Statistical Techniques in Robotics (16-831, S22) Lecture #22 (Monday, April 11)

Imitation Learning

Lecturer: Kris Kitani *Scribes: Xuanbai Chen, Tzu-Hsuan Yang (Group 1)*

1 Review

In the last lecture, we continue talking about the RL approaches. At this time, we mainly focused on the Policy-based approaches which will be mainly included in Section 1.1 and the Actor-Critic method, which is the hybrid of Policy-based and Value-based methods, in Section 1.2

1.1 Policy Gradient Methods

By leveraging the gradient ascent, we can update the parameters to obtain the best policy that maximizes the expected return. The objective of policy gradient method is listed below:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{p_\theta(\zeta)} \left[\sum_{t=0}^{T} r^{(t)} \right] \tag{1}$$

$$= \arg\max_{\theta} J(\theta) \tag{2}$$

By changing the linear approximation of the objective function with quadratic regularization and solving for the parameter that optimizes the Lagrangian, we can make the optimization process as follows:

$$\nabla_\theta \{\alpha(J(\theta') + \langle \theta - \theta', \nabla'_\theta J(\theta') \rangle - \frac{1}{2}||\theta - \theta'||^2)\} = 0$$

$$\Rightarrow \theta \leftarrow \theta' + \alpha \nabla_{\theta'} J(\theta')$$

From the definition of $\nabla_\theta J(\theta)$, we can derive that it equals to:

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(n,t)}|s^{(n,t)}) \right) \left(\sum_{t=1}^{T} r^{(n,t)} \right)$$

1.1.1 Monte-Carlo Policy Gradient

By plugging the equation above to the Monte-Carlo algorithm, we can acquire the algorithm of Monte-Carlo policy gradient below.

Algorithm 1 MC-Policy-Gradient(π_θ, a)

for $e = 1, \cdots, E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  $G^{(0)} = \sum_{t=0}^{T} r^{(t)}$
  for $t = 0, \cdots, T$ **do**
    $\theta \leftarrow \theta + \alpha\Big(\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})\Big)\Big(G^{(0)}\Big)$
  end for
end for
return π_θ

However, there exists a big problem which the variance of the algorithm is high. Two methods can ease the problem, which the former is to enforce causality and the latter to remove gradient bias with a baseline offset. The principle enforcing causality is to depend only on future reward at time t.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a^{(n,t)}|s^{(n,t)}) \right) \left(\sum_{t'=t}^{T} r^{(n,t')} \right)$$

By adjusting the gradient By subtracting some offset b, we can adjust the gradient and reduce the problem of large gradient variance,

$$\nabla_\theta J(\theta) = \int_\theta p_\theta(\zeta) \nabla_\theta \ln p_\theta(\theta) r(\zeta) d\zeta$$

1.1.2 Policy Gradient

By using these two strategies and putting them together, we can obtain the policy gradient algorithm as follows:

Algorithm 2 Policy Gradient(π_θ, α, b)

for $e = 1, \cdots, E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \mathcal{E}|\pi_\theta$
  for $t = 0, \cdots, T$ **do**
    $G^{(t)} = \sum_{i=t}^{T} r^{(i)}$ (1) Enforce causality
    $\theta \leftarrow \theta + \alpha[G^{(t)} - b]\nabla_\theta \log \pi_\theta(a^{(t)}|s^{(t)})$ (2) Subtract baseline
  end for
end for
return π_θ

In conclusion, the positives of the policy gradient are as follows: (1) Doesn't require model and learns from interaction. (2) Effective for high-dimensional or continuous action spaces (all methods so far assumed finite action space). (3) Can encode prior knowledge when designing policy architecture. (4) Finds the optimal stochastic policy and naturally explores due to stochasticity. While the negatives of the policy gradient are as follows: (1) Gradient is typically high variance and leads to slow convergence. (2) Small step size leads to slow convergence (3) Typically converge to local minima instead of global minima. (4) Exploding or zero gradient.

