Random Expert Distillation: Imitation Learning via Expert Policy Support Estimation

Ruohan Wang [1] **Carlo Ciliberto** [1] **Pierluigi V. Amadori** [1] **Yiannis Demiris** [1]

Abstract

We consider the problem of imitation learning from a finite set of expert trajectories, without access to reinforcement signals. The classical approach of extracting the expert's reward function via inverse reinforcement learning, followed by reinforcement learning is indirect and may be computationally expensive. Recent generative adversarial methods based on matching the policy distribution between the expert and the agent could be unstable during training. We propose a new framework for imitation learning by estimating the support of the expert policy to compute a fixed reward function, which allows us to re-frame imitation learning within the standard reinforcement learning setting. We demonstrate the efficacy of our reward function on both discrete and continuous domains, achieving comparable or better performance than the state of the art under different reinforcement learning algorithms.

1. Introduction

We consider a specific setting of imitation learning - the task of policy learning from expert demonstrations - in which the learner only has a finite number of expert trajectories without any further access to the expert. Two broad categories of approaches to this setting are behavioral cloning (BC) (Pomerleau, 1991), which directly learns a policy mapping from states to actions with supervised learning from expert trajectories; and inverse reinforcement learning (IRL) (Abbeel & Ng, 2004; Ng & Russell, 2000), which learns a policy via reinforcement learning, using a cost function extracted from expert trajectories.

Most notably, BC has been successfully applied to the task of autonomous driving (Bansal et al., 2018; Bojarski et al., 2016). Despite its simplicity, BC typically requires a large amount of training data to learn good policies, as it may suffer from compounding errors caused by covariate shift (Ross & Bagnell, 2010; Ross et al., 2011). BC is often used as a policy initialization step for further reinforcement learning (Nagabandi et al., 2018; Rajeswaran et al., 2017).

IRL estimates a cost function from expert trajectories and uses reinforcement learning to derive policies. As the cost function evaluates the quality of trajectories rather than that of individual actions, IRL avoids the problem of compounding errors. IRL is effective with a wide range of problems, from continuous control benchmarks in the Mujoco environment (Ho & Ermon, 2016), to robot footsteps planning (Ziebart et al., 2008).

Generative Adversarial Imitation Learning (GAIL) (Baram et al., 2017; Ho & Ermon, 2016) connects IRL to the general framework of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), and re-frames imitation learning as distribution matching between the expert policy and the learned policy via Jensen-Shannon Divergence. However, GAIL naturally inherits the challenges of GAN training, including possible training instability, and overfitting to training data (Arjovsky & Bottou, 2017; Brock et al., 2018). Similarly, generative moment matching imitation learning (GMMIL) considers distribution matching via maximum mean discrepancy. Both GAIL and GMMIL iteratively update reward function and learned policy during training.

In this paper, we propose imitation learning via expert policy support estimation, a new general framework for recovering a reward function from expert trajectories. We propose Random Expert Distillation (RED) by providing a connection between Random Network Distillation (RND) (Burda et al., 2018) – a method to design intrinsic rewards for RL exploration based on the "novelty" of states visited – and support estimation ideas. Our method computes a fixed reward function from expert trajectories, and obviates the need for dynamic update of reward functions found in classical IRL, GAIL and GMMIL. Evaluating RED using different reinforcement learning algorithms on both discrete and continuous domains, we show that the proposed method achieves comparable or better performance than the state-of-arts methods on a variety of tasks, including a driving

[1]Department of Electrical and Electronic Engineering, Imperial College London, UK;. Correspondence to: Ruohan Wang <r.wang16@imperial.ac.uk>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

scenario (see Figure 4). To the best of our knowledge, our method is the first to explore expert policy support estimation for imitation learning.

2. Background

We review the formulation of Markov Decision Process (MDP) in the context of which we consider imitation learning. We also review previous approaches to imitation learning and support estimation related to our proposed method.

Setting. We consider an infinite-horizon discounted MDP, defined by the tuple $(S, A, P, r, p_0, \gamma)$, where S is the set of states, A the set of actions, $P : S \times A \times S \to [0, 1]$ the transition probability, $r : S \times A \to \mathbb{R}$ the reward function, $p_0 : S \to [0, 1]$ the distribution over initial states, and $\gamma \in (0, 1)$ the discount factor. Let π be a stochastic policy $\pi : S \times A \to [0, 1]$ with expected discounted reward $\mathbb{E}_\pi(r(s, a)) \triangleq \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t))$ where $s_0 \sim p_0$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$ for $t \geq 0$. We denote π_E the expert policy.

2.1. Imitation Learning

We briefly review the main methods for imitation learning:

Behavioral Cloning (BC) learns a control policy $\pi : S \to A$ directly from expert trajectories via supervised learning. Despite its simplicity, BC is prone to compounding errors: small mistakes in the policy cause the agent to deviate from the state distribution seen during training, making future mistakes more likely. Mistakes accumulate and lead to eventual catastrophic errors (Ross et al., 2011). While several strategies have been proposed to address the issue (Ross & Bagnell, 2010; Sun et al., 2017), they often require access to the expert policy during the entire training process, rather than a finite set of expert trajectories. BC is commonly used to initialize control policies for reinforcement learning (Nagabandi et al., 2018; Rajeswaran et al., 2017).

