GAUSSIAN PROCESSES IN INVERSE REINFORCEMENT LEARNING

ZHUO-JUN JIN , HUI QIAN, MIAO-LIANG ZHU

College of Computer Science, Zhejiang University, Hangzhou 310027, China
E-MAIL: jinzhuojun@zju.edu.cn, qianhui@zju.edu.cn, zhum@zju.edu.cn

Abstract:

Inverse reinforcement learning (IRL) is the general problem of recovering a reward function from demonstrations provided by an expert. By incorporating Gaussian process (GP) into IRL, we present an approach to recovering both rewards and uncertainty information in continuous state and action spaces. To predicate value in every point in spaces, we use GP models for value function and reward function separately. Our contribution is threefold: First, we extend the existing IRL algorithm to the case of continuous spaces. Second, reward GP provides not only the reward function with flexible forms, but also uncertainty about rewards, which helps the learner make a tradeoff between exploitation and exploration. Third, by introducing the kernel function, our approach takes sample points in the demonstration as learning features. It prevents manually designating features. Experimental results show the proposed method works well and demonstrate good learning in a traditional learning setting.

Keywords:

Inverse reinforcement learning; Gaussian process; Reward learning; Reinforcement learning; Markov decision process

1. Introduction

A central issue for Reinforcement learning (RL) algorithms is the speed of learning, that is, the number of trials necessary to learn a task. Many learning algorithms require a large number of trials to succeed. However, in practice the number of actual trials is very limited due to time or physical constraints. So our goal is to extract as more useful information as possible from available experience from an expert.

Russell gives the first formal treatment of Inverse Reinforcement learning (IRL) [1], which addresses the general problem of recovering a reward function from samples of a policy provided by an expert. This problem is discussed further in [2] and the first algorithm based on max-margin principle is given by Abbeel [3] .This problem in discrete state space and deterministic rewards settings has been explored in various ways in the literature [4-7]. Recently, Neu [8] presented a unified view of IRL algorithms. Our research is mainly motivated by three observations: First, most of real systems are inherently in continuous state and action spaces, Discretization facilitates computation but makes the results hardly apply to the original problem. Second, the assumption of a perfect expert is usually too strong in many real applications, so a probabilistic representation of rewards is necessary for the purpose of encouraging visiting the states whose rewards are still inaccurate.

There are two tasks closely correlated with IRL. One task is reward learning, which is in nature supervised learning. In the context of IRL, an expert provides label-like optimal actions, and then the IRL algorithm outputs the estimate of the reward function, which makes the given policy most likely happen. The other task is apprenticeship learning, which can be viewed as a generative learning. In this situation, the latent rewards can be regarded as the unobserved model. Both kinds of tasks assume the full access to optimal demonstrations from an expert. However, due to the absence of a perfect expert, it is more reasonable to treat the expert as a heuristic in real applications, or in Bayesian terminology, a prior. By taking the heuristic from the expert into account, we can get a faster leaner robust to environmental changes or an imperfect expert. The advantage of modeling the reward function by Gaussian process (GP) is that it allows for a fully flexible reward function in continuous state and action spaces. One common approach to generalizing RL and IRL problem to continuous-valued domains is the parametric function approximation, such as polynomials or radial basis function networks, etc. The key idea is to model the value function and the reward function in a function space rather than representing these functions in a table-based manner. However, a fundamental problem of the traditional parametric function approximation is that the basis functions are fixed before observing the training data. We shall deal with this problem by introducing the kernel method, as we shall discuss later.

Application of GP in RL is not a completely new idea, Rasmussen etc. offer promising prospects for some fundamental problems of reinforcement learning [9-10]. It is

978-1-4244-6527-9/10/$26.00 ©2010 IEEE

natural to extend this idea to IRL problems. Applying GPs to IRL yields not only sparse non-linear rewards in continuous spaces, but also the level of uncertainty about the reward function. It has been proved that uncertainty of rewards has beneficial effects on directing exploration in a Bayesian RL framework [11].

