Inverse Reinforcement Learning in Contextual MDPs

Philip Korsunsky∗
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
philip.korsunsky@gmail.com

Stav Belogolovsky∗
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
stav.belo@gmail.com

Tom Zahavy∗
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
tomzahavy@campus.technion.ac.il

Chen Tessler
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
chen.tessler@campus.technion.ac.il

Shie Mannor
Department of Electrical Engineering
Technion Israel Institute of Technology
Haifa, Israel 3200003
shie@ee.technion.ac.il

∗equal contribution

Abstract

We consider the Inverse Reinforcement Learning (IRL) problem in Contextual Markov Decision Processes (CMDPs). Here, the reward of the environment, which is not available to the agent, depends on a static parameter referred to as the context. Each context defines an MDP (with a different reward signal), and the agent is provided demonstrations by an expert, for different contexts. The goal is to learn a mapping from contexts to rewards, such that planning with respect to the induced reward will perform similarly to the expert, even for unseen contexts. We suggest two learning algorithms for this scenario. (1) For rewards that are a linear function of the context, we provide a method that is guaranteed to return an $\epsilon$-optimal solution after a polynomial number of demonstrations. (2) For general reward functions, we propose black-box descent methods based on evolutionary strategies capable of working with nonlinear estimators (e.g., neural networks). We evaluate our algorithms in autonomous driving and medical treatment simulations and demonstrate their ability to learn and generalize to unseen contexts.

1 Introduction

We study a sequential decision-making problem where the environment is a Markov Decision Process (MDP), but the dynamics and the reward depend on a static parameter referred to as the context. For example, consider a lifelong learning task, in which an autonomous driving car must navigate the road while avoiding other vehicles. The car is likely to incur similar instances of this problem with different conditions, such as visibility and weather. While we expect some tasks to be similar, the agent is also required to adapt. For instance, when encountering fog, the agent is expected to drive more safely and slowly than during optimal weather conditions.
Another example is the dynamic treatment regime [Chakraborty and Murphy, 2014]. Here, there is a sick patient and a clinician who acts to improve the patient’s health. The state space of the MDP is composed of the patient’s clinical measurements, and the actions are the clinician’s decisions. Traditionally, treatments are targeted to the “average” patient. Instead, in personalized medicine, people are separated into different groups and the medical decisions are tailored to the individual patient based on the predicted response or risk of disease (Fig. 1b). Recently, with the cost of genetic sequencing dropping dramatically, and with the growth of patients that are willing to track and share their healthcare records, personalized medicine is being developed to match the specific needs of patients. One success story of personalized medicine was the development of a drug called Herceptin for a group of cancers termed HER2+ that are highly aggressive and often have a poor prognosis. Herceptin was diagnosed to treat women (the context) with HER2+ breast cancer and it improved their survival time from 20 months to 5 years.

For acute respiratory distress syndrome (ARDS), clinicians argue that treatment goals should rely on individual patients’ physiology [Berngard et al., 2016]. In [Wesselink et al., 2018], the authors study organ injury that might occur when mean arterial pressure decreases below a certain threshold, and report that this threshold varies across different patient groups.

These examples highlight the importance of patient information in the online treatment regime and motivate us to consider contextual information within the RL framework. One possibility is to expand the state such that it will include the patient information (the context). However, this approach can increase the complexity of the problem significantly, as the set of possible MDPs grows exponentially with the dimension of the context. Therefore, in the Contextual MDP framework [Hallak et al., 2015], the goal is to learn the mapping from contexts to the environment (dynamics and rewards). A learning algorithm for this problem should learn a mapping that generalizes to unseen contexts, improves at that task as it observes contexts, and achieves desired sample complexity [Modi et al., 2018] or regret [Modi and Tewari, 2019].

Another issue that is prevalent in real-world problems is that the reward function may be sparse and misspecified. For example, in online treatment problems like sepsis [Komorowski et al., 2018], the only available signal is the mortality of the patient at the end of the treatment. Manually designing a reward function for this problem is complicated and could lead to poor performance [Raghu et al., 2017] [Lee et al., 2019]. In many cases, it is easier for humans to define the reward implicitly by providing demonstrations of what constitutes a proper treatment. Inverse Reinforcement Learning [Ng et al., 2000] IRL] is concerned with inferring the reward function by observing an expert in order to find a policy that guarantees a value that is close to that of the expert.

Finally, deploying RL algorithms to treat patients or to drive cars cannot be regarded in the same way as solving a video game due to safety considerations and lack of a simulator. To address these issues, we propose the Contextual Inverse Reinforcement Learning (COIRL) model. We study a safe, online learning framework where an expert supervises the RL algorithm as follows. The agent observes a context, estimates the reward and proposes a policy. The expert evaluates the agent’s actions and decides if they are $\epsilon$-optimal. If not, the expert provides a demonstration to the agent. The goal of the agent is to learn a mapping from contexts to rewards by observing expert demonstrations.

\footnote{Source: bit.ly/2P6iCRr}
We design and analyze two algorithms for COIRL: (1) For linear reward mappings, we study the ellipsoid method, for which we provide theoretical guarantees and analyze the sample complexity for finding an $\epsilon$-optimal solution. (2) For nonlinear mappings, we study a black-box optimization solution that minimizes a surrogate loss using evolutionary strategies [Salimans et al., 2017]. We consider two loss functions that enable feature expectation matching; an intuitive but non-differentiable loss that minimizes the distance of the value of the agent from the value of the expert, and a differentiable loss that is based on a min-max objective. We evaluate our algorithms in autonomous driving and online treatment simulators and demonstrate their ability to generalize to unseen contexts.

