Proceedings of Machine Learning Research vol vvv:1–28, 2025

Conditional Kernel Imitation Learning for Continuous State Environments

Rishabh Agrawal[1] RISHABHA@USC.EDU
Nathan Dahlin[2] NDAHLIN@ALBANY.EDU
Rahul Jain[1] RAHUL.JAIN@USC.EDU
Ashutosh Nayyar[1] ASHUTOSN@USC.EDU
[1] *University of Southern California,* [2] *University at Albany, SUNY*

Abstract

Imitation Learning (IL) is an important paradigm within the broader reinforcement learning (RL) methodology. Unlike most of RL, it does not assume availability of reward-feedback. Classical methods such as behavioral cloning and inverse reinforcement learning are highly sensitive to estimation errors, especially in continuous state space problems. Meanwhile, state-of-the-art (SOTA) IL algorithms often require additional online interaction data to be effective. In this paper, we consider the problem of imitation learning in continuous state space environments based solely on observed behavior, without access to transition dynamics information, reward structure, or, most importantly, any additional interactions with the environment. Our approach is based on the Markov balance equation and introduces a novel conditional kernel density estimation-based imitation learning framework. It uses conditional kernel density estimators for transition dynamics and seeks to satisfy a balance equation for the environment. We establish that our estimators satisfy asymptotic consistency and present a detailed analysis of their sample complexity. Through a series of numerical experiments on continuous state benchmark environments, we show consistently superior empirical performance over many SOTA IL algorithms. The full paper with the appendix is available at: https://github.com/rishabh-1086/CKIL.

Keywords: Imitation Learning, Reinforcement Learning, Offline learning.

1. Introduction

Reinforcement Learning (RL) has produced a series of breakthroughs over the last decade from exceeding human proficiency at playing simple games such as in the Atari suite (Mnih et al., 2015) to Go (Silver et al., 2016) and StarCraft (Vinyals et al., 2019), and to protein structure prediction systems (Jumper et al., 2021), etc. A fundamental premise that the 'reward is enough' (Silver et al., 2021) underlies all of such RL methodology. And yet, in most problems, a natural reward function is not available. Nor may it be possible to engineer one from intuition. Thus, a lot of effort is spent on reward shaping (Ng et al., 1999) to make RL algorithms work, often without success.

This problem is particularly acute with humans in the loop, either as demonstrators, or as evaluators. Often, demonstration data comes from human experts and it is impossible to infer precisely what reward function human experts really have in mind while taking actions. To be fair, several inverse RL (IRL) algorithms such as MaxEntropy-IRL (Ziebart et al., 2008) use a methodology wherein a reward function is first inferred from the demonstration data, and then used in conjunction with RL algorithms to design near-optimal policies. This has two lacunae. First, the performance of the RL algorithms can be very sensitive to errors in the reward function estimate. And second, the expert may not be using a policy that is optimal with respect to any reward objective at all! There is

thus a need to develop imitation learning (IL) algorithms that do not depend on reward inference as a first step (Arora and Doshi, 2021).

Behavioral Cloning (BC) is a straightforward approach to imitation learning (Pomerleau, 1988), using supervised learning to map states to actions. In fact, it largely ignores the inherent sequential nature of reinforcement learning problems. Unfortunately, it suffers from severe covariate shift issues as it fails to generalize to less-visited parts of the state space and ignores the sequential decision-making aspect. This results in propagation of errors in the agent's performance (Ross and Bagnell, 2010) resulting in a limited practical ability to generalize effectively.

To address the issue of compounding errors that afflicts methods like BC, IRL algorithms (Abbeel and Ng, 2004; Ng and Russell, 2000; Syed and Schapire, 2007; Ziebart et al., 2008; Levine et al., 2011) first learn a reward function, and then use reinforcement learning to find a policy, which makes them expensive to run, and sensitive to reward function estimation errors.

Adversarial Imitation Learning (AIL) on the other hand, is based on distribution matching through adversarial learning (Ho and Ermon, 2016; Fu et al., 2018; Ke et al., 2021) between the target state-action distribution and that generated by the behavioral policy during its interaction with the environment. Unfortunately, this is also its drawback: it needs new samples from the behavioral policy in every iteration, and is thus not suitable when only offline data is available. It is also challenging for continuous control problems since each visited state is visited at most once. There is broader IL literature (Ross et al., 2011) that also assumes access to a generative model to generate more data on the fly.

In this paper, we propose an imitation learning algorithm designed to address several key challenges. Our approach does not require *reward feedback*, avoids *distribution matching* through on-policy samples, and moves beyond *behavioral cloning* by leveraging the Markovian structure of the dynamics. It does not rely on access to a *generative model*, supports continuous state spaces, and enables batch processing of offline datasets. This formulation is particularly relevant for real-world decision-making applications, such as healthcare, robotics, and autonomous vehicles (Le Mero et al., 2022), where experimentation is costly or unsafe.

We present a novel framework grounded in the premise that demonstration trajectory data satisfies the balance equation linking the demonstration policy, the Markov decision process (MDP) transition density, and the induced Markov chain. We then estimate the MDP and Markov chain transition densities using conditional kernel density estimators. Starting with the discrete state and action setting, we extend the framework to continuous state spaces, proving estimators' asymptotic consistency and analyzing their sample complexities. Validation on continuous state problems demonstrates strong numerical performance. The *novel* combination of these two ideas together enables its excellent numerical performance. While extending to continuous action spaces is conceptually straightforward, additional effort is needed for numerical robustness.

Other Related Work. As already mentioned, behavioral cloning often has poor performance due to failure to account for Markovian dynamics and discarding distributional insights from the demonstrations (Ross and Bagnell, 2010; Piot et al., 2014). This is sought to be addressed (Ross et al., 2011; Piot et al., 2016) by either further online interactions with the environment, or the demonstrator, or using insights into model dynamics or the sparsity of rewards, all of which are in general impractical. The recent work (Xu et al., 2022) aims to overcome these by using additional data from non-expert policies without needing additional online interactions but such offline data may not be available. The EDM approach (Jarrett et al., 2020a) captures the expert's state occupancy measure

by training an explicit energy-based model but has many limitations as discussed in (Swamy et al., 2021).

There also have been efforts to further develop IRL approaches to overcome the limitations of earlier algorithms. (Klein et al., 2011, 2012) introduce *LSTD-μ*, a temporal difference technique that shares the weaknesses of least squares estimators, and is highly sensitive to basis feature selection and training data distribution. (Lee et al., 2019b) propose *DSFN* which uses a transition-regularized imitation network that produces an initial policy close to expert behavior and an efficient feature representation but assumes complete knowledge about reward features which is unrealistic (Arora and Doshi, 2021).

(Piot et al., 2014) introduced *RCAL*, a non-parametric algorithm that employs a boosting method to minimize a large margin objective augmented with a regularization term taking into account the MDP structure, avoids feature selection steps and can help tackle some of the above issues. (Chan and van der Schaar, 2021b) propose *AVRIL*, which jointly learns an approximate posterior distribution over reward and policy. (Garg et al., 2021a) introduce IQ-Learn, an off-policy IRL method that implicitly represents reward and policy using a learned soft Q-function. Unfortunately, both suffer from significant covariate shift problems, and thus encounter significant reward extrapolation errors, leading to misguided outcomes in novel environments. To address this, the *CLARE* (Yue et al., 2023) model-based offline Inverse Reinforcement Learning (IRL) approach introduces conservatism to its estimated reward but assumes access to an additional diverse dataset. It employs an IRL algorithm within an estimated dynamics model to learn the reward. However, it has been shown (Zeng et al., 2023) to have poor performance when there are predominantly low-quality transition samples from a behavior policy.

Adversarial Imitation Learning (AIL) approaches (Ho and Ermon, 2016) were a breakthrough when they were introduced a few years ago (Blondé and Kalousis, 2019; Kostrikov et al., 2019a). However, these approaches require online interactions with the environment, and thus are not applicable when we must work only with offline data. (Kostrikov et al., 2019b) introduces *ValueDICE* that employs a distribution matching strategy between the imitator and expert policies, and undertakes a complex alternating maximization-minimization optimization procedure. Still, it has difficulties in estimating the expectation of an exponential that introduces bias when approximating gradients using mini-batches (Jarrett et al., 2020a). The algorithm we present is quite distinct from the above, and demonstrates promising preliminary empirical results.

2. Preliminaries

The Imitation Learning Problem. An infinite horizon discounted Markov decision process (MDP) M is defined by the tuple (S, A, T, r, γ) with states $s \in S$, actions $a \in A$ and successor states $s' \in S$ drawn from the transition function $T(s'|s, a)$. The reward function $r : S \times A \to \mathbb{R}$ maps state-action pairs to scalar rewards, and γ is the discount factor. Policy π is a probability distribution over actions conditioned on state and is given by $\pi(a_t|s_t) = P_\pi(A_t = a_t|S_t = s_t)$, where $a_t \in A$, $s_t \in S$, $\forall t = 0, 1, 2, \cdots$. The induced occupancy measure of a policy is given as $\rho_\pi(s, a) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{s_t=s, a_t=a}]$, where the expectation is taken over $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim T(\cdot|s_t, a_t)$ for all t, and the initial state s_0. The corresponding state-only occupancy measure is given as $\rho_\pi(s) = \sum_a \rho_\pi(s, a)$. In the offline imitation learning (IL) framework, the agent is provided with trajectories generated by a demonstration policy π_D, collected as $D = \{(s_0, a_0), (s_1, a_1), (s_2, a_2), ...\}$; and is not allowed any further interaction with the environment. The data D does *not* include any reward

r_t at each time step. Indeed, rather than long-term reward maximization, the IL objective is to learn a policy $\pi^\star$ that is close to π_D in the following sense (Yue and Le, 2018):

$$\pi^\star \in \arg\min_{\pi\in\Pi} \mathbb{E}_{s\sim\rho_\pi}[\mathcal{L}(\pi(\cdot|s), \pi_D(\cdot|s))], \tag{1}$$

where Π is the set of all randomized (Markovian) stationary policies, and $\mathcal{L}$ is a chosen loss function. In practice, (1) can only be solved approximately since π_D is unknown and only transitions are observed in the dataset D.

