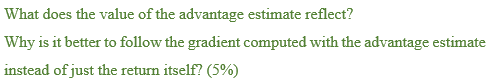
Assignment 2 – Policy Gradients Methods

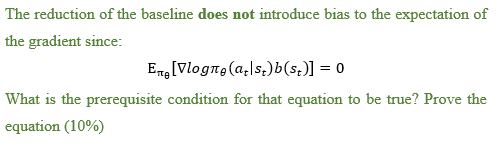
Yonatan Amaru, 316365998  
Etay Lorberboym, 314977596

# Section 1 – Monte-Carlo Policy Gradient (REINFORCE)

In this section we focus on the REINFORCE algorithm.   
The value of the advantage estimate reflects how much better an action is compared to the average performance in a given state, using a baseline set as the value . This approach is preferred for two key reasons:

1. **Variance Reduction:** It lowers the variance in policy gradient estimates by subtracting the baseline from the returns, leading to a more stable and efficient learning process.
2. **Focused Improvement:** By measuring the quality of actions relative to the policy average, it directs learning towards actions that are not just good, but significantly so, enhancing the policy’s performance more effectively.

Using the advantage leads to more stable, focused learning by emphasizing actions that significantly outperform the average, leading to faster and more precise policy improvement.

  
The equation states that the expected value of the product of the gradient of the log policy and the baseline equals zero. This means that introducing a baseline does not add bias into the gradient estimates for the policy optimization.

The prerequisite condition for the equation to be true is that the baseline is independent of the action . The baseline can be a function of but not the action chosen at state .

**Proof:**

To prove , we start by expanding the expectation in the context of policy gradient methods:

Using the log derivative trick

We rewrite the expression:

However, since we assume the is independent of , we factor out the differentiation:

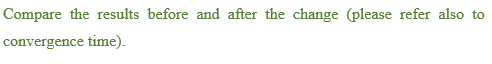
The sum of the derivatives of the probability of over all actions must sum to zero because the probabilities sum to 1, which is a constant.

Therefor we get:

This property is crucial for ensuring that the introduction of a baseline for variance reduction does not affect the unbiased nature of the policy gradient estimate.

### Section 1: Cart pole – REINFORCE with and without baseline

To incorporate the value-function approximation baseline in our code we committed the following updates:

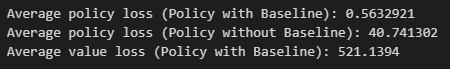
1. **Introduce a Value Network:** We added a second NN to approximate the value function . This network will predict the expected return from state under the current policy.
2. **Calculate Advantage Estimates:** Instead of using the total return directly, we calculate the advantage estimate using the predicted value from the value network as the baseline.
3. **Update both networks:** We updated the policy network using gradients calculated with the advantage estimates. The value network will be updated to minimize the difference between its predictions and the observed returns.

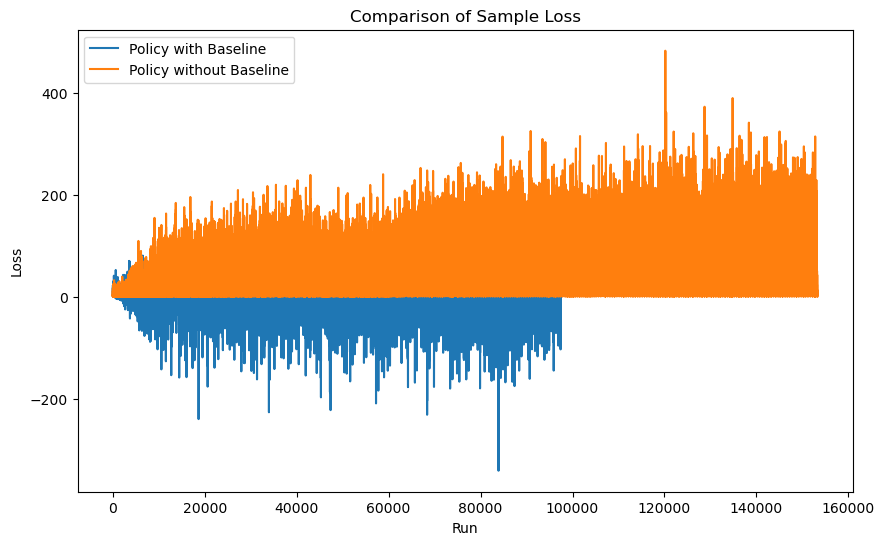
To Compare the results of REINFORCE with REINFORCE BASELINE, we tested the model with each configuration ten times, this averaging out the results and reducing the randomness.

