Intro To AI
(CSCI - 6600)
Fall 2022

A Comparison Between Model-Less And Model-Based RL Approaches

Presented by:

Rajdeep Bhattacharya

Adesh Agarwal

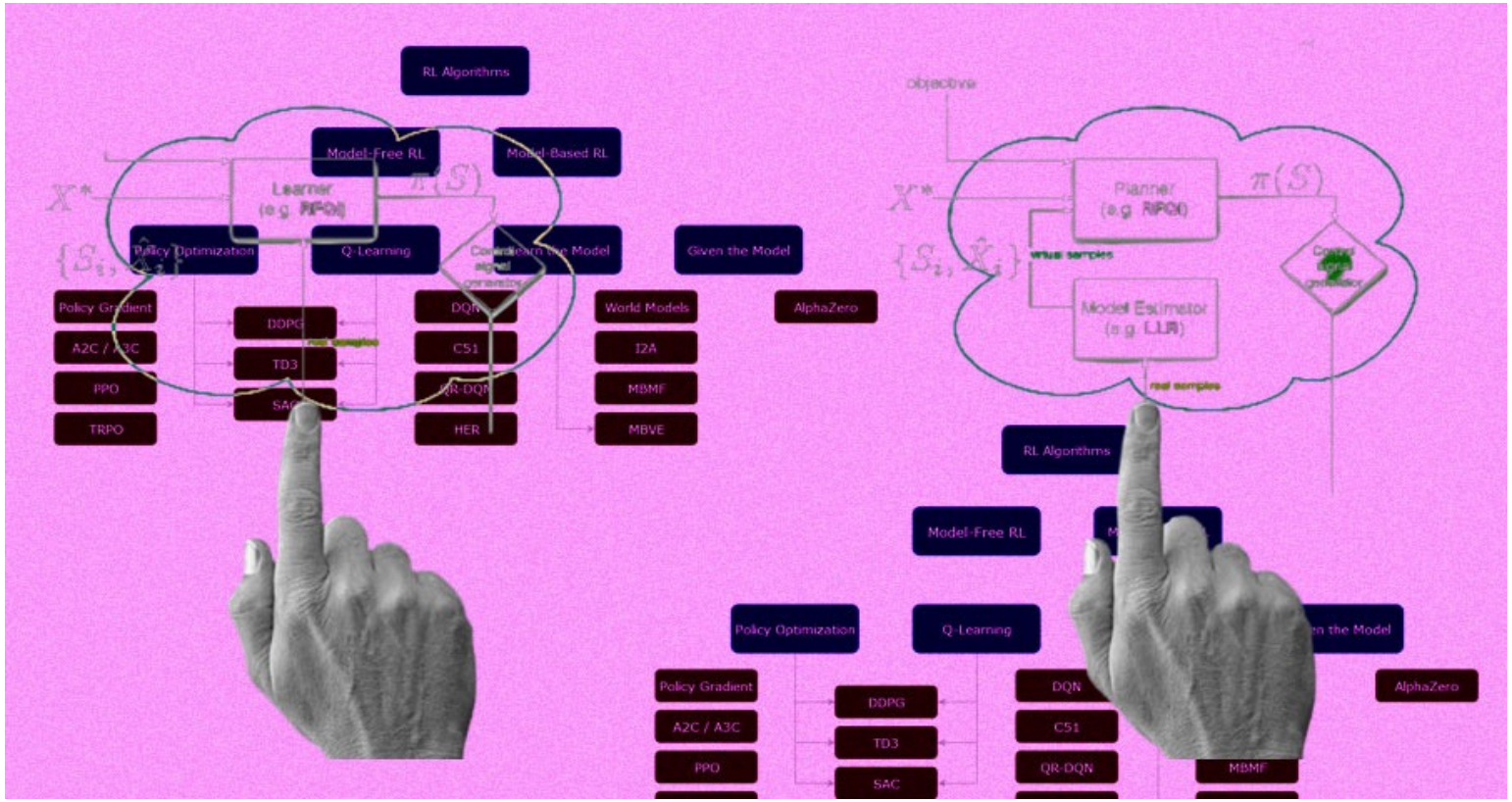
Pranay Udaygiri

Rajwinder Singh

YouTube : https://www.youtube.com/watch?v=_055zRVdDZ0

Reinforcement Learning

Reinforcement learning systems in AI learns from its environment through interaction and evaluates what it learns in real-time for example self-driving or driverless cars, different games, etc.