Inverse Reinforcement Learning (IRL) models the expert trajectories with a Boltzmann distribution (Ziebart et al., 2008), where the likelihood of a trajectory is defined as:

$$p_\theta(\tau) = \frac{1}{Z} \exp(-c_\theta(\tau)) \tag{1}$$

where $\tau = \{s_1, a_1, s_2, a_2 ...\}$ is a trajectory, $c_\theta(\tau) = \sum_t c_\theta(s_t, a_t)$ a learned cost function parametrized by θ, and the partition function $Z \triangleq \int \exp(-c_\theta(\tau)) d(\tau)$ the integral over all trajectories consistent with transition function of the MDP. The main computational challenge of IRL is the efficient estimation of the partition function Z. IRL algorithms typically optimize the cost function by alternating between updating the cost function and learning an optimal policy with respect to the current cost function with reinforcement learning (Abbeel & Ng, 2004; Ng & Russell, 2000).

Imitation Learning via Distribution Matching frames imitation learning as distribution matching between the distribution of state-action pairs of the expert policy and that of the learned policy. In (Finn et al., 2016; Ho & Ermon, 2016), the authors connect IRL to distribution matching via Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). Known as Generative Adversarial Imitation Learning (GAIL), the method imitates an expert policy by formulating the following minimax game:

$$\min_{\pi} \max_{D \in (0,1)} \mathbb{E}_\pi(\log D(s, a)) + \mathbb{E}_{\pi_E}(\log(1 - D(s, a))) - \lambda H(\pi) \tag{2}$$

where the expectations $\mathbb{E}_\pi$ and $\mathbb{E}_{\pi_E}$ denote the joint distributions over state-action pairs of the learned policy and the expert policy, respectively, and $H(\pi) \triangleq \mathbb{E}_\pi(-\log \pi(a|s))$ is the entropy of the learned policy. GAIL has been successfully applied to various control tasks in the Mujoco environment (Baram et al., 2017; Ho & Ermon, 2016). However, GAIL inherits the challenges of GANs, including possible training instability such as vanishing or exploding gradients, as well as overfitting to training data (Arjovsky & Bottou, 2017; Brock et al., 2018). While numerous theoretical and practical techniques have been proposed to improve GANs (e.g (Arjovsky et al., 2017; Salimans et al., 2016)), a large-scale study of GANs show that many GAN algorithms and architectures achieve similar performance with sufficient hyperparameter tuning and random restarts, and no algorithm or network architecture stands out as the clear winner on all tasks (Lucic et al., 2018).

Similar to GAIL, (Kim & Park, 2018) proposed generative moment matching imitation learning (GMMIL) by minimizing the maximum mean discrepancy between the expert policy and the learned policy. Though GMMIL avoids the difficult minimax game, the cost of each reward function evaluation grows linearly with the amount of training data, which makes scaling the method to large dataset potentially difficult. In addition, we demonstrate in our experiments that GMMIL may fail to estimate the appropriate reward functions.

2.2. Support Estimation with Kernel Methods

As we will motivate in detail in Sec. 3, we argue that estimating the support of the expert policy can lead to good reward functions for imitation learning. In this section we review one of the most well-established approaches to support estimation of a distribution from a finite number of i.i.d. samples, which relies on a kernelized version of principal component analysis (Schölkopf et al., 1998). The idea is to leverage the *Separating Property* (De Vito et al., 2014) of suitable reproducing kernel Hilbert spaces, which guarantees the covariance operator of the embedded data to precisely capture the geometrical properties of the support of the underlying distribution. This allows us to derive a test function that is zero exclusively on points belonging to the support of the distribution.

Formally, let $\mathcal{X} \subseteq \mathbb{R}^d$ be a set (in our setting $\mathcal{X} = S \times A$ is the joint state-action space) and let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel with associated reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ (Aronszajn, 1950) and feature map $\phi : \mathcal{X} \to \mathcal{H}$, such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ for any $x, x' \in \mathcal{X}$. For any set $U \subseteq \mathcal{X}$, denote by $\Phi(U) = \{\phi(x) \mid x \in U\}$ and $\overline{\Phi(U)}$ the closure of its span in $\mathcal{H}$. The separating property guarantees that for any closed subset U of $\mathcal{X}$, $\Phi(U) = \Phi(\mathcal{X}) \cap \overline{\Phi(U)}$. In other words, the separating property ensures that

$$x \in S \iff \phi(x) \in \overline{\Phi(U)}. \tag{3}$$

As shown in (De Vito et al., 2014), several kernels allow the separating property to hold, including the popular Gaussian kernel $k(x, x') = \exp(-\|x - x'\| / \sigma)$ for any $\sigma > 0$.

The separating property suggests a natural strategy to test whether a point $x \in \mathcal{X}$ belongs to a closed set U. Let $P_U : \mathcal{H} \to \mathcal{H}$ be the orthogonal projection operator onto $\overline{\Phi(U)}$ (which is a linear operator since $\overline{\Phi(U)}$ is a linear space), we have

$$x \in U \iff \|(I - P_U)\phi(x)\|_{\mathcal{H}} = 0,$$

since Eq. (3) corresponds to $\phi(x) = P_U \phi(x)$, or equivalently $(I - P_U)\phi(x) = 0$.

