Bayesian Inverse Transition Learning: Learning Dynamics From Near-Optimal Trajectories

Leo Benac
Harvard University

Abhishek Sharma
Harvard University

Sonali Parbhoo
Imperial College London

Finale Doshi-Velez
Harvard University

Abstract

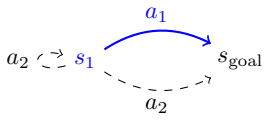
We consider the problem of estimating the transition dynamics T^* from near-optimal expert trajectories in the context of offline model-based reinforcement learning. We develop a novel constraint-based method, Inverse Transition Learning, that treats the limited coverage of the expert trajectories as a *feature*: we use the fact that the expert is near-optimal to inform our estimate of T^*. We integrate our constraints into a Bayesian approach. Across both synthetic environments and real healthcare scenarios like Intensive Care Unit (ICU) patient management in hypotension, we demonstrate not only significant improvements in decision-making, but that our posterior can inform when transfer will be successful.

1 INTRODUCTION

In traditional planning scenarios, the rewards R and transition dynamics T^* of the environment are known, and the goal is to compute the optimal policy π^* that maximizes long-term returns. However, in many real-world situations, the transition dynamics T^* are unknown. Model-Based Reinforcement Learning (MBRL) addresses this by first learning T^* and then performing planning, which improves data efficiency and enables counterfactual reasoning (Sutton and Barto, 2018; Ghavamzadeh et al., 2015; Kidambi et al., 2020; Yu et al., 2021; Lee et al., 2021; Poupart et al., 2006; Ha and Schmidhuber, 2018; Oh et al., 2015; Buesing et al., 2018).

This paper focuses on learning the true dynamics T^* in offline settings using batch data generated by a near-optimal expert. This is a common setting in fields like healthcare and education, where one can presume that the behavior policy is imperfect but generally reasonable. Learning T^* from observational data is challenging due to low coverage of the state-action space. Without the ability to interact with the environment, it is crucial to utilize the limited data effectively. We leverage the knowledge that the trajectories are near-optimal to better estimate T^*. Consider the simple 2-state MDP shown below. The blue path shows the observed optimal behavior leading to the goal state s_{goal} from s_1 after action a_1. The dashed lines represent the alternative hypothetical paths following an unobserved action a_2. The fact that the expert chose action a_1 implies there must be a higher likelihood of reaching s_{goal} via a_1 than a_2 (i.e., $T^*(s_{\text{goal}}|s_1, a_1) > T^*(s_{\text{goal}}|s_1, a_2)$).

A popular definition of near-optimality models an expert's actions as being proportional to the action's values, as presented in Maximum Causal Entropy (MCE) Inverse RL (Ziebart et al., 2010).



Observed expert: a_1
Unobserved: a_2

In clinical settings, clinicians often prefer to administer treatments from drug families that have similar, reasonably good effects, without assigning a ranking. Conversely, they avoid poorly performing treatments, also without assigning a ranking. Instead of selecting a single best action, which may be influenced by data collection or artifacts, it is often better to focus on sets of actions that are nearly equivalent in performance (Tang et al., 2020) and identify which poor behaviors to avoid (Fatemi et al., 2021; Tang et al., 2022).

The most closely related works to ours (Herman et al., 2016; Reddy et al., 2018) rely on the Maximum Causal Entropy (MCE) approach. While modeling near-optimal expert behavior this way is reasonable, our approach shows better performance in clinical settings. Moreover, solving for T^* in the MCE framework requires a gradient-based alternating optimiza-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

tion, which is computationally intensive and prone to local optima. Finally, the constraints in these methods are soft, so their learned dynamics lack guarantees.

To address these limitations, we introduce a novel approach called *Inverse Transition Learning* (ITL), based on hard constraints on the true dynamics T^* and the distinction between two groups of actions. Our constraints ensure that executed actions have higher value than actions not taken. If multiple actions are taken in the same state, they are constrained to be close in value. These thresholds are governed explicitly by a transparent, human-understandable parameter, rather than implicitly through the entropy of an action-value distribution. They can also be solved deterministically via a quadratic program (e.g., using CVXPY (Diamond and Boyd, 2016)), avoiding the drawbacks of gradient-based optimization. This yields an optimization procedure that provides guarantees for the estimated dynamics and converges significantly faster.

The fact that the trajectories are near-optimal provides some information about what transition dynamics T^* are feasible, but limited coverage in the batch setting means that some uncertainty will still remain. Thus, we extend our approach to the Bayesian model-based setting (Dearden et al., 2013). We develop an efficient approach to posterior estimation that includes our constraints. Our approach narrows the gap to the true dynamics T^* compared to baseline Bayesian MBRL methods. Additionally, given a new reward function, we demonstrate how keeping a posterior over the true dynamics T^* can be used to predict when the transfer will be successful. Our main contributions are:

- We propose a fast, transparent, and more reliable method to learn a point estimate of the true unknown dynamics T^* by leveraging expert demonstrations. Our method avoids gradient-based optimization, does not rely on the MCE modeling approach, and enforces constraint guarantees on the learned dynamics.
- We incorporate Bayesian inference to learn a posterior distribution over T^*. The posterior offers the same constraints guarantees for each sample, allows us to quantify the uncertainty over actions, and can be used to predict on which reward functions we except to perform well.

Although our method is designed for tabular MDPs, we demonstrate its effectiveness in continuous domains as well. We tested our approach in a range of environments, including a synthetic Gridworld, 10 different Randomworld environments, and a real-world healthcare setting. These experiments covered various levels of data coverage and expert optimality. Our results show that our method performs well across different metrics, consistently outpacing baseline methods in both synthetic and real-world scenarios.

2 RELATED WORKS

Model-Based Reinforcement Learning. We explore the utilization of forward models, which predict the subsequent state s' from the current state s and action a ($(s, a) \rightarrow s'$), as defined in (Moerland et al., 2023). These models are central for understanding action-induced state transitions. In contrast, backward models identify potential antecedents to states ($s' \rightarrow (s, a)$) and are used for backward planning (Moore and Atkeson, 1993). Similarly, inverse models compute actions required to transition between states ($s, s' \rightarrow a$), beneficial in RRT planning (LaValle, 1998). Additionally, non-parametric methods like replay buffers enable precise estimations (Lin, 1992; Vanseijen and Sutton, 2015; Van Hasselt et al., 2019), and approximation methods, such as Gaussian processes, offer alternatives (Wang et al., 2005; Deisenroth and Rasmussen, 2011). In discrete MDPs like ours, tabular maximum likelihood estimation models, noted as T^{MLE}, represent the state of the art (Sutton, 1991).

Bayesian Model-Based Reinforcement Learning Bayesian MBRL integrates uncertainty into model learning, as detailed in foundational works (Ghavamzadeh et al., 2015; Ross and Pineau, 2008; Dearden et al., 2013). Unlike earlier approaches, such as (Poupart and Vlassis, 2008), which address partially observable settings in discrete factored domains by updating beliefs with new data, our method applies Bayesian MBRL in an offline setting. We infer posterior distributions over transition dynamics solely from expert knowledge. This approach combines prior knowledge with batch data to develop models that yield reliable policies (Guo et al., 2022). In contrast, studies like (Zhang et al., 2020a,b) focus on learning invariant representations for control without explicitly modeling transition dynamics.

Learning from Demonstrations. *Imitation Learning* (IL) learns policies directly from expert demonstrations via per-timestep training (Stéphane et al., 2010), iterative expert feedback (DAgger) (Ross et al., 2011), or imposing linear constraints during policy optimization (APID) (Kim et al., 2013). In contrast, *Inverse Reinforcement Learning* (IRL) infers the reward R from dynamics T and demonstrations to facilitate transferability (Ng et al., 2000), using methods like Maximum Margin and Maximum Entropy to robustly mimic experts (Abbeel and Ng, 2004; Ziebart et al., 2008; Scobee and Sastry, 2019). To address R's non-identifiability, Bayesian IRL learns a distribution

over R (Ramachandran and Amir, 2007). Distinct from these policy and reward-focused approaches, our *Inverse Transition Learning* framework utilizes expert demonstrations to directly refine the estimation of the true dynamics T^*, contrasting with standard baselines that rely solely on maximum likelihood.

3 PRELIMINARIES

Markov Decision Processes (MDPs). An MDP $\mathcal{M}$ can be represented as a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T^*, \gamma, R\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, T^* is the true dynamics of the environment, γ is the discount factor, and R is a bounded reward function. In planning, the goal is to find the best policy π^* corresponding to an MDP $\mathcal{M}$. In this paper, we focus on tabular MDPs (discrete state and action spaces). Discretization of continuous clinical variables can be achieved using domain knowledge and is a common and effective practice in healthcare RL applications. This approach aligns naturally with clinical decision-making, where physicians often categorize patient states and treatment options into discrete, clinically meaningful groups. For example, (Choudhary et al., 2024) use the same healthcare dataset as us (MIMIC (Johnson et al., 2020)) to create an RL simulator by discretizing both state and action spaces. Such discretization not only reflects clinical practice but also makes the learned dynamics more interpretable in low-data settings.

Value Functions. For a policy π, the state–value function $V^\pi(s)$ is the expected discounted return starting from state s, and the action–value function $Q^\pi(s,a)$ is the expected discounted return starting from state s and taking action a.

