DS 5004: Applied Reinforcement Learning

Reinforcement Learning Journal

Bruce McGregor (BM3PK)

Journaling Instructions

The purpose of this exercise is to track your learning and growth through this course.
You will add an entry for each Module (ideally on a weekly basis).
You will submit your journal over the course of the term and it will grow with your new entries.

Each entry should answer these questions:

- What were some things you learned in the module?
- What do you think were the most important concepts?
- What was challenging for you? How can you learn it better?
- Which parts did you enjoy?

Constructive feedback is also fine (e.g., Exercise X, Part Y wasn't clear) but negative criticism should be avoided.

Each entry should:

- Use full sentences to show good writing and clear thought process
- Be limited to one or two paragraphs (it should be fairly brief)

File format: Use a Word doc or similar. Recycle the same file throughout the course.

Module 1

Reinforcement Learning Fundamentals

What were some things you learned in the module?

One of things I learned is that RL ability to achieve the effects of planning and lookahead without using a model of the opponent in the tic tac toe example. In addition, it does not need to do an explicit search over possible sequences of future states and actions. I think that is incredible, and hard to grasp. So, I am looking forward to exploring further how RL achieves such things.

What do you think were the most important concepts?

I think understanding the four main elements of TL system, that is the policy, reward signal, a value function, and model (optional). Understanding the difference between reward and value is a ky concept to grasp. It is also important to understand how RL allows for individual states to be evaluated and adjust policy. This contrasts with evolutionary methods, where the individual states are not evaluated, only the final result.

What was challenging for you? How can you learn it better?

Retention is a challenge. I agree that learning is recursive. I find that during my first reading through the text (even when concentrating hard), I struggle to recall it later. I find that re-reading after the lecture helps and highlighting and underlining key concepts helps. I iteratively must review the previous material to ensure retention. Also, I still use hard copy textbooks that allow me to flip pages back and forth to really absorb the material. I find it hard to replicate this on Kindle.

Which parts did you enjoy?

I enjoyed the cart pole lab. It was fun and challenging at the same time. I also enjoyed reading about this history of RL and what other domain it pulls from, such as control theory and statistics.

Module 2

K Armed Bandits and MDP

What were some things you learned in the module?

I learned about the K-armed bandit problem, and how it assumes a single state, which simplifies the problem. I learned how the incremental implementation is applied in RL to save computation space, by only requiring us to store the value of Q_n and n, and only the small computation for each new reward. I also learned about the Markov process, and how it provides the mathematical foundation for RL. I learned that the Bellman equation used recursion and is based around dynamic programming.

What do you think were the most important concepts?

Of the important concepts is balancing exploitation with exploration, and how this translatesinto application of greedy and e-greedy approaches. Another important concept is the difference between stationary and non-stationary problems and using different approaches to setting the step-size parameters. The two biggest take away for me was state value function vs the action value function, where the latter is used to adjust the agent's policy, and the former is a weighted summation of all the possible action in a state.

Some other important concepts are how the gains function works and the discounting factor. I found it challenging to understand how an infinite sum can provide a finite value, so I had to dig into the math behind the geometric series to really validate how it works. I learned this better by exploring beyond the text.

What was challenging for you? How can you learn it better?

I struggled a bit to understand the concepts of stochastic approximation theory and the two conditions that must be satisfied to assure convergence. In particular, understanding why both conditions are met for the sample average case, but not for constant step size parameter. When I struggle with concepts in the text, Chat GPT has been a big help.

Which parts did you enjoy?

I enjoyed the hands-on examples in the notebook, which put the concepts into action, and helped reinforce the readings. I enjoyed the lecture and explanation on the Bellman equation by breaking it down and using the backup diagram to understand how it works.

Module 3

Solving MDPs

I began learning about ways to solve MDPs. The first was dynamic programming, which is a good foundation, but not used in practice. It requires a perfect model of the environment, which is not practical in many real-world problems. As a side note, I had just completed the Algorithm class, where I first studied dynamic programming, so it was nice to see how it relates in another class. I also learned the key difference in Monte Carlo is that it samples (often from a simulator) and does not require a model of the environment

Some of the most important concepts I feel are the value iteration and policy iteration algorithms. I found these require a lot of study and concentration to fully comprehend. The lab offers a useful application to see how value iteration worked in practice. This was useful and helped me understand the algorithm over the pseudocode example in the textbook.

I enjoyed reading about the Blackjack Examples in the Chapter on Monte Carlo. We will be discussing this in class on Thursday, which should provide additional reinforcement.

Module 4

Q Learning

The main takeaway from Module 4 for me was learning the difference between **on-policy** and **off-policy** learning, and how these concepts are implemented through **SARSA** and **Q-learning**. Both methods build upon the foundation of **temporal-difference learning**, which combines ideas from Monte Carlo methods and dynamic programming. Understanding this foundation helped me see how agents can learn directly from experience without needing a full model of the environment.

