Deep Reinforcement Learning Exercise 2

Shahaf Yamin 311530943
Ron Sofer 204738637

December 24, 2021

Running Instruction for the script of this assignment.

- Question 1:

  for Vanilla REINFORCE run the following command -

  ```
  python policy_gradients.py
  ```

  for REINFORCE using Advantage trained **with** Replay-Buffer & Target-Network run the following command -

  ```
  python policy_gradients.py --N 0
  ```

  for REINFORCE using Advantage trained **without** Replay-Buffer & Target-Network run the following command -

  ```
  python policy_gradients.py --N 501
  ```

- Question 2: run the following command -

  ```
  python policy_gradients.py --N 1
  ```

- Extras: run the following command -

  ```
  python policy_gradients.py
  ```

  with the following optional arguments to change configurations.

  1. "−−N <n>", for n-step Actor-Critic (for n>0)
  2. "−−exp <exp_name>", to set experiment name, and save results under this name.
  3. "−−n_episodes <integer>" to set the number of max episodes
  4. "−−gpu <x>" to set CUDA_VISIBLE_DEVICES environment variable, and use gpu indexed with x.

1 Monte-Carlo Policy Gradient (REINFORCE)

1.1 Introduction

As a first step, we will describe the Policy Gradient objective mathematically in order to reduce the complexity in our further answers. Our goal is to find a policy $\pi_{\theta^\star}$ such that it maximizes our expected rewards:

$$\theta^\star = \arg\max_\theta J(\theta) = \arg\max_\theta \mathbb{E}^{\pi_\theta}[\sum_{k=0}^{T-1} \gamma^k \cdot r_{k+1}] \tag{1}$$

We denote γ as our discount factor, the term $\mathbb{E}^{\pi_\theta}$ means that we calculate the expectation under the assumption that we follow the policy π_θ. Thus, we can expand this term in the following manner,

$$\mathbb{E}^{\pi_\theta}\underbrace{[\sum_{k=0}^{T-1} \gamma^k \cdot r_{k+1}]}_{G_0} = \int \pi_\theta(\tau) G(\tau) d\tau \tag{2}$$

Where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ represent an available trajectory, and $G(\tau) = \sum_{k=0}^{T-1} \gamma^k \cdot r(s_k, a_k)$ represents the total discounted return for following the trajectory τ. The term $\pi_\theta(\tau)$ represents the probability of experiencing trajectory τ which induced by following policy π_θ and the underline transition probabilities. Now we get that,

$$\theta^\star = \arg\max_\theta J(\theta) = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta(\cdot)}[G(\tau)] = \arg\max_\theta \int \pi_\theta(\tau) \cdot G(\tau) d\tau \tag{3}$$

As a next step, we would like to integrate gradient ascent based method to solve this problem. Therfore, we shall use a derivative w.r.t θ to our objective $J(\theta)$.

$$\nabla_\theta J(\theta) = \nabla_\theta \int \pi_\theta(\tau) \cdot G(\tau) d\tau \overset{(1)}{=} \int \nabla_\theta \pi_\theta(\tau) \cdot G(\tau) d\tau \overset{(2)}{=} \int \pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau) \cdot G(\tau) d\tau \tag{4}$$

Where,

1. We can insert the derivative to the integral since the integral borders are not dependent on θ.
2. We use the following equailty $\pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta$.

As a final step, we use the following transition,

$$\int \pi_\theta(\tau) \cdot \nabla_\theta \log \pi_\theta(\tau) \cdot G(\tau) d\tau = \mathbb{E}_{\tau \sim \mathbb{P}(\cdot|\pi_\theta)}[\nabla_\theta \log \pi_\theta(\tau) \cdot G(\tau)] \tag{5}$$

This means that we can represent the objective gradient derivative w.r.t the policy parameters θ as an expectation. Which implies that we can approximate this term using sampling method. Using the Markov property we can define the probability $\pi_\theta(\tau)$ as followed,

$$\pi_\theta(\tau) = \mathbb{P}(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) \cdot \mathbb{P}(s_{t+1}|s_t, a_t) \tag{6}$$

Taking log on the two sides of the equation we get,

$$\log \pi_\theta(\tau) = \log \mathbb{P}(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log \mathbb{P}(s_{t+1}|s_t, a_t) \tag{7}$$

Now, applying the derivative w.r.t θ we get,

$$\nabla_\theta \log \pi_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{8}$$

Which leads to the following approximation using sampling methods:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{k=1}^{N} ((\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^k, s_t^k)) \cdot (\sum_{t=0}^{T-1} \gamma^t \cdot r(a_t^k, s_t^k)) \tag{9}$$