Notation. Let V^π denote the vector of values $V^\pi(s)$. We use shorthand R_a , Q_a, and T_a to represent the vectors $R(\cdot, a)$, $Q(\cdot, a)$ and the matrix $T(\cdot \mid \cdot, a)$. We also use shorthand R_π, Q_π, and T_π to represent the vectors $\mathbb{E}_{a\sim\pi}[R_a]$. $\mathbb{E}_{a\sim\pi}[Q_a]$ and the matrix $\mathbb{E}_{a\sim\pi}[T_a]$. For dynamics T, we let $\pi^*(T)$ and $Q^*(T)$ denote the corresponding optimal policy and Q values function respectively with respect to dynamics T. In tabular settings, the value functions V^π and Q^π_a can be calculated directly through a closed-form solution:

$$V^\pi = R_\pi + \gamma T_\pi V^\pi = (I - \gamma T_\pi)^{-1} R_\pi \tag{1}$$

$$Q^\pi_a = R_a + \gamma T_a (I - \gamma T_\pi)^{-1} R_\pi. \tag{2}$$

Methods for Estimating the Transition Dynamics T^*. In tabular settings, a common method to estimate the true transition dynamics T^* is the Maximum Likelihood Estimate (MLE), denoted as T^{MLE} (e.g., (Barto et al., 1995; Kim and Oh, 2023; Ornik and Topcu, 2021)). The number of times the tuple (s, a, s') is observed is denoted by $N_{s,a,s'}$. We apply Laplace smoothing, represented by δ, to address issues such as zero occurrences in the count data due to low coverage in our batch data $\mathcal{D}$. The T^{MLE} is defined as:

$$T^{MLE}(s'|s,a) = \frac{N_{s,a,s'} + \delta}{\sum_{s'}(N_{s,a,s'} + \delta)} \tag{3}$$

In offline tabular settings, (Herman et al., 2016) leverage near-optimal batch data to infer dynamics by reusing the MCE IRL framework (Ziebart et al., 2010). We include this method in our baseline and refer to it as MCE (or T^{MCE}) moving forward. Their method consist of an interative procedure between taking one gradient step of the dynamics parameters θ with respect to the loss in equation 4 and then performing soft-Value Iteration, where Q is the soft Q function.

$$\sum_{(s,a,s')\in\mathcal{D}} \left[\log \frac{\exp(Q_\theta(s,a))}{\sum_{a'} \exp(Q_\theta(s,a'))} + \log T_\theta(s' \mid s,a) \right]. \tag{4}$$

To model the uncertainty of transitions T^* probabilistically, (Ghavamzadeh et al., 2015) uses a Multinomial likelihood in conjunction with a Dirichlet prior. This combination yields the posterior $P(T^*|\mathcal{D})$, which estimates the transition probabilities given data $\mathcal{D}$.

$$\begin{aligned} P(T^*(\cdot \mid s,a) \mid \mathcal{D}) &= \mathrm{Dir}(\mathbf{N}_{s,a} + \delta \mid s,a) \\ &\propto \mathrm{Multinomial}(\mathbf{N}_{s,a} \mid s,a) \cdot \mathrm{Dir}(\delta \mid s,a) \end{aligned} \tag{5}$$

Note that T^{MLE} represents the mean of the distribution $P(T^*|\mathcal{D})$. However, this posterior distribution is not as tight as it could be because it does not account for the near-optimality of expert trajectories. To refine this, we utilize expert signals to develop a tighter posterior $P(T^*|\mathcal{D})$, enhancing the estimation of the transition dynamics. To the best of our knowledge, we are the first to infer a distribution over the dynamics using expert demonstrations. Furthermore, we propose a method distinct from (Herman et al., 2016) to derive a point estimate of T^*. This involves formulating constraints based on expert behaviors, offering a more accurate approach to modeling the true dynamics.

4 ESTIMATING TRANSITION DYNAMICS WITH ϵ-OPTIMAL EXPERT

This section introduces key definitions to relate the expert's optimality to the true dynamics T^*.

Definition 1 (ϵ-*ball*)**.** *For any state s and transition dynamics T, an action a is in the ϵ-ball $\epsilon(s;T)$ if it is ϵ-close to the optimal action according to its optimal Q-function $Q^*(.,.;T)$ with respect to dynamics T:*

$$a \in \epsilon(s;T) \iff \max_{a'} Q^*(s,a';T) - Q^*(s,a;T) \leq \epsilon$$

For each state s, $\epsilon(s; T^*)$ contains the actions that are near-optimal with respect to the true dynamics T^*.

Definition 2 (ϵ-*optimality*)**.** *A policy $\pi_\epsilon(.|s; T^*)$ is ϵ-optimal with respect to the true unknown transition dynamics T^* if it exclusively selects actions from the ϵ-ball $\epsilon(s; T^*)$ for all states s:*

$$\pi_\epsilon(a|s; T^*) > 0 \iff a \in \epsilon(s; T^*), \quad \forall s \in \mathcal{S}$$

Such a policy $\pi_\epsilon(.|s; T^*)$ can make a mistake of at most ϵ at each time step relative to $Q^*(., .; T^*)$. We assume our demonstrations are generated by such policy.

Definition 3 (ϵ-*ball property*)**.** *Dynamics T satisfy the ϵ-ball property for state s if $\epsilon(s; T) = \epsilon(s; T^*)$ and* ***every*** *action $a' \notin \epsilon(s; T)$ is at least ϵ-away from all actions $a \in \epsilon(s; T)$ with respect to $Q^*(s, .; T)$:*

$$\begin{aligned} &\epsilon(s; T) = \epsilon(s; T^*) \quad \textit{and} \\ &Q^*(s, a; T) - Q^*(s, a'; T) \geq \epsilon, \\ &\qquad \forall a \in \epsilon(s; T),\, \forall a' \notin \epsilon(s; T) \end{aligned} \tag{6}$$

We assume the true dynamics T^* satisfy the ϵ-ball property for each state s, splitting actions into two groups: near-optimal and suboptimal. Actions inside $\epsilon(s; T^*)$ are **valid**, and those outside $\epsilon(s; T^*)$ **invalid**. Our goal is to learn dynamics T that satisfy this property for every state s. Such dynamics can distinguish near-optimal actions from suboptimal ones.

Definition 4 (*Deterministic/stochastic-policy state*)**.** *A deterministic-policy state occurs when the ϵ-optimal expert $\pi_\epsilon(.|s; T^*)$ selects a single action in state s (meaning the expert is deterministically optimal). A stochastic-policy state occurs when the expert selects multiple actions in such state:*

$$\textit{Deterministic-policy state:} \quad |\epsilon(s; T^*)| = 1$$

$$\textit{Stochastic-policy state:} \quad |\epsilon(s; T^*)| > 1$$

Deterministic-policy states are akin to conditions well-understood by clinicians who are certain of the best treatment, whereas stochastic-policy states resemble conditions where multiple reasonable treatments exist without clear superiority.

Problem Setting We assume we are given the MDP $\mathcal{M}$ except for the true transition dynamics T^* (i.e., $\mathcal{M} \setminus \{T^*\}$). We also have access to batch data $\mathcal{D} = \{(s_i, a_i, s_i')\}_{i=1}^N$, where the data is assumed to have been generated by the behavior policy $\pi_\epsilon(.|s; T^*)$, which is ϵ-optimal with respect to the true dynamics T^*. Note that when $\epsilon = 0$, the expert $\pi_\epsilon(.|., T^*)$ is fully optimal, and hence we would only encounter deterministic-policy states.

Why Access to the Reward Function is Reasonable Unlike transition dynamics, which describe the complex physiological response of the human body to treatments and are difficult to infer even with domain knowledge, the reward function in healthcare applications is often more accessible. In the case of hypotension management, clinical expertise can reasonably define rewards that distinguish good and bad outcomes, such as stabilizing blood pressure or preventing adverse events. Therefore, in our setup, it is natural to assume access to the reward function while focusing on learning the transition dynamics T^*. However, we recognize that in other domains, such as robotics, specifying a reward function can be challenging. For such settings, we provide an extension of our algorithm in Appendix B that jointly learns the transition dynamics and the reward function simultaneously. Our results demonstrate that this joint learning approach achieves performance comparable to when the true reward function is provided.

4.1 Constraints on Transition Dynamics Given $\pi_\epsilon(.|.; T^*)$

In this section, we develop a set of constraints for learning transition dynamics T motivated by the ϵ-optimal assumption we have on the expert (See Definition 2). We denote these constraints on the dynamics T collectively as **Constraints(T)**, which consist of two types:

Constraint 1(T): Separation Between Valid and Invalid Actions For every valid action $a \in \epsilon(s; T^*)$ and every invalid action $a' \notin \epsilon(s; T^*)$, T must satisfy:

$$Q^*(s, a; T) - Q^*(s, a'; T) \geq \epsilon \tag{7}$$

Expanding the Q-values in terms of T:

$$\begin{aligned} &R(s, a) - R(s, a') + \gamma\big(T(\cdot \mid s, a) \\ &-T(\cdot \mid s, a')\big)^\top (I - \gamma T_{\pi^*(T)})^{-1} R_{\pi^*(T)} \geq \epsilon \end{aligned} \tag{8}$$

where $\pi^*(T)$ denotes the optimal policy induced by dynamics T. This constraint ensures that for every state s, *all* invalid actions are at least ϵ-away from *all* valid actions with respect to $Q^*(s, \cdot; T)$.

Constraint 2(T): ϵ-Closeness Among Valid Actions For each pair of valid actions $(a, a') \in \epsilon(s; T^*) \times \epsilon(s; T^*)$ where $a \neq a'$, T must satisfy:

$$|Q^*(s, a; T) - Q^*(s, a'; T)| \leq \epsilon \tag{9}$$

Expanding the Q-values in terms of T:

$$\begin{aligned} \Big|&R(s, a) - R(s, a') + \gamma\big(T(\cdot \mid s, a) - T(\cdot \mid s, a')\big)^\top \\ &(I - \gamma T_{\pi^*(T)})^{-1} R_{\pi^*(T)}\Big| \leq \epsilon \end{aligned} \tag{10}$$

This constraint ensures that for each state s, valid actions remain ϵ-close with respect to $Q^*(s, \cdot; T)$.

Enforcing the ϵ-ball property: Together, these constraints enforce the ϵ-ball property (Definition 3) for the learned dynamics T. **Constraint 2(T)** ensures that all valid actions remain ϵ-close to each other, while **Constraint 1(T)** ensures that all invalid actions stay at least ϵ-away from valid actions with respect to dynamics T. This is exactly what the ϵ-ball property requires. Hence, if dynamics T satisfy both sets of constraints, then it satisfies the ϵ-ball property.

4.2 Constraints on Transition Dynamics in practice

While the true dynamics T^* and sets $\epsilon(s; T^*)$ are unknown, we approximate them by collecting all actions a taken in state s from the dataset $\mathcal{D}$, denoted $\hat{\epsilon}(s; T^*)$. This relies on the assumption that $\mathcal{D}$ was generated by an ϵ-optimal expert under the true dynamics T^*.

In practice, we do not have direct access to the expert policy $\pi_\epsilon(\cdot \mid s; T^*)$. Instead, we rely on the dataset $\mathcal{D}$, which may not cover all (state, action) pairs. We therefore define an estimated policy $\hat{\pi}_\epsilon(\cdot \mid s; T^*)$: for states $s \in \mathcal{D}$ it assigns equal probability to all observed actions, while for $s \notin \mathcal{D}$ it selects the best action under the proposed transition model T (using $\pi^*(T)$). This policy is then used to define the constraints in Algorithm 1 & 2, which integrates these components to estimate the true but unknown dynamics T^*.

5 METHODOLOGY

In this section, we describe how, given (near-)optimal data, our constraints can be applied to infer both a posterior distribution over the true dynamics T^* and a point estimate. These are referred to as *Bayesian Inverse Transition Learning* (BITL) and *Inverse Transition Learning* (ITL), respectively.

Bayesian Inverse Transition Learning Our constraints (Inequalities 7 and 9) lead to an underdetermined problem with infinitely many solutions. Instead of introducing a loss function as the baseline MCE (Herman et al., 2016), we infer a posterior distribution on the true transition dynamics T^* to quantify such uncertainty. We introduce a sampling-based technique to infer this distribution, denoted as $P_\epsilon(T^*|\mathcal{D})$, assuming data is generated by an ϵ-optimal expert. We use our constraints to ensure each sample satisfies the ϵ-ball property (See Definition 3) for each state.

