Reading Guide for CMPUT 609: Reinforcement Learning II

by Rich Sutton

The primary goal of this course is to obtain a full understanding of Reinforcement Learning and its potential power as a means to Artificial Intelligence. To this end, we will study the textbook, Part 2 and Chapter 17, and additional material from the literature and from current work. These are the main activities and you must be an active participant. The technical material will not be presented in lectures. But neither are you on your own with the readings. I will act as your guide and coach, providing perspective, answering questions, and encouraging you. I will also provide feedback, challenge you to fully understand it, and test your understanding so that you will know if you are really getting the important points.

For Class 1:

For each reading I will provide some questions for you to think about while you are doing the reading. For example, the first reading is a bit different. It involves reviewing all of the first part of the textbook, which you should have covered in previous courses. While reviewing this, I want you to ask yourself, and prepare for answering a questions about the strengths and weaknesses of the various classes of methods in Part 1. Those classes of methods are, roughly, Bandits, Dynamic Programming, Monte Carlo methods, Temporal-Difference methods, n-step methods, and Dyna. Why would you use each method? For each method, what is its strength? When would you use it? What is its weakness?

As you read, some questions may occur to you; write these down as you go along. The act of writing them down is key. It will help you formulate them more precisely. Try to write them in a clear way that makes as few implicit assumptions as possible. Then prepare to ask them in class.

I have listed two additional readings to the first class. We will be talking a lot about the "common model of the intelligent agent" in class this year.

For Class 2:

To get started, we need to get the big picture clear, and for this I am asking you to read a very short provocative essay, "The Bitter Lesson", a provocative video, and the first parts of the book directly dealing with approximation (thru p. 210). The essay can be found in the Readings subfolder of the course folder (a pdf of the textbook is there too). The video is linked to from the schedule pdf file.

I also recommend this introductory video: https://youtu.be/vY-voHb22io?t=220

What methods scale with computation in the sense of the video? How can we define the approximation goal of approximate value functions? Suppose we tried to estimate transition probabilities using a non-Markov state; what problems might occur? Does supervised learning scale with computation? Does model-free reinforcement learning scale with computation? Can we ever know the real world exactly? Can we hope to obtain the best possible approximation given our computational limitations?

For Class 3:

The reading for class3 is the textbook thru page 222, including linear approximation with various feature representations. Pay attention to the key ideas of gradient descent and stochastic gradient descent (SGD) — what is the difference? There is also this new idea of semi-gradient methods. We also introduce overall error measure, the mean squared value error of a weight vector, denoted $\overline{\text{VE}}(\mathbf{w})$. Note that it depends on a state distribution.

Prepare your questions to be asked and answered in class.

A second reading for this class is the python code in SLplay.py. And that will direct you to the perceptron tutorial at https://www.simplilearn.com/what-is-perceptron-tutorial. You will want to read through the tutorial and try out the code to make sure you can run it by the end of this class day. We will release a new written assignment that uses this code on this day.

For Class 4:

The reading assignment for class4 is the rest of Chapter 9 and the regression/classification tutorial at https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7.

Step sizes. Emphasis and the importance of state distributions. Artificial Neural Networks. Least-squares methods.

There are a great variety of function approximators. Artificial Neural Networks are key to the most impressive applications. And yet the book ends up focusing on linear function approximators. I think this is clearly the right choice, but I would not be surprised if it seems limiting to you. We need to hash that out. So think about it. Why would one prefer linear over non-linear approximators? Linear approximators are limited, but how could we easily move beyond those limitations?

In general, what are some of the important properties for a function approximator to have in order for it to be suitable for use in reinforcement learning?

For Class 5

The reading for class5 is just the external documentation for the standard tile coding software. We will use this in class and in the written assignment w2, which you should be working hard on by this point.

Tile coding. Is it just a form of state aggregation? Or is it more powerful than state aggregation in important ways? In what sense is particularly computationally frugal?

For Class 6

With this class we will be getting back to reading the book and answering questions in class. The reading is all of Chapter 10. On the one hand the extension is fairly straightforward from prediction in Chapter 9, and from control methods in Part 1, but on the other there is the whole new average-reward setting and giving up on our old friend discounting.

Here are some questions to ask yourself while you are reading:

What happens when a constant is added to the value function of all states — does the value function still satisfy the differential Bellman equations? What does this mean? Did this happen with the discounted formulation?

How many Bellman equations and how many unknown do we have in the average-reward setting? Do we need another equation/constraint? What should it be?

