Machine Learning (2021) 110:2295–2334
https://doi.org/10.1007/s10994-021-05984-x

Inverse reinforcement learning in contextual MDPs

Stav Belogolovsky[1] · Philip Korsunsky[1] · Shie Mannor[1,2] · Chen Tessler[1] · Tom Zahavy[1]

Received: 15 May 2020 / Revised: 30 September 2020 / Accepted: 12 April 2021 /
Published online: 12 May 2021

Abstract
We consider the task of Inverse Reinforcement Learning in Contextual Markov Decision Processes (MDPs). In this setting, contexts, which define the reward and transition kernel, are sampled from a distribution. In addition, although the reward is a function of the context, it is not provided to the agent. Instead, the agent observes demonstrations from an optimal policy. The goal is to learn the reward mapping, such that the agent will act optimally even when encountering previously unseen contexts, also known as zero-shot transfer. We formulate this problem as a non-differential convex optimization problem and propose a novel algorithm to compute its subgradients. Based on this scheme, we analyze several methods both theoretically, where we compare the sample complexity and scalability, and empirically. Most importantly, we show both theoretically and empirically that our algorithms perform zero-shot transfer (generalize to new and unseen contexts). Specifically, we present empirical experiments in a dynamic treatment regime, where the goal is to learn a reward function which explains the behavior of expert physicians based on recorded data of them treating patients diagnosed with sepsis.

Keywords Reinforcement learning · Contextual · Inverse

Stav Belogolovsky and Philip Korsunsky have equally contributed to this work.

Editors: Yuxi Li, Alborz Geramifard, LihongLi, Csaba Szepesvari, Tao Wang.

✉ Stav Belogolovsky
stav.belo@gmail.com

Philip Korsunsky
philip.korsunsky@gmail.com

[1] Faculty of Electrical and Computer Engineering, Technion Israel Institute of Technology, Haifa, Israel

[2] Nvidia Research, Tel Aviv, Israel

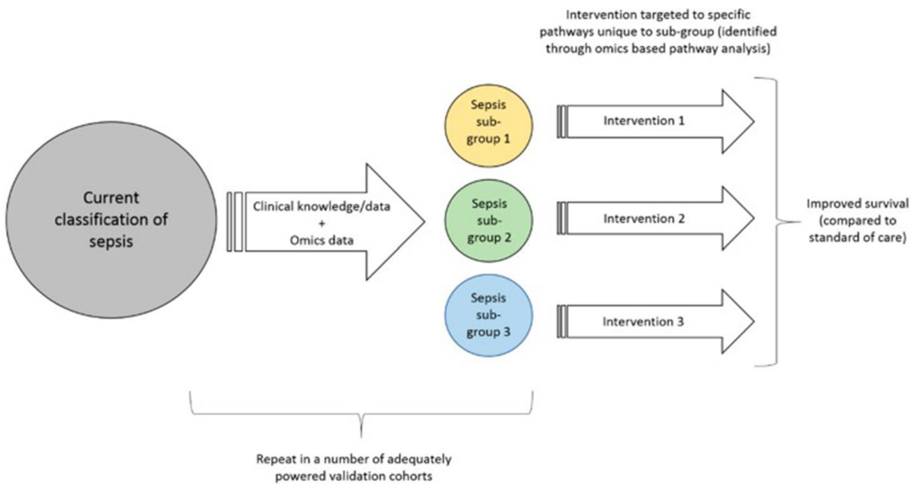

Intervention targeted to specific pathways unique to sub-group (identified through omics based pathway analysis)
Current classification of sepsis
Clinical knowledge/data + Omics data
Sepsis sub-group 1
Sepsis sub-group 2
Sepsis sub-group 3
Intervention 1
Intervention 2
Intervention 3
Improved survival (compared to standard of care)
Repeat in a number of adequately powered validation cohorts

Fig. 1 Personalized medicine in sepsis treatment. Credit: Itenov et al. (2018)

1 Introduction

Real-world sequential decision making problems often share three important properties — (1) *the reward function is often unknown*, yet (2) expert demonstrations can be acquired, and (3) the reward and/or dynamics often depend on a *static parameter*, also known as the context. For a concrete example, consider a dynamic treatment regime (Chakraborty & Murphy 2014), where a clinician acts to improve a patient's medical condition. While the patient's dynamic measurements, e.g., heart rate and blood pressure, define the state, there are static parameters, e.g., age and weight, which determine how the patient reacts to certain treatments and what form of treatment is optimal.

The contextual model is motivated by recent trends in personalized medicine, predicted to be one of the technology breakthroughs of 2020 by MIT's Technology Review (Juskalian et al. 2020). As opposed to traditional medicine, which provide a treatment for the "average patient", in the contextual model, patients are separated into different groups for which the medical decisions are tailored (Fig. 1). This enables the decision maker to provide tailored decisions (e.g., treatments) which are more effective, based on these static parameters.

For example, in Wesselink et al. (2018), the authors study organ injury, which may occur when a specific measurement (mean arterial pressure) decreases below a certain threshold. They found that this threshold varies across different patient groups (contextual behavior). In other examples, clinicians set treatment goals for the patients, i.e., they take actions to drive the patient measurements towards some predetermined values. For instance, in acute respiratory distress syndrome (ARDS), clinicians argue that these treatment goals should depend on the static patient information (the context) (Berngard et al. 2016).

In addition to the contextual structure, we consider the setting where the reward itself is unknown to the agent. This, also, is motivated by real-world problems, in which serious issues may arise when manually attempting to define a reward signal. For instance, when treating patients with sepsis, the only available signal is the mortality of the patient at the end of the treatment (Komorowski et al. 2018). While the goal is to improve the patients' medical condition, minimizing mortality does not necessarily capture this objective. This

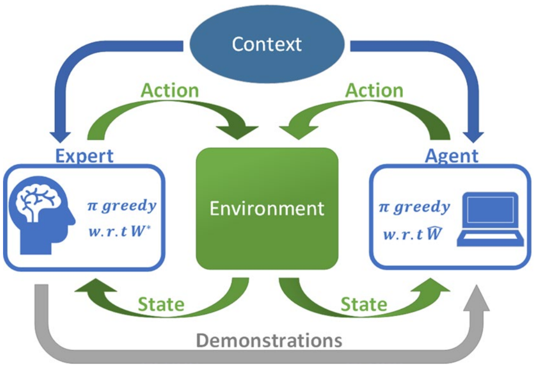


Fig. 2 The COIRL framework: a context vector parametrizes the environment. For each context, the expert uses the true mapping from contexts to rewards, W^*, and provides demonstrations. The agent learns an estimation of this mapping $\hat{W}$ and acts optimally with respect to it

model is illustrated in Fig. 2. The agent observes expert interactions with the environment, either through pre-collected data, or through interactive expert interventions. The agent then aims to find a reward which *explains* the behavior of the expert, meaning that the experts policy is optimal with respect to this reward.

To tackle these problems, we propose the Contextual Inverse Reinforcement Learning (COIRL) framework. Similarly to Inverse Reinforcement Learning (Ng & Russell 2000, IRL), provided expert demonstrations, the goal in COIRL is to learn a reward function which explains the expert's behavior, i.e., a reward function for which the expert behavior is optimal. In contrast to IRL, in COIRL the reward is not only a function of the state features but also the context. Our aim is to provide theoretical analysis and insights into this framework. As such, throughout most of the paper we consider a reward which is linear in both the context and the state features. This analysis enables us to propose algorithms, analyze their behavior and provide theoretical guarantees. We further show empirically in Sect. 4 that our method can be easily extended to mappings which are non-linear in the context using deep neural nets.

The paper is organized as follows. In Sect. 2 we introduce the Contextual MDPs and provide relevant notation. In Sect. 3.1 we formulate COIRL, with a linear mapping, as a convex optimization problem. We show that while this loss is not differentiable, it can be minimized using subgradient descent and provide methods to compute subgradients. We propose algorithms based on Mirror Descent (MDA) and Evolution Strategies (ES) for solving this task and analyze their sample complexity. In addition, in Sect. 3.2, we adapt the cutting plane (ellipsoid) method to the COIRL domain. In Sect. 3.3 we discuss how existing IRL approaches can be applied to COIRL problems and their limitations. Finally, in Sect. 3.4 we discuss how to efficiently (without resolving the MDP) perform zero-shot transfer to unseen contexts.

These theoretical approaches are then evaluated, empirically, in Sect. 4. We perform extensive testing of our methods and the relevant baselines both on toy problems and on a dynamic treatment regime, which is constructed from real data. We evaluate the run-time of IRL vs COIRL, showing that when the structure is indeed contextual, standard IRL schemes are computationally inefficient. We show that COIRL is capable of generalizing (zero-shot transfer) to unseen contexts, while behavioral cloning (log-likelihood action matching) is sub-optimal and struggles to find a good solution. These results show that in contextual problems, COIRL enables the agent to quickly recover a reward mapping that explains the expert's behavior, outperforming previous methods across several metrics and can thus be seen as a promising approach for real-life decision making.

Our contribution is three fold: First, the formulation of COIRL problem as a convex optimization problem, and the novel adaptation of the descent methods to this setting. Second, we provide theoretical analysis for the *linear* case for all of the proposed methods. Third, we bridge between the theoretical results and real-life application through a series of experiments that aim to apply COIRL to sepsis treatment (Sect. 4).

2 Preliminaries

2.1 Contextual MDPs

A Markov Decision Process (Puterman 1994, **MDP**) is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, \xi, R, \gamma)$ where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ a finite action space, $P : S \times S \times A \to [0,1]$ the transition kernel, ξ the initial state distribution, $R : \mathcal{S} \to \mathbb{R}$ the reward function and $\gamma \in [0,1)$ is the discount factor. A Contextual MDP (Hallak et al. 2015, **CMDP**) is an extension of an MDP, and is defined by $(\mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{M}, \gamma)$ where $\mathcal{C}$ is the context space, and $\mathcal{M}$ is a mapping from contexts $c \in \mathcal{C}$ to MDPs: $\mathcal{M}(c) = (\mathcal{S}, \mathcal{A}, P_c, \xi, R_c, \gamma)$. For consistency with prior work, we consider the discounted infinite horizon scenario. We emphasize here that all the results in this paper can be easily extended to the episodic finite horizon and the average reward criteria.

We consider a setting in which each state is associated with a feature vector $\phi : \mathcal{S} \to [0,1]^k$, and the reward for context c is a linear combination of the state features: $R^*_c(s) = f^*(c)^T \phi(s)$. The goal is to approximate $f^*(c)$ using a function $f_W(c)$ with parameters W. This notation allows us to present our algorithms for any function approximator $f_W(c)$, and in particular a deep neural network (DNN).

For the theoretical analysis, we will further assume a *linear setting*, in which the reward function and dynamics are linear in the context. Formally:

$$f^*(c) = c^T W^*, f_W(c) = c^T W,\ W^* \in \mathcal{W}, \text{ and } P_c(s'|s,a) = c^T \begin{bmatrix} P_1(s'|s,a) \\ \vdots \\ P_d(s'|s,a) \end{bmatrix}$$

for some convex set $\mathcal{W}$. In order for the contextual dynamics to be well-defined, we assume the context space is the standard $d-1$ dimensional simplex: $\mathcal{C} = \Delta_{d-1}$. One interpretation of this model is that each row in the mapping W^* along with the corresponding transition kernels defines a base MDP, and the MDP for a specific context is a convex combination of these base environments.

We focus deterministic policies $\pi : \mathcal{S} \to \mathcal{A}$ which dictate the agent's behavior at each state. The value of a policy π for context c is:

$$V^\pi_c = E_{\xi, P_c, \pi}\left[\sum_{t=0}^{\infty} \gamma^t R^*_c(s_t)\right] = f^*(c)^T \mu^\pi_c,$$

where $\mu^\pi_c := E_{\xi, P_c, \pi}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)] \in \mathbb{R}^k$ is called the *feature expectations* of π for context c. For other RL criteria there exist equivalent definitions of feature expectations; see Zahavy et al. (2020b) for the average reward. We also denote by $V^\pi_c(s)$, $\mu^\pi_c(s)$ the value and feature expectations for $\xi = \mathbb{1}_s$. The action-value function, or the Q-function, is defined by: $Q^\pi_c(s,a) = R^*_c(s) + \gamma E_{s'\sim P_c(\cdot|s,a)} V^\pi_c(s')$. For the optimal policy with respect to (w.r.t.) a context c, we denote the above functions by V^*_c, Q^*_c, μ^*_c. For any context c, π^*_c denotes the optimal policy w.r.t. R^*_c, and $\hat{\pi}_c(W)$ denotes the optimal policy w.r.t. $\hat{R}_c(s) = f_W(c)^T \phi(s)$.

For simpler analysis, we define a "flattening" operator, converting a matrix to a vector: $\mathbb{R}^{d\times k} \to \mathbb{R}^{d\cdot k}$ by $\underline{W} = \left[w_{1,1}, \dots, w_{1,k}, \dots, w_{d,1}, \dots, w_{d,k}\right]$. We also define the operator $\odot$ to be the composition of the flattening operator and the outer product: $u \odot v = \left[u_1 v_1, \dots, u_1 v_k, \dots, u_d v_1, \dots, u_d v_k\right]$. Therefore, the value of policy π for context c is given by $V^\pi_c = c^T W^* \mu^\pi_c = \underline{W^*}^T (c \odot \mu^\pi_c)$, where $||c \odot \mu^\pi_c||_1 \le \frac{k}{1-\gamma}$.

2.2 Apprenticeship learning and inverse reinforcement learning

In Apprenticeship Learning (AL), the reward function is unknown, and we denote the MDP without the reward function (also commonly called a controlled Markov chain) by MDP\R. Similarly, we denote a CMDP without a mapping of context to reward by **CMDP\M**.

Instead of manually tweaking the reward to produce the desired behavior, the idea is to observe and mimic an expert. The literature on IRL is quite vast and dates back to (Ng & Russell 2000; Abbeel & Ng 2004). In this setting, the reward function (while unknown to the apprentice) is a linear combination of a set of known features as we defined above. The expert demonstrates a set of trajectories that are used to estimate the feature expectations of its policy π_E, denoted by μ_E. The goal is to find a policy π, whose feature expectations are close to this estimate, and hence will have a similar return with respect to any weight vector w.

Formally, AL is posed as a two-player zero-sum game, where the objective is to find a policy π that does at least as well as the expert with respect to any reward function of the form $r(s) = w \cdot \phi(s), w \in \mathcal{W}$. That is we solve

$$\max_{\pi\in\Pi} \min_{w\in\mathcal{W}} \left[w \cdot \mu(\pi) - w \cdot \mu_E\right] \tag{1}$$

where Π denotes the set of mixed policies (Abbeel & Ng 2004), in which a deterministic policy is sampled according to a distribution at time 0, and executed from that point on. Thus, this policy class can be represented as a convex set of vectors – the distributions over the deterministic policies.

They define the problem of approximately solving Eq. (1) as AL, i.e., finding π such that

$$\forall w \in \mathcal{W} : w \cdot \mu(\pi) \ge w \cdot \mu_E - \epsilon + f^\star. \tag{2}$$

If we denote the value of Eq. (1) by $f^\star$ then, due to the von-Neumann minimax theorem we also have that

$$f^\star = \min_{w\in\mathcal{W}} \max_{\pi\in\Pi} \left[w \cdot \mu(\pi) - w \cdot \mu_E\right]. \tag{3}$$

We will later use this formulation to define the IRL objective, i.e., finding $w \in \mathcal{W}$ such that

$$\forall \pi \in \Pi : w \cdot \mu_E \ge w \cdot \mu(\pi) - \epsilon - f^\star; \tag{4}$$

Abbeel & Ng (2004) suggested two algorithms to solve Eq. (2) for the case that $\mathcal{W}$ is a ball in the Euclidean norm; one that is based on a maximum margin solver and a simpler projection algorithm. The latter starts with an arbitrary policy π_0 and computes its feature expectation μ_0. At step t they define a reward function using weight vector $w_t = \mu_E - \bar{\mu}_{t-1}$

and find the policy π_t that maximizes it. $\bar{\mu}_t$ is a convex combination of feature expectations of previous (deterministic) policies $\bar{\mu}_t = \sum_{j=1}^{t} \alpha_j \mu(\pi_j)$. They show that in order to get that $\|\bar{\mu}_T - \mu\| \le \epsilon$, it suffices to run the algorithm for $T = O(\frac{k}{(1-\gamma)^2\epsilon^2} \log(\frac{k}{(1-\gamma)\epsilon}))$ iterations.