2 Problem formulation and notation

Contextual Markov Decision Processes (CMDPs): A Markov Decision Process (MDP) is defined by the tuple $(S, A, P, \xi, R)$ where $S$ is the state space, $A$ the action space, $P : S \times S \times A \rightarrow [0, 1]$ the transition kernel, $\xi$ the initial state distribution and $R : S \rightarrow \mathbb{R}$ the reward function. A Contextual MDP (CMDP) is an extension of an MDP, and is defined by $(C, S, A, M)$ where $C$ is the context space, and $M$ is a mapping on $C$ such that $M(c)$ is an MDP with space and action spaces $S, A$ for each $c \in C$. We consider a CMDP with finite state and action spaces, and associate each state with a feature vector $\phi : S \rightarrow [0, 1]^k$. Additionally, we assume that the transitions $P$ and the initial state distribution $\xi$ are not context dependent.

In our work we will further assume a linear setting, in which the reward function for context $c$ is linear in the state features: $R^c(s) = r^c_T \phi(s)$, where $r^c$ is the rewards coefficients vector which is given by the linear mapping $r^c = c^T W^*$, where $W^* \in \mathbb{R}^d \times k$. We assume $||W^*||_\infty \leq 1$, and $c \in C = \Delta_{d-1}$, i.e., the standard $d - 1$ dimensional simplex. This allows a straight-forward expansion to a model in which the transitions are also given by a linear mapping of the context, as seen in [Modi et al., 2018]. One way of viewing this model is that each row in the mapping $W^*$ is a base rewards coefficient vector, and the reward for a specific context is a convex combination of these base rewards.

We consider deterministic policies $\pi : S \rightarrow A$ which dictate the agent’s behaviour at each state. The (normalized) value of a policy $\pi$ for reward coefficients vector $r$ is: $V^\pi_r = \frac{1}{k} E_{\xi, P, \pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right] = r^T \frac{1}{k} E_{\xi, P, \pi} \left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right] = r^T \mu(\pi)$ where $\mu(\pi) := \frac{1}{k} E_{\xi, P, \pi} \left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right] \in \mathbb{R}^k$ is called the feature expectations of $\pi$, and $\gamma \in [0, 1)$ is the discount factor. For the optimal policy with respect to (w.r.t.) the reward coefficients vector, we denote the value by $V_r^*$. The normalization of the value function by the constant $\frac{1}{k}$ is for convenience, i.e. to claim $\forall \pi : ||\mu(\pi)||_1 \leq 1$, and does not affect the resulting policies. Given $W^*$, for each context $c \in C$ we may calculate the reward coefficients vector $r^*_c$ and find the optimal policy, i.e. the policy with the highest value, using standard methods such as policy/value iteration.

Inverse Reinforcement Learning in CMDPs:

In standard IRL the goal is to learn a reward which best explains the behavior of an observed expert. The model describing this scenario is the MDPR which is an MDP without a reward function (also commonly called a controlled Markov chain). Similarly, we denote a CMDP without a mapping of context to reward by CMDPDM. The goal in Contextual IRL (COIRL) is to infer a mapping $W$ from observations of an expert, which will induce near-optimal policies for all contexts. As shown in [Ng et al., 2000], the IRL problem is ill-defined and we aren’t ensured to learn the real reward or, in this case, the mapping $W^*$; however, it is still possible to learn a mapping which induces $\epsilon$-optimal policies and enables generalization to new contexts.

While learning a transition kernel and an initial distribution which are parametrized by the context is related to the IRL problem, it can be seen as a separate, precursory problem, allowing us to make the simplifying assumptions presented previously. By using existing methods to learn the mappings for the transition kernel and initial distribution in a contextual model, such as in [Modi et al., 2018], and by using the simulation lemma [Kearns and Singh, 2002], our results can be extended to a more general CMDP setting.

3 Methods

In this section, we study learning algorithms for COIRL that are motivated by the online treatment regime. We begin with an online learning framework, where we design algorithms that do not have access to a simulator of the environment, and the agent is only allowed to explore near-optimal actions. We then consider an offline learning framework, where observational (off-policy) data of expert
demonstrations was collected a priori. For example, in the medical domain, these demonstrations may represent collected data of clinicians treating patients. Such data is publicly available, for example, in the MIMIC-III data set [Johnson et al., 2016]. Finally, we consider a warm start framework where the agent policy is initialized in the offline framework and continues to learn in the online framework.

More explicitly, in the online framework, the agent learns under the supervision of an expert. We propose a setting, in which at each time-step $t$ a new context $c_t$ is revealed, possibly adversarially, and the agent acts based on the optimal policy w.r.t. its estimated mapping $W_t$, denoted by $\pi_t$. The expert provides two forms of supervision for the agent. First, the expert evaluates the agent’s behavior and produces a binary signal which determines if the agent’s policy is $\epsilon$-optimal, i.e., $V_{c_t}^* W^* - V_{c_t}^{\pi_t} W^* \leq \epsilon$; Second, when the agent is sub-optimal, the expert provides a demonstration in the form of its policy (or feature expectations) for $c_t$. The goal is to learn a mapping which induces $\epsilon$-optimal policies for all contexts based on a minimal number of examples from the expert.

Next, we present two approaches to solving the COIRL problem. We begin with the linear model, for which we propose an ellipsoid-based approach with proven polynomial upper bounds. We then consider nonlinear models and propose descent-based algorithms.

### 3.1 Ellipsoid algorithms for COIRL

The goal of the algorithms in this section is to find a linear mapping $W^*$ from contexts to rewards by observing expert demonstrations. We study ellipsoid-based algorithms that maintain an ellipsoid-shaped feasibility set that contains $W^*$. At any step, the current estimation $W_t$ of $W^*$ is defined as the center of the ellipsoid, and the agent acts optimally w.r.t. this estimation. If the agent performs sub-optimally, the expert provides a demonstration in the form of the optimal feature expectations for $c_t$, $\mu(\pi_t^*)$. The 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 (a well-known result [Boyd and Barratt, 1991]) until its center is a mapping which induces $\epsilon$-optimal policies. Theorem 1 shows that this algorithm achieves a polynomial upper bound on the number of sub-optimal time-steps.

