Reward Elicitation

1 Introduction

Reward quantification is hard, cognitively complex in practice, and time consuming. It is problem dependent, and when it comes to deciding a value for quantities like "good" or "mediocre", users tend to not have a clear answer.

Despite the lack of thorough knowledge about the reward function, it is crucial in reinforcement learning to find a way to obtain the optimal policy for an MDP. The limited information about the reward function can either be gained from a domain experts, or observed by an agent and enforced to the MDP as a set of constraints. We are looking for a robust policy that achieves the highest reward to the best of agent's knowledge. Please note that our focus in this paper is different than apprenticeship learning [5]. We do not try to mimic a demonstrator. But rather, we are trying to find a policy that can guarantee robustness.

One approach to tackle this problem is to use *maximin*. The agent gets to choose a policy, and an adversary chooses a reward function from a set of feasible rewards. What the agent is really doing, is to maximize the worst reward that the adversary chooses in order to beat the agent.

Dynamic Programming (DP) is one of the main methods of finding an optimal policy in the field of reinforcement learning. It is also used to frame the problem of robust MDPs. In case of full knowledge of MDP, Approximate Linear Programming (ALP) was successfully used in order to find an approximation to the optimal policy [1]. However, specifying the reward function is sometimes impossible. And, therefore, we are looking for ways to find the optimal policy, given a limited knowledge we have about an MDP.

In order to solve the problem of imprecisely determined MDP, we start off with formulating a DP as an LP [6]. The reason to choose LP to represent the problem is mainly because this formulation makes it possible to incorporate an imprecise constrained reward function into the problem in the form of a convex polytope, which is assumed to be bounded. This bounds on the reward function are observed from the MDP's behavior.

DeFarias et al. in [8] give the foundation of Approximate Linear Program, and the error bound for the approximation in LP approach to DP.

In [4] Marek et al. used L_1 regularization to improve the quality of approximation in ALP.

Van Roy expands the idea of approximation in value function to state aggregation in [9].

In [7], Regan et al. studied the effect of precision in specifying the reward function on finding the optimal policy. The key idea is to use minimax regret decision criterion, which is intuitively based on minimum regret or loss. The *minimax regret* approach minimizes the worst-case regret. One benefit of minimax (as opposed to expected regret) is that it is independent of the probabilities of the various outcomes: thus if regret can be accurately computed, one can reliably use minimax regret. However, probabilities of outcomes are hard to estimate.

Another contribution of their work is an elicitation criteria that guides the *querying process*. This, in turn, can give us a boundary for "how good" the optimal policy can be determined, or "how close" is the obtained optimal policy to the "real" optimal policy. Minimax criteria is employed in this work, which gives the maximum security level, or the best worst-case scenario. For reward elicitation they used an approach called "bound queries", which is simply finding boundary for $r(s,a)$ by having a user answering to the question "Is $r(s,a) \geq b$".

Huang's work in [2], is heavily based on inverse RL and apprenticeship learning in the sense of *expert demonstration*. Apprenticeship learning has a performance that is, at least, as good as an expert's policy given the set of all possible rewards. It can be formulated using a maximin. In this study, unlike apprenticeship, we want to only use the data that is coming from the expert, rather than strictly following an expert. By doing so, we guarantee more robustness to the worst case scenario.

2 Robust Optimization for MDPs

Finding $v^*(s)$ is achieved by a minimization over the vector $v(s)$ when $v(s) \geq r(s,a) + \sum_{j \in S} \gamma P(j|s,a)v(j)$ for all a's. We can formulate this for a single state as

$$\begin{aligned} &\min v(s) \\ s.t. \quad &v(s) \geq r(s,a) + \textstyle\sum_{j \in S} \gamma P(j|s,a)v(j) \quad \forall a \in \mathcal{A} \end{aligned}$$

Please note that the constraint consists of m inequality, for all actions that can take place on state s. The minimization can be generalized for all states as

$$\begin{aligned} &\min \textstyle\sum_{j \in S} \alpha(j)v(j) \\ s.t. \quad &v(s) \geq r(s,a) + \textstyle\sum_{j \in S} \gamma P(j|s,a)v(j) \quad \forall a, s \in \mathcal{A} \times \mathcal{S} \end{aligned}$$

And, in vector form it would be

$$\begin{aligned} &\min_v \alpha^\top v \\ s.t. \quad &Av \geq r \end{aligned}$$

In which $A_{nm*n} = E_{nm*n} - P_{nm*n}$, and E_{nm*n} is a matrix consist of m identity matrices of $n * n$ that is repeated m times vertically. Note that the constraints has $n * m$ inequality.