We can leverage the machinery introduced above to estimate the support supp$(\pi) \subseteq \mathcal{X}$ of a probability distribution π on $\mathcal{X}$. We observe that the covariance operator $C_\pi = \mathbb{E}_\pi[\phi(x)\phi(x)^\top]$ encodes sufficient information to recover the orthogonal projector $P_{\text{supp}(\pi)}$ (denoted P_π in the following for simplicity). More precisely, let $C_\pi^\dagger$ denote the pseudoinverse of C_π. A direct consequence of the separating property is that $P_\pi = C_\pi^\dagger C_\pi$ (De Vito et al., 2014).

When π is unknown and observed only through a finite number of N i.i.d examples $\{x_i\}_{i=1}^N$, it is impossible to obtain C_π (and thus P_π) exactly. A natural strategy is to consider the *empirical covariance* operator $\hat{C} = \frac{1}{N}\sum_{i=1}^N \phi(x_i)\phi(x_i)^\top$ as a proxy of the ideal one. Then, the projector is estimated as $\hat{P} = \hat{C}_m^\dagger \hat{C}_m$, where C_m is the operator comprising the $m \leq n$ principal component directions of C, where m is a hyperparameter to control the stability of the pseudoinverse when computing $\hat{P}$. We can then test whether a point $x \in \mathcal{X}$ belongs to supp(π), by determining if

$$\begin{aligned} \left\| (I - \hat{P})\phi(x) \right\|_{\mathcal{H}}^2 &= \left\langle \phi(x), (I - \hat{P})\phi(x) \right\rangle_{\mathcal{H}} \\ &= k(x, x) - \left\langle \phi(x), \hat{P}\phi(x) \right\rangle_{\mathcal{H}}, \end{aligned} \tag{4}$$

is greater than zero (in practice a threshold $\tau > 0$ is introduced). Note that we have used the fact that $\hat{P}^\top \hat{P} = \hat{P}$, since $\hat{P}$ is an orthogonal projector. Although $\hat{C}$ and $\hat{P}$ are operators between possibly infinite dimensional spaces, we have for any $x \in \mathcal{X}$ (see Schölkopf et al., 1998)

$$\left\langle \phi(x), \hat{P}\phi(x) \right\rangle_{\mathcal{H}} = K_x^\top K_m^\dagger K_x,$$

where $K \in \mathbb{R}^{n \times n}$ is the empirical kernel matrix of the training examples, with entries $K_{ij} = k(x_i, x_j)$. Here, K_m denotes the matrix obtained by performing PCA on K and taking the first $m \leq n$ principal directions and $K_x \in \mathbb{R}^n$ is the vector with entries $(K_x)_i = k(x_i, x)$. Therefore $\langle \phi(x), \hat{P}\phi(x) \rangle_{\mathcal{H}}$ can be computed efficiently in practice.

The theoretical properties of the strategy described above have been thoroughly investigated in the previous literature (De Vito et al., 2014; Rudi et al., 2017). In particular, it has been shown that the Housedorff distance between supp(π) and the set induced by $\hat{P}$, tends to zero when $n \to +\infty$, provided that the number of principal components m grows with the number of training examples. Moreover, it has been shown that the support estimator enjoys fast learning rate under standard regularity assumptions.

3. Imitation Learning via Expert Policy Support Estimation

Formally, we are interested in extracting a reward function $\hat{r}(s, a)$ from a finite set of trajectories $D = \{\tau_i\}_{i=1}^N$ produced by an expert policy π_E within a MDP environment. Here each τ_i is a trajectory of state-action pairs of the form $\tau_i = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$. Assuming that the expert trajectories are consistent with some unknown reward function $r^*(s, a)$, our goal is for a policy learned by applying RL to $\hat{r}(s, a)$, to achieve good performance when evaluated against $r^*(s, a)$ (the standard evaluation strategy for imitation learning). In contrast with traditional imitation learning, we do not explicitly aim to match π_E exactly.

We motivate our approach with a thought experiment. Given a discrete action space and a deterministic expert such that $a_s^* = \pi_E(s)$, the resulting policy is a Dirac's delta at each state $s \in S$, for all (s, a). Consider the reward function

$$r_E(s, a) = \begin{cases} 1 \text{ if } (s, a) \in \text{supp}(\pi_E) \\ 0 \text{ if } (s, a) \notin \text{supp}(\pi_E) \end{cases} \tag{5}$$

which corresponds to the indicator function of supp(π_E). It follows that a RL agent trained with this reward function would be able to imitate the expert exactly, since the discrete action space allows the random exploration of the agent to discover optimal actions at each state. In practice, the expert policy is unknown and only a finite number of trajectories sampled according to π_E are available. In these context we can use support estimation techniques to construct a reward function $\hat{r}$.

For continuous action domains, sparse reward such as r_E is unlikely to work since it is improbable for the agent, via random exploration, to discover the optimal actions, while all other actions are considered equally bad. Instead, since support estimation produces a score with Eq. (4) for testing each input, we may directly use that score to construct the reward function $\hat{r}$. The rationale is that actions similar to that of the expert in any given state would still receive high scores from support estimation, allowing the RL agent to discover those actions during exploration.

