10-703 Deep Reinforcement Learning and Control
Assignment 3
Fall 2018

ielshar
mharding

October 26, 2018

Problem 1: REINFORCE

1. Describe your implementation:
   - Neural Network Architecture: same as given in JSON file. No baseline was used.
   - Learning rate: 0.001
   - Discount factor γ : 1

2.

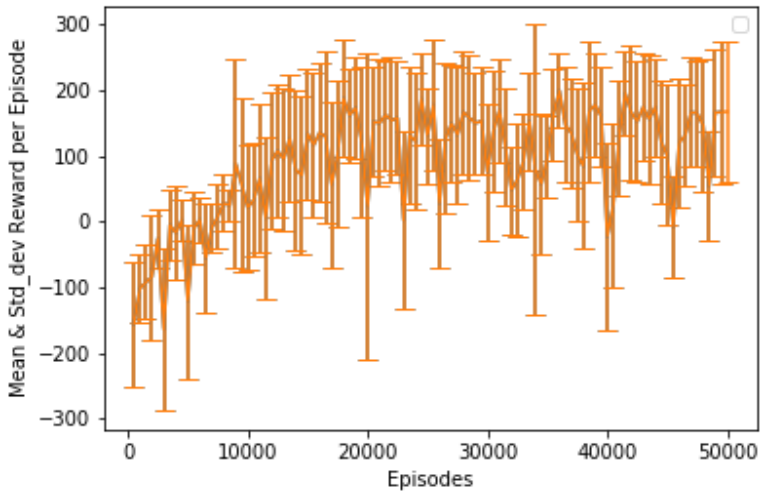

Figure 1: REINFORCE Algorithm without baseline: learning curve - Every k=500 episodes the current policy is tested on 100 episodes. The plot shows the mean and standard deviation of each of this tests.

Figure 1 above shows that our agent was able to achieve a mean reward of 200 or slightly more or less at several points (after 36000 and 41500 episodes) throughout the training. Since no baseline was used in our REINFORCE algorithm one can see that the test results have a high variance. This is somehow expected though as the total return at the end of episode varies greatly from one sampled episode to another.

Problem 2: Advantage-Actor Critic

1. Implementation details:

Table 1: A2C Implementation

A2C		
N	Settings	value
1	Actor NN Architecture	Same as given in JSON file
	Critiic NN Architecture	MLP with 3 layers each with 30 hidden units and relu activation
	Actor learning rate	0.001
	Critic learning rate	0.001
20	Actor NN Architecture	Same as given in JSON file
	Critiic NN Architecture	MLP with 3 layers each with 20 hidden units and relu activation
	Actor learning rate	0.001
	Critic learning rate	0.001
50	Actor NN Architecture	Same as given in JSON file
	Critiic NN Architecture	MLP with 3 layers each with 20 hidden units and relu activation
	Actor learning rate	0.001
	Critic learning rate	0.001
100	Actor NN Architecture	Same as given in JSON file
	Critiic NN Architecture	MLP with 3 layers each with 20 hidden units and relu activation
	Actor learning rate	0.0008
	Critic learning rate	0.001

* discount factor was set to 1 in all implementations. Output activations were all linear.

2. For plots below, every k=500 episodes the policy being trained is tested on 100 episodes. The plot shows the mean and standard deviation of each of this tests.)

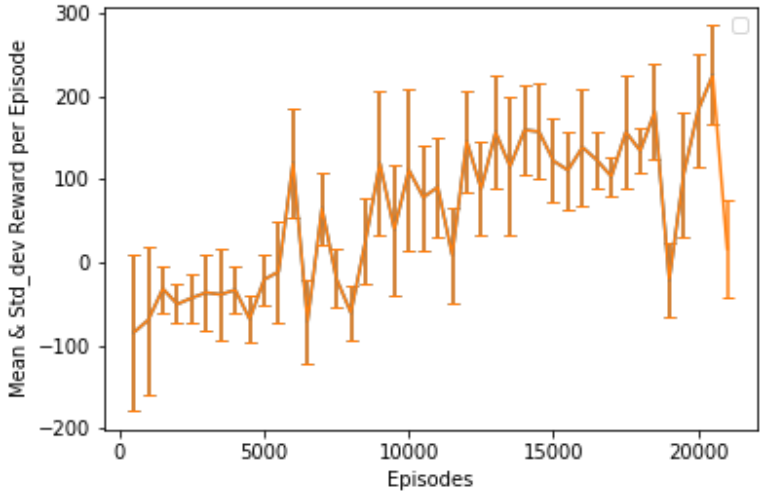

Figure 2: A2C Algorithm N=1: learning curve. Wider critic network (30x30x30, relu activations for hidden layers, linear activation for output layer), critic lr = 0.001, actor lr = 0.001. First reached a mean reward of 225 after 20500 episodes.