1.2 Actor Critic

In the policy gradient algorithm, we need to acquire $\sum_{t=1}^{T} r^{(t)}$ to compute the gradient. Actor Critic methods can use the model-free prediction for the return value estimator. The equation are as below:

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)})(G_\phi^{(t)})]$$

where G_ϕ is function approximation of the return value parameterize by ϕ. After removing the gradient bias of policy gradient, we can get the advantage function.

$$A(a^{(t)}, s^{(t)}) = Q_\phi(a^{(t)}, s^{(t)}) - V_\psi(s^{(t)})$$

So the policy gradient with advantage function is as follow:

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta}[(\sum_{t=1}^{T} \nabla_\theta \ln \pi_\theta(a^{(t)}|s^{(t)}))(\sum_{i=t}^{T} A_\phi(a, s))]$$

The Advantage-MC-Actor-Critic algorithm and Advantage-TD-Actor-Critic algorithm are as below:

Algorithm 3 Advantage-MC-Actor-Critic($\pi_\epsilon, Q_\phi, V_\psi, \alpha, \beta, \kappa$)

for $e = 0, \cdots, E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \epsilon|\pi_\epsilon$
  for $t = 0, \cdots, T$ **do**
    $G^{(t)} \leftarrow \sum_{t=1}^{T} r^{(i)}$ — MC estimate
    $\psi \leftarrow \psi + \kappa[G^{(t)} - V_\psi(s^{(t)})]\nabla_\psi V_\psi(s^{(t)})]$ — V update
    $\phi \leftarrow \phi + \beta[G^{(t)} - Q_\phi(a^{(t)}, s^{(t)})]\nabla Q_\phi(a^{(t)}, s^{(t)})$ — Q update
    $\theta \leftarrow \theta + \alpha\nabla \log \pi_\theta(a^{(t)}|s^{(t)})\dot{Q}_\phi(a^{(t)}, s^{(t)})$ — Policy update
  end for
end for

Algorithm 4 Advantage-TD-Actor-Critic($\pi_\epsilon, Q_\phi, V_\psi, \alpha, \beta, \kappa$)

for $e = 0, \cdots, E$ **do**
  $\{s^{(t)}, a^{(t)}, r^{(t)}\}_{t=0}^{T} \sim \epsilon|\pi_\epsilon$
  for $t = 0, \cdots, T$ **do**
    $\delta_V \leftarrow r^{(t)} + V_\psi(s^{(t+1)})$ — TD V estimate
    $\psi \leftarrow \psi + \kappa[\delta_V - V_\psi(s^{(t)})]\nabla_\psi V_\psi(s^{(t)})]$ — V update
    $\delta_Q \leftarrow r^{(t)} + Q_\phi(a^{t+1}, s^{(t+1)})$ — TD Q estimate
    $\phi \leftarrow \phi + \beta[\delta_Q - Q_\phi(a^{(t)}, s^{(t)})]\nabla Q_\phi(a^{(t)}, s^{(t)})$ — Q update
    $\theta \leftarrow \theta + \alpha\nabla \log \pi_\theta(a^{(t)}|s^{(t)})\dot{Q}_\phi(a^{(t)}, s^{(t)})$ — Policy update
  end for
end for

2 Summary

2.1 Imitation Learning

2.1.1 Motivation for Imitation Learning

As what we have learned before, the training process of the reinforcement learning algorithm is that the model learns on its own by a combination of exploration and exploitation of its environment and tries to maximize a scalar reward to get the policy. However, this training process can take a very long time to lean a policy because it has to learn everything from experience. To think of a better (faster) way to learn, we then have the Imitation Learning also known as Learning from Demonstration, apprenticeship learning, behaviour cloning, inverse reinforcement learning. Just as the algorithm name, the Imitation Learning (IL) learns a policy from demonstrated expert behaviour. In addition, sometimes the reward function defined in the reinforcement learning is too complex to define, it is easier to just show examples as what have be given in the IL. There are three types of IL algorithm, including Passive IL, Active IL, and Interactive IL, which will be discussed into detail later.

