COMP-579: Reinforcement Learning - Assignment 3

Posted Thursday, February 22, 2024
Due Tuesday, March 19, 2024

1. **Value-based methods with linear function approximation [40 points]**

   Implement Q-learning and Exptected SARSA for both MountainCar-v0 [1] and CartPole-v1[2] environments from the Gym suite using the following guidelines:

   - Use a linear function approximation for Q, that is, if $\mathbf{x}$ is a vector representing the state and $\mathbf{a}$ is the action vector, use $Q(\mathbf{x}, a) = \sum_{i=1}^{d} \theta_{a,i} x_i$, where θ are the parameters of the Q-fct you need to learn, d is the dimension of x and $a \in \{1, \ldots, m\}$ is a discrete action.
   - Discretise the state space for both the environments using an appropriate tilecoding (see section 9.5.4 of the RL book `http://incompleteideas.net/book/RLbook2020.pdf` for how to do tilecoding). (It is generally recommended to use $<= 10$ tiles (bins) per state variable, but you are free to choose more tiles if it results in performance improvements). It easier to imagine tiling each dimension of the state-space independently, in that case, one tiling of s_2, for example, is just one way to make bins out of s_2. So suppose the state-space is 2D so we have the state is $\mathbf{s} = (s_1, s_2)$. Suppose we have 2 tilings of 5 tiles (bins) each per dimension of s, idea is to convert the state representation from $\mathbf{s}$ to the 2*2*5=20dim vector

     $$\mathbf{x} = (x_{1,1,1}, x_{1,1,2}, \ldots, x_{1,1,5}, x_{1,2,1}, \ldots, x_{1,2,5}, x_{2,1,1}, x_{2,1,2}, \ldots, x_{2,1,5}, x_{2,2,1}, \ldots, x_{2,2,5}),$$

     which contains only 1s and 0s where $x_{i,j,k} = 1$ iff s_i is in tile k of the j^{th} tiling (of the i^{th} state dimension), and 0 otherwise.
   - Initalise the parameters for the value function uniformly between -0.001 and 0.001.
   - Use an ϵ- greedy policy with three choices of ϵ and step-size parameters $1/4, 1/8, 1/16$. and run 50 learning trials with different initialisations for Q, each having 1000 episodes, for each configuration. (That means 3 configs * 50 runs * 1000 episodes).
   - Plot the average performance of the policy on the Y-axis and the number of episodes on the X-axis. The plots should also include the interquantile range of the 50 independent runs. Note that you are expected to plot 9 results corresponding to each ϵ and step-size parameters for both the environments, and document your findings in a separate pdf along with the results. Explain why one algorithm performs better than the other or why a particular configuration results in better performance.
   - **Implement all the methods without using any automatic differentiation package**. It is highly recommended that you undertake the development of software independently. Furthermore, it is essential to appropriately cite any resources or materials sourced from the internet in your work.

[1] `https://gymnasium.farama.org/environments/classic_control/mountain_car/`
[2] `https://gymnasium.farama.org/environments/classic_control/cart_pole/`

2. **Policy Gradient Theorem [20 points]**

   Given an MDP with a state space $\mathcal{S}$, Discrete action space $\mathcal{A} = [a_1, a_2, a_3]$, Reward function $\mathcal{R}$, discount factor γ, and a policy with the follwing functional representation:

$$\pi(a_1|s) = \frac{\exp(z(s, a_1))}{\sum_{a \in \mathcal{A}} \exp(z(s, a))}. \tag{1}$$

   Use the policy gradient theorem to show the follwing:

$$\nabla_z J(\pi) = d^\pi(s)\pi(a|s)A^\pi(s, a), \tag{2}$$

   where d^π is the steady state distribution of the Markov chain induced by π and $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$

3. **Policy-based methods with linear function approximation [40 points]**

   Implement REINFORCE and Actor-Critic method for both the MountainCar-v0 and CartPole-v1 environemnts.

$$\pi(a_i|s) = \frac{\exp(z(s, a_i)/T)}{\sum_{a \in \mathcal{A}} \exp(z(s, a)/T)}. \tag{3}$$

   - Implement a Boltzman's Policy as in eq. (3) and use a linear approximation for z. That is $z(\mathbf{x}, a) = \sum_{i=1}^{d} \theta_i^a x_i$, where θ are the parameters of z you need to learn, d is the dimension of x (the state space representation) and $a \in \{1, \ldots, m\}$ is a discrete action. Like in part 1, x is a tilecoding of the state space s.
   - Similar to Value based methods, use appropriate initialisation of the policy parameters.
   - Implement a Boltzman's Policy and run 50 learning trials with different initialisations for Q, each having 1000 episodes for the following two configurations. 1. A fixed temperature $T > 0$ (of your choice) and 2. A decreasing temperature T. (50 runs * 1000 episodes * 2 configs) You are free to chose your own stepsizes for these implementations.
   - Plot the average performance of the policy on the Y-axis and the number of episodes on the X-axis. The plots should also include the interquantile range of the 50 independent runs. Note that you are expected to plot 4 results for 2 configurations per environment and document your findings in a separate pdf along with the results. Explain why one algorithm performs better than the other or why a particular configuration results in better performance.
   - Similar to value based methods, you have to implement all the code without using any automatic differentiation package, and you are required to cite any material sourced from the internet.