The rest of the paper is organized as follows: Section 2 gives a brief overview of MDP and IRL. Section 3 involves the GPIRL algorithm in detail. Section 4 and Section 5 presents the empirical results and conclusions.

2. Notation and Problem Formulation

We formalize our problem using Markov decision processes (MDP). In this section we review some background material on MDPs and IRL.

A Markov decision process (MDP) is a tuple $\{S, A, P, R, \gamma\}$, where S represents the state space, A is the action space, P is the transition probabilities, R is the reward function and γ is a discount factor. $P^{a}_{s,s'}$ denotes the probability of transitioning from state s to state s' when action a is taken. In this paper we are concerned with the problems with unknown rewards, everything else in the specification of the MDP is assumed to be known. A deterministic policy is a mapping π: $S \to A$. Associated with any such policy there is a value function:

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^{t} R(s_t, a_t)].$$

Here, the expectation is taken with respect to the random state sequence drawn by picking actions according to π. In standard reinforcement learning setting, the agent is to find a policy π maximizing the value function. Indeed, it has been proved (Sutton Barto Bertsekas Tsitsiklis 1996) for any given MDP there exists at least one optimal policy π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)$ for all $s \in S$.

In RL context, the consistency between short-term rewards and long-term returns are given by the well known Bellman equation. In a continuous state space, by substituting sums with integrals, Bellman equation is generalized straightforwardly to this form:

$$V^{\pi} = \int P^{\pi(s)}_{s,s'} R(s')ds' + \gamma \int P^{\pi(s)}_{s,s'} V^{\pi}(s')ds'.$$

This involves two parts. The first part corresponds to short-term rewards and the second part corresponds to expected long term rewards while following the given policy.

On the basis of the max-margin principle, IRL problem can be posed as an optimization problem of the form:

$$\begin{aligned} \max \quad & \tau \\ s.t. \quad & V_R(\pi^*) \geq V_R(\pi_i) + \tau \quad i = 1, \ldots, t-1, \\ & \|R\|_2 \leq 1 \end{aligned} \quad (1)$$

where π^* is the expert's policy, π_i denote the other policies. Intuitively, our goal is to make objective $V_R(\pi^*) - V_R(\pi_i)$ as large as possible.

3. Gaussian Processes in inverse Reinforcement learning

We shall first revisit GP models for regression. In GP models we put a prior directly on functions and condition on observations to make predictions. The noisy targets $y_i = f(x_i) + \varepsilon_i$ are assumed jointly Gaussian:

$$\mathbf{y} \sim N(\mathbf{0}, \mathbf{K}),$$

where $\mathbf{K}_{nm} = k(x_n, x_m)$. The covariance function k is also known as a kernel function. One commonly used stationary and non-isotropic kernel for Gaussian process regression is given by:

$$\begin{aligned} & k(x_n, x_m \mid \boldsymbol{\theta}) \\ & = v^2 \exp\{-\frac{1}{2}(x_n - x_m)^T \Lambda^{-1} (x_n - x_m)\} \end{aligned}. \quad (2)$$

Hyper-parameters are collected within the vector $\boldsymbol{\theta}$, which can be fit by maximizing the marginal likelihood using a standard optimization algorithm such as conjugate gradient method. In this section and the next we shall omit the explicit dependence on the hyper-parameters to keep the notation uncluttered.

The predictive distribution for a novel test input x^* is also Gaussian and given by:

$$\begin{aligned} & p(y^* \mid x^*, \mathbf{x}, \mathbf{y}) \\ & = N(\mathbf{k}^T \mathbf{K}^{-1} \mathbf{y}, c - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}) \end{aligned}, \quad (3)$$

where $c = k(x^*, x^*)$.