The dual of this minimization is:

$$\begin{array}{c} \max_u r^\top u \\ s.t. \quad A^\top u = \alpha \\ u \geq 0 \end{array}$$

In robust optimization we have two agents, one is trying to maximize the return, while the other one is minimizing it. This is how the best worst case scenario is picked. Applying the same idea to this problem we'll have

$$\begin{array}{c} \min_r \max_u r^\top u \\ s.t. \quad A^\top u = \alpha \\ u \geq 0 \end{array}$$

By strong duality we know that we can swap min and max.

$$\begin{array}{c} \max_u \min_r r^\top u \\ s.t. \quad A^\top u = \alpha \\ u \geq 0 \end{array}$$

The term $\min_r r^\top u$ is independent of the constraints, since the variable that we are minimizing over is r, and the constraint is not a function of r.

$$\begin{array}{cc} \max\limits_u & \min\limits_r r^\top u \\ A^\top u = \alpha & \\ u \geq 0 & \end{array}$$

This minimization problem, misses a crucial element; the bound over r (it'll go to $-\infty$ if r is not bounded). r can either be bounded by a set of limited feasible values such as $B = \{r_1, r_2, ...r_l\}$. Or, it can be considered to be bounded by a set of linear constraints $Cr \leq d$, where C is an $l \times nm$ matrix, where l is the number of constraints on r. Both approaches are studied as follows.

B as a Set

Given $r \in B = \{r_1, r_2, ...r_l\}$, we can reformulate the minimization as

$$\begin{array}{ccc} \begin{array}{c} \min_r r^\top u \\ r \in \{r_1, ..., r_l\} \end{array} & = & \begin{array}{c} \min \sum_{k=1}^{k=l} \beta_k r_k^\top u \\ \sum_{k=1}^{k=l} \beta_k = 1 \\ \beta_k \geq 0 \end{array} \\ & = & \begin{array}{c} \max \beta r^\top u \\ 1^\top \beta = 1 \\ \beta \geq 0 \end{array} \end{array}$$

Where $\beta = [\beta_1, \beta_2, ..., \beta_l]^\top$. The dual form of this problem is

$$\begin{array}{c} \max_t t \\ t < r_i^\top u \quad \forall i \leq l \end{array}$$

And the maximin problem becomes

$$\begin{array}{c} \max_{t,u} t \\ t < r_i^\top u \quad \forall i \leq l \\ A^\top u = \alpha \\ u \geq 0 \end{array}$$

B as a Polytope

The constraints can also be induced to the system using a set of linear inequalities. So, to solve the inner minimization problem we have:

$$\begin{array}{ccc} \min_r r^\top u & = \min_r r^\top u & = \min_r r^\top u \\ r \in B & Cr \leq d & -Cr \geq -d \end{array}$$

The dual of this problem is

$$\begin{array}{c} \max_t -d^\top t \\ s.t. \quad -C^\top t = u \\ t \geq 0 \end{array}$$

So, for the robust MDPs we have

$$\begin{array}{c} \max_{u,t} -d^\top t \\ s.t. \quad A^\top u = \alpha \\ u \geq 0 \\ -C^\top t = u \\ t \geq 0 \end{array}$$

Miscellaneous Notes

Minimax Regret for Imprecise MDPs

We know that $r \in \mathcal{R}$, where $\mathcal{R}$ is the feasible set for rewards, which reflects the current knowledge of the reward.

$$\begin{array}{c} R(f,r) = \max_{g \in \mathcal{F}} r.g - r.f \\ MR(f,\mathcal{R}) = \max_{r \in \mathcal{R}} R(f,r) \\ MMR(\mathcal{R}) = \min_{f \in \mathcal{F}} MR(f,\mathcal{R}) \end{array}$$