Based on the motivation above, we hypothesize that support estimation of the expert policy's state-action distribution provides a viable reward function for imitation learning. Intuitively, the reward function encourages the RL agent to behave similarly as the expert at each state and remain on the estimated support of the expert policy. Further, this allows us to compute a fixed reward function based on the expert trajectories and re-frame imitation learning in the standard context of reinforcement learning.

We note that for stochastic experts, the support of π_E might coincide with the whole space $S \times A$. Support estimation would hence produce an uninformative, flat reward function r_E when given an infinite amount of training data. However, we argue that an infinite amount of training data should allow BC to successfully imitate the expert, bypassing the intermediate step of extracting a reward function from the expert trajectories.

3.1. Practical Support Estimation

Following the strategy introduced in Sec. 2.2, we consider a novel approach to support estimation. Our method takes inspiration from kernel methods but applies to other models such as neural networks.

Let $\mathcal{H}$ be a RKHS and $f \in \mathcal{H}$ a function parametrized by $\theta \in \Theta$. Consider the regression problem that admits a minimizer on $\mathcal{H}$

$$\inf_{f \in \mathcal{H}} \int (f_\theta(x) - f(x))^2 \, d\pi(x),$$

The minimal $\|\cdot\|_{\mathcal{H}}$ solution corresponds to $f_{\pi,\theta} = C_\pi^\dagger C_\pi f_\theta = P_\pi f_\theta$, the orthogonal projection of f_θ onto the range of C_π (see e.g Engl et al., 1996) and let $P_\pi^\perp = (I - P_\pi)$. For any $x \in \mathcal{X}$ we have

$$f_\theta(x) - f_{\pi,\theta}(x) = \left\langle f_\theta, P_\pi^\perp \phi(x) \right\rangle_{\mathcal{H}}.$$

In particular, if $x \in \operatorname{supp}(\pi)$, we have $P_\pi^\perp \phi(x) = (I - P_\pi)\phi(x) = 0$ and hence $f_\theta(x) - f_{\pi,\theta}(x) = 0$. The converse is unfortunately not necessarily true: for a point $x \notin \operatorname{supp}(\pi)$, $P_\pi^\perp \phi(x)$ may still be orthogonal to f_θ and thus $f_\theta(x) - f_{\pi,\theta}(x) = 0$.

A strategy to circumvent this issue is to consider multiple f_θ, and then average their squared errors $(f_\theta - f_{\pi,\theta})^2$. The rationale is that if $x \notin S_\pi$, by spanning different directions in $\mathcal{H}$, at least one of the f_θ will not be orthogonal to $P_\pi^\perp \phi(x)$, which leads to a non-zero value when compared against $f_{\pi,\theta}(x)$. In particular, if we sample θ according to a probability distribution over Θ, we have

$$\begin{aligned} \mathbb{E}_\theta (f_\theta - f_{\pi,\theta})^2 &= \mathbb{E}_\theta \left\langle P_\pi^\perp \phi(x), f_\theta f_\theta^\top \, P_\pi^\perp \phi(x) \right\rangle_{\mathcal{H}} \\ &= \left\langle P_\pi^\perp \phi(x), \Sigma \, P_\pi^\perp \phi(x) \right\rangle_{\mathcal{H}}, \end{aligned}$$

where $\Sigma = \mathbb{E}_\theta[f_\theta f_\theta^\top]$ is the covariance of the random functions f_θ generated from sampling the parameters θ. Ideally, we would like to sample θ such that $\Sigma = I$, which allows us to recover the support estimator $\left\| P_\pi^\perp \phi(x) \right\|_{\mathcal{H}}$ introduced in Sec. 2.2. However, it is not always clear how to sample f_θ so that its covariance corresponds to the identity in practice. In this sense, this approach does not offer the same theoretical guarantees as the kernelized method reviewed in Sec. 2.2.

The discussion above provides us with a general strategy for support estimation. Consider a hypotheses space of functions f_θ parametrized by $\theta \in \Theta$ (e.g. neural networks). We can sample $\theta_1, \ldots, \theta_K$ functions according to a distribution over Θ. Given a training dataset $\{x_i\}_{i=1}^N$ sampled from π, we solve the K regression problems

$$\hat{\theta}_k = \min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N (f_\theta(x_i) - f_{\theta_k}(x_i))^2, \tag{6}$$

for every $k = 1, \ldots, K$. Then for any $x \in \mathcal{X}$, we can test whether it belongs to the support of π by checking whether

$$\frac{1}{K} \sum_{k=1}^K (f_{\hat{\theta}_k}(x) - f_{\theta_k}(x))^2, \tag{7}$$

is larger than a prescribed threshold. As K and N grow larger, the estimate above will approximate increasingly well the desired quantity $\mathbb{E}_\theta (f_{\pi,\theta}(x) - f_\theta(x))^2$.