Conditional Kernel Density Estimation (CKDE). The imitation learning approach we introduce depends on transition density estimation. Though statistical theory exists for it, conditional density estimation is a difficult problem due to a lack of clarity on what parametric families of density functions are good candidates. Thus, we adopt kernel density estimation (KDE), a nonparametric framework for the estimation of general continuous distributions (Wand and Jones, 1994).

We next outline the method for two continuous random variables, X and Y for the sake of simplicity. Let f and g denote the joint density of (X, Y) and the marginal density of X, respectively. The conditional distribution of Y, given X, is denoted as $h_{Y|X}(y|x) = f_{X,Y}(x, y)/g_X(x)$. Selecting a pair of kernel functions $K : \mathbb{R} \to \mathbb{R}$ and $K' : \mathbb{R} \to \mathbb{R}$ with respective scalar bandwidth parameters $h > 0$ and $h' > 0$ and given a set of n samples $\{(x_i, y_i)\}_{i=1}^n$, the KDE approximations $\widehat{f}$ and $\widehat{g}$ for the joint and marginal distributions, respectively, are obtained as follows:

$$\begin{aligned}\widehat{f}_{X,Y}(x, y) &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h}K\left(\frac{x - x_i}{h}\right)\frac{1}{h'}K'\left(\frac{y - y_i}{h'}\right),\\ \widehat{g}_X(x) &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h}K\left(\frac{x - x_i}{h}\right).\end{aligned} \tag{2}$$

Using the approximations in (2), the approximate conditional density $\widehat{h}_{Y|X}$ can be computed as

$$\widehat{h}_{Y|X}(y|x) = \frac{\widehat{f}_{X,Y}(x, y)}{\widehat{g}_X(x)}. \tag{3}$$

In more general cases involving random vectors, analogous estimates to those in (2) and (3) may be obtained using kernel functions defined according to

$$K_H(x) = |H|^{-\frac{1}{2}}K(H^{-\frac{1}{2}}x), \tag{4}$$

where H is a symmetric positive definite *bandwidth matrix* of appropriate dimension, m, with determinant $|H|$, and K is a real-valued function satisfying $\int_{\mathbb{R}^m} K(x)dx = 1$. For example, the KDE estimate for the marginal distribution of random vector X is defined as

$$\widehat{g}_X(x; H) = \frac{1}{n}\sum_{i=1}^{n} K_H\big(x - x_i\big). \tag{5}$$

An example of such a multivariate kernel function is the standard m-variate normal density function

$$K(x) := (2\pi)^{-\frac{m}{2}} \exp\left(-\frac{x^T x}{2}\right).$$

We note that conditional density estimation is quite difficult numerically, and conditional kernel density estimation (CKDE), the adaptation of KDE to conditional density estimation, is amongst the most effective methods available (see (Chacón and Duong, 2018) for more details).

3. Conditional Kernel Imitation Learning

We next describe our imitation learning algorithm. A key premise of our algorithm is that the demonstration trajectories from an expert must satisfy the Markov balance equation under the demonstrator's policy π_D, and thus we must use that to guide the agent's learning. The use of the balance equation then requires estimation of certain transition (conditional probability) density functions which we obtain via conditional kernel density estimation methods. Then, the problem reduces to identifying policies that best fit the balance equation. We elucidate this procedure in Algorithm 1, prove its theoretical properties and then present numerical evidence of its efficacy on a number of benchmark environments.

The Markov Balance Equation. Consider a demonstration policy π_D that is used to take actions starting from an initial state s_0. Let $T(s'|s,a)$ denote the transition density function of the MDP. Note that $\pi_D(a|s)$ is a randomized Markovian, stationary policy that denotes the probability of taking action a in state s. This will induce a Markov chain on the state space S. Let its transition density be denoted $P(s'|s)$. Then, the Markov balance equation is given by

$$P(s'|s) = \sum_a \pi_D(a|s) T(s'|s,a).$$

Unfortunately, this involves a sum, and hence is difficult to use. Therefore, we use the following alternate form which is a transition density of the induced Markov chain on the state-action space,

$$P_{\pi_D}(s',a'|s,a) = \pi_D(a'|s') T(s'|s,a). \tag{6}$$

The above balance equation is the basis of our IL approach. If we can estimate P_{π_D} and T in (6) (estimates denoted by $\widehat{P}$ and $\widehat{T}$ respectively), we can then infer a policy π_D that satisfies it. Unfortunately, the problem is ill-conditioned, and we will need to impose additional criterion such as a regularization term.

We consider a class of policies parametrized by θ and formulate the following optimization problem:

$$\begin{aligned}\min_{\theta\in\Theta} \int_{(s',a')} \int_{(s,a)} \left[\widehat{P}(s',a'|s,a) - \pi_\theta(a'|s')\widehat{T}(s'|s,a)\right]^2 d\mu(s,a)\, d\mu(s',a') \\ -\lambda \int_{s'} H(\pi_\theta(\cdot|s'))\, d\nu(s').\end{aligned} \tag{7}$$

In (7), the squared loss first term ensures that the balance equation is satisfied approximately. It is a simple but novel loss function rarely used in imitation learning in conjunction with a balance equation. The second term, $H(\pi_\theta(\cdot|s'))$ is the entropy of the probability distribution $\pi_\theta(\cdot|s')$ on actions when the state is s'. It penalizes less randomized policies in favor of more randomized policies. $\lambda \geq 0$ is a regularization parameter. Here, μ and ν denote reference probability measures on state-action pairs and states respectively. For example, they can be the counting measures obtained from the dataset. Θ is a given parameter set. The parameters could be weights of a neural network, for example.

Transition Density Estimation. We now discuss how to use kernel density estimation methods for estimating the two conditional densities P_{π_D} and T, first for the discrete state and action space setting, where the form of estimates is intuitive, and then for the continuous setting.

Discrete Spaces. When both the state and action spaces are discrete, the estimates $\widehat{P}$ and $\widehat{T}$ can be calculated as:

$$\widehat{T}(s'|s,a) := \frac{\eta(s,a,s')}{\eta(s,a)}, \text{ and } \widehat{P}(s',a'|s,a) := \frac{\eta(s,a,s',a')}{\eta(s,a)} \tag{8}$$

where η represents the counting measure, i.e., the frequency of a tuple or sequence in dataset D. If the denominator is zero, the conditional density is assumed uniform.

Continuous Spaces. Estimation of transition densities in this setting is more challenging since no visited state would appear twice, and most of the states would never be visited in any given dataset. This, thus calls for conditional density estimation using more sophisticated methods.

These include a range of techniques such as parametric approaches like mixture density network (Bishop, 1994), normalizing flows (Trippe and Turner, 2018); non-parametric methods like Gaussian process conditional density estimation (Dutordoir et al., 2018), CKDE (Li and Racine, 2006); and semi-parametric methods like least squares conditional density estimation (Sugiyama et al., 2010). In this study, we opt for using CKDE since it is a closed-form, non-parametric method that can be easily implemented and adapted to different data types. Further, CKDE provides a consistent estimator under appropriate conditions (Chacón and Duong, 2018).

As described in Section 2, the kernel functions use a difference between two samples/values (e.g., $x - x_i$) as their argument (5). This difference can be alternatively replaced by a suitable distance metric, as indicated in prior work (Haasdonk and Bahlmann, 2004). We define three distinct distance metrics: one to assess the dissimilarity between (next state, next action) pairs, another for (state, action) pairs, and a final one for next states. These metrics are denoted as $d_1 : (S \times A) \times (S \times A) \to \mathbb{R}_+$, $d_2 : (S \times A) \times (S \times A) \to \mathbb{R}_+$, and $d_3 : S \times S \to \mathbb{R}_+$ respectively. Similarly, we define H_1, H_2, and H_3 as bandwidth matrices for the kernels K_{H_1}, K_{H_2}, and K_{H_3}, respectively. H_1, H_2, and H_3 are square matrices with dimensions matching those of the (s', a') pair, (s, a) pair, and s', respectively. The CKDE approximations $\widehat{P}$ and $\widehat{T}$ are then computed as

$$\widehat{P}(s',a'|s,a) = \frac{\sum_{l=1}^{n} K_{H_1}\big(d_1((s',a'),(s'_l,a'_l))\big) K_{H_2}\big(d_2((s,a),(s_l,a_l))\big)}{\sum_{l=1}^{n} K_{H_2}\big(d_2((s,a),(s_l,a_l))\big)},$$

$$\text{and } \widehat{T}(s'|s,a) = \frac{\sum_{l=1}^{n} K_{H_3}\big(d_3(s',s'_l)\big) K_{H_2}\big(d_2((s,a),(s_l,a_l))\big)}{\sum_{l=1}^{n} K_{H_2}\big(d_2((s,a),(s_l,a_l))\big)}. \tag{9}$$

We combine the transition estimation procedures of (8) and (9) with the balance equation based optimization problem in (7) in our conditional kernel imitation learning (CKIL) algorithm whose pseudo-code is presented in Algorithm 1.

