CSC 498: Assignment 4

Matthew Zhang, Claas Voelcker, Liquan Wang, Prof. Animesh Garg

Released: Friday 11/18/2021 – Due: Friday 12/03/2021

Name, First Name: ______________________________

Student number: ______________________________

Total points: 50 points

To complete the exercise, you can use the tex template provided in the materials github. Insert your answers into the solution space below each question. In case you are unfamiliar with Latex, you may also submit handwritten solutions, but make sure they are clean and legible.

Submit the exercise before 23:59 pm on the due date on Quercus. To submit, please bundle your completed exercise sheet and your jupyter notebook into one zip file. Name the zip file `studentnumber_lastname_firstname.zip` and upload it on quercus.

Each student will have 3 grace days throughout the semester for late assignment submissions. Late submissions that exceed those grace days will lose 33% of their value for every late day beyond the allotted grace days. Late submissions that exceed three days of delay after the grace days have been used will unfortunately not be accepted. The official policy of the Registrar's Office at UTM regarding missed exams can be found here `https://www.utm.utoronto.ca/registrar/current-students/examinations`. If you have a compelling reason for missing the deadline, please contact the course staff as soon as possible to discuss hand in. For this assignment, you can hand in up to one week late with no penalty.

For assignment questions, please use Piazza and the office hours, but refrain from posting complete or partial solutions.

I Theoretical Analysis of Model-Based Algorithms

We will proceed to show properties for model-based algorithms. In particular, we will be discussing the Dyna algorithm [3], and comparing it the MBPO algorithm [1].

1. A variant of the Dyna algorithm is given in the following algorithm box. (3)

```
Require: Initial $Q : S \times A \mapsto \mathbb{R}$, Model : $S \times A \mapsto S \times \mathbb{R}$ (predicts next state and reward), Initial State $S$
  while Not Converged do
    Sample trajectories $\tau_1, \tau_2, \dots \tau_n$ from the environment using policy $\pi$.
    Fit Model($S, A$) using trajectories $\tau_1, \tau_2 \dots \tau_n$.
    for i = 1, ... m do
      Sample $S_{train}, A_{train}$ randomly from within $\tau_1 \dots \tau_n$
      $R_{train}, S'_{train} \sim$ Model($S_{train}, A_{train}$)
      $Q(S_{train}, A_{train}) \leftarrow Q(S_{train}, A_{train}) + \alpha \left(R_{train} + \gamma \max_A Q(S'_{train}, A) - Q(S_{train}, A_{train}\right)$
    end for
  end while
  return Q
```

   Summarize in your words how Dyna trains (1) and one benefit of this approach compared to model-free reinforcement learning (2).

   Solution:

2. Suppose that you train a policy π_1 on an environment, and then train a model M on data generated with π_1. What is an issue when you try to train a different policy π_2 when we use only M instead of the environment? (3) What does this imply about model-based algorithms like Dyna? (2) (5)

   Solution:

3. We want to generalize Dyna so that it uses the model to predict not just the next state-reward S', R, but an entire trajectory $S_{t+1}, R_{t+1}, S_{t+2}, R_{t+2} \dots$. (3)

   First, we need to consider the following problem. Suppose that there are two models D_1 and D_2 trained from two different policies π_1, π_2.

   Then, suppose we wanted data from D_2, but could only sample trajectories from D_1. We can sample long trajectories or short trajectories. Of the two, which one is likely to be more similar to data sampled from D_2? Briefly justify your answer.

   Solution:

4. One way of generalizing Dyna to longer trajectories is $MBPO$. (4)

A simplified version of the $MBPO$ algorithm is shown below.

```
Require: Initial policy $\pi$, model $M$, Empty dataset $D$
  while Not Converged do
    Sample trajectories $\tau_1, \tau_2, \dots \tau_n$ from environment using policy $\pi$.
    Train model $M$ using trajectories $\tau_1 \dots \tau_n$.
    for i = 1, ... m do
      Sample $S_t$ randomly from states found in trajectories $\tau_1, \dots \tau_n$.
      Roll-out a trajectory $\sigma = \{S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, \dots S_{t+k}, A_{t+k}, R_{t+k}\}$ of length
      $k + 1$ using the model $M$ and the policy $\pi$, starting from $S_t$.
      Train $\pi$ using the trajectory $\sigma$.
    end for
  end while
  return Final policy
```

Explain how the algorithm generalizes Dyna to trajectories (1). Using your answers from Parts 2 and 3, explain why smaller k may be preferred compared to larger k when we roll-out trajectories. (3)

Solution:

II Policy Gradients revisited

1. *Estimator variance* In class, we discussed that the estimator for the policy gradient has a high variance. In this exercise we want to quantify how large this variance is. (5)

   Give an unbiased estimator of the variance of the gradient estimator of vanilla policy gradient with N samples.