Recently, Zahavy et al. (2020a) showed that the projection algorithm is in fact equivalent to a Frank-Wolfe method for finding the projection of the feature expectations of the expert on the feature expectations polytope – the convex hull of the feature expectations of all the deterministic policies in the MDP. The Frank-Wolfe analysis gives the projection method of Abbeel & Ng (2004) a slightly tighter bound of $T = O(\frac{k}{(1-\gamma)^2\epsilon^2})$. In addition, a variation of the FW method that is based on taking "away steps" (Garber & Hazan 2016; Jaggi 2013) achieves a linear rate of convergence, i.e., it is logarithmic in ϵ.

Another type of algorithms, based on online convex optimization, was proposed by Syed & Schapire (2008). In this approach, in each round the "reward player" plays an online convex optimization algorithm on losses $l_t(w_t) = w_t \cdot (\mu_E - \mu(\pi_t))$; and the "policy player" plays the best response, i.e, the policy π_t that maximizes the return $\mu(\pi_t) \cdot w_t$ at time t. The results in Syed & Schapire (2008) use a specific instance of MDA where the optimization set is the simplex and distances are measured w.r.t $\|\cdot\|_1$. This version of MDA is known as multiplicative weights or Hedge. The algorithm runs for T steps and returns a mixed policy ψ that draws with probability $1/T$ a policy $\pi_t, t = 1, \dots, T$. Thus,

$$\begin{aligned} f^\star &\le \frac{1}{T} \sum_{t=1}^{T} \max_{\pi\in\Pi} \left[w_t \cdot \mu(\pi) - w_t \cdot \mu_E\right] \\ &= \frac{1}{T} \sum_{t=1}^{T} \left[w_t \cdot \mu(\pi_t) - w_t \cdot \mu_E\right] \end{aligned} \tag{5}$$

$$\le \min_{w\in\mathcal{W}} \frac{1}{T} \sum_{t=1}^{T} w \cdot \left[\mu(\pi_t) - \mu_E\right] + O\left(\frac{\sqrt{\log(k)}}{(1-\gamma)\sqrt{T}}\right) \tag{6}$$

$$= \min_{w\in\mathcal{W}} w \cdot (\mu(\psi) - \mu) + O\left(\frac{\sqrt{\log(k)}}{(1-\gamma)\sqrt{T}}\right), \tag{7}$$

where Eq. (5) follows from the fact that the policy player plays the best response, that is, π_t is the optimal policy w.r.t the reward w_t; Eq. (6) follows from the fact that the reward player plays a no-regret algorithm, e.g., online MDA. Thus, they get that $\forall w \in \mathcal{W} : w \cdot \mu(\psi) \ge w \cdot \mu + f^\star - O\left(\frac{1}{\sqrt{T}}\right)$.[1]

2.3 Learned dynamics

Finally, we note that majority of AL papers consider the problem of learning the transition kernel and initial state distribution as an orthogonal 'supervised learning' problem to the AL problem. That is, the algorithm starts by approximating the dynamics from samples and then follows by executing the AL algorithm on the approximated dynamics (Abbeel & Ng 2004; Syed & Schapire 2008). In this paper we adapt this principle. We also note that it is possible to learn a transition kernel and an initial state distribution that are parametrized

[1] The O notation hides the dependency in k and γ .

by the context. Existing methods, such as in Modi et al. (2018), can be used to learn contextual transition kernels. Furthermore, in domains that allow access to the real environment, Abbeel & Ng (2005) provides theoretical bounds for the estimated dynamics of the frequently visited state-action pairs. Thus, we assume P_c is known when discussing suggested methods in Sect. 3, which enables the computation of feature expectations for any context and policy. In Sect. 4.5 we present an example of this principle, where we use a context-dependent model to estimate the dynamics.

3 Methods

In the previous section we have seen AL algorithms for finding a policy that satisfies Eq. (2). In a CMDP this policy will have to be a function of the context, but unfortunately, it is not clear how to analyze contextual policies. Instead, we follow the approach that was taken in the CMDP literature and aim to learn the linear mapping from contexts to rewards (Hallak et al. 2015; Modi et al. 2018; Modi & Tewari 2019). This requires us to design an IRL algorithm instead of an AL algorithm, i.e., to solve Eq. (4) rather than Eq. (2). Concretely, the goal in Contextual IRL is to approximate the mapping $f^*(c)$ by observing an expert (for each context c, the expert provides a demonstration from π_c^*).

This Section is organized as follows. We begin with Sect. 3.1, where we formulate COIRL as a convex optimization problem and derive subgradient descent algorithms for it based on the Mirror Descent Algorithm (MDA). Furthermore, we show that MDA can learn efficiently even when there is only a single expert demonstration per context. This novel approach is designed for COIRL but can be applied to standard IRL problems as well.

In Sect. 3.2 we present a cutting plane method for COIRL that is based on the ellipsoid algorithm. This algorithm requires, in addition to demonstrations, that the expert evaluate the agent's policy and provide its demonstration only if the agent's policy is sub-optimal.

In Sect. 3.3 we discuss how existing IRL algorithms can be adapted to the COIRL setting for domains with finite context spaces and how they compare to COIRL, which we later verify in the experiments section. Finally, in Sect. 3.4 we explore methods for efficient transfer to unseen contexts without additional planning.

3.1 Mirrored descent for COIRL

3.1.1 Problem formulation

In this section, we derive and analyze convex optimization algorithms for COIRL that minimize the following loss function,

$$\text{Loss}(W) = \mathbb{E}_c \max_{\pi} \left[f_W(c) \cdot \left(\mu_c^{\pi} - \mu_c^* \right) \right] = \mathbb{E}_c \left[f_W(c) \cdot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^* \right) \right]. \tag{8}$$

Remark 3.1 We analyze the descent methods for the linear mapping $f(c) = c^T W$. It is possible to extend the analysis to general function classes (parameterized by W), where $\frac{\partial f}{\partial W}$ is computable and f is convex. In this case, $\frac{\partial f}{\partial W}$ aggregates to the descent direction instead of the context, c, and similar sample complexity bounds can be achieved.

The following lemma suggests that if W is a minimizer of Eq. (8), then the expert policy is optimal w.r.t. reward $\hat{R}_c$ for any context.

Lemma 3.1 $\mathrm{Loss}(W)$ *satisfies the following properties*: **(1)** *For any W the loss is greater or equal to zero*. **(2)** *If* $\mathrm{Loss}(W) = 0$ *then for any context, the expert policy is the optimal policy w.r.t. reward $\hat{R}_c(s) = c^T W\phi(s)$.*

Proof We need to show that $\forall W$, $\mathrm{Loss}(W) \geq 0$, and $\mathrm{Loss}(W^*) = 0$. Fix W. For any context c, we have that $\mu_c^{\hat{\pi}_c(W)}$ is the optimal policy w.r.t. reward $f_W(c)$, thus, $f_W(c) \cdot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\right) \geq 0$. Therefore we get that $\mathrm{Loss}(W) \geq 0$. For W^*, we have that $\mu_c^{\hat{\pi}_c(W)} = \mu_c^*$, thus $\mathrm{Loss}(W^*) = 0$.

For the second statement , note that $\mathrm{Loss}(W) = 0$ implies that $\forall c, f_W(c) \cdot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\right) = 0$. This can happen in one of two cases. (1) $\mu_c^{\hat{\pi}_c(W)} = \mu_c^*$, in this case $\pi_c^*, \hat{\pi}_c(W)$ have the same feature expectations. Therefore, they are equivalent in terms of value. (2) $\mu_c^{\hat{\pi}_c(W)} \neq \mu_c^*$, but $f_W(c) \cdot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\right) = 0$. In this case, $\pi_c^*, \hat{\pi}_c(W)$ have different feature expectations, but still achieve the same value w.r.t. reward $f_W(c)$. Since $\hat{\pi}_c(W)$ is an optimal policy w.r.t. this reward, so is π_c^*. □

To evaluate the loss, the optimal policy $\hat{\pi}_c(W)$ and its features expectations $\mu_c^{\hat{\pi}_c(W)}$ must be computed for all contexts. Finding $\hat{\pi}_c(W)$, for a specific context, can be solved using standard RL methods, e.g., value or policy iteration. In addition, computing $\mu_c^{\hat{\pi}_c(W)}$ is equivalent to performing policy evaluation (solving a set of linear equations).

However, since we need to use an algorithm (e.g. policy iteration) to solve for the optimal policy, Eq. (8) is not differentiable w.r.t. W. We therefore consider two optimization schemes that do not involve differentiation: (i) subgradients and (ii) randomly perturbing the loss function (finite differences). Although the loss is non-differentiable, Lemma 3.2 below shows that in the special case that $f_W(c)$ is a linear function, Eq. (8) is convex and Lipschitz continuous. Furthermore, it provides a method to compute its subgradients.

Lemma 3.2 *Let* $f_W(c) = c^T W$ *such that* $\mathrm{Loss}(W)$, *denoted by* $L_{\text{lin}}(W)$, *is given by*

$$L_{\text{lin}}(W) = \mathbb{E}_c\left[c^T W \cdot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\right)\right].$$

We have that:

1. $L_{\text{lin}}(W)$ *is a convex function.*
2. $g(W) = \mathbb{E}_c\left[c \odot \left(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\right)\right]$ *is a subgradient of* $L_{\text{lin}}(W)$.
3. L_{lin} *is a Lipschitz continuous function, with Lipschitz constant* $L = \frac{2}{1-\gamma}$ *w.r.t.* $\|\cdot\|_\infty$ *and* $L = \frac{2\sqrt{dk}}{1-\gamma}$ *w.r.t.* $\|\cdot\|_2$.

In the supplementary material we provide the proof for the Lemma (Appendix A). The proof follows the definitions of convexity and subgradients, using the fact that for each W we compute the optimal policy for reward $c^T W$. The Lipschitz continuity of $L_{\text{Lin}}(W)$ is related to the simulation lemma (Kearns & Singh 2002), that is, a small change in the reward results in a small change in the optimal value.

Note that $g(W) \in \mathbb{R}^{d\times k}$ is a matrix; we will sometimes refer to it as a matrix and sometimes as a flattened vector, depending on the context. Finally, $g(W)$ is given in expectation over contexts, and in expectation over trajectories (feature expectations). We will later see how to replace $g(W)$ with an unbiased estimate, which can be computed by aggregating state features from a single expert trajectory sample.

3.1.2 Algorithms

Lemma 3.2 identifies $L_{\text{Lin}}(W)$ as a convex function and provides a method to compute its subgradients. A standard method for minimizing a convex function over a convex set is the subgradient projection algorithm (Bertsekas 1997). The algorithm is given by the following iterates:

$$W_{t+1} = \text{Proj}_{\mathcal{W}}\{W_t - \alpha_t g(W_t)\},$$

where $f(W_t)$ is a convex function, $g(W_t)$ is a subgradient of $f(W_t)$, and α_t the learning rate. $\mathcal{W}$ is required to be a convex set; we will consider two particular cases, the ℓ_2 ball (Abbeel & Ng 2004) and the simplex (Syed & Schapire 2008).[2]

Next, we consider a generalization of the subgradient projection algorithm that is called the mirror descent algorithm (Nemirovsky & Yudin 1983, MDA):

$$W_{t+1} = \arg\min_{W\in\mathcal{W}} \left\{ W \cdot \nabla_f(W_t) + \frac{1}{\alpha_t} D_\psi(W, W_t) \right\}, \tag{9}$$

where $D_\psi(W, W_t)$ is a Bregman distance,[3] associated with a strongly convex function ψ. The following theorem characterizes the convergence rate of MDA.

Theorem 3.1 (Convergence rate of MDA) *Let ψ be a σ-strongly convex function on $\mathcal{W}$ w.r.t. $\|\cdot\|$, and let $D^2 = \sup_{W_1,W_2\in\mathcal{W}} D_\psi(W_1, W_2)$. Let f be convex and L-Lipschitz continuous w.r.t. $\|\cdot\|$. Then, MDA with $\alpha_t = \frac{D}{L}\sqrt{\frac{2\sigma}{t}}$ satisfies:*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} W_t\right) - f(W^*) \le DL\sqrt{\frac{2}{\sigma T}}.$$

We refer the reader to Beck & Teboulle (2003) and Bubeck (2015) for the proof. Specific instances of MDA require one to choose a norm and to define the function ψ. Once those are defined, one can compute σ, D and L which define the learning rate schedule. Below, we provide two MDA instances (see, for example Beck & Teboulle (2003) for derivation) and analyze them for COIRL.

Projected subgradient descent (PSGD): Let $\mathcal{W}$ be an ℓ_2 ball with radius 1. Fix $||\cdot||_2$, and $\psi(W) = \frac{1}{2}||W||_2^2$. ψ is strongly convex w.r.t. $||\cdot||_2$ with $\sigma = 1$. The associated Bregman divergence is given by $D_\psi(W_1, W_2) = 0.5||W_1 - W_2||_2^2$. Thus, mirror descent is equivalent to PSGD. $D^2 = \max_{W_1,W_2\in\mathcal{W}} D_\psi(W_1, W_2) \le 1$, and according to Lemma 3.2, $L = \frac{2\sqrt{dk}}{1-\gamma}$. Thus, we have that the learning rate is $\alpha_t = (1-\gamma)\sqrt{\frac{1}{2dkt}}$ and the update to W is given by

[2] Scaling of the reward by a constant does not affect the resulting policy, thus, these sets are not restricting.

