CS698R: Deep Reinforcement Learning

Assignment #1

Name: Naman Gupta
Roll NO.: 190527
Github Link: `www.tinyurl.com`

Solution to Problem 1: Multi-armed Bandits

Pseudo random number generators and Seeding: Throughout the assignment, manual seeds in numpy for random number generators are set by passing rng objects (np.random.default_rng(seed)). This is the recommended way of setting manual seeds (NEP 19) since it is easier to track which seed is used where.

1. The `test_bernoulli_env` function takes α, β as inputs and generates `maxSteps` episodes of the bernoulli bandit. Tests are carried out using this function with $(\alpha, \beta) = (0, 0), (1, 0), (0, 1), (1, 1), (0.5, 0.5)$ and `maxSteps = 10`. The episodes are tabulated below –

Episode	Action	$R(0,0)$	$R(1,0)$	$R(0,1)$	$R(1,1)$	$R(0.5,0.5)$
1	0	0	1	0	1	0
2	1	0	0	1	1	1
3	1	0	0	1	1	1
4	1	0	0	1	1	1
5	0	0	1	0	1	1
6	1	0	0	1	1	0
7	0	0	1	0	1	0
8	0	0	1	0	1	0
9	1	0	0	1	1	0
10	0	0	1	0	1	1

The actions are same for all settings of (α, β) since the same seed is used. $R(\alpha, \beta)$ denotes the reward received when the bernoulli environment is initialized with (α, β) and a corresponding action is taken by the agent.
$(\alpha, \beta) = (0, 0)$ – The agent has 0 probability of getting reward as 1. Hence it is expected that in all episodes whatever be the action the reward would be 0. The observed values of $R(0, 0)$ follow our expectation.
$(\alpha, \beta) = (1, 0)$ – The agent always gets a reward of 1 on taking action 0 since $\alpha = 1$, a reward of 0 on taking action 1 since $\beta = 0$. The observed values of $R(1, 0)$ follow this expectation.
$(\alpha, \beta) = (0, 1)$ – The agent always gets a reward of 0 on taking action 0 since $\alpha = 0$, a reward of 1 on taking action 1 since $\beta = 1$. The observed values of $R(0, 1)$ follow this expectation.
$(\alpha, \beta) = (1, 1)$ – The agent always gets a reward of 1 whatever be the action. The observed values of $R(1, 1)$ follow this expectation.
$(\alpha, \beta) = (0.5, 0.5)$ – The agent gets a reward 1 with probability 0.5 whatever be the action. The expected rewards are $E[R_0] = 0.5, E[R_1] = 0.5$. Although it is not always guaranteed with that we get reward 1 `maxSteps`/2 times when `maxSteps` is small. The observed values indicate that we get reward 1 and 0 equal number of times, which follows our expectation.

The above test cases show that the implementation of the environment is correct. The `test_bernoulli_env` function can also be used with a more general value of (α, β) with `maxSteps` set to a large value, then the true expected reward and the calculated expected rewards can be compared. On testing with $(\alpha, \beta) = (0.4, 0.3)$ with `maxSteps=1000` the calculated expected reward is $(E[R_0], E[R_1]) = (0.41, 0.29)$ which is close to the true value $(0.4, 0.3)$ again showing that the implementation is correct for a general (α, β).

2. Similar to the previous part the `test_gaussian_env` function takes σ as input and generates `maxSteps` episodes of the gaussian bandit. It samples the optimal action value function $q_*(k) \sim \mathcal{N}(\mu = 0, \sigma^2) = 1$ and then samples rewards as $R_k \sim \mathcal{N}(\mu = q_*(k), \sigma^2 = \sigma^2)$.

   For $\sigma = 0$ as input, it is expected that the optimal action value and the reward is same for each episode. We verify this by running for `maxSteps=30` and taking the mean of the rewards. Note that the mean here doesn't effect because `maxSteps` is very small, each action is expected to be taken only thrice. We obtain
   $q_* = [0.35, 0.82, 0.33, -1.30, 0.91, 0.45, -0.54, 0.58, 0.36, 0.29]$
   $q = [0.35, 0.82, 0.33, -1.30, 0.91, 0.45, -0.54, 0.58, 0.36, 0.29]$
   $\sigma = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
   which follows our expectation.

   We further test with $\sigma = 0.5$ and $\sigma = 1$ with `maxSteps=1000`. An important observation here is that it takes very large number of episodes for the mean and standard deviation of rewards to converge to the true value. Even with `maxSteps=100000` we observe an error $> 1\%$ in some of the actions.

