Imitation Learning Notes

Blake Wulfe
blake.w.wulfe@gmail.com

October 6, 2019

Abstract

Summary notes on imitation learning and inverse reinforcement learning.

1 Introduction

These notes summarize research related to the question, "given access either to an expert or to a set of expert demonstrations within an environment, how can one find a policy for achieving similar performance in that environment?"

1.1 Common Notation

- Markov Decision Process (MDP) containing a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities $p(s'|s,a)$, initial state distribution $p_0(s)$, discount factor γ
- cost of taking action a in state s: $c(s,a) \leq 0$
- stochastic policy: $\pi(a|s)$ giving probability of taking action a in state s
- expert policy: $\pi_E(a|s)$
- the set of possible policies: Π
- expected discounted return of taking action a in state s and then proceeding according to π: $Q^c_\pi(s,a) = \mathbf{E}_{\rho_\pi}[\sum_{t=0}^{\infty} \gamma^t c(s_t,a_t)] = \mathbf{E}_\pi[c(s,a)]$. Note that this is not the expected single-timestep cost, but rather the cumulative cost over the trajectory generated by π. The impact of the transition probability and initial state probability are implicit in the distribution over the trajectories of π.
- state visitation distribution (unnormalized): $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi)$ is the discounted, infinite horizon probability of being in state s.
- state action distribution (unnormalized): $\rho_\pi(s,a) = \rho_\pi(s)\pi(a|s)$ is the infinite horizon, discounted probability of being in state s and taking action a. Can also view this as $\mathbf{E}_{\pi,p_0,p}[\sum_{t=0}^{\infty} 1[s_t = s \Theta a_t = a]]$.
- features of a state action pair: $\phi(s,a)$
- expected features wrt a policy: $\phi(\pi) = \mathbf{E}_{\rho_\pi}[\sum_{i=0}^{\infty} \gamma^t \phi(s_t,a_t)]$
- a trajectory $(s_0, a_0, s_1, a_1, ...)$: τ

1.2 Terminology

- **behavioral cloning**: Learning a policy purely through supervised learning.
- **inverse reinforcement learning**: Learning a cost function most likely to have produced expert behavior, where "most likely" can carry different meanings. Also referred to as "rationalizing" expert behavior.

- **imitation learning**: Ambiguous. In all cases, the goal is to learn a policy from trajectories, but it can refer to behavioral cloning or learning a policy through indirect methods like inverse reinforcement learning
- **apprenticeship learning**: Synonymous with imitation learning.

2 Behavioral Cloning

The simplest approach to learning a policy from expert demonstrations is to fit a model to predict the expert actions using MLE. This can be viewed as substituting a different loss function, l, for the unknown MDP cost/reward function, for example a 0-1 loss in the case of discrete actions. This is choosing

$$\pi = \arg\min_{\pi} E_{s\sim\rho_{\pi_E}}[l(s,\pi)]$$

The problem with this approach is that this loss is computed wrt to the distribution over states encountered by the expert policy rather than that of the learned policy. When acting in the environment, the learned policy may make a small mistake leading to an unknown state, and make mistakes for the remainder of the task horizon T. This is the "cascading errors" problem, which leads to an upper bound on the cost of the policy of $J(\pi) \leq J(\pi^*) + T^2\epsilon$, where ϵ is the probability of the learned classifier making a mistake and costs are assumed to be at most 1 [1].

3 Dataset Aggregation

The problem is fundamentally that supervised learning makes the assumption that an agent's actions do not impact the future state distribution, and as such the training and testing distributions over the states differ. The solution is to make it so that the learner is trained according to the distribution of states it actually encounters. Ross et al. accomplish this in the Dataset Aggregation (DAgger) algorithm by assuming online interaction with an expert [2].

The algorithm starts with a random policy, and then for some number of iterations collects a trajectory using that policy, asks the expert to label the correct action for each state encountered, adds this labeled state-action pair to a dataset, fits the policy to the dataset, and repeats.

4 Inverse Reinforcement Learning

To solve the cascading errors problem without expert interaction requires some manner of evaluating a learned policy on full trajectories. This cannot be accomplished directly using the expert trajectories [1]. IRL assumes the expert is acting optimally within some unknown MDP, and attempts to learn the cost function of that MDP so as to evaluate arbitrary trajectories of the learned policy.