1. **How quickly does each method solve the environment?**

****

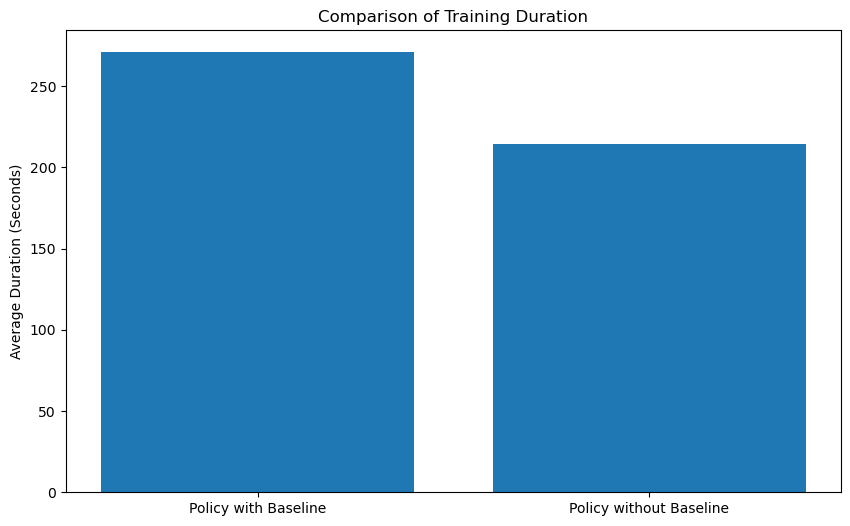
1. **How does the average loss evolve?**

****

****

1. **How efficient is the learning process in terms of duration?**

****

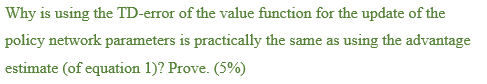
****

**Key Insights:**

* Incorporating the baseline significantly reduced the number of steps it took to learn a good policy.
* We ran the model on a CPU. Sense matrix multiplication is an expensive operation on the CPU, despite converging faster stepwise, on average the model took a longer time duration to train.
* The policy loss is much more stable with a baseline.
* For harder problem, this added stability can be a significant factor.
* We expect the time difference to be minimized when running on a GPU in an optimized setting.

# Section 2 - Advantage Actor-Critic

In this section we will focused on the Advantage Actor-Critic evolvement of policy gradients.



In the Advantage Actor-Critic method, the advantage estimate is given by

. However, in practice, the Temporal Difference (TD) error is used for updating the policy network parameters. The TD error in the context the Actor-Critic methods is defined as:

Why is using practically the same as using the advantage estimate for updating the policy parameters?

The Q-value for state-action pair can be estimated using the bellman equation as:

Substituting this into the advantage estimate, we get:

Therefore, the TD error is an unbiased estimate of the advantage function because it is derived directly from the definition of the advantage, which is the difference between the estimated Q-value and the value function .

This allows us to simplify the computation of Actor-Critic by using the TD error instead of explicitly computing the advantage estimate at each step, which would require maintaining an additional set of weights for the action-value function.



**The Actor:**

the actor represents the policy function. Its role is to select actions from the environment. The actor is responsible for the actual decision-making process and is updated based on the policy gradient, which is influenced by the TD error provided by the critic. This update is intended to increase the probability of good actions that lead to higher rewards and decrease the probability of bad actions.

**The Critic:**

The critic represents the value function. Its role is to evaluate the actions taken by the actor by estimating the value of the current policy for the given state. The critic computes the TD error, which serves to estimate the advantage function and provides the actor with feedback on the quality of the actions it has taken. The critic helps the actor to understand which actions are better or worse than what it typically expects to receive in return from the state.

**Interaction:**

The actor and critic work together to improve the policy. The interaction creates a feedback loop where the policy is continuously being adjusted based on the critic’s assessments, which in turn are based on the actor’s actions.

### Section 2: Cart Pole – Actor Critic Agent

In this section we Implement an Actor-Critic algorithm. Using policy\_gradients.py, we committed the following updates:

1. **Introduce a Value Network:** We added a second NN to approximate the value function. This network will predict the expected return from state under the current policy.
2. **Update the Actor:** Update the policy network (actor) using the TD error from the critic instead of the total discounted return.
3. **Update the Critic:** Update the value network (critic) using the TD error.



1. **How quickly does each method solve the environment?**
2. **How does the average loss evolve?**
3. **How efficient is the learning process in terms of duration?**

**Key Insights:**