 64 and 2 units respectively. We use ReLU activation function apart from the last layer where we use softmax activation. The Critic network is same as the Actor network apart from the output layer which consists of 1 output unit and no activation is used. We train both the networks for $5,000$ episodes using Adam optimizer with a learning rate of 0.001 and set rest of the hyperparameters to their default values. The discount factor (γ) was set to 0.99.

4.2 Deep Q-Network (DQN)

The DQN network consists of 1 input, 2 hidden and 1 output layers. The input, first hidden, second hidden and output layer comprises of 4, 64, 64 and 2 units respectively. We use ReLU activation function apart from the last layer where no activation function is used. We train the network for $5,000$ episodes using Adam optimizer with a learning rate of 0.0005 and rest of hyperparameters were set to their default values. The ϵ or the exploration factor in the ϵ-greedy policy is chosen to be 1.0 initially and is decreased by a factor of 0.995 after every epsiode but we ensure that it never falls below 0.01. This ensures that each of the state is explored infinitely often. The discount factor (γ) was set to 0.99.
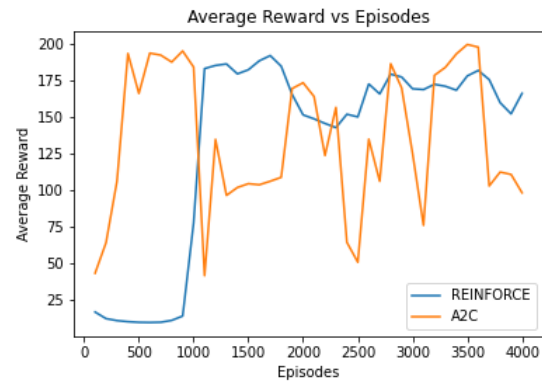
Figure 6: Average episodic reward as a function of episodes for CartPole problem using different Deep Reinforcement Learning algorithms.