Remarks. 1. Aside from the regularization parameter λ in (11), the only other hyperparameters in our algorithm are the three kernel bandwidth parameters, which are not particularly sensitive. 2. While scalability can be a concern, conditional density estimation is a difficult problem, and kernel methods are among the best for them in terms of numerical performance while also coming with theoretical consistency guarantees which we provide next. 3. CKIL allows for batch learning, wherein it uses only N_{batch} (the batch size) tuples at each gradient step, thus allowing it to scale since it does not need to grow with the dataset size.

Theoretical Guarantees. Given standard assumptions, we demonstrate that as the training dataset size n approaches infinity, the CKDE estimates in (9) converge in probability to the true conditional distributions in Theorem 1. We then provide sample complexity result to show how fast these estimates concentrate around the true distribution in 2.

Theorem 1 *Let $\widehat{P}_n$ and $\widehat{T}_n$ be the CKDE estimates constructed using (9) and a buffer B with n tuples. Then, under appropriate conditions, for each (s, a, s', a'), as $n \to \infty$,*

$$\widehat{P}_n(s', a'|s, a) \xrightarrow{P} P_{\pi_D}(s', a'|s, a), \quad \text{and} \quad \widehat{T}_n(s'|s, a) \xrightarrow{P} T(s'|s, a). \tag{10}$$

Theorem 2 *Let $\widehat{P}_n$ and $\widehat{T}_n$ be the CKDE estimates constructed using (9). Then, under appropriate conditions, for each (s, a, s', a'), the following holds:*

1. *For $\widehat{T}_n(s'|s, a)$,*

$$\sup_{T \in \Sigma(\beta, L)} \mathbb{P}(|\widehat{T}_n\left(s'|s, a\right) - T(s'|s, a)| > \epsilon) < \delta$$

$$, \forall n > log\left(\frac{2}{\delta}\right) max\left[\frac{C_1}{h_3^{m_3} h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_1(h_3^2 + h_2^2)^{\frac{\beta}{2}}\right)}\right), \frac{C_2}{h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_2 h_2^{\beta}\right)}\right)\right]$$

2. *For $\widehat{P}_n(s', a'|s, a)$,*

$$\sup_{P \in \Sigma(\beta, L)} \mathbb{P}(|\widehat{P}_n\left(s', a'|s, a\right) - P(s', a'|s, a)| > \epsilon) < \delta$$

$$, \forall n > log\left(\frac{2}{\delta}\right) max\left[\frac{C_3}{h_1^{m_1} h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_3(h_1^2 + h_2^2)^{\frac{\beta}{2}}\right)}\right), \frac{C_4}{h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_4 h_2^{\beta}\right)}\right)\right]$$

, where m_i is the order of diagonal matrix $H_i, \forall i \in [3]$. $C_1, C_2, C_3, C_4 > 0$ and c_1, c_2, c_3, c_4 are some constants.

We prove the Theorems 1 and 2 and outline the assumptions made therein in the Appendix A.

4. Experimental Results

Experimental Setup. We evaluate our algorithm's performance by assessing its effectiveness across diverse benchmark environments sourced from OpenAI Gym (Brockman et al., 2016). These environments represent a wide spectrum of complexities commonly encountered in reinforcement learning. These include the MountainCar environment (Moore, 1990), CartPole (Barto et al., 1983), Acrobot (Sutton, 1995), and LunarLander (Klimov, 2019). To generate demonstration datasets D, we leverage pre-trained and hyperparameter-optimized agents available in the RL Baselines Zoo (Raffin, 2020). Specifically, we employ a PPO agent for LunarLander-v2, a DQN agent for CartPole-v1, and an A2C agent for Acrobot-v1.

Baseline Algorithms. We compare the performance of our CKIL algorithm (Algorithm 1), with a range of offline IRL/IL/AIL baselines, including several recent SOTA algorithms. These include Behavioral Cloning (BC), ValueDICE (VDICE) (Kostrikov et al., 2019b), reward-regularized classification (RCAL) (Piot et al., 2014), Energy-based Distribution Matching (EDM) (Jarrett et al., 2020a), AVRIL (Chan and van der Schaar, 2021b), and Deep Successor Feature Network (DSFN) (Lee et al., 2019b). Furthermore, we also compare against IQ-Learn (Garg et al., 2021a), a state-of-the-art model-free offline IRL algorithm. Please note that recent offline IRL/IL methods, like

Algorithm 1 Conditional Kernel Imitation Learning (CKIL)

Input: Expert dataset of trajectories $D = \{(s_i, a_i)\}_{i=1}^{n}$
Output: θ^*

1: Initialize policy parameter θ
2: Transform dataset D into (s, a, s', a') tuples, then store them in buffer B.
3: **for** $iter = 0, 1, \ldots$ **do**
4: Sample a batch b_{iter} of (s, a, s', a') tuples from B
5: Obtain $\widehat{P}, \widehat{T}$ in (9) via CKDE on b_{iter}
6: Calculate empirical estimate of the objective function in (7) using all $(s, a, s', a') \in b_{iter}$ as:

$$\sum_{(s',a')} \sum_{(s,a)} \left[\widehat{P}(s', a'|s, a) - \pi_\theta(a'|s')\widehat{T}(s'|s, a)\right]^2 + \lambda \sum_{s'} \sum_{a'} \pi_\theta(a'|s') log(\pi_\theta(a'|s')) \quad (11)$$

7: Update the policy parameter θ using gradient update to minimize the calculated empirical estimate of the objective function
8: **end for**
9: **return** θ^*

CLARE (Yue et al., 2023), have emerged in the literature. Some of these methods require additional diverse data alongside expert data, while others permit interaction with the environment, with "offline" meaning that these methods cannot query the expert. However, our work strictly assumes access to only expert data without further interactions or access to additional diverse data. Hence, we exclude these algorithms from comparison to maintain fairness. More detailed discussion can be found in Appendix C.

Implementation. The policy π_θ in (11) is embodied by a neural network (NN) architecture. This NN comprises two hidden layers featuring the Rectified Linear Unit (ReLU) activation function. The final layer employs a softmax function to produce a probability distribution over actions when given a state as an input. To facilitate comparison, all benchmarks adopt a common neural network architecture consisting of two hidden layers comprising 64 units each, with Exponential Linear Unit (ELU) activation functions. Training is carried out using the Adam optimizer (Kingma and Ba, 2015) with individually tuned learning rates. The implementation details, including hyperparameters of CKIL and benchmark algorithms, can be found in Appendix B and Appendix C, respectively.

Choice of Kernel. We use the Gaussian kernel due to its ability to uniformly approximate any continuous target function on a compact subset (Micchelli et al., 2006) which is particularly helpful in density estimation. We consider a Euclidean distance metric for d_1, d_2, and d_3 and utilize a diagonal bandwidth matrix with the same values across its diagonal elements. These matrices can then be denoted as $H_i = h_i I_{m_i}$, where m_i is the corresponding appropriate dimension for $i = 1, 2, 3$. Each h_i was tuned slightly but the conclusions of our experimental results are not very sensitive to this choice. We would like to emphasize that in prior research (Mammen et al., 2011; Schölkopf and Smola, 2002; Silverman, 1986; Ahmad and Ran, 2004), approaches for systematic selection of bandwidth parameters, which should decrease as the dataset size grows, have been developed. These methods can be applied to more intricate problems where manual tuning is impractical. Specific values of h_i employed for various experiments are detailed in Appendix B.

