Exploration of Different Gradient Descent Methods for Bayesian Inverse Reinforcement Learning

Daniel Brown and Dohyun Kim

November 29, 2016

1 Introduction

Markov Decision Processes (MDPs) are a commonly used model for sequential decision making tasks. Many different methods including as dynamic programming and temporal difference methods have been proposed to allow a learning agent to solve for the optimal policy of an MDP. However, in some tasks it can be hard to specify a reward function and may be easier to provide demonstrations. This leads to the problem of Inverse Reinforcement Learning (IRL) [?]. Given an MDP\R (an MDP without a specified reward function), and a set of demonstrations consisting of state-action pairs, we wish to recover the reward function that makes the demonstrations optimal.

Many approaches have been proposed to solve this problem. We choose to focus on the Bayesian IRL setting [?]. Gradient methods have been proposed to solve this problem [?, ?] but contain no details about the specifics of how to perform gradient descent. We seek to find the MAP estimate of the true reward by maximizing the posterior of the reward R given a set of demonstrations D where

$$P(R|D) \propto P(D|R)P(R) \tag{1}$$

The likelihood $P(D|R)$ is defined in a form of softmax function [?] as

$$P(D|R) = \frac{e^{\alpha \sum_i Q^*(s_i,a_i;R)}}{\sum_{b\in A} e^{\alpha \sum_i Q^*(s_i,b;R)}} \tag{2}$$

where $Q^*(s,a;R)$ is the optimal Q-value function for reward R and α is a confidence parameter defining how reliable the demonstrations are. The prior $P(R)$ can be any function.

We want to use gradient ascent to update to find

$$R_{MAP} = \arg\max_R P(R|D) = \arg\max_R [\log(P(D|R) + \log P(R)] \tag{3}$$

reward function to maximize the posterior, thus our update takes the form

$$R_{new} \leftarrow R + \eta_t \nabla_R [\log(P(D|R) + \log P(R)] \tag{4}$$

We propose to explore some of the different flavors of gradient descent we have discussed in class as they apply to the problem of solving the IRL problem.

2 Gradient Computation

We first considered just the maximum likelihood estimate of the reward (corresponding to a uniform prior). To perform gradient descent we need to find the gradient of the likelihood function. We have

$$\nabla_R \log P(D|R) = \nabla_R \log \prod_{(s,a)\in D} P(a|s,R) \tag{5}$$

$$= \sum_{(s,a)\in D} \nabla_R \log \frac{e^{\alpha Q^*_R(s,a)}}{\sum_{b\in A} e^{\alpha Q^*_R(s,b)}} \tag{6}$$

$$= \sum_{(s,a)\in D} \nabla_R \alpha Q^*_R(s,a) - \log \sum_{b\in A} e^{\alpha Q^*_R(s,b)} \tag{7}$$

$$= \sum_{(s,a)\in D} \alpha \nabla_R Q^*_R(s,a) - \frac{\alpha}{Z_a}\left(\sum_{b\in A} e^{\alpha Q^*_R(s,b)} \nabla_R Q^*_R(s,b)\right) \tag{8}$$

$$\tag{9}$$

where

$$Z_a = \sum_{b\in A} e^{\alpha Q^*(s,b|R)} \tag{10}$$

To compute $\nabla_R Q^*_R(s,u)$ for some action u, we note that

$$V^\pi = R + \gamma T^\pi V^\pi \tag{11}$$

so

$$Q^*_u = R + \gamma T^a V^{\pi^*} = R + \gamma T^a (I - \gamma T^{\pi^*})^{-1} R \tag{12}$$

and

$$\frac{\partial Q(s,u)}{\partial R_i} = \delta_i(s) + \gamma W(s,i) \tag{13}$$

where $W = T^a (I - \gamma T^{\pi^*})^{-1}$

3 Approach

We propose to investigate a simple, yet scalable navigation domain where a subset of states have negative reward (obstacles) most states have zero reward and the goal state has positive reward. Given a few demonstration trajectories, we wish to recover the reward using gradient descent.

We propose to investigate the following gradient descent methods: standard gradient descent (GD) with full step size, GD with BTLS, and accelerated GD.

We also hope to also investigate whether using a sparsity-inducing l1 regularization term can help find a sparse reward function that matches the demonstrator's reward. We will compare the Frank-Wolfe, and subgradient descent methods to maximize the likelihood minus the regularization.

4 To Do List

First we need a way to generate navigation tasks. We will develop a simple grid world simulator where there are a finite number of cells on a map that a robot can drive on. Certain cells will contain dangerous terrain that should be avoided and one cell will be the goal location. This domain can easily be scaled up or down as needed by increasing the number of states.

We also need to compute a closed form expression for the gradient of the BIRL likelihood function for performing gradient descent. To compute the gradient we need a way to solve for the optimal policy given a hypothesis reward function. We will implement the policy iteration algorithm to accomplish this task.

We will then proceed to experiment with different methods for choosing the step size and ways of adding sparsity as mentioned above.

We will show plots of the error between the learned reward and the true reward as well as the error between the policy performance between the optimal policy and the policy corresponding to the learned reward. We will compare the number of iterations required to achieve low error as well as the computation time needed. Due to the requirement to solve an MDP at every step, it will be important to see which methods can reduce the error in the least amount of gradient steps, thus we hypothesize that BTLS and accelerated methods will improve upon standard gradient descent techniques. Additionally, we hypothesize that adding a regularization term will allow gradient descent to find a better reward function that matches our sparse domain.