Naive Approach: Rejection Sampling The complexity of deriving a posterior from expert trajectories with many constraints precludes an analytic solution. An initial attempt might involve drawing samples from the simpler $P(T^*|\mathcal{D})$ and rejecting those that fail to meet our constraints. This turns out to be very inefficient and rejects nearly all samples.

HMC with Reflection To enhance sampling efficiency within our constrained, high-dimensional space, we employ Hamiltonian Monte Carlo (HMC) with reflection (Betancourt, 2011). The log-likelihood function of our target distribution takes the following form: $\log P(T^* \mid \mathcal{D}) = \sum_{s,a} \log \mathrm{Dir}(\mathbf{N}_{s,a} + \delta \mid s, a)$. Applying standard HMC to Dirichlet distributions is challenging due to their simplex constraints. To address this, we use the invertible transformations from (Betancourt, 2012), combined with a logit transformation, to map the constrained transition dynamics T into an unconstrained position variable w.

Our sampling procedure (Algorithm 1) utilizes first-order leapfrog integration. Let m denote the momentum, α the step size, $\nabla E(w)$ the energy gradient, and L the number of integration steps. During each spatial step, we transform w back to $T^{(i)}$ to compute the optimal policy $\pi^*(T^{(i)})$ and check **Constraints 1 & 2** (Inequalities 7 & 9). If a constraint $c_{s,a,a'}(T^{(i)})$ is violated, the trajectory "bounces" by reflecting the momentum m off the constraint's normal vector $\hat{n}$.

Algorithm 1 HMC with reflection for Bayesian ITL

$m \leftarrow m - \frac{1}{2}\alpha\nabla E(w)$ *// First momentum half step*
for $i = 0$ to L **do**
  $w \leftarrow w + \alpha m$ *// Full spatial step*
  $T^{(i)} \leftarrow \mathrm{w\_to\_T}(w)$
  Compute $\pi^*(T^{(i)})$
  if all **constraints**$(T^{(i)})$ are satisfied **then**
    $m \leftarrow m - \alpha\nabla E(w)$ *// Full momentum step*
  else
    $\hat{n} \leftarrow \nabla c_{s,a,a'}(T^{(i)}) / \|\nabla c_{s,a,a'}(T^{(i)})\|$
    $m \leftarrow m - 2(m \cdot \hat{n})\hat{n}$
  end if
end for
$w \leftarrow w + \alpha m$
$m \leftarrow m - \frac{1}{2}\alpha\nabla E(w)$ *// Final momentum half step*

A final clean-up step ensures that if the leapfrog integration terminates outside the feasible region, the sample is immediately rejected. This guarantees that every accepted sample satisfies the ϵ-ball property for each state $s \in \mathcal{D}$. To ensure initial feasibility, we warm-start the HMC chain with the point estimate $\widehat{T}^*$ obtained via ITL in Algorithm 2 (See below). By reflecting off boundaries rather than blindly rejecting, our algorithm achieves a rejection rate of about 20% to 60% across our experiments—a significant improve-

ment over the nearly 100% rejection rate of standard rejection sampling.

Algorithm 2 Point estimate $\widehat{T}^*$ (ITL)

$i \leftarrow 0$
$\pi^{(0)}(\cdot|s) \leftarrow \widehat{\pi}_\epsilon(\cdot|s; T^*)$ for $s \in \mathcal{D}$
$\pi^{(0)}(\cdot|s) \leftarrow$ uniform distribution for $s \notin \mathcal{D}$
$T^{(0)} \leftarrow T^{\text{MLE}}$
$T^{(1)} \leftarrow solve_{ITL}(\pi^{(0)}, T^{(0)})$
$i \leftarrow i + 1$
while $T^{(i)}$ does not satisfy ϵ-ball property for each $s \in \mathcal{D}$ **do**
  $\pi^{(i)}(\cdot|s) \leftarrow \widehat{\pi}_\epsilon(\cdot|s; T^*)$ for $s \in \mathcal{D}$
  $\pi^{(i)}(\cdot|s) \leftarrow \pi^*(T^{(i)})$ for $s \notin \mathcal{D}$
  $T^{(i+1)} \leftarrow solve_{ITL}(\{\pi^{(k)}, T^{(k)}\}_{k=0}^{i})$
  $i \leftarrow i + 1$
end while

Inverse Transition Learning In addition to our posterior estimate, we can also infer a point estimate $\widehat{T}^*$. By introducing a quadratic loss that enforces such an estimate to remain close to T^{MLE}, and by linearizing **Constraints 1 & 2** using a fixed reference policy $\tilde{\pi}$ and fixed transition dynamics $\tilde{T}$, we form a quadratic convex optimization problem that can be solved efficiently using CVXPY (Diamond and Boyd, 2016):

$$\min_{T} \sum_{(s,a,s')} N_{s,a,s'} \cdot \left[T(s' \mid s,a) - T^{\text{MLE}}(s' \mid s,a)\right]^2$$
$$\text{subject to} \quad \forall (s,a) \in \mathcal{D}, \forall a' \notin \hat{\epsilon}(s; T^*):$$
$$R(s,a) - R(s,a') + \gamma \left(T(\cdot \mid s,a) - T(\cdot \mid s,a')\right)^\top \times \left(I - \gamma \tilde{T}_{\tilde{\pi}}\right)^{-1} R_{\tilde{\pi}} \geq \epsilon,$$
$$\text{and subject to} \quad \forall (s,a) \in \mathcal{D}, \forall a' \in \hat{\epsilon}(s; T^*):$$
$$\left| R(s,a) - R(s,a') + \gamma \left(T(\cdot \mid s,a) - T(\cdot \mid s,a')\right)^\top \times \left(I - \gamma \tilde{T}_{\tilde{\pi}}\right)^{-1} R_{\tilde{\pi}} \right| \leq \epsilon \tag{11}$$

In general, we denote the result of this optimization framework as $solve_{ITL}(\tilde{\pi}, \tilde{T})$. Because the variables $\tilde{\pi}$ and $\tilde{T}$ are held constant, the problem remains strictly convex with respect to the optimization variable T. For the first iteration, we initialize these fixed inputs as $\tilde{\pi} = \widehat{\pi}_\epsilon$ and $\tilde{T} = T^{\text{MLE}}$ because they provide the most reliable empirical estimates for the state-action pairs (s,a) where we have data (i.e., $(s,a) \in \mathcal{D}$). This convex formulation avoids the alternative local optima associated with gradient-based methods like MCE (Herman et al., 2016) and significantly accelerates training.

This yields the iterative procedure described in Algorithm 2. We initialize our constraints using $\widehat{\pi}_\epsilon$ and T^{MLE}. In subsequent steps, $solve_{ITL}(\{\pi^{(k)}, T^{(k)}\}_{k=0}^{i})$ denotes solving the optimization problem with additional constraints accumulated from all iterations up to i. We keep augmenting the constraint set until we reach a point estimate $\widehat{T}^*$ that satisfies the ϵ-ball property for each state $s \in \mathcal{D}$ (Definition 3). In essence, $\widehat{T}^*$ represents the dynamics that best fit the batch data $\mathcal{D}$ while also explaining the ϵ-optimal expert behavior.

6 EXPERIMENTAL SETUP

Environments Our evaluations span 11 synthetic environments and a real-life ICU setting. Specifically, we use a 25 states Gridworld with four actions, 10 different 15-states Randomworld with five actions each, and a healthcare scenario focusing on ICU patients with hypotension, utilizing the MIMIC-IV dataset (Johnson et al., 2020). In Gridworld, we generate 100 batches of data $\mathcal{D}$, each consisting of five episodes with 15 steps each. For each 10 of the Randomworld, 50 batches are generated, each containing three episodes of ten steps, due to their smaller size. We also evaluate our methods across varying "Coverage %" levels in these environments, defined as the percentage of states observed within each batch data $\mathcal{D}$ and various ϵ values to see how our method compare to baselines for various degrees of sub-optimality. Detailed descriptions of both the synthetic and real-world environments are available in Appendix C and D respectively.

In synthetic scenarios, we simulate a suboptimal expert, where approximately 40% of states are stochastic-policy states (See definition 4), demonstrating the robustness of our approach to suboptimal expert behavior. We also include in the Appendix (Table 7, 8, Figure 8 and 9), results for 20% and 0% (fully optimal expert) of stochastic-policy states which correspond to lower ϵ values. Note that increased optimality from the expert (lower ϵ values) leads to less stochasticity (ie. less stochastic-policy states) amongst the actions selected in the batch data $\mathcal{D}$. Our method does not require the expert to be perfectly optimal and can adapt to various degrees of optimality through ϵ. In the results below, we use $\gamma = 0.95$ and smoothing parameter defined in Equation 3 $\delta = 0.001$.

Baselines. Our methods, Bayesian Inverse Transition Learning (BITL) and Inverse Transition Learning (ITL), are evaluated against T^{MLE} (Maximum Likelihood Estimation, MLE), T^{MCE} (Maximum Causal Entropy, MCE) (Herman et al., 2016), and a non-expert-informed posterior $P(T^*|\mathcal{D})$ (PS).

Metrics. We use the following metrics to evaluate learnt dynamics $T's$ and induced policies $\pi's$:

Best/ϵ-ball action matching: Proportion of states

where the best or ϵ-good action is taken:

$$\text{Best: } \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\mathcal{I}\Big\{\arg\max_a \pi(a|s) = \arg\max_a \pi^*(a|s)\Big\},$$

$$\epsilon\text{-ball: } \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\mathcal{I}\Big\{\arg\max_a \pi(a|s) \in \epsilon(s;T^*)\Big\}.$$

Bayesian regret: Regret of a sample-based empirical posterior distribution $\widehat{P}(T|\mathcal{D})$, where $|\widehat{P}(T|\mathcal{D})|$ is number of T's in the sample-based distribution:

$$\frac{1}{|\widehat{P}(T|\mathcal{D})|^2}\sum_{T\in\widehat{P}(T|\mathcal{D})}\sum_{T'\in\widehat{P}(T|\mathcal{D})} \times \mathbb{E}_{s_0\sim\mu_0}\Big[\Big|V^{\pi^*(T)}(s_0;T) - V^{\pi^*(T')}(s_0;T)\Big|\Big]$$

We also report the (normalized) **Value**, calculated as the value achieved by the optimal policy of the learned dynamics under the true dynamics T^*, i.e., $\mathbb{E}_{s_0}[V^{\pi^*(T)}(s_0;T^*)]$. In addition, we report the **CVaR** of Value across datasets, **Total Variation**, the **number of violated constraints (Inequalities 7, 9)**, and training **Time** (in seconds) for both the baseline MCE and our ITL method. Finally, **Value** and **Best/ϵ-ball action matching** are evaluated in both standard and transfer settings.

