**CPSC 448 Directed Studies in Reinforcement Learning**

**Schedule and Outline**

From Sept. 2018 – Dec. 2018 five UBC students including three graduate students and two undergraduate students met once a week for 1.5 hours in ICICS x718 from 9:30am – 11am. From Jan. 2019 – May 2019 four UBC students (two graduates and two undergraduates) met twice a week, i.e., between 4pm – 5pm on Tuesday and between 9:30am – 11am on Thursday. During these sessions we conducted activities related to our directed studies in reinforcement learning (RL). These activities include reading and discussing the first 13 chapters of Richard Sutton’s RL text, exploring a small number of textbook exercises and coding examples, attending relevant seminars as well as reading and discussing advanced research papers. Additional details of these activities are provided in what follows.

**Textbook –** *Reinforcement Learning an Introduction (2e) by Richard S. Sutton and Andrew G. Barto*

Part I: Tabular Solution Methods

After the introductory chapter the second chapter introduces several methods of balancing between exploration and exploitation. The epsilon-greedy methods choose a random action selection a small fraction of the time. The UCB methods choose deterministically but achieve exploration at each step by subtly favoring actions that have thus received fewer samples. Gradient bandit algorithms estimate action preferences and favor the more preferred actions in a graded, probabilistic manner using a soft-max distribution. The simple expedient method of initializing estimates optimistically causes even greedy methods to explore significantly. The author uses a parameter study graph to demonstrate the performance of these methods before concluding that all of the methods seem to perform best for the middle value of their own parameters and seem reasonably sensitive as the parameter varies. Overall, UCB seems to have the best performance using its optimal parameter.

In Chapter 3 the author defines RL and several terms that are frequently used. Reinforcement learning is about learning from interaction how to behave in order to achieve a goal. The RL agent and its environment interact over a sequence of discrete time steps. The actions are the choices made by the agent, the states are the basis for making the choices and the rewards are the basis for evaluating the choices. The policy is a stochastic rule by which the agent selects actions as a function of states. The agent’s objective is to maximize the amount of reward it receives over time. A policy’s value function assigns to each state, or state–action pair, the expected return from that state, or state–action pair, given that the agent uses the policy. The optimal value functions assign to each state, or state–action pair, the largest expected return achievable by any policy. Even if the agent has a complete and accurate environment model, the agent is typically unable to perform enough computation per time step to fully use it. In the context of RL, we are concerned with cases in which optimal solutions cannot be found but must be approximated.

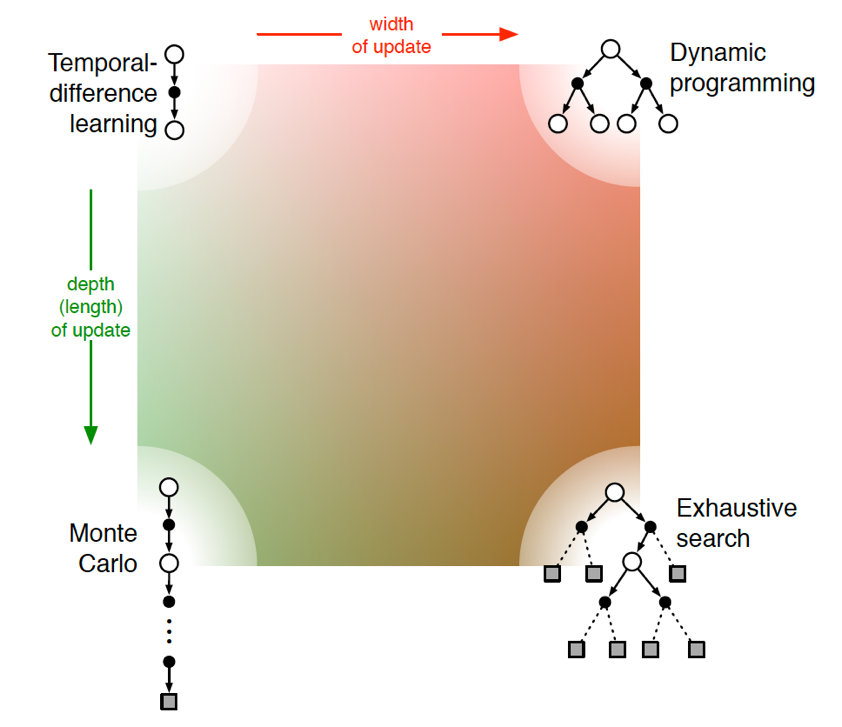
Chapter 4 introduces the idea of dynamic programming (DP) and how it is related to solving finite Markov decision processes (MDPs). Policy evaluation refers to the iterative computation of the value function for a given policy. Policy improvement refers to the computation of an improved policy given the value function for that policy. Classical DP performs an expected update operation on each state, updating the value of one state based on the value of all possible successor states and their probability of occurring.  The generalized policy iteration is the idea of two interacting processes revolving around an approximate policy and an approximate value function. One takes the policy as given and performs policy evaluation to update the value function, and another takes the value function as given and performs policy improvement. Asynchronous DP methods are in-place iterative methods that update states in an arbitrary order, perhaps stochastically determined and using out-of-date information. The idea of bootstrapping occurs in this method, referring to making estimations based on other estimates.

