Imperial College London

October 18, 2019

Department of Computing

CO424/BBB – Reinforcement Learning/Bits, Brain & Behaviour – Dr Aldo Faisal

Lab Assignment 3

The lab assignments are voluntary exercises for students to engage in the contents of the course and deepen them through hand's on programming in a supported manner by the GTAs. They do not form part of the course assessment (they are not coursework and no programming is required in the exam).

Dynamic Programming

Consider the 5x5 GridWorld MDP demonstrated with 22 states $\mathcal{S} = \{s_1, \ldots, s_{22}\}$, of which five are absorbing - one with a positive and four with negative rewards. Available actions are $\mathcal{A} = \{up, down, left, right\}$, which may in general be noisy. Discounting is geometric with discount factor γ. Note: this assignment covers more than 1 week of labs.

1. Unpack the folder LabAssignmentB into a folder and run the function in GridWorldClass.m

2. For a uniform policy (with equal probability of choosing any action), use the implementation of Policy Evaluation in *PolicyEvaluation.m* to evaluate the state-values for that policy with $\gamma = 0.9$. Investigate how the state-values change for different values of γ. Think of a way to measure convergence, and investigate how the rate of convergence of policy evaluation depends on γ. Can you explain why?

3. Choosing actions that are greedy with respect to a value function is guaranteed to give a better policy (the *Policy Improvement Theorem*). How would you use this fact to find the optimal policy? Implement an iterative algorithm that finds the optimal policy by alternating policy evaluation and greedy policy improvement.

4. The described method for policy improvement is part of a family of algorithms collectively known as *generalised policy iteration*. PolicyEvaluation.m iterates until the state-values change less than a threshold θ between iterations. It can be shown that this method converges to the optimal policy if alternating any number of evaluation and improvement steps.

   Implement policy iteration that improves the policy between every step of policy evaluation. How does the time taken to converge to the optimal policy compare to the previous method? How does this depend on γ? Why?

5. Investigate how the optimal policy depends on features of the MDP. For example, use different values of γ, change the 'noisiness' of actions or the magnitude of the positive and negative rewards. Can you explain why the optimal policy changes?

6. By construction, the optimal policy found by solving the Bellman equation (e.g. using dynamic programming) corresponds to the policy that actually maximises the average (discounted) reward experienced by an agent.

   Starting in a random state and following the optimal policy, simulate many actual runs of the MDP, recording the individual *traces* (i.e. recording state, action, reward sequences). Do the average experienced discounted cumulative rewards experienced after visiting a state agree with the value functions calculated above?

7. The online method of estimating state-values through experience is known as a *Monte-Carlo* method. How does performance compare to that of dynamic programming methods? Can you think of some reasons why you might want to use Monte-Carlo over dynamic programming methods? How might you combine desirable features of Monte-Carlo and dynamic programming?