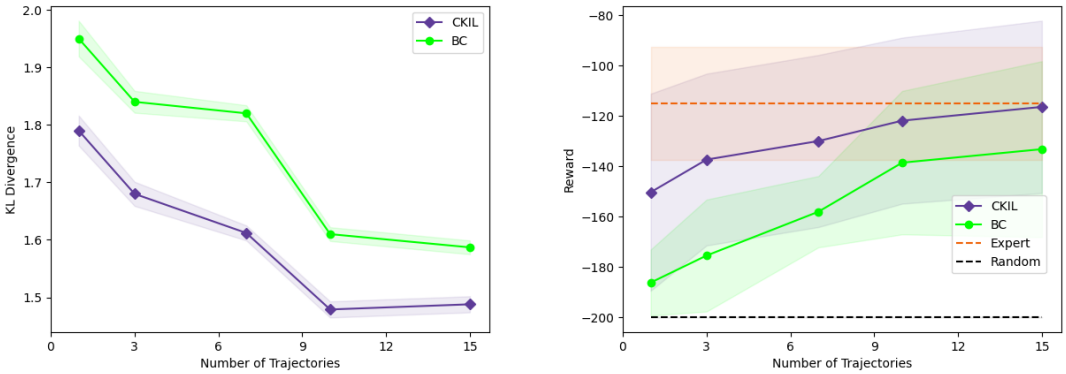


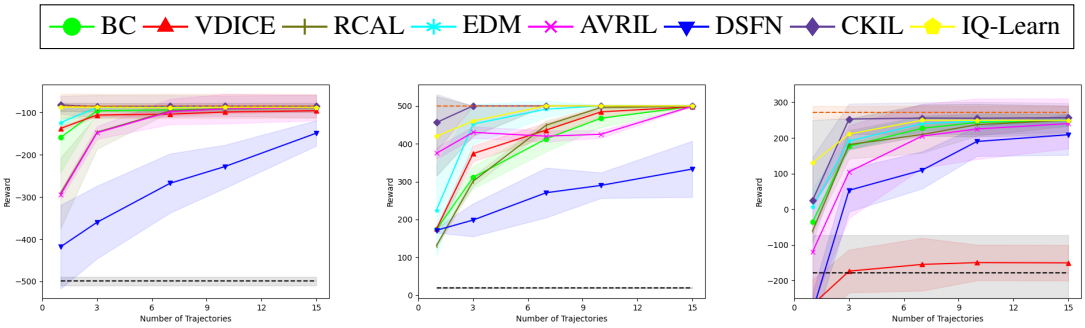
Figure 1: Plots for CKIL agent in a discretized MountainCar environment for a varying number of trajectories (a) Empirical KL divergence (lower values indicate better performance) (b) Average rewards attained (higher values indicate better performance).

Results

Discrete States and Actions. While our focus is primarily on continuous state-space settings requiring conditional density estimation, we first illustrate the effectiveness of our general approach in a discrete state-space problem. We compare CKIL's performance against Behavior Cloning and a random action agent using the discretized MountainCar environment. The 2-dimensional state space is discretized into a 15×15 grid, and $\widehat{P}$ and $\widehat{T}$ are estimated using equation (8). The dataset D is generated with an ϵ-perturbation of the policy in (Xiao, 2019), with $\epsilon = 0.05$. Starting with one trajectory in D and increasing to 15, Figure 1 summarizes the results. The KL-divergence plot shows increasing alignment between agent and expert policies as dataset size grows. With 15 trajectories, CKIL achieves expert-level proficiency, consistently outperforming Behavioral Cloning.

Continuous States. We next address our main goal, namely imitation learning for continuous state environments. When provided ample demonstration data, all benchmarks exhibit the capability to attain performance comparable to the optimized demonstration agents in each environment. Thus, we evaluate the algorithms' capacity to manage sample complexity in scenarios with limited data. In particular, we assess the change in performance of each algorithm as the size of the expert dataset D ranges from a single trajectory to a collection of 15 trajectories. This setup mirrors the configuration described in (Jarrett et al., 2020a). The average trajectory lengths in the expert data are 80 for Acrobot, 500 for CartPole, and 300 for LunarLander. The algorithms were trained to convergence on datasets of 1, 3, 7, 10, or 15 trajectories, each uniformly drawn from 1000 expert trajectories. The trained policies were deployed in each environment, and average scores over 300 episodes, serving as standard benchmarks for IL comparisons, were recorded for each algorithm. This process was repeated 10 times with diverse initialization and observed trajectories in each iteration.

Figure 2 compares the average rewards achieved by all algorithms as the demonstration dataset size increased in the Acrobot, CartPole, and LunarLander environments. Across all tasks, the results showcase CKIL's capability to learn effective policies, manifesting robust and consistently superior performance compared to the baselines considered, especially when data is scarce. We observe that CKIL attains expert-level performance in three environments of increasing difficulty, the CartPole, the Acrobot, and the LunarLander, within use of three trajectories. Remarkably, it is



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 2: Average rewards of benchmark IRL/IL/AIL and CKIL policies during real-time deployment, plotted against the number of trajectories in dataset D (higher values indicate better performance). Expert and random agents are represented as --- and ---, respectively.

able to approach expert-level performance on the CartPole environment by use of a single trajectory. Also note that CKIL's performance is almost uniformly better than that of the various baseline algorithms across the three environments. Among these baselines, the IQ-Learn algorithm demonstrates performance nearly comparable to that of CKIL. However, IQ-Learn, being an IRL algorithm, has significantly higher computational overhead than CKIL. Moreover, instabilities of the IQ-Learn approach is discussed in (Al-Hafez et al., 2023). It is notable that the off-policy adaptations of online algorithms (VDICE, DSFN) do not exhibit the same degree of consistent performance as their inherently offline counterparts. This highlights the inadequacy of solely adopting online algorithms in offline scenarios. The challenges in estimating the expectation of an exponential distribution may contribute to VDICE's underperformance. Moreover, ablation study is done in Appendix D.

The results suggest that combining a loss function based on the Markov balance equation, conditional kernel density estimation, and an entropy regularizer enables effective imitation learning in continuous state problems. Additionally, CKIL's design requires minimal hyperparameter tuning, unlike other model-based IL methods. CKIL also demonstrates superior computational efficiency, with training times significantly lower than other baselines. For example, CKIL converges in just a couple of minutes, while EDM and IQ-Learn require about half an hour.

5. Conclusions

In this paper, we introduced Conditional Kernel Imitation Learning (CKIL), a simple yet novel approach to imitation learning for continuous state-space problems. CKIL avoids reward modeling by leveraging the Markov balance equation and conditional kernel density estimators for transition densities of both the MDP and induced Markov chain. The algorithm is supported by theoretical consistency and sample complexity analysis, demonstrating strong empirical performance compared to SOTA offline IL, IRL, and AIL methods. Unlike some imitation learning algorithms, CKIL does not require access to a generative model or additional datasets. While imitation learning with limited training data often faces distribution shift challenges, our results suggest CKIL handles this issue more effectively than other strictly batch IL algorithms. Further details on this can be found in Appendix E. Future work may explore using scalable density estimation techniques, such as normalizing flows.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

Ibrahim A Ahmad and Iris S Ran. Data based bandwidth selection in kernel density estimation with parametric start via kernel contrasts. *Journal of Nonparametric Statistics*, 16(6):841–877, 2004.

Firas Al-Hafez, Davide Tateo, Oleg Arenz, Guoping Zhao, and Jan Peters. Ls-iq: Implicit reward regularization for inverse reinforcement learning. *arXiv preprint arXiv:2303.00599*, 2023.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

Christopher M. Bishop. Mixture density networks. Technical report, Aston University, 1994.

Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via generative adversarial nets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3138–3148. PMLR, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

José E. Chacón and Tarn Duong. *Multivariate kernel smoothing and its applications*. Chapman and Hall/CRC, 2018.

Alex James Chan and Mihaela van der Schaar. Scalable bayesian inverse reinforcement learning. https://github.com/XanderJC/scalable-birl, 2021a.

Alex James Chan and Mihaela van der Schaar. Scalable bayesian inverse reinforcement learning. In *International Conference on Learning Representations*, 2021b.

Jonathan D Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data without great coverage. *arXiv preprint arXiv:2106.03207*, 2021.

Vincent Dutordoir, Hugh Salimbeni, James Hensman, and Marc Deisenroth. Gaussian process conditional density estimation. *Advances in neural information processing systems*, 31, 2018.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039, 2021a.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. https://github.com/Div99/IQ-Learn, 2021b.

Bernard Haasdonk and Claus Bahlmann. Learning with distance substitution kernels. In *Joint pattern recognition symposium*, pages 220–227. Springer, 2004.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33:7354–7365, 2020a.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by batch imitation learning by energy-based distribution matching. https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/edm, 2020b.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pages 313–329. Springer, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, off-policy and model-free apprenticeship learning. In *European Workshop on Reinforcement Learning*, pages 285–296. Springer, 2011.

Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. *Advances in neural information processing systems*, 25, 2012.

Oleg Klimov. Openai gym: Rocket trajectory optimization is a classic topic in optimal control. https://github.com/openai/gym, 2019.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=Hk4fpoA5Km.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019b.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. https://github.com/google-research/google-research/tree/master/value_dice, 2020.

Luc Le Mero, Dewei Yi, Mehrdad Dianati, and Alexandros Mouzakitis. A survey on imitation learning techniques for end-to-end autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14128–14147, 2022.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. https://github.com/dtak/batch-apprenticeship-learning, 2019a.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5909–5915, 2019b.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. *Advances in neural information processing systems*, 24, 2011.

Qi Li and Jeffrey Scott Racine. *Nonparametric Econometrics: Theory and Practice*. Number 8355 in Economics Books. Princeton University Press, 2006.

Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020.

Enno Mammen, Maria Dolores Martinez Miranda, Jens Perch Nielsen, and Stefan Sperlich. Do-validation for kernel density estimation. *Journal of the American Statistical Association*, 106 (494):651–660, 2011.

Henry B Mann and Abraham Wald. On stochastic limit and order relationships. *The Annals of Mathematical Statistics*, 14(3):217–226, 1943.

Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12):2651–2667, 2006.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Andrew William Moore. Efficient memory-based learning for robot control. Technical report, University of Cambridge, 1990.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, page 663–670, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, volume 99, page 278–287, 1999.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, page 1249–1256, 2014.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems*, 28 (8):1814–1826, 2016.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1:305–313, 1988.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2(5), 2019.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pages 627–635, 2011.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

Masashi Sugiyama, Ichiro Takeuchi, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Daisuke Okanohara. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, 93(3):583–594, 2010.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995.