2.1.2 Definition for Imitation Learning

The Imitation Learning can be formally defined as :

$$\mathcal{A}_{\text{IL}} : \langle \mathcal{D}^*, \pi^*, \mathcal{T}, \mathcal{R} \rangle \rightarrow \pi \tag{3}$$

And this formula includes the following parts

Expert Demonstrations

$$\mathcal{D}^* = \{\boldsymbol{\zeta}_n\}_{n=1}^N \sim \mathcal{E} \mid \pi^* \tag{4}$$

Expert demonstrations are trajectories, denoted as ζ_n, sampled from the optimal policy π^*, which is used by IL algorithm to learn a good policy. These trajectories are a sequence of variables such as a sequence of states and actions.

$$\zeta = \left\{ s^{(0)}, s^{(1)}, \ldots, s^{(T)} \right\} \tag{5}$$

$$\zeta = \left\{ s^{(0)}, a^{(0)}, s^{(1)}, \ldots, a^{(T-1)}, s^{(T)} \right\} \tag{6}$$

Eq.(5) shows the expert demonstrations in state sequence and Eq.(6) gives the state-action pair sequence in expert demonstrations. We also discussed in the lecture that we can have other different forms for expert demonstrations depends on specific algorithms.

Optimal Policy (Oracle)

$$a \sim \pi^*(s) \tag{7}$$

The optimal policy also called oracle is where expert demonstrations are sampled. In the IL algorithm, this is not always accessible during the training process. The optimal policy is only available for interactive IL, which is, during the learning, we do not get the entire policy, but we can get samples from it (as the equation shows). For example, during the learning, the algorithm can query the optimal policy at every time step, and this interactive oracle feedback would tell you what action you should but not gives you the full parametric form.

Dynamics Function

$$\mathcal{T} = P\left(s' \mid s, a\right) \tag{8}$$

Dynamic Function shows the probability that an action a in the state s leads to state s' and . In the IL algorithm, this is also not always accessible during the training process. Only active IL and interactive IL can optionally take this into training progress.

Reward Function

$$\mathcal{R} = P\left(s, a, s'\right) \tag{9}$$

Similar as the Reinforcement Learning, the reward function in imitation learning is also defined as what the agent achieves at a particular state s, taking action a and reaches next state s'. In the next section, we can find that the reward function of some algorithms can be sampled from the environment, while the others can't.

2.2 3 Types of Imitation Learning

As we discussed in the previous section, based on what information the model could access to, we can divide the IL algorithm into three categories.

	Passive IL	Active IL	Interactive IL
Demonstrations $\mathcal{D}$	yes	yes	optional
Environment $\mathcal{E}$	no	yes	yes
Oracle π	no	no	yes
Dynamics $\mathcal{T}$	no	optional	optional
Reward $\mathcal{R}$	no	optional	optional

Table 1: 3 Types of Imitation Learning

2.2.1 Passive Imitation Learning

Passive Imitation Learning also known as supervised learning, behaviour cloning only have access to demonstrations. Passive IL involves learning a policy that performs nearly as well as an expert's policy based on a set of trajectories of that policy. However, if the agent visits the state that the expert trajectories never learns, the agent doesn't know what to do at this state and may cause error. With many unseen states in out test distribution, the accumulation of small errors will

lead to Covariate Shift. In other words, the Passive IL works well if the problem is not difficult and having expert demonstrations responds to all possible states then the model knows how to behaviour when encountering every possible state. Generally speaking, the Passive IL can only models short-term (one-shot) behaviour and cannot model long-term behaviour well.

Taking that learning how to play Golf as an analogy. The objective policy function is to being experts in this sport and know how to play. For passive learning, people don't have any access to the real golf court. They are only given a book contained with the instructions of how to play golf (seems like the expert demonstrations). Based on these expert demonstrations, they have to consider how to play golf in the final testing process.