$R(f,r)$ is the regret of policy f (as represented by its visitation frequencies) relative to reward function r: it is simply the loss or difference in value between f and the optimal policy under r. $MR(f,R)$ is the maximum regret of f w.r.t. feasible reward set $\mathcal{R}$. Should we chose a policy with visitation frequencies f, $MR(f,R)$ represents the worst-case loss over all possible realizations of the reward function; i.e., the regret incurred in the presence of an adversary who chooses the r from $\mathcal{R}$ to maximize our loss. Finally, in the presence of such an adversary, we wish to minimize this max regret: $MMR(R)$ is the minimax regret of feasible reward set $\mathcal{R}$. This can be viewed as a game between a decision

maker choosing f who wants to minimize loss relative to the optimal policy, and an adversary who chooses a reward to maximize this loss given the decision makers choice of policy. Any f^* that minimizes max regret is a minimax optimal policy, while the r that maximizes its regret is the witness or adversarial reward function, and the optimal policy g for r is the witness or adversarial policy.

Inverse Reinforcement Learning

The problem of extracting a reward function using an observation that is made from the optimal policy [3]. The optimal policy is usually coming from an expert demonstration.

Implementation Considerations

In order to implement a robust MDP solver on the Invasive Species problem, we need to have the followings: Matrix A, which depends on $P_{s'}(s,a)$ (the model of the system), vector α, and either set B, or matrix C and vector d, which are the set of constraint on the reward.

However, in Invasive Species project, we are dealing with a large state space. We cannot represent $P_{s'}(s,a)$ in a tabular format. In such cases, it's impossible for the solver to obtain the optimal policy, unless we replace the tabular representations in the problem with an approximation, and in particular, a linear approximation.

Based on [1], we can formulate an approximate MDP solver as a Linear Program. The formulation looks very similar to the robust MDP solver, but, without the constraints on the reward function.

So, I believe if I can formulate an ALP, and solve the invasive species problem with that, given the current reward function, then I would be able to incorporate the reward elicitation part into the problem, and solve the problem as a whole.

One issue that we come across in ALPs is dealing with high number of samples. We are looking for the best possible way to pick a set of sample that best represent the state-space. (this part needs to be more studied from the Van Roy's paper)

Separation Oracles

An algorithm that outputs TRUE if a given point (inside the ellipsoid E_k) is actually inside the set $P : \{x|Ax \leq b\}$ (feasible set). And if not, it returns a half space (the separator plan or a hyperplane $(w,c) s.t. \quad w.y \leq c \quad \forall y \in P$) that the set P is on it.

Ellipsoid method

It's an alternative solution to an LP. It's slower than simplex in practice. The algorithm considers an ellipsoid in each iteration, and checks if the center of the ellipsoid

3 Questions

References

[1] D. P. de Farias and B. Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, 2003.

[2] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. Learning Safe Policies with Expert Guidance. (Theorem 1):1–17, 2018.

[3] Andrew y. Ng and Stuart Russell. algorithms for inverse reinforcement learning.

[4] Marek Petrik, Gavin Taylor, Ron Parr, and Shlomo Zilberstein. Feature Selection Using Regularization in Approximate Linear Programs for Markov Decision Processes. 2010.

[5] Pieter Abbeel and Andrew Y. Apprenticeship Learning via Inverse Reinforcement Learning. (346):1–2, 2004.

[6] PUTERMAN. Markov Decision Processes Discrete Stochastic Dynamic Programming Chapter 6: Discounted Markov Decision Problems. 1994.

[7] Kevin Regan and Craig Boutilier. Regret-based Reward Elicitation for Markov Decision Processes. pages 444–451, 2012.

[8] Benjamin Van Roy. Approximate Dynamic Programming via Linear Programming. *Comparative and General Pharmacology*, (1995):255–269, 2002.

[9] Benjamin Van Roy and Benjamin. Performance Loss Bounds for Approximate Value Iteration with State Aggregation. *Mathematics of Operations Research*, 31(2):234–244, may 2006.