   Solution:

2. *Baseline variance* We can add a baseline to reduce the variance of the estimator without changing the mean. State the requirements for a baseline to not bias the estimator. (2) From the previous derivation, derive the optimal baseline for the policy gradient estimator of an actor-critic algorithm. (5) (7)

   Solution:

3. What is the challenge of implementing this baseline in practice? (3)

   Solution:

III Reinforcement Learning in Practice: MuZero

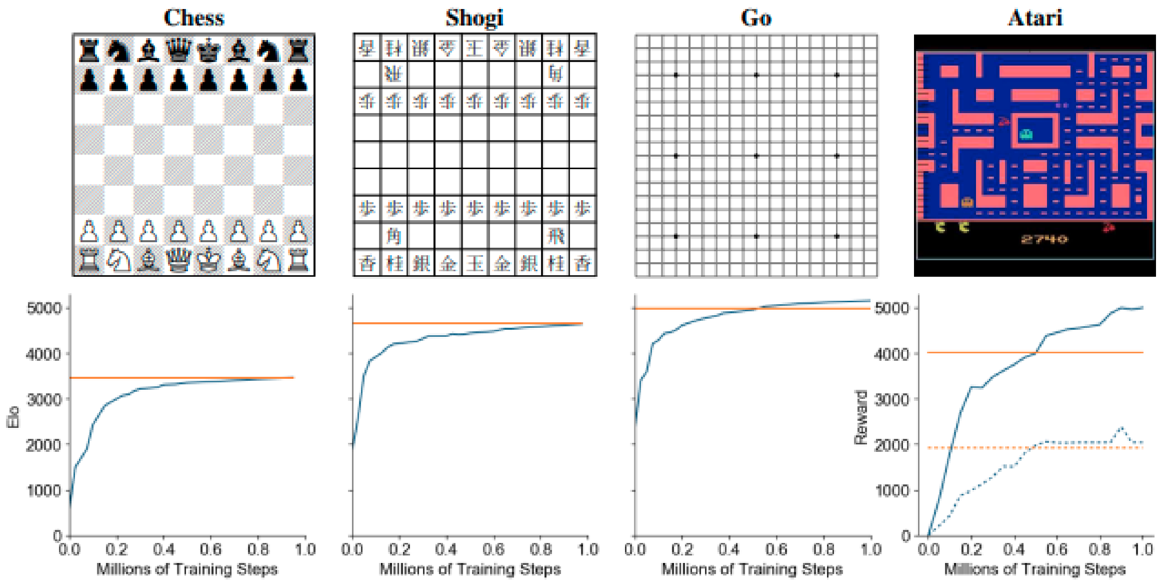

Figure 1: MuZero's Performance on Multiple Game Benchmarks. From [2].

One of the most exciting developments in Model-based Reinforcement Learning in the last few years was the extension of the seminal AlphGo algorithm to a model based architecture called MuZero. MuZero achieves state of the art performance both on classic board games such as chess and Go as well as the Arcade Learning Environment.

1. Please summarize (in less than one page) the general approach adopted by MuZero, with proper reference to scientific literature. In your summary, also state the advantages of using MuZero over AlphaZero. (20)

Solution:

References

[1] Michael Janner et al. “When to trust your model: Model-based policy optimization”. In: *arXiv preprint arXiv:1906.08253* (2019).

[2] Julian Schrittwieser et al. “Mastering atari, go, chess and shogi by planning with a learned model”. In: *Nature* 588.7839 (2020), pp. 604–609.

[3] Richard S Sutton. “Dyna, an integrated architecture for learning, planning, and reacting”. In: *ACM Sigart Bulletin* 2.4 (1991), pp. 160–163.