3.2. Reward by Random Network Distillation

We may interpret the recent method of Random Network Distillation (RND) (Burda et al., 2018) as approximate support estimation. Formally, RND sets up a randomly generated prediction problem involving two neural networks: a fixed and randomly initialized target network f_θ which sets the prediction problem, and a predictor network $f_{\hat{\theta}}$ trained on the state-action pairs of the expert trajectories. f_θ takes an observation to an embedding $f_\theta : S \times A \to \mathbb{R}^K$ and similarly so for $f_{\hat{\theta}} : S \times A \to \mathbb{R}^K$, which is analogous to the K prediction problems defined in Eq. (6). The predictor network is trained to minimize the expected mean square error by gradient descent $L(s,a) = ||f_{\hat{\theta}}(s,a) - f_\theta(s,a)||_2^2$

with respect to its parameters $\hat{\theta}$. After training, the score of a state-action pair belonging to the support of π_E is predicted by $L(s,a)$, analogous to Eq. (7). One concern to RND is that a sufficiently powerful optimization algorithm may recover f_θ perfectly, causing $L(s,a)$ to be zero everywhere. Our experiments confirm the observations in (Burda et al., 2018) that standard gradient-based methods do not behave in this undesirable way.

With the prediction error $L(s,a)$ as the estimated score of (s,a) belonging to the support of π_E , we propose a reward function that has worked well in practice,

$$r(s,a) = \exp(-\sigma_1 L(s,a)), \tag{8}$$

where σ_1 is a hyperparameter. As $L(s,a)$ is positive, $r(s,a) \in [0,1]$. We choose σ_1 such that $r(s,a)$ from expert demonstrations are mostly close to 1.

3.3. Terminal Reward Heuristic

For certain tasks, there are highly undesirable terminal states (e.g. car crashes in autonomous driving). In such contexts, we further introduce a terminal reward heuristic to penalize the RL agent from ending an episode far from the estimated support of the expert policy. Specifically, let $\bar{r} = -\frac{1}{N}\sum_{i=1}^{N} r(s_i,a_i)$ be the average reward computed using expert trajectories, and $r(s_T,a_T)$ the reward of the final state of an episode, we define

$$r_{term} = \begin{cases} -\sigma_2 \bar{r} & \text{if } \sigma_3 \bar{r} > r(s_T,a_T) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where σ_2, σ_3 are hyperparameters. We apply the heuristic to an autonomous driving task in the experiment, and demonstrate that it corresponds nicely with dangerous driving situations and encourages RL agents to avoid them. We note that the heuristic is not needed for all other tasks considered in the experiments.

3.4. Random Expert Distillation

We present our algorithm Random Expert Distillation (RED) in Algorithm 1. The algorithm computes the estimated support of the expert policy and generates a fixed reward function according to Eq. (8). The reward function is used in RL for imitation learning. As we report in the experiments, a random initial policy π_0 is sufficient for the majority of tasks considered in the experiments, while the remaining tasks requires initialization via BC. We will further discuss this limitation of our method in the results.

3.5. Alternative to RND

AutoEncoder (AE) networks (Vincent et al., 2008) set up a prediction problem of $\min_\theta ||f_\theta(x) - x||_2^2$. AE behaves similarly to RND in the sense that it also yields low prediction errors for data similar to the training set. AE could be also seen as approximate support estimation in the context of Eq. (7), by replacing randomly generated f_{θ_k} with identity functions on each dimension of the input. Specifically, AE sets up K prediction problems

$$\min_\theta \frac{1}{N}\sum_{i=1}^{N} (f_\theta(x_i) - I_k(x_i))^2$$

for $k = 1, \ldots, K$ where $I_k(x) = x_k$ and K is the input size. However, as discussed in (Burda et al., 2018), AE with a bottleneck layer may suffer from input stochasticity or model misspecification, which are two sources of prediction errors undesirable for estimating the similarity of input to the training data. Instead, we have found empirically that overparametrized AEs with a ℓ_2 regularization term prevent the trivial solution of an identity function, allow easier fitting of training data, and may be used in place of RND. We include AE in our experiments to demonstrate that support estimation in general appears to be a viable strategy for imitation learning.

Algorithm 1 RANDOM EXPERT DISTILLATION

```
Input: data $D = \{(s_i,a_i)\}_{i=1}^{N}$, $\Theta$ function models, initial policy $\pi_0$.

Randomly sample $\theta \in \Theta$
$\hat{\theta}$ = MINIMIZE($f_{\hat{\theta}}, f_\theta, D$)
$r(\cdot) = \exp(-\sigma_1 \|f_{\hat{\theta}}(\cdot) - f_\theta(\cdot)\|_2^2)$
Compute $r_{\text{term}}$ from Eq. (9)
$\pi$ = REINFORCEMENTLEARNING($r, r_{\text{term}}, \pi_0$).

Return: $\pi$.
```

4. Experiments

We evaluate the proposed method on multiple domains. We present a toy problem to motivate the use of expert policy support estimation for imitation learning and to highlight the behaviors of different imitation learning algorithms. We then evaluate the proposed reward function on five continuous control tasks from the Mujoco environment. Lastly, we test on an autonomous driving task with a single trajectory provided by a human driver. The code for reproducing the experiments is available online.[1]