Is the average reward formulation the same as the discounted formulation with gamma to 1?

Have we lost the policy improvement theorem with function approximation?

What is the difference between a Markov reward process and Markov chain?

What is a limiting state distribution?

Do average rewards make sense for episodic tasks?

For Classes 7, 8 and 9 (Chapter 11, Off-policy learning)

Post your questions here: https://app.sli.do/event/5qpglfu4.

We start our exploration of one of the trickiest and most important issues in reinforcement learning: off-policy training. First the deadly triad of function approximation, bootstrapping, and off-policy learning.

Here are some questions to ask yourself while you are reading:

1. Is off-policy a problem or a solution?
2. Do we have to give up on one of the three in the deadly triad?
3. Is the deadly triad a problem in theory or practice?
4. Why is off-policy learning so important?

This chapter treats some subtle technical points, so we will spend some time with it. Pay attention to view of value functions as vectors with $|\mathcal{S}|$ components, whereas the weight vector has only d components. Try to understand the technical aspects of Figure 11.3 and the two A-split examples, 11.2 and 11.3.

The contents of Section 11.6 is deep, but not really that hard. If you can understand this one section, then you will understand reinforcement-learning function-approximation theory better than most scientists working in that field. Figure 11.4 summarizes the whole view.

Sections 11.7 and 11.8 present the two best convergent off-policy algorithms currently known. The two adopt very different strategies. Can you summarize the general idea of the two strategies in your own words?

For Class 10 (Chapter 7 - n-step Bootstrapping)

This chapter shows that there is nothing special about the single time step in TD methods; you can do them with 2-step, 3-step, or n-step bootstrapping.

We also revisit the random walk prediction task from Chapter 6, this time in 19-state form. We use it to show a detailed parameter study, showing how prediction performance varies with α and the number of steps of bootstrapping.

Be sure to engage with the idea of a control variate. We will come across this idea again, and its interaction with importance sampling, in the chapter on eligibility traces (Chapter 13). It will be much easier to understand it first in the simpler setting here.

Sections 7.5 and 7.6 present more cutting-edge n-step algorithms.

For Classes 11-13 (Chapter 12 - Eligbility Traces)

This chapter is a little intense. I want you to go with it and get into it, so I have arranged the whole of w5 as detailed questions from this chapter. By the time you are done you will really understand the core online algorithms of reinforcement learning that I think will one day lie at the heart of artificial intelligence. It will really help if you read carefully and try to get into the spirit of these algorithms. Look for the patterns in the calculations and algorithms.

For Class 14 (Chapter 13 - Policy Gradient)

This chapter introduces a rather different style of solution methods. They are old, but also currently popular. This chapter does just the foundational stuff. Assignment w6 will help drive you to become acquainted with the ideas in a more substantive way. These are the methods most commonly used today in episodic control applications such as in OpenAI's Gym.

For Class 15 (Experience and Intelligence)

The reading for this class is a video, and there is a chance I won't be able to make it to class this day. Come to class prepared to discuss the video.

1. Do you agree that AI has moved grudgingly but inexorably toward working in terms of sensorimotor data?
2. Can you think of some additional examples from your own experience?
3. How about *counter*-examples or counter trends?
4. Do you thing it's really possible to define knowledge and intelligence entirely in terms of the data of experience?

For Class 16 (GVFs and subproblems)

This is a small reading—only two pages from the book—so you have a little time to catch up if you have not finished fully absorbing the policy gradient chapter. The two pages ask you to think about the significance of the fully generalized form of value functions. Here is where we start to ask you to imagine that GVFs might be a general representation language for knowledge about the world. We know they are a form of prediction that is computationally convenient for being learned. Now we think about how expressive they are. In particular, recognize that they are a language for representing questions about the world that are grounded in experiential data. They express a wide class of problems, and we have learned in this course a powerful set of solution methods for these problems. With these two we can have an agent that poses and solves its own subproblems. We will see over the next few weeks how this may lead to a powerful RL agent.

For Class 17 (Model-based RL)

Our ultimate agent will learn a model and plan policies and values with it (in addition to learning policies and values directly from experience). So let us re-read Chapter 8 from Part 1 of the book, which presents a super-simple tabular architecture (Dyna) which does all of the basic things (learn, plan, model learn). This will be our starting place. The first extension will be to add function approximation. This is not a simple thing in general, but the first step (Dyna with linear models and value functions for the prediction case) works out well. This is what is developed in the reading LinearDyna.pdf from the literature (in the course dropbox).