The Project Idea

- In this project we are comparing Model-Free and Model-Based reinforcement learning processes through the online game of tennis.
- To explore Model-Free RL, we are using a mathematical approach to playing tennis. And for the Model-Based RL we are using Tennis game played through the Deep Q Network.
- We have used NumPy, PyGame, Keras and Tensor Flow Libraries. As Tensor Flow is a heavy one it took some time to install.
- We had major problem to run Tensor Flow on Mac (from previous experience we knew it can create issues sometimes), we arranged Windows Laptop, but in the end the issue was resolved.
- We were thinking of running both the algos parallelly but had to scrap that idea as we thought the laptops might not be able to take the load.

The Process

Like any RL problem we have the Agent, the Environment, the Rewards and the Policy.

Agent is the program which controls the object of concern (for instance, a robot). **Environment** defines the outside world in the program, everything the agent(s) interacts with. **Rewards** gives us a score of how the algorithm is performing with respect to the environment. It's stated as 1 or 0. 1 means that the policy network made the right move, 0 means wrong move. And the **Policy** is the algorithm used by the agent to decide its actions. This where the **Model-Free** or **Model-Based** ideas come into play.

Every RL solution starts with creating an environment for the agent, then we build a policy network that controls the agent, then the policy is evaluated by considering if the corresponding action resulted in Score-1 (For gain) or Score-2 (For Loss).

In this project policy is the point that we are going to focus on. Policy can be Model-Free or Model-Based. While creating the RL solution we focus on optimization of the policy network through policy gradient. **Policy Gradient Algorithms** try to optimize the policy to get more rewards.

The Process

Many of these algorithms (especially the ones with discrete actions) are based on **Markov decision processes (MDPs).** MDPs have fixed number of states, the agent randomly moves from one state to another at each step and the probability for it to move from a specific state to another is fixed.

The agent in the discrete action algorithms has no initial clue on the next transition state or the rewarding principle, so it must explore all possible states to begin to decode how to adjust to a perfect rewarding system, hence it uses **Q-Learning**.

The Q-Learning algorithm uses the **Q-Value Iteration**. Q-Values are optimal estimates of the State Action Value in an MDP, as the agent has no prior knowledge it uses the Q-Values.

It is a belief that Q-Learning doesn't scale well for bigger MDPs with many states and actions, in that case we approximate the Q-Value of any state-action pair (s,a) which we call **Approximate Q-Learning**.

DeepMind proposed the use of **deep neural networks**, which work much better for complex problems without any feature engineering. A deep neural network which estimates Q-Values is called a **deep Q-network (DQN)** and using DQN for approximated Q-learning is called **Deep Q-Learning**.

Model-Free RL Vs Model-Based RL

Like I said earlier RL algorithms can be mainly divided into model-based and model-free.

In the Model-Based as it sounds, the agent tries to understand its environment and creates a model for it. There preferences take priority over the consequences of the actions which means that the greedy agent always try to perform an action that will get the maximum reward irrespective of the consequences.

On the other hand, in the Model-Free, the agent seeks to learn the consequences of its actions through experience with the help of algorithms such as Policy Gradient, Q-Learning, etc. In other words, such an algorithm will carry out an action multiple times to adjust the policy for optimal rewards, based on the outcomes.

This results in different applications for the above two algorithms, like a model-based approach may be the perfect fit for playing chess or for a robotic arm in the assembly line of a product, where the environment is static and getting the task done most efficiently is our main concern. However, in the case of real-world applications such as self-driving cars, a model-based approach might prompt the car to run over a pedestrian to reach its destination in less time, but a model-free approach would make the car wait till the road is clear.

The Environment

To better analyze and understand this in our Project we created an application using the above two approaches. We tried to build a Model-Free and a Model-Based RL rendition for tennis games. To build the model we didn't build the environment but have imported one.