[3] We refer the reader to the supplementary material (Appendix A) for definitions of the Bregman distance, the dual norm, etc.

$$\tilde{W} = W_t - \alpha_t g_t, W_{t+1} = \tilde{W}/||\tilde{W}||_2,$$

and according to Theorem 3.1 we have that after T iterations,

$$L_{\text{lin}}\left(\frac{1}{T}\sum_{t=1}^{T} W_t\right) - L_{\text{lin}}(W^*) \le \mathcal{O}\left(\frac{\sqrt{dk}}{(1-\gamma)\sqrt{T}}\right).$$

Exponential Weights (EW): Let $\mathcal{W}$ be the standard $dk-1$ dimensional simplex. Let $\psi(W) = \sum_i W(i)\log(W(i))$. ψ is strongly convex w.r.t. $||\cdot||_1$ with $\sigma = 1$. We get that the associated Bregman divergence is given by

$$D_\psi\left(W_1, W_2\right) = \sum_i W_1(i)\log\left(\frac{W_1(i)}{W_2(i)}\right),$$

also known as the Kullback-Leibler divergence. In addition,

$$D^2 = \max_{W_1, W_2 \in \mathcal{W}} D_\psi\left(W_1, W_2\right) \le \log(dk)$$

and according to Lemma 3.2, $L = \frac{2}{1-\gamma}$. Thus, we have that the learning rate is $\alpha_t = (1-\gamma)\sqrt{\frac{\log(dk)}{2t}}$. Furthermore, the projection onto the simplex w.r.t. to this distance amounts to a simple renormalization $W \leftarrow W/||W||_1$. Thus, we get that MDA is equivalent to the exponential weights algorithm and the update to w is given by

$$\forall i \in [1..dk], \tilde{W}(i) = W_t(i)\exp\left(-\alpha_t g_t(i)\right), W_{t+1} = \tilde{W}/||\tilde{W}||_1.$$

Finally, according to Theorem 3.1 we have that after T iterations,

$$L_{\text{lin}}\left(\frac{1}{T}\sum_{t=1}^{T} W_t\right) - L_{\text{lin}}(W^*) \le \mathcal{O}\left(\frac{\sqrt{\log(dk)}}{(1-\gamma)\sqrt{T}}\right).$$

Algorithm 1 MDA for COIRL

```
Input: a norm $||\cdot||$, a strongly convex function $\psi$ with strong convexity parameter $\sigma$ w.r.t.to the norm, a convex set $\mathcal{W}$ with diameter $D$ w.r.t. the norm, $L$ a Lipschitz constant, $T$ number of iterations
Initialize $W_1 \in \mathcal{W}$
for $t = 1, \dots, T$ do
    Observe $c, \mu_c^*$
    Compute $\hat{\pi}_c(W_t), \mu_c^{\hat{\pi}_c(W_t)}$
    Compute a subgradient $g_t = c \odot \left(\mu_c^{\hat{\pi}_c(W_t)} - \mu_c^*\right)$
    Set learning rate $\alpha_t = \frac{D}{L}\sqrt{\frac{2\sigma}{t}}$
    Update: $W_{t+1} = \arg\min_{W \in \mathcal{W}} \left\{W \cdot g_t + \frac{1}{\alpha_t} D_\psi(W, W_t)\right\}$
end for
return $\frac{1}{t}\sum_{t=1}^{T} W_t$
```

Evolution strategies for COIRL: Next, we consider a derivative-free algorithm for computing subgradients, based on *Evolution Strategies* (Salimans et al. 2017, ES). For convex optimization problems, ES is a gradient-free descent method based on computing finite differences (Nesterov & Spokoiny 2017). The subgradient in ES is computed by sampling m random perturbations and computing the loss for them, in the following form

```
For $j = 1, ..., m$ do
    Sample $u_j \sim \mathcal{N}(0, \rho^2) \in \mathcal{R}^{dk}$,
    $g^j = \text{Loss}\left(W_t + \frac{\nu u_j}{||u_j||}\right)\frac{\nu u_j}{||u_j||}$,
End For,
```

and the subgradient is given by

$$g_t = \frac{1}{m\rho}\sum_{j=1}^{m} g^j. \tag{10}$$

Theorem 3.2 presents the sample complexity of PSGD with the subgradient in Eq. (10) for the case that the loss is convex, as in L_{Lin}. While this method has looser upper-bound guarantees compared to MDA (Theorem 3.1), Nesterov & Spokoiny (2017) observed that in practice, it often outperforms subgradient-based methods. Thus, we test ES empirically and compare it with the subgradient method (Sect. 3.1). Additionally, Salimans et al. (2017) have shown the ability of ES to cope with high dimensional non-convex tasks (DNNs).

Theorem 3.2 (ES Convergence Rate (Nesterov & Spokoiny 2017)) *Let $L_{\text{lin}}(W)$ be a non-smooth convex function with Lipschitz constant L, such that $||W_0 - W^*|| \le D$, step size of $\alpha_t = \frac{D}{(dk+4)\sqrt{T+1}L}$ and $\nu \le \frac{\epsilon}{2L\sqrt{dk}}$ then in $T = \frac{4(dk+4)^2D^2L^2}{\epsilon^2}$ ES finds a solution which is bounded by $\mathbb{E}_{U_{T-1}}[L_{\text{lin}}(\hat{W}_T)] - L_{\text{lin}}(W^*) \le \epsilon$, where $U_T = \{u_0, \dots, u_T\}$ denotes the random variables of the algorithm up to time T and $\hat{W}_T = \arg\min_{t=1,\dots,T} L_{\text{lin}}(W_t)$.*

Practical MDA: One of the "miracles" of MDA is its robustness to noise. If we replace g_t with an unbiased estimate $\tilde{g}_t$, such that $\mathbb{E}\tilde{g}_t = g_t$ and $\mathbb{E}\|\tilde{g}_t\| \le L$, we obtain the same convergence results as in Theorem 3.1 (Robbins & Monro 1951) (see, for example, Bubeck 2015, Theorem 6.1). Such an unbiased estimate can be obtained in the following manner: (i) sample a context c_t, (ii) compute $\mu_{c_t}^{\pi^*_{c_t}(W_t)}$, (iii) observe a single expert demonstration $\tau_i^E = \{s_0^i, a_0, s_1^i, a_1, \dots, \}$, where a_i is chosen by the expert policy $\pi^*_{c_t}$ (iv) let $\hat{\mu}_i = \sum_{t\in[0,\dots,|\tau_i^E|-1]} \gamma^t \phi(s_t^i)$ be the accumulated discounted features across the trajectory such that $\mathbb{E}\hat{\mu}_i = \mu^*_{c_t}$.

However, for $\hat{\mu}_i$ to be an unbiased estimate of $\mu^*_{c_t}$, τ_i^E needs to be of infinite length. Thus one can either (1) execute the expert trajectory online, and terminate it at each time step with probability $1-\gamma$ (Kakade & Langford 2002), or (2) execute a trajectory of length $H = \frac{1}{1-\gamma}\log(1/\epsilon_H)$. The issue with the first approach is that since the trajectory length is unbounded, the estimate $\hat{\mu}_i$ cannot be shown to concentrate to $\mu^*_{c_t}$ via Hoeffding type inequalities. Nevertheless, it is possible to obtain a concentration inequality using the fact that the length of each trajectory is bounded in high probability (similar to Zahavy et al. (2020b)). The second approach can only guarantee that $\|g_t - \mathbb{E}\tilde{g}_t\| \le \epsilon_H$ (Syed & Schapire 2008). Hence, using the robustness of MDA to adversarial noise (Zinkevich 2003), we get that MDA converges with an additional error of ϵ_H, i.e.,

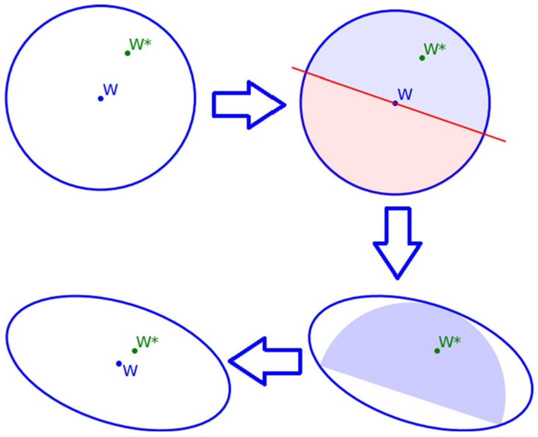


Fig. 3 The ellipsoid algorithm proceeds in an iterative way, using linear constraints to gradually reduce the size of the ellipsoid until the center defines an ϵ-optimal solution

$$L_{\text{lin}}\left(\frac{1}{T}\sum_{t=1}^{T} W_t\right) - L_{\text{lin}}(W^*) \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \epsilon_H.$$

While this sampling mechanism has the cost of a controlled bias, usually it is more practical, in particular, if the trajectories are given as a set of demonstrations (offline data).

3.2 Ellipsoid algorithms for COIRL

In this section we present the ellipsoid method, introduced to the IRL setting by Amin et al. (2017). We extend this method to the contextual setting, and focus on finding a linear mapping $W \in \mathcal{W}$ where $\mathcal{W} = \{W : ||W||_\infty \leq 1\}$, and $W^* \in \mathcal{W}$. The algorithm, illustrated in Fig. 3, maintains an ellipsoid-shaped feasibility set for W^*. In each iteration, the algorithm receives a demonstration which is used to create a linear constraint, halving the feasibility set. The remaining half-ellipsoid, still containing W^*, is then encapsulated by a new ellipsoid. With every iteration, this feasibility set is reduced until it converges to W^*.

Formally, an ellipsoid is defined by its center – a vector u, and by an invertible matrix Q: $\{x : (x-u)Q^{-1}(x-u) \leq 1\}$. The feasibility set for W^* is initialized to be the minimal sphere containing $\{W : ||W||_\infty \leq 1\}$. At every step t, the current estimation W_t of W^* is defined as the center of the feasibility set, and the agent acts optimally w.r.t. the reward function $\hat{R}_c(s) = c^T W_t \phi(s)$. If the agent performs sub-optimally, the expert provides a demonstration in the form of its feature expectations for c_t: $\mu^*_{c_t}$. These feature expectations are used to generate a linear constraint (hyperplane) on the ellipsoid that is crossing its center. Under this constraint, we construct a new feasibility set that is half of the previous ellipsoid, and still contains W^*. For the algorithm to proceed, we compute a new ellipsoid that is the minimum volume enclosing ellipsoid (MVEE) around this "half-ellipsoid". These updates are guaranteed to gradually reduce the volume of the ellipsoid, as shown in Lemma 3.3, until its center is a mapping which induces ϵ-optimal policies for all contexts.

Lemma 3.3 (Boyd & Barratt (1991)) *If* $B \subseteq \mathbb{R}^D$ *is an ellipsoid with center* w, *and* $x \in \mathbb{R}^D \setminus \{0\}$, *we define* $B^+ = \text{MVEE}(\{\theta \in B : (\theta - w)^T x \geq 0\})$, *then*: $\frac{Vol(B^+)}{Vol(B)} \leq e^{-\frac{1}{2(D+1)}}$.

Theorem 3.3 below shows that this algorithm achieves a polynomial upper bound on the number of sub-optimal time-steps. The proof, found in Appendix B, is adapted from (Amin et al. 2017) to the contextual setup.

Algorithm 2 Ellipsoid algorithm for COIRL

```
Initialize: $\Theta_0 \leftarrow B_\infty(0,1) = \{x \in \mathbb{R}^{d\cdot k} : ||x||_\infty \le 1\}$
$\Theta_1 \leftarrow \text{MVEE}(\Theta_0) : \underline{W}_1 = 0, Q_1 = dkI$
for $t = 1, 2, \ldots$ do
    Observe $c_t$, let $\underline{W}_t$ be the center of $\Theta_t$
    Play episode using $\hat{\pi}_t = \arg\max_\pi c_t^T W_t \mu_{c_t}^\pi$
    if $V_{c_t}^* - V_{c_t}^{\hat{\pi}_t} > \epsilon$ then
        $\mu_{c_t}^*$ is revealed
        Let $a_t = c_t \odot \left(\mu_{c_t}^* - \mu_{c_t}^{\hat{\pi}_t}\right)$
        $\Theta_{t+1} \leftarrow \text{MVEE}\left(\{\theta \in \Theta_t : \theta^T a_t \ge \underline{W}_t^T a_t\}\right)$
    else
        $\Theta_{t+1} \leftarrow \Theta_t$
    end if
end for

MVEE($\{\theta \in \Theta_t : \theta^T a_t \ge \underline{W}_t^T a_t\}$):
    $\tilde{a}_t = \frac{-1}{\sqrt{a_t^T Q_t a_t}} a_t$
    $\underline{W}_{t+1} = \underline{W}_t - \frac{1}{dk+1} Q_t \tilde{a}_t$
    $Q_{t+1} = \frac{d^2k^2}{d^2k^2-1}(Q_t - \frac{2}{dk+1} Q_t \tilde{a}_t \tilde{a}_t^T Q_t)$
```

Theorem 3.3 *In the linear setting where $R_c^*(s) = c^T W^* \phi(s)$, for an agent acting according to Algorithm 1, the number of rounds in which the agent is not ϵ-optimal is $\mathcal{O}(d^2k^2 \log(\frac{dk}{(1-\gamma)\epsilon}))$.*

Remark 3.2 Note that the ellipsoid method presents a new learning framework, where demonstrations are only provided when the agent performs sub-optimally. Thus, the theoretical results in this section cannot be directly compared with those of the descent methods. We further discuss this in Appendix D.2.1.

Remark 3.3 The ellipsoid method does not require a distribution over contexts - an adversary may choose them. MDA can also be easily extended to the adversarial setting via known regret bounds on online MDA (Hazan 2016).

3.2.1 Practical ellipsoid algorithm

In real-world scenarios, it may be impossible for the expert to evaluate the value of the agent's policy, i.e. check if $V_{c_t}^* - V_{c_t}^{\hat{\pi}_t} > \epsilon$, and to provide its policy or feature expectations $\mu_{c_t}^*$. To address these issues, we follow Amin et al. (2017) and consider a relaxed approach, in which the expert evaluates each of the individual actions performed by the agent rather than its policy (Algorithm 3). When a sub-optimal action is chosen, the expert provides finite roll-outs instead of its policy or feature expectations. We define the expert criterion for providing a demonstration to be $Q^*_{c_t}(s,a) + \epsilon < V^*_{c_t}(s)$ for each state-action pair (s, a) in the agent's trajectory.

Algorithm 3 Batch ellipsoid algorithm for COIRL

```
Initialize: $\Theta_0 \leftarrow B_\infty(0,1) = \{x \in \mathbb{R}^{d\cdot k} : ||x||_\infty \le 1\}$
$\Theta_1 \leftarrow \text{MVEE}(\Theta_0)$
$i \leftarrow 0, \bar{Z} \leftarrow 0, \bar{Z}^* \leftarrow 0$
for $t = 1, 2, 3, ...$ do
    $c_t$ is revealed, Let $\underline{W}_t$ be the center of $\Theta_t$
    Play episode using $\hat{\pi}_t = \arg\max_\pi c_t^T W_t \mu^\pi_{c_t}$
    $\Theta_{t+1} \leftarrow \Theta_t$
    if a sub-optimal action $a$ is played at state $s$ then
        Expert provides H-step trajectory $(s^E_0 = s, s^E_1, ..., s^E_H)$. Let $\hat{x}^{*,H}_i$ be the H-step sample of the expert's feature expectations for $\xi'_i = \mathbb{1}_s$: $\hat{x}^{*,H}_i = \sum_{h=0}^H \gamma^h \phi(s^E_h)$
        Let $x_i$ be the agent's feature expectations for $\xi'_i$ : $E_{\xi'_i, P_{c_t}, \pi_t}[\sum_{h=0}^\infty \gamma^h \phi(s_h)]$
        Denote $z_i = c_t \odot x_i$, $\hat{z}^{*,H}_i = c_t \odot \hat{x}^{*,H}_i$
        $i \leftarrow i+1, \bar{Z} \leftarrow \bar{Z} + z_i, \bar{Z}^* \leftarrow \bar{Z}^* + \hat{z}^{*,H}_i$
        if $i = n$ then
            $\Theta_{t+1} \leftarrow \text{MVEE}\Big(\Big\{\theta \in \Theta_t : \big(\theta - \underline{W}_t\big)^T \cdot (\frac{\bar{Z}^*}{n} - \frac{\bar{Z}}{n}) \ge 0\Big\}\Big)$
            $i \leftarrow 0, \bar{Z} \leftarrow 0, \bar{Z}^* \leftarrow 0$
        end if
    end if
end for
```

Near-optimal experts: In addition, we relax the optimality requirement of the expert and instead assume that, for each context c_t, the expert acts optimally w.r.t. W^*_t which is close to W^*; the expert also evaluates the agent w.r.t. this mapping. This allows the agent to learn from different experts, and from non-stationary experts whose judgment and performance slightly vary over time. If a sub-optimal action w.r.t. W^*_t is played at state s, the expert provides a roll-out of H steps from s to the agent. As this roll-out is a sample of the optimal policy w.r.t. W^*_t, we aggregate n examples to assure that with high probability, the linear constraint that we use in the ellipsoid algorithm does not exclude W^* from the feasibility set. Note that these batches may be constructed across different contexts, different experts, and different states from which the demonstrations start. Theorem 3.4, proven in Appendix B, upper bounds the number of sub-optimal actions that Algorithm 3 chooses.[4]

Theorem 3.4 *For an agent acting according to Algorithm* 3, $H = \lceil \frac{1}{1-\gamma} \log(\frac{8k}{(1-\gamma)\epsilon}) \rceil$ *and* $n = \lceil \frac{512k^2}{(1-\gamma)^2\epsilon^2} \log(4dk(dk+1)\log(\frac{16k\sqrt{dk}}{(1-\gamma)\epsilon})/\delta) \rceil$, *with probability of at least* $1-\delta$, *if* $\forall t : \underline{W}^*_t \in B_\infty(\underline{W}^*, \frac{(1-\gamma)\epsilon}{8k}) \cap \Theta_0$ *the number of rounds in which a sub-optimal action is played is* $\mathcal{O}\Big(\frac{d^2k^4}{(1-\gamma)^2\epsilon^2} \log\big(\frac{dk}{(1-\gamma)\delta\epsilon} \log(\frac{dk}{(1-\gamma)\epsilon})\big)\Big)$.