Transfer Task Setting In this paper, a transfer task refers to the evaluation of the learned dynamics and/or policy under a reward function different from the one used during training. This setting is particularly relevant in healthcare, where the transition dynamics, which describe the physiological response of the human body to treatments, remain largely consistent. However, the reward function, which captures the clinical objectives being optimized, can vary across hospitals, patient populations, and individual clinicians. By evaluating our method under different reward functions, we assess its robustness to variations in clinical goals, ensuring that the learned dynamics remain useful even when the optimization criteria shift. This aligns with real-world applications, where treatment strategies may change while the underlying patient physiology remains unchanged. To demonstrate that our learned dynamics can adapt to actions not seen in the demonstrations, we set up a transfer task where the optimal actions under this new reward function differ from those in the original task (See Figure 4, 5). This ensures the new task requires actions we didn't observe in the initial demonstrations. This tests the model's adaptability to changes in task conditions.

Computing the results. For each dynamics T, we compute its optimal policy $\pi^*(T)$ using Value Iteration (Sutton and Barto, 2018), when needed to compute metrics in terms of policies. For posterior distributions $P(T|\mathcal{D})$ and $P_\epsilon(T|\mathcal{D})$, we determine the optimal policies of 5,000 sampled $T's$ and average them.

How to Tune ϵ: In the healthcare setting, we only have access to an offline dataset. To select an appropriate ϵ, we evaluate the performance of ITL on a held-out validation set. We choose the ϵ that performs best on this validation set based on the **Best/ϵ-ball action matching metric** for both standard and transfer tasks (defined by different reward functions). As shown in Figure 6 (See Appendix), $\epsilon = 5$ appears to be suitable, and this value is used for training and computing the results in the real-life healthcare experiments (See Table 2). We also present results for $\epsilon = 10$ and $\epsilon = 15$, as these values are also reasonable. Results for $\epsilon = 10$ and $\epsilon = 15$ are provided in Tables 9 and 10 in the Appendix, showing similar performance.

7 RESULTS

ITL consistently outperforms baseline methods across all metrics and coverage settings. Table 1 (synthetic environments), Table 2 (ICU dataset), and Figure 1 show that ITL and BITL outperform baselines across metrics and coverage levels in both synthetic and real healthcare settings. Additional results for other ϵ values are provided in the appendix (Tables 7, 8, Figures 8, 9). In Randomworlds, BITL and ITL clearly surpass MCE, which itself outperforms MLE and PS. In Gridworld, ITL slightly outperforms MCE on average but achieves substantially better worst-case performance, underscoring the value of enforcing hard constraints and the risk of MCE getting stuck in poor local optima due to gradient-based optimization. In the ICU hypotension management task, ITL and BITL significantly outperform all baselines (Table 2), demonstrating the robust applicability of our approach to high-stakes, data-limited clinical settings. ITL also trains substantially faster. In the standard task, ITL and BITL are the only methods that converge to optimal performance as coverage increases (Figure 1, left column), achieving higher values without constraint violations—thanks to our hard-constraint design versus MCE's soft constraints.

ITL enables reliable, fast, and data-efficient dynamics estimation. In Gridworld, MCE required far more training time than ITL (137s vs. 1s) and produced poorer CVaR values, reflecting the difficulties of gradient optimization in non-convex settings. ITL's efficiency also holds in Randomworld, with training times of 0.7s versus 35s for MCE (Table 6). Our methods consistently converge to optimal performance as coverage increases, aligning with expert demonstrations. MCE's tendency to get stuck in local optima

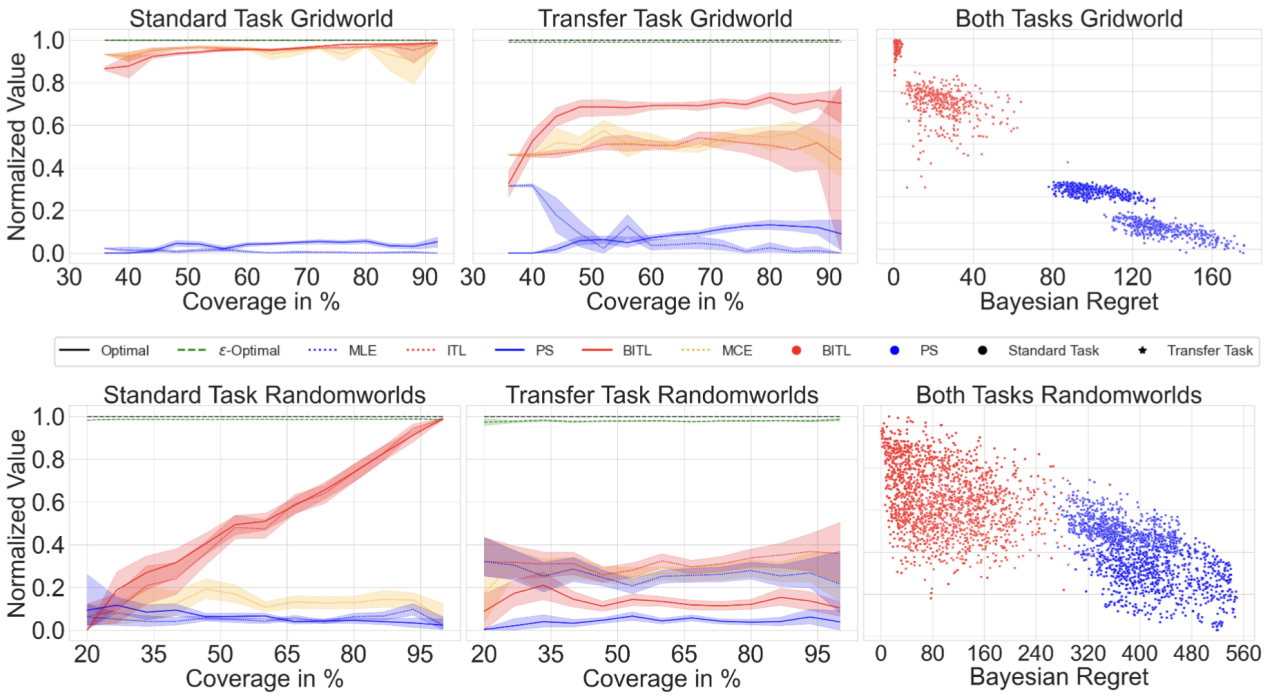


Figure 1: Top row: Normalized Value vs. Coverage for Gridworld (left: Standard Task, middle: Transfer Task), Bottom row: Normalized Value vs. Coverage for Randomworlds (left: Standard Task, middle: Transfer Task). Rightmost plots: Normalized Value vs. Bayesian Regret of both Tasks (top: Gridworld, bottom: Randomworlds). Overall ITL and BITL outerperforms other baselines and BITL help predicting when we can transfer well.

Gridworld					
Method	**MCE**	**BITL**	**ITL**	**MLE**	**PS**
ϵ matching	0.65 ± 0.12	$\mathbf{0.78 \pm 0.08}$	0.65 ± 0.12	0.37 ± 0.06	0.38 ± 0.02
ϵ matching transfer	0.5 ± 0.11	$\mathbf{0.64 \pm 0.08}$	0.51 ± 0.11	0.39 ± 0.07	0.39 ± 0.02
Best matching	0.55 ± 0.11	$\mathbf{0.64 \pm 0.08}$	0.56 ± 0.11	0.31 ± 0.06	0.29 ± 0.02
Best matching transfer	0.45 ± 0.1	$\mathbf{0.54 \pm 0.08}$	0.45 ± 0.11	0.34 ± 0.07	0.31 ± 0.02
Time	117.47 ± 26.34	14384.26 ± 489.49	0.32 ± 0.26	$\mathbf{0.01 \pm 0.00}$	8.65 ± 0.24
Total Variation	137.62 ± 4.39	159.36 ± 3.98	$\mathbf{137.05 \pm 4.32}$	141.37 ± 3.8	160.05 ± 4.43
Value CVaR 5%	108.76 ± 26.54	109.04 ± 4.26	$\mathbf{109.7 \pm 9.01}$	-4.71 ± 0.47	-2.35 ± 1.35
Nbr constraints violated	3.06 ± 3.68	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	23.13 ± 6.59	16.37 ± 3.21
Randomworlds					
Method	**MCE**	**BITL**	**ITL**	**MLE**	**PS**
ϵ matching	0.54 ± 0.12	0.75 ± 0.12	$\mathbf{0.76 \pm 0.13}$	0.43 ± 0.11	0.3 ± 0.05
ϵ matching transfer	0.46 ± 0.12	0.38 ± 0.1	$\mathbf{0.5 \pm 0.13}$	0.46 ± 0.12	0.31 ± 0.05
Best matching	0.38 ± 0.13	0.57 ± 0.12	$\mathbf{0.58 \pm 0.15}$	0.29 ± 0.11	0.19 ± 0.05
Best matching transfer	0.34 ± 0.13	0.26 ± 0.09	$\mathbf{0.37 \pm 0.15}$	0.34 ± 0.13	0.2 ± 0.05
Time	36.61 ± 26.36	5561 ± 223.43	0.65 ± 0.39	$\mathbf{0.0 \pm 0.0}$	2.61 ± 0.36
Total Variation	111.08 ± 2.42	123.3 ± 4.06	$\mathbf{102.02 \pm 4.84}$	111.07 ± 2.3	127.02 ± 2.66
Value CVaR 5%	-459.71 ± 23.48	-364.34 ± 34.06	$\mathbf{-366.43 \pm 16.03}$	-481.98 ± 23.31	-434.24 ± 17.22
Nbr constraints violated	11.81 ± 6.47	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	17.23 ± 6.75	11.66 ± 3.12

Table 1: Gridworld and Randomworld results. BITL performs best in Gridworld, ITL in Randomworld.

further underscores the efficiency and reliability of ITL and BITL in complex environments.

Our policies stay within the support of expert actions in *every* visited state, with hard constraints yielding better worst-case outcomes. By enforcing hard constraints, our method ensures policies adhere to expert actions in every visited state, recovering an ϵ-optimal action and minimizing violations. In

contrast, MLE, PS, and MCE occasionally breach constraints (Tables 6, 2), while ITL and BITL maintain strict compliance, enhancing reliability. CVaR results (Table 6) show ITL and BITL achieve stronger worst-case performance by enforcing the ϵ-ball property on estimated dynamics (Definition 3), preserving robustness even in challenging real-world scenarios such as ICU hypotension management.

Our method integrates efficiently with Bayesian inference, enabling exploration and predicting transfer success. By inferring calibrated uncertainties, BITL leverages expert knowledge to enhance performance in environments like Gridworld, where it avoids repeated suboptimal decisions (e.g., wall collisions) and achieves the best overall results. This shows that modeling uncertainty over dynamics while incorporating demonstrations provides a clear advantage over point-estimate methods. The issue is less pronounced in Randomworld and the ICU, highlighting the adaptability of our approach across different dynamics. The Bayesian framework also enables more accurate predictions of which tasks will yield successful outcomes based on the learned distributions. Regret plots (Figure 1, right column) show a strong correlation between Bayesian regret values and task performance, with higher regret in transfer tasks than in standard tasks, as expected. Table 2 further illustrates this in the ICU dataset, where tasks with lower regret align with better performance.