4.1. Simple Domain

We consider a stateless task where $s \sim \text{unif}(-1,1)$, a discrete action space $a \in \{-1,1\}$ and the reward function $r(s,a) = as$. It is clear that the expert policy to this problem is $\pi_E(s) = \text{sign}(s)$. Using Deep Q Learning (Mnih et al.,

[1] https://github.com/RuohanW/RED

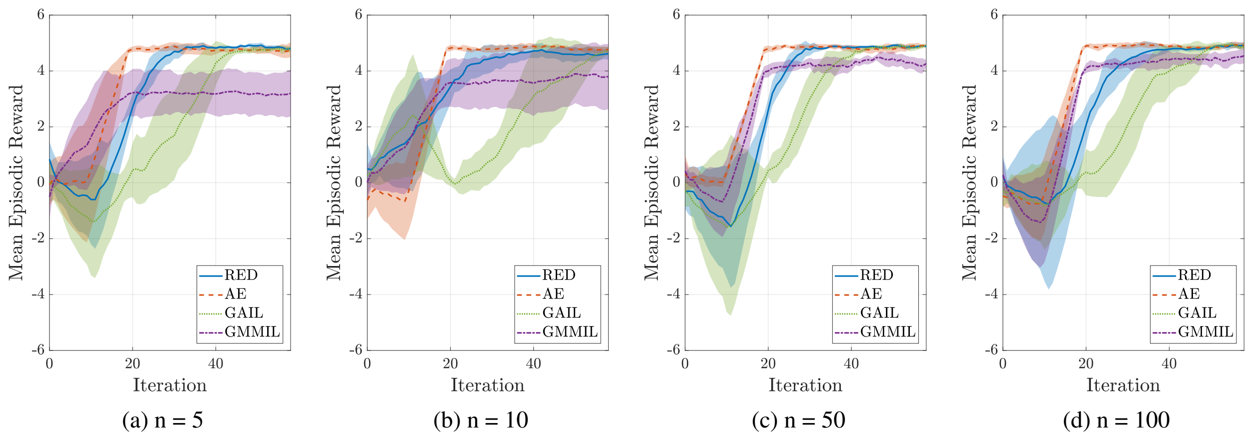


Figure 1. True mean episodic reward during training on the simple domain, with different expert dataset sizes

	HOPPER	HALFCHEETAH	REACHER	WALKER2D	ANT
GAIL	3614.22 ± 7.17	4515.70 ± 549.49	-32.37 ± 39.81	4877.98 ± 2848.37	3186.80 ± 903.57
GMMIL	3309.30 ± 26.28	3464.18 ± 476.50	-11.89 ± 5.27	2967.10 ± 702.03	-
AE	3478.31 ± 3.09	3380.74 ± 101.94	-10.91 ± 5.62	4097.61 ± 118.06	3778.61 ± 422.63
RED	3625.96 ± 4.32	3072.04 ± 84.71	-10.43 ± 5.20	4481.37 ± 20.94	3552.77 ± 348.67

Table 1. Episodic reward (as provided by the environment) on Mujoco tasks by different methods evaluated over 50 runs. HalfCheetah and Ant uses BC initialization in RED and AE. We are unsuccessful with GMMIL on Ant.

2015) as the reinforcement learning algorithm, we compare the proposed method against AE, GAIL and GMMIL with an expert dataset of size $n = 5, 10, 50, 100$ respectively.

In Figure 1, we show the true mean episodic reward of different algorithms during training. Except for GMMIL, all other algorithms converge to the expert policy for all sizes of the dataset. In addition, RED and AE converge to the expert policy faster as the extracted reward functions recover the correct reward function nearly perfectly (Figures 2f, 2c). In contrast, GAIL requires adversarial training to recover the correct reward function, a noisy process during which all actions from the learned policy, both optimal or not, are considered "wrong" by the discriminator. In fact, when the discriminator is overparametrized, it would overfit to the training data gradually and generate arbitrary reward for some region of the state-action space (Figure 2e). The results are consistent with observations from (Brock et al., 2018) that the discriminator is overfitting, and that early stopping may be necessary to prevent training collapse. For GMMIL, we observe that the method intermittently generates wrong reward functions (Figure 2d), causing the performance to oscillate and converges to lower mean reward, especially when the expert data is sparse. This is likely due to the noise in estimating distribution moments from limited samples.

4.2. Mujoco Tasks

We further evaluate RED on five continuous control tasks from the Mujoco environment: Hopper, Reacher and HalfCheetah, Walker2d and Ant. Similarly, we compare RED against AE, GAIL and GMMIL, using Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) in Table 1. Similar to the experiments in (Ho & Ermon, 2016), we consider learning with 4 trajectories of expert demonstration generated by an expert policy trained with RL. All RL algorithms terminate within 5M environment steps. We note that we were unsuccessful in the Ant task with GMMIL after an extensive search for the appropriate hyperparameters.

