Giorgio Mendoza

RBE595-S24-S04

Dr. Dadkhah Terani

Reinforcement Learning

Model-based RL Assignment

1- What is “planning” in the context of Reinforcement Learning?

In this chapter's context, planning in reinforcement learning is a process where a model of the environment helps in creating or refining decision-making policies. This involves:

Model Usage: The model predicts responses to actions, helping foresee possible outcomes.

Experience Simulation: It allows the agent to simulate potential future scenarios.

State-Space Search: Searching within this model to find optimal actions or policies.

Computing Value Functions: Estimating future rewards to shape decision strategies.

Learning Integration: Merging planning with both model-based and model-free learning methods.

Dyna Architecture: Demonstrates how planning integrates with other aspects of reinforcement learning.

Essentially, planning uses simulated scenarios to enhance decision-making in a reinforcement learning environment.

2- What is the difference between Dyna-Q and Dyna-Q+ algorithms?

The Dyna-Q and Dyna-Q+ algorithms are both approaches in RL that integrate planning, learning from real experiences, and acting. However, they have a key difference in how they handle exploration in environments where certain transitions are only discovered after a long period of exploration.

Dyna-Q

Basic Framework: Dyna-Q combines direct experience with simulated experiences (from a learned model) for updating value functions and policies.

Model Learning: It learns a model of the environment from real experiences.

Planning: Uses this model for simulated experiences, applying Q-learning updates to these simulations.

Exploration: Relies on standard exploration strategies like ε-greedy to explore new states and actions.

Dyna-Q+

Handling Unexplored Transitions: Dyna-Q+ extends Dyna-Q by encouraging exploration of states and actions that have not been tried for a long time.

Reward Bonus for Unexplored Transitions: It introduces a small reward bonus for state-action pairs that have not been visited in a while. This bonus decreases over time as the state-action pair continues to remain unvisited.

Exploration and Exploitation: This reward bonus system incentivizes the agent to explore lesser-known parts of the environment, thus balancing exploration and exploitation more effectively, especially in changing environments.

Effectiveness in Dynamic Environments: Dyna-Q+ is particularly effective in environments that change over time, as it encourages revisiting previously explored states and actions, which might now yield different outcomes.

In summary, while both Dyna-Q and Dyna-Q+ integrate planning with learning and acting, Dyna-Q+ specifically addresses the challenge of ensuring sufficient exploration, especially in environments that can change or where certain pathways are only discovered after extensive exploration. It does this by adding a mechanism to reward the revisiting of less frequently visited state-action pairs.

3- Model-based RL methods suffer more bias than model-free methods. Is this statement correct? Why or why not?

“Model-based RL methods suffer more bias than model-free methods" can be seen as partially correct due to the following reasons:

Model-Based RL Bias: These methods use a model of the environment for planning and decision-making. If the model inaccurately represents the environment, it can lead to biased decisions and policies. The bias here is due to inaccuracies in the environmental model.

Model-Free RL: Since model-free methods don't rely on a model, they avoid this kind of bias. However, they might face other biases, such as those from policy representation or learning algorithms, and typically require more real-world interactions to learn effectively.

Trade-Offs: Model-based methods have the advantage of efficiency through planning, but this comes with the risk of bias if the model is imperfect. Model-free methods avoid model bias but can be less efficient and have different kinds of biases.

While model-based RL can be more prone to bias due to potential inaccuracies in the model of the environment, model-free RL has its own challenges and biases, and the choice between them depends on the specific task and environment.

4- Model-based RL methods are more sample efficient. Is this statement correct? Why or why not?

Model-based RL methods are more sample efficient and are generally correct because:

Model-Based RL and Sample Efficiency

Use of a Model: Model-based RL methods involve creating a model of the environment. This model is used to simulate and predict outcomes of different actions without the need for actual interactions with the environment.

Less Real-World Interactions: Because these methods can 'imagine' or simulate future states, they often require fewer real-world interactions (samples) to learn effective policies. They can explore various scenarios and learn from them without actual trials, making them more sample efficient.

Planning and Decision Making: Model-based methods can use planning techniques to foresee the outcomes of actions over multiple steps, which allows for more strategic decision-making based on fewer real-world experiences.

Model-Free RL and Sample Efficiency

Direct Learning from Interaction: Model-free methods learn directly from real interactions with the environment. They typically require more interactions (samples) to accurately learn and optimize their policies.

No Predictive Model: Without a model to simulate outcomes, each piece of knowledge about the environment must come from actual experience, which can be less efficient in terms of the number of samples needed.

Trade-Offs

Model Accuracy: The efficiency of model-based methods depends on the accuracy of the model. If the model poorly represents the environment, the efficiency can be compromised.

Complexity and Computation: Model-based methods might require more computational resources for maintaining and updating the model.

Model-based RL methods are generally more sample efficient because they leverage a model to learn from simulated experiences, reducing the need for extensive real-world interactions. However, this efficiency relies on the accuracy and quality of the model used. In contrast, model-free methods, while often less sample efficient, avoid the complexities and potential inaccuracies of maintaining a model.

5- What are the 4 steps of the MCTS algorithm? How does MCTS balance exploration/exploitation?

The MCTS algorithm balances the exploration of new, uncharted moves with the exploitation of moves known to be effective. The algorithm typically consists of four main steps:

Selection: Starting at the root node (the current state), the algorithm selects child nodes down the tree until it reaches a node that represents an unexplored move or an end state. The selection is typically guided by a policy that balances exploration and exploitation, often using the Upper Confidence Bound (UCB) formula, particularly the UCB1 or a variation of it.