Figure 1: Passive Imitation Learning Paradigm

2.2.2 Active Imitation Learning

In the Passive IL we knows that if we have the full expert trajectories, our models can learn well. However, generating such trajectories is often tedious or even impractical for an expert (e.g., real-time low-level control of multiple game agents). In order to address this issue, active Imitation Learning have accessed to expert demonstrations and environments and can optionally access the dynamics and rewards. In other words, the model can learn policy from expert demonstrations and have opportunities to implement those learned policy in the environment and to improve it. However, sometimes just mimicking expert demonstrations is not helpful because we can't generalize to new environments. In this scenario, understanding why (reward function) we take an action help us to generalize our learning better. And this lead to Inverse Reinforcement Learning, a type of Active IL, which will be discussed in the later section.

Similar as passive imitation learning, people are still given a book contained the instructions of how to play golf in active imitation learning. However, in this time, they have the chance to enter the golf court for practising. They can experience what the real environment is like and conduct lots of experiments in it based on the instruction book (expert demonstrations).

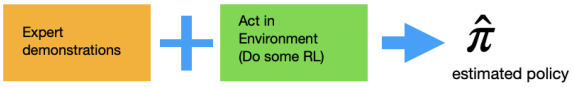

Figure 2: Active Imitation Learning Paradigm

2.2.3 Interactive Imitation Learning

Interactive Imitation Learning accesses the expert demonstrations optionally. Instead, it accesses the optimal policy (oracle) during the learning process. This iterative algorithm assumes that

we have accessed to an interactive demonstrator at the same time during the training, who we can query. This involves learning some demonstrations from the expert, and applying supervised learning to learn a policy.Then rolling out this policy in the environment, and querying the expert to evaluate the roll-out trajectory. In this way, we get more training data, which we feedback to supervised learning.

As for the interactive imitation learning algorithm, people can also act in the environment to do exercises for playing the golf. However, instead of only obtaining the original static instructions, they can get interactive feedback this time. To be specific, it seems like there exists a teacher helping you on the skills of playing golf. He can provide a unique instruction for you. If you perform poor in swinging, they can design corresponding exercise for you to improve the swinging skills.

One simple concern is that since we have already had access to the environment and the expert demonstrations, why can't we just get all demonstrations and the optimal policy? The reason is that we only get access to samples of the optimal policy and demonstrations. In this way, we have no access to the whole optimal policy.

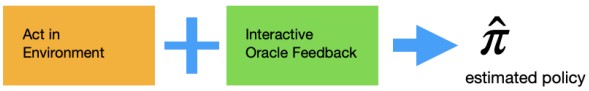

Figure 3: Interactive Imitation Learning Paradigm

2.3 Inverse Reinforcement Learning(IRL)

2.3.1 Motivation for Inverse Reinforcement Learning

Inverse Reinforcement Learning is a type of active IL, involves learning an expert's objectives, values, or rewards with the aid of using insights of its behavior. In this algorithm, the model not only learns an estimated optimal policy but also attempt to learn an reward function.

2.3.2 IRL vs RL

Model-based Reinforcement Learning learns the optimal policy by the dynamics model and reward function. However, sometimes the reward function can be really hard to define. Therefore, in IRL, We learn the reward function from the expert demonstrations and feed this reward function into the RL training process to get the optimal policy. The advantage of using IRL is that the expert demonstrations can be transferred to new (but similar) environments, not just mimicry. The output of IRL is a reward function.

In this setting, the Inverse reinforcement learning is crucial since some reinforcement learning algorithm does not show good generalization ability when leveraging in a new environment. The reason is that sometimes some RL algorithms just simply mimic the experts demonstrations, they have no idea about the principle of the whole environment. In this way, understanding why (reward function) we take an action can help us to generalize our learning better.