Monte Carlo methods differs from DP mainly in two ways, i.e., they operate on sample experience for direct learning without a model and they do not bootstrap. It is easy and efficient to focus Monte Carlo methods on a small subset of the states and this method may be less harmed by violations of the Markov property. The design of the Monte Carlo control method follows the schema of generalized policy iteration, involving the interacting process of policy evaluation and policy improvement. Averaging returns that start in the state provides a good approximation to the value of the state. An issue of this method is to maintain sufficient exploration. One way is to ignore the problem, assuming the randomly selected initial state-action pairs cover all possibilities. In on-policy methods, the agent commits to always exploring and tries to find the best policy that still explores. In off-policy methods, the agent also explores but learns a deterministic optimal policy that may be unrelated to the policy followed. Off-policy prediction refers to learning the value function of a target policy from data generated by a different behavior policy, based on the idea of importance sampling. Ordinary importance sampling uses a simple average of the weighted returns, whereas weighted importance sampling uses a weighted average.

In Chapter 6 the author introduces the temporal-difference (TD) learning method and how it can be applied to RL problems. We can classify TD control methods according to whether they deal with this complication by using an on-policy or an off-policy approach. Sarsa is an on-policy method, while expected Sarsa and Q-learning are off-policy methods. These methods are popular due to their simplicity. They can be applied online, with a minimal amount of computation, to experience generated from interaction with an environment. They can be expressed nearly completely by single equations that can be implemented with small computer programs. The special case of TD methods introduced in the present chapter should rightly be called one-step, tabular, model-free TD methods. Besides their application in RL, they are general methods for learning to make long-term predictions about dynamical systems. For example, TD methods may be relevant to predicting financial data, life spans, election outcomes, weather patterns, animal behavior, demands on power stations, or customer purchases.

In the seventh chapter a range of methods that lie in between one-step TD and Monte Carlo methods are introduced. These methods involve an intermediate amount of bootstrapping and they tend to perform better than either extreme. For an *n*-step method, we look ahead to the next *n* rewards, states, and actions.  The diagram below indicates the 4 step version. All *n*-step methods involve a delay of *n* time steps before updating, therefore requiring more memory and computation complexity to record the states, actions, and rewards. However, they have the great benefit of being computational clear. The authors have sought to take advantage of this by developing two approaches to off-policy learning in the *n*-step case. One, based on importance sampling is conceptually simple but can be of high variance. If the target and behavior policies are very different it probably needs some new algorithmic ideas before it can be efficient and practical. The other, based on tree-backup updates, is the natural extension of Q-learning to the multi-step case with stochastic target policies. It involves no importance sampling, but again if the target and behavior policies are substantially different, the bootstrapping may span only a few steps even if *n* is large.