Our method applies to continuous healthcare settings and provides insights through counterfactual reasoning. In the ICU environment, we discretize states using Oxygen ratio (O2), Blood pressure (BP), Creatinine (Crea), and Glasgow Coma Scale (GCS) (Table 4 in Appendix). After prescribing IV treatment in state (O2=1, BP=1, GCS=1, Crea=2), we examine the three most likely next states under MLE, ITL, and MCE. Since clinicians never prescribed IV here, MLE gives uniform predictions and is omitted. ITL spreads probability across states that improve or maintain BP and creatinine, while MCE collapses all probability on a single state—unrealistic given physiological complexity and missing data. This illustrates how MCE can yield implausible results in real-world settings. ITL's next-state probabilities in Figure 2 appear near-uniform due to the prior and absence of data, yet ITL still ranks states and quantifies uncertainty more informatively than MLE (exactly uniform) or MCE (which reduces uncertainty to zero).

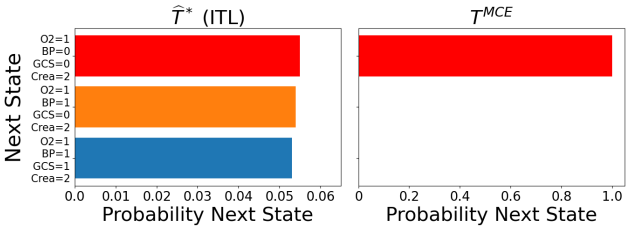


Figure 2: Most likely next three states after IV in state O_2=1, BP=1, GCS=1, Crea=2. MLE omitted (uniform). ITL spreads mass over improving/maintaining states; MCE collapses onto one state.

Method	MLE	ITL	MCE	BITL	PS
Standard Task					
Best match.	0.33	**0.51**	0.31	0.49	0.31
ϵ match.	0.52	**1.0**	0.56	**1.0**	0.51
Constraints	61	**0**	47	**0**	58
Time (s)	-	**2.23**	118	-	-
Bayes Regret	-	-	-	**2.49**	10
Transfer Task					
Best match.	0.34	**0.52**	0.32	0.47	0.34
ϵ match.	0.58	**0.97**	0.68	0.90	0.58
Bayes Regret	-	-	-	**2.80**	5.40

Table 2: ICU results ($\epsilon = 5$). ITL beats all point-estimates and BITL surpasses all Bayesian-estimates.

8 DISCUSSION

Summary. We addressed the challenge of estimating transition dynamics T^* from near-optimal expert trajectories in offline model-based RL. Our approach, Inverse Transition Learning (ITL), leverages limited expert coverage and near-optimality to estimate T^*. We introduced a novel constraint-based method and integrated it within a Bayesian framework to learn a posterior over the dynamics. To our knowledge, this is the first method to combine posterior estimation of dynamics with expert demonstrations. ITL improves decision-making in both synthetic environments and real-world healthcare, such as ICU hypotension management. It not only enhances decision quality but also provides insights into task transfer.

Future Work. Although ITL provides a robust approach for estimating dynamics from demonstrations, it is currently limited to discrete, fully observable state spaces. Future work could extend ITL to high-dimensional, continuous, or partially observable settings, along with deeper theoretical analysis.

Acknowledgments

We thank the fantastic AISTATS reviewers for their constructive comments and suggestions. This material is based upon work supported by the National Science Foundation under Grant No. IIS-2007076 as well as the Cooperative AI Foundation. Any opinions, findings, and conclusions or recommendations expressed in this

material are those of the authors and do not necessarily reflect the views of the National Science Foundation nor the Cooperative AI Foundation.

References

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

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138.

Betancourt, M. (2011). Nested sampling with constrained hamiltonian monte carlo. In *AIP Conference Proceedings*, volume 1305, pages 165–172. American Institute of Physics.

Betancourt, M. (2012). Cruising the simplex: Hamiltonian monte carlo and the dirichlet distribution. In *AIP Conference Proceedings 31st*, volume 1443, pages 157–164. American Institute of Physics.

Buesing, L., Weber, T., Zwols, Y., Racaniere, S., Guez, A., Lespiau, J.-B., and Heess, N. (2018). Woulda, coulda, shoulda: Counterfactually-guided policy search. *arXiv preprint arXiv:1811.06272*.

Choudhary, K., Gupta, D., and Thomas, P. S. (2024). Icu-sepsis: A benchmark mdp built from real medical data. *arXiv preprint arXiv:2406.05646*.

Dearden, R., Friedman, N., and Andre, D. (2013). Model-based bayesian exploration. *arXiv preprint arXiv:1301.6690*.

Deisenroth, M. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472.

Diamond, S. and Boyd, S. (2016). Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913.

Fatemi, M., Killian, T. W., Subramanian, J., and Ghassemi, M. (2021). Medical dead-ends and learning to identify high-risk states and treatments. *Advances in Neural Information Processing Systems*, 34:4856–4870.

Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483.

Guo, K., Yunfeng, S., and Geng, Y. (2022). Model-based offline reinforcement learning with pessimism-modulated dynamics belief. *Advances in Neural Information Processing Systems*, 35:449–461.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31.

Herman, M., Gindele, T., Wagner, J., Schmitt, F., and Burgard, W. (2016). Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Artificial intelligence and statistics*, pages 102–110. PMLR.

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. (2015). The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189.

Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L. A., and Mark, R. (2020). Mimic-iv. *PhysioNet. Available online at: https://physionet.org/content/mimiciv/1.0/(accessed August 23, 2021)*.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823.

Kim, B., Farahmand, A.-m., Pineau, J., and Precup, D. (2013). Learning from limited demonstrations. *Advances in Neural Information Processing Systems*, 26.

Kim, B. and Oh, M.-h. (2023). Model-based offline reinforcement learning with count-based conservatism. In *International Conference on Machine Learning*, pages 16728–16746. PMLR.

LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*.

Lee, B. J., Lee, J., and Kim, K. E. (2021). Representation balancing offline model-based reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321.

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., et al. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118.

Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13:103–130.

Ng, A. Y., Russell, S., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using

deep networks in atari games. *Advances in neural information processing systems*, 28.

Ornik, M. and Topcu, U. (2021). Learning and planning for time-varying mdps using maximum likelihood estimation. *The Journal of Machine Learning Research*, 22(1):1656–1695.

Poupart, P. and Vlassis, N. (2008). Model-based bayesian reinforcement learning in partially observable domains. In *Proc Int. Symp. on Artificial Intelligence and Mathematics,*, pages 1–2.

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704.

Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591.

Reddy, S., Dragan, A., and Levine, S. (2018). Where do you think you're going?: Inferring beliefs about dynamics from behavior. *Advances in Neural Information Processing Systems*, 31.

Ross, S., Gordon, G., and 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*, pages 627–635. JMLR Workshop and Conference Proceedings.

Ross, S. and Pineau, J. (2008). Model-based bayesian reinforcement learning in large structured domains. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 2008, page 476. NIH Public Access.

Scobee, D. R. and Sastry, S. S. (2019). Maximum likelihood constraint inference for inverse reinforcement learning. *arXiv preprint arXiv:1909.05477*.

Stéphane, R., Gordon Geoffrey, J., and Andrew, B. J. (2010). No-regret reductions for imitation learning and structured prediction. *arXiv preprint arXiv: 1011.0686*.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tang, S., Makar, M., Sjoding, M., Doshi-Velez, F., and Wiens, J. (2022). Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. *Advances in Neural Information Processing Systems*, 35:34272–34286.

Tang, S., Modi, A., Sjoding, M., and Wiens, J. (2020). Clinician-in-the-loop decision making: Reinforcement learning with near-optimal set-valued policies. In *International Conference on Machine Learning*, pages 9387–9396. PMLR.

Van Hasselt, H. P., Hessel, M., and Aslanides, J. (2019). When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32.

Vanseijen, H. and Sutton, R. (2015). A deeper look at planning as learning from replay. In *International conference on machine learning*, pages 2314–2322. PMLR.

Wang, J., Hertzmann, A., and Fleet, D. J. (2005). Gaussian process dynamical models. *Advances in neural information processing systems*, 18.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. (2021). Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967.

Zhang, A., Lyle, C., Sodhani, S., Filos, A., Kwiatkowska, M., Pineau, J., Gal, Y., and Precup, D. (2020a). Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pages 11214–11224. PMLR.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. (2020b). Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2010). Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1255–1262.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. (2008). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator if your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Bayesian Inverse Transition Learning: Learning Dynamics from Near-Optimal Trajectories Supplementary Materials

A Non-Convex Feasible Region

When plotting our constraints (equations 7 and 9) within a three-dimensional subspace of a toy example, as illustrated in Figure 3, it becomes apparent that the feasible region can be non-convex, primarily due to the inverse operations within the constraints.

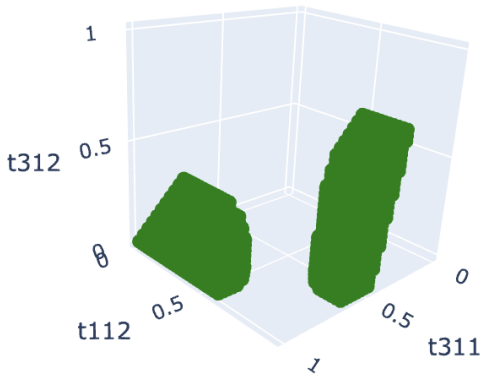


Figure 3: Example of a subspace of the feasible region defined by the constraints on T. The label 'tsas" indicates the probability of transitioning to state s' after being in state s and taking action a.

B Learning Dynamics and Rewards Jointly

A key concern is that assuming a known reward function (beyond simple success/failure signals) is often unreasonable, particularly in settings like robotics. To address this, we extend our ITL algorithm to simultaneously learn the dynamics T and rewards R.

This extension follows the exact same framework as our main method. However, to ensure the constraints remain linear with respect to both R and T, we adopt an iterative approach. Specifically, for the value function terms $(I - \gamma T_\pi)^{-1} R_\pi$, we fix the matrices T_π and vectors R_π to their values from the previous iteration (initialized with T^{MLE} and R_{prior}). This allows us to proceed with the same convex optimization procedure as before.

The new loss function becomes:

$$\min_{T,R} \sum_{s,a,s'} N_{s,a,s'} \left(T(s' \mid s,a) - T^{\text{MLE}}(s' \mid s,a) \right)^2 + \|R - R_{\text{prior}}\|^2 \tag{12}$$

Here, R_{prior} represents a sparse prior: 0 everywhere except at the goal state, which is assigned a value of 10. Table 3 summarizes performance on the Gridworld environment averaged over 20 different datasets, with $\epsilon = 3$. We compare standard ITL (using the true rewards R) against ITL-No-R, which learns the reward jointly with

the dynamics. Remarkably, ITL-No-R recovers a reward function that produces dynamics with performance comparable to ITL, despite not having access to the true rewards during training.