For Class 18 (Options)

As Section 17.2 explains, options allow our agents to work with transitions that abstract over time. This is a big deal, yet it builds almost trivially on all that you have already learned. Options are a generalization of actions and can be used in place of actions in almost all our learning and planning algorithms. Or so we strive to make true.

The reading for options is the original paper from 1999. This paper is well known and highly cited (and a recent winner of the AI Journal's classic paper award), but its full significance is still overlooked (IMHO). There are three common misconceptions; be alert and on guard against them, as they are seductive. The first is that options must be executed. Although this is an easy way to think about them, and one way to use them, it is not the only way. It is possible that options are never executed and instead are used only in planning. One thinks about executing them but never actually does in the sense of committing to following them to their termination.

The second misconception is to think of them as data structures. Again, they might be one-to-one with some data structures, but then they might not. An option is a mathematical ideal, like a number or a function. A data structure implements a function, but if it is changed, the function is not changed; rather the data structure has been changed to implement a new function.

The third misconception is that the agent can work only with a finite, discrete set of options. Just as actions may be finite in number, but also may be continuous and thus infinite in number, so it is for options. One could have a family of options individuated by a parameter, such as for walking with a parameter for speed. The option for walking with speed 2 m/sec is formally a different option than walking at 2.01 m/sec, but everything about it (its policy, its termination, its predictions and transition models) may be stored in function approximation approximators that treat the two options as very similar.

For Class 19 (state construction)

Another, important, big extension that is intended to greatly increases the range of applicability of our RL prediction and control algorithms without changing their core ideas and mechanisms. This is a major current frontier. The reading start to get you familiar with its basics. We learn about some of the ways partial observability is commonly thought about in the literature, yet we try not to be overly influenced by these conventional ways of thinking.

We ask: Is partial observability the same, essentially, as function approximation? Is state construction the same, essentially, as representation learning? Does the notion of the recurrent state-update function u replace the notion of feature vector function $\mathbf{x} : \mathcal{S} \rightarrow \mathfrak{R}^d$ used throughout Part 2 of the book? In what senses does the agent state successfully take the place of the Markov environmental state?

For Class 20 (New ideas in Deep Learning)

This reading is a new paper.

For Class 21 (You are the agent, discovering the POMDP)

Here we will do some work in class, in groups, trying to get a feel for the problem facing the agent as he tries to predict and model a partially observable environment.

For Class 22-23 (STOMP and Oak)

Now we have all the pieces and we can consider the whole architecture, including the discovery question: Where do the features and options come from? If we can answer this in a satisfactory way, then maybe we have a full and complete agent architecture.

The reading is the rest of MBRL2.pdf, STOMP.pdf, MarrLevels.pdf, and the Oak slides Oak.pdf. I am not sure what is the best order to read these things. MBRL2 attempts to specify everything down to the algorithms, but that is too ambitious to do in a wholly satisfactory manner. The STOMP paper is current research trying to bring the vision of ambitious model-based RL alive. This short paper covers in quick form all of the ideas except the construction of the agent state (Class 18).

Oak is an attempt to take a step upwards, to present an outline of everything but not detail it all the way down. The Oak idea is to present a "computational theory," in the sense of Marr, of intelligence and cognitive development. This means the *purpose*, the *what*s and the *why*s, of intelligence, without committing to particular algorithms and representations. Oak does not have a regular writeup beyond the STOMP paper, just a talk. Still, it is an outline, and as such maybe you should read it first or at the same time as the MBRL2 text. Maybe try the MBRL text until you feel the need for an outline, then consult the Oak slides to see how the pieces fit together. I think I will start with the Oak slides (some of which you have already seen) on the first class.

For Class 24 (Step-size adaptation - Generalization sculpting - Representation learning)

The readings for this class are several. First, read the first full paragraph on page 473 of the book. Then, if you want more background or to see how things fit into the wider literature of ANNs, you might read MetagradientHistory.pdf. Or, if you want to dive right in, read IDBD.pdf. Finally, get a sense of how IDBD might be used in Mahmood.pdf, Chapters 11 & 12. Keep in mind how the step sizes might be used in an RL architecture to rank features as described in MBRL2.pdf and in the Oak architecture.

Representation learning is a big old topic. It's very important, and maybe the field has developed some perspective on it, but I would say that not all that much has been learned even after all these years. So, tread carefully. I would not be surprised if this whole area becomes increasingly of interest in the next decade.

For Class 25 (Last regular class)

The reading is the last three sections of the book. Please use them to reflect on all that you have learned about RL and AI in the course.