And the following parameter update rule,

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta) \tag{10}$$

1.2 What does the value of the advantage estimate reflect?

The advantage function defines the increase or decrease of the return when taking action a in state s with respect to mean expected return at state s.

$$\hat{A}_t = G_t - V^{\pi_\theta}(S_t) \tag{11}$$

For example, if an action is better than expected ($\hat{A}_t > 0$), we want to encourage the actor to take more of that action. If an action is worse than expected ($\hat{A}_t < 0$), we want to discourage the actor to take that action. If an action performs exactly as expected ($\hat{A}_t = 0$), the actor doesn't learn anything from that action.

1.3 Why is it better to follow the gradient computed with the advantage estimate instead of just the return itself?

Intuitively speaking, using the advantage estimate instead of the return provides more qualitative information, in terms of "how better than expected" choosing a specific action was, which increase the efficiency in the learning process. Furthermore, it reduces the variance in the gradients as shown in the following equation.

$$\mathrm{Var}\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\big(G_t - b(s_t)\big)\right) \overset{(i)}{\approx}$$

$$\underbrace{\sum_{t=0}^{T-1}\mathbb{E}^{\pi_\theta}\left[\left(\nabla_\theta \log \pi_\theta(a_t|s_t)\big(G_t - b(s_t)\big)\right)^2\right]}_{E_1} - \underbrace{\sum_{t=0}^{T-1}\mathbb{E}^{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\big(G_t - b(s_t)\big)\right]^2}_{E_2}$$

$$E_1 \overset{(ii)}{\approx} \sum_{t=0}^{T-1}\mathbb{E}^{\pi_\theta}\left[\big(\nabla_\theta \log \pi_\theta(a_t|s_t)\big)^2\right]\mathbb{E}^{\pi_\theta}\left[\big(G_t - b(s_t)\big)^2\right]$$

$$E_2 = \sum_{t=0}^{T-1}\mathbb{E}^{\pi_\theta}\left[\big(\nabla_\theta \log \pi_\theta(a_t|s_t)\big)G_t\right]^2 - \underbrace{\mathbb{E}^{\pi_\theta}\left[\big(\nabla_\theta \log \pi_\theta(a_t|s_t)\big)b(s_t)\right]^2}_{=0}$$

Generally speaking, T is a R.V, we assume that we set its value as the maximal value and therefore, it is deterministic. We approximate the variance of a sum by computing the sum of the variances (approximation (i)). This is not true in general, but if we assume this we can neglect the effect of E_2 - since E_2 will not change w.r.t $b(s_t)$, due to next section results, which shows that the bias function doesn't introduce bias to the gradient estimation. Approximation (ii) is true if we assume independence among the values involved in the expectation. Using these 2 approximations, we receive that the gradient is linearly dependent in $\mathbb{E}^{\pi_\theta}\left[\big(G_t - b(s_t)\big)^2\right]$ which can be considered as Mean Squared Error (MSE) between G_t and $b(s_t)$. Therefore, by using the baseline function $b(s_t) = \mathbb{E}^{\pi_\theta}\left[G_t\right] = V^{\pi_\theta}(s_t)$ we will minimize the MSE term. Although the assumptions are not generally true, empirically using baseline functions as such produces less variance. The equations above provide rough explanation of the intuition behind the baseline-functions.

1.4 The reduction of the baseline does not introduce bias to the expectation of the gradient. What is the prerequisite condition for the given equation to be true?

The given equation is $\mathbb{E}^{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right] = 0$, we will prove it.

$$\mathbb{E}^{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\Big] = \int \Bigg(\underbrace{\int \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)da}_{I_A}\Bigg)ds$$

$$I_A \overset{(1)}{=} b(s)\int_a \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a_t|s_t)da \overset{(2)}{=} b(s)\int_a \nabla_\theta \pi_\theta(a,s)da \overset{(3)}{=}$$
$$b(s)\nabla_\theta \int_a \pi_\theta(a,s)da \overset{(4)}{=} b(s)\nabla_\theta 1 = 0$$

1. Holds if $b(s_t)$ is not a function of the action a
2. 'Log trick' as shown previously
3. The limits of the integral are independent w.r.t θ
4. The probability of taking an action (any action) from state s equals 1