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Zhiwei Steven Wu. A critique of strictly batch imitation learning. *arXiv preprint arXiv:2110.02063*, 2021.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994.

Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In *International Conference on Machine Learning*, pages 6536–6544. PMLR, 2019.

Zhiqing Xiao. *Reinforcement Learning: Theory and Python Implementation*. China Machine Press, 2019.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. In *International Conference on Machine Learning*, pages 24725–24742. PMLR, 2022.

Sheng Yue, Guanbo Wang, Wei Shao, Zhaofeng Zhang, Sen Lin, Ju Ren, and Junshan Zhang. CLARE: Conservative model-based reward learning for offline inverse reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Yisong Yue and Hoang M. Le. Imitation learning (tutorial). *International Conference on Machine Learning (ICML)*, 2018.

Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Understanding expertise through demonstrations: A maximum likelihood framework for offline inverse reinforcement learning. *arXiv preprint arXiv:2302.07457*, 2023.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, page 1433–1438. AAAI Press, 2008.

Appendix A. Theoretical Guarantees

We state the following technical lemma for convergence in probability of kernel density estimators from i.i.d samples which we used for Theorem 1.

Lemma 3 *(Chacón and Duong, 2018) Suppose $X_1, X_2, \ldots, X_n$ are i.i.d vectors with probability density g. Let $\widehat{g}(\cdot; H)$, as given in Eq. (5), be the kernel density estimator constructed from these samples using kernel K and bandwidth matrix $H = H(n)$. Suppose the following assumptions hold.*

(C1) *Each entry of $\mathcal{H}_g(\cdot)$ be piecewise continuous and square integrable, where $\mathcal{H}_g$ is the $m \times m$ Hessian-matrix of g.*

(C2) *The kernel K, is square integrable, spherically symmetric and with a finite second order moment; this means that $\int_{\mathbb{R}^m} zK(z)dz = 0$ and $\int_{\mathbb{R}^m} zz^T K(z)dz = m_2(K)I_m$ (where $m_2(K)$ is independent of i, for $i \in \{1, 2, \ldots m\}$). Furthermore, $\int_{\mathbb{R}^m} K(z) = 1$.*

(C3) *The bandwidth matrices $H = H(n)$ form a sequence of positive definite, symmetric matrices such that as $n \to \infty$, vec $H(n) \to 0$, i.e. all entries of $H(n)$ approaches 0 and $n^{-1}|H(n)|^{-1/2} \to 0$, where vec is the vectorization operator which acts on a matrix by stacking its columns on top of one another.*

Then, $\widehat{g}(\boldsymbol{x}; H)$ converges in probability to $g(\boldsymbol{x})$ for each $\boldsymbol{x}$.

We introduce the kernel functions K_i, $i = 1, 2, 3$, so that using (4), the kernels K_{H_i} for $i = 1, 2, 3$ appearing in (9) can be expressed as

$$K_{H_i}(x) = |H_i|^{-\frac{1}{2}} K_i(H^{-\frac{1}{2}}x), \tag{12}$$

where x is of appropriate dimension. We will make the following assumptions (Chacón and Duong, 2018):

(A1) Suppose the buffer B in Algorithm 1 consists of n iid tuples (s, a, s', a') generated according to a probability distribution $P(s, a, s', a') = \mu(s, a)P_{\pi_D}(s', a'|s, a)$, where P_{π_D} is the transition probability density of the induced Markov chain on the state-action space under the demonstration policy π_D (see (6)) and μ is a reference probability measure on (s, a). Further, P has a density function g that is square-integrable and twice differentiable, with all of its second-order partial derivatives bounded, continuous and square integrable. Also assume that the marginals $P(s', s, a)$ and $P(s, a)$ satisfy these properties.

(A2) The kernels K_i for $i \in \{1, 2, 3\}$ in (12) are square integrable, zero-mean, spherically symmetric, and with common finite second-order moment $\int_{\mathbb{R}^{m_i}} zz^T K_i(z)dz = \sigma^2 I_{m_i}$.

(A3) For each kernel K_{H_i} as defined in (4), the bandwidth matrices $H_i(n)$ (where n is the number of tuples in B) form a sequence of positive definite, symmetric matrices such that $H_i(n) \to 0$ and $n^{-1/2}|H_i(n)|^{-1/2} \to 0$ as $n \to \infty$.

With these assumptions, we re-state Theorem 1 as follows:

Theorem 1 *Suppose assumptions (A1)-(A3) are true. Let $\widehat{P}_n$ and $\widehat{T}_n$ be the CKDE estimates constructed using (9) and a buffer B with n tuples. Then, for each (s, a, s', a'), as $n \to \infty$,*

$$\widehat{P}_n(s', a'|s, a) \xrightarrow{P} P_{\pi_D}(s', a'|s, a), \quad \text{and} \quad \widehat{T}_n(s'|s, a) \xrightarrow{P} T(s'|s, a). \tag{13}$$

Proof We now adopt Lemma 3 to give a detailed outline of proof for Theorem 1 under assumptions (A1)-(A3). First of all, we define the following:

$$\begin{aligned} \widehat{f}(s', s, a) &= \sum_{l=1}^{n} K_{H_3}\big(d_3(s', s'_l)\big) K_{H_2}\big(d_2((s, a), (s_l, a_l))\big), \\ \widehat{g}(s, a) &= \sum_{l=1}^{n} K_{H_2}\big(d_2((s, a), (s_l, a_l))\big). \end{aligned} \tag{14}$$

We can then argue as follows:

1. We assume (A1) that $P(s, a, s', a')$ has a density function g that is square-integrable and twice differentiable, with all of its second-order partial derivatives bounded, continuous and square integrable and so does its marginals $P(s', s, a)$ and $P(s, a)$. This leads to the satisfaction of condition (C1).

2. From assumption (A2), $\int_{\mathbb{R}^{m_i}} zK_i(z)dz = 0$ for $i = \{2, 3\}$, where z_i is a vector of size m_i. Partition the vector z as $z = [z_3, z_2]$ and let $m = m_2 + m_3$ and $K(z) = K_3(z_3)K_2(z_2)$. Then for $t \leq m_3$,

$$\begin{aligned} \int_{\mathbb{R}^m} z_t K(z)dz &= \int_{\mathbb{R}^m} z_t K_3(z_3)K_2(z_2)dz \\ &= \int_{\mathbb{R}^{m_2}} K_2(z_2)dz_2 \int_{\mathbb{R}^{m_3}} z_t K_3(z_3)dz_3 \\ &= \int_{\mathbb{R}^{m_3}} z_t K_3(z_3)dz_3 = 0, \end{aligned} \tag{15}$$

   which follows from (A2). This can be shown for any $t \in \{1, 2, \ldots, m\}$. Hence, $\int_{\mathbb{R}^m} zK(z)dz = 0$ is satisfied corresponding to condition (C2).

   Now,

$$\begin{aligned} &\int_{\mathbb{R}^m} zz^T K(z)dz \\ &\quad = \int_{\mathbb{R}^m} \begin{bmatrix} z_3 z_3^T & z_3 z_2^T \\ z_2 z_3^T & z_2 z_2^T \end{bmatrix} K_3(z_3)K_2(z_2)dz_3 dz_2 \\ &\quad = \sigma^2 \begin{bmatrix} I_{m_3} & 0 \\ 0 & I_{m_2} \end{bmatrix} = \sigma^2 I_m. \end{aligned}$$

   Hence, $K(z) = K_3(z_3)K_2(z_2)$ satisfies condition (C2).

3. Consider $H(n)$ to be a block diagonal matrix with $H_3(n)$ and $H_2(n)$ as the two block diagonal entries with $H_3(n)$ and $H_2(n)$ satisfying assumption (A3). Then the matrices $H(n)$

form a sequence of positive definite, symmetric matrices. Using (5) with this H, the kernel estimate for $P(s', s, a)$ takes the product kernel form as seen for $\widehat{f}(\cdot)$ in (14). Now, $|H(n)| = |H_3(n)||H_2(n)|$, this implies that as $n \to \infty$, $n^{-1}|H(n)|^{-1/2} \to 0$ because $n^{-1/2}|H_i(n)|^{-1/2} \to 0$ for $i = \{2, 3\}$. Also, vec $H(n) \to 0$ as vec $H_i(n) \to 0$ for $i = \{2, 3\}$. Therefore, condition (C3) is satisfied.

Having satisfied conditions (C1)-(C3), we may apply the argument found in Sections 2.6-2.9 of (Chacón and Duong, 2018) and conclude that

$$\widehat{f}(s', s, a) \xrightarrow{P} P(s', s, a),$$
$$\widehat{g}(s, a) \xrightarrow{P} P(s, a).$$

Finally, it follows from the Continuous Mapping Theorem (Mann and Wald, 1943) that taking the ratio of $\widehat{f}$ and $\widehat{g}$ produces a consistent estimator of

$$\frac{P(s', s, a)}{P(s, a)} = T(s'|s, a),$$

i.e.,

$$\widehat{T}_n(s'|s, a) = \frac{\widehat{f}(s', s, a)}{\widehat{g}(s, a)} \xrightarrow{P} T(s'|s, a).$$

A similar argument can be used to establish the asymptotic convergence in probability for the CKDE of $P_{\pi_D}(s', a'|s, a)$. ■

We now provide the proof of Theorem 2. We start by defining Hölder Class.