Finally, notice that in Algorithm 1 we use an underline notation to denote a “flattening” operator for matrices, and $\odot$ to denote a composition of an outer product and the flattening operator. The proof for Theorem 1 is provided in the supplementary material, and is adapted from [Amin et al., 2017] to the COIRL problem.

**Theorem 1.** In the linear setting where $R^*_c(s) = e^T W^* \phi(s)$, for an agent acting according to Algorithm 1, the number of rounds in which the agent is not $\epsilon$-optimal is $O(d^2k^2 \log(k/d))$.

**Practical ellipsoid algorithm:** In many real-world scenarios, the expert cannot evaluate the value of the agent’s policy and cannot provide its policy or feature expectations. To address these issues, we consider a relaxed approach, in which the expert evaluates a single trajectory of the agent and cannot provide its policy or feature expectations. To address these issues, we consider an alternative approach, in which the expert evaluates a single trajectory of the agent and cannot provide its policy or feature expectations. To address these issues, we consider an alternative approach, in which the expert evaluates the agent’s policy and cannot provide its policy or feature expectations. To address these issues, we consider an alternative approach, in which the expert evaluates the agent’s policy and cannot provide its policy or feature expectations. To address these issues, we consider an alternative approach, in which the expert evaluates the agent’s policy and cannot provide its policy or feature expectations. To address these issues, we consider an alternative approach, in which the expert evaluates the agent’s policy and cannot provide its policy or feature expectations.

**Sub-optimal experts:** In addition, we relax the requirement that the expert must be optimal 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

---

**Algorithm 1 Online ellipsoid algorithm for COIRL**

| Initialize: $\Theta_0 \leftarrow B_{\infty}(0,1) = \{x \in \mathbb{R}^d : ||x||_{\infty} \leq 1\}$
| $\Theta_1 \leftarrow$ MVEE($\Theta_0$)
| while stopping conditions are not met do
| \hspace{1cm} Observe $c_t$, let $W_{t-1}$ be the center of $\Theta_t$
| \hspace{1cm} Play episode using $\pi_t = \arg \max_\pi V^*_{c_t} W_t$
| \hspace{1cm} if $V^*_{c_t} W^* - V^*_{c_t} W_t > \epsilon$ then
| \hspace{2cm} $\mu(\pi_t^*)$ is revealed
| \hspace{2cm} Let $a_t = c_t \odot (\mu(\pi_t^*) - \mu(\pi_t))$
| \hspace{2cm} $\Theta_{t+1} \leftarrow$ MVEE ($\{\theta \in \Theta_t : \theta^T a_t \geq W_t^T a_t\}$)
| \hspace{1cm} else
| \hspace{2cm} $\Theta_{t+1} \leftarrow \Theta_t$

---

2 This procedure follows a sequence of linear algebra operations which we explain in the appendix.
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 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 \( \tilde{H} \) 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. In the supplementary material, we provide pseudo code for this process (Algorithm 3). Theorem 2 below upper bounds the number of sub-optimal actions that Algorithm 3 chooses.

**Theorem 2.** For an agent acting according to Algorithm 3, with probability of at least \( 1 - \delta \), for \( H = \left\lfloor \frac{\log(\frac{2}{1-\gamma})}{1-\gamma} \right\rfloor \) and \( n = \left\lfloor \frac{4k \log(\frac{4k}{\delta})}{\epsilon^2} \right\rfloor \), if \( \forall t : W^*_t \in B_\infty(W^*, \frac{\epsilon}{2}) \cap \Theta_0 \) the number of rounds in which a sub-optimal action is played is \( O\left( \frac{d^2k^2}{\epsilon^2} \log \left( \frac{d}{\delta} \log \left( \frac{4k}{\epsilon} \right) \right) \right) \).

The proof for Theorem 2 is provided in the supplementary material, and is adapted from [Amin et al., 2017] to the setup of COIRL with near optimal experts.

**Warm-start for the ellipsoid algorithm:** In this setup, the goal is to use offline data to initiate the ellipsoid algorithm with a smaller feasibility set. Although this approach leads to lesser regret, similarly to the online setting and in order to ensure the optimal solution remains within the feasibility set, an expert’s supervision is required for training. We simulate the online setting by iterating over the trajectories in the data. The expert evaluates the agent w.r.t. this mapping. This allows the agent to learn from different experts, and as each trajectory is an expert demonstration, we use it as an alternative to the online expert demonstration. By adhering to the conditions of Theorem 2, its theoretical guarantees remain.

### 3.2 Optimization methods for COIRL with nonlinear mappings

In the previous section, we analyzed a scenario in which the mapping from contexts to rewards was linear, i.e. \( R^*_c(s) = c^T W^* \phi(s) \). This reward structure enabled the analysis of the sample complexity of the ellipsoid algorithm and guaranteed its convergence. In this section, we extend the COIRL framework to nonlinear mappings, i.e. \( R^*_c(s) = f^*(c)^T \phi(s) \), where \( f^* \) is a nonlinear function. We formulate COIRL as an optimization problem and provide descent algorithms to solve it.

The goal is to find a mapping which induces policies that have feature expectations that match the expert’s feature expectations for any context, i.e., minimize \( ||\mu(\hat{f}_\theta(c)) - \mu(c^*_\pi) || \), an approach known as feature expectation matching [Abbeel and Ng, 2004; Ziebart et al., 2008]. However, minimizing such a loss is difficult, as it is piece-wise constant in \( f_\theta(c) \) (or \( W \) in the linear case). For this reason, we explore two surrogate loss functions (alternative loss functions whose minimization leads to feature expectation matching).