The theoretical guarantees of the algorithms presented so far are summarized in Table 1. We can see that MDA, in particular EW, achieves the best scalability. In the unrealistic case

[4] MDA also works with near optimal experts due to the robustness of MDA. The analysis of this case is identical to the analysis of biased trajectories, as we discuss in the end of Sect. 3.1.

Table 1 Summary of theoretical guarantees

		Scalability		Sample complexity		Extension to DNNs
		Feature expectations	Sampled trajectory	Feature expectations	Sampled trajectory	
MDA	PSGD	$\mathcal{O}(dk)$		$\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$	✓
	EW	$\mathcal{O}(\log dk)$				✗
ES		$\mathcal{O}(dk)$	$\mathcal{O}(d^2k^2)$			✓
Ellipsoid		$\mathcal{O}(d^2k^2)$	$\mathcal{O}(d^2k^4)$	$\mathcal{O}\left(\log \frac{1}{\epsilon}\right)$		✗

where the expert can provide its feature expectations, the ellipsoid method has the lowest sample complexity. However, in the realistic scenario where only samples are provided, the sample complexity is identical across all methods. We also note that unlike MDA and ES, it isn't possible to extend the ellipsoid method to work with DNNs. Overall, the theoretical guarantees favor the MDA methods when it comes to the realistic setting.

3.3 Existing approaches

We focus our comparisons to methods that can be used for zero-shot generalization across contexts or tasks. Hence, we omit discussion of "meta inverse reinforcement learning" methods which focus on few-shot generalization (Xu et al. 2018). Our focus is on two approaches: (1) standard IRL methods applied to a model which incorporates the context as part of the state, and (2) contextual policies through behavioral cloning (BC) (Pomerleau 1989).

3.3.1 Application of IRL to COIRL problems

We first examine the straight-forward approach of incorporating the contextual information into the state, i.e., defining $\mathcal{S}' = \mathcal{C} \times \mathcal{S}$, and applying standard IRL methods to one environment which captures all contexts. This construction limits the context space to a finite one, as opposed to COIRL which works trivially with an infinite number of contexts. At first glance, this method results in the same scalability and sample complexity as COIRL; however, when considering the inner loop in which an optimal policy is calculated, COIRL has the advantage of a smaller state space by a factor of $|\mathcal{C}|$. This results in significantly better run-time when considering large context spaces. In Sect. 4.1, we present experiments that evaluate the run-time of this approach, compared to COIRL, for increasingly large context spaces. These results demonstrate that the run-time of IRL scales with $|\mathcal{C}|$ while the run-time of COIRL is unaffected by $|\mathcal{C}|$, making COIRL much more practical for environments with many or infinite contexts.

3.3.2 Contextual policies

Another possible approach is to use Behavioral Cloning (BC) to learn contextual policies, i.e., policies that are functions of both state and context $\pi(c, s)$. In BC, the policy is learned using supervised learning methods, skipping the step of learning the reward function. While BC is an intuitive method, with successful applications in various domains

(Bojarski et al. 2016; Ratliff et al. 2007), it has a fundamental flaw; BC violates the i.i.d. assumptions of supervised learning methods, as the learned policy affects the distribution of states it encounters. This results in a covariate shift in test-time leading to compounding errors (Ross & Bagnell 2010; Ross et al. 2011). Methods presented in Ross et al. (2011); Laskey et al. (2017) mitigate this issue but operate outside of the offline framework. This explains why BC compares unfavorably to IRL methods, especially with a limited number of available demonstrations (Ho & Ermon 2016; Ghasemipour et al. 2019). In Sect. 4.4.2, we provide experimental results that exhibit the same trend. These results demonstrate how matching actions on the train set poorly translates to value on the test set, until much of the expert policy is observed. While a single trajectory per context suffices for COIRL, BC requires more information to avoid encountering unfamiliar states. We also provide a hardness result for learning a contextual policy for a linear separator hypothesis class, further demonstrating the challenges of this approach.

3.4 Transfer across contexts in test-time

In this section, we examine the application of the learned mapping W when encountering a new, unseen context in test-time. Unlike during training, in test-time the available resources and latency requirements may render re-solving the MDP for every new context infeasible. We address this issue by leveraging optimal policies $\{\pi^*_{c_j}\}_{j=1}^N$ for contexts $\{c_j\}_{j=1}^N$ which were previously calculated during training or test time. We separately handle context-independent dynamics and contextual dynamics by utilizing (1) generalized policy improvement (GPI) (Barreto et al. 2017), and (2) the simulation lemma (Kearns & Singh 2002), respectively.

For context-independent dynamics, the framework of Barreto et al. (2017) can be applied to efficiently transfer knowledge from previously observed contexts $\{c_j\}_{j=1}^N$ to a new context c. As the policies $\{\pi^*_{c_j}\}_{j=1}^N$ were computed, so were their feature expectations, starting from any state. As the dynamics are context-independent, these feature expectations are also valid for c, enabling fast computation of the corresponding Q-functions, thanks to the linear decomposition of the reward. GPI generalizes policy improvement, allowing us to use these Q-functions to create a new policy that is as good as any of them and potentially strictly better than them all. The following theorem, a parallel of Theorem 2 in Barreto et al. (2017), defines the GPI calculation and provides the lower bound on its value. While these theorems and their proofs are written for W^*, the results hold for any $W \in \mathcal{W}$.

Theorem 3.5 (Barreto et al. (2017)) *Let* $\phi_{max} = \max_s ||W^*\phi(s)||_1$, $\{c_j\}_{j=1}^N \subseteq \mathcal{C}$, $c \in \mathcal{C}$, *and* $\pi(s) \in \arg\max_a \max_j Q_c^{\pi^*_{c_j}}(s,a)$. *If the dynamics are context independent, then*:

$$V_c^* - V_c^\pi \leq 2\frac{\phi_{max}}{1-\gamma} \min_j ||c - c_j||_\infty .$$

When the dynamics are a function of the context, the feature expectations calculated for $\{c_j\}_{j=1}^N$ are not valid for c, thus GPI can not be used efficiently. However, due to the linearity and therefore continuity of the mapping, similar contexts induce similar environments. Thus, it is intuitive that if we know the optimal policy for a context, it should transfer well to nearby contexts without additional planning. This intuition is formalized in the simulation lemma, which is used to provide bounds on the performance of a transferred policy in the following theorem.

Theorem 3.6 *Let* $c, c_j \in \mathcal{C}$, $\phi_{max} = \max_s ||W^* \phi(s)||_1$, $V_{max} = \max_{c,s} |V_c^*(s)|$. *Then*:

$$V_c^* - V_c^{\pi_{c_j}^*} \le 2 \frac{\phi_{max} + \gamma d V_{max}}{\gamma(1-\gamma)} ||c - c_j||_\infty .$$

Remark 3.4 The bound depends on $\mathcal{W}$. For example, for $\mathcal{W} = \Delta_{dk-1}$, the bound is $2\frac{1-\gamma+\gamma d}{\gamma(1-\gamma)^2}||c - c_j||_\infty$, and for $\mathcal{W} = B_\infty(0,1)$ the bound is $\frac{2dk}{\gamma(1-\gamma)^2}||c - c_j||_\infty$.

Remark 3.5 If the dynamics are independent of the context, the term $\gamma d V_{max}$ is omitted from the bound.

Using these methods, one can efficiently find a good policy for a new context c, either as a good starting point for policy/value iteration which will converge faster or as the final policy to be used in test-time. The last thing to consider is the construction of the set $\{c_j\}_{j=1}^N$. As COIRL requires computing the optimal policies for W during training, the training contexts are a natural set to use. In addition, as suggested in Barreto et al. (2017), we may reduce this set or enhance it in a way that maintains a covering radius in $\mathcal{C}$ and guarantees a desired level of performance. If the above methods are used as initializations for calculating the optimal policy, the set can be updated in test-time as well.

4 Experiments

In the previous sections we described the theoretical COIRL problem, proposed methods to solve it and analyzed them. In this section our goal is to take COIRL from theory to practice. This section presents the process and the guidelines we follow to achieve this goal in a step-by-step manner, to bridge the gap between theoretical and real-life problems through a set of experiments.[5]

We begin by focusing on the *grid world* and *autonomous driving simulation* environments. As these are relatively small domains, for which we can easily compute the optimal policy, they provide easily accessible insight into the behavior of each method and allow us to eliminate methods that are less likely to work well in practice. Then we use the *sepsis treatment simulator* in a series of experiments to test and adjust the methods towards real-life application. The simulator is constructed from real-world data in accordance with the theoretical assumptions of COIRL. Throughout the experiments we strip the assumptions from the simulator and show that the methods perform well in an offline setting. Furthermore, we show that a DNN estimator achieves high performance when the mapping from the context to the reward is non-linear.

Finally, we test the methods in *sepsis treatment* – without the simulator. Here, we use real clinicians' trajectories for training and testing. For COIRL, we estimate a CMDP\ M model from the train data (states and dynamics) which is used for training purposes. We then show that COIRL achieves high action matching on unseen clinicians trajectories.

[5] The code used in these experiments is provided in following repository https://github.com/coirl/coirl_code.

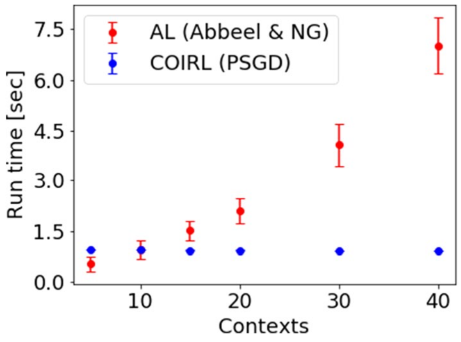


Fig. 4 Run-time comparison between COIRL and AL. AL run-time grows as the number of contexts grows while COIRL run-time stays fixed

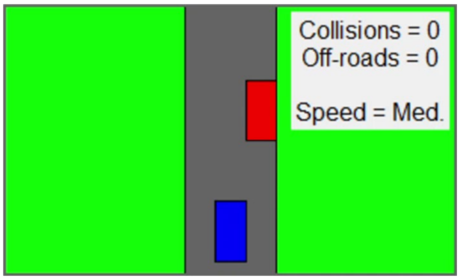


Fig. 5 An illustration of the driving simulator

4.1 Grid world

The grid world domain is an n by m grid which makes $|\mathcal{S}| = n \cdot m$ states. The actions are $\mathcal{A} = \{left, up, right, down\}$ and the dynamics are deterministic for each action, i.e., if the action taken is up, the next state will be the state above the current state in the grid (with cyclic transitions on the borders, i.e., taking the action *right* at state $(n-1, y)$ will transition to $(0, y)$). The features are one-hot vectors ($\phi(s_i) = e_i \in \mathbb{R}^{n \cdot m}$). The contexts correspond to "preferences" of certain states on the grid. The contexts are sampled from a uniform distribution over the $n \cdot m$ dimensional simplex.

This domain is used to evaluate the application of IRL to COIRL problems. We compare the performance of PSGD (COIRL) and the projection algorithm (AL) of Abbeel & Ng (2004) as a function of the context space size. This framework is applied on a grid with dimensions of $3 \cdot 4$, overall 12 states. The PSGD method trains on a CMDP model and the projection algorithm trains on a large MDP model, with a state space that includes the contexts, as noted in Sect. 3.3.1. The new states are $s' = (s, c)$, and the new features are $\phi(s') = c \odot \phi(s)$. We measure the run-time of every iteration. The most time consuming part of both methods is the optimal policy computation time for a given reward. Both methods use the same implementation of value iteration in order to enable a comparison of the run-time.

The results shown in Fig. 4 show that the projection algorithm in the large MDP requires significantly more time to run as the number of contexts grows, while the run-time of PSGD is not affected by the number of contexts.

4.2 Conclusion

Applying IRL methods in a large MDP environment limits the number of contexts that can be used, and as seen in the results, its run time grows when the number of contexts increases. We conclude that applying IRL to COIRL problems is inefficient and exclude this method from the following experiments (Sect. 4.2 through Sect. 4.4).

4.3 Autonomous driving simulation

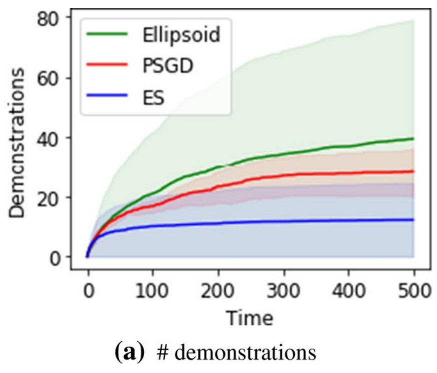
While the grid world focused on comparing COIRL with the standard IRL method, in this section we compare the various methods for performing COIRL in an autonomous driving simulator (Fig. 5). This domain involves a three-lane highway with two visible cars, cars A and B. The agent, controlling car A, can drive both on the highway and off-road. Car B drives in a fixed lane, at a slower speed than car A. Upon leaving the frame, car B is replaced by a new car, appearing in a random lane at the top of the screen. The features denote the speed of car A, whether or not it is on the road and whether it has collided with car B. The context implies different priorities for the agent; should it prefer speed or safety? Is going off-road a valid option? For example, an ambulance will prioritize speed and may drive off-road, as long as it goes fast and avoids collisions, while a bus will prioritize avoiding both collisions and off-road driving as safety is its primary concern. The mapping from the contexts to the true reward is constructed in a way that induces different behaviors for different contexts, making generalization a challenging task.

4.3.1 Ellipsoid setting

The ellipsoid method requires its own framework. Here, the agent's policy is evaluated by an expert for every new context revealed. Only if its value is not ϵ-close to the optimal policy value, an expert demonstration will be provided (feature expectations of an expert for the revealed context). While the ellipsoid method can only perform a single update for each demonstration, the descent methods can utilize all of the previously revealed demonstrations and perform update steps until convergence. We measure the accumulated amount of expert demonstrations given at each time-step and the value of the agent on a holdout test set, for each new given demonstration.

The amount of given demonstrations is important in the ellipsoid framework, as it is equal to the number of times that the agent is not ϵ-close to the optimal policy value. In addition, it is a way to measure how much intervention is required by an external expert. We would expect a 'good' method to be ϵ-optimal for most revealed contexts and therefore it should observe a small amount of demonstrations.

The results, presented in Fig. 6, show that all methods eventually reach the expert's value; however, the descent methods are more sample efficient than the ellipsoid method and require fewer expert demonstrations. While according to the theoretical guarantees (Table 1, feature expectations setting) the ellipsoid method should have better sample complexity, in practice it is surpassed by the results of the descent methods. Note that in this experiment each demonstration may be used more than once by the descent methods, hence the theoretical results are not valid for them.