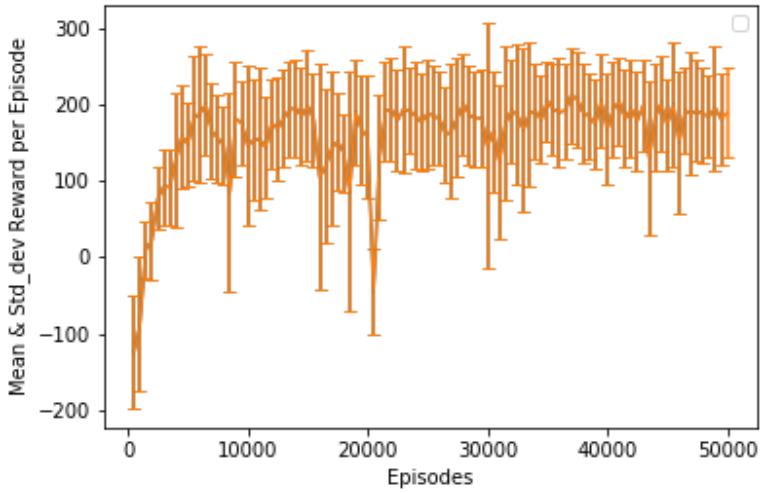

Figure 3: A2C Algorithm N=20: learning curve. Actor and critic lr = 0.001. First reached a mean reward of 200 after 6500 episodes.

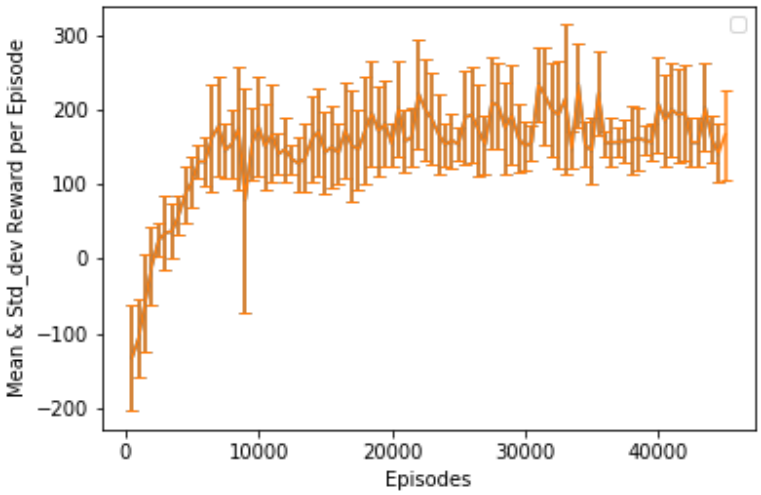

Figure 4: A2C Algorithm N=50: learning curve. Actor and critic lr = 0.001. First reached a mean reward of 201 after 20500 episodes.

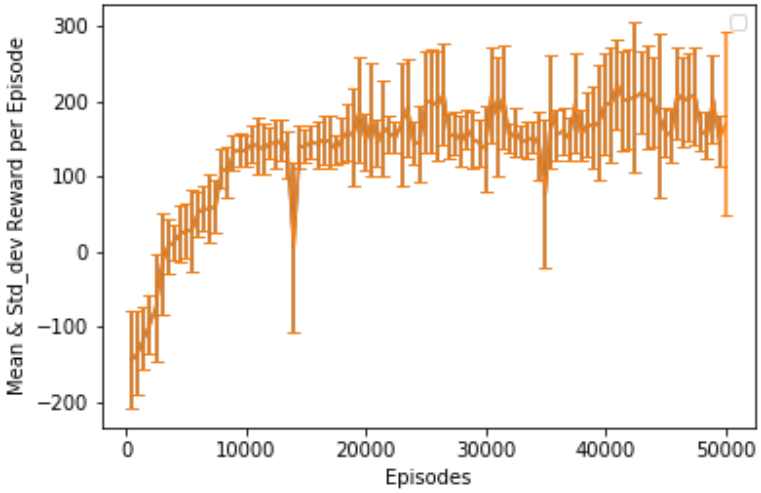

Figure 5: A2C Algorithm N=100: learning curve. Actor lr=0.0008 and critic lr=0.001. First reached a mean reward of 200 after 25500 episodes.