The first surrogate loss function is the MSE between the estimated value of the expert and the agent:

\[
L(\theta) = \sum_{j=1}^{D} \left( f_\theta(c_j)^T (\mu(\hat{c}_e) - \mu(c^*_\pi)) \right)^2, \tag{1}
\]

where \( c^*_\pi \) denotes the optimal policy w.r.t. \( R^*_c(s) \) and \( \hat{c}_e \) denotes the optimal policy w.r.t. \( f_\theta(c) \). Note that in order to evaluate the loss, we have to compute the optimal policies w.r.t. \( f_\theta(c) \), which involves solving tabular MDPs (e.g. with policy iteration). This fact makes Eq. (1) non-differentiable w.r.t. to \( \theta \) as solving an MDP is non-differentiable. On the other hand, the loss function is Lipschitz continuous w.r.t. \( f_\theta(c) \), as the following lemma states. The proof can be found in the supplementary material and is based on the simulation lemma [Kearns and Singh, 2002].

---

**Algorithm 2 Black-box algorithm for COIRL**

**Input:** \( \{c_i\}_{i=1}^D, \{\hat{c}_i\}_{i=1}^D \) - contexts and their respective expert feature expectations, \( \alpha \) - learning rate, \( \sigma \) - noise standard derivation, \( m \) - number of evaluations

**Define** \( L(\theta) \equiv \left( f_\theta(c)^T (\mu(c^*_\pi) - \mu(\hat{c}_e)) \right)^2 \)

**Initialize:** \( \theta \in \mathbb{R}^k \)

while stopping conditions are not met do

for \( c \in \{c_1, ..., c_D\} \) do

for \( j = 1, ..., m \) do

Sample \( \epsilon_j \sim N^k(0, \sigma^2) \)

Compute \( L_j = L(\theta + \epsilon_j) \)

Set \( d_L(\theta) = \sum_{j=1}^m L_j \epsilon_j, b = ||\hat{V}_c||_2 \)

if \( L(\theta + \frac{\epsilon}{\sqrt{d}} d_L(\theta)) < L(\theta) \) then

\( \theta \leftarrow (\theta + \frac{\epsilon}{\sqrt{d}} d_L(\theta)) \)

Decay learning rate \( \alpha \), noise parameter \( \sigma \)

return \( \theta \)
Lemma 1. The objective function \( f_\theta(c) \) is Lipschitz continuous in \( f_\theta(c) \).

To minimize Eq. (1), we design a black box algorithm (Algorithm 2) that is based on Evolution Strategies [Salimans et al., 2017; ES]; a gradient-free descent method for solving black-box optimization problems based on computing finite differences [Nesterov and Spokoiny, 2017]. The algorithm receives a set of \( D \) context-demonstration tuples and returns parameters \( \hat{\theta} \). At each step, a context-demonstration tuple is sampled at random. Next, a set of \( m \) random Gaussian noise vectors \( \epsilon_1, ..., \epsilon_m \) is sampled at random, and used to perturb \( \theta \) to yield a set of \( m \) reward functions \( f_{\theta + \epsilon_j} \). Each reward is used to evaluate the losses (Eq. (1)) \( L_j \). Finally, the descent direction \( d_{L_j}(\theta) \) is computed as a the sum of perturbed vectors, weighted by the losses \( d_{L_j}(\theta) = \sum L_j \epsilon_j \).

While the loss in Eq. (1) is intuitive, it has a few drawbacks. First, its evaluation requires solving \( L_j \), which can be inefficient. Second, it is difficult to minimize in some settings (see the experiments section for more details). For these reasons, we consider a second surrogate optimization problem, defined by:

\[
\hat{\theta} = \arg \min_{\theta} \max_{\hat{\pi}_c} \sum_{j=1}^{D} f_\theta(c_j)^T (\mu(\hat{\pi}_c) - \mu(\pi_c^*)),
\]

a similar problem to the IRL formulation in [Syed and Schapire, 2008; Ho and Ermon, 2016]. This approach requires a two-step optimization process. At each iteration: (1) given the current estimates \( f_\theta \), compute the optimal policies for \( \{c_j\}_{j=1}^{D} \) and their corresponding feature expectations. Then (2) given the feature expectations, perform ES on the loss and take a single step to update \( \theta \). On the positive side, this loss is differentiable, and can be optimized with standard backpropagation.

4 Experiments

This section is organized as follows. We begin with analyzing our approach on a common IRL task, an autonomous driving simulation [Abbeel and Ng, 2004; Syed and Schapire, 2008], adapted to the contextual setup. We then test our method in a medical domain, using a data set of expert (clinicians) trajectories for treating patients with sepsis.

More details will follow in the relevant subsections.

We experimented with the methods that we presented in the previous section, namely, the ellipsoid algorithm, and the ES method with losses Eq. (1) and Eq. (2). We evaluate and compare their cumulative regret, the number of demonstrations they require, and their ability to generalize to a holdout test set. In each experiment, we create a random sequence of contexts \( \{c_j\} \), average the results across several seeds and report the mean and the standard deviation. Note that once an algorithm achieves an \( \epsilon \)-optimal value in the online framework, it will stop requesting demonstrations from the expert. For that reason, algorithms that perform better request fewer contexts from the expert and their generalization graph appears truncated. We emphasize here that if a plot ends abruptly in these experiments, the reason is that at that point, the algorithm achieves an \( \epsilon \)-optimal value and stops requesting for demonstrations. For nonlinear reward model, we take \( f_\theta(c) \) to be a multilayer perceptron; for the ES methods, we use value iteration to compute optimal policies; all details and hyper-parameters can be found in the supplementary material.

4.1 Driving simulation

The driving task simulates a three-lane highway, in which there are two visible cars - car A and car B. The agent, controlling car A can drive both on the highway and off-road. Car B drives on 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 reward is defined to be linear in the feature expectations \( R_c^*(s) = r_c^T \phi(s) \), where \( \phi(s) \) is composed of 3 features: (1) a speed feature, (2) a collision feature, which is valued 0 in case of a collision and 0.5 otherwise, and (3) an off-road feature, which is 0.5 if the car is on the road and 0 otherwise. 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

\[ \text{The data and code that we used to construct these simulators, as well as the implementation of our algorithms can be found in} \]
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.