The following figure is provided in Chapter 8 where the author provides a summary of Part I and some discussion on planning and learning with tabular methods. Figure 1 below shows the two most important dimensions distinguishing different RL methods, i.e., the horizontal dimension is whether they are sample updates (based on a sample trajectory) or expected updates (based on a distribution of possible trajectories) and the vertical dimension corresponds to the depth of updates, that is, to the degree of bootstrapping. A third dimension that is emphasized in this book is the binary distinction between on-policy and off-policy methods.



**Figure 1: Summarizing RL methods based on update width and depth**

Part II: Approximate Solution Methods

The second part of the textbook extends tabular methods to arbitrarily large state spaces by implement value function approximation. In Chapter 9, the author discusses supervised learning methods that use parameterized function approximation. The policy is parametrized by a weight vector *w*, and a mean squared value error is defined as the measure of approximation error. Focusing on policy evaluation or prediction, one of the most popular methods is *n*-step semi-gradient TD, which includes gradient Monte Carlo when *n* = ∞ and semi-gradient TD method when *n* = 1. Semi-gradient methods can obtain a good result in the case of linear approximation, where the approximated values are a linear combination of features with corresponding weights. Feature selection and representation thus become very important. Methods for feature representation include polynomial basis (not well-generalized), Fourier basis, tile coding (computationally efficient and flexible) and radial basis functions (useful for low dimensional tasks). Least squares TD is the most data-efficient linear TD prediction method, but with relatively high computation complexity; linear semi-gradient *n*-step TD is proven to be convergent to optimal error asymptotically under standard conditions. Non-linear methods include artificial neural networks trained by backpropagation and stochastic gradient descent, also known as deep reinforcement learning.

Chapter 10 focuses on the on-policy control problem with a parametric approximation of the action-value function. In the episodic case, the extension of the semi-gradient prediction methods to action values is straightforward, and we call it the episodic semi-gradient one-step Sarsa method. Then we can obtain an *n*-step version of episodic semi-gradient Sarsa by using an *n*-step return as the update target. The average reward setting is more commonly considered in classical dynamic programming than in the reinforcement learning context. With function approximation the discounted problem formulation is problematic because the returns from each state cannot be separately identified. Therefore, the average reward setting is required to replace the discounted reward setting. The average reward setting is similar to the discounted setting as they both apply to continuing problems, but now the agent cares just as much about delayed rewards as it does about immediate reward. Differential returns and corresponding differential value functions are introduced in this setting.  It is proven that the ordering of all policies in the average discounted return setting would be exactly the same as in the average-reward setting, meaning the discount rate has no effect towards the problem formulation.

Off-policy methods with approximation are covered in Chapter 11. The off-policy algorithm provides flexibility when dealing with the tradeoff between exploration and exploitation. Moreover, off-policy methods provide a separation between behavior and learning which frees the agent from the tyranny of the target policy. However, there are two main challenges: correcting the targets of learning for the behavior policy and dealing with the instability of semi-gradient TD methods that involve bootstrapping. Current attempts include true stochastic gradient descent in the Bellman residual, a gradient TD method that performs SGD in the projected Bellman error and emphatic-TD methods. As a relatively new area of research the selection of best methods remains an open question.

The eligibility trace methods introduced in Chapter 12 often offer faster learning at the expense of increased computational complexity. By adjusting the weight factor, we can place eligibility trace methods anywhere between Monte Carlo and one-step TD methods. It appears significantly better to use eligibility trace methods on tasks with many steps per episode.  Methods using eligibility trace require more computation than one-step methods, but they offer significantly faster learning when rewards are delayed by many steps. In the case of online applications, this method yields the behavior of expensive ideal methods while retaining the efficient computation of conventional TD methods. The possibility to conduct derivations that automatically convert from intuitive forward-view methods to more efficient backward-viewing algorithms is another appealing aspect of eligibility traces. At present, it is still unclear how to vary the trade-off between TD and Monte Carlo methods reliably and usefully.