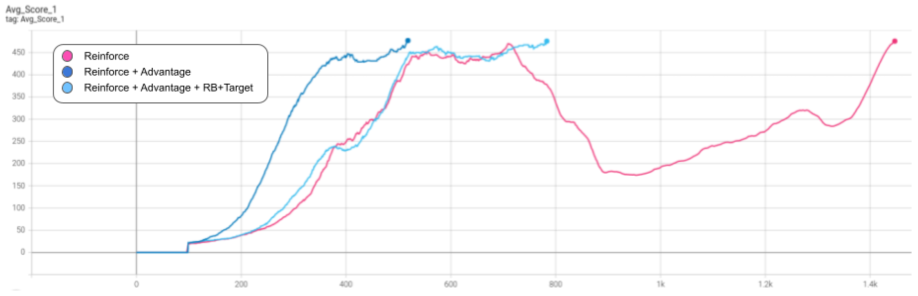
In conclusion, the prerequisite condition for the equation to be true is that $b(s_t)$ is not a function of the action.

1.5 Results

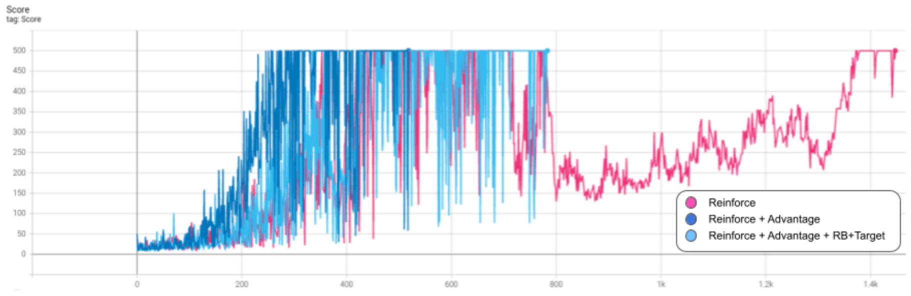
In this section, we implement the REINFORCE algorithm with an advantage estimate that uses a value-function approximation baseline instead of the actual return of every episode. The hyperparameters used for these experiments are shown in Table 1. α, $\mathcal{B}$, $\mathcal{K}$, $\mathcal{C}$ represents the learning-rates, batch-size, target network update frequency, and buffer-capacity respectively. In our implementation, we used two different versions of value approximation. The first version is similar to the previous assignment's estimator, i.e., using target-network and a replay-buffer (RB) during the training process. The second version did not include any of the aforementioned extensions, i.e., Vanilla approximation. For both versions, we didn't change the original policy network architecture. Although the REINFORCE algorithm is an on-policy algorithm, we chose to examine the impact of using a baseline-function that doesn't approximate V^{π_θ} directly, which refer to the use of RB and target-network extensions. We can do that since the only prerequisite over the baseline function is that it will not be dependent on the action. The approach was suggested in order to utilize the stability shown in the previous assignment. In practice we see that using this function as a baseline does improve the performance. Although using the 'real' estimation of the policy's value-function, which is suggested to be the Minimum MSE estimator in Section 1.3, improves it even further, as illustrated in Fig.1. From the results shown in Fig.1, we are able to determine that using a baseline indeed improves performance and reduces variance. For example, we can see that all the algorithms performance are very similar at the beginning, while due to higher variance in the gradient estimation, the traditional REINFORCE algorithm suffer from high degradation while the others manage to converge to the

Table 1: Hyperparameters

Model	γ	α	$\mathcal{B}$	Network-Layers	$\mathcal{K}$	$\mathcal{C}$
π_θ	0.99	0.0004	X	(12,12,2)	X	X
$V + RB + Target$		0.0012	256		50	100000
V		0.0012	X		X	X



Average Score



Score

Figure 1: Scores and Average scores

solution faster. The Vanilla REINFORCE algorithm finishes at episode 1447, using baseline with RB and target-network the algorithm finishes at episode 783, and by using the policy's value-estimator as baseline the algorithm finishes at episode 518.

2 Actor-Critic

2.1 Why is using the TD-error of the value function for the update of the policy network parameters is practically the same as using the advantage estimate (of equation 1)?