In this task, the context vector implies different priorities for the agent; should it prefer speed or safety? Is going off-road to avoid collisions a valid option? For example, an ambulance will prioritize speed and may allow going off-road as long as it goes fast and avoids collisions, while a bus will prioritize avoiding both collisions and off-road driving as structural integrity is its main concern. The optimal behavior is defined using a linear mapping $W^*$ or a nonlinear mapping $f : C \mapsto [-1, 1]^k$.

To demonstrate the effectiveness of our solutions, our mappings are constructed in a way that induces different behaviors for different contexts, making generalization a challenging task. For the nonlinear task, we consider two reward coefficient vectors $r_1$ and $r_2$, and define the mapping by $f^*(c) = r_1$ if $||c||_\infty \geq 0.55$, and $r_2$ otherwise.

Results: For the online linear setting (Fig. 3), we define the optimality threshold to be $\epsilon = 10^{-3}$ for all algorithms. We report the cumulative regret (Fig. 3a), the number of demonstrations that each algorithm requested (Fig. 3b), and their ability to generalize to a holdout test set (Fig. 3c), which were calculated using 20 seeds. Examining the results, we can see that despite the theoretical guarantees, the descent methods achieve better sample efficiency and regret than the ellipsoid. Also, Eq. (2) leads to better regret overall and requires significantly fewer demonstrations to reach $\epsilon$-optimal performance.

For the nonlinear online setting (Fig. 4) we compare the ES method for minimizing loss (2) with the ellipsoid algorithm, with $\epsilon = 10^{-3}$ across 5 seeds. These results demonstrate that the ellipsoid does not perform well in nonlinear settings, highlighted in the inability to generalize and the linear regret growth, while the ES method with a non-linear model is able to converge to a near-optimal solution.

Notably, loss (1) is excluded from the nonlinear results as it was unable to generalize and thus required a demonstration at nearly every time-step, making it about on par with the ellipsoid. A possible explanation for this is that this loss discourages advancing in the correct direction under certain circumstances. For example, consider the case where the agent’s coefficient for the speed feature is 0.1, and the agent’s and expert’s feature expectations are 0.5, 1 respectively. A speculative step increasing the coefficient to 0.2 may not be sufficient to change the agent’s feature expectations, and thus will increase the loss. On the other hand, an increase in the coefficient is necessary to match the feature expectations, therefore the step the ES algorithm takes would go in the opposite direction. Loss (2) avoids such issues, which may explain its superior results.
4.2 Dynamic treatment regime

In this setup, there is a sick patient and a clinician who acts to improve the patient’s medical condition. The context (static information) represents patient features which do not change during treatment, such as age and gender. The state of the agent summarizes the dynamic measurements of the patient throughout the treatment, such as blood pressure and EEG readouts. The action space, i.e., the clinician’s actions, consists of a sequence of decision rules, one per stage of intervention, and represent a combination of intervention categories. Dynamic treatment regimes are particularly useful for managing chronic disorders and fit well into the larger paradigm of personalized medicine [Komorowski et al., 2018, Prasad et al., 2017].

We focus on an intensive care task, where the agent needs to choose the right treatment for a patient that is diagnosed with sepsis. We use the MIMIC-III data set [Johnson et al., 2016] and follow the data processing steps that were taken in Jeter et al. [2019]. However, performing off-policy evaluation is not possible using this data-set, as it does not satisfy basic requirements [Gottesman et al., 2018, 2019]. Therefore, we designed a simulator of a CMDP, based on this data.

The data-set consists of 5366 trajectories. Each trajectory represents a sequential treatment that was provided by a clinician to a patient. The available information for each patient consists of $d = 8$ static features (the context, e.g. gender, age), $k = 42$ dynamic measurements of the patient at each time step (e.g. heart rate, body temperature). In addition, each trajectory contains the reported clinician actions (the amount of fluids and vasopressors given to a patient at each time-step and binned to 25 different values), and a mortality signal which indicates whether the patient was alive 90 days after his hospital admission. In order to create a tabular MDP, we cluster the dynamic features 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. As in the previous sections, we consider a reward which is linear in $W$, i.e., $R^*_c(s) = c^T W^* \phi(s)$, where $W^* \in \mathbb{R}^8 \times 42$ is a matrix we construct from the data for the simulator. In the simulator, the expert acts optimally w.r.t. this $W^*$.

When treating a sepsis patient, the clinician has several decisions to make. One such decision is whether or not to provide a patient with vasopressors, drugs which are commonly applied to restore and maintain blood pressure in patients with sepsis. However, what is regarded as normal blood pressure differs based on the age and weight of the patient [Wesselink et al., 2018]. In our setting, $W$ captures this information, as it maps from contextual information (age) and dynamic information (blood pressure) to reward.

Results: Here we compare our algorithms within the online framework, over 1000 time-steps, where $\epsilon = 5 \times 10^{-4}$, across 5 seeds for all algorithms. Similarly to the autonomous vehicle experiments, we measure the regret (Figure 5a) and the number of demonstrations that each algorithm requested (Figure 5b). In addition to generalization to a holdout set in Figure 5c, we provide results for the in-accuracy (miss rate) of the agents, i.e., in how many states the policy of the agent differs from that of the expert. These results suggest that in this more complicated environment, the ES approaches perform even better compared to the ellipsoid method. While all algorithms are able to learn and generalize, both ES approaches require significantly fewer demonstrations and accumulate less regret. We also note that although the miss rate decreases over time, it does not go below 11% for any of the methods. This shows that while the accuracy metric is indicative of good performance, it may not be a good metric when evaluating policies learned through IRL, as it only measures the ability to imitate the expert rather than the ability to learn the latent contextual reward structure.
5 Discussion

We studied the COIRL problem with linear and nonlinear reward mappings. While nonlinear mappings are more appropriate to model real-world problems, for a linear mapping, we were able to provide theoretical guarantees and sample complexity analysis. Moreover, when applying AI agents to real-world problems, interpretability of the learned model is of major importance, in particular when considering deployment in medical domains [Komorowski et al., 2018]. Interpretability of linear models can be achieved by analyzing the mapping $W$ and providing insights on the importance of specific features for specific contexts.