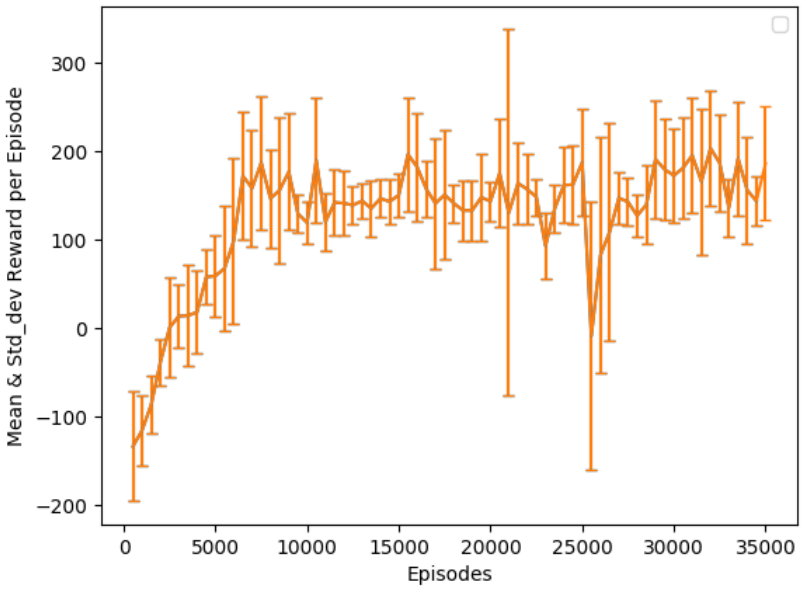

Figure 6: A2C Algorithm N=100: 2nd learning curve showing slower convergence. Actor lr=0.001 and critic lr=0.001. First reached a mean reward of 204 after 32000 episodes.

3. REINFORCE and A2C comparisons:
   Compared to REINFORCE, n-steps A2Cs have much lower variance and accelerated learning, except for the case when n=1. As discussed, the REINFORCE algorithm without a baseline in its target introduces significant variance inherent to Monte Carlo sampling of episodic rewards. This hinders learning as rewards for optimal actions are not normalized and many more noisy gradient updates are required for convergence. The case of A2C with one-step critic introduce much more noise than n=20,50 and 100 A2Cs due to its greater dependence on bootstrapping from the learned critic network, which meant convergence was more difficult. It seems that A2C algorithm with n=20, 50 and 100 learn faster than all other algorithms. Even though we were able to achieve a better performance for n=100 with an actor learning rate of 0.0008 as seen in Figure 5 above, we run A2C n=100 again for 35000 episodes with an actor lr=0.001 (Figure 6) which is the same as the one used for A2C n=20 and n=50. This was done to make sure the comparison regarding speed of learning is fair enough since now the only difference between them is n, the number of steps for the critic's lookahead. (Note that we fix the random seed to 1234 for all algorithms).

   A2C n=20 was the fastest to reach a mean reward of 200 or greater. It achieved that after only 6500 episodes. A2C n=50 (50-steps return) comes second with 20500 episodes. Third comes A2C n=100 with 32000 episodes. Fourth comes Reinforce with 36000 episodes. However, for n=20, 50, and 100, these algorithms stably achieve 200 reward in the same range of episodes (between 20k and 31k). A2C n=1 was not able to achieve 200 with the same neural architecture for the critic as the other algorithms.

However using a critic NN architecture that include 3 layers each with 30 units this time A2C n=1 achieved a reward of 225 after 20500 episodes similar to A2C n=50. Comparing figures 3, 4 and 6, it can be seen that A2C n=20 was quickest to reach 200 reward, then comes n=50 and finally n=100. More precisely, n=20 and n=50 had similar convergence, as both stably reached 200 reward around 20.5k episodes.

We believe this is the case since 20-step and 50-step A2Cs struck the best balance for the LunarLander-v2 environment between bootstrapping using the learned critic value network and using the MC sampled episodic returns. 20-step A2C demonstrated wider variance in its training than 50-step, so 50-step A2C may arguably be preferrable. Generally, larger N for number of steps for A2C may be favored due to the longer nature of a typical successful episode of the LunarLander-v2 environment, in which the time to land is not greatly penalized and, instead, stable, complete deceleration of the lander agent within the landing zone is all that is required for reaching a score above 200.