HW5 Report

Y.S.S.V Sasi Kiran

December 11, 2018

1 Written

1.1 Prob 1

GIven, $A_t \sim N(\theta^T\phi(S_t), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)$, where $\mu = \theta^T\phi(S_t)$

$$\begin{aligned}
\implies \frac{\partial \log \pi(s,a,\theta)}{\partial\theta} &= \frac{1}{\pi(s,a,\theta)}\frac{\partial\pi(s,a,\theta)}{\partial\theta} \\
&= \frac{1}{\pi(s,a,\theta)}\left[\frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)\right]\left(-\frac{2(a-\mu)}{2\sigma^2}\right)\left(-\frac{\partial\mu}{\partial\theta}\right) \\
&= \frac{1}{\pi(s,a,\theta)}\pi(s,a,\theta)\left(\frac{(a-\mu)}{\sigma^2}\right)\left(\frac{\partial\mu}{\partial\theta}\right) \\
\implies \frac{\partial \log \pi(s,a,\theta)}{\partial\theta} &= \left(\frac{(a-\theta^T\phi(s))}{\sigma^2}\right)\phi(s)
\end{aligned}$$

Similarly,

$$\begin{aligned}
\frac{\partial \log \pi(s,a,\theta)}{\partial\sigma} &= \frac{1}{\pi(s,a,\theta)}\frac{\partial\pi(s,a,\theta)}{\partial\sigma} \\
&= \frac{1}{\pi(s,a,\theta)}\left[\frac{1}{\sqrt{2\pi}}.\left(\frac{-1}{\sigma^2}\right) exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right) + \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right)\left(\frac{-(a-\mu)^2}{2}.\frac{-2}{\sigma^3}\right)\right] \\
&= \left[\frac{-1}{\sigma} + \left(\frac{(a-\mu)^2}{\sigma^3}\right)\right] \\
&= \frac{1}{\sigma}\left[\frac{(a-\theta^T\phi(s))^2}{\sigma^2} - 1\right]
\end{aligned}$$

1.2 Prob 2

The residual gradient like algorithm mentioned uses the weight update given by

$$w \leftarrow w + \alpha(R_t + \gamma w^T\phi(S_{t+1}) - w^T\phi(S_t))(\phi(S_t) - \gamma\phi(S_{t+1}))$$

The fixed point using this update occurs when expected update is zero. i.e.

$$\begin{aligned}
&\mathbb{E}[(R_t + \gamma w^T\phi(S_{t+1}) - w^T\phi(S_t))(\phi(S_t) - \gamma\phi(S_{t+1}))] = 0 \\
\implies &\mathbb{E}[R_t(\phi(S_t) - \gamma\phi(S_{t+1}))] + \mathbb{E}[w^T(\gamma\phi(S_{t+1}) - \phi(S_t))(\phi(S_t) - \gamma\phi(S_{t+1}))] = 0 \\
&\implies \mathbb{E}[R_t(\phi(S_t) - \gamma\phi(S_{t+1}))] = \mathbb{E}[w^T(\phi(S_t) - \gamma\phi(S_{t+1}))(\phi(S_t) - \gamma\phi(S_{t+1}))] \\
&\implies \underbrace{\mathbb{E}[(\phi(S_t) - \gamma\phi(S_{t+1}))(\phi(S_t) - \gamma\phi(S_{t+1}))^T]}_{A} w = \underbrace{\mathbb{E}[R_t(\phi(S_t) - \gamma\phi(S_{t+1}))]}_{b} \qquad (1.1)
\end{aligned}$$

Equation (1.1) can be solved using least squares and the algorithm for it using collected (s, a, r, s') tuples goes as follows.

Algorithm 1 Least Squares residual-gradient-like algorithm

Initialize $A \leftarrow 0$, $b \leftarrow 0$
for all (s, a, r, s') **do**
    $d \leftarrow \phi(s) - \gamma\phi(s')$
    $A \leftarrow A + dd^T$
    $b \leftarrow b + rd$
end for
$w \leftarrow solve\_least\_squares(A, b)$

2 Programming

2.1 Sarsa-λ and Q-λ

For gridworld domain, we found that $\epsilon = 0.2$, $\alpha = 0.75$, $\lambda = 0.3$ gave the best performance for SARSA-λ. Similarly for Q-λ, we found that $\epsilon = 0.1$, $\alpha = 0.5$, $\lambda = 0.3$ gave the best performance. The learning curves for gridworld for SARSA, SARSA-λ, Q-learning, Q-λ are presented below.

