APS1080 A2

Exercises

1. Explain clearly why V_pi is not useful in the MC development above?

   Since in our Monte Carlo Development we do not necessarily know where the action leads to, therefore knowing the state value function will not actually help to improve the policy. Therefore it should be more straightforward to know the state action value function.

2. The MC algorithm so far (ref: p 99), requires an infinite number of episodes for Eval to converge on Q_pi_k (step k). We can modify this algorithm to the practical variant where Eval is truncated (c.f., DynProg GPI). In this case:

   a. Will we obtain Q_pi_k from eval?

      No we can not obtain Q_pi_k from eval, since the evaluation is truncated so we can only get an value that’s close to Q_pi_k.

   b. If not why are we able to truncate Eval? Explain clearly.

      Since we know its result will not lose convergence guarantees of policy, which means it will converge to Q_pi_k, and since we are trying to get state action value and we do not need exact Q_pi_k to know the optimal policy. we also know only the first couple iterations have large impact on policy improvement, after certain steps evaluation update will become small and it is likely to be computational expensive.

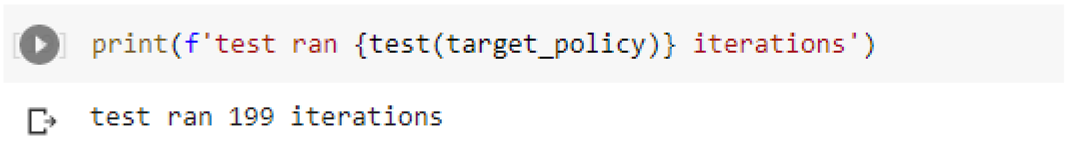
   c. Assuming ES (i.e., thorough sampling of the S x A space), and the above truncated Eval_trunc, is it possible to converge on a sub-optimal policy pi_c? Is this a stable fixed point of the GPI for MC? Explain clearly.

      Yes it is possible, since it’s possible to encounter some rare states only a couple times and due to Eval_trunc its value will only be updated a limit amount of time, in this case we might converge on a sub-optimal policy, this is not a stable fixed point because by tuning exploration we might escape from this point.

3. Explain how you can synthesize a stochastic policy given what you know so far (you don't need to read ahead).

   The simplest way is to use off-policy method and use a stochastic policy to generate episodes, then update target policy accordingly. One sample stochastic policy can be taking random action at each state.

Part B



```
print(f'test ran {test(target_policy)} iterations')
```

```
test ran 199 iterations
```

After discretizing the states by 40 and train for 100000 iterations I obtained this result, it’s 199 due to the limit of cartpole-v0. The model’s ability to keep the pole straight mostly depend on how many intervals we discretize the state since the more we discretize the more accurate action the model can take.