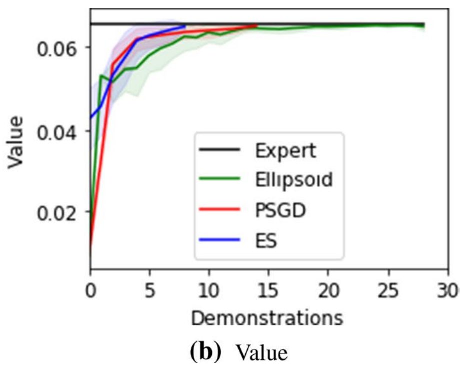
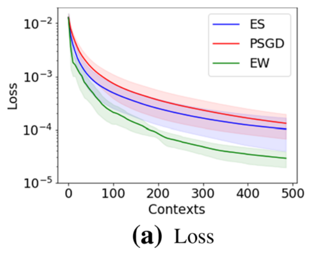
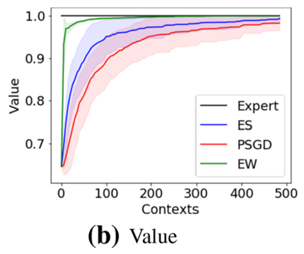


Fig. 6 Comparison of the ellipsoid method with the ES and PSGD methods in the autonomous driving simulation. The graph on the left compares the number of demonstrations required by each method, while the graph on the right compares the performance at each time-step. We observe that while, as theoretically shown, all methods eventually find an ϵ-optimal solution, the descent methods attain better sample efficiency (converge faster and require less expert interaction)





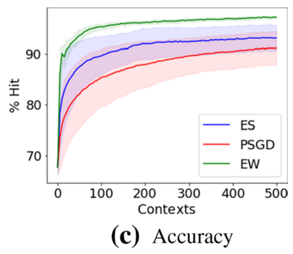


Fig. 7 Online learning curve in the autonomous driving simulation—learning from feature expectations. The expert demonstrations are provided in the form of the feature expectations of the expert's policy. We compare the loss, value and accuracy, where the value and accuracy are relative to the expert's behavior. As can be seen, all descent methods minimize the loss and achieve high value. Additionally, we observe that while they do attain relatively high accuracy, they find policies which are optimal yet differ from the expert in the actions taken

4.3.2 Online setting

Here, we compare the descent methods presented in Sect. 3 in an online setting. Each descent step is performed on a context-μ pair, where the context is sampled uniformly from the simplex and μ is the feature expectations of a policy that is optimal for this context. For each method, we measure the normalized value of the proposed policies with respect to the real reward, the loss (Eq. (8)), and the accuracy, which represents how often the expert and agent policies match. These criteria are evaluated on a holdout set of contexts, unseen during training. The x-axis corresponds to the number of contexts seen during training, i.e., the number of subgradient steps taken.

In this setting we use two setups, which differ by the observed feature expectations. First, in the **feature expectations** setup, we assume that the whole optimal policy can be observed, therefore, for training we use the feature expectations of the expert's policy. The results are shown in Fig. 7. They show a strong correlation between 'loss minimization' and 'value maximization'. EW converges faster than PSGD and the ES method consistently

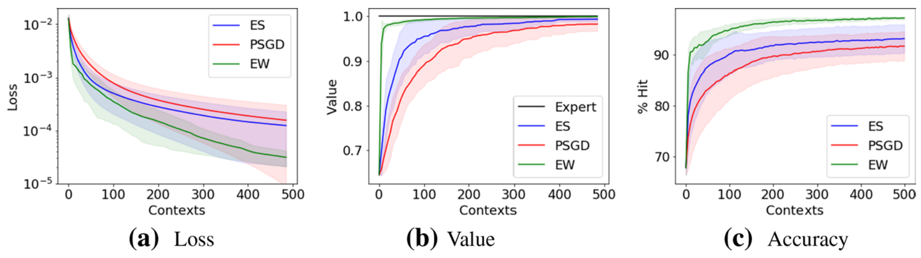


Fig. 8 Online learning curve in the autonomous driving simulation—learning from trajectories. While in Fig. 7 the demonstrations were in the form of feature expectations, here we provide trajectories, a less informative approach. Although less informative, we observe that, similarly to Fig. 7, all methods perform well, attaining similar performance as when given the full information

lies between EW and PSGD, displaying comparable sample complexity. These results match the theoretical guarantees (Table 1, feature expectations) as EW has tighter bounds when it comes to scalability compared to PSGD and ES.

The second setup we use is the **trajectories** setup. Here we construct the feature expectations using a finite number of samples taken from the expert's policy, each context correspond to a finite rollout of an expert (motivated by real life limitations). The results in Fig. 8 show that all three descent methods attain high value and accuracy in this setup. As in the feature expectations setting, the results validate the theoretical sample complexity, with the exception that ES performs slightly better than PSGD. Comparing the results of the different setups we observe similar performance for training with the whole expert's policy or a sample of it, as expected (Sect. 3.1, practical MDA). Training with trajectories is closer to the available data in real-life applications, since only samples of policies are provided.

4.3.3 Conclusion

The ellipsoid method is not as sample efficient as the descent methods. Furthermore, it demands constant expert monitoring, which in real-world problems might be unavailable. In many real-world tasks, such as the sepsis treatment domain, there is an abundance of offline data, yet evaluation in real-life may not be available. Thus, we do not include experiments of the ellipsoid method in the sepsis treatment domain.

The ES and EW methods also have their drawbacks: ES requires computation of the loss function at a considerably large number of points for every descent step. This requirement makes the ES method computationally expensive and prevents it from scaling to larger environments. The EW method assumes that the model parameters lay within the simplex, an assumption that limits the policy space in the linear case, and may not hold in the non-linear case, where the mapping between the context and the reward is modeled by a neural network. As such, we do not include these methods in the sepsis treatment domain.

4.4 Sepsis treatment simulator

This domain simulates a decision-making process for treating sepsis. Sepsis is a severe, life-threatening infection, where the treatment applied to a patient is crucial for saving its life. To create a sepsis treating simulator, we leverage the MIMIC-III data set (Johnson

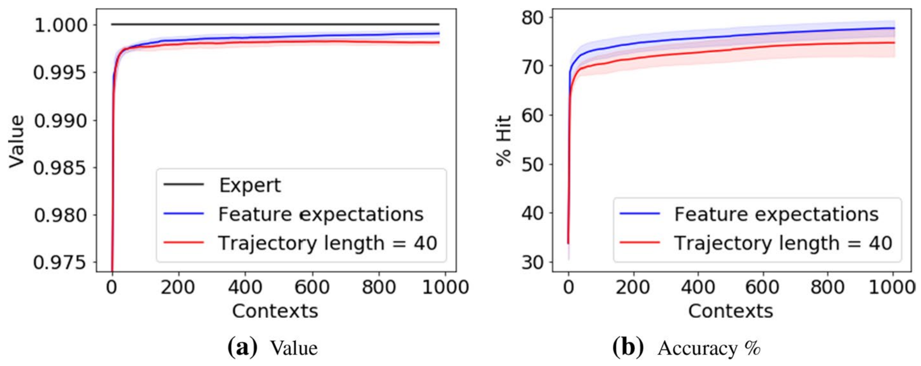


Fig. 9 Online setting in sepsis treatment. We compare the relative value and accuracy when the agent is provided the feature expectations or finite length trajectories. We observe that while as the feature expectations are more informative, the performance is slightly better. However, notice that the difference is negligible and amounts to less than 0.5% difference in the relative value

et al. 2016). This data set includes data from hospital electronic databases, social security, and archives from critical care information systems, that had been acquired during routine hospital care. We follow the data processing steps that were taken in Jeter et al. (2019) to extract the relevant data in a form of normalized measurements of sepsis patients during their hospital admission and the treatments that were given to each patient. The measurements include dynamic measures, e.g., heart rate, blood pressure, weight, body temperature, blood analysis standard measures (glucose, albumin, platelets count, minerals, etc.), as well as static measures such as age, gender, re-admission (of the patient), and more.

From the processed data we construct a dynamic treatment regime, modeled as a CMDP, in which a clinician acts to improve a sick patient's medical condition. The context represents patient features that are constant during treatment, such as age and height. The state summarizes dynamic measurements of the patient, e.g., blood pressure and EEG readouts. The actions represent different combinations of fluids and vasopressors, drugs commonly provided to restore and maintain blood pressure in sepsis patients. The mapping from the context to the true reward is constructed from the data. Dynamic treatment regimes are particularly useful for managing chronic disorders and fit well into the broader paradigm of personalized medicine (Komorowski et al. 2018; Prasad et al. 2017). Furthermore, dynamic treatment regimes have contextual properties; what is defined as healthy blood pressure for a patient differs based on age and weight (Wesselink et al. 2018). In our setting, W^* captures this information – mapping from contextual (e.g., age) and dynamic information (e.g., blood pressure) to reward.

As noted in previous sections, we move toward real-life application and eliminate the inefficient methods. In this section we evaluate the PSGD and compare it with GPI (Sect. 3.4) and contextual BC (Sect. 3.3.2).

4.4.1 Online setting

In this setting we evaluate only the PSGD method. Similarly to the autonomous driving simulation we use two setups: (1) we train the methods with the expert's feature expectations for each context, and (2) instead of using the expert's feature expectations for each given context, we use an estimation, calculated from a given expert trajectory (Sect. 3.1, practical MDA). We present the results of both setups in the same figure, so a comparison between the setups can be done.

We observe in Fig. 9 that PSGD performs well in both setups, with slightly better performance with feature expectations, as expected. This supports the theory, as using samples should not affect the convergence results and truncation after 40 steps should incur only a small penalty. An important observation is that high accuracy is not necessary for high value, as our agents achieve near-perfect value with relatively low accuracy. This reinforces the use of IRL for imitation learning tasks, as it supports the claim that the reward function, not the policy, is the most concise and accurate representation of the task (Abbeel & Ng 2004).

4.4.2 Offline setting

Here, we evaluate the COIRL, GPI and contextual BC methods. We test the ability of these methods to generalize with a limited amount of data. The motivation for this experiment comes from real-life applications, where the data available is often limited in size. The data, similarly to the online setting, is constructed from context-trajectory pairs. In this setting we minimize the loss function (Eq. (8)) by taking descent steps on mini-batches sampled from the data set, with repetition, which invalidates the theoretical results. We conduct two experiments that evaluate the performance as a function of the train-set size (the amount of context-trajectory pairs used for training). We consider two mappings from the context to the reward; a **linear** mapping, and a **non-linear** mapping. For the **non-linear** mapping we use a DNN estimator which constitutes another step towards real-world applicability.

Remark 4.1 Contextual BC is a method to learn a contextual policy, instead of a contextual reward. In its implementation we use a DNN that, given a *context* and *state-features*, computes a probability vector, $\hat{\pi}_c(s)$, representing the agent's policy – i.e., the probability to take action $a \in \mathcal{A}$ is the a'th element of the DNN output $\hat{\pi}_c(s,a)$. The *state-features* that are given as an input greatly affect BC performance, especially when we compare it to COIRL, which uses the real dynamics as well as features that represent each state. BC can make good use of the dynamics, as states with similar dynamics should be mapped to similar actions. To improve the performance for BC, we use the same *state-features* that COIRL uses (HR, blood pressure, etc...), in addition to a feature-vector that represents the dynamics. For each state, $s \in \mathcal{S}$, the dynamics can be represented as a concatenation of the probability vectors, $\left\{P(s,a)\right\}_{a\in\mathcal{A}}$, where $P(s,a)[i] = P(s,a,s_i)$. The dimension of the dynamics for each state is $|\mathcal{S}| \cdot |\mathcal{A}|$ which is relatively large in the sepsis treatment simulator, hence we reduce its dimensionality with PCA.

In Fig. 10 we compare the performance of COIRL, GPI and contextual BC in the **linear** setting, when provided with a fixed amount of data. The results show that in the sepsis treatment domain, the COIRL and GPI methods perform similarly and able to generalize well for a small amount of train data compared to contextual BC. As expected, in Fig. 10(b) BC attains better accuracy on the train data while in Fig. 10(a) COIRL and GPI methods attain better value on the train data. Another observation is that COIRL achieves similar performance on the training data and on the test data; it is able to generalize to unseen contexts, even when the amount of training data is small. On the other hand, BC achieves almost perfect accuracy and high value on the train data but performs poorly on

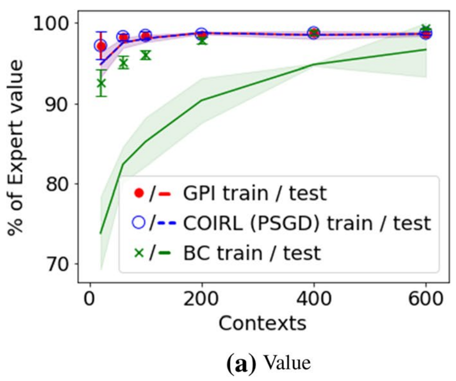


Fig. 10 Offline setting in sepsis treatment. The x-axis denotes the number of contexts in the training set. Results on the train data are represented using circles and x's, the results on a holdout test data-set represented as lines. Given a sufficient amount of contexts seen, GPI is comparable to re-solving the domain, hence there is a large overlap between the results of GPI and COIRL. Contextual BC requires much more data to generalize well

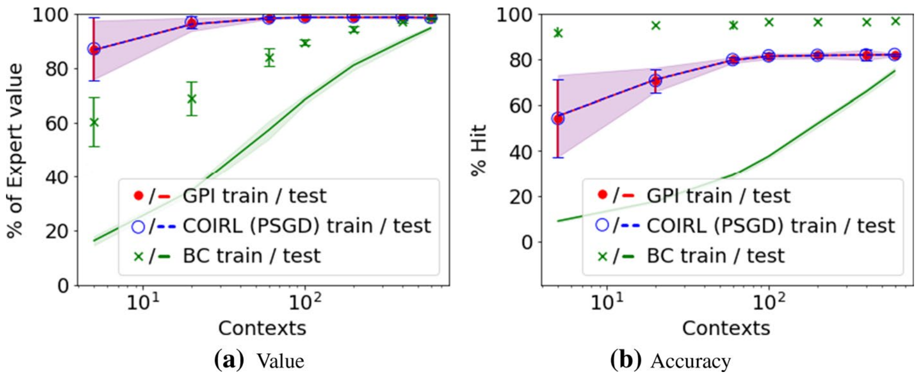


Fig. 11 Offline setting in sepsis treatment: non-linear mapping. The x-axis denotes the number of contexts in the training set (logarithmic scale). Results on the train data are represented using circles and x's, the results on a holdout test data-set represented as lines. Similar to the linear setup, GPI and COIRL generalize well for a small amount of train data where the performance on the train data and on the test data is similar. Contextual BC performance on the train set is almost perfect, where its performance on the test data requires a large amount of expert demonstrations

the test data. This generalization gap goes away only when a large amount of data is available for training.

The **non-linear** setup results presented in Fig. 11. Here, the x-axis is in logarithmic scale. The performance of all methods is similar to the **linear** setup; COIRL and GPI methods perform similarly and generalize to unseen contexts even when given a small amount of train data. Contextual BC generalizes to unseen contexts only for a large amount of train data. As in the **linear** setup, the BC method attains better accuracy while the COIRL and GPI methods attain better value.

4.5 Sepsis treatment in real-life

In the previous subsections we focused on analyzing COIRL in simulated environments. We have taken a sequence of steps with the aim of making the simulations more and more realistic. In all of these simulations, the expert trajectories were always generated from the optimal policy (for a given context) w.r.t to the true context-reward mapping. Our results suggest that the reward estimated by COIRL induces a policy that attains a close-to-expert value in both linear and non-linear settings. Now we turn to examine our algorithms in a real world data set. Since the true mapping is no longer known, we can only measure the accuracy of our resulted policies. In previous sections we observed that while accuracy does not necessarily imply value (i.e., a policy can have optimal value but not be 100% accurate), these measures are often correlated. In addition, since the true dynamics of the MDP is now unknown, we estimate it from the data itself.

4.5.1 Data processing

We follow the steps done in Komorowski et al. (2018) to construct a time-series data of static and dynamic measurements. The data is divided to trajectories, where each trajectory represent a different patient. We consider only trajectories of length greater than 10 that represent 40 hours. The processed data is consisted of 14, 591 trajectories, divided to a 60-20-20 train-validation-test partition. Each trajectory corresponds to a static measurements vector and a time series of dynamic measurements vectors, with time steps of 4 hours. In the following experiments each model is trained on the training set, until an early stopping criteria is met on the validation set. We then report the accuracy (action matching with the clinicians actions) on the holdout test set.