A simple approach to this problem is to assume the cost function can be expressed as a linear combination of features of each state-action pair ($\phi(s,a) = [f_1(s,a),...,f_k(s,a)]$ and $c(s,a) = \theta\phi(s,a)$ and $c \in C_{linear}$) [3]. The IRL objective is then:

$$\delta_C(\pi,\pi_E) = \arg\max_{c\in C} \mathbf{E}_{\rho_\pi}[c(s,a)] - \mathbf{E}_{\rho_{\pi_E}}[c(s,a)]$$

$$= \arg\max_{\theta:||\theta||\leq 1} \mathbf{E}_{\rho_\pi}[\sum_{t=0}^{\infty}\gamma^t\theta\phi(s_t,a_t)] - \mathbf{E}_{\rho_{\pi_E}}[\sum_{t=0}^{\infty}\gamma^t\theta\phi(s_t,a_t)]$$

$$= \arg\max_{\theta:||\theta||\leq 1} \theta(\phi(\pi) - \phi(\pi_E))$$

$$= \frac{\phi(\pi)-\phi(\pi_E)}{||\phi(\pi)-\phi(\pi_E)||}$$

And the RL objective is

$$\min_{\pi} \delta_C(\pi,\pi_E)$$

[1]For example, suppose the policy is for generating sentences (trajectories) one word (action) at a time. Once the learner generates a sequence of words not in the dataset, it's not clear how to evaluate that sequence.

which can be accomplished by matching the feature expectations of the learned policy with those of the expert (since this achieves the minimum of 0 in the simplified IRL objective). The algorithm holds $\tilde{c} = \delta_C(\pi, \pi_E)$ constant and runs any RL algorithm to solve for π, and then recomputes $\tilde{c}$ with π fixed. The reason this works is that once we acquire a policy with equal feature expectations to that of the expert, we know we have obtained a cost function s.t. the optimal policy under that cost is equivalent to the expert.

4.1 Maximum Entropy Inverse Reinforcement Learning

Multiple policies can be optimal for a given $\tilde{c}$. Which should be preferred if this is the case? Ziebart et al. proposed to prefer the policy with maximum entropy [4]. This approach models the expert as sampling trajectories, τ, from the distribution

$$p(\tau; w) = \frac{1}{Z} exp(c_\theta(\tau))$$

where $c_\theta(\tau) = \sum_{s,a \in \tau} c_\theta(s,a)$ and Z is the partition function $\sum_\tau exp(c_\theta(\tau))$. This assigns high probability to trajectories with cost close to 0. Fitting this model to the demonstrated trajectories with MLE produces the cost function most likely to have produced the data. Finding this cost function again requires matching the feature expectations (assuming cost is a linear combination of the features)($\phi(\tau) = \sum_{s,a \in \tau} \phi(s,a)$)

$$\theta^* = \arg\max_\theta \prod_{i=1}^m p(\tau_i; \theta)$$

$$= \sum_{i=1}^m \log(\frac{1}{Z} exp(c_\theta(\tau_i))$$

$$= \sum_{i=1}^m \theta\phi(\tau) - \log(Z)$$

Taking the gradient wrt θ:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^m \theta\phi(\tau) - \log(Z)$$

$$= \frac{1}{m} \sum_{i=1}^m \phi(\tau) - \nabla_\theta \log(Z)$$

$$= \hat{\phi}(\tau) - \frac{1}{Z} \sum_\tau \exp(c_\theta(\tau)) * \phi(\tau)$$

$$= \hat{\phi}(\tau) - \mathbf{E}_\pi[\phi(\tau)]$$

Gradient ascent on the cost function weights therefore tries to match the empirical feature expectations of the expert ($\hat{\phi}(\tau)$) with feature expectations of the learner ($\mathbf{E}_\pi[\phi(\tau)]$). This second term is intractable for large MDPs because it requires computing an expectation over an exponential number of trajectories. Ziebart et al. compute it exactly using a method resembling value iteration, but do so for relatively small MDPs [2]. The algorithm takes the cost function from the previous iteration, computes the feature expectations under the optimal policy using dynamic programming, and then takes a gradient step.

5 Imitation Learning via Policy Optimization

5.1 Model-Free Imitation Learning

Recovering the cost function for large MDPs can be slow because it requires repeatedly running RL or planning to find an optimal policy under the current cost. If you only care about recovering a

[2]see `https://github.com/MatthewJA/Inverse-Reinforcement-Learning/blob/master/irl/maxent.py` for an implementation of the algorithm

good policy and not about the cost function can you save time? Ho, Gupta, and Ermon accomplish this by running policy optimization on local cost functions [5].