The combination of GP and IRL will be called GPIRL in the sequel. GPRL, which is highly correlated with GPIRL, has been widely investigated and became a well-developed approach in recent years. The key idea is to represent the value function at a finite number of support points, then generalize it to the entire space using GP. Therefore, given $P^{\pi}_{s_i,s'} = N(\mu_i, \boldsymbol{\Sigma}_i)$, the Bellman equation for the values at support points is:

$$\mathbf{V} = (\mathbf{I} - \gamma \mathbf{W} \mathbf{K}_v^{-1})^{-1} \mathbf{R}, \quad (4)$$

where

$$\mathbf{W}_{ij} = \left|\mathbf{\Lambda}^{-1}\Sigma_i + \mathbf{I}\right|^{-1/2} v^2 \exp\{-\frac{(s_j-\mu_i)^T(\mathbf{\Lambda}+\mathbf{\Sigma}_i)^{-1}(s_j-\mu_i)}{2}\}$$

and $\mathbf{K}_V$ denotes the covariance matrix. The reward function at support point s_i is given by:

$$\mathbf{R}_i = \int P_{s_i,s'}^{\pi(s_i)} R(s')ds'. \qquad (5)$$

Equations (4) and (5) will play an important role in iterative policy evaluation.

3.1. The GPIRL Algorithm

We assume that there is some vector of features $\varphi(s)$ over states, and that the reward function $R(s)$ is expressed in terms of a linear combination of these features:

$$R(s) = \mathbf{w}^T\varphi(s) = \sum_{n=1}^{M} w_m k(s, s_m),$$

where $\mathbf{w}$ is the M-dimensional weight vector. In particular, the basis functions are given by kernels, with one Gaussian kernel associated with each of the data points sampled from the demonstration trajectory. Notice we treat the rewards as the function of state for simplicity. The extension to the case of rewards $R(s,a)$ is trivial.

Now consider a prior distribution over rewards at finite support points. The distribution is given by:

$$\mathbf{R} \sim N(\mathbf{0}, \mathbf{K}),$$

where $\mathbf{K}$ denotes the Gram matrix with elements given by formula (2):

$$\mathbf{K}_{nm} = k(s_n, s_m \mid \mathbf{\theta}).$$

In particular, we choose the projection method [3] as the prototype and then extend it to the continuous state case. It is worth noting that similar extensions can be readily applied to other IRL algorithms. A detailed description of the GPIRL algorithm is given below.

TABLE 1. THE GPIRL ALGORITHM

Algorithm 1 the GPIRL algorithm

```
1.  Given: Model $MDP/R = \{S, A, P, \gamma\}$,
demonstration $O = \{s_0, \cdots, s_T\}$, set of support points
$S_{sp} = \{s_1, \cdots, s_m\}$.
2.  $\mathbf{w} = 1/\sqrt{|T|}$.
3.  Initialize $GP_R(m_R, \sigma_R^2)$, $\mathbf{R} \sim N(0, \mathbf{K}_R)$.
4.  Set $\varphi$, $\varphi_i = k(s, s_i)$ for $s_i \in O$.
5.  Compute $\mathbf{\mu}^*$, $\mu_i^* = E[\sum_{t=0}^{\infty}\gamma^t\varphi_i(s_t) \mid \pi^*]$.
6.  Repeat
7.      $R = \mathbf{w}^T\varphi$.
8.      $\pi$ = GPRL ($\{S, A, P, \gamma, R\}$, $S_{sp}$).
9.      Compute $\mathbf{\mu}$, $\mu_i = E[\sum_{t=0}^{\infty}\gamma^t\varphi_i(s_t) \mid \pi]$.
10.     $\mathbf{w} = \mathbf{\mu}^* - \mathbf{\mu}$ .
11.     $\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|$.
12. Until stabilization of $\mathbf{w}$ .
13. Train $GP_{\mathbf{R}}$ using $\mathbf{R} = [R(s_0), \cdots, R(s_T)]^T$ .
Return $m_R(s) = \mathbf{k}^T\mathbf{K}_R^{-1}\mathbf{R}$, $\sigma_R^2(s) = c - \mathbf{k}^T\mathbf{K}_R^{-1}\mathbf{k}$.
```

The above algorithm proceeds as follows. First of all, rewards were initialized uniformly. Then GP of rewards is initialized and features are set in line 3-4. Next the feature expectation of the demonstration $\mathbf{\mu}^*$ is evaluated. Line 6-12 involves the main loop. At each round, a new estimate of $\mathbf{w}$ is computed according to (1). This operation is repeated until $\mathbf{w}$ is stabilized. At last, training $GP_{\mathbf{R}}$ using rewards on support points yields the solution, which is a direct application of equation (3). Note that training Gaussian processes implicitly involves two parts: the distribution prediction and the hyper-parameters update, hyper-parameters can be fit using conjugate gradient optimization of the marginal likelihood.

