Reinforcement Learning Assignment-5
Function Approximation, Policy Gradients and Actor-Critic Methods

Utkarsh Prakash
180030042

April 12, 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.

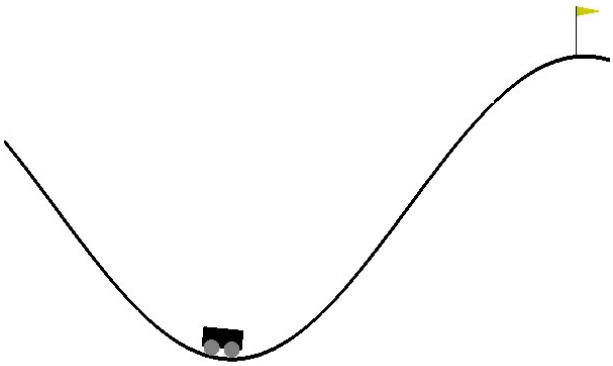

Figure 1: MountainCar. For more details click here

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 each of the algorithm for $5,000$ episodes and average the results over 10 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.

2.1.2 Radial Basis Function Coding

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 each of the algorithm for $5,000$ episodes and average the results over 10 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.

2.2.2 Radial Basis Function Coding

3 REINFORCE Algorithm

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. We compare the performance of versions of the REINFORCE algorithms which differ in their policy gradient calculation, 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 **REINFORCE_var** respectively.