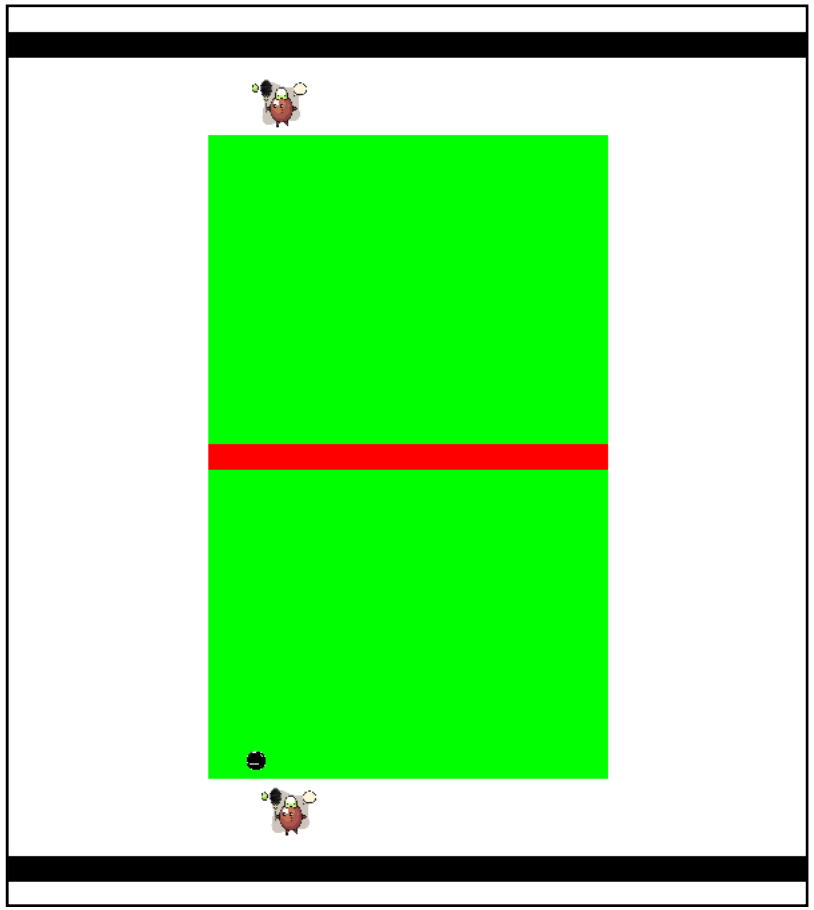
Now there are many kinds of RL environments like Deterministic environment, Stochastic environment, Fully observable environment, Partially observable environment, Discrete environment, Continuous environment, Episodic and non-episodic environment, Single and multi-agent environment, etc.

We created the **Tennis environment** which is a Discrete environment for both Model-Free and Model-Based RL system.

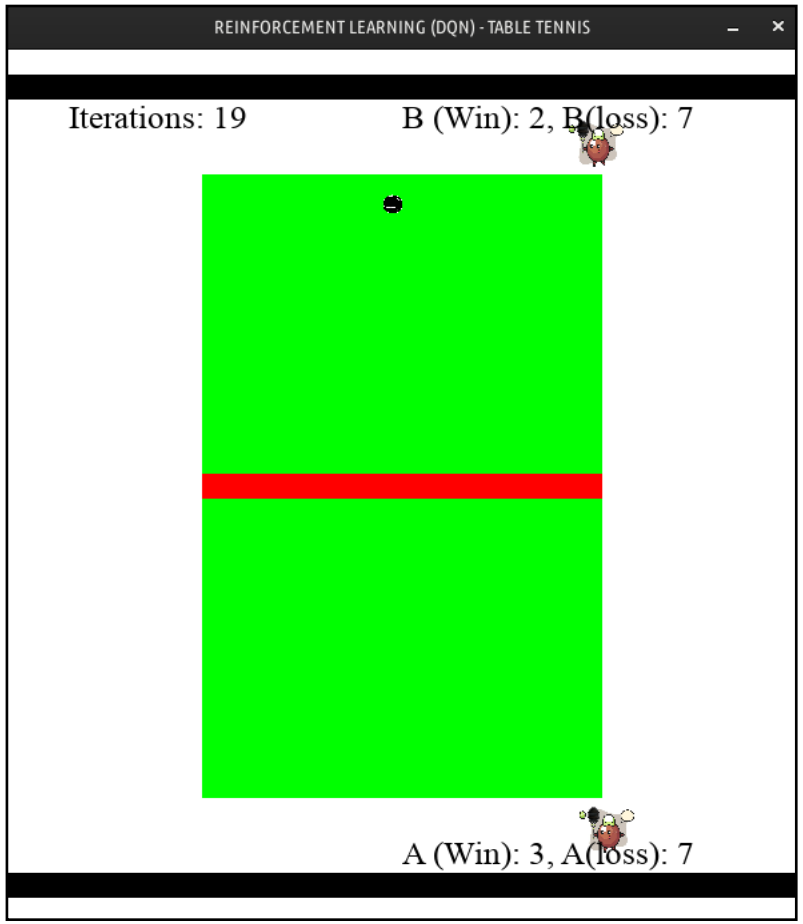
A tennis game requires the following:
>> 2 players which implies 2 agents.
>> A tennis lawn – main environment.
>> A single tennis ball.
>> Movement of the agents left-right (or right-left direction).

The Environment

Tennis Model-Less Environment



Tennis Model-Based Environment



Comparison And Evaluation

Having played this game model-free and model-based, here are some differences that we need to be aware of:

1. In Mode-Free the rewards are not accounted for (since this is automated, reward = 1) but in Model-Based rewards are accounted for.

2. In Model-Free no modelling (no decision policy is required) but in Model-Based modelling is required (policy network).

3. Model-Free doesn’t require the use of initial states to predict the next state but Model-Based requires the use of initial states to predict the next state using the policy network.

4. In Model-Free the rate of missing the ball with respect to time is zero but in Model-Based the rate of missing the ball with respect to time approaches zero.