The results suggest that support estimation of expert policies is a viable strategy for imitation learning, as the AE and RED are both able to achieve good results with the fixed reward functions constructed from support estimation of the expert policy. While RED and AE underperform on the HalfCheetah, both methods are comparable or better than GAIL and GMMIL in all other tasks. In particular, RED and AE achieve much smaller variance in performance, likely due to the fixed reward function. Consistent with the observations from the simple domain in Sec. 4.1, RED appears to achieve faster performance improvements during early training (Figure 3). We note that though the best performance AE can achieve is similar to RED, AE is much

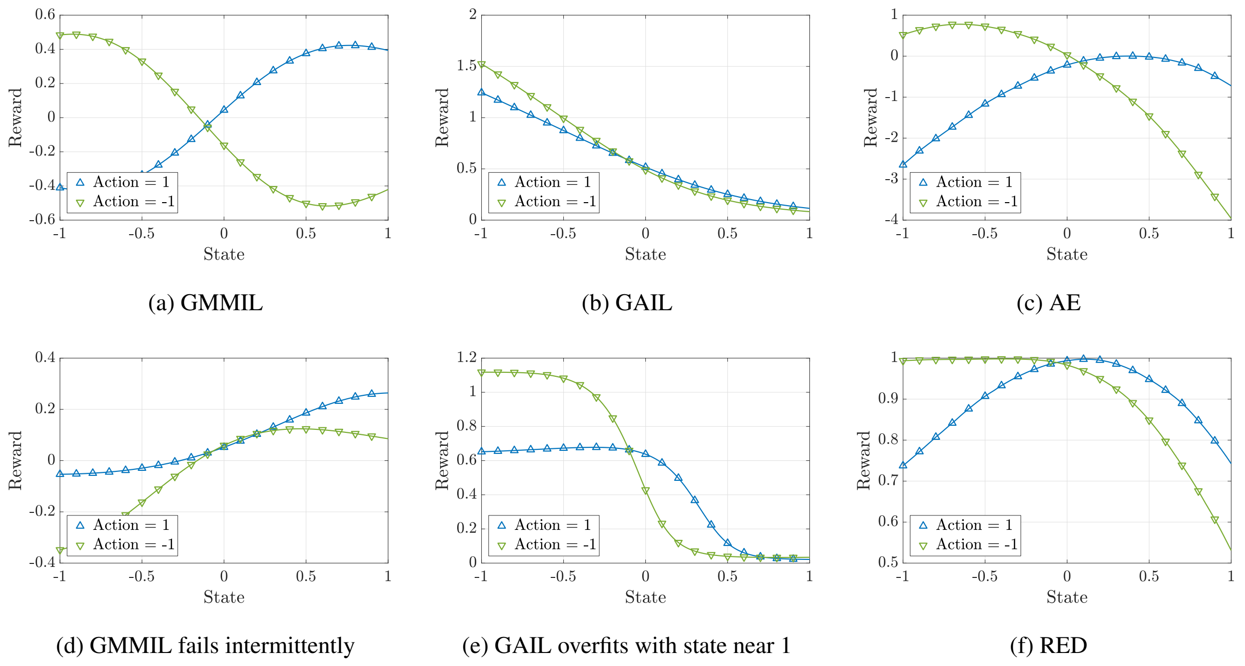


Figure 2. Estimated reward functions of the expert using different imitation learning algorithms on the simple domain. For better visualization of the reward, we use $r(s,a) = 1 - \alpha_1 L(s,a)$ as the reward function for AE and RED.

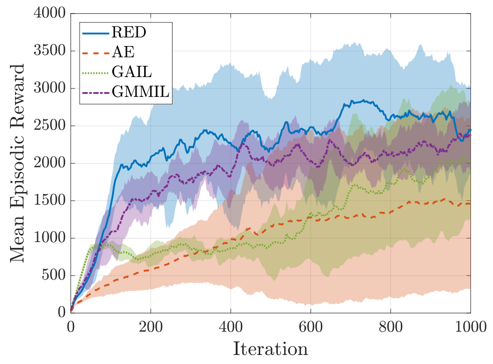


Figure 3. True mean episodic reward of GAIL, AE, GMMIL and RED on Hopper during training. Agents trained with RED improve faster compared to other methods.

more sensitive to the random initialization of parameters, seen from the large standard deviation in Figure 3.

A limitation of our method is that HalfCheetah and Ant require a policy initialized with BC to achieve good results while GAIL and GMMIL could start with a random policy in the two tasks. We hypothesize that the evolving reward functions of GAIL and GMMIL may provide better exploration incentives to the RL agent. As GAIL is orthogonal to our method and may be used together, we leave it to future work on how to combine the benefits of both RED and GAIL to achieve more robust algorithms.

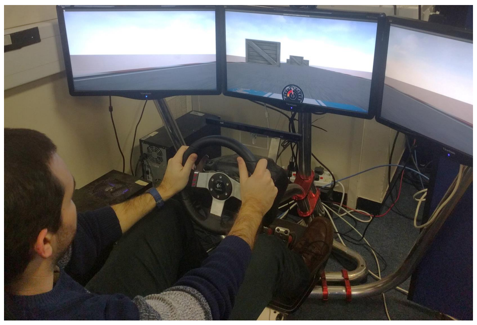
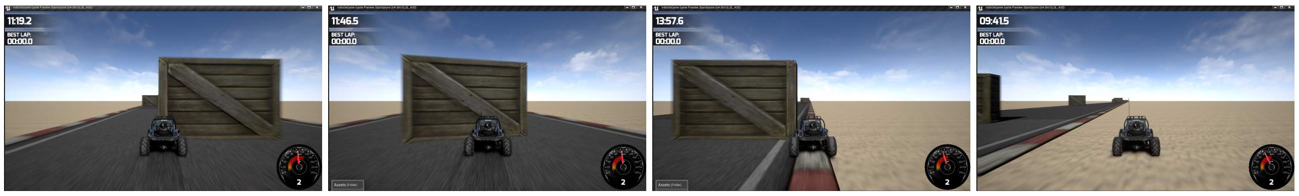


Figure 4. The driving scenario consists of an expert driving through a straight track while avoiding obstacles in simulation.