We experimented with two approaches for COIRL in the linear setup - the Ellipsoid and the ES methods. While the Ellipsoid has theoretical guarantees, we observed that ES performed better in all of our experiments. This raises an important question - what is the lower bound on the number of samples required? In [Amin et al., 2017], the ellipsoid method was proposed in a non-contextual IRL setup, and was shown to achieve a sample complexity of $d^2 \log(1/\epsilon)$ while the lower bound is $d \log(1/\epsilon)$. This may explain the fact that ES achieved better performance than the ellipsoid, even in the linear setup, although we cannot analyze its performance.

Finally, the literature on contextual MDPs is concerned with providing theoretical guarantees and sample complexity analysis for the scenario in which we can model each patient as a tabular MDP. However, when the measurements of the patient are continuous, deep learning methods are likely to perform better than state aggregation. In the deep setup, the critical question is how to design an architecture that will leverage the structure of the static and dynamic information. While there has been some preliminary work in robotics domains [Xu et al., 2018], these works often focus on meta-learning, i.e., few-shot adaptation, whereas COIRL considers the zero-shot scenario.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1. ACM, 2004.

Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. In Advances in Neural Information Processing Systems, pages 1815–1824, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

S Clark Berngard, Jeremy R Beitler, and Atul Malhotra. Personalizing mechanical ventilation for acute respiratory distress syndrome. Journal of thoracic disease, 8(3):E172, 2016.

Stephen P Boyd and Craig H Barratt. Linear controller design: limits of performance. Prentice Hall Englewood Cliffs, NJ, 1991.

Bibhas Chakraborty and Susan A Murphy. Dynamic treatment regimes. Annual review of statistics and its application, 1:447–464, 2014.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 854–863. JMLR. org, 2017.

Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. arXiv preprint arXiv:1805.12298, 2018.

Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. Nature medicine, 25(1):16–18, 2019.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. arXiv preprint arXiv:1502.02259, 2015.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In Advances in Neural Information Processing Systems, pages 4565–4573, 2016.
Theis Itenov, Daniel Murray, and Jens Jensen. Sepsis: Personalized medicine utilizing ‘omic’ technologies—a paradigm shift? In Healthcare, page 111. Multidisciplinary Digital Publishing Institute, 2018.

Russell Jeter, Christopher Josef, Supreeth Shashikumar, and Shamim Nemati. Does the "artificial intelligence clinician" learn optimal treatment strategies for sepsis in intensive care?, 2019. URL https://github.com/point85AI/Policy-Iteration-AI-Clinician.git.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii, a freely accessible critical care database. Scientific Data, 3:160035, May 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.35. URL http://dx.doi.org/10.1038/sdata.2016.35.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. Machine learning, 49(2-3):209–232, 2002.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. Nature Medicine, 24(11):1716, 2018.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. arXiv preprint arXiv:1903.10077, 2019.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pages 281–297. Oakland, CA, USA, 1967.

Aditya Modi and Ambuj Tewari. Contextual markov decision processes using generalized linear models. arXiv preprint arXiv:1903.06187, 2019.

Aditya Modi, Nan Jiang, Satinder Singh, and Ambuj Tewari. Markov decision processes with continuous side information. In Algorithmic Learning Theory, pages 597–618, 2018.

Yuriii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. Foundations of Computational Mathematics, 17(2):527–566, 2017.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In Icml, volume 1, page 2, 2000.

Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. UAI, 2017.

Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. arXiv preprint arXiv:1711.09602, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.05864, 2017.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In Advances in neural information processing systems, pages 1449–1456, 2008.

EM Wesselink, TH Kappen, HM Torn, AIC Slooter, and WA van Klei. Intraoperative hypotension and the risk of postoperative adverse outcomes: a systematic review. British journal of anaesthesia, 2018.

Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. arXiv preprint arXiv:1805.12573, 2018.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In Aaai, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.
A Ellipsoid Algorithm for trajectories

Algorithm 3 Batch ellipsoid algorithm for COIRL

Initialize: \( \Theta_0 \leftarrow B_\infty(0,1) = \{ x \in \mathbb{R}^{d \times k} : ||x||_\infty \leq 1 \} \)
\( \Theta_1 \leftarrow \text{MVEE}(\Theta_0) \)
\( i \leftarrow 0, Z \leftarrow 0, Z^* \leftarrow 0 \)

for \( t = 1, 2, 3, \ldots \) do

\( c_t \) is revealed. Let \( W_t \) be the center of \( \Theta_t \)

Play episode using \( \hat{\pi}_t = \arg \max_\pi V_{c_t}^* W_t \)

\( \Theta_{t+1} \leftarrow \Theta_t \)

if a sub-optimal action \( a \) is played at state \( s \) then

Expert provides H-step trajectory \((s_0^\gamma, s_1^\gamma, \ldots, s_H^\gamma)\). Let \( \hat{x}_t^\gamma \) be the H-step sample of the expert’s feature expectations for \( x_t^\gamma = 1: \hat{x}_t^\gamma = \frac{1}{k} \sum_{h=0}^{H} \gamma^h \phi(s_h^\gamma) \)

Denote \( z_t = c_t \circ x_t, \tilde{z}_t^\gamma = c_t \circ \hat{x}_t^\gamma \)

\( i \leftarrow i + 1, Z \leftarrow Z + z_t, Z^* \leftarrow Z^* + \tilde{z}_t^\gamma \)

if \( i = n \) then

\( \Theta_{t+1} \leftarrow \text{MVEE} \left( \left\{ \theta \in \Theta_t : \left( \theta - W_t \right)^T \cdot \left( \frac{Z^*}{n} - \frac{Z}{n} \right) \geq 0 \right\} \right) \)

\( i \leftarrow 0, Z \leftarrow 0, Z^* \leftarrow 0 \)

B MVEE computation

This computation is commonly found in optimization lecture notes and textbooks. First, we define an ellipsoid by \( \{ x : (x - c)Q^{-1}(x - c) \leq 1 \} \) for a vector \( c \), the center of the ellipsoid, and an invertible matrix \( Q \). Our first task is computing \( \Theta_1 \), the MVEE for the initial feasibility set \( \Theta_0 = B_\infty(0,1) = \{ x \in \mathbb{R}^{d \times k} : ||x||_\infty \leq 1 \} \). The result is of course a sphere around 0: \( c_1 = 0, Q_1 = dkI \).