The goal is to accomplish the RL objective $\min_\pi \delta_c(\pi, \pi_E)$, but this requires solving for c everywhere. This approach only solves for a local cost function $\hat{c}$ using a set of policy rollouts. Essentially, the previous approaches used a model $(p(s|s', a))$ of the environment and a learned cost function to either arrive at a policy or a state-action visitation frequency. This method instead samples trajectories from the environment, thereby forgoing both the model and the planning loop.

It works as follows:

- Start with some initial policy π_0
- Collect rollouts / trajectories $\tau_1, ..., \tau_B$
- Compute the local cost function $\hat{c}$. For example, the linear cost function case takes the same feature-matching form as that of Abbeel and Ng:

  $$\hat{c} = \frac{\hat{\phi}(\pi) - \hat{\phi}(\pi_E)}{||\hat{\phi}(\pi) - \hat{\phi}(\pi_E)||}$$

  Where $\hat{\phi}(\pi_E)$ is the average features for the expert trajectories, and $\hat{\phi}(\pi)$ are the average features for the sample trajectories of the current policy.
- Compute the policy gradient of the current cost function $\hat{c}$ wrt the policy parameters θ. This is the same update used in REINFORCE [6]:

  $$\nabla_\theta \mathbf{E}_{\rho_{\pi_\theta}} = \mathbf{E}_{\rho_{\pi_\theta}}[\nabla_\theta \log \pi_\theta(a|s) * Q_\pi^{\hat{c}}(s, a)]$$

  Where $Q_\pi^{\hat{c}}(s, a)$ is computed using the discounted sample returns

  $$Q_\pi^{\hat{c}}(s_t, a_t) = \sum_{k=0}^{T-t} \gamma^k \hat{c}(s_k, a_k)$$
- Update the policy via gradient descent

The gradients are high variance even when using a baseline, and so most of the paper deals with a trust region (TRPO) [7] version of the algorithm.

5.2 Generative Adversarial Imitation Learning

Designing a cost function linear in the set of features that is capable of capturing the true cost function can be challenging. It would be nice when dealing with e.g., images to learn a cost function directly from the input, but in this case it's not clear how to learn the cost function because the feature matching approach no longer holds [3].

5.2.1 Generative Adversarial Imitation Learning

The problem is that solving for cost functions nonlinear in the features is challenging. Generative Adversarial Imitation Learning (GAIL) gets around this problem by instead formulating imitation learning as an occupancy measure matching problem that can be solved approximately using GANs [8]. The drawback of this approach is that you don't end up with the cost function, and the advantage is that you can extract a good policy even in large MDPs with nonlinear, unknown cost functions.

The goal is to convert the IRL problem to an occupancy measure matching problem. Starting with the IRL objective with additional regularization term $\psi(c)$:

$$IRL(\pi_E) = \max_c \min_\pi -H(\pi) - \psi(c) + \mathbf{E}_\pi[c(s, a)] - \mathbf{E}_{\pi_E}[c(s, a)]$$

$$= \max_c \min_\pi -H(\pi) - \psi(c) + \sum_{s,a}(\rho - \rho_E)c(s, a)$$

[3]It's not entirely clear to me why more complex cost functions don't work. It might be as simple as the math no longer working out, and there being no obvious way to match the learned features.

When setting $\psi(c)$ to a constant, this is dual to the optimization problem
minimize $_\rho - H(\rho)$ subject to $\rho = \rho_E$

Where the $c(s,a)$ cost values act as Lagrange multipliers enforcing the constraints, which require that the long-term, discounted state-action probabilities of the policy match those of the expert. The switch to ρ from π is justified because there's a one-to-one correspondence between them. Solving this optimization problem exactly is not possible in large MDPs because there are S $*$ A constraints, so we would like to move the constraints into the objective by minimizing some divergence between the two occupancy measures:

$$= \min_\pi -H(\pi) + d_\psi(\rho_\pi, \rho_{\pi_E})$$

$$= \min_\pi -H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E})$$

In the tabular case, the set from which ρ is selected is convex, $-H$ is convex, and the equality constraints are affine, which makes the optimization problem convex. Strong duality holds, and the solution of this problem also gives the solution of the IRL problem. The reason this matters is that this form allows for choosing ψ^* to be Jensen-Shannon divergence, the result of which you can show can be solved approximately using GANs - the generator acts as the policy and is updated using reinforce, where the return in the reinforce update is the expected, discounted sum of rewards, which are the discriminator predictions.