The **Cliff Walking lab** was especially helpful in solidifying my understanding of the **Q-learning algorithm**. Seeing the agent iteratively learn to avoid the cliff by updating Q-values through repeated episodes made the concept of off-policy learning much more concrete. It also illustrated how Q-learning's reliance on the maximum future Q-value drives the agent toward an optimal policy without explicitly following it during training.

The most difficult concept for me to fully grasp was the distinction between on-policy and off-policy learning. I also found **importance sampling** challenging, especially when trying to understand how it is used to correct for the mismatch between the behavior policy and the target policy. I now understand that importance sampling reweights returns from the behavior policy distribution to estimate what would have happened under the target policy.

What I'm still trying to understand better is **why Q-learning avoids the need for importance sampling**, and how this makes it more efficient in certain settings. I think as I move forward into later modules, continuing to work through examples and visualizations of these algorithms will help the intuition behind these ideas click more naturally.

Addendum: As I have learned in Module 6, I now have learned that the Q learning introduces bias from use of the max operator, and that was one of the motivations behind Double Q-learning.

Module 5

Deep Q Networks

In Module 5, I learned how Deep Q-Networks (DQNs) extend traditional Q-learning by using neural networks to approximate the Q-function, making it possible to handle large or continuous state spaces. This module helped me understand why simply combining Q-learning with a neural network is unstable, because consecutive samples are highly correlated and the targets shift as the model learns. The *"Playing Atari with Deep Reinforcement Learning"* introduced two key innovations to solve this: **experience replay**, which randomizes past experiences to break correlations, and **target networks**, which stabilize learning by holding parameters fixed for a period.

I really enjoyed reading the Atari paper and found the outcomes quite impressive, in that the agent directly learned from raw pixels only. This felt like a major leap in how we think about intelligence. I was especially intrigued by the choice of games. The paper featured mostly fast-action titles like Breakout and Pong, and found myself wondering why more exploratory games such as *Adventure* (one of my all-time favorites) were not included. I suspect the reason is that those games require long-term planning and deal with sparse rewards, which would have challenged early DQNs.

Coding and visualizing results will help me internalize how these algorithms learn over time.

Module 6

Deep Q-Networks Extensions 1

I am finding journaling to be a very effective learning tool as I reflect on what I've learned. Writing things down helps me internalize the concepts and makes it easier to recall the main takeaways later. Thanks for this exercise. I think other professors should consider adopting it. I also wanted to say that the course content is very well organized, and I appreciate the use of weekly agendas to recap what we are focusing on each week and what deliverables are coming due.

It has been enjoyable to see the progression and innovation that have been brought to RL field through these research papers. They serve as a nice bridge from the textbook that is rooted in the theory, to how the theory has been applied and improved upon over time with new discoveries.

In this module, I learned about some of the drawbacks of Deep Q-Networks (DQNs), particularly their tendency to **overestimate Q-values** due to the **max operator**, which can bias learning toward inflated estimates. The **Double Q-Learning** paper helped me understand how separating the action selection and evaluation steps across two networks effectively reduces this overestimation bias. Reading the original paper made the concept much clearer, as I could see how researchers identified the problem and designed a practical solution.

The **Prioritized Experience Replay (PER)** paper built on these ideas by addressing a different challenge, that is how to make learning more efficient by replaying experiences that are more informative. Unlike uniform stochastic sampling, PER prioritizes experiences with higher temporal-difference errors. However, this introduces sampling bias, which is corrected through **importance sampling**. This balance between prioritization and correction struck me as an elegant way to maintain fairness while improving learning efficiency.

One of my key takeaways from this module is that **bias is a major theme across these DQN extensions**, but the biases they address are different yet complementary. Double Q-learning reduces *overestimation bias* in the target values, while PER corrects *sampling bias* introduced by non-uniform replay. Together, these techniques show how researchers iteratively refine reinforcement learning algorithms to make them more stable and reliable. Reading both papers reinforced how innovation in this field often comes from building thoughtfully on earlier discoveries.

Module 7

Deep Q-Network Extensions 2

In Module 7, I learned how the dueling network architecture builds on what we've already done with Deep Q-Networks by changing how the network represents value. Until now, most of what I studied, i.e, Double DQN and Prioritized Experience Replay, focused on improving the learning process: making updates more stable, reducing bias, or improving sample efficiency. This module was different because it focused on the structure of the network itself and how separating state value from action advantage can make learning faster and more efficient.

