A Theory Project Report on Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization

Chia-Chun Chung
Department of Computer Science
National Chiao Tung University
zivzhong.cs07g@nctu.edu.tw

1 Introduction

Reinforcement learning can learn some very complicated movement, but usually need to well define a reward function. However, peopole is hard to define the spefic reward function to let agent to learn. So here comes a research area called inverse reinforcement learning which we want to learn a reward function first, and then use this reward function to do reinforcement learning.
But there are two ket problem this paper mention that they want to solve:

- Common inverse reinforcement learning is hard to use under unknown dynamics for high-dimensional continuous systems
- Common inverse reinforcement learning usually need to need for informative features and effective regularization to impose structure on the cost

To address these two key issue, this paper formulate an efficient sample-based approximation for MaxEnt IOC for the first one, which is based on the work of Ziebart and Dey [2008], and use neural network as reward function for the second one.
Comparing this paper with other former works, the key contributions is this paper make the reward function is learned in the inner loop of a policy search procedure, using samples collected for policy improvement to also update the cost function. Other former works usually finding an optimal policy given the current reward function in a inner loop and learn reward function in other inner loop which will cause the system is inefficient and hard to apply to high , high-dimensional and continuous real world environment with unknown dynamics.

2 Problem Formulation

This paper is based on the Ziebart and Dey [2008], so this paper assume demonstrated behavior is the result of an expert acting randomly and near-optimally under an unknown reward function. The demonstration is also the target behavior we want agent to mimic. This paper also assumes that the expert samples the demonstrated trajectories $\{ \tau_i \}$ from the distribution:

- $P(\tau) = \frac{1}{Z} \exp\left(-c_\theta(\tau)\right)$
- $\tau = x_1, u_1, ...x_T, u_T$ is a trajectory sample, which x_t is the state and the u_t is the action at time t
- $c_\theta(\tau) = \sigma_t c_\theta(x_t, u_t)$ is the unknown reward function and is what we want to learn, and is parameterized by θ.

With this assumption, the expert will do the optimal action mostly, but may do some sub-optimal action with a propabilty.

Preprint. Under review.

The key part is that partition function Z is hard to compute under huge and continue domains. So this paper choose to use a sample-based approach which is estimated with the samples from a background distribution to solve this problem:

- $Z = \int \exp(-c_\theta(\tau))d\tau$

The negative log-likelihood corresponding to the MaxIOC model will be:

- $L_{IOC} = \frac{1}{N} \sum_{\tau_i \in D_demo} c_\theta(\tau_i) + \log Z$
- D_{demo} means the set of N demo trajtories.

But like what we mention before, its hard to compute the Z value, so we use sample here:

- $L_{IOC} \approx \frac{1}{N} \sum_{\tau_i \in D_demo} c_\theta(\tau_i) + \log \frac{1}{M} \sum_{\tau_i \in D_samp} \frac{\exp(-c_\theta(\tau))}{q(\tau_j)}$
- D_{samp} means the set of M background samples
- q denotes the background distribution from which trajectories τ_j were sampled, and we can choose unifrom here.

Finally, we can compute the gradient:

- $\frac{dL_{IOC}}{d\theta} = \frac{1}{N} \sum_{\tau_i \in D_demo} \frac{dc_\theta(\tau_i)}{d\theta} - \frac{1}{Z} \sum_{\tau_i \in D_samp} \omega_j \frac{dc_\theta(\tau))}{d\theta}$
- $\omega_j = \frac{\exp(-c_\theta(\tau))}{q(\tau_j)}$
- $Z = \sum_j \omega_j$
- Then if we have a reward function built by neural network, we can get gradient easily by backpropagating $-\frac{\omega_j}{Z}$

3 Theoretical Analysis

First, let's take a look about the whole algorithm

Algorithm 1 How to write algorithms

Result: optimized cost parameters and trajectory distribution q( )

Initialize qk( ) as either a random initial controller or from demonstrations **while** *iteration i != I* **do**

  Generate samples Dtraj from qk( )
  Append samples: Dsamp Dsamp Dtraj
  Use Dsamp to update cost c
  Update qk( ) using Dtraj and the method from (Levine Abbeel, 2014) to obtain qk+1( )

end

The authur choose their prior work as policy optimization part. And we can see the authur combine the reward learning and policy optimization part together which can let the learning process run efficiently.And the trajectory distributions are Gaussian, and each iteration of the policy optimization procedure satisfies a KL-divergence constraint, which has the additional benefit of avoiding overfitting to poor initial estimates of the cost function.

4 Conclusion

The contribution of this paper:

- This paper use neural network to approximate the reward function which means that can learn complex, nonlinear reward function, and can be applied to high-dimensional systems with unknown dynamics.

- Using a sample-based approximation of the MaxIOC objective, which can let whole computation efficiently.
- This paper is the first paper to leverage the powerful of sample-based IOC under unknown dynamics with nonlinear reward function that directly use the raw state as input, without feature engineering manually.
- The author also try to deploy the algorithm on the real world robotic and get the indeed get the good result.

The potential future research directions:

- Inverse reinforcement learning seems that can learn more suitable reward function to let reinforcement learning perform well.
- The key spirit of inverse reinforcement learning is quite like the GAN or min-max game which want the reward function can give the expert trajectories higher score, and give lower score to the non-expert trajectories. Maybe we can have a survey on the work which combine GAN and inverse reinforcement learning.

References

Maas A. Bagnell J. A. Ziebart, B. and A. K. Dey. Maximum entropy inverse reinforcement learning. *AAAI Conference on Artificial Intelligence,*, 2008.