Chapter 13 is the final chapter in Part II and it considers methods that learn a parameterized policy that enables actions to be taken without consulting action-value function, such as the policy-gradient method. This method updates the policy parameter on each step in the direction of increasing performance using an estimate of the gradient of performance with respect to the policy parameter. Advantages of this method include learning specific probabilities of taking an action, learning appropriate levels of exploration, approaching deterministic policies asymptotically, naturally handling continuous action spaces and simplifying the parametric representation of some problems. The policy gradient theorem provides an exact formula for how performance is affected by the policy parameter that does not involve derivatives of the state distribution. The REINFORCE method follows this theorem, in which adding a state-value function as a baseline can reduce the variance without introducing bias. The state-value function assigns credit to the policy’s action selections where the learned policy is a reference of the actor and the learned value function is a reference to the critic. The author mentions that this is a subject of excitement and ongoing research.

**Research Papers** – *Listed in the order in which we covered them*

1. Henaff M, Canziani A, LeCun Y. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. arXiv preprint arXiv:1901.02705. 2019 Jan 8.

The methods developed in this paper address scenarios similar to autonomous driving in which observational data is abundant but a single poor action is unacceptable and thus real agent-environment interactions are not a viable learning solution. A model-based RL approach is motivated for situations where building a comprehensive simulated environment is not practical. In these situations the model-based methods provide better sample complexity than model-free methods and therefore allow fewer environment interactions to learn an effective policy. Purely observational data is used to learn a model-based control method that uses a learned stochastic dynamics model based on variational autoencoders to generate trajectories that are used to train a policy network which minimizes both a policy cost and an uncertainty cost over the predicted trajectories. A series of dropout masks are applied to the dynamics model to represent uncertainty. This approach is applied to real-world traffic data and shown to effectively learn policies for dense traffic navigation. Ultimately, human performance is still superior.

1. Nagabandi A, Kahn G, Fearing RS, Levine S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) 2018 May 21 (pp. 7559-7566). IEEE.

This work explores hybrid combinations of model-based and model-free RL methods in the context of complex locomotion tasks. Although model-based RL methods generally achieve much better sample efficiency, due to model bias they often exhibit an inferior asymptotic performance relative to model-free RL methods. A hybrid approach is proposed wherein a learned dynamics model (based on a deep neural network) generates state estimates which are combined with a reward function (assumed to be known) and fed to a model predictive controller to determine the next action. A model-free learner based on a policy gradient algorithm (i.e., trust region policy optimization) is initialized using a neural network policy that is trained to match the actions generated by the MPC controller which are queried as ‘expert’ trajectories for ‘true’ action values. The hybrid model-based model-free (Mb-Mf) method is compared to a pure model free approach and shown to achieve 3-5x sample efficiency gains on benchmark locomotion tasks.

1. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. InICML 2014 Jun 21.

In this paper it is shown that the deterministic policy gradient exists with a simple model-free form and that it can outperform stochastic policy gradient methods in high-dimensional action spaces. In stochastic policy gradient algorithms a policy is represented by a parametric probability distribution that stochastically selects an action according to a parameter vector which is adjusted in the direction of increased cumulative reward. The deterministic policy gradient is demonstrated to be the limiting case as the stochastic policy variance tends to zero. An off-policy learning algorithm ensures exploration by choosing actions with a stochastic behavior policy in order to learn the deterministic target policy. To ensure the action-value function approximator does not bias the policy gradient a notion of compatible function approximation is introduced that includes a condition for linearity with respect to features of the policy and a condition that the parameters minimize the mean squared error between the action-value function gradient and the true gradient. The compatible off-policy deterministic actor-critic (COPDAC) algorithm is compared favorably to stochastic on-policy and off-policy actor-critic methods on various standard RL benchmarks for continuous action spaces.