For the update \( \Theta_{t+1} \leftarrow \text{MVEE} \left( \left\{ \theta \in \Theta_t : \left( \theta - W_t \right)^T \cdot a_t \geq 0 \right\} \right) \), we define \( \tilde{a}_t = \frac{1}{\sqrt{a_t^T Q_t a_t}} a_t \)

and calculate the new ellipsoid by \( c_{t+1} = c_t - \frac{1}{dk+1} Q_t \tilde{a}_t, Q_{t+1} = \frac{d^2 k^2}{dk+1} (Q_t - \frac{2}{dk+1} Q_t \tilde{a}_t \tilde{a}_t^T Q_t) \).

C Proof of Theorem 1

For simpler analysis, we define a “flattening” operator, converting a matrix to a vector: \( \mathbb{R}^{d \times k} \rightarrow \mathbb{R}^{d \times k} \)

by \( W = [w_1, \ldots, w_{d \times k}] \). We also define the operator \( \circ \) to be the composition of the flattening operator and the outer product: \( u \circ v = [u_1 v_1, \ldots, u_{d \times k} v_{d \times k}] \).

Therefore, the value of policy \( \pi \) for context \( x \) is given by \( V_{c_t}^* W^* \mu(\pi) = W^T (c_t \mu(\pi)) \), where \( ||W^*||_\infty \leq 1, ||c_t \mu(\pi)||_1 \leq 1 \).

Lemma 2 [Boyd and 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 \( \frac{V_{c_t}^* W^*}{V_{c_t}^* W^*} \leq \frac{\epsilon}{||\pi^* - \hat{\pi}_t||_1} \).

Proof of Theorem 2. We prove the theorem by showing that the volume of the ellipsoids \( \Theta_t \) for \( t = 1, 2, \ldots \) is bounded from below. In conjunction with Lemma 2, 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 \( W^* \) always remains in the ellipsoid. We note that in rounds where \( V_{c_t}^* W^* - V_{c_t}^* W^* > \epsilon \), we have \( W^* \left( c_t \circ \left( \mu(\pi^*) - \mu(\hat{\pi}_t) \right) \right) > \epsilon \). In addition, as the agent acts optimally w.r.t. the reward \( r_t = c_t^T W_t \), we have that \( W_t^T (c_t \circ \left( \mu(\pi^*) - \mu(\hat{\pi}_t) \right) \leq 0 \). Combining these observations yield:

\[
(W^* - W^*)^T \cdot (c_t \circ \left( \mu(\pi^*) - \mu(\hat{\pi}_t) \right) > \epsilon > 0 .
\]
This shows that $W^*$ is never disqualified when updating $\Theta_t$. Since $W^* \in \Theta_0$ this implies that $\forall t : W^* \in \Theta_t$. Now we show that not only $W^*$ remains in the ellipsoid, but also a small ball surrounding it. If $\theta$ is disqualified by the algorithm: $(\theta - W_t^*)^T \cdot \left( c_t \odot \left( \mu(\pi_t) - \mu(\bar{\pi}_t) \right) \right) < 0$. Multiplying this inequality by -1 and adding it to (3) yields: $(W^* - \theta)^T \cdot \left( c_t \odot \left( \mu(\pi_t) - \mu(\bar{\pi}_t) \right) \right) > \epsilon$. We apply Hölder inequality to LHS: $\epsilon < \text{LHS} \leq ||W^* - \theta||_\infty \cdot |\left( c_t \odot \left( \mu(\pi_t) - \mu(\bar{\pi}_t) \right) \right) |_1 \leq 2||W^* - \theta||_\infty$.

Therefore for any disqualified $\theta$: $||W^* - \theta||_\infty > \frac{\epsilon}{2}$, thus $B_\infty(W^*, \frac{\epsilon}{2})$ is never disqualified. This implies that $\forall t : \text{vol}(\Theta_t) \geq \text{vol}(\Theta_0 \cap B_\infty(W^*, \frac{\epsilon}{2})) \geq \text{vol}(B_\infty(W^*, \frac{\epsilon}{2}))$. Finally, let $M_T$ be the number of rounds by $T$ in which $V_\pi(W^* - V_{\pi_t}W^*) > \epsilon$. Using Lemma 3 we get that: $\frac{M_T}{2dk+1} \leq \log(\text{vol}(\Theta_0)) - \log(\text{vol}(\Theta_T)) \leq \log(\text{vol}(B_\infty(B_\infty(W^*, \frac{\epsilon}{2})))) \leq dk \log 4\frac{4k}{\epsilon^2}$. Therefore $M_T \leq 2dk(dk+1) \log 4\frac{4k}{\epsilon^2} = O(d^2k^2 \log(\frac{1}{\delta}))$.