Let δ be the TD Error, and it is defined as followed,

$$\delta_t = R_t + \gamma \cdot V^{\pi_\theta}(S_{t+1}) - V^{\pi_\theta}(S_t) \tag{12}$$

Let us recall that we define the advantage function as the following term,

$$\hat{A}_t = G_t - V^{\pi_\theta}(S_t) \tag{13}$$

Using the properties of the expectation w.r.t π_θ we get,

$$\mathbb{E}^{\pi_\theta}[\hat{A}_t] = \mathbb{E}^{\pi_\theta}[G_t - V^{\pi_\theta}(S_t)] \overset{(1)}{=} \mathbb{E}^{\pi_\theta}[\delta_t] \tag{14}$$

Where (1) is true since $R_t + \gamma \cdot V^{\pi_\theta}(S_{t+1})$ is an unbiased estimator of G_t. Thus, we conclude that δ_t is an unbiased estimator of $\hat{A}_t$. Now, since we are calculating the objective gradient under the expectation operator we get,

$$\nabla_\theta J(\theta) = \mathbb{E}^{\pi_\theta}\left[\sum_{t=0}^{T-1} \hat{A}_t \nabla_\theta \log \pi_\theta(a_t, s_t)\right] = \mathbb{E}^{\pi_\theta}\left[\sum_{t=0}^{T-1} \delta_t \nabla_\theta \log \pi_\theta(a_t, s_t)\right] \tag{15}$$

2.2 Explain which function is the actor and which is the critic of the model and what is the role of each one.

We model the actor as our policy network, which we term as $\pi_\theta : \mathcal{S} \to [0,1]^{|\mathcal{A}|}$, where $\mathcal{S}, \mathcal{A}$ represents the state and action spaces, respectively. The role of π_θ is to map a state to a probability distribution. Using this probability distribution, we are sampling the next action. In addition, we model the critic as our value approximation network, which we term as $V^{\pi_\theta} : \mathcal{S} \to \mathbb{R}$. The role of V^{π_θ} is to approximate $\mathbb{E}^{\pi_\theta}[\sum_{k=t}^{T-1} \gamma^{k-t} \cdot R_{k+1} | S_t = s]$ for each $s \in \mathcal{S}$ which determines the averaged discounted return from this state onward while following policy π_θ.

2.3 Results

In this section, we implement the Actor-Critic algorithm with TD-Error estimation that uses a value-function approximation instead of the actual return of every episode. By doing that, we are able to apply this algorithm even for an infinite time horizon Markov decision process. The hyperparameters used for these networks are shown in Table 2. Here our critic is used to estimate the return $\hat{G}_t$ as well as the baseline-function $b(s_t)$. Under this scenario, using a target-network and replay-buffer will cause an irrelevant return estimation of another policy $R_t + \gamma \cdot V^{\hat{\pi}_\theta}(S_{t+1})$, and may resolve in an unstable learning procedure. This is because we are constantly changing the policy and therefore, the samples which lies in the replay buffer are not relevant any more. Same applies for the target network, which doesn't provide an updated "destination" of the current policy value function. Due to the aforementioned facts, we used only the Vanilla version of value estimator. Now, as the agent explores its environment, the critic network is attempting to drive the advantage function to 0. At the beginning of the learning process, the critic will likely make large errors causing the calculated TD error to be quite incorrect. Because the algorithm starts out with the critic having no knowledge of the environment, the actor similarly can't learn much from the critic. As the critic starts to make more and more accurate predictions, the calculated TD error (Advantage) becomes more accurate. The actor is able to learn from the increasingly accurate TD error to decide if a move was good or bad. Thus, we encounter a longer training process by using this algorithm. The results of this method in comparison to the previous two methods are shown in Fig.2, the goal was achieved at episode 2561 in this scenario.

Table 2: Hyperparameters

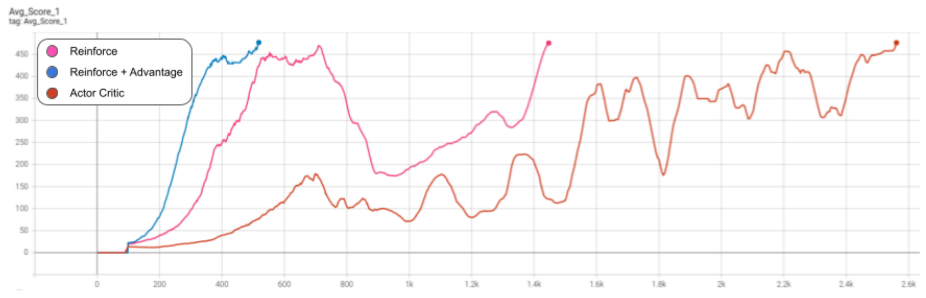
model	γ	α	Network-Layers
π_θ		0.0004	
V	0.99	0.0012	(12,12)