4.5.2 Model fitting

As in Sect. 4.4, the contexts and the states constructed from static and dynamic measurements respectively. In our model, the contexts are in $\mathbb{R}^7$ and include the gender, age, weight, GCS, elixhauser co-morbidity score, whether the patient was mechanically ventilated at s_0 and whether the patient has been re-admitted to the hospital. The actions are defined to be the amount of vasopressors given to a patient at each time slot, and five discrete actions are constructed by dividing the possible values into five bins. The state space is constructed by clustering the observed patient dynamic measurements from the data with K-means (MacQueen et al. 1967). The clustering process is repeated for different numbers of states and different weights for each measurement (to control the importance of each measurement for the state space). Each model is evaluated by two terms: (1) number of different actions taken on the same state for the same patient: $\mathbb{E}_\tau\left[\mathbb{E}_{s\in\tau}[|\hat{\mathcal{A}}_s^\tau|]\right]$, where $\hat{\mathcal{A}}_s^\tau = \{a \in \mathcal{A} : (s,a) \in \tau\}$. (2) number of different states in each trajectory: $\mathbb{E}_\tau\left[|\hat{\mathcal{S}}^\tau|\right]$, where $\hat{\mathcal{S}}^\tau = \{s \in \mathcal{S} : s \in \tau\}$. In both terms, τ is a trajectory drawn from the data. We require the first term to be as small as possible, to achieve a consistent experts policies in the CMDP model, the second term required to be large, to force the resulted model to distinguish between different states in the same patient's trajectory. Obviously, the model has to be as small as possible, to enable generalization. The chosen model consists 5000 states.

Table 2 Results on real world data. We measure the accuracy of each method over a holdout test set. In the non-linear setting, COIRL achieves the best accuracy and outperforms BC

Method		Accuracy %
COIRL	Non-linear	83.74 ± 1.02
	Linear	45.17 ± 7.14
BC		73.12 ± 0.82

While processing the data, we noticed that clinicians behavior with respect to some measurements is random. To address this matter we consulted with clinicians and defined a set of important dynamic measurements, among them we use the clustering process to choose the patients relevant dynamic measurements for the states; states were clustered for any possible single measurement and the five best dynamic measurements were chosen: mean blood pressure, diastolic blood pressure, shock index, cumulative balance of fluids and the fluids given to a patient. The features in this CMDP are action-dependent and set to be a concatenation of $e_i \in \mathbb{R}^{|\mathcal{S}|}$ and $e_j \in \mathbb{R}^{|\mathcal{A}|}$ where e_i is a vector of all zeros and a single 1 that represents each state and e_j represents the action, overall the there are 5, 005 features for each state-action pair.

As described in Sect. 2, learning the transition kernel is an orthogonal problem to the COIRL problem, and can be viewed as a part of the model fitting process. Our dynamics model is context-dependent; the contexts (patients) clustered into five clusters and the dynamics of each cluster are then estimated using the training data.

4.5.3 Methods

For COIRL we report results for the *linear* and the *non-linear* mappings. In both setups, we use a discount factor $\gamma = 0.7$ and a mini-batch of size 32. The stopping criteria is set to stop when five consecutive steps do not increase the validation accuracy. To speed-up the validation process we sample a subset of 300 patients from the validation partition at the beginning of each seed and use them to validate the model. In the *linear* setup the step size is $\alpha_t = 0.25 \cdot 0.95^t$. The *non-linear* setup use a DNN to learn the mapping $f_W : \mathcal{C} \longrightarrow \mathbb{R}^{|S|+|A|} = \mathbb{R}^{5,005} \approx \mathbb{R}^{5K}$, it has four layers with a Leaky ReLU activation and batch-normalization between the first and second layers, and Leaky ReLu activation between the second and third layers, their sizes are $20K$, $10K$, $10K$, and $5K$, respectively. Here, the step size is $\alpha_t = 0.2 \cdot 0.95^t$

For BC we also use a DNN for function approximation, as we found it to work much better than a linear model. We also experimented with different sets of features as inputs. The features that we found to give the best performance were computed in a similar manner to the features that we used for BC in Remark 4.1, using the dynamics of the estimated CMDP, resulted with $5K$ features that represent each state. Concretely, the DNN received a concatenation of the context and the features that represent the current state (size of 5, 007) and outputs a stochastic policy (softmax over the outputs of the last layer). The network architecture is composed of three linear layers of sizes 625, 125, and 5, respectively. Each layer is followed by a Leaky ReLU activation, and a Softmax activation is used on the output. Similar to COIRL, the model is trained over the training set partition and the stopping criteria is set to stop after 5 epochs of non-increasing validation accuracy. The loss of the DNN is the binary cross-entropy loss between the DNN output and the observed action, $e_i \in \mathbb{R}^{|\mathcal{A}|}$. The mini-batch size is 32 and the optimizer is SGD with step size $\alpha_t = 0.1 \cdot \frac{1}{1+10^{-7}t}$.

Each method trained and evaluated over five seeds, the results are presented in Table 2. We can see that COIRL with a non-linear mapping attains the best performance, while the linear mapping achieves poor accuracy. BC performs well overall, but not as good as COIRL. In Lee et al. (2019) the authors use similar data set and action space. Their methods, TRIL and DSFN, achieve $80 \pm 2\%$ and $79 \pm 5\%$, respectively, which is lower than COIRL and with higher variance. These results suggest that the contextual hypothesis better represents the real world, i.e., that physicians indeed use personalized treatments based on context.

5 Discussion

Motivated by current trends in personalized medicine (Juskalian et al. 2020), we proposed the Contextual Inverse Reinforcement Learning framework. While most works in RL assume the agent is provided with a reward signal, we focused on a more realistic setting, where the reward is unknown to the agent and, instead, it observes and receives feedback from an expert. As opposed to the standard IRL setting, in the contextual case, each context defines a new MDP. This leads to a new form of generalization in RL, where the agent is trained and learns how to act optimally on a set of contexts, followed by an evaluation procedure on a set to which the agent was not exposed during training.

We show that solving the COIRL objective can be performed by minimizing a convex optimization task. As this objective is not differentiable, we proposed two schemes based on subgradient descent (MDA and ES) and an adaptation of cutting plane methods (ellipsoid). We analyzed the convergence properties of each algorithm and summarized the results in Table 1.

All of the proposed methods assume that the dynamics are known, but in many applications the dynamics and even the state space are unknown. Following the description in Sect. 2, any method that learns the dynamics efficiently can be used prior to COIRL. For example, in online frameworks, where the expert provides demonstrations in an online manner, the dynamics can be learned as proposed in Abbeel & Ng (2005). In this case, the dynamics estimation and COIRL should run iteratively, such that every change in the estimation of the dynamics introduces a new COIRL problem that should be solved. In offline frameworks the dynamics can be estimated prior to COIRL, similarly to Sect. 4.5.

In addition to the theoretical analysis, we performed extensive empirical evaluation between all proposed algorithms, including baseline approaches. Here, we see a mixed correlation between theoretical and practical results. Regarding the ellipsoid schemes, we observe that indeed as shown theoretically, they are sub-optimal compared to the other methods. However, comparing MDA to ES, we see that ES matches and sometimes outperforms MDA even though the theoretical upper-bounds are tighter for MDA. These results correlate with observations seen by Nemirovsky & Yudin (1983), where ES often provides better empirical results.

Aside from comparing between our proposed methods, we also compared to a common learning scheme—behavioral cloning. While IRL aims to find a reward function which explains the experts behavior, behavioral cloning (log-likelihood action matching) simply converts the RL task into a supervised learning problem. Previous works (Abbeel & Ng 2004) talk about the importance of IRL, compared to BC. In our experiments we see this clearly. While the reward/value is smooth (Lipschitz) w.r.t. the context, the policy is not. As a small change in the context may lead to a large switch in the policy (the optimal actions change in certain states), we observe that BC struggles. This can also be seen in the fact that COIRL often reaches imperfect action-matching (accuracy) yet attains the optimal return.

We demonstrated how existing policies can be transferred to new contexts, avoiding planning in test-time. This is important, as planning complexity is a function of the size of the MDP, thus this form of transfer may be crucial for real-world scenarios. Our experiments illustrate how combining offline COIRL with GPI eases the computational load on the agent while maintaining strong performance with few training examples.

Finally, COIRL achieved the highest accuracy in the challenging task of predicting the clinicians treatment in the real world sepsis treatment data set. This suggests that sepsis treatment can be modeled as a contextual MDP; we hope that these findings will motivate future work in using contextual MDPs to model real-world decision making.

To conclude, we proposed the COIRL framework and analyzed it under a linear mapping assumption. In real-world cases, where the linear assumption holds, COIRL can be used effectively. Future work may combine COIRL with schemes such as meta-learning (Finn et al. 2017) in order to cope with infinitely large MDPs and non-linear mappings.

Appendices

Proofs for Section 3

Definition A.1 (Bregman distance) Let $\psi : \mathcal{W} \to R$ be strongly convex and continuously differential in the interior of $\mathcal{W}$. The Bregman distance is defined by $D_\psi(x, y) = \psi(x) - \psi(y) - (x - y) \cdot \nabla\psi(y)$, where ψ is strongly convex with parameter σ.

Definition A.2 (Conjugate function) The conjugate of a function $\psi(y)$, denoted by $\psi^*(y)$ is

$$\max_{x \in \mathcal{W}} \{x \cdot y - \psi(x)\}.$$

Example let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The associated dual norm, denoted $\|\cdot\|_*$, is defined as $\|z\|_* = \sup\{z^\intercal x \mid \|x\| \le 1\}$. The dual norm of $\|\cdot\|_2$ is $\|\cdot\|_2$, and the dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$.

Before we begin with the proof of Lemma 3.2, we make the following observation. By definition, $\hat{\pi}_c(W)$ is the optimal policy w.r.t. $c^T W$; i.e., for any policy π we have that

$$c^T W \cdot \mu_c^{\hat{\pi}_c(W)} \ge c^T W \cdot \mu_c^{\pi}. \tag{11}$$

Proof of Lemma 3.2 **1.** We need to show that $\forall W_1, W_2 \in \mathcal{W}, \forall \lambda \in [0, 1]$, we have that

$$L_{\text{lin}}\big(\lambda W_1 + (1-\lambda)W_2\big) \le \lambda L_{\text{lin}}(W_1) + (1-\lambda)L_{\text{lin}}\big(W_2\big)$$

$$\begin{aligned}
&L_{\text{lin}}\big(\lambda W_1 + (1-\lambda)W_2\big)\\
&= \mathbb{E}_c\Big[c^T\big(\lambda W_1 + (1-\lambda)W_2\big)\cdot\Big(\mu_c^{\hat{\pi}_c(\lambda W_1+(1-\lambda)W_2)} - \mu_c^*\Big)\Big]\\
&= \lambda\mathbb{E}_c\Big[c^T W_1\cdot\Big(\mu_c^{\hat{\pi}_c(\lambda W_1+(1-\lambda)W_2)} - \mu_c^*\Big)\Big]\\
&\quad + (1-\lambda)\mathbb{E}_c\Big[c^T W_2\cdot\Big(\mu_c^{\hat{\pi}_c(\lambda W_1+(1-\lambda)W_2)} - \mu_c^*\Big)\Big]\\
&\le \lambda\mathbb{E}_c\Big[c^T W_1\cdot\Big(\mu_c^{\hat{\pi}_c(W_1)} - \mu_c^*\Big)\Big] + (1-\lambda)\mathbb{E}_c\Big[c^T W_2\cdot\Big(\mu_c^{\hat{\pi}_c(W_2)} - \mu_c^*\Big)\Big]\\
&= \lambda L_{\text{lin}}(W_1) + (1-\lambda)L_{\text{lin}}\big(W_2\big),
\end{aligned}$$

where the inequality follows from Eq. (11).

2. Fix $z \in \mathcal{W}$. We have that

$$\begin{aligned}
L_{\text{lin}}(z) &= \mathbb{E}_c\big[c^T z\cdot\big(\mu_c^{\hat{\pi}_c(z)} - \mu_c^*\big)\big]\\
&\ge \mathbb{E}_c\big[c^T z\cdot\big(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\big)\big]\\
&= L_{\text{lin}}(W) + (z-W)\cdot\mathbb{E}_c\big[c\odot\big(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\big)\big],
\end{aligned}$$

where the inequality follows from Eq. (11).

3. Recall that a bound on the dual norm of the subgradients implies Lipschitz continuity for convex functions. Thus it is enough to show that $\forall W \in \mathcal{W}, \|g(W)\|_p = \|\mathbb{E}_c\Big[c\odot\Big(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\Big)\Big]\|_p \le L$. Let $p = \infty$, we have that

$$\begin{aligned}
\|g(W)\|_\infty &= \Big\|\mathbb{E}_c c\odot\big(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\big)\Big\|_\infty\\
&\le \mathbb{E}_c\|c\odot\big(\mu_c^{\hat{\pi}_c(W)} - \mu_c^*\big)\|_\infty
\end{aligned} \qquad \text{(Jensen inequality)}$$

$$\le \mathbb{E}_c\|c\|_\infty\Big\|\mu_c^{\pi_i} - \mu_c^{\pi_j}\Big\|_\infty \le \frac{2}{1-\gamma}. \tag{12}$$

where in Eq. (Jensen inequality) we used the fact that $\forall\pi$ we have that $\|\mu_c^\pi\|_\infty \le \frac{1}{1-\gamma}$, thus, for any π_i, π_j,

$$\Big\|\mu_c^{\pi_i} - \mu_c^{\pi_j}\Big\|_\infty \le \frac{2}{1-\gamma}.$$

Therefore, $L = \frac{2}{1-\gamma}$ w.r.t. $\|\cdot\|_\infty$. Since $\|\cdot\|_2 \le \sqrt{dk}\|\cdot\|_\infty$ we get that $L = \frac{2\sqrt{dk}}{1-\gamma}$ w.r.t. $\|\cdot\|_2$.

□

Proofs for Section 3.2

Proof of Theorem 3.3

Proof of Theorem 3.3 We prove the theorem by showing that the volume of the ellipsoids Θ_t for $t = 1, 2, \ldots$ is bounded from below. In conjunction with Lemma 3.3, which claims there is a minimal rate of decay in the ellipsoid volume, this shows that the number of times the ellipsoid is updated is polynomially bounded.