Definition 4 (Hölder Class) *Consider L and β to be positive numbers. We define the Hölder Class $\Sigma(\beta, L)$ as:*

$$\Sigma(\beta, L) = \{g : |D^s g(x) - D^s g(y)| \leq L\|x - y\|\}, \quad \forall s \text{ such that } |s| = \beta - 1, \text{ and all } x, y\}$$

, where $s = (s_1, \ldots, s_d)$ and $|s| = s_1 + \cdots + s_d$, $s! = s_1! \cdots s_d!$, $x^s = x_1^{s_1} \cdots x_d^{s_d}$ and

$$D^s = \frac{\partial^{s_1 + \cdots + s_d}}{\partial x_1^{s_1} \cdots \partial x_d^{s_d}}$$

Example 1 *If $x \in \mathbb{R}^d$, $\beta = 2$ and $d = 1$, the Holder Class $\Sigma(\beta, L)$ defined in Definition 4 is given as:*

$$\left|g'(x) - g'(y)\right| \leq L|x - y|, \quad \forall x, y \in \mathbb{R}^d.$$

When $\beta = 2$, it implies that the functions have bounded second derivatives.

Definition 5 (Taylor-Series Approximation) *The Taylor-Series approximation $g_{x,\beta}(u)$ for a function $g(x)$ can be defined as:*

$$g_{x,\beta}(u) = \sum_{|s| \leq \beta} \frac{(u - x)^s}{s!} D^s g(x).$$

With this Taylor-Series approximation, we deduce that if $g \in \Sigma(\beta, L)$ then $g(x)$ is close to its Taylor series approximation:

$$|g(u) - g_{x,\beta}(u)| \leq L\|u - x\|^{\beta}.$$

Example 2 *Consider a case of $\beta = 2$ and $g \in \Sigma(\beta, L)$, this implies*

$$\left|g(u) - \left[g(x) + (x-u)^T \nabla g(x)\right]\right| \leq L\|x - u\|^2$$

We now state assumptions that are needed to prove Theorem 2, parts of which are taken from Assumptions (A1-A3).

Assumption (A4) *Assume that the kernel K has the form $K(x) = G(x_1) \cdots G(x_d)$ where G has support on $[-1, 1]$, $\int G = 1$, $\int |G|^p < \infty$ for any $p \geq 1$, $\int |t|^{\beta}|K(t)|dt < \infty$ and $\int t^s K(t)dt = 0$ for $s \leq \beta$.*

We next provide lemmas that will be useful for proving Theorem 2. We consider the $\widehat{g}_X(x; H)$ defined in (5) as Kernel Density Estimator. Let the true distribution is given by $g(x)$. We start by bounding the bias and variance of $\widehat{g}_X(x; H)$. We define $g_X(x; H) = \mathbb{E}[\widehat{g}_X(x; H)]$. The bias for this estimator $\widehat{g}_X(x; H)$ is given by $g_X(x; H) - g(x)$. Consider the bandwidth matrix H to be a diagonal matrix with each diagonal entries having the value h.

Lemma 6 *The bias of $\widehat{g}_X(x; H)$ can be bounded as:*

$$\sup_{g \in \Sigma(\beta, L)} |g_X(x; H) - g(x)| \leq ch^{\beta}$$

for some c.

Proof Using the definition of bias, we expand on it as follows:

$$\begin{aligned}
|g_X(x; H) - g(x)| &= \int \frac{1}{h^d} K(\|u - x\|/h) g(u) du - g(x) \\
&= \left| \int K(\|v\|)(g(x + hv) - g(x)) dv \right| \\
&\leq \left| \int K(\|v\|) \left(g(x + hv) - g_{x,\beta}(x + hv)\right) dv \right| + \left| \int K(\|v\|) \left(g_{x,\beta}(x + hv) - g(x)\right) dv \right|
\end{aligned}$$

■

The first term is bounded by $Lh^{\beta} \int K(s)|s|^{\beta}$ since $g \in \Sigma(\beta, L)$. The second term is 0 from the properties on K since $g_{x,\beta}(x + hv) - g(x)$ is a polynomial of degree β and with no constant term.

We now provide a lemma to bound the variance of the estimator $\widehat{g}_X(x; H)$.

Lemma 7 *The variance of $\widehat{g}_X(x; H)$ can be bounded as:*

$$\sup_{g \in \Sigma(\beta, L)} \text{Var}\left(\widehat{g}_X(x; H)\right) \leq \frac{c}{nh^d}$$

for some $c > 0$.

Proof We write $\widehat{g}(x) = n^{-1}\sum_{i=1}^{n} Z_i$ where $Z_i = \frac{1}{h^d}K\left(\frac{\|x-X_i\|}{h}\right)$. Then,

$$\begin{aligned}\mathrm{Var}\left(Z_i\right) \leq \mathbb{E}\left(Z_i^2\right) &= \frac{1}{h^{2d}}\int K^2\left(\frac{\|x-u\|}{h}\right)g(u)du = \frac{h^d}{h^{2d}}\int K^2(\|v\|)g(x+hv)dv \\ &\leq \frac{\sup_x g(x)}{h^d}\int K^2(\|v\|)dv \leq \frac{c}{h^d}\end{aligned}$$

for some c since the densities in $\Sigma(\beta, L)$ are uniformly bounded. ■

We now state the Bernstein's inequality that will be used in the sample complexity result.

Theorem 8 (Bernstein's inequality) *Suppose that $Y_1, \ldots, Y_n$ are iid with mean μ, $\mathrm{Var}\left(Y_i\right) \leq \sigma^2$ and $|Y_i| \leq M$. Then,*

$$\mathbb{P}(|\bar{Y} - \mu| > \epsilon) \leq 2\exp\left\{-\frac{n\epsilon^2}{2\sigma^2 + 2M\epsilon/3}\right\}$$

Now, we derive a result that says how fast $\widehat{p}(x)$ concentrates around $p(x)$, thereby providing the sample complexity result.

Theorem 9 *For all small $\epsilon > 0$,*

$$\mathbb{P}\left(|\widehat{g}_X(x;H) - g_X(x;H)| > \epsilon\right) \leq 2\exp\left\{-cnh^d\epsilon^2\right\}.$$

Hence, for any $\delta > 0$,

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \sqrt{\frac{C\log(2/\delta)}{nh^d}} + ch^\beta\right) < \delta$$

for some constants C and c.

Proof By the triangle inequality,

$$|\widehat{g}_X(x;H) - g(x)| \leq |\widehat{g}_X(x;H) - g_X(x;H)| + |g_X(x;H) - g(x)|$$

From Lemma 6, $|g_X(x;H) - g(x)| \leq ch^\beta$ for some c. Now $\widehat{g}_X(x;H) = n^{-1}\sum_{i=1}^{n} Z_i$, where

$$Z_i = \frac{1}{h^d}K\left(\frac{\|x-X_i\|}{h}\right).$$

Since $|Z_i| \leq c_1/h^d$, where $c_1 = K(0)$. Moreover, $\mathrm{Var}\left(Z_i\right) \leq c_2/h^d$ from Lemma 7. Hence, by Bernstein's inequality,

$$\mathbb{P}\left(|\widehat{g}_X(x;H) - g_X(x;H)| > \epsilon\right) \leq 2\exp\left\{-\frac{n\epsilon^2}{2c_2h^{-d} + 2c_1h^{-d}\epsilon/3}\right\} \leq 2\exp\left\{-\frac{nh^d\epsilon^2}{4c_2}\right\}$$

, whenever $\epsilon \leq 3c_2/c_1$. If we choose $\epsilon = \sqrt{C\log(2/\delta)/\left(nh^d\right)}$ where $C = 4c_2$ then

$$\mathbb{P}\left(|\widehat{g}_X(x;H) - g_X(x;H)| > \sqrt{\frac{C\log(2/\delta)}{nh^d}}\right) \leq \delta.$$

Finally, from the triangle inequality, we get that:

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \sqrt{\frac{C\log(2/\delta)}{nh^d}} + ch^\beta\right) < \delta$$

for some constants C and c. Hence, we have for $n > \frac{Clog(2/\delta)}{h^d} max\left(\frac{1}{\epsilon^2}, \frac{1}{(\epsilon - ch^\beta)}\right)$,

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \epsilon\right) < \delta$$

■

With these results, we now provide the sample complexity analysis for the CKDE estimates $\widehat{P}_n$ and $\widehat{T}_n$ constructed using (9). Then, we operate under the assumption that the H_i are diagonal matrices with same diagonal entries h_i respectively for $i \in \{1, 2, 3\}$. We rewrite Theorem 2 below.