Table 3: Comparison of ITL (known rewards) and ITL-No-R (jointly learned dynamics and rewards) on the Gridworld environment ($\epsilon = 3$).

Metric	**ITL**	**ITL-No-R**
Value	**115.70 ± 2.02**	115.37 ± 2.02
Best matching	**0.65 ± 0.11**	0.64 ± 0.12
ϵ-matching	**0.73 ± 0.14**	**0.73 ± 0.14**
Total-Variation	**132.46 ± 6.40**	132.76 ± 6.44
Value Transfer	**41.79 ± 8.85**	41.15 ± 9.95
Best matching Transfer	**0.54 ± 0.14**	**0.54 ± 0.13**
ϵ-matching Transfer	**0.59 ± 0.09**	**0.59 ± 0.09**
Time	4.82 ± 2.92	**0.61 ± 0.60**

C Synthetic Environments

C.1 Gridworld Environment

The Gridworld environment is structured as follows:

- **Grid Size**: The world is a grid consisting of 5×5 tiles, resulting in a total of 25 distinct states.
- **Actions**: At each state, an agent can choose from four possible actions: move right, move up, move left, or move down.
- **Initial State**: The agent always starts from the bottom left corner of the grid, which is designated as the initial state.
- **Goal State**: The objective for the agent is to reach the goal state, located at the top right corner of the grid.
- **Dynamics**:
  - **Intended Actions**: When the agent selects an action, there is an 80% chance that it will move deterministically to the intended adjacent state.
  - **Slipping**: There is a 20% chance that the agent will slip, leading to a non-deterministic outcome. In such cases, the agent might end up in any one of the four neighboring tiles (right, left, up, down) of the intended state.
  - **Wall Interactions**: If an action would result in the agent moving into a wall (the edge of the grid), the agent remains in its current state. This mechanic ensures that the agent does not leave the confines of the grid.

The definition of the reward R in the grid world environment is structured as follows:

- **Soft-Wall Penalty**: If the agent attempts to move across a tile designated as a *soft-wall*, it incurs a penalty of -5 reward points for each attempt. This mechanic discourages the agent from crossing these specific tiles.
- **Movement Penalty**: For every other tile that the agent moves through, it receives a minor penalty of -0.1 reward points. This encourages the agent to find the shortest possible path to the goal.
- **Goal Reward**: Upon successfully reaching the goal state, the agent is awarded $+10$ reward points. This substantial reward signifies the completion of the episode and serves as the primary incentive for the agent to navigate the grid efficiently.

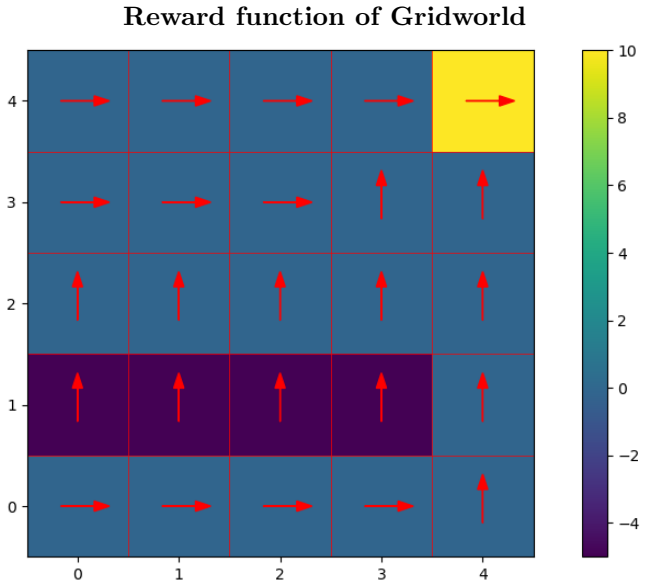


Figure 4: Visualization of the grid world environment. Each square in the 5x5 grid represents a unique state, colored based on the associated reward. The 'soft-wall' tiles are distinctively colored to represent a reward of -5. Red arrows on each tile indicate the direction of the optimal policy from that state, leading towards the goal state at the top right corner, which is marked in a different color and has a reward of 10. The starting point is at the bottom left corner, from where the arrows guide the optimal path through the grid.

This reward structure is designed to balance the objective of reaching the goal state as quickly as possible with the challenge of navigating around soft-wall tiles. The penalties for unnecessary movements and soft-wall crossings ensure that the agent must carefully consider each action, while the reward for reaching the goal state motivates the agent to complete its objective efficiently. Figure 4 shows the grid world environment, showcasing both the rewards for each state and the optimal policy indicated by red arrows. This environment presents a challenge for an agent to learn the most efficient path from the initial state to the goal state, taking into account the probabilistic nature of movement due to slipping. The deterministic and non-deterministic outcomes necessitate strategic planning and adaptability in the agent's approach to navigating the grid.

C.1.1 Transfer Task

In the transfer task, we preserve the structure of the original grid world environment, with no alterations to the state space or action set. However, we introduce a significant change to the reward function: the location of the soft-wall is shifted, thereby altering the reward landscape. This modification necessitates the derivation of a new optimal policy that accounts for the updated rewards and navigates the agent from the initial state to the goal state via a different path. See Figure 5.

The updated reward function is defined identically to the original environment, with soft-wall tiles incurring a penalty of -5 reward points, other tiles a penalty of -0.1 points, and a reward of $+10$ points assigned upon reaching the goal state. The shift in the soft-wall location directly affects the agent's trajectory, demonstrating the agent's ability to adjust its policy in response to changes in environmental dynamics. This transfer task effectively evaluates the flexibility and robustness of the learned policy.

C.2 Randomworld Environment

We introduce the RandomWorld environment (inspired by (Jiang et al., 2015)), which is designed to evaluate the performance of reinforcement learning algorithms under conditions of high uncertainty and stochasticity. The RandomWorld environment is characterized by the following properties:

- **State Space**: The environment comprises 15 distinct states.

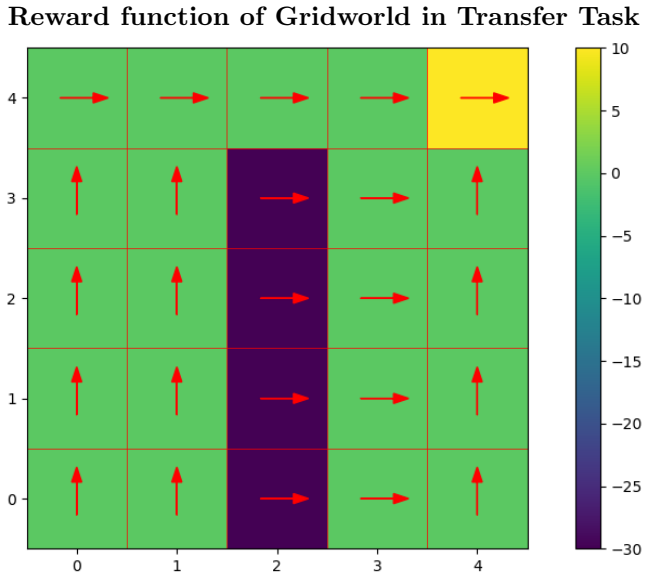


Figure 5: The grid world environment after the transfer task modification. The soft-wall tiles, previously located, have been repositioned, visibly changing the reward distribution across the grid. As a result, the optimal policy, indicated by red arrows, now follows a novel route that adapts to the new reward structure, aiming to minimize penalties and maximize returns en route to the goal state at the top right corner.

- **State Space**: The environment comprises 5 distinct actions.
- **Dynamics** : For each state-action pair, 5 successor states are chosen at random to have nonzero transition probability. These probabilities are drawn independently from Uniform$[0, 1]$ and normalized to sum to one.
- **Initial State Distribution**: The initial state for each episode is selected with uniform probability across all 15 states.
- **Absence of a Goal State**: RandomWorld is devoid of a specified goal state, thus simulating scenarios where an agent's exploration is continuous and without a predetermined endpoint.
- **Reward Function**: Rewards are assigned randomly yet structured such that state 1 yields the highest expected reward, and state 15 the lowest. Specifically, the reward for state s, $R(s)$, is uniformly distributed within the interval $[16 - s - 1, 16 - s]$, aligning with the descending order of state desirability.

The inherent randomness in state transitions and rewards within RandomWorld poses a significant challenge to reinforcement learning strategies, necessitating the development of policies that are robust to uncertainty and variability in environmental dynamics.

C.2.1 Transfer Task

In the RandomWorld environment's transfer task, we introduce a modification to the reward function while preserving all other environmental characteristics. This adjustment is aimed at assessing the adaptability of reinforcement learning algorithms when confronted with a new reward paradigm. The specifics of the transfer task are as follows:

- **Inverted Reward Structure**: We reverse the ranking of state desirability; state 1 is now the least desirable state, and state 15 is the most desirable.
- **Random Reward Generation**: The reward for state s in the transfer task, $R_{\text{transfer}}(s)$, is determined by a random draw from a uniform distribution over the range $[s-1, s]$, thus ensuring that higher state numbers correspond to higher expected rewards.

This reversal in the reward hierarchy necessitates that the agent recalibrates its policy to align with the new set of rewards. It provides an insightful measure of the algorithm's capacity to adapt to drastic changes in the reward structure within a stochastic environment.

C.3 Generating batch data $\mathcal{D}$

A critical component of our experiments in offline reinforcement learning is the generation of a batch data $\mathcal{D}$, which is constructed based on a predefined coverage percentage of the state space and a given ϵ, measuring the degree of optimality of the expert $\pi_\epsilon(.|.;T^*)$ with respect to the true unknown dynamics T^*. The batch data $\mathcal{D}$ is created through the following procedure:

1. **State Selection**: We randomly select a certain percentage of the total states, corresponding to the coverage parameter, to include in our batch data.
2. **Action Selection**: For each state s included in our selection, we identify actions in the ϵ-ball $\epsilon(s;T^*)$, with respect to the ϵ-optimal expert $\pi_\epsilon(.|.;T^*)$.
3. **Transition Sampling**: We sample K transitions for each state-action pair (s,a) from the true dynamics T^*.
4. **Dataset Construction**: The batch dataset $\mathcal{D}$ is comprised of the transitions collected, each represented as a tuple (s,a,s'), with s as the state, a as the action, s' as the next state.

In our experimental setup, we define the value of K, which dictates the number of transitions sampled for each state-action pair that aligns with the ϵ-optimal expert policy $\pi_\epsilon(.|.;T^*)$. For the Gridworld environment, we set $K = 10$, acknowledging the larger state space and the need for a comprehensive dataset that encapsulates the dynamics around the optimal policy. In contrast, for RandomWorld, we set $K = 5$, suitable for its smaller size and complexity.