We begin by showing that $\underline{W}^*$ always remains in the ellipsoid. We note that in rounds where $V^*_{c_t} - V^{\hat{\pi}_t}_{c_t} > \epsilon$, we have $\underline{W}^{*T}\left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right) > \epsilon$. In addition, as the agent acts optimally w.r.t. the reward $r_t = c_t^T W_t$, we have that $\underline{W}_t^T\left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right) \leq 0$. Combining these observations yields:

$$\left(\underline{W}^* - \underline{W}_t\right)^T \cdot \left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right) > \epsilon > 0. \tag{13}$$

This shows that $\underline{W}^*$ is never disqualified when updating Θ_t. Since $\underline{W}^* \in \Theta_0$ this implies that $\forall t : \underline{W}^* \in \Theta_t$. Now we show that not only $\underline{W}^*$ remains in the ellipsoid, but also a small ball surrounding it. If θ is disqualified by the algorithm: $(\theta - \underline{W}_t)^T \cdot \left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right) < 0$. Multiplying this inequality by -1 and adding it to (13) yields: $(\underline{W}^* - \theta)^T \cdot \left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right) > \epsilon$. We apply Hölder inequality to LHS:

$$\begin{aligned}
\epsilon &< \text{LHS} \\
&\leq ||\underline{W}^* - \theta||_\infty \cdot ||\left(c_t \odot \left(\mu^*_{c_t} - \mu^{\hat{\pi}_t}_{c_t}\right)\right)||_1 \\
&\leq \frac{2k}{1-\gamma}||\underline{W}^* - \theta||_\infty
\end{aligned}$$

Therefore for any disqualified θ: $||\underline{W}^* - \theta||_\infty > \frac{(1-\gamma)\epsilon}{2k}$, thus $B_\infty\left(\underline{W}^*, \frac{(1-\gamma)\epsilon}{2k}\right)$ is never disqualified. This implies that:

$$\forall t : \text{vol}\left(\Theta_t\right) \geq \text{vol}\left(\Theta_0 \cap B_\infty(\underline{W}^*, \frac{(1-\gamma)\epsilon}{2k})\right) \geq \text{vol}\left(B_\infty(\underline{W}^*, \frac{(1-\gamma)\epsilon}{4k})\right).$$

Finally, let M_T be the number of rounds by T in which $V^*_{c_t} - V^{\hat{\pi}_t}_{c_t} > \epsilon$. Using Lemma 3.3 we get that:

$$\begin{aligned}
\frac{M_T}{2(dk+1)} &\leq \log\left(\text{vol}(\Theta_1)\right) - \log\left(\text{vol}\left(\Theta_{T+1}\right)\right) \\
&\leq \log\left(\text{vol}\left(\text{MVEE}(B_\infty(0,1))\right)\right) - \log\left(\text{vol}\left(B_\infty(0, \frac{(1-\gamma)\epsilon}{4k})\right)\right) \\
&\leq \log\left(\text{vol}\left(\text{MVEE}\left(B_2(0, \sqrt{dk})\right)\right)\right) - \log\left(\text{vol}\left(B_2\left(0, \frac{(1-\gamma)\epsilon}{4k}\right)\right)\right) \\
&\leq \log\left(\left(\frac{4k\sqrt{dk}}{(1-\gamma)\epsilon}\right)^{dk}\right) \\
&\leq dk \log\frac{4k\sqrt{dk}}{(1-\gamma)\epsilon}.
\end{aligned}$$

Therefore $M_T \leq 2dk(dk+1)\log\frac{4k\sqrt{dk}}{(1-\gamma)\epsilon} = \mathcal{O}(d^2k^2\log(\frac{dk}{(1-\gamma)\epsilon}))$. □

Proof of Theorem 3.4

Lemma B.1 (Azuma's inequality) *For a martingale* $\{S_i\}_{i=0}^n$, *if* $|S_i - S_{i-1}| \leq b$ *a.s. for* $i = 1, ..., n$:

$$P\left(|S_n - S_0| > b\sqrt{2n\log(\tfrac{2}{\delta})} \right) < \delta$$

Proof of Theorem 3.4 We first note that we may assume that for any t: $||W^* - W_t||_\infty \leq 2$. If $\underline{W}_t \notin \Theta_0$, we update the ellipsoid by $\Theta_t \leftarrow \text{MVEE}\left(\left\{ \theta \in \Theta_t : \left(\theta - \underline{W}_t\right)^T \cdot e_j \lesseqgtr 0 \right\} \right)$ where e_j is the indicator vector of coordinate j in which $\underline{W}_t$ exceeds 1, and the inequality direction depends on the sign of $(\underline{W}_t)_j$. If $\underline{W}_t \notin \Theta_0$ still, this process can be repeated for a finite number of steps until $\underline{W}_t \in \Theta_0$, as the volume of the ellipsoid is bounded from below and each update reduces the volume (Lemma 3.3). Now we have $\underline{W}_t \in \Theta_0$, implying $||W^* - W_t||_\infty \leq 2$. As no points of Θ_0 are removed this way, this does not affect the correctness of the proof. Similarly, we may assume $||W_t^* - W_t||_\infty \leq 2$ as $W_t^* \in \Theta_0$.

We denote W_t which remains constant for each update in the batch by W. We define $t(i)$ the time-steps corresponding to the demonstrations in the batch for $i = 1, ..., n$. We define $z_i^{*,H}$ to be the expected value of $\hat{z}_i^{*,H}$, and z_i^* to be the outer product of $c_{t(i)}$ and the feature expectations of the expert policy for $W_{t(i)}^*, c_{t(i)}, \xi'_{t(i)}$. We also denote $W_{t(i)}^*$ by W_i^*. We bound the following term from below, as in Theorem 3.3:

$$\begin{aligned}
&\left(\underline{W}^* - \underline{W}\right)^T \cdot \left(\frac{\bar{Z}^*}{n} - \frac{\bar{Z}}{n} \right) \\
&= \frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(\hat{z}_i^{*,H} - z_i\right) \\
&= \frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(z_i^* - z_i\right) + \frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(z_i^{*,H} - z_i^*\right) + \\
&\quad \frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(\hat{z}_i^{*,H} - z_i^{*,H}\right) \\
&= \underbrace{\frac{1}{n}\sum_{i=1}^n \left(\underline{W}_i^* - \underline{W}\right)^T \cdot \left(z_i^* - z_i\right)}_{(1)} + \underbrace{\frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}_i^*\right)^T \cdot \left(z_i^* - z_i\right)}_{(2)} + \\
&\quad \underbrace{\frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(z_i^{*,H} - z_i^*\right)}_{(3)} + \underbrace{\frac{1}{n}\sum_{i=1}^n \left(\underline{W}^* - \underline{W}\right)^T \cdot \left(\hat{z}_i^{*,H} - z_i^{*,H}\right)}_{(4)}
\end{aligned}$$

(1): Since the sub-optimality criterion implies a difference in value of at least ϵ for the initial distribution which assigns 1 to the state where the agent errs, we may use identical arguments to the previous proof. Therefore, the term is bounded from below by ϵ.

(2): By assumption $||\underline{W}^* - \underline{W}_i^*||_\infty \leq \frac{(1-\gamma)\epsilon}{8k}$ thus since $||(z_i^* - z_i)||_1 \leq \frac{2k}{1-\gamma}$ by Hölder's inequality the term is bounded by $\frac{\epsilon}{4}$.

(3): We have $||x_i^{*,H} - x_i^*||_1 \leq \frac{k\gamma^H}{1-\gamma}$ from definitions, thus $||z_i^{*,H} - z_i^*||_1 \leq \frac{k\gamma^H}{1-\gamma}$ since $c \in \Delta_{d-1}$. As mentioned previously we may assume $||W^* - W_t||_\infty \leq 2$, therefore by Hölder's inequality the term is bounded by $\frac{\epsilon}{4}$ due to our choice of H:

$$\begin{aligned}
\gamma^H &= (1-(1-\gamma))^H \\
&= \left((1-(1-\gamma))^{\frac{1}{1-\gamma}}\right)^{(1-\gamma)H} \\
&= \left((1-(1-\gamma))^{\frac{1}{1-\gamma}}\right)^{\log(\frac{8k}{(1-\gamma)\epsilon})} \\
&\leq e^{-\log(\frac{8k}{(1-\gamma)\epsilon})} \\
&= \frac{(1-\gamma)\epsilon}{8k}.
\end{aligned}$$

(4): The partial sums $\sum_{i=1}^N (\underline{W^*} - \underline{W})^T \cdot (z_i^{*,H} - \hat{z}_i^{*,H})$ for $N = 0, ..., n$ form a martingale sequence. Note that:

$$||z_i^{*,H}||_1 \leq \frac{k}{1-\gamma}, \quad ||\hat{z}_i^{*,H}||_1 \leq \frac{k}{1-\gamma}, \quad ||W^* - W_t||_\infty \leq 2,$$

thus, we can apply Azuma's inequality (Lemma B.1) with $b = \frac{4k}{(1-\gamma)}$ and with our chosen n this yields: $\sum_{i=1}^n (\underline{W^*} - \underline{W})^T \cdot (z_i^{*,H} - \hat{z}_i^{*,H}) \leq \frac{n\epsilon}{4}$ with probability of at least $1 - \frac{\delta}{2dk(dk+1)\log(\frac{16k\sqrt{dk}}{(1-\gamma)\epsilon})}$.

Thus $(\underline{W^*} - \underline{W})^T \cdot (\frac{\bar{Z}^*}{n} - \frac{\bar{Z}}{n}) > \frac{\epsilon}{4}$ and as in Theorem 3.3 this shows $B_\infty(\underline{W^*}, \frac{(1-\gamma)\epsilon}{8k})$ is never disqualified, and the number of updates is bounded by $2dk(dk+1)\log(\frac{16k\sqrt{dk}}{(1-\gamma)\epsilon})$, and multiplied by n this yields the upper bound on the number of rounds in which a sub-optimal action is chosen. By union-bound, the required bound for term **(4)** holds in all updates with probability of at least $1-\delta$. □

Proofs for Section 3.4

Proof of Theorem 3.6 This proof follows the proof for Lemma 1 in (Barreto et al. 2017), with additional arguments taken from proofs of the simulation lemma. We define $\epsilon_P = \max_{s,a} ||P_c(\cdot|s,a) - P_{c_j}(\cdot|s,a)||_1, \epsilon_R = \max_s |R_c^*(s) - R_{c_j}^*(s)|$.

We first note that:

$$Q_c^*(s,a) - Q_c^{\pi_{c_j}^*}(s,a) \leq |Q_c^*(s,a) - Q_{c_j}^*(s,a)| + |Q_{c_j}^*(s,a) - Q_c^{\pi_{c_j}^*}(s,a)|$$

and bound each of these terms:

$$
\begin{aligned}
&|Q^*_c(s,a) - Q^*_{c_j}(s,a)| \\
&\quad = \Big|R^*_c(s) - R^*_{c_j}(s) + \gamma\left(\sum_{s'} P_c(s'|s,a)\max_b Q^*_c(s',b) - \sum_{s'} P_{c_j}(s'|s,a)\max_b Q^*_{c_j}(s',b)\right)\Big| \\
&\quad \le |R^*_c(s) - R^*_{c_j}(s)| + \gamma\Big|\sum_{s'} P_c(s'|s,a)\Big(\max_b Q^*_c(s',b) - \max_b Q^*_{c_j}(s',b)\Big)\Big| + \\
&\qquad \gamma\Big|\sum_{s'}\Big(P_c(s'|s,a) - P_{c_j}(s'|s,a)\Big)\max_b Q^*_{c_j}(s',b)\Big| \\
&\quad \le \epsilon_R + \gamma\sum_{s'} P_c(s'|s,a)\Big|\max_b Q^*_c(s',b) - \max_b Q^*_{c_j}(s',b)\Big| + \gamma V_{max}||P_c(\cdot|s,a) - P_{c_j}(\cdot|s,a)||_1 \\
&\quad \le \epsilon_R + \gamma V_{max}\epsilon_P + \gamma\sum_{s'} P_c(s'|s,a)\max_b \big|Q^*_c(s',b) - Q^*_{c_j}(s',b)\big| \\
&\quad \le \epsilon_R + \gamma V_{max}\epsilon_P + \gamma\max_{s',b}|Q^*_c(s',b) - Q^*_{c_j}(s',b)|
\end{aligned}
$$

taking $\max_{s,a}$ of the resulting inequality and solving for LHS yields:

$$\max_{s,a}|Q^*_c(s,a) - Q^*_{c_j}(s,a)| \le \frac{\epsilon_R + \gamma V_{max}\epsilon_P}{1-\gamma}.$$

For the second term we follow similar steps:

$$
\begin{aligned}
&|Q^*_{c_j}(s,a) - Q_c^{\pi^*_{c_j}}(s,a)| \\
&\quad = \epsilon_R + \gamma\Big|\sum_{s'} P_c(s'|s,a)Q^*_{c_j}\Big(s',\pi^*_{c_j}(s')\Big) - \sum_{s'} P_{c_j}(s'|s,a)Q_c^{\pi^*_{c_j}}\Big(s',\pi^*_{c_j}(s')\Big)\Big| \\
&\quad \le \epsilon_R + \gamma V_{max}\epsilon_P + \gamma\sum_{s'} P_c(s'|s,a)\Big|Q^*_{c_j}\Big(s',\pi^*_{c_j}(s')\Big) - Q_c^{\pi^*_{c_j}}\Big(s',\pi^*_{c_j}(s')\Big)\Big| \\
&\quad \le \epsilon_R + \gamma V_{max}\epsilon_P + \gamma max_{s',b}|Q^*_{c_j}(s',b) - Q_c^{\pi^*_{c_j}}(s',b)|
\end{aligned}
$$

taking $\max_{s,a}$ of the resulting inequality and solving for LHS yields:

$$\max_{s,a}|Q^*_{c_j}(s,a) - Q_c^{\pi^*_{c_j}}(s,a)| \le \frac{\epsilon_R + \gamma V_{max}\epsilon_P}{1-\gamma}.$$

Plugging these bounds into the first inequality yields:

$$Q^*_c(s,a) - Q_c^{\pi^*_{c_j}}(s,a) \le 2\frac{\epsilon_R + \gamma V_{max}\epsilon_P}{1-\gamma}.$$

Now, we express ϵ_R, ϵ_P in terms of the distance between the contexts:

$$\begin{aligned}
\epsilon_P &= \max_{s,a} \sum_{s'} |P_c(s'|s,a) - P_{c_j}(s'|s,a)| \\
&= \max_{s,a} \sum_{s'} |(c - c_j)^T \begin{bmatrix} P_1(s'|s,a) \\ \vdots \\ P_d(s'|s,a) \end{bmatrix}| \\
&\le ||c - c_j||_\infty \sum_{s'} || \begin{bmatrix} P_1(s'|s,a) \\ \vdots \\ P_d(s'|s,a) \end{bmatrix} ||_1 \\
&= d||c - c_j||_\infty, \\
\epsilon_R &= \max_s |R^*_c(s) - R^*_{c_j}(s)| \\
&= \max_s |\big(c - c_j\big)^T (W^* \phi(s)| \\
&\le ||c - c_j||_\infty \max_s ||W^* \phi(s)||_1 \\
&= \phi_{max} ||c - c_j||_\infty
\end{aligned}$$

which, plugged into our inequality, yields:

$$Q^*_c(s,a) - Q_c^{\pi^*_{c_j}}(s,a) \le 2\frac{\phi_{max} + \gamma d V_{max}}{1-\gamma} ||c - c_j||_\infty.$$

Note that as a special case, if the dynamics are identical for all contexts, $\epsilon_P = 0$, therefore:

$$Q^*_c(s,a) - Q_c^{\pi^*_{c_j}}(s,a) \le 2\frac{\phi_{max}}{1-\gamma} ||c - c_j||_\infty.$$

To convert the bound to the value function, we add a dummy initial state s_0, with $\phi(s_0) = 0$ and $\forall a : P(\cdot|s_0, a) = \xi$. In this case, applying the above inequality for the initial state yields:

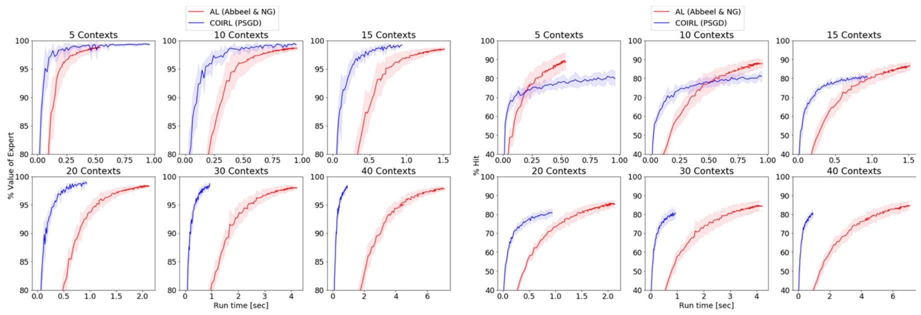
$$V^*_c - V_c^{\pi^*_{c_j}} = \frac{1}{\gamma}\big(Q^*_c(s_0,a) - Q_c^{\pi^*_{c_j}}(s_0,a)\big) \le 2\frac{\phi_{max} + \gamma d V_{max}}{\gamma(1-\gamma)} ||c - c_j||_\infty$$

□

Proof of Theorem 3.5 We denote the maximizing index and action by

$$a_i, i \in arg\,max_a arg\,max_i Q_c^{\pi^*_i}(s,a).$$

We have that



(a) Value **(b)** Accuracy

Fig. 12 value and accuracy as a function of run-time in various context space size. The advantage of COIRL grows as the context space grows

$$
\begin{aligned}
V_c^*(s) - V_c^\pi(s) &= \max_a Q_c^*(s,a) - Q_c^\pi(s,a_i) \\
&\le \max_a Q_c^*(s,a) - Q_c^{\pi^*_{c_i}}\left(s,a_i\right) \\
&\le \max_a Q_c^*(s,a) - \max_a Q_c^{\pi^*_{c_j}}(s,a) \\
&\le \max_a \left(Q_c^*(s,a) - Q_c^{\pi^*_{c_j}}(s,a) \right) \\
&\le 2\frac{\phi_{max}}{1-\gamma} ||c - c_j||_\infty
\end{aligned}
$$

where the first inequality is due to the Generalized Policy Improvement Theorem, Theorem 1 in (Barreto et al. 2017), which claims: $Q_c^\pi(s,a) \ge Q_c^{\pi^*_{c_j}}(s,a)$, the second inequality is from the definition of i, a_i, and the last inequality is the second to last inequality from the previous proof. Taking $\min_j$ then expectation w.r.t. $s \sim \xi$ finishes this proof. $\square$

Experiments

In this section, we describe the technical details of our experiments, including the hyper-parameters used. To solve MDPs, we use value iteration. Our implementation is based on a stopping condition with a tolerance threshold, τ, such that the algorithm stops if $|V_t - V_{t-1}| < \tau$. In the autonomous driving simulation and grid world domains we use $\tau = 10^{-4}$ and in the dynamic treatment regime we use $\tau = 10^{-3}$.