1. Negenborn RR, De Schutter B, Wiering MA, Hellendoorn H. Learning-based model predictive control for Markov decision processes. IFAC Proceedings Volumes. 2005 Jan 1;38(1):354-9.

This paper was disappointing from a reinforcement learning perspective. The underlying concepts used for the problem formulation and the solution method are quite interesting but they are only introduced at a surface level. Moreover the approach that is developed is not implemented on any experimental simulations or case studies so there are effectively no results and thus no validation of the proposed methodology. The paper begins by providing a basic introduction to model predictive control (MPC) and Markov decision processes (MDP). A straightforward implementation of MPC and MDPs is first provided before value functions are introduced as a means of considering the most up to date system model and accounting for slowly changing system and performance desires. Since the value function is expensive to compute at each iteration the authors then introduce RL, specifically TD learning, to incrementally determine the value function online. Initially since the uncertainty in the value function is high conventional MPC is used but as the uncertainty of the value function decreases it can be used to reduce the intensive computation required by MPC. Unfortunately the paper concludes with some general commentary after developing the MPC-TD technique without actually implementing the technique.

1. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971. 2015 Sep 9.

This paper furthers the recent developments made towards deep RL using the “Deep Q Network” (DQN) algorithm and the deterministic policy gradient (DPG) algorithm by combining elements of both to develop a model-free approach known as the Deep DPG (DDPG) for continuous action spaces. Unless the action space is discretized deep RL techniques like DQN can only handle low-dimensional action spaces because it relies on maximizing the action-value function which requires iterative optimization at each step for continuous domains. Discretizing the action space can result in loss of critical information about the action domain and it can complicate scalability due to the curse of dimensionality. To address the challenges of applying DQN to continuous action spaces the DPG approach is adopted and modified for neural network function approximators. Unfortunately there are no convergence guarantees with DDPG (due to the non-linear function approximation) but the DDPG method is still demonstrated to use fewer steps of experience to find solutions in the Atari domain than the DQN method.