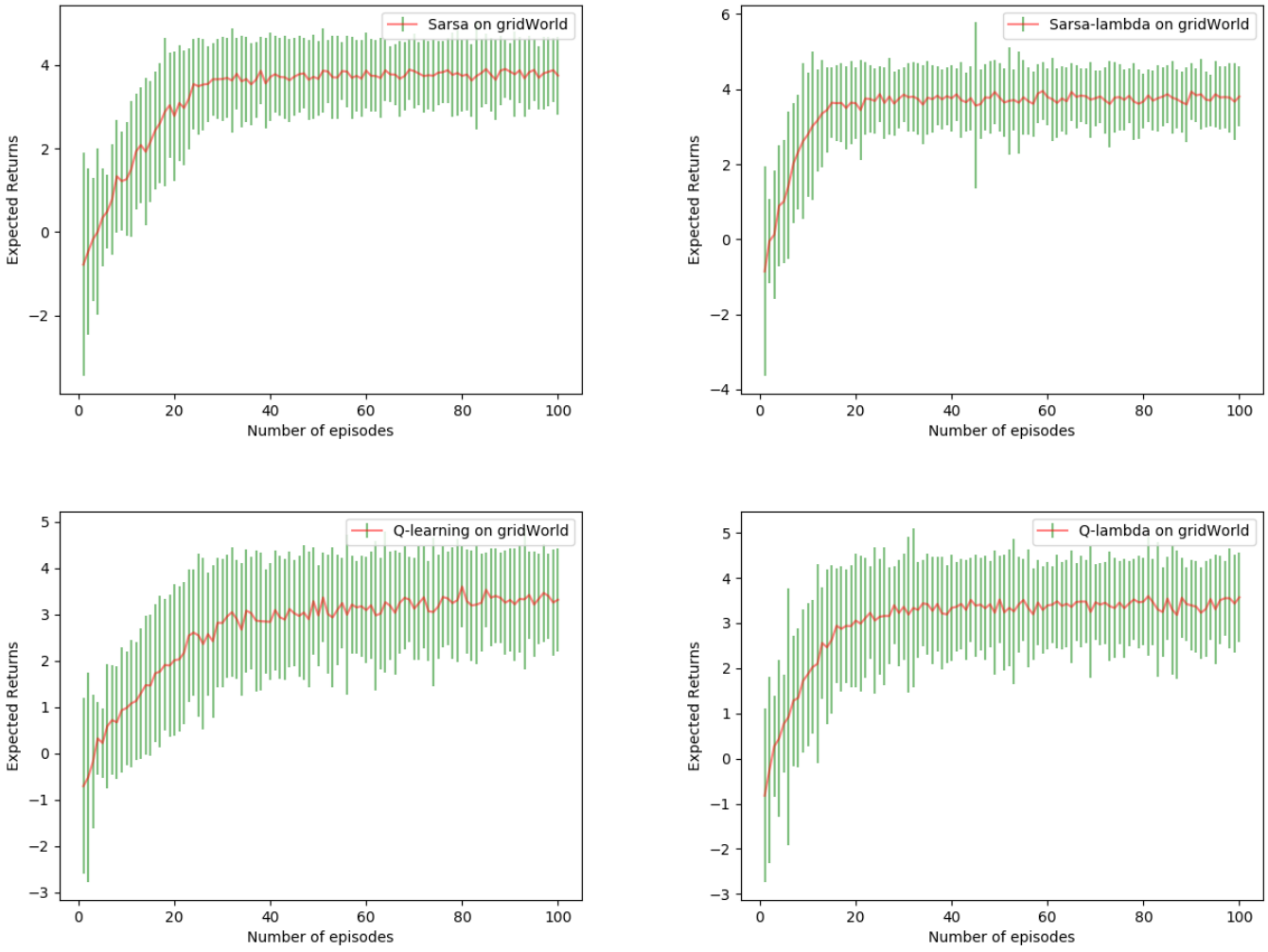

Figure 2.1: The above plots show the learning curves for SARSA, SARSA-λ, Q-learning, Q-λ on gridWorld domain from top left to bottom right.

Similarly for the mountain car domain, we found that fourier basis with $order = 5$, $\epsilon = 0.02$, $\alpha = 0.0075$, $\lambda = 0.2$ worked the best for SARSA-λ. For Q-λ, fourier basis with $order = 5$, $\epsilon = 0.01$, $\alpha = 0.005$, $\lambda = 0.3$ worked the best. The learning curves for each of them are presented below.