4.3. Autonomous Driving Task

Lastly, we evaluate RED, AE, GAIL, GMMIL and BC on an autonomous driving task, using a single demonstration provided by a human driver. The environment consists of a straight track with obstacles placed randomly at one of the three lanes of the track (Figure 4). A human demonstrator was instructed to drive through the track and avoid any obstacles, while keeping the speed around 100 km/h. We sampled the expert driving actions at 20 Hz. For the environment, we use a vector of size 24 to represent state (20 dimensions for the LIDAR reading, 3 dimensions for the relative position, and orientation of the track with respect to the car, and 1 dimension for the vehicle speed) to output



(a) Near Collision (b) Collision (c) Off-road Collision (d) Off-road

Figure 5. Representative scenarios where the reward functions of AE and RED assign near zero rewards. The scenarios correspond well with various dangerous states as they are dissimilar to those from expert demonstrations.

Table 2. Episodic reward (distance travelled without collision) on the driving task with different methods. The expert performance of 7485 corresponds with track completion without collision. Both AE and RED use the terminal reward heuristic.

	AVERAGE	STD	BEST
GAIL	795	395	1576
GMMIL	2024	981	3624
BC	1033	474	1956
AE	4378	1726	7485
RED	4825	1552	7485
EXPERT	7485	0	7485

the steering and the accelerator command for the vehicle. We include the terminal reward heuristic defined in Eq. 9. For evaluation, we measure the distance a learned policy is able to control the vehicle through the track without any collision with the obstacles. The task is challenging as the learner must generalize from the limited amount of training data, without explicit knowledge about the track width or the presence of obstacles. The driving task also allows us to qualitatively observe the behaviors of learned policies.

In this task, we initialize all policies with BC and use stochastic value gradient method with experience replay (Heess et al., 2015) as the reinforcement learning algorithm. The algorithm is referred to as SVG(1)-ER in the original paper.

Table 2 shows the average and best performance of each method evaluated over 5 runs through the same track. We note that the expert performance of 7485 corresponds with track completion without collision. The results suggest that RED achieves the best performance. For both RED and AE, the reward functions correctly identify dangerous situations, such as collision or near-collision, by assigning those states with zero rewards. Figure 5 shows a few representative states where the reward function outputs zero reward. In contrast, we observe that GAIL tends to overfit to the expert trajectory. During training, GAIL often ”forgets” how to avoid the same obstacle after the discriminator update the reward function. Such behaviors prevent the learned policy from avoiding obstacles consistently. For GMMIL, we observe behaviors similar to that found in Sec. 4.1: the reward function intermittently produces problematic incentives, such as assigning positive rewards for states immediately preceding collisions.

5. Conclusion

We propose a new general framework of imitation learning via expert policy support estimation. We connect techniques such as Random Network Distillation and AutoEncoders to approximate support estimation; and introduce a method for efficiently learning a reward function suitable for imitation learning from the expert demonstrations. We have shown empirically in multiple tasks that support estimation of expert policies is a viable strategy for imitation learning, and achieves comparable or better performance compared to the state-of-art. For future works, combining different approaches of recovering the expert’s reward function appears to be a promising direction.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by UK DSTL/EPSRC Grant EP/P008461/1, a Singapore National Science Scholarship, and a Royal Academy of Engineering Chair in Emerging Technologies to Yiannis Demiris.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, pp. 1, 2004.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. *International Conference on Learning Representation*, pp. 1, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Aronszajn, N. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Bansal, M., Krizhevsky, A., and Ogale, A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

Baram, N., Anschel, O., Caspi, I., and Mannor, S. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pp. 390–399, 2017.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

De Vito, E., Rosasco, L., and Toigo, A. A universally consistent spectral estimator for the support of a distribution. *Appl Comput Harmonic Anal*, 37:185–217, 2014.

Engl, H. W., Hanke, M., and Neubauer, A. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

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

Kim, K.-E. and Park, H. S. Imitation learning via kernel mean embedding. *AAAI*, 2018.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are gans created equal? A large-scale study. In *Advances in Neural Information Processing Systems*, pp. 698–707, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *International Conference on Robotics and Automation*, pp. 7559–7566, 2018.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 663–670, 2000.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 661–668, 2010.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Rudi, A., De Vito, E., Verri, A., and Odone, F. Regularized kernel algorithms for support estimation. *Frontiers in Applied Mathematics and Statistics*, 3:23, 2017.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Schölkopf, B., Smola, A., and Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *arXiv preprint arXiv:1703.01030*, 2017.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine learning*, pp. 1096–1103, 2008.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, pp. 1433–1438, 2008.