Numbers

The assignment is worth 15% of your final grade.

Why?

In some sense, we have spent the semester thinking about machine learning techniques for various forms of function approximation. It's now time to think about using what we've learned in order to allow an agent of some kind to act in the world more directly. This assignment asks you to consider the application of some of the techniques we've learned from reinforcement learning to make decisions.

The same ground rules apply for programming languages as with the previous assignments.

***Read everything below carefully!***

The Problems Given to You

You are being asked to explore Markov Decision Processes (MDPs):

1. Come up with two interesting MDPs. Explain why they are interesting. They don't need to be overly complicated or directly grounded in a real situation, but it will be worthwhile if your MDPs are inspired by some process you are interested in or are familiar with. It's ok to keep it somewhat simple. For the purposes of this assignment, though, make sure one has a "small" number of states, and the other has a "large" number of states.
2. Solve each MDP using value iteration as well as policy iteration. How many iterations does it take to converge? Which one converges faster? Why? Do they converge to the same answer? How did the number of states affectthings, if at all?
3. Now pick your favorite reinforcement learning algorithm and use it to solve the two MDPs. How does it perform, especially in comparison to the cases above where you knew the model, rewards, and so on? What exploration strategies did you choose? Did some work better than others?

**Coding Resources**  
[Brown-UMBC Reinforcement Learning and Planning (BURLAP) java code library  (Links to an external site.)Links to an external site.](http://burlap.cs.brown.edu/)

What to Turn In

You must submit:

* a file named *README.txt* that contains instructions for running your code
* your code (link only in the README.txt)
* a file named yourgtaccount-*analysis.pdf* that contains your writeup.

The file yourgtaccount-*analysis*.pdf should contain:

* A description of your MDPs and why they are interesting.
* A discussion of your experiments.

**Note: Analysis writeup is limited to 10 pages total.**

Grading Criteria

As always you are being graded on your analysis more than anything else.

**grid 20 or 30**

**maze??**

**effect of size on time etc.**

**# packages - python: openAI gym; pipy mdp toolbox; bearlab - also easy**

**# look up the libraries - they all have examples to choose from! "openAI gym reinforcement learning"**

**- policy iteration, value iteration, Q-learning**

**- for these focus on whether they reach optimal answer, if so study convergence properties - when they do and how long it takes to converge**

**- compare across methods - changing size of problem and how effect on performance (can be specific to problem, or choose 2 different problems then change size of both of them and explore)**

**- policy they reach, is it the same, how long it takes them**

**- expect Q-learning - need to talk about exploration/exploitation, explore different values of the parameters etc. and see how they change**

**- frozen lake – what is it? Mdp with grid and some uncertainty in transition prob, baits and goals.**

# **Markov Decision Processes and Reinforcement Learning**

**Tichakunda Mangono**

### ***CS7641 Machine Learning Paper 4, April 15th, 2019***

**Introduction**

Machine Learning is largely made up of Supervised, Unsupervised, and Reinforcement Learning (RL). While the first two involve function approximation and function/data description respectively, Reinforcement Learning is about learning to make decisions and take actions to optimize rewards. The decision making processes of RL are modeled well using Markov Decision Processes (MDP) a mathematical model of discrete time stochastic control processes where outcomes are partly random and partly under the control of a decision maker[[1]](#footnote-1). Therefore, RL allows us to model a world where an agent makes these decisions, takes actions and we can observe the rewards and develop the optimal policy for the agent to maximize rewards and achieve desired outcomes. Some of the most popular applications for RL include robotics, gaming (checkers, chess, backgammon, etc.), and even self-driving cars and unmanned flying vehicles. Reinforcement Learning allows us to understand and solve dynamic real-world problems at a lower cost.

**Objective**

This paper will introduce two important MDP problems and use both policy and value iteration to study the nature, quality and performance during convergence to a solution. The effect of problem size, number of states and complexity will also be studied for each problem. Finally, both problems will be solved using Q-Learning, a choice algorithm for reinforcement learning to compare the outcomes. By using policy iteration, value-iteration, and Q-learning, this paper will successfully examine the performance of two planning approaches and one learning approach.

**Use this for problem definition – first few states, then more states plus obstacles.** <https://www.youtube.com/watch?v=nSxaG_Kjw_w>

Blog and code here: <https://amunategui.github.io/reinforcement-learning/index.html>

MDP worked example fire mgmt. here: <http://sawcordwell.github.io/mdp/conservation/2015/01/10/possingham1997-1/>

**Using value iter and policy iter – in MDP:** <https://www.youtube.com/watch?v=dOa_CA2JdLE>

**Use first vid also for the Q-Learning as it comes done out of the box**

**Markov Decision Processes (MDP)**

**Policy Iteration (PI)**

**Value Iteration (VI)**

**Q-Learning (QL)**

**Reinforcement Learning (RL)**

**Two Interesting MDP Problems**

**Problem 1: Grid World**

Description of problem set up and why it is interesting/important

**Problem 2: Frozen Lake**

Description of problem set up and why it is interesting/important

**MDP Solutions**

Use PI, VI, and QL and analyze convergence nature, time etc.

**Reinforcement Learning Solution**

Q-Learning and compare

1. <https://en.wikipedia.org/wiki/Markov_decision_process> [↑](#footnote-ref-1)