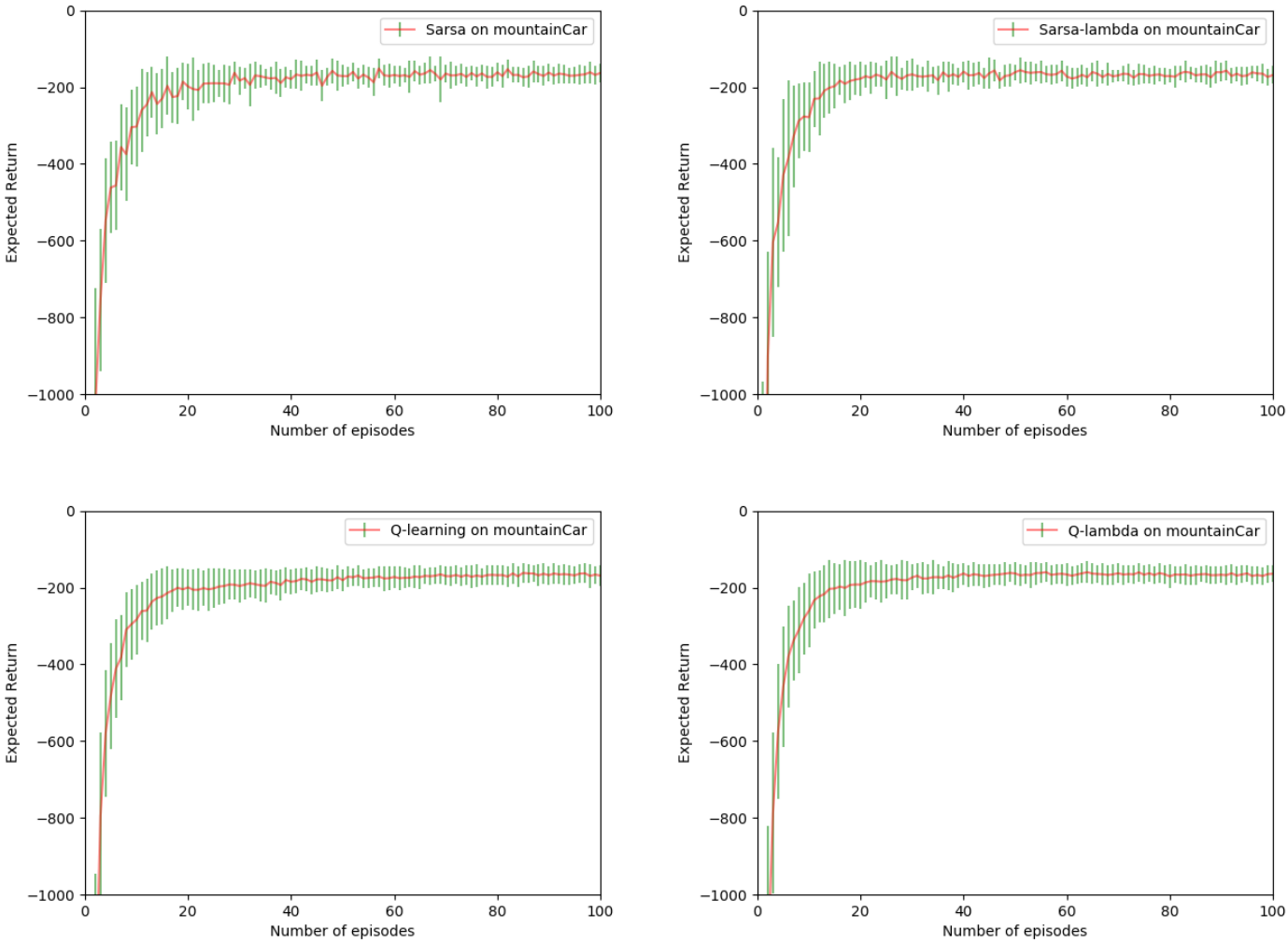

Figure 2.2: The above plots show the learning curves for SARSA, SARSA-λ, Q-learning, Q-λ on mountainCar domain from top left to bottom right.

We noticed that for mountain car, if we take the average of returns for last 50 epsiodes to get convergence value, we obtain ~ -160 for both SARSA-λ and Q-λ and a maximum of ~ -140 for both of them. The lambda version of the algorithms were observed to converge faster than its non-lambda counterpart. This can be clearly seen in gridWorld domain. This happens because of lesser bias in lambda variants. For higher values of lambda, these lambda variants ($\lambda > 0.6$) showed lots of oscillations showing higher variance.

