Project Report

Tanay Dixit (EE19B123), Vibhhu Sharma (EE19B128)

1 Acrobot-v1

Algorithms experimented Since Acrobot-v1 is an environment with continuous state and action space, we are faced with two options: either discretize the continuous spaces and then apply traditional algorithms like Q-Learning OR experiment with the policy gradients, a method known to work well in this scenario. We first attempted the former and implemented the Q-learning algorithm after discretizing the space using the Tile-Coding approach that has shown to be an very effective method in literature blue[1]. We tuned the hyper-parameters, primarily the number of bins and α (a factor that controls how much to weigh the old Q value of a particular (state,action) pair as compared to the newly calculated value) to obtain the best results. We made use of automatic parameter search by using hyperopt, which uses a Bayesian optimization method to search for the best parameters in the given space. This implementation gave a Leaderboard(LB) score of -150.45. Next, we implemented a softmax policy gradient method, involving a single linear layer blue[2]. This involved finding the optimal probability distribution on actions to be chosen in each state. Note that all this implementation was done using only *NumPy* based operations. We tuned the parameters and obtained the optimal values as mentioned in 1 and achieved a LB score of -80.25, which is our current best.
Best Results Based on the best parameters we obtained for each algorithm we implemented, we found out that the best results were obtained on using policy gradients, both in terms of speed and reward.

Algorithm	α	**Number of iterations/sec**	**LB score**
Q-Learning with Tiling	0.0041	3.35	-150.45
Policy Gradients	0.06	59.51	-80.25

Table 1: Acrobot-v1

2 Taxi-v3

Algorithms experimented As the environment was based upon a discrete space, the natural choice was Q-learning with ϵ-greedy approach. Same as before, we tuned its parameters using hyperopt, which led us to get a LB score of 8.14. As there was a lot of scope for improvement we tried other variants of Q-learning: primarily *expected-SARSA* and *sarasmax*. The optimal parameters identified for the different algorithms can be found in 2. Both performed at par with Q learning with minor differences on local evaluations and very similar scores on LB. Ultimately, we decided to use Q-learning itself since it achieved the best LB score.
Best Results Based on the best parameters we got for each algorithm we implemented, we found out that the best results were obtained on using Q-learning ϵ-greedy approach.

Algorithm	α	ϵ	γ	**LB score**
Q-Learning	0.8	0.9	0.96	8.14
Expected SARSA	0.3	1	0.8	8.0

Table 2: Taxi-v3

3 QuizBot for KBC

For all the following environments we found that it was necessary to fix the seed as there were several random operations, hence we fix the random seed to 0.

3.1 KBCA

Algorithms experimented Since the environment was again a discrete space, we began by experimenting with Q-learning. One of the main challenges we foresaw was dealing with the exploration and exploitation trade-off. For the ϵ-greedy method we experimented with different techniques to choose the action for exploration. We discovered that, the most suitable method was to always answer while exploring. We also added an ϵ-decay term that reduced the ϵ value every iteration in order to gradually move from exploration to exploitation. We also tried another variant of Q-learning: using a probability distribution based upon the Q-values(Class Notes-Page 83, Option-3). We tuned the temperature parameter to control the greediness of the action chosen. However, we found the ϵ-greedy method to work the best. The main parameters we focused on tuning were α, γ, ϵ and ϵ decay and we made use of hyperopt as before to obtain their values($\alpha = 0.6$, $\gamma = 0.99$, $\epsilon = 1$ and $\epsilon - decay = 0.99999$).
Best Results Based on the various methods we implemented we found that modified ϵ-greedy Q-learning worked the best. For exploration we found that the most suitable method was to always answer, while for exploitation choosing the optimal action worked the best. The optimal ϵ decay coefficient was found to be only slightly less than 1 at 0.99999. This hinted towards the fact that the given task required a serious amount of exploration as the environment had sparse rewards. This gave us 81920 as the best LB score.

3.2 KBCB

Algorithms experimented Since the only difference between KBC-A and KBC-B was the presence of a checkpoint question (unknown) that allowed the participant to receive a reward even after answering wrong beyond a certain stage, we hypothesized that the same algorithm used for KBC-A(ϵ-greedy Q-learning) would also work for KBC-B if tuned correctly as it would be able to learn the presence of the checkpoint question. We focused on tuning the values of hyperparameters: namely, α, γ, ϵ and ϵ decay in order to best suit the problem. We also attempted to tune the probability distribution on actions to be taken while exploring and found that always taking action-1, i.e., answering the question while exploring, provided the best reward.
Best Results Based on the different approaches we tried, we found that Q-Learning with ϵ-greedy with suitable changes as mentioned before worked best. This gave us 180480 as the best LB score.

3.3 KBCC

Algorithms experimented Similar to the previous environments, we implemented ϵ-greedy Q-learning with key changes to capture the nuances of the environment. For Instance along with adding a $\epsilon - decay$ term, we also played with the random action's probability distribution. Instead of choosing between action 1 and 2, with equal probability we decided to modify it to favour one, for this we made use of hyperopt to automatically find the best parameters as done before. Also, based upon our intuition, we also tried introducing another condition that increases the probability value for action 0 , when action 1 is chosen. We also tried other variants of Q-learning: primarily *expected-SARSA* but it did not give us a better score than our LB submission.
Best Results Based upon all the various methods we implemented we found that modified ϵ-greedy Q-learning worked the best, with the random action during exploration-stage being sampled from a probability distribution that assigned a probability of 0.4 to action-1 and 0.6 to action-2 while exploitation always chose the optimal action from the Q-Table. The best hyperparamters obtained were: $\alpha = 0.75$, $\gamma = 0.99$, $\epsilon = 1$ and $\epsilon - decay$ =0.99999. This gave us 35950 as the best LB score.

4 References

1. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.478.1711rep=rep1type=pdf
2. https://github.com/GerardMaggiolino/OpenAi-Gym-CartPole-Acrobot-Solutions
3. https://paperswithcode.com/method/expected-sarsa
4. https://github.com/udacity/deep-reinforcement-learning