EECS 598: Reinforcement Learning, Homework 2

Abhishek Venkataraman

October 6, 2017

Problem 1. Stochastic Dyna-Q
In order to account for stochasticity, :

Converged policy for Dyna-Q:

Parameters:

Reason:

Converged policy for Dyna-Q+:

Parameters:

Reason:

Problem 2. Dyna-Q vs. Dyna-Q+

Problem 3. Prioritized Sweeping

Policy obtained by value iteration with parameter $\gamma = 0.95$, $\theta = 0.0001$

$$Policy = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 0. \end{bmatrix}$$

Problem 4. Reproducibility & Verification

Policy obtained by value iteration with parameter $\gamma = 0.95$, $\theta = 0.0001$

$$Policy = \begin{bmatrix} 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 1. & 0. \\ 1. & 0. & 0. & 0. \end{bmatrix}$$

Problem 5. Monte Carlo Control

I get a Q table and policy similar (but not exactly the same) to what I got earlier. However, when I looked through the optimal policy returned by the method, it seems to make sense, while looking at the frozen lake map. The terminal states have a Q value of 0 for all actions.

$$Q = \begin{bmatrix} 0.02493814 & 0.03838601 & 0.0294193 & 0.02647063 \\ 0.0112437 & 0.03006502 & 0.03687614 & 0.0466167 \\ 0.06729078 & 0.03979415 & 0.05595731 & 0.04642278 \\ 0.03108625 & 0.02096409 & 0.02698308 & 0. \\ 0.03284949 & 0.04799333 & 0.02831384 & 0.0189512 \\ 0. & 0. & 0. & 0. \\ 0.08847326 & 0.06405801 & 0.11297363 & 0.05231165 \\ 0. & 0. & 0. & 0. \\ 0.03040877 & 0.07710062 & 0.06266217 & 0.11310736 \\ 0.14465794 & 0.1400729 & 0.19451944 & 0.09086186 \\ 0.30967946 & 0.28228565 & 0.10998967 & 0.04645254 \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0.11252327 & 0.31696174 & 0.33995534 & 0.24603164 \\ 0.177179 & 0.44868949 & 0.26029546 & 0.18340379 \\ 0. & 0. & 0. & 0. \end{bmatrix}$$

$$Policy = \begin{bmatrix} 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. \\ 0. & 0. & 1. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 0. \end{bmatrix}$$

Problem 6. Q-Learning

The final Q table after 500 episodes :

$$Q = \begin{bmatrix} 0.04463358 & 0.04950337 & 0.05075302 & 0.05127952 \\ 0.02430698 & 0.0145442 & 0.04119943 & 0.05007132 \\ 0.06869312 & 0.05069395 & 0.01123521 & 0.03206709 \\ 0.0063983 & 0.00689914 & 0.00972855 & 0.04960062 \\ 0.05176841 & 0.00674974 & 0.00778528 & 0.02978962 \\ 0. & 0. & 0. & 0. \\ 0.04027403 & 0.00101772 & 0.09471334 & 0. \\ 0. & 0. & 0. & 0. \\ 0.00441195 & 0.02145365 & 0.010348 & 0.04226591 \\ 0.03960406 & 0. & 0. & 0.02059928 \\ 0.09953314 & 0.53531923 & 0. & 0.07445918 \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0.05700875 \\ 0.23986141 & 0.37037029 & 0.80054314 & 0. \\ 0. & 0. & 0. & 0. \end{bmatrix}$$

The Q values of terminal states are all 0.

Problem 7. Rate of convergence

Proof. We know that the maximum reward in any step is given by,

$$V^*(s) \leq \sum_{i=1}^{\infty} \gamma^i R_{max}$$

Given that the reward for all state action pair is between 0 and 1, we know that R_{max}=1;

$$V^*(s) \leq \sum_{i=1}^{\infty} \gamma^i$$

By using summation over infite series, (since $\gamma \leq 1$),

$$V^*(s) \leq \frac{1}{1-\gamma}$$

The value function over k^{th} iteration can be written as:

$$V_k(s) = \sum_{i=1}^{k} \gamma^i$$

$$\therefore V_k(s) = \frac{1-\gamma^k}{1-\gamma}$$

Taking the difference between the optimal value and k^{th} iteration,

$$
\begin{aligned}
|V_k(s) - V^*(s)| &\leq \epsilon \\
|\frac{1-\gamma^k}{1-\gamma} - \frac{1}{1-\gamma}| &\leq \epsilon \\
|\frac{-\gamma^k}{1-\gamma}| &\leq \epsilon \\
\frac{\gamma^k}{1-\gamma} &\leq \epsilon
\end{aligned}
$$

taking logarithm on both sides,

$$
\begin{aligned}
\log \frac{\gamma^k}{1-\gamma} &\leq \log \epsilon \\
\log \gamma^k - \log(1-\gamma) &\leq \log \epsilon \\
k \log \gamma &\leq \log \epsilon + \log(1-\gamma) \\
k &\leq \frac{\log(\epsilon.(1-\gamma))}{\log \gamma}
\end{aligned}
$$

Hence the number of steps that guarantee the absolute error to be less than ϵ is given by:

$$
\frac{\log(\epsilon.(1-\gamma))}{\log \gamma}
$$

□

Problem 8. Exact Policy Evaluation for an MDP

Proof. Value function is given by,

$$
V(s) = \mathbb{E}[G_t | S_t = s]
$$

where, G_t is the discounted return starting at time step t for state s

$$
\begin{aligned}
V(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s] \\
V(s) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
V(s) &= \mathbb{E}[R_{t+1} + \gamma v S_{t+1} | S_t = s] \\
V(s) &= R_s + \gamma \sum_{s' \in S} P_{ss'} V(s')
\end{aligned}
$$

where s' is the set of next possible state.
Enumerating over s, we can write it in matrix form as

$$
\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{bmatrix} + \gamma . \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}
$$

Writing in vectorized form,

$$\mathbf{V}_\pi = \mathbf{R}_\pi + \gamma.\mathbf{P}_\pi.\mathbf{V}_\pi$$
$$\therefore \mathbf{V}_\pi = (\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}\mathbf{R}_\pi$$

□

Problem 8. Meta

I got a late enrollment to the class and hence I had to start the assignment really late. I worked for over 3 days and spent around 20 hours for the assignment. I had to catch up with the lectures within this period.

- Prob 1 & 2:It took a lot of time to try out the parameters.
- Prob 3: The implementation was not too hard. I spent a total of 3 hours on this problem
- Prob 4: I spent around 4 hours on this problem. 1 hour was spend on understanding the use fo requirements.txt and tesing it on a another linux machine. The README.md took around 0.5 hours since I have never made one before. 2.5 hrs was spent towards manually testing the algorithm.
- Prob 5: I spent around 3 hours on this problem. Most of the time was spent in trying to understand the question.
- Prob 6: I spent around 3 hours on this problem
- Prob 7: I spent around 3 hours on this problem