2.2 Actor Critic

We use tabular representation for the gridWorld domain. For gridWorld, we found that $\alpha = 0.8$ and $\lambda = 0.35$ was the best setting. We intialize the weights for value function to be all 4 i.e. optimistic initialization. We decay the learning rate α as $\alpha \leftarrow \alpha/\sqrt{N}$ where N is incremented for each episode. For assessing the convergence point we find the mean return for the last 50 episodes and found it to be 3.43. The learning curve for gridWorld is given below.

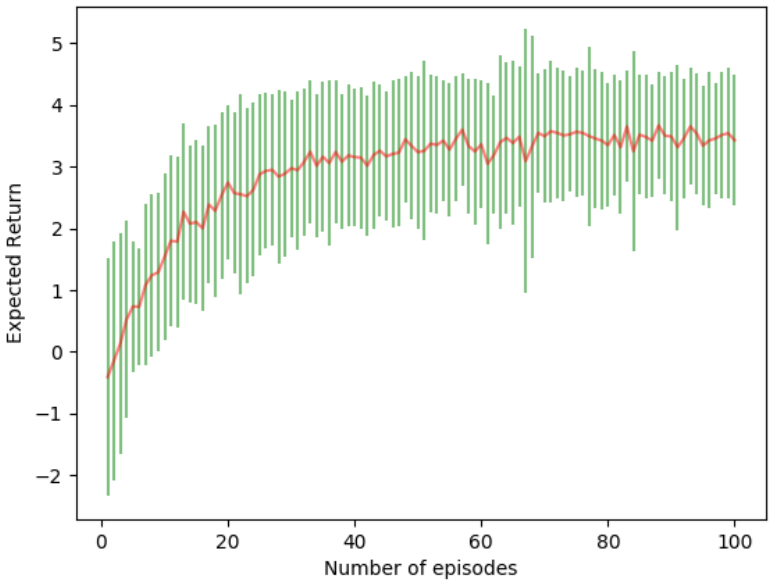

Figure 2.3: Learning curve for gridWorld using actor critic.

Similarly for mountain car, we use fourier basis of *order* $= 5$. We found that $\alpha = 0.002$ and $\lambda = 0.6$ was the best setting. We initialize the weights for value function to be all 1. We do not decay the learning rate for the first 50 episodes and then decay it as $\alpha \leftarrow \alpha/\sqrt{N}$ where N is incremented by a maximum of 10 for each episode. For assessing the convergence point, we find the mean return for the last 50 episodes and found it to be -165. The learning curve for mountainCar is given below.

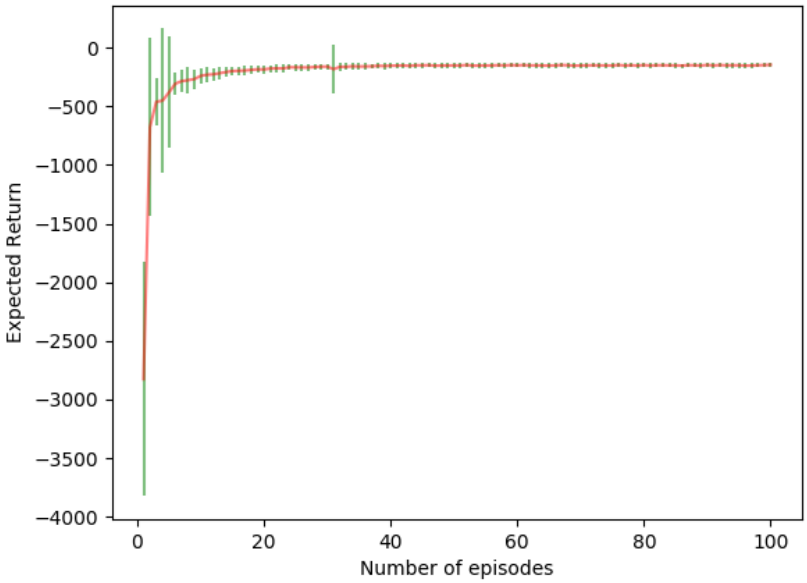

Figure 2.4: Learning curve for mountainCar using actor critic

Compared to Sarsa and Q-learning, this actor-critic was quite easy to tune and it started to easily work on several ranges of step sizes. Just using the optimal stepSize from SARSA or Q-learning and tuning the λ parameter was good enough for actor-critic. All the three algorithms converge to almost the same expected returns. At the same time, actor-critic for mountain car converges quite early to the optimal range compared to Sarsa and Q-learning. To compare the performance, we show below the mean expected return curves for SARSA, Q-learning and actor-critic for both gridWorld and mountain Car side-by-side.