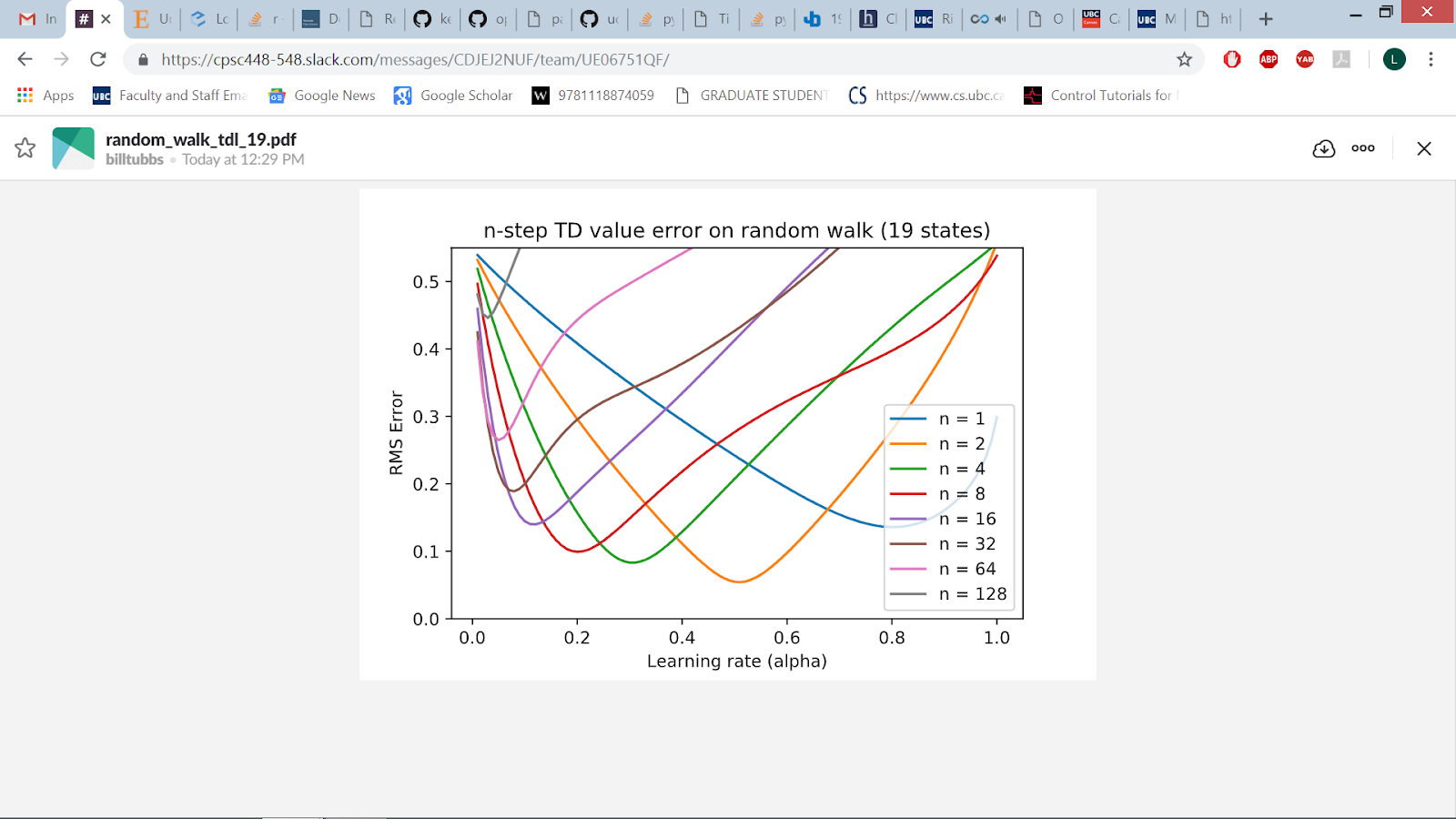
1. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013 Dec 19.

As we move into the summer we are planning on continuing our RL reading group and this is the paper we are covering at the moment. Reviewing this paper will hopefully provide some level of completeness to our coverage of the DPG and DDPG articles summarized above. In hindsight it probably would have been wiser to cover this article prior to the DDPG article. One important aspect of this paper that we will explore is the use of non-linear value function approximators which were originally thought to lack robustness and be unstable. This paper develops a stable and robust use of non-linear value function approximators by using a replay buffer for off-policy network training and implementing a separate target Q network to provide consistent targets during TD backups for training the critic network. Since we have not yet completed our reading and discussion of this article it is best to regard this summary as incomplete.

**Additional Activities**

Coding and textbook exercises

As a group we informally discussed a variety of theoretical textbook exercises and went through a couple of brief coding exercises to modify some parameters and observe the results. The base code from the RL book is available online so this was simply modified for our purposes. One specific exercise involved experimenting with the grid-world example from Chapter 4 by modifying the discount rate and the policy. Another coding exercise involved adding double Q-learning to the cliff-walking example and experimenting with different penalties to find different optimal policies between Sarsa and expected Sarsa. Finally, we spent quite a bit of time discussing exercise 7.3 including modifying code to determine how the value of *n* changes with a smaller random walk. From Figure 2 below it can be shown that the optimal value of *n* changes from 4 to 2 when the random walk is changed from 19 to 5 states. We continued to discuss textbook exercises in Part II of the text but otherwise most of our focus shifted from coding exercises to research papers in the second semester.



**Figure 2: Performance of *n*-step TD methods with a 5 state random walk**