3.2. The Policy Iteration Algorithm

In the main loop of the GPIRL algorithm, in order to obtain the optimal policy, we invoke the GPRL routine at each round, which is indeed a policy iteration procedure in continuous spaces. This algorithm is listed below.

TABLE 2 GPRL ALGORITHM

Algorithm 2 the GPRL algorithm

```
1.  Given: Model $MDP = \{S, A, P, \gamma, R\}$, set of
support points $S_{sp} = \{s_1, \cdots, s_m\}$.
2.  Initialize $GP_V(m_V, \sigma_V^2)$, $\mathbf{V} \sim N(0, \mathbf{K}_V)$.
3.  $\mathbf{V} = R(s_i)$ for $s_i \in S_{sp}$.
4.  Repeat
5.      Train $GP_{\mathbf{V}}$, $V(s) = \mathbf{k}^T\mathbf{K}_V^{-1}\mathbf{V}$.
6.      $\pi(s) = \arg\max_{a \in A} \int P_{s,s'}^a [R(s') + \gamma V(s')] ds'$.
```

```
7.      Compute $\mathbf{R}$, $\mathbf{R}_i = \int P^{\pi(s_i)}_{s_i,s'} R(s')ds'$ for all
$s_i \in S_{sp}$ .
8.      $\mathbf{V} = (\mathbf{I} - \gamma \mathbf{W} \mathbf{K}_V^{-1})^{-1} \mathbf{R}$ .
9.  Until stabilization of $\mathbf{V}$ .
10. Return $\pi$ .
```

Here $\mathbf{R}$ is an N-dimensional vector, whose i th element denotes the reward value at the i th support point. The loop of line 4-9 constitutes the main part of the algorithm. Line 6 involves policy improvement, Line 7-8 performs policy evaluation equations (4) and (5) respectively. In the main loop policy evaluation and policy improvement are alternatively performed to reach the equilibrium. The value function is represented only on support points, and is generalized to the entire state space by Gaussian process GP_{V} . In line 5, equation (3) is again used for predication.

4. Experiments

We validate the proposed method by applying it to a continuous grid-world problem (we still call it grid-world to follow conventions). As known, in cases where the input vector is two dimensional, GP is also known as a Gaussian random field. So the grid-world in our experiment is a squared random field with both the width and the height set to 5. The state thereby takes the form $s = [x, y]^T$, where $x, y \in [0,5]$. The Action is defined by $a \in [0, 2\pi)$, where the quantity denotes the angle between the moving direction and the north direction.

The automated agent is initially located at the lower left corner and supposed to reach the goal in the upper right corner. Assuming a demonstration is provided, as depicted in Figure 1. Features are sample points in the demonstration, here we choose points (1, 1), (2, 2), (3, 3) and (4, 4). To simplify matters, we assumed the access to the dynamic model, which we defined as:

$$P^a_{s_i,s'} = N(\mu_i, \Sigma_i) = N(s + \begin{bmatrix} \sin a \\ \cos a \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}),$$

which is a Gaussian distribution with the mean centering on the consecutive state and the covariance a unit matrix. In practical applications, the selection of the support points is on a case-by-case basis. For example, in an online setting, support points could naturally be chosen as the state visited. In our experiment, for simplicity of exposition, we use simply a regular grid of support points.

