## Introduction to Markov Decision Processes, Part 1

This week we start working on the technical content of the course.

Following are [4 ways](https://docs.google.com/spreadsheets/d/1ZdLaygbVu3LtlTUegGsOAE4TmQEOhJH7qU8hSNHbvxo/edit?usp=sharing) to present *the same material*. Watch/read *all of them*. One version might help you understand another version. During the semester, we will be jumping back and forth among these sources. It’s important to get used to each style. The actual material this week should not be too challenging. You will have time to watch/read *all versions*.

1. Udacity [Reinforcement Learning](https://classroom.udacity.com/courses/ud600) course. Watch videos for Lesson 1 and Lesson 2 up to 2.15. Be sure to do the quizzes. **(Let’s do the quiz:** [**Lesson 2.12**](https://classroom.udacity.com/courses/ud600/lessons/4100878601/concepts/6512308700923)**.)**
2. Silver, [Markov Decision Processes](https://www.youtube.com/watch?v=lfHX2hHRMVQ&index=2&list=PLV_1KI9mrSpGFoaxoL9BCZeen_s987Yxb) up to minute 13. This will be a more mathematical version of the Udacity course, but it’s important to get used to the notation.
3. CS 188 (Berkeley). MDT’s [Lecture 8](https://youtu.be/Oxqwwnm_x0s?t=2m50s) until 14:47 ([slides](http://ai.berkeley.edu/slides/Lecture%208%20--%20MDPs%20I/SP14%20CS188%20Lecture%208%20--%20MDPs%20I.pptx)). We are going to use the software from this course; so it’s good to get used to their style.
4. [Sutton and Bartow](http://incompleteideas.net/book/the-book-2nd.html). (This is the new edition of the classic book on the subject.) Read Chapter 1 and Chapter 3 through section 3.2. (You can skip Section 1.7 and Chapter 2). Be sure you understand example 3.3.

I found that when reading this book I often want to look back to refresh my memory about something. To make that easy, create additional copies of the pdf on your screen:

Window -> New Window

Consider these points from the reading in [Sutton and Bartow](http://incompleteideas.net/book/the-book-2nd.html).

* + - Section 1.5 provides an overview of Reinforcement Learning, which the other sources do not present.
    - **Do the exercises at the end of Section 1.5 (pp 12 and 13) and submit your results to CSNS.**
    - Section 1.5 includes this update equation at the bottom of page 9.

V(St) ← V(St) + α[V(St+1) − V(St)]

Show that the right hand side is a weighted average of V(St) and V(St+1) and determine what the weights are. That is, write a function of three arguments

weighted\_average(item\_1, weight, item\_2)

that computes the weighted average of item\_1 and item\_2, giving weight to item\_1 and (1 - weight) to item\_2.

**Show that your function computes the same value as the equation above. That is, V(St) would be the same in either case.**

V(St) ← weighted\_average(V(St), weight, V(St+1))

**Submit your result to CSNS.**

* + - **What is the Markov property? Submit your answer to CSNS.**
    - **Do exercises 3.1 (p 51) and 3.4 (p 53). Submit your results to CSNS.**
    - Chapter 3 through 3.2 is only about 8 pages long. But it includes quite a few mathematical expressions and equations. They are not difficult, but you have to learn how to read them. For each one, take it slowly and decode what it means.

For example, equation 3.5 (p 49) includes a double summation. A good way to understand a summation is as a loop. Double summations are nested loops: the left-most summation is the outer loop; the summations to the right are inner loops. So, equation 3.5 defines the function *r(s, a), t*he expected reward of taking action *a* when in state *s* in terms of a nested loop

First, understand the notation. Recall that *p(s’, r | s, a)* means the probability of going to state *s’* and receiving reward *r* when in state *s* and performing action *a*. If this were a computer program you would have a table telling you the possible outcomes of performing action *a* when in state *s*. The way the equation is written we assume that we don’t know what those possible outcomes are. So we look at all possible combinations of states *s’* and rewards *r*. For most of them the probability is 0. The only ones that matter are the outcomes that might actually occur when performing action *a* when in state *s*.

Take your time and think about it the way you would try to understand a program you have never seen before. It takes time to decode it and understand it. You’ll find it’s not too hard. Ask questions on the Forum if you get stuck.

David Silver’s course outline