C.4 Averaging Procedure for Experimental Results

To achieve statistical rigor in our experiments, we average our results over multiple independently generated datasets for both Gridworld and RandomWorld environments:

- **Gridworld**: We generate 100 independent batch datasets for the Gridworld environment. The experimental results for each dataset are recorded, and the final result is obtained by averaging these outcomes.
- **RandomWorld**: For the RandomWorld environment, we create 10 independent instances of the environment. Each of these RandomWorld instances is accompanied by 50 independently generated batch datasets, leading to a total of 500 unique datasets (10 worlds multiplied by 50 datasets each). The experimental outcomes across these datasets are compiled, and their average is computed to determine the overall performance in the RandomWorld environment.

This methodology, involving the independent generation of each dataset and each world instance, provides a comprehensive and unbiased evaluation of the algorithms, ensuring that our results are not influenced by any specific configuration or sample of the environment.

D Real-life ICU Environment

Following the experiments within a synthetic environments, we now transition to the evaluation of our methodology in a real-world scenario. To this end, we selected the Medical Information Mart for Intensive Care IV (MIMIC-IV) dataset as our experimental field. This dataset offers a rich, diverse, and challenging setting for testing our method, especially given its potential to contribute to advancements in healthcare analytics and patient care strategies.

D.1 About MIMIC-IV Dataset

The MIMIC-IV dataset, developed by the MIT Lab for Computational Physiology and publicly available, aggregates a vast range of anonymized health data from critical care units at Beth Israel Deaconess Medical Center in Boston. Covering over a decade's worth of patient admissions, it provides detailed records on demographics, vital signs, lab tests, medications, and more, establishing itself as a critical resource for healthcare model development. Its comprehensive scope spans all patient care aspects, enabling the creation of holistic models for predicting diverse patient outcomes. The dataset's richness lies in its variety, covering over 40,000 patients of different ages, ethnicities, and conditions, and its granularity, offering high-resolution data points and time-stamped records, which are essential for developing precise, dynamic healthcare models. Moreover, MIMIC-IV's public accessibility fosters a global research community's collaboration, enhancing healthcare analytics advancements.

Utilizing the MIMIC-IV dataset, we showcase out the learning applicability of our method in real-world healthcare, to get valuable insights from the data in such a complicated environemnt.

D.2 Data Preprocessing for Hypotension Analysis

In our investigation into hypotension within ICU settings, we tailored our preprocessing steps to exclusively include patients affected by this condition. Our methodology commenced with the application of specific filters on the MIMIC-IV dataset to accurately identify the patient cohort of interest. These filters were designed to capture adults aged 18 to 80 years, who had ICU stays of a minimum duration of 24 hours, and exhibited Mean Arterial Pressure (MAP) readings of 65mmHg or below, indicative of acute hypotension.

The analytical framework of our study is built around a carefully selected set of five clinical variables that constitute the state space, namely: creatinine levels (**(Crea)**), Glasgow Coma Scale score (**(GCS)**), mean blood pressure (**(BP)**), and the ration of partial pressure of oxygen, over fraction of inspired oxygen (**(O2)**). The action space encompasses two primary treatment modalities: intravenous (IV) fluid bolus therapy and vasopressor therapy. This precise filtering approach yielded a dataset comprising 1,684 distinct ICU admissions, from which we derived approximately 100,000 tuples (state, action, next_state) $\in \mathcal{D}$. This dataset serves as the foundation to evaluate our method and the baselines.

D.3 State Space Construction

The state space for our model is constructed by discretizing five key clinical variables extracted from the MIMIC-IV dataset: partial pressure of oxygen, fraction of inspired oxygen, mean blood pressure, Glasgow Coma Scale (GCS), and creatinine levels. Discretization involves binning these variables into distinct categories based on clinically relevant thresholds, as follows:

Table 4: Discretization of Clinical Variables into Bins

Abbrev	Clinical Variable	Threshold	Bin Value
O2	Partial Pressure of Oxygen / Fraction Inspired Oxygen	≥ 200	0
O2	Partial Pressure of Oxygen / Fraction Inspired Oxygen	< 200 and ≥ 100	1
O2	Partial Pressure of Oxygen / Fraction Inspired Oxygen	< 100	2
BP	Mean Blood Pressure	≥ 70 mmHg	0
BP	Mean Blood Pressure	< 70 mmHg	1
GCS	Glasgow Coma Scale (GCS)	≤ 12	0
GCS	Glasgow Coma Scale (GCS)	> 14	1
Crea	Creatinine	≤ 1.9 mg/dL	0
Crea	Creatinine	> 1.9 and ≤ 4.9 mg/dL	1
Crea	Creatinine	> 4.9 mg/dL	2

Based on the binning schema presented, the state space comprises all possible combinations of these bins, leading to a total of $3 \times 3 \times 2 \times 2 = 36$ unique states. This structure effectively captures diverse clinical scenarios within a manageable framework for analyzing the dynamics of hypotension treatment. To illustrate the discretization process and the resultant bin mapping, consider a hypothetical patient data point with the following clinical variable values:

- Partial Pressure of Oxygen / Fraction Inspired Oxygen: 150
- Mean Blood Pressure: 65 mmHg
- Glasgow Coma Scale: 10
- Creatinine: 5 mg/dL

Based on the discretization schema provided in Table 4, this patient data point would be mapped to the following bins:

- Partial Pressure of Oxygen / Fraction Inspired Oxygen (150): Bin 1 (since $100 \leq 150 < 200$)
- Mean Blood Pressure (65 mmHg): Bin 1 (since $65 < 70$ mmHg)
- Glasgow Coma Scale (GCS) (10): Bin 0 (since $10 \leq 12$)
- Creatinine (5 mg/dL): Bin 2 (since $5 \geq 4.9$)

Thus, the tuple $(150, 65, 10, 5)$ would be mapped to the discretized state $(1, 1, 0, 2)$ according to our binning process. This discretization approach allows us to capture a comprehensive yet manageable representation of the patient's clinical status, facilitating the application of our offline reinforcement learning model to infer the unknown dynamics T^*.

D.4 Action Space Definition

The action space in our model encapsulates the range of possible treatments administered to patients suffering from hypotension. It consists of four discrete actions, each representing a specific treatment strategy. The actions are enumerated as follows:

Action	Description
0	No treatment administered
1	Vasopressor therapy administered
2	Intravenous (IV) fluid bolus administered
3	Both vasopressor therapy and IV fluid bolus

Table 5: Definition of Actions in the Treatment Strategy Space

Each action is designed to reflect the clinical decisions made in the intensive care unit for managing patients' blood pressure levels. Action 0 (no treatment) represents a conservative approach, where no immediate intervention is applied. Action 1 (vasopressor therapy) and Action 2 (IV fluid bolus) correspond to the administration of specific treatments aimed at increasing blood pressure.

D.5 Reward Function Definition

The reward function, $R(s)$, quantifies the desirability of each state $x = (s_1, s_2, s_3, s_4)$ based on the bin values corresponding to the discretized clinical variables (each s_i correspond to a bin number). Formally, the reward function is defined as:

$$R(s) = 60 - 10 \times (s_1 + s_2 + s_4)$$

This formulation encapsulates our intuition that higher bin values for any of the clinical variables signify a deterioration in the patient's condition, indicating more severe or dangerous vital signs. Consequently, the reward decreases linearly by a factor of 10 for each increment in the bin values of the state components. The choice of this linear penalty ensures a straightforward interpretation of the state's severity, with a base reward of 60 being adjusted downward based on the sum of the bin values in the state.

While this reward function offers a reasonable approximation for assessing the clinical states in the context of hypotension, it is important to acknowledge that other formulations could be equally valid. The essential

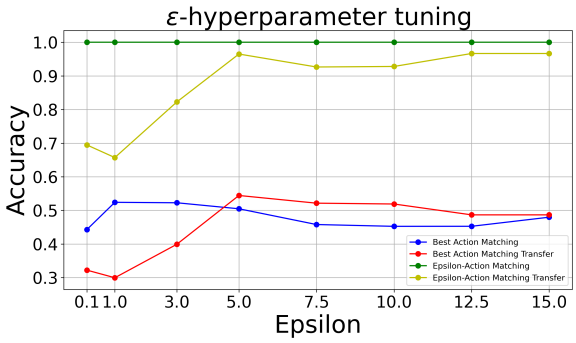


Figure 6: ITL performance across ϵ values on a held-out dataset for the healthcare setting.

criterion for any chosen reward function is its ability to accurately differentiate between clinically favorable and unfavorable states, thereby guiding the reinforcement learning model towards optimizing treatment strategies that mitigate the risks associated with hypotension.

D.6 Transfer Task and Modified Reward Function

We introduce a transfer task to evaluate the model's adaptability and performance under a different reward function and keeping evrything else identical. The modified reward function for the transfer task is defined as:

$$R_{\text{transfer}}(s) = 60 - 10 \times (s_2 + s_4)$$

This adjustment means that the reward now decreases quadratically, rather than linearly, with the values of the features within each state $s = (s_1, s_2, s_3, s_4)$. The quadratic penalty intensifies the impact of higher bin values, more aggressively penalizing states indicative of worsening patient conditions. This change aims to test the method's sensitivity and response to more severe deteriorations in the clinical variables, pushing the reinforcement learning algorithm to prioritize avoiding high-risk states with even greater emphasis. Such a modification in the reward function's structure is pivotal for assessing the robustness and flexibility of our method. It allows us to explore how different reward formulations can influence decision-making strategies in the context of medical treatment optimization, particularly under scenarios with escalating risks.

D.7 How to Tune ϵ:

In the healthcare setting, we only have access to an offline dataset. To select an appropriate ϵ, we evaluate the performance of ITL on a held-out validation set. We choose the ϵ that performs best on this validation set based on the **Best/ϵ-ball action matching metric** for both standard and transfer tasks (defined by different reward functions). As shown in Figure 6, $\epsilon = 5$ appears to be suitable, and this value is used for training and computing the results in the real-life healthcare experiments (See Table 2). We also present results for $\epsilon = 10$ and $\epsilon = 15$, as these values are also reasonable. Results for $\epsilon = 10$ and $\epsilon = 15$ are provided in Tables 9 and 10 in the Appendix, showing similar performance.

D.8 Construction of the ϵ-Optimal Expert Policy π_ϵ

In the absence of explicit knowledge about the true dynamics T^* governing the environment, our methodology for obtaining an estimate of the ϵ-optimal expert policy, $\widehat{\pi}_\epsilon(\cdot|\cdot; T^*)$, leverages the historical batch data $\mathcal{D}$ collected from the ICU. We define an action a to be valid for a state s if and only if action a was executed in at least 5% of the instances where state s was observed in $\mathcal{D}$. Actions not meeting this criterion are considered invalid for the state, reflecting an approach that filters actions based on their historical prevalence and relevance to specific states.