**D Proof of Theorem 2**

**Lemma 3 (Azuma’s inequality).** For a martingale $\{S_t\}_{t=0}^n$ if $|S_t - S_{t-1}| \leq b$ a.s. for $i = 1, ..., n$: $P\left( |S_n - S_0| > b \sqrt{2n \log(\frac{2}{\delta})} \right) < \delta$

**Proof of Theorem 2** We first note that we may assume that for any $t$: $||W^* - W_t||_\infty \leq 2$. If $W_t \not\in \Theta_0$, we update the ellipsoid by $\Theta_t \leftarrow \text{MVEE}(\left\{ \theta \in \Theta_t : (\theta - W_t)^T \cdot e_j \leq 0 \right\})$ where $e_j$ is the indicator vector of coordinate $j$ in which $W_t$ exceeds 1, and the inequality direction depends on the sign of $(W_t)_j$. If $W_t \not\in \Theta_0$ still, this process can be repeated for a finite number of steps until $W_T \in \Theta_0$, as the volume of the ellipsoid is bounded from below and each update reduces the volume (Lemma 2). Now we have $W_T \in \Theta_0$, implying $||W^* - W_T||_\infty \leq 2$. As no points of $\Theta_0$ 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 $W_i$ 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 $z_t^{*,H}$ to be the expected value of $\hat{z}_t^{*,H}$, and $z_t^*$ to be the outer product of $c_t(i)$ and the feature expectations of the expert policy for $W^{t(i)}\circ c_t(i) \circ \xi_t(i)$. We also denote $W_t^{*,i}$ by $W_i^*$. We bound the following term from below, as in Theorem 1

$$ (W^* - W)^T \cdot \left( \frac{Z^*}{n} - \bar{Z} \right) = \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (\hat{z}_t^{*,H} - z_t) = $$

$$ \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (z_t^* - z_t) + \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (z_t^{*,H} - z_t^*) + $$

$$ \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (\hat{z}_t^{*,H} - z_t^{*,H}) = $$

$$ \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (z_t^* - z_t) + \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (z_t^{*,H} - z_t) + $$

$$ \frac{1}{n} \sum_{i=1}^{n} (W^* - W)^T \cdot (\hat{z}_t^{*,H} - z_t^{*,H}) $$

(1): is bounded from below by $\epsilon$, identically to the previous proof.

(2): by assumption $||W^* - W_t^*||_\infty \leq \frac{b}{\sqrt{2}}$ thus since $||(z_t^* - z_t)||_1 \leq 2$ by Hölder’s inequality the term is bounded by $\frac{b}{6}$. 

(3)

(4)
F.1 Autonomous driving simulation

In these experiments, we define our mappings in a way that induces different behaviours for different contexts, making generalization a more challenging task. Specifically, for the linear setting we use $W^* = (-1, 0.75, 0.75, 0.5, -1, 0.75)$, before normalization. For our nonlinear mapping, contexts with $|c| > 0.55$ are mapped to reward coefficients vector $(1, -1, -0.05)$, otherwise they are mapped to $(-0.01, 1, -1)$, which induce the feature expectations $(9.75, 3.655, 5), (5.25, 5, 2.343)$ respectively. The decision regions for the nonlinear mapping are visualized in Appendix F.1. The contexts are sampled uniformly in the 2-dimensional simplex. We evaluate all algorithms on the same sequences of contexts, and average the results over 20 such sequences.

**Hyper-parameter selection:** By definition, the ellipsoid algorithm is hyper-parameter free and does not require tuning.
ES algorithms: for the linear model the algorithms maintained a $3 \times 3$ matrix to estimate $W^*$. For loss (1), the algorithm was executed with the parameters: $\sigma = 0.1$, $m = 8$, $\alpha = 0.1$ with decay rate of 0.94, for 50 epochs, where the algorithm takes one step to minimize the loss for each context and the order of the contexts randomized when a new context added but not during the algorithm run. For loss (2), 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 the contexts, but rather used the entire training set for each step. Note, for this loss additional points sampled for the descent direction estimation do not require solving MDPs and thus more can be used for a more accurate calculation. For both losses, the matrix was normalized according to $||\cdot||_2$, and so was the step calculated by the ES algorithm, before it was multiplied by $\alpha$ and applied.

For the nonlinear setting, the model used for the nonlinear mapping was a fully connected neural net, with layers of sizes 15, 10, 5, 3. The activation function used was the leaky ReLU function, with a parameter of 0.1. Note that we can’t normalize the parameters here as in the linear case; therefore an L2-normalization layer is added to the output. The same parameters were used as in the linear case, except with 120 iterations over the entire training set. They were originally optimized for this model and setting and worked as-is for the linear environment. As we aim to estimate the gradient, a small $\sigma$ was used and performed best. The number of points, $m = 250$, was selected as fewer points produced noisy results. The step size, decay rate and the number of iterations were selected in a way that produced fast yet accurate convergence of the loss. Note that here the steps were also normalized before application, and the normalization was applied per layer.

We also provide results for the offline framework, demonstrating the ellipsoid method isn’t suited for this framework and must be initiated in the manner we describe in the warm start section. Here, we used a training and test set of contexts to evaluate the algorithms. The ellipsoid method uses all contexts in the training set to update its estimation of $W^*$, as it would for $\epsilon = 0$. The results show that the descent methods are appropriate for the offline framework, and the ellipsoid is not.

![Figure 6: Visualization of nonlinear decision boundaries](image)

**Figure 6:** Visualization of nonlinear decision boundaries

**F.2 Sepsis treatment**

The environment we describe in 4.2 simulates a decision-making process for treating sepsis. Sepsis is a life-threatening severe infection, where the treatment applied to a sepsis patient is crucial for saving
his life. To create a sepsis treating simulator, we leverage the MIMIC-III data set [Johnson 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.

The processed data from Jeter et al. [2019] 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 data is limited, not all state-action pairs are available. In order to ensure the agent does not attempt to perform such 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 prevalent 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.

**Hyper-parameter selection:** For loss (1), the algorithm was executed with the parameters: $\sigma = 0.1$ with decay rate 0.94, $m = 8$, $\alpha = 0.1$ with decay rate 0.9 for 15 epochs, over one context at a time. An outer decay on the step size was added for faster convergence, the initial $\alpha$ becomes $0.6 + \alpha$ for every 10 additional contexts that are added to the train set, until the training set include 70 contexts. For loss (2), the same method as in the autonomous driving is applied, with the parameters $\sigma = 10^{-4}$, $m = 1000$, $\alpha = 0.25$ with decay rate 0.95, for 80 iterations over the entire training set.