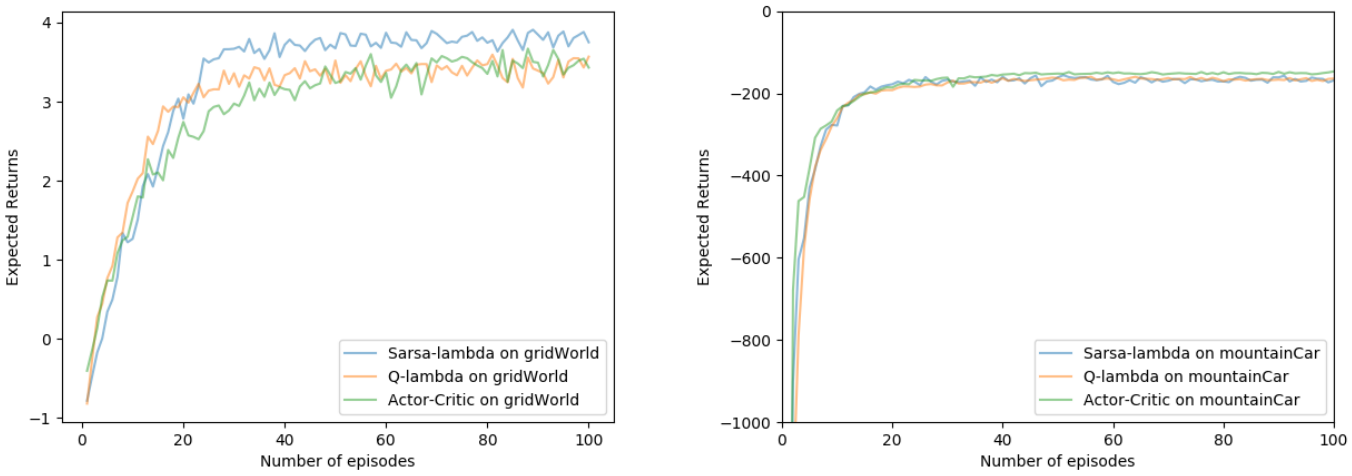

Figure 2.5: Plot showing a side-by-side comparison of performance of the algorithms on both gridWorld (left) and mountainCar (right)

2.3 Reinforce

We implemented Reinforce with no baseline and apply it to gridWorld setting. We used tabular representation and $\alpha = 0.045$. We decay the step size α as $\alpha \leftarrow \alpha/\sqrt{N}$ where N is not updated for first 250 episodes and then incremented by 1 for each episode. We observe the following learning curve.

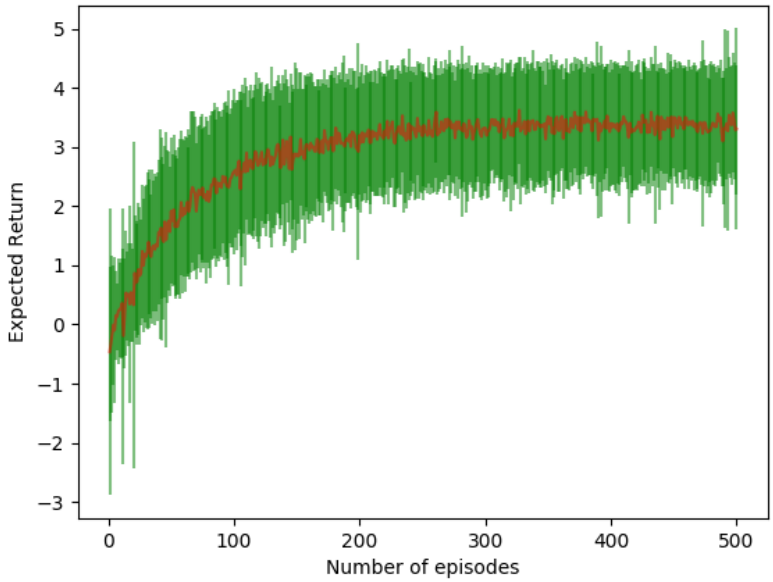

Figure 2.6: Plot shows learning curve for gridWorld using reinforce with no baseline.

We notice that it converges to > 3 but the variance is too high as this is reinforce without baseline. We have also tried to get the Reinforce to work on mountain car but was not successful. We tried to decode the reasons for this and found that after first episode, the gradient was too large when using moderate step size and it immediately diverges and there is no update for small step sizes, resulting in curve staying always constant at horizon (if mountain car was limited by a horizon of -2000).