Figure 4: Inverse Reinforcement Learning Paradigm

2.3.3 5 Types of IRL Algorithm

2.3.4 Linear Program IRL

Introduced in [2], the linear function approximation in large state spaces is defined as follows:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{n} \hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_n}(s_0) \right\} \tag{10}$$

$$s.t. \, |\theta_d| \leq 1 \quad \forall d \tag{11}$$

2.3.5 Matrix Game IRL

The method [4] is based on a game-theoretic view of the problem, which leads naturally to a direct application of the multiplicative-weights algorithm for playing repeated matrix games. The equation is defined as follow:

$$\min_{\theta} \left\{ \max_{\pi} \theta \cdot \mu(\pi) - \theta \cdot \mu(\pi_E) \right\} \tag{12}$$

2.3.6 Max Margin IRL

They set an assumption that the expert tries to maximize a reward function that is expressible as a linear combination of known features, and gives an algorithm for learning the task demonstrated by the expert. Abbeel *et al.* [1] design the method that is based on using "inverse reinforcement learning" to try to recover the unknown reward function.

$$\begin{array}{ll} \min_{\theta} & \lambda \|\theta\|_2 + \sum_i \xi_i \\ \text{s.t.} & \theta^\top (\mu(\pi^*) - \mu(\pi_i)) \geq (1 - \xi_i) \quad \forall i \end{array} \tag{13}$$

2.3.7 Structured Output Max Margin IRL

Ratlif *et al.* [3] frame learning imitation learning of sequential, goal-directed behavior as a maximum margin structured prediction problem over a space of policies.

$$\min_{\theta} \frac{1}{2} \|\theta\|_2^2 + \frac{\lambda}{D} \sum_{d} \beta_d \left\{ \max_{\eta} \left(\theta^\top F_d + l_d^\top \right) \eta - \theta^\top F_d \eta_d \right\} \tag{14}$$

2.3.8 Maximum Entropy IRL

[5] introduces a probabilistic approach based on the principle of maximum entropy by providing a well-defined, globally normalized distribution over decision sequences.

$$\min_{\theta} \sum_{\zeta} p(\zeta;\theta) \log p(\zeta;\theta) \tag{15}$$

$$\text{s.t.} \sum_{\zeta} p(\zeta;\theta)\mu(\zeta) = \sum_{d} p_u\left(\zeta_d\right) \mu\left(\zeta_d\right) \tag{16}$$

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[3] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[4] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

[5] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

3 Appendix

In this section, we would like to introduce some foundational methods for IRL, which can mainly be divided into four classes: margin based optimization, entropy based optimization, Bayesian inference, classification, and regression. In this scribe note, we mainly introduce the first two types of methods.

3.1 Margin Optimization

Maximum margin prediction aims to learn a reward function that explains the demonstrated policy better than alternative policies by a margin. The methods under this category aim to address IRL's solution ambiguity by converging on a solution that maximizes some margin. We broadly organize the methods that engage in margin optimization based on the type of the margin that is used.

3.1.1 Margin of optimal from other actions or policies

One of the earliest and simplest margins chosen for optimization is the sum of differences between the expected value of the optimal action and that of the next-best action over all states,

$$\sum_{s \in S} Q^{\pi}(s, a^*) - \max_{a \in A | a^*} Q^{\pi}(s, a)$$

where a^* is the optimal action for s.

An early and foundational method that optimized the margin given in equation above is [A. Ng, S. Russell, Algorithms for inverse reinforcement learning], which takes in the expert's policy as input. It formulates a linear program to retrieve the reward function that not only produces the given policy as optimal output from the complete MDP, but also maximizes the margin shown above. In addition to maximizing this margin, it also prefers reward functions with smaller values as a form of regularization.