3. (a) In the Pure Exploitation (Greedy) Strategy, the agent selects the action having maximum expected action value till now. In the case of Bernoulli Bandit, the rewards are always non negative, hence the agent sticks to one specific action in all episodes. The `np.argmax` function will return the lower index when all multiple indices are maximum, so the agent always selects action 0. For $(\alpha, \beta) = (0.5, 0.5)$ the expected action value function after large number of episodes is hence $(0.5, 0)$. The observed actions and rewards are as follows –

episode	a0	a1	action	reward
0	0	0		
1	0	0	0	0
2	0	0	0	0
3	0.33	0	0	1
4	0.25	0	0	0
5	0.4	0	0	1
6	0.5	0	0	1
7	0.43	0	0	0
8	0.5	0	0	1
9	0.44	0	0	0
10	0.5	0	0	1

   The episode 0 corresponds to the initial q, the action and reward should be ignored here. A reward 1 is obtained in the third episode, the count of action 0 at this point is 3, hence $q(0) = 1/3$. In the next episode 0 is received, which makes $q(0) = 1/4$ and so on. By the 10th episode $q(0) = 5/10 = 0.5$. It is important to note that even if we got $q(0) = 0.5$ in the end it is not guaranteed for small number of episodes. This value will converge as the number of episodes increase.
   As expected $q(1)$ remains 0 throughout the episodes.

   (b) In the Pure Exploration Strategy the agent randomly selects an action in each episode. For $(\alpha, \beta) = (0.5, 0.5)$ the expected action value function is $(0.5, 0.5)$ given large number of episodes. Infact, for any general (α, β) the expected action value function is guaranteed to converge to (α, β). The observed actions and rewards are as follows –

episode	a0	a1	action	reward
0	0	0		
1	0	0	1	0
2	0	0	1	0
3	0	0.33	1	1
4	0	0.33	0	0
5	0.5	0.33	0	1
6	0.66	0.33	0	1
7	0.5	0.33	0	0
8	0.6	0.33	0	1
9	0.5	0.33	0	0
10	0.5	0.5	1	1

A reward 1 is obtained in the third episode, the count of action 1 at this point is 3, hence $q(1) = 1/3$. A reward 1 is obtained in the fifth episode, the count of action 0 at this point is 2, hence $q(0) = 1/2$. By the 10th episode the count of action 0 is 6 and the total reward is 3, the count of action 1 is 4 and the total reward is 2 giving the action value as $(0.5, 0.5)$.

(c) In the ϵ-Greedy Strategy the agent explores with probability ϵ and exploits with probability $1 - \epsilon$. The implementation can be verified at the extreme values of ϵ that is, 0 and 1 in which case the strategy reduces to part (a) and (b) respectively. A different setting of $(\alpha, \beta) = (0.2, 0.8)$ is used here, the observed actions and rewards are as follows –

episode	a0	a1	action	reward
0	0	0		
1	0	0	0	0
2	0	0	0	0
3	0.33	0	0	0
4	0.25	0	0	1
5	0.2	0	0	0
6	0.16	0	0	0
7	0.14	0	0	0
8	0.125	0	0	0
9	0.11	0	0	0
10	0.2	0	0	1

Table 1: $\epsilon = 0$

$\epsilon = 0$ is pure greedy, hence $q(1)$ is expected to be 0 throughout, $q(0)$ is expected to converge to α. The observation follows this expectation.
$\epsilon = 1$ is pure exploratory, hence q is expected to converge to (α, β). The observation follows this expectation.

(d) In the decaying ϵ-Greedy strategy the agent explores more in the initial episodes, exploits more in the later episodes. This is implemented by decaying ϵ. The observed action, rewards for $(\alpha, \beta) = (0.2, 0.8)$ with decaying ϵ are as follows –

episode	a0	a1	action	reward	epsilon
0	0.000000	0.000000			
1	0.000000	1.000000	1.0	1.0	1.000000
2	0.000000	0.500000	1.0	0.0	0.888889
3	1.000000	0.500000	0.0	1.0	0.777778
4	1.000000	0.333333	1.0	0.0	0.666667
5	0.500000	0.333333	0.0	0.0	0.555556
6	0.500000	0.500000	1.0	1.0	0.444444
7	0.333333	0.500000	0.0	0.0	0.333333
8	0.333333	0.600000	1.0	1.0	0.222222
9	0.333333	0.666667	1.0	1.0	0.111111
10	0.333333	0.714286	1.0	1.0	0.000000

We observe that in the later episodes the action 1 is preferred since ϵ is close to 0, while in the early episodes the action choice is more random.