Theorem 2 *Suppose that Assumption (A4) is true. Let $\widehat{P}_n$ and $\widehat{T}_n$ be the CKDE estimates constructed using (9). Then, for each (s, a, s', a'), the following holds:*

1. *For $\widehat{T}_n(s'|s, a)$,*

$$\sup_{T\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{T}_n\left(s'|s,a\right) - T(s'|s,a)| > \epsilon\right) < \delta$$

$$,\forall n > log\left(\frac{2}{\delta}\right) max\left[\frac{C_1}{h_3^{m_3}h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_1(h_3^2 + h_2^2)^{\frac{\beta}{2}}\right)}\right), \frac{C_2}{h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_2 h_2^\beta\right)}\right)\right]$$

2. *For $\widehat{P}_n(s', a'|s, a)$,*

$$\sup_{P\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{P}_n\left(s', a'|s,a\right) - P(s', a'|s,a)| > \epsilon\right) < \delta$$

$$,\forall n > log\left(\frac{2}{\delta}\right) max\left[\frac{C_3}{h_1^{m_1}h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_3(h_1^2 + h_2^2)^{\frac{\beta}{2}}\right)}\right), \frac{C_4}{h_2^{m_2}} max\left(\frac{1}{\epsilon^2}, \frac{1}{\left(\epsilon - c_4 h_2^\beta\right)}\right)\right]$$

, where m_i is the order of diagonal matrix $H_i, \forall i \in [3]$. $C_1, C_2, C_3, C_4 > 0$ and c_1, c_2, c_3, c_4 are some constants.

Proof First of all, we define the following:

$$\begin{aligned}\widehat{f}(s', s, a) &= \sum_{l=1}^{n} K_{H_3}\big(d_3(s', s'_l)\big) K_{H_2}\big(d_2((s, a), (s_l, a_l))\big),\\ \widehat{g}(s, a) &= \sum_{l=1}^{n} K_{H_2}\big(d_2((s, a), (s_l, a_l))\big).\end{aligned} \tag{16}$$

Using Lemma 6, we get that the bias of $\widehat{f}(s',s,a) \leq c_1(h_3^2+h_2^2)^{\frac{\beta}{2}}$. Using Lemma 7, the variance of $\widehat{f}(s',s,a) \leq \frac{c_2}{nh_3^{m_3}h_2^{m_2}}$. Using these in Lemma 9, we have,

$$\sup_{f\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{f}(s',s,a)-f(s',s,a)|>\epsilon\right)<\delta$$
$$,\forall n > \frac{C_1 log(2/\delta)}{h_3^{m_3}h_2^{m_2}}\max\left(\frac{1}{\epsilon^2},\frac{1}{\left(\epsilon-c_1(h_3^2+h_2^2)^{\frac{\beta}{2}}\right)}\right)$$

Similarly, using Lemma 6, we get that the bias of $\widehat{g}(s,a) \leq c_3(h_2)^{\beta}$ and using Lemma 7, the variance of $\widehat{g}(s,a) \leq \frac{c_4}{nh_2^{m_2}}$. Using these in Lemma 9, we have,

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}(s,a)-g(s,a)|>\epsilon\right)<\delta$$
$$,\forall n > \frac{C_2 log(2/\delta)}{h_2^{m_2}}\max\left(\frac{1}{\epsilon^2},\frac{1}{\left(\epsilon-c_2h_2^{\beta}\right)}\right)$$

Hence, by combining these results, we get that:

$$\sup_{T\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{T}_n\left(s'|s,a\right)-T(s'|s,a)|>\epsilon)<\delta$$
$$,\forall n > log\left(\frac{2}{\delta}\right)\max\left[\frac{C_1}{h_3^{m_3}h_2^{m_2}}\max\left(\frac{1}{\epsilon^2},\frac{1}{\left(\epsilon-c_1(h_3^2+h_2^2)^{\frac{\beta}{2}}\right)}\right),\frac{C_2}{h_2^{m_2}}\max\left(\frac{1}{\epsilon^2},\frac{1}{\left(\epsilon-c_2h_2^{\beta}\right)}\right)\right]$$

A similar argument can be used to compute the sample complexity of $\widehat{P}_n$. ■

Appendix B. Hyperparameters for CKIL

We report the hyperparameters used for CKIL. For discrete case, since number of states and number of actions are both finite, instead of using a parameterized policy, we defined a policy for each states and learnt it via optimizing the objective in (7). We used a learning rate of 0.5 for the same. We set $\lambda = 0.001$.

For continuous case, we adopted a neural network architecture for learning the policy. This neural network consisted of 2 hidden layers with 64 nodes followed by 32 nodes. Final layer consisted of a Softmax function to output the policy when a state was provided as an input. We used Adam optimizer and a learning rate of 0.01. We use the same value for bandwidth parameters h_1 and h_2. Let m_1 be the dimension of a state s, we then take the value of h_3 as $h_3 = h_1^{\frac{m_1+1}{m_1}}$. We report the values of bandwidth parameter h_1 in Table 1. We set $\lambda = 0.001$.

	Environment		
Trajectories (τ)	Acrobot-v1	CartPole-v1	LunarLander-v2
$\tau = 1$	0.1	0.01	0.009
$\tau = 3$	0.05	0.005	0.005
$\tau = 7$	0.01	0.001	0.0005
$\tau = 10$	0.008	0.0008	0.00008
$\tau = 15$	0.005	0.0001	0.00005

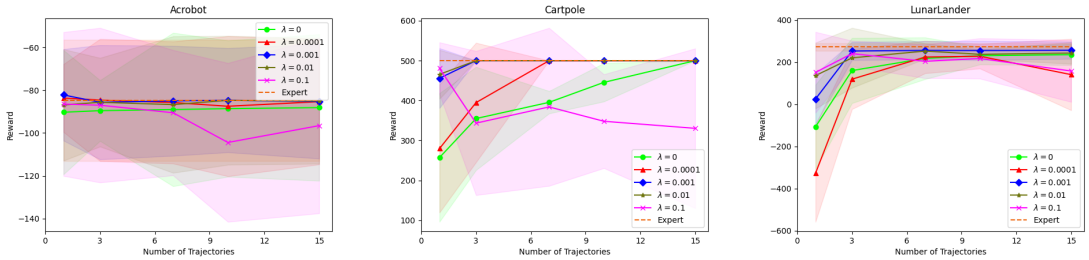
Table 1: h_1 values used for CKIL on different environments for varying number of trajectories during training.

Appendix C. Benchmark Algorithms and Hyperparameters

Baseline Algorithms. We compare the performance of our CKIL algorithm (Algorithm 1), with a range of offline IRL/IL/AIL baselines, including several recent state-of-the-art algorithms. This comprehensive assessment covers a spectrum of methodologies, including the inherently offline Behavioral Cloning (BC); ValueDICE (VDICE), a sample-efficient AIL approach designed for offline scenarios by removing replay regularization; reward-regularized classification (RCAL), a large margin classification approach, which introduces a sparsity-based penalty on inferred rewards to exploit dynamics information; Energy-based Distribution Matching (EDM), an offline imitation learning algorithm that captures the expert's state occupancy patterns through explicit training of an energy-based model; AVRIL, a recent model-free offline IRL technique employing a variational approach to simultaneously learn an approximate posterior distribution over rewards and policies; and Deep Successor Feature Network (DSFN), an offline adaptation of the max-margin IRL algorithm that transcends linear approaches by introducing a deep network architecture and employing least-squares temporal-difference learning to produce both reward and policy outputs. Furthermore, we compare against IQ-Learn, a state-of-the-art model-free offline IRL algorithm.

Implementation details. In the case of VDICE, we used the open-sourced code provided at (Kostrikov et al., 2020). It is worth noting that, for VDICE, offline learning is achieved by configuring the "replay regularization" coefficient to zero. Our execution of EDM leveraged the source code accessible at (Jarrett et al., 2020b). It is essential to highlight that the contrast between BC and EDM predominantly stems from the introduction of L_ρ, an occupancy loss defined in the EDM work, while deriving the RCAL loss is a straightforward process involving the inversion of the Bellman equation. As for AVRIL and DSFN, the applicable source codes are accessible at (Chan and van der Schaar, 2021a), (Lee et al., 2019a) respectively. Similarly, for IQ-Learn, we utilised the source code available at (Garg et al., 2021b).

We consider the hyperparameters associated with various benchmarks, as outlined in (Jarrett et al., 2020a). To ensure comprehensiveness, we present them herein. When feasible, the policies trained by all imitation algorithms utilize an identical policy network structure, comprising of two fully connected hidden layers, each containing 64 units with ELU activation function. Across all environments, we adopt the Adam optimizer with a batch size of 64, conducting $10,000$ iterations, and employing a learning rate of $1e-3$. With the exception of the explicit standardization of policy networks among imitation algorithms, all comparators are realized using the unaltered publicly



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 3: Average rewards achieved by CKIL agent when trained using different λ values during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

accessible source code. When relevant, we employ the optimal hyperparameters as indicated in the original implementations.