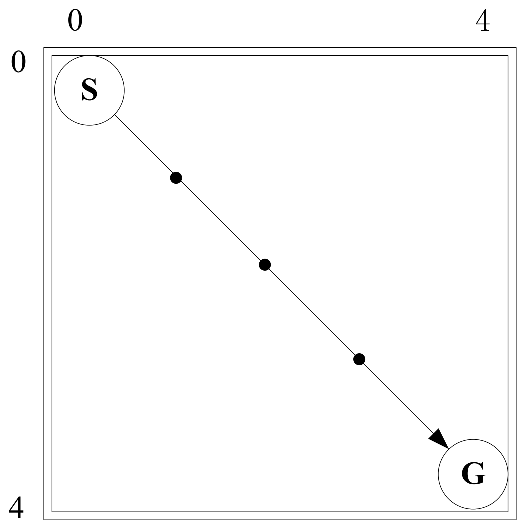


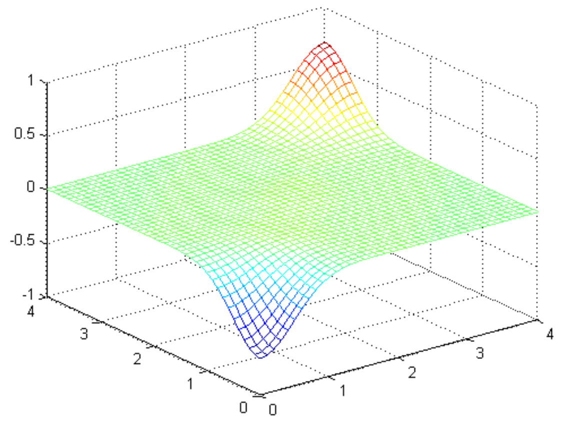
Figure 1. The demonstration of the expert. Features are shown as black dots.

Figure 2(a) plots the recovered reward. Figure 2(b) plots the level of uncertainty, where larger values indicate higher levels of uncertainty. Note that the path near the middle area has lower quantities, while the quantities on the edge are relatively higher because the demonstration has not sufficiently explored areas far from the center.

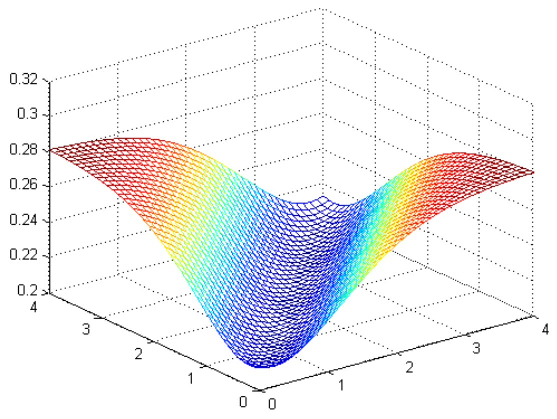
Figure 3 shows the corresponding value function. Benefiting from the continuous reward function and the continuous dynamic model, the value function is also represented in continuous spaces. From the value function, we can derive the optimal policy by simply choosing the gradient as the target direction. So, there is an optimal action corresponding to each point in the whole field.

5. Conclusions and Future Work

To summarize, in the current paper, first a brief review of IRL and GP is given. Then by incorporating GP into IRL, we proposed an approach to recovering continuous rewards with the corresponding variance. At last, a grid-word like experiment is conducted to validate the proposed method. As we known, probabilistic models appropriately quantify knowledge, alleviate model bias, and lead to very data-efficient solutions. In RL, the variance of rewards can help the agent favor actions that explore uncertain regions. Our contribution is threefold: First, we extend the projection method to the case of the continuous state space. Second, incorporating Gaussian processes provides not only the rewards, but also uncertainty about them. Third, our approach uses sample points in the demonstration as features so as to avoid manually designing the learning features.



(a)



(b)

Figure 2. (a) The recovered reward function. (b) Uncertainty about the reward function.

One noteworthy limitation of GPIRL is its scalability. As an instance-based algorithm, its time and space complexity grow with the amount of data it collects. It should be possible to stop adding data after reaching a fixed threshold, or keep the most useful samples without introducing significant bias. A simple solution is to stop adding data if the dataset grows too large to update the model efficiently. An alternative way is to reduce the data points by sparse Bayesian method [12].