(e) In the decaying Softmax strategy the agent explores according to the action value. Actions having higher value are given more importance. Initially it explores randomly (high temperature), as the episodes progress it explores according to the softmax of the action values, by the end it greedily picks action (low temperature). Hyperparameters are optimized in part 4.

(f) In the UCB strategy, we use uncertainity as a bonus for exploration. Actions having more uncertainity in their action value are given more importance in exploration. Hyperparameters are optimized in part 4.

4. The optimal hyperparameters are obtained using grid search. They are listed below – The average reward vs episodes plots are given in the Jupyter notebook. We see that for the bernoulli bandit all the strategies

Strategy	Hyperparameter
ϵ-Greedy	0.1
decaying ϵ-Greedy	0.1 (exponential)
Softmax	0.1 (linear)
UCB	0.6

Table 2: Hyperparameters - Bernoulli Bandit

are almost similar. The best strategy is UCB followed by Softmax. They converge to a value of 0.65 approximately. This is because for the bernoulli bandit the reward space is discrete, even the action space is very low dimensional hence immediate benefit of these strategies is not scene.

5. Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Strategy	Hyperparameter
ϵ-Greedy	0.05
decaying ϵ-Greedy	0.03 (exponential)
Softmax	1 (linear)
UCB	0.6

Table 3: Hyperparameters - Gaussian Bandit

The average reward vs episodes plots are given in the Jupyter notebook. The best strategy for the Gaussian Bandit is UCB followed by epsilon Greedy and then Softmax. Here the difference between strategies is more apparent due to a larger action space and continuous reward space. A plot similar to what is shown in the slides is given, the number of environments are 2000 (as given in Sutton).

6. The regret vs episodes plots are given in the Jupyter notebook. The difference between various strategies is very clearly visible from the regret vs episodes plot. This is mainly because regret is cumulative and values vary over a large range, the strategy which converges earlier has a clear benefit over other strategies. We observe that UCB is the best strategy, since it has the minimum regret which converges implying an optimal policy has been achieved by the agent. The regret plots for other strategies have not converged which means that those agents have still not achieved an optimal policy. The PureExploration and PureExploitation are the worst strategies as apparent from the increasing regret values.

7. The regret vs episodes plots are given in the Jupyter notebook. The difference between various strategies is very clearly visible from the regret vs episodes plot. This is mainly because regret is cumulative and values vary over a large range, the strategy which converges earlier has a clear benefit over other strategies. We observe that UCB is the best strategy, since it has the minimum regret which converges implying an optimal policy has been achieved by the agent. The Softmax agent has also converged but the regret value is higher than UCB which means that the optimal policy is achieved later as compared to UCB. The regret plots for other strategies have not converged which means that those agents have still not achieved an optimal policy. An interesting observation is the PureExploitation regret plot. Even though it is lower than decaying ϵ-Greedy for almost 700 episodes, it is still a worse strategy since it's slope is constant(positive) and the regret values continue to increase, meaning it has not converged.

8. The % optimal actions vs episodes plots are given in the Jupyter notebook. We see that the UCB agent quickly goes to a 90% optimal action value (in about 100 episodes). This shows that it is the quickest to obtain an optimal policy. The PureExploitation and PureExploration strategies have fluctuate around 0.5 since the agent doesn't learn anything there. The ϵ and Softmax strategies have lower optimal actions initially as compared to UCB but converge to almost the same value by the end of 1000 episodes.

9. The % optimal actions vs episodes plots are given in the Jupyter notebook. The UCB and ϵ-Greedy strategies are the best strategies according to this plot. This plot is more interesting because we see that

decaying ϵ-Greedy has higher percent optimal actions initially as compared to Softmax but still it isn't the better strategy as seen from the regret plot.

Solution to Problem 2: MC Estimates and TD Learning

The numbering starts from 3. Please refer to the plots in the notebook.

1. We see that the v values are propagated as the number of episodes increase, which is what we expect.
2. We see that the v values are propagated as the number of episodes increase, which is what we expect. In TD the values propagate faster than MC which is again what we expect.
3. The MC-FVMC has high variance initially but converges around the optimal value as the episodes increase.
4. The MC-EVMC has high variance initially but converges around the optimal value as the episodes increase. It has slightly higher variance than FVMC initially.
5. The TD has lower variance than the MC strategies.
6. The high variance is more apparent in the logarithmic scale.
7. The high variance is more apparent in the logarithmic scale.
8. Even in the logarithmic scale, we see that the variance is low for TD.
9. TD is better than MCMC due to lower variance and faster, better convergence.
10. The target values are discrete (0, 1).
11. The target values are discrete same as FVMC.
12. The target values are more continous here and the variance is lower.