C.1. VDICE

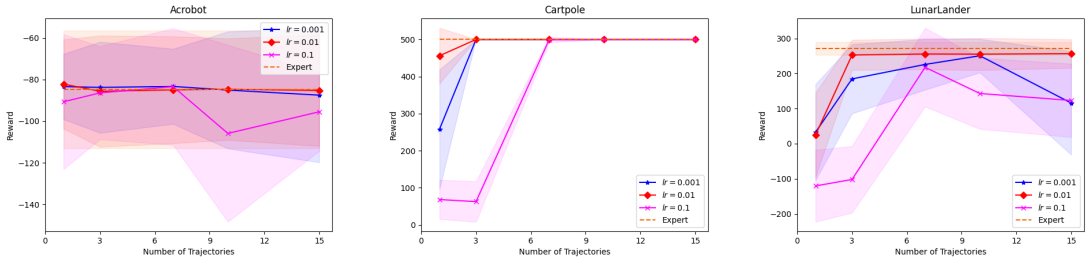
We employ the publicly accessible source code from https://github.com/google-research/google-research/tree/master/value_dice. To accommodate discrete action spaces, we incorporate a Gumbel-softmax parameterization for the final layer of the actor network. Both the actor and discriminator architecture encompass two fully connected hidden layers, each composed of 64 units activated by ReLU functions. Consistent with the original framework, the output is merged with the action and propagated through two additional hidden layers, each containing 64 units. In addition, we set the "replay regularization" coefficient at zero for strict batch learning. Furthermore, the actor network is subjected to "orthogonal regularization" with a coefficient of 1e-4. The actor network's learning rate is set at 1e-5, while the discriminator operates with a learning rate of 1e-3.

C.2. RCAL

This introduces an expansion of the policy loss by incorporating an extra sparsity-driven loss concerning the inferred rewards $\widehat{R}(s,a)$, defined as $f_\theta(s)[a] - \gamma softmax_{a'} f_\theta(s')[a']$, acquired through the inversion of the Bellman equation. The policy network employed is the fully-connected type detailed previously. The coefficient for sparsity-based regularization is designated as 1e-2.

C.3. EDM

We utilize the code accessible at https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/edm. Particularly for EDM, the hyperparameters for joint Energy-Based Model (EBM) training are adopted from https://github.com/wgrathwohl/JEM. These parameters include a noise coefficient of $\sigma = 0.01$, a buffer size of $\kappa = 10000$, a length of $\iota = 20$, and a reinitialization value of $\delta = 0.05$. These predefined configurations align effectively with the SGLD (Stochastic Gradient Langevin Dynamics) step size of $\alpha = 0.01$.



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 4: Average rewards achieved by CKIL agent when trained using different learning rates during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

C.4. BC

The sole distinction between Behavior Cloning (BC) and EDM lies in the inclusion of L_ρ, which is omitted in the implementation of BC. The policy network remains consistent with the description provided earlier.

C.5. AVRIL

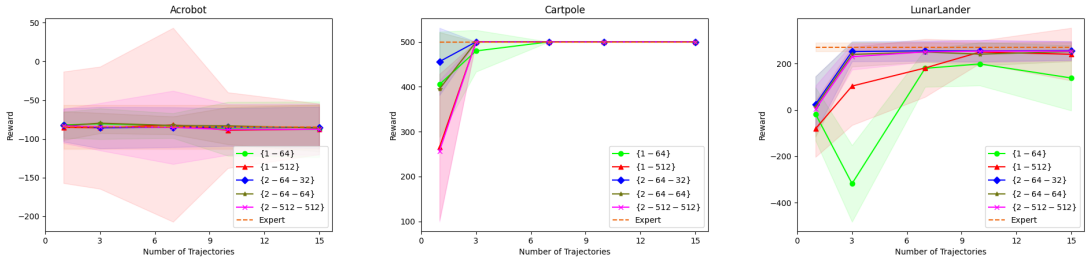
We use the code available at https://github.com/XanderJC/scalable-birl. The policy network remains consistent with the description provided earlier. γ, used while computing the TD error, equals 1. We used the default parameters provided in their GitHub repository.

C.6. DSFN

We adopt the source code accessible at https://github.com/dtak/batch-apprenticeship-learning. We utilize a "warm-start" policy network consisting of two shared layers with dimensions 128 and 64, employing tanh activation. The hidden layer with a size of 64 serves as the feature map within the IRL algorithm. Each multitask head within the warm-start policy network features a hidden layer comprising 128 units and is activated by tanh. The Deep Q-Network (DQN), utilized for learning the optimal policy based on a set of reward weights, comprises two fully-connected layers, each containing 64 units. Similarly, the DSFN, employed for estimating feature expectations, comprises two hidden fully-connected layers, each containing 64 units. Across all environments, the warm-start policy network undergoes training for $50,000$ steps, employing the Adam optimizer with a learning rate of 3e-4 and a batch size of 64. The DQN network is trained for $30,000$ steps, using a learning rate of 3e-4 and a batch size of 64 (with the Adam optimizer). Lastly, the DSFN network is trained for $50,000$ iterations, utilizing a learning rate of 3e-4 and a batch size of 32 (with the Adam optimizer).

C.7. IQ-LEARN

We use the code available at https://github.com/Div99/IQ-Learn. The policy network remains consistent with the description provided earlier. As highlighted in their implementation, we use a batch size of 32 and Q-network learning rate of 1e-4 with entropy coefficient of 0.01.



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 5: Average rewards achieved by CKIL agent when trained using different Neural Network Architectures during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

All experiments involving CKIL and benchmarks were conducted on an M2 MacBook Air machine equipped with 16 GB of RAM and a 512 GB SSD.

Appendix D. Ablation Study

In this section, we present the evaluation of CKIL's performance under various ablations.

D.1. Varying λ for entropy regularization

Figure 3 shows the average cumulative reward on the considered gym environments as a function of different λ values. We observe that when the data is very scarce (eg. 1 trajectory), having a higher λ value helps as we are less certain about which action is best to take in a given state. Conversely, we observe that having a higher λ performs poorly in comparison to lower λ values with increased data.

D.2. Varying learning rate

Figure 4 shows the average cumulative reward on the considered gym environments as a function of different learning rates (lr) values. We observe that the learning rate of 0.01 does well across tasks where the variance in performance is low across different episodes for any given environment along with a similar or better mean performances than other learning rate values.

D.3. Varying Neural Network Size

Figure 5 shows the average cumulative reward on the considered gym environments as a function of different Neural Network size values. The legend is in the form $\{a - b\}$ or $\{a - b - c\}$ where a indicates the number of hidden layers, followed by number of hidden nodes in each layer, which are denoted by b and c. For example, $\{1 - 64\}$ represents a neural network with one hidden layer containing 64 hidden nodes in it. Similarly, $\{2 - 64 - 32\}$ represents a neural network with two hidden layers, with 64 hidden nodes in the first layer followed by 32 hidden nodes in the second hidden layer.

	Algorithm		
Trajectories (τ)	EDM	IQ-Learn	CKIL
$\tau = 1$	-406.37 ± 340.03	5.12 ± 190.12	-0.81 ± 169.56
$\tau = 3$	-299.21 ± 184.60	159.726 ± 120.18	202.73 ± 93.03
$\tau = 7$	-240.38 ± 202.53	205.96 ± 99.68	239.78 ± 62.59
$\tau = 10$	-38.36 ± 130.54	232.28 ± 87.66	245.41 ± 57.85
$\tau = 15$	89.32 ± 101.45	242.32 ± 77.59	247.4 ± 58.82

Table 2: Performance of EDM, IQ-Learn, and CKIL in the presence of initial distribution shift (higher values indicate better performance)

For all the environments, we observe that having 2 hidden layers provides good performance, and the performance is not sensitive to the number of hidden nodes in the two hidden layers. Furthermore, with one hidden layer, spread is very high in some cases. Therefore, we selected a neural network with 2 hidden layers, containing 64 nodes in the first hidden layer and 32 nodes in the second hidden layer for our gym experiments.

Appendix E. Discussion on Distribution Shift

In this work, we have discussed the task of imitation learning in a strictly batch setting. Specifically, we assumed that only expert data was available, with no possibility for further interaction with the environment. Another line of research in imitation learning aims to incentivize the imitating policy to remain within the distribution of states encountered in expert demonstrations. This research typically follows two approaches. The first approach assumes access to additional data from a behavioral policy (which may be sub-optimal) along with the expert data. This additional data is used to provide coverage, as expert data is generally narrow. Examples of this approach include methods like CLARE (Yue et al., 2023) and MILO (Chang et al., 2021). The second approach involves techniques such as assigning a unit reward to all demonstrated actions in demonstrated states and zero otherwise (Reddy et al., 2019), such as random expert distillation (Wang et al., 2019). Generally, these methods follow a "two-step" formula: first, a surrogate reward function is derived or defined; second, this reward function is optimized through environment interactions, making these techniques inherently online, rendering it inapplicable in our strictly batch setting (Liu et al., 2020).

Nevertheless, we investigated the effects of an initial distribution shift in the LunarLander-v2 environment, drawing inspiration from the approach in (Garg et al., 2021a). Typically, the agent starts in a small area at the center-top of the screen. However, we modified the environment so that the agent begins near the top-left corner instead. Using expert data from the standard, unmodified environment, we aimed to determine if the agent could still successfully learn to land the lunar module despite the shift in its initial conditions during testing. For comparisons, we consider EDM and IQ-Learn algorithms as these methods don't rely on additional data or further interactions with the environment, thus utilizing the same setup as ours. The findings are reported in Table 2. As anticipated, all algorithms perform poorly when the available data is very scarce. However, as the amount of data gradually increases, the CKIL agent demonstrates the ability to effectively land the lunar lander despite the initial distribution shift, even with limited training data (approximately 10

trajectories). Additionally, CKIL outperforms baseline algorithms like EDM and IQ-Learn under these conditions. Our experimental results suggest that our algorithm handles the distribution shift problem more effectively than the other baseline algorithms in the same setup.