Expansion: If the node reached has unexplored moves, one of these moves is chosen, and a new child node is added to the tree representing this move.

Simulation: From the new node, the algorithm performs a simulation or "rollout". This is a play-through to the end of the game (or a certain depth), using random or lightweight moves, to obtain a quick estimation of the node's value.

Backpropagation: After the simulation, the outcome is used to update the information in the nodes visited during the selection phase. Typically, this involves updating the values that guide the selection process in future iterations, like the win/loss ratio for each node.

Balancing Exploration and Exploitation

MCTS balances exploration and exploitation through its selection step, often utilizing the UCB1 formula in the form of:

UCB1 = $w_i/n_i + C\sqrt{(lnN_i/n_i)}$

Where:

w_i = number of wins after the i-th move

n_i = number of simulations after the i-th move

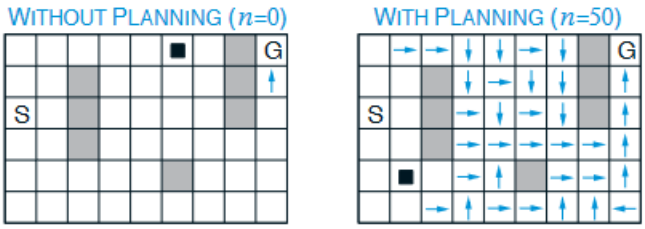
N_i = total number of simulations after the parent node move

C = exploration parameter (higher values increase exploration)

This formula ensures that moves that haven't been explored much (low n_i) are more likely to be chosen (exploration), while also considering the moves' success rates (exploitation). The balance between these two is crucial for the effectiveness of MCTS, and it can be adjusted through the C parameter. This dynamic allows MCTS to explore potentially good moves that haven't been tried often, while also exploiting moves known to be effective based on past simulations.

6- Exercise 8.1 of the textbook (page 166)

Exercise 8.1 The nonplanning method looks particularly poor in Figure 8.3 because it is a one-step method; a method using multi-step bootstrapping would do better. Do you think one of the multi-step bootstrapping methods from Chapter 7 could do as well as the Dyna method? Explain why or why not. □



The figure shows that the agent with planning (n=50) has developed a more directed policy towards the goal than the agent without planning (n=0).

Multi-step Bootstrapping Methods:

Multi-step bootstrapping methods, as discussed in Chapter 7 of the book, combine the ideas of Monte Carlo methods with temporal difference learning. These methods use multi-step updates based on TD targets involving several time steps, rather than the single-step updates used in one-step TD methods like Q-learning.

Comparison with Dyna-Q: Dyna-Q uses planning steps that are akin to additional learning updates based on simulated experience generated by the model. This allows for faster

propagation of reward information throughout the state space, as many updates can be made between actual interactions with the environment.

Multi-step bootstrapping methods could potentially do well compared to a Dyna method without planning (n=0) because they also propagate information from rewards over multiple steps, which can lead to a more informed and effective policy than one-step methods.

However, Dyna-Q with planning (n=50) might still outperform multi-step methods, particularly in environments where the model is accurate. This is because the planning component in Dyna-Q allows for rapid policy improvement without additional interactions with the environment. The model enables the agent to benefit from learning opportunities that are created by simulating experiences based on the known dynamics of the environment.

Effectiveness of Multi-step Bootstrapping:

The effectiveness of multi-step bootstrapping methods relative to Dyna-Q would largely depend on the environment's complexity and the accuracy of the model. If the model is highly accurate, Dyna-Q can exploit this to effectively plan and improve its policy.

In contrast, if the environment is highly stochastic or the model is not accurate, multi-step bootstrapping methods may perform better because they rely on actual experienced transitions, avoiding the propagation of model bias.

7- Exercise 8.2 of the textbook (page 168)

Exercise 8.2 Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments? □

To address this question, we must consider the primary difference between Dyna-Q and Dyna-Q+: the exploration bonus. Dyna-Q+ is designed to encourage exploration by providing a bonus reward for state-action pairs that have not been tried in a while. This bonus is calculated

using a formula that takes into account the time since the state-action pair was last tried. Specifically, if the normal reward for a transition is r, and the transition has not been tried in τ time steps, then the bonus reward would be $r + k\sqrt{\tau}$, with k being a small positive constant. So Dyna-Q+ could perform better in both cases because:

First Phase (Blocking Task): Initially, the environment does not change, and both Dyna-Q and Dyna-Q+ learn the optimal path. However, Dyna-Q+ with its exploration bonus is more likely to explore alternative paths as well. This means it's not only learning the current best strategy but also gathering information about other possible strategies. When the environment changes (the path gets blocked), Dyna-Q+ has already explored other options and can quickly adapt to find a new optimal path.

Second Phase (Shortcut Task): When the shortcut opens up, Dyna-Q continues exploiting the longer path because its model has no knowledge of the shortcut, and without an incentive to explore, it sees no reason to change its policy. Dyna-Q+, on the other hand, has been encouraged to keep exploring due to the exploration bonus. It's more likely to discover the new shortcut because the exploration bonus has made it less certain about its model, prompting it to try actions that have not been selected recently, including potentially the new shortcut.

In both phases, the key to Dyna-Q+'s superior performance is its intrinsic motivation to explore due to the exploration bonus. Even when the current model seems to provide a good strategy, Dyna-Q+ continues to seek out new information that could lead to better rewards, making it more robust to changes in the environment and enabling it to discover new opportunities faster than Dyna-Q.