2.4 Further Experiments

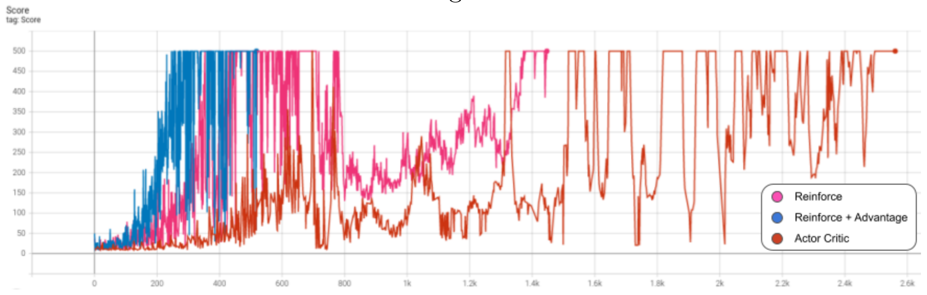
The main difference between the Actor-Critic and REINFORCE algorithms is that REINFORCE performs offline-learning, i.e., we use a full trajectory to calculate the return G_t in order to perform training steps. While Actor-Critic method perform online-learning, i.e., the episode does not need to end in order to perform training steps due to an estimate of the return $\hat{G}_t$, therefore can learn how to act in an infinite-horizon MDP environments. In this subsection we examine the influence of using N-step return estimation in the TD-error term, i.e.,

$$\delta_t = \sum_{k=0}^{N-1} \gamma^k \cdot R_{t+k+1} + V^{\pi_\theta}(S_{t+N}) - V^{\pi_\theta}(S_t) \tag{16}$$

For convenience, in an episode of length T: $V^{\pi_\theta}(S_t) = 0$ and $R_{t+1} = 0$ if $t \geq T$. We evaluate the performance for $N = 1,\ 4,\ 16,\ 32,\ 500$. Since at the beginning, the critic does not know how to evaluate each state - the return estimation based on a single step, $R_{t+1} + V^{\pi_\theta}(S_{t+1})$, is less informative than the return estimation based on several steps, $\sum_{k=0}^{N-1} \gamma^k \cdot R_{t+k+1} + V^{\pi_\theta}(S_{t+N})$, due to the fact that $V^{\pi_\theta}(S_{t+1})$ is not yet optimized, and does not yet stand for the return in S_{t+1}. Using more steps in the return estimation resolve in a better estimate of the return especially in the early steps.



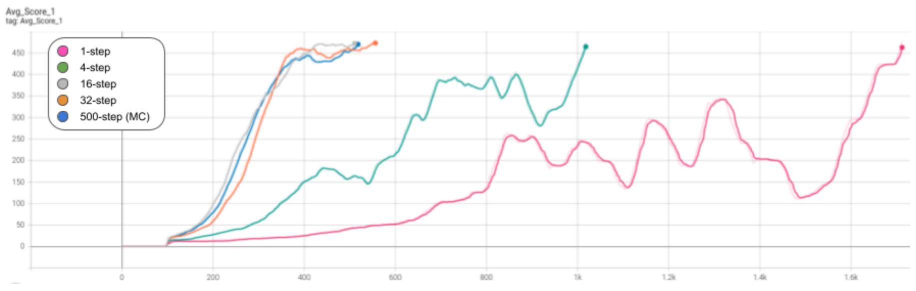
Average Score



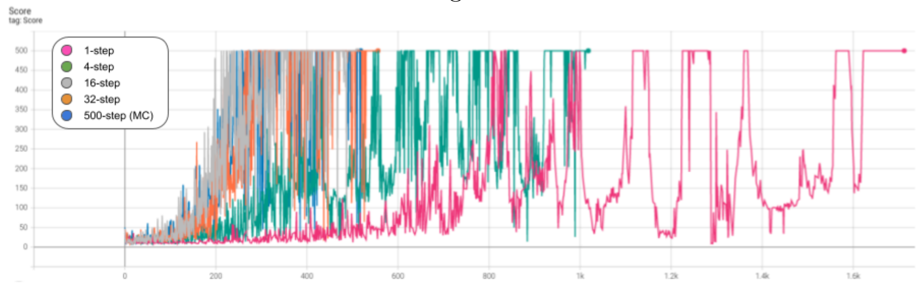
Score

Figure 2: Scores and Average scores

From Fig.3 we conclude that by manipulating the estimated return and inserting some further steps to the estimation, we can keep the advantages of online-learning, while preserving the stability granted by the better estimation of the return. The goal was achieved in episodes 1712, 1018, 511, 556, 518 for N equals 1, 4, 16, 32, 500 respectively. It should be noted that the graph of 1-step provided in this section was produced by an similar but independent experiment than shown in the previous subsection. This emphasize the variance of the results by using this method.



Average Score



Score

Figure 3: Scores and Average scores