3.1.2 Margin of observed from learned feature expectations

Adoption of the feature-based reward function led to several methods optimizing margins that utilized feature expectations. Some of these methods seek a reward function that minimizes the margin between the feature expectations of a policy computed by the learner and the empirically computed feature expectations from the expert's trajectory.

$$|\mu^{\phi}(\pi) - \hat{\mu}^{\phi}(D)|$$

We refer to this margin as the feature expectation loss.

Two foundational methods [P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning] that maximize the feature expectation loss margin of the equation above are max-margin and projection. Noting that the learner does not typically have access to the expert's policy, both these methods take a demonstration as input. The methods represent the reward function as a linear, weighted sum of feature functions.

3.1.3 Observed and learned policy distributions over actions

An alternative to minimizing the feature expectation loss is to minimize the probability difference between stochastic policies

$$\hat{\pi}_E(a \mid s) - \pi_E(a \mid s)$$

for each state. As the behavior of expert is available instead of its policy, the difference above is computed using the empirically estimated state visitation frequencies and the frequencies of taking specific actions in the states. HYBRID-IRL [G. Neu, C. Szepesv´ari, Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods] uses the equation above in the margin optimization problem, solving the optimization using gradient descent in the space of reward hypotheses.

3.2 Entropy Optimization

IRL is essentially an ill-posed problem because multiple reward functions can explain the expert's behavior. The maximum margin approaches of Section 3.1 introduce a bias into the learned reward function. To avoid this bias, multiple methods take recourse to the maximum entropy principle [E. T. Jaynes, Information theory and statistical mechanics] to obtain a distribution over behaviors, parameterized by the reward function weights. According to this principle, the distribution that maximizes the entropy makes minimal commitments beyond the constraints and is least wrong. We broadly categorize the methods that optimize entropy based on the distribution, whose entropy is being used, that is chosen by the method.

3.2.1 Entropy of the distribution over trajectories or policies

We may learn a reward function that yields the distribution over all trajectories with the maximum entropy

$$\max_{sd} - \sum_{r \in (S \times A)^l} Pr(\tau) \log Pr(\tau)$$

while being constrained by the observed demonstration. However, the search space of trajectories in this optimization $(S \times A)^l$ grows exponentially with the length of the trajectory l. To avoid this disproportionate growth, we may learn a reward function that alternately yields the distribution over all policies with the maximum entropy.

$$\max_{sd} - \sum_{\pi \in (S \times A)} Pr(\pi) \log Pr(\pi)$$

where sd is the space of all distributions. Notice that the space of policies grows with the sizes of the state and action sets as $O(\mid A \mid^{\mid S \mid})$ but not with the length of the trajectory.

A foundational and popular IRL technique [B. D. Ziebart, A. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning] MAXENTIRL optimizes the entropy formulation while adding two constraints. First, the distribution over all trajectories should be a probability distribution. Second, the expected feature count of the demonstrated trajectories $\sum_{\tau \in D} Pr(\tau) \sum_{t=1}^{l} \gamma^t \phi_k(s_t, a_t)$ must match the empirical feature count.

3.2.2 Relative entropy of the distribution over trajectories

A different approach to entropy optimization for IRL involves minimizing the relative entropy between two distributions P and Q over the trajectories. More formally,

$$\min_{P \in sd} \sum_{r \in (S \times A)^l} P(\tau) \log \frac{P(\tau)}{Q(\tau)}$$

REIRL [A. Boularias, J. Kober, J. Peters, Relative entropy inverse reinforcement learning] is a prominent technique that utilizes the optimization objective given the equation above. Distribution Q in REIRL is obtained empirically by sampling trajectories under a baseline policy. Distribution P is obtained such that the expected feature count of the trajectories matches the empirical feature count. This constraint is similar to previous approaches in this section and constrains REIRL to the demonstration data. The baseline policy serves as a way to provide domain-specific guidance to the method. While an analytical solution would need the transition dynamics to be pre-specified, Boularias et al. shows that the presence of the baseline policy allows importance sampling to be used and REIRL can be solved model-free using stochastic gradient descent