In summary, solving for a cost function nonlinear in the features is difficult. GAIL addresses this issue by avoiding solving for the cost function at all, and instead solves an occupancy measure matching problem.

5.2.2 Information Maximizing Generative Adversarial Imitation Learning

If the expert behavior contains a set of different "styles" for performing a task, how is this manifested in the learned policy? GAIL tends to select a single mode of the behavior to emulate. It may be beneficial for either environmental performance or other reasons to capture the different expert styles.

This is accomplished by infoGAN by introducing a latent code c, and then by encouraging high mutual information between c and the generated output a [9]. In practice, this amounts to learning a supervised classifier that given s predicts c, and then incorporating the performance of this classifier into the GAN objective:

$$\min_{\theta,\psi} \max_w \mathbf{E}_{\pi_\theta}[\log D_w(s,a)] + \mathbf{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda_1 H(\pi_\theta) - \lambda_2 L_I(\pi, Q_\psi)$$

Where

$$L_I(\pi, Q_\psi) = \mathbf{E}_{c\sim p(c), a\sim\pi_\theta(.|s,c)}[\log Q(c|s,a)] + H(c)$$

is a lower bound on the mutual information, $Q(c|s,a)$ gives the probability of the latent code given the state and action, and the entropy of the code $H(c)$ is constant in the randomly-sampled case.

Applying this approach to the imitation learning case allows for disentangling the different expert styles in the data [10] [4].

What if you want to learn the behavior of experts with multiple styles performing multiple tasks? Hausman et al. present a method for accomplishing this, though I have not yet gone through the details of it [11].

6 Guided Cost Learning

This paper takes a sample-based approach to MaxEnt IRL (computing the partition function Z), where they use a policy to effectively perform importance sampling of states.

[4]This paper actually uses image features extracted using a pre-trained CNN for the policy. The discriminator operated directly on the visual input.

7 Unified View

8 Extensions and Questions

References

[1] S. Ross and D. Bagnell, "Efficient reductions for imitation learning", in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.

[2] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning.", in *AISTATS*, vol. 1, 2011, p. 6.

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

[4] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.", 2008.

[5] J. Ho, J. Gupta, and S. Ermon, "Model-free imitation learning with policy optimization", in *International Conference on Machine Learning*, 2016, pp. 2760–2769.

[6] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[7] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization", in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[8] J. Ho and S. Ermon, "Generative adversarial imitation learning", in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets", in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[10] Y. Li, J. Song, and S. Ermon, "Inferring the latent structure of human decision-making from raw visual inputs", *ArXiv preprint arXiv:1703.08840*, 2017.

[11] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim, "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets", *ArXiv preprint arXiv:1705.10479*, 2017.

[12] R. Cogill, "An analysis of primal-dual algorithms for discounted markov decision processes", *ArXiv preprint arXiv:1601.04175*, 2016.

[13] U. Syed, M. Bowling, and R. E. Schapire, "Apprenticeship learning using linear programming", in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1032–1039.

A Generative Adversarial Networks

$$L_{GAN} = \min_G \max_D V(D, G) = \mathbf{E}_{x \sim p_{data}}[\log D(x)] + \mathbf{E}_{z \sim noise}[\log(1 - D(G(z))]$$

B Solving MDPs with Linear Programming

You can solve MDPs by formulating them as LPs, and you can view algorithms like policy iteration as iteratively solving subproblems that can also be formulated as LPs [12].

The primal formulation is typically:

$$\min_V p_0^T V$$

subject to:

$$V(s) \geq \max_a R(s, a) + \gamma \sum_{s'} p(s'|s, a)V(s')$$

Where the constraint is the bellman optimality equation. This is typically the value-centric view you take when solving MPDs, and to get the optimal policy you use a model to determine which action has the highest value for every state. For exactly how this maps to value iteration, see [12], but I think the logic is that you view the final constraint as defining a sequence of subproblems, each of which has a constraint specific to the horizon of that subproblem. Solving each of these given the solution to the prior problem is simple, and is exactly what you do when running value iteration.