Grid world

The grid world domain, presented at Sect. 3.3.1 is constructed for computational comparisons between methods. The test data includes 100 contexts. Here we include more results in this domain, all measured on the same setup as in Sect. 3.3.1.

The results shown in Fig. 12(a) present the value as a function of run-time for each context space size $|\mathcal{C}|$. In Fig. 12(b) we show the accuracy w.r.t. the expert policy in the same manner. We observe that COIRL achieves better value for any size of the train data and that AL achieves better accuracy after convergence. The accuracy over run-time show that the convergence of AL in a large context space takes more time and that the accuracy gap between the methods after convergence is reduced.

Autonomous driving simulation

The environment is modeled as a tabular MDP that consists of 1531 states. The speed is selected once, at the initial state, and is kept constant afterward. The other 1530 states are generated by 17 X-axis positions for the agent's car, 3 available speed values, 3 lanes and 10 Y-axis positions in which car B may reside. During the simulation, the agent controls the steering direction of the car, moving left or right, i.e., two actions. The feature vector $\phi(s)$ is composed of 3 features: (1) speed, (2) "collision", which is set to 0 in case of a collision and 0.5 otherwise, and (3) "off-road", which is 0.5 if the car is on the road and 0 otherwise.

In these experiments, we define our mappings in a way that induces different behaviors for different contexts, making generalization a more challenging task. We evaluate all algorithms on the same sequences of contexts, and average the results over 5 such sequences. The test data in this domain contains 80 unobserved contexts.

Ellipsoid setting

This section describe the technical details about the experiments in Sect. 4.3.1. Here, the real mapping between the context to the reward is **linear**. We define $W^* = \begin{pmatrix} -1 & 0.75 & 0.75 \\ 0.5 & -1 & 1 \\ 0.75 & 1 & -0.75 \end{pmatrix}$, before normalization. The contexts are sampled uniformly in the 2-dimensional simplex.

Hyper-parameter selection and adjustments: The algorithms maintained a 3×3 matrix to estimate W^*.

Ellipsoid: By definition, the ellipsoid algorithm is hyper-parameter free and does not require tuning.

PSGD: The algorithm was executed with with the parameters: $\alpha_0 = 0.3, \alpha_t = 0.9^t \alpha_{t-1}$, and iterated for 40 epochs. An outer decay on the step size was added for faster convergence, the initial α_0 becomes $0.94 \cdot \alpha_0$ every time a demonstration is presented. The gradient, g_t is normalized to be $g_t = g_t \frac{g_t}{||g_t||_\infty}$ and the calculated step is taken if: $cW_t\left(\mu(\hat{\pi}_c^t) - \mu(\pi_c^*)\right) > cW_{t+1}\left(\mu(\hat{\pi}_c^{t+1}) - \mu(\pi_c^*)\right)$, where $\hat{\pi}_c^t$ denotes the optimal policy for a context c according to W_t.

ES: The algorithm was executed with the parameters: $\sigma = 10^{-3}, m = 250, \alpha = 0.1$ with decay rate of 0.95, for 50 iterations which didn't iterate randomly over one the contexts, but rather used the entire training set (all of the observed contexts and expert demonstrations up to the current time-step) for each step. The matrix was normalized according to $|| \cdot ||_2$, and so was the step calculated by the ES algorithm, before it was multiplied by α and applied.

Online setting

This section describes the experiments of the online setting (Sect. 4.3.2). All of the compared methods minimize the same objective, where the subgradients for the descent direction are computed using either the feature expectations (**feature expectations** setup) or expert trajectories of length 40 (**trajectories** setup). In this framework at every iteration we sample one context and its corresponding feature expectations (or trajectory, sampled from the expert policy), and take one descent step according to it. The mapping from context to reward W is linear, and projected to the EW algorithm requirement to be in the $dk - 1$ simplex. In the autonomous driving simulation: $W^* = \begin{pmatrix} 0.043 & 0 & 0.043 \\ 0 & 0.434 & 0 \\ 0.043 & 0.434 & 0 \end{pmatrix}$.

Hyper-parameter selection and adjustments: The PSGD and EW algorithms are configured as the theory specifies, where each descent step is calculated from the one sample. The ES algorithm is applied with the parameters $\sigma = 10^{-3}, m = 500, \alpha = 0.1$ with decay rate 0.95, for every iteration. The ES implementation include a special enhancement; a descent step is taken if the objective function value decreases (after the descent step).

Sepsis treatment

For the sepsis treatment we construct two environments, one for simulation purposes - simulator, and another for evaluation on real-life data. The experiments with the simulator presented in Sect. 4.4, the evaluation on real-life data presented in Sect. 4.5.

Sepsis treatment simulator

As described in Sect. 4.4 we use the processed data from Jeter et al. (2019). It consists of 5366 trajectories, each representing the sequential treatment provided by a clinician to a patient. At each time-step, the available information for each patient consists of 8 static measurements and 41 dynamic measurements. In addition, each trajectory contains the reported actions performed by the clinician (the number of fluids and vasopressors given to a patient at each time-step and binned to 25 different values), and there is a mortality signal which indicates whether the patient was alive 90 days after his hospital admission.

In order to create a tabular CMDP from the processed data, we separate the static measurements of each patient and keep them as the context. We cluster the dynamic measurements using K-means (MacQueen et al. 1967). Each cluster is considered a state and the coordinates of the cluster centroids are taken as its features $\phi(s)$. We construct the transition kernel between the clusters using the empirical transitions in the data given the state and the performed actions. Two states are added to the MDP and the feature vector is extended by 1 element, corresponding to whether or not the patient died within the 90 days following hospital release. This added feature receives a value of 0 on all non-terminal states, a value of -0.5 for the state representing the patient's death and 0.5 for the one representing survival. In addition, as the number of trajectories is limited, not all state-action pairs are represented in the data. In order to ensure the agent does not attempt to perform an action for which the outcome is unknown, we add an additional terminal state. At this state, all features are set to -1 to make it clearly distinguishable from all other states in the CMDP.

In our simulator, we used the same structure as the raw data, i.e., we used the same contexts found in the data and the same initial state distribution. Each context is projected onto the simplex and the expert's feature expectations for each context are attained by solving the CMDP. While we focus on a simulator, as it allows us to analyze the performance of the algorithms, our goal is to have a reward structure which is influenced by the data. Hence, we produce W^* by running the ellipsoid algorithm on trajectories obtained from the data. As done in the autonomous driving simulation, we average our results over 5 different seeds. The test data size in this domain is 300.

Online setting

Similarly to the autonomous driving simulation, there are two setups. For the **trajectories** setup we use expert trajectories of length 40. Again, the mapping from context to reward W is linear, and projected to the EW algorithm requirement to be in the $dk-1$ simplex (the true mapping can be found in the supplementary code at http://www.github.com/coirl/coirl_code).

Hyper-parameter selection and adjustments: The PSGD method is configured as specified by the theory, where each descent step is calculated from one sample.

Offline setting

The offline setting evaluates the methods' performance on a limited train data set. In this framework, at every iteration we sample a mini-batch of contexts (from a finite set) and their corresponding trajectories (sampled from the expert policy) then taking one descent step according to them. We conduct two experiments that evaluate the performance on a fixed-size data set. First, we consider a **linear** mapping, followed by an analysis of the convergence when a DNN estimator of the reward is used, when the mapping is **non-linear**. Of the various COIRL methods, for these experiments, we focus on PSGD, as it is less restrictive on $\mathcal{W}$. In all experiments the train data consists of pairs of context and feature expectations of a trajectory of length 40.

We evaluate the PSGD method and the GPI method (using the mapping calculated by PSGD) along with BC. The evaluation is done after convergence on a changing train data size, measured as 'contexts', which refer to the number of expert trajectories given (one per context). In the non-linear setting The non-linear model of PSGD implemented by a DNN with the context as its input, three layers with a leakyReLU activation and batch-normalization, each one of size 336. BC in both environments implemented by a DNN that has three layers of sizes 250,125,25 respectively, a leakyReLU activation between the first and second layers and a Softmax activation on the output to ensure a probability vector.

Hyper-parameter selection and adjustments: In the **linear** setting the PSGD algorithm is configured with step-size $\alpha_t = 0.25 \cdot 0.95^t$, the mini-batch size is 10, similarly to the autonomous driving simulation. In this domain the stopping criteria is 60 iterations.

The **non-linear** setting computes the descent direction by backpropagation of the sub-gradient. Each descent step is calculated over a mini-batch of size 32, where the step-size is $\alpha_t = 0.3 \cdot 0.96^t$. We measure the results for 200 iterations. The train data consists 4000 contexts. The true mapping is defined by $f^*(c) = r_1 \text{ if} age > 0.1, \text{and} r_2$ otherwise, where **age** refers to the normalized age of the patient, an element of the context vector.

Acknowledgements This research was partially supported by the ISF under contract 2199/20.

References

Abbeel, P., & Ng, A.Y.(2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning* (pp. 1). ACM.

Abbeel, P., & Ng, A.Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05 (pp. 1-8). New York, NY, USA: Association for Computing Machinery. ISBN 1595931805. https://doi.org/10.1145/1102351.1102352.

Amin, K., Jiang, N., & Singh, S. (2017). Repeated inverse reinforcement learning. *Advances in Neural Information Processing Systems,* 1815–1824.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor features for transfer in reinforcement learning. *Advances in neural information processing systems,* 4055–4065.

Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters, 31,* 167–175.

Berngard, S. C., Beitler, J. R., & Malhotra, A. (2016). Personalizing mechanical ventilation for acute respiratory distress syndrome. *Journal of thoracic disease, 8*(3), E172.

Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society, 48*(3), 334–334.

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

Boyd, S. P., & Barratt, C. H. (1991). *Linear controller design: Limits of performance*. Hoboken: Prentice Hall Englewood Cliffs.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning, 8*(3–4), 231–357.

Chakraborty, B., & Murphy, S. A. (2014). Dynamic treatment regimes. *Annual Review of Statistics and its Application, 1,* 447–464.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 1126–1135). JMLR. org.

Garber, D., & Hazan, E. (2016). A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization, 26*(3), 1493–1528.

Ghasemipour, S. K. S., Gu, S. S., & Zemel, R. (2019). Smile: Scalable meta inverse reinforcement learning through context-conditional policies. *Advances in Neural Information Processing Systems,* 7879–7889.

Hallak, A., Di Castro, D., & Mannor, S. (2015). Contextual markov decision processes. *arXiv preprint*arXiv:1502.02259.

Hazan, E. (2016). Introduction to online convex optimization. *Foundations and Trends® in Optimization, 2*(3–4), 157–325.

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

Itenov, T., Murray, D., & Jensen, J. (2018). Sepsis: Personalized medicine utilizing 'omic'technologies–a paradigm shift? In *Healthcare* (pp. 111). Multidisciplinary Digital Publishing Institute.

Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization.

Jeter, R., Josef, C., Shashikumar, S., & Nemati, S. (2019). Does the "artificial intelligence clinician" learn optimal treatment strategies for sepsis in intensive care?. URL https://github.com/point85AI/Policy-Iteration-AI-Clinician.git.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-W.H., Feng, M., Ghassemi, M., et al. (2016). Mimic-iii, a freely accessible critical care database. *Scientific Data, 3,* 160035. https://doi.org/10.1038/sdata.2016.35.

Juskalian, R., Regalado, A., Orcutt, M., Piore, A., Rotman, D., Patel, N. V., Lichfield, G., Hao, K., Chen, A., & Temple, J. (2020). Mit technology review. URL https://www.technologyreview.com/lists/technologies/2020/.

Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning. *International conference on Machine learning,* 267–274.

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning, 49*(2–3), 209–232.

Komorowski, M., Celi, L. A., Badawi, O., Gordon, A. C., & Faisal, A. A. (2018). The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine, 24*(11), 1716.

Laskey, M., Lee, J., Hsieh, W., Liaw, R., Mahler, J., Fox, R., & Goldberg, K. (2017). Iterative noise injection for scalable imitation learning. *arXiv preprint*arXiv:1703.09327 .

Lee, D., Srinivasan, S., & Doshi-Velez, F. (2019). Truly batch apprenticeship learning with deep successor features. *arXiv preprint*arXiv:1903.10077 .

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297). Oakland, CA, USA.

Modi, A. & Tewari, A. (2019). Contextual markov decision processes using generalized linear models. *arXiv preprint*arXiv:1903.06187 .

Modi, A., Jiang, N., Singh, S., & Tewari, A. (2018). Markov decision processes with continuous side information. *Algorithmic Learning Theory,* 597–618.

Nemirovsky, A. S., & Yudin, D. B. (1983). *In Problem complexity and method efficiency in optimization*. New York: Wiley.

Nesterov, Y., & Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics, 17*(2), 527–566.

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. *ICML, 1,* 2.

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems,* pp. 305–313.

Prasad, N., Cheng, L.-F., Chivers, C., Draugelis, M., & Engelhardt, B. E. (2017). A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *UAI*.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. London: John Wiley & Sons.

Ratliff, N., Bagnell, J. A., & Srinivasa, S. S. (2007). Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots* (pp. 392–397). IEEE.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of Mathematical Statistics, 22,* 400–407.

Ross, S., & Bagnell, D. (2010). Efficient reductions for imitation learning. *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 661–668).

Ross, S., Gordon, G., & Bagnell, D. (2011). 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* (pp. 627–635).

Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint*arXiv:1703.03864 .

Syed, U., & Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. *Advances in Neural Information Processing Systems,* 1449–1456.

Wesselink, E., Kappen, T., Torn, H., Slooter, A., & van Klei, W. (2018). Intraoperative hypotension and the risk of postoperative adverse outcomes: a systematic review. *British Journal of Anaesthesia, 121,* 706–721.

Xu, K., Ratner, E., Dragan, A., Levine, S., & Finn, C. (2018). Learning a prior over intent via meta-inverse reinforcement learning. *arXiv preprint*arXiv:1805.12573 .

Zahavy, T., Cohen, A., Kaplan, H., Mansour, Y. (2020). Apprenticeship learning via frank-wolfe.

Zahavy, T., Cohen, A., Kaplan, H., & Mansour, Y. (2020). Average reward reinforcement learning with unknown mixing times. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial*. (**Intelligence**).

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (pp. 928–936).