Finally, it is also an interesting future research direction that extending our algorithm to the case of unknown dynamics. On the other hand, in this paper we only consider the algorithm in a batch setting and solve the problem in an iterative manner. It might be promising to generalize our method to an online setting.

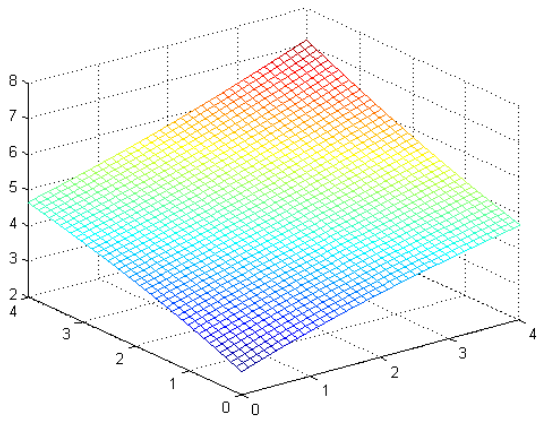


Figure 3. The continuous value function derived from the learned rewards.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (No. 90820306). We'd like to thank the reviewers for the hard work and enlightening comments.

References

[1] Russell, S., "Learning agents for uncertain environments (extended abstract)", Proceeding of *11th Annual Conference on Computational Learning Theory*, Madison, Wisconsin, USA, pp. 101-103, July 1998.

[2] Ng, A. and Russell, S., "Algorithms for Inverse Reinforcement Learning", Proceeding of *17th International Conference on Machine Learning*, San Francisco, USA, pp. 663-670, June 2000.

[3] Abbeel, P. and Y. Ng, A., "Apprenticeship learning via inverse reinforcement learning", Proceeding of *21th International Conference on Machine learning*, Alberta, Canada, pp. 1-8, June 2004.

[4] Ratliff, D.N., Bagnell, J.A., and Zinkevich, M., "Maximum margin planning", Proceeding of *23rd International Conference on Machine learning*, Pittsburgh, USA, pp. 729-736, June 2006.

[5] Neu, G. and Szepesvari, C., "Apprenticeship learning using inverse reinforcement learning and gradient methods", Proceeding of *23rd Conference Conference on Uncertainty in Artificial Intelligence* Vancouver, British Columbia, Canada, pp. 295-302, July 2007.

[6] Syed, U., Bowling, M., Schapire, R. E., "Apprenticeship learning using linear programming", Proceeding of *25th International Conference on Machine Learning* Helsinki, Finland., pp. 1032-1039, July 2008.

[7] Ziebart, B., Maas, A., Bagnell, J., and Dey, A., "Maximum entropy inverse reinforcement learning", Proceeding of *23rd National Conference on Artificial intelligence*, Chicago, Illinois, pp. 1433–1438, July 2008.

[8] Neu, G. and Szepesvári, C., "Training parsers by inverse reinforcement learning" *Machine Learning,* vol. 77, No. 2, pp. 303-337, 2009.

[9] Rasmussen, C. and Kuss, M., "Gaussian processes in reinforcement learning" *Advances in Neural Information Processing Systems,* vol. 16, No. 1, pp. 751–759, 2004.

[10] Deisenroth, M.P., Rasmussen, C.E., and Peters, J., "Gaussian process dynamic programming" *Neurocomputation,* vol. 72, No. 7-9, pp. 1508-1524, 2009.

[11] Dearden, R., Friedman, N., and Russell, S., "Bayesian Q-learning", Proceeding of *The 10th National Conference on Artificial Intelligence* Madison, Wisconsin, United States, pp. 761-768, July 1998.

[12] Tipping, M., "Sparse Bayesian Learning and the Relevance Vector Machine" *Journal of Machine Learning Research,* vol. 1, No. 1, pp. 211-244, 2001.