The dual LP, which apparently is not as frequently used, is the following: [13]

$$\max_{\rho} \sum_{s} \sum_{a} \rho(s, a) R(s, a)$$

subject to:

$$\rho(s, a) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s, a_t = a) = p_0(s) + \sum_{s'} \rho(s', a) p(s'|s, a)$$

Here we're saying "maximize the occupancy measure for states with high reward, while also ensuring that that occupancy measure satisfies the transition probabilities of the MDP". The constraint is also called the Bellman flow constraint. Solving this problem yields the optimal occupancy measure $\rho^*(s, a)$, and you can easily get the policy by for each state normalizing across actions.

C When is behavioral cloning just as useful as other imitation learning methods?

C.1 Short answer

Let d_π be the state distribution for a policy π, $\hat{\pi}_{sup}$ be the policy learned from supervised learning, and π^* be the optimal policy. If $d_{\hat{\pi}_{sup}} = d_{\pi^*}$, then behavioral cloning can in theory perform just as well as the other methods.

C.2 Long answer

- The goal is to find a good policy for an MDP
- Ideally, we would just solve the MDP, but we don't have access the to reward / cost function $c(s, a)$
- Instead we have access to trajectories from the optimal policy π^*
- So we would instead like to minimize some surrogate loss between the policy being learned and the optimal policy, l, which might be 0-1 loss: $l(\pi, \pi^*, s) = 1\{\pi(s) \neq \pi^*(s)\}$ [5]
- The goal is then to find:

  $$\hat{\pi} = \arg\min_{\pi \in \Pi} \mathbf{E}_{s \sim d_\pi} [l(\pi, \pi^*, s)]$$

  Which is the expected surrogate loss wrt the optimal policy under the distribution of states visited by the agent d_π. Some notes on d_π:

  - $d_\pi^t(s) = p(s_t = s|\pi)$ (this is the probability that the policy π is in state s at time t)
  - $d_\pi(s) = \frac{1}{T} \sum_{t=1}^{T} d_\pi^t(s) = \frac{1}{T} \sum_{t=1}^{T} p(s_t = s|\pi)$ (this is the average state occupancy probability for policy π)
  - This differs from the discounted state occupancy measure, $\rho_\pi(s)$, used earlier in these notes, largely because Ross et al. wanted to give regret bounds [2], which I believe is easier in the average case.

[5] This loss is defined in [2] without π^* as an argument, but I don't understand how the loss could be evaluated without it. It's possible to do this wrt a dataset, but for defining the actual loss π^* seems needed.

- Assuming our goal is to minimize some surrogate loss, then finding $\hat{\pi}$ by solving the above problem is the best we can do; however, we cannot actually compute $l(\pi, \pi^*, s)$ for states not present in the dataset, which means that we cannot solve the problem under the distribution of states visited by the agent $d_{\hat{\pi}}$.
- The supervised / behavioral cloning approach instead solves the problem according to the distribution of states visited by the expert:

$$\hat{\pi}_{sup} = \arg\min_{\pi \in \Pi} \mathbf{E}_{s \sim d_{\pi^*}} [l(\pi, \pi^*, s)]$$

- Ross et al. show that this gives a performance bound of $J(\hat{\pi}_{sup}) \leq J(\pi^*) + \epsilon T^2$, where ϵ is the error probability of the policy wrt l, and give an example where this bound is tight.
- So, the question is, when is the supervised method just as good as the original surrogate loss method?
- A simple answer to this question is that they are equal when $d_{\hat{\pi}} = d_{\pi^*}$
- Under the assumption that there's a 1-to-1 relationship between d_π and π[6], a sufficient condition for this to be the case is $\hat{\pi}_{sup} = \pi^*$.
- This won't be the case unless the dataset contains every state infinitely many times.
- However, if we make additional assumptions about the agent, the MDP, and the state distribution, then it's possible that $d_{\hat{\pi}_{sup}} = d_{\pi^*}$ even if $\hat{\pi}_{sup} \neq \pi^*$. This is the condition for which supervised learning / behavioral cloning will perform equally as well as the original method.
- Three criteria that seem to ensure that this will be the case [7] are (1) The initial state distribution only contains states in the dataset (2) the MDP is deterministic and (3) the supervised policy is deterministic. These criteria mean only states in the dataset will be visited.

[6]There's not a 1-to-1 relationship between d_π and π because the state visitation order doesn't matter, but [13] show that there is a 1-to-1 relationship between the discounted measure ρ_π and π

[7]There might be other criteria