The main idea that stuck with me is that not every state requires the agent to learn a separate value for every action. In a lot of states, most actions are basically the same. The dueling architecture takes advantage of this by splitting the network into two streams. The first stream learns how good the state is ($V(s)$), and another that learns how much better or worse a particular action is compared to the others ($A(s,a)$). These two parts combine to give the Q-value, but in a way that makes learning much more efficient.

One of the big takeaways was the concept of the Advantage. Advantage tells me how much better a specific action is compared to what my policy would normally do in that state. The corridor environment example in the paper made that easier to grasp. In that simple world, many actions don't really change the outcome, so being able to learn the general state value separately really speeds things up. Another nuance was the identifiability issue, which I had come across in Bayesian Machine Learning class. This was a good refresher on when this happens and how to address it, as the authors did by subtracting the mean advantage instead of using the max, helping smooth the gradients and makes the network more stable during training.

Reading Wang et al. alongside the earlier Double DQN paper helped me see how each improvement in reinforcement learning builds on the last. Double DQN fixed the overestimation problem in the target, while the dueling network fixed representational inefficiency in the Q-function itself. When you put the two together, as in later applications like the sepsis treatment study, the result is both more accurate and more data-efficient learning.

This module's lab on the Dueling Q-Network Lab, helped me to understand better how the value and advantage streams are implemented in code and how they interact during training. The level of coding requirements and OO design were a bit of a stretch for my skills. Use of AI to work through a design and code this in an efficient and elegant way was also very instructional.

Overall, this module helped me appreciate how much the architecture of a network matters, not just the algorithm. It’s a reminder that how we represent the problem can be just as important as how we solve it.

Module 8

Policy Gradient

In this module, I learned about policy gradient methods, which directly optimize the policy, rather than relying on a value function, which acts as an indirect proxy. The math at first is a bit more complicated in policy gradient. In particular, the log-derivative trick and the concept of taking gradients of expectations. My understanding eventually landed on understanding how the policy parameters control action probabilities, and the gradient points in the direction that increases the expected return.

A key concept I explored was the distinction between normalizing rewards and using a baseline in REINFORCE. Initially, I assumed they accomplished the same thing, but I learned they work in very different ways. Reward normalization is simply a statistical standardization of the returns to keep gradients numerically stable. A baseline, on the other hand, is usually a learned value function that estimates the expected return from each state. Subtracting this baseline from the actual return gives the advantage, which significantly reduces variance in the gradient estimate. While normalization and baselines operate differently, they both aim to reduce variance and stabilize training.

The Cartpole lab was helpful in fully understanding REINFORCE. I implemented Reward-to-Go to reduce variance. I also ran the code with and without normalizing the rewards and saw dramatic improvements in performance after normalization. Using normalized rewards, the agent scored all 500 rewards, much higher than without normalization (~250). These were greatly improved over the simple policy. I came to realize that the algorithm improves by not memorizing actions, but by adjusting the probability distribution over actions to favor those that lead to longer balance times.

These concepts are tricky to internalize and the hands-on lab is very helpful. I really appreciate the AI assistance in lowering the learning curve in using PyTorch. AI also serves an instructional purpose by doing code walkthroughs. This enabled me to focus my time on experimenting with the algorithm, and not the nuances of PyTorch or debugging. This improves my understanding of RL overall and make learning enjoyable.

Module 9

Policy Gradient Extensions

In Module 8 I learned to appreciate that policy-gradient extensions are part of a single, connected effort to reduce variance and stabilize learning. Starting from REINFORCE, I learned how reward-to-go and normalization provide basic variance reduction, while baselines and advantage estimation add a deeper level of stability by incorporating learned value functions. Actor–critic methods made this even clearer by blending policy and value learning into a hybrid approach that produces a much cleaner learning signal. TRPO and PPO then expanded this idea by showing how cautious, constrained updates can prevent instability in deep networks, with PPO offering a more practical form of TRPO's trust-region concept. Finally, DDPG illustrated how these themes extend into continuous-action spaces by using a Q-function to directly guide the policy. Across all of these methods, the recurring themes were variance reduction, stable update rules, and the integration of value information. These were my main takeaways that helped me understand how modern RL algorithms evolve from simple ideas into practical, high-performance approaches.

I would like to experiment on my own with code examples to try some of the additional algorithms, since we did not go beyond the basic REINFORCE in the lab. I plan to incorporate these into our team's project to deepen my understanding of policy gradient updates.

This is my last elective in the MSDS program, and next semester I will take Ethics and the Capstone before graduating in Spring of 2026. Coming back to graduate school after being away from for quite a while has been both exciting and a little intimidating. I took on this program as a personal challenge to immerse myself in the world of AI and push my technical abilities much further than I thought possible. This class has certainly been challenging, and the courses I have taken thus far helped me to not only get through it, but made leaning the material more rewarding. The concepts from several previous courses, such as Deep Learning and Bayesian ML, were instrumental in pulling this together for me.