5. Variance - A discrete policy produces more or less same in different runs so less variance over different runs.

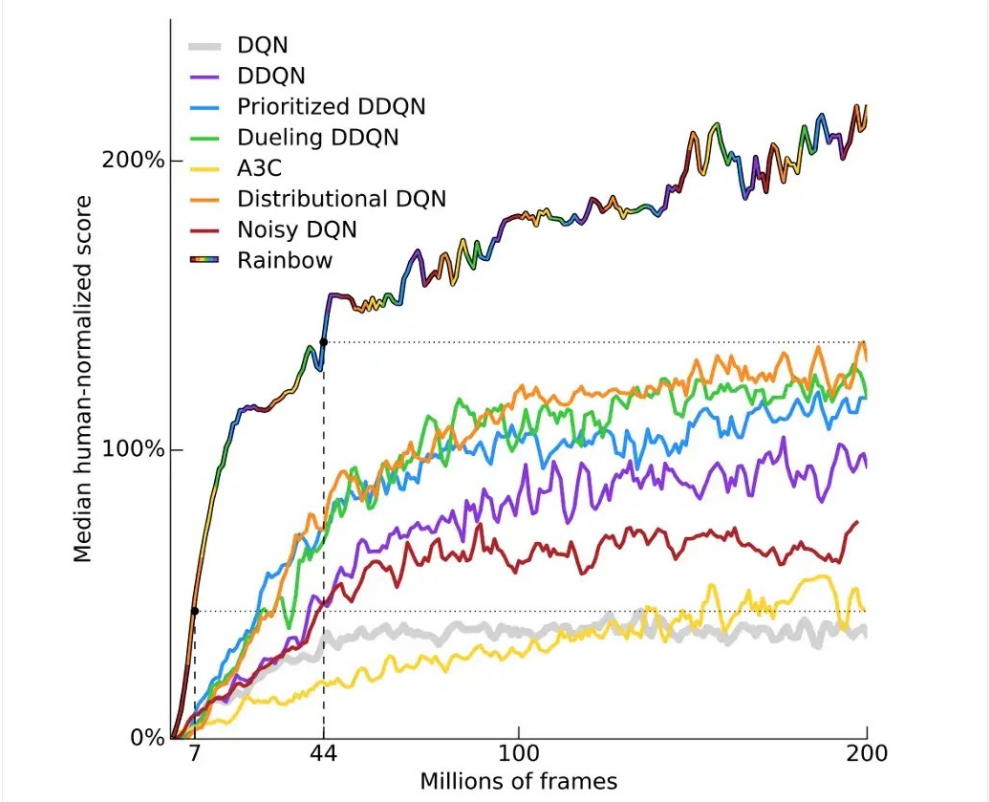
Comparison And Evaluation

6. Assumption 1 - In value fitting methods and some model-based methods, we assume the action space and/or the state space is continuous.

7. Assumption 2 - QN is mainly for low-dimensional discrete control space.

8. DQN is an off-policy method, samples are drawn from a replay buffer to fit the Q-value. In DQN, the replay buffer improves both stability and sample efficiency.

Here is the plot on the performance of many model-free methods from the Internet. As noted, it easily takes 80+ million frames for many advanced methods to outperform the human expert in playing Atari games. The graph below is normalized. An average human expert score at 100% below.



Conclusion

There were a few problems we ran into, like earlier I talked about the Tensor Flow package, again there was a minor issue with the PyGame version we were initially using.

Tennis might be simple compared to self-driving cars, but hopefully this example showed us a few things about RL.

The main difference between model-free and model-based RL is the policy network, which is required for model-based RL and unnecessary in model-free.

It’s worth noting that oftentimes, model-based RL takes a massive amount of time for the DNN to learn the states perfectly without getting it wrong.

It is more like Tenured Job and Gig hehe, the first one needs more preparation but is more reliable.

But every technique has its drawbacks and advantages, choosing the right one depends on what exactly you need your program to do.

Conclusion

	Strength	Weakness
Model-Based	It can be self-trained and therefore more scaleable. Sample efficient. The learned dynamic model is transferable.	But need retraining to optimize the controller again for a specific task. Not optimizing the policy directly. More assumptions and does not work with all tasks. A model can be much complex to train than a policy.
Model-Less	It has fewer approximations and assumptions that work with a wider spectrum of tasks. Good at learning complex policies. The policy can be generalized better in some tasks.	Less sample efficient. Vulnerable to overfit with a complex model. Lead to poor decisions.

Model-Based Vs Model-Less RL Methods – Strength And Weakness

Conclusion

We could have done it more robust study had we had more time. Among the different ML options some of us have already become fan of RL and we hope to explore more about it in the future.

Hopefully, you learned more about the two kinds of RL approaches from our project. Thank you.