Leveraging domain knowledge within the critical care domain, we set the ϵ parameter to 5 for our experiments. This parameter choice reflects a balance, aiming to capture the degree of optimality in the actions taken by medical professionals in the ICU, under the assumption that the most frequently taken actions represent a near-optimal strategy given the complex dynamics and uncertainties inherent in patient care. Our experimental results

demonstrate non-significant changes across various ϵ values, suggesting that the selected ϵ-optimal policy robustly encapsulates the expert behavior within the dataset, without significant sensitivity to the exact ϵ threshold.

E Results

We include in this section the full set of results for the synthetic worlds where we include expert with various degree of optimality leading to 20% and 0% of stochastic-policy states) as well as multiple ϵ values as well for the results with the real healthcare dataset.

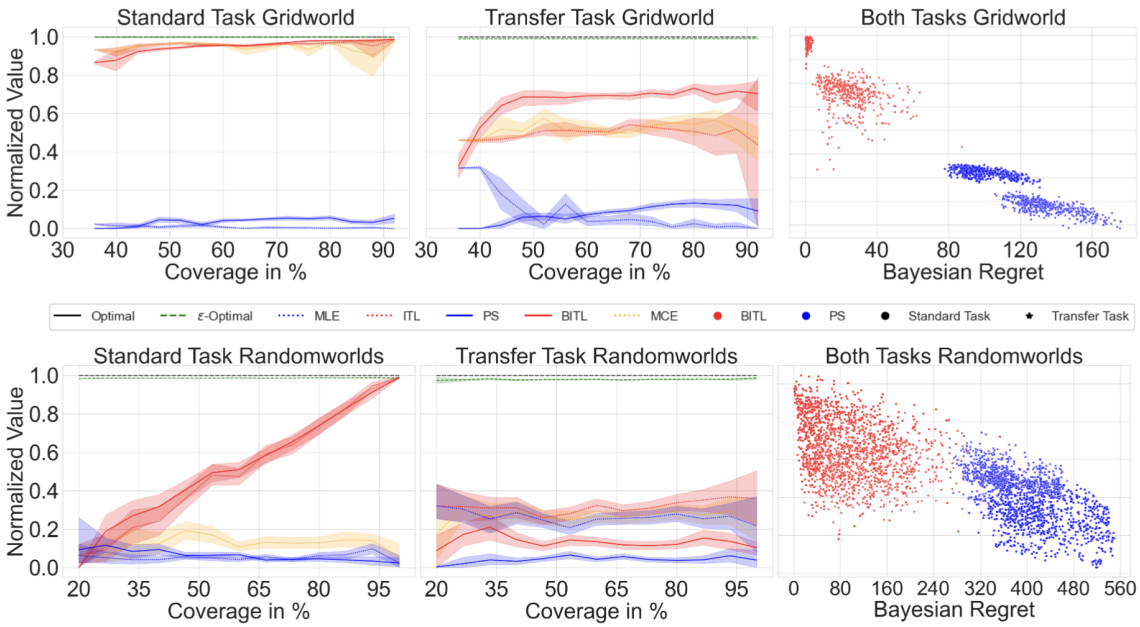


Figure 7: (40% stochastic-policy states, ϵ Gridworld = 3.0, ϵ's RandomWorlds = [2.37, 2.53, 2.21, 1.89, 4.27, 1.74, 2.37, 2.84, 2.37, 2.53]) Top row: Normalized Value vs. Coverage for Gridworld (left: Standard Task, middle: Transfer Task), Bottom row: Normalized Value vs. Coverage for Randomworlds (left: Standard Task, middle: Transfer Task). Rightmost plots: Normalized Value vs. Bayesian Regret of both Tasks (top: Gridworld, bottom: Randomworlds).

Table 6: Gridworld and Randomworlds Results (40% stochastic-policy states, ϵ Gridworld = 3.0, ϵs RandomWorlds = [2.37, 2.53, 2.21, 1.89, 4.27, 1.74, 2.37, 2.84, 2.37, 2.53])

	Gridworld				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.65 ± 0.12	**0.78 ± 0.08**	0.65 ± 0.12	0.37 ± 0.06	0.38 ± 0.02
ϵ matching transfer	0.5 ± 0.11	**0.64 ± 0.08**	0.51 ± 0.11	0.39 ± 0.07	0.39 ± 0.02
Best matching	0.55 ± 0.11	**0.64 ± 0.08**	0.56 ± 0.11	0.31 ± 0.06	0.29 ± 0.02
Best matching transfer	0.45 ± 0.1	**0.54 ± 0.08**	0.45 ± 0.11	0.34 ± 0.07	0.31 ± 0.02
Time	117.47 ± 26.34	14384.26 ± 489.49	0.32 ± 0.26	**0.01 ± 0.00**	8.65 ± 0.24
Total Variation	137.62 ± 4.39	159.36 ± 3.98	**137.05 ± 4.32**	141.37 ± 3.8	160.05 ± 4.43
Value CVaR 1%	56.05 ± 15.08	**104.64 ± 5.43**	103.87 ± 15.67	-5.34 ± 0.54	-4.33 ± 1.68
Value CVaR 2%	76.7 ± 17.51	**107.36 ± 5.19**	106.46 ± 12.51	-5.18 ± 0.51	-3.49 ± 1.56
Value CVaR 5%	108.76 ± 26.54	109.04 ± 4.26	**109.7 ± 9.01**	-4.71 ± 0.47	-2.35 ± 1.35
Nbr constraints violated	3.06 ± 3.68	**0.0 ± 0.0**	**0.0 ± 0.0**	23.13 ± 6.59	16.37 ± 3.21
	Randomworlds				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.54 ± 0.12	0.75 ± 0.12	**0.76 ± 0.13**	0.43 ± 0.11	0.3 ± 0.05
ϵ matching transfer	0.46 ± 0.12	0.38 ± 0.1	**0.5 ± 0.13**	0.46 ± 0.12	0.31 ± 0.05
Best matching	0.38 ± 0.13	0.57 ± 0.12	**0.58 ± 0.15**	0.29 ± 0.11	0.19 ± 0.05
Best matching transfer	0.34 ± 0.13	0.26 ± 0.09	**0.37 ± 0.15**	0.34 ± 0.13	0.2 ± 0.05
Time	36.61 ± 26.36	5561 ± 223.43	0.65 ± 0.39	**0.0 ± 0.0**	2.61 ± 0.36
Total Variation	111.08 ± 2.42	123.3 ± 4.06	**102.02 ± 4.84**	111.07 ± 2.3	127.02 ± 2.66
Value CVaR 1%	-522.35 ± 5.49	-423.21 ± 29.84	**-404.02 ± 0.13**	-525.94 ± 3.86	-452.19 ± 22.88
Value CVaR 2%	-514.69 ± 7.06	-398.25 ± 30.3	**-397.26 ± 5.34**	-519.63 ± 6.42	-444.69 ± 20.16
Value CVaR 5%	-459.71 ± 23.48	-364.34 ± 34.06	**-366.43 ± 16.03**	-481.98 ± 23.31	-434.24 ± 17.22
Nbr constraints violated	11.81 ± 6.47	**0.0 ± 0.0**	**0.0 ± 0.0**	17.23 ± 6.75	11.66 ± 3.12

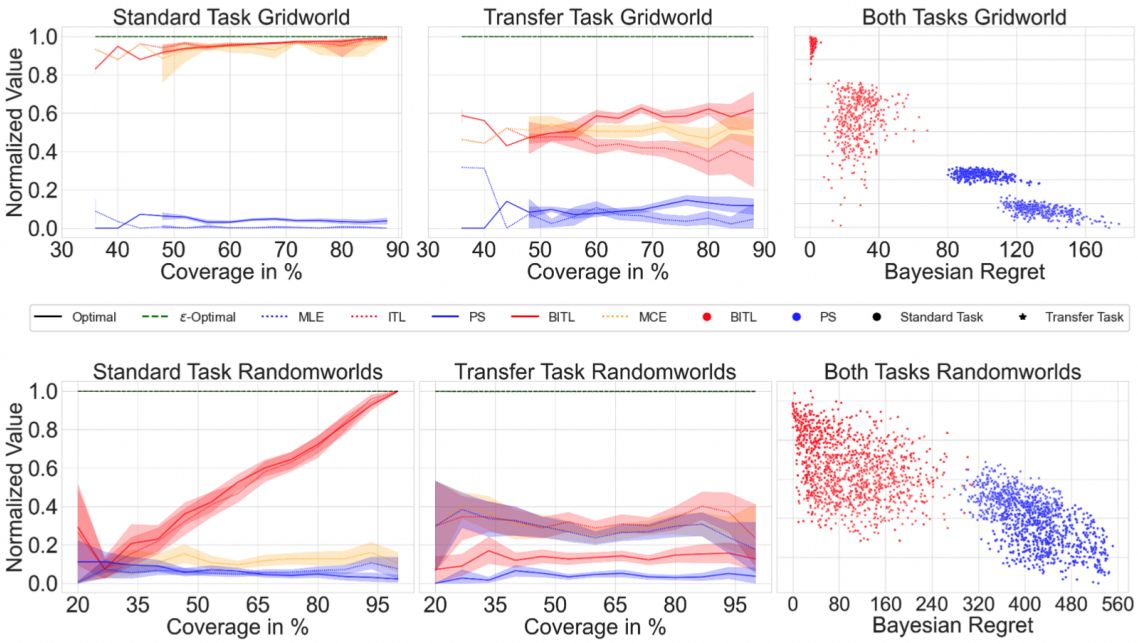


Figure 8: (20% stochastic-policy states, ϵ Gridworld = 0.3, ϵs RandomWorlds = [0.79, 1.11, 0.32, 0.63, 1.16, 0.33, 0.57, 0.47, 0.99, 0.32]) Top row: Normalized Value vs. Coverage for Gridworld (left: Standard Task, middle: Transfer Task), Bottom row: Normalized Value vs. Coverage for Randomworlds (left: Standard Task, middle: Transfer Task). Rightmost plots: Normalized Value vs. Bayesian Regret of both Tasks (top: Gridworld, bottom: Randomworlds).

Table 7: Comparison of Gridworld and Randomworlds (20% stochastic-policy states, ϵ Gridworld = 0.3, ϵs RandomWorlds = [0.79, 1.11, 0.32, 0.63, 1.16, 0.33, 0.57, 0.47, 0.99, 0.32])

	Gridworld				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.64 ± 0.11	**0.74 ± 0.08**	0.64 ± 0.1	0.37 ± 0.06	0.32 ± 0.01
ϵ matching transfer	0.41 ± 0.06	**0.51 ± 0.05**	0.4 ± 0.06	0.32 ± 0.04	0.32 ± 0.01
Best matching	0.59 ± 0.11	**0.69 ± 0.08**	0.6 ± 0.1	0.34 ± 0.06	0.29 ± 0.02
Best matching transfer	0.4 ± 0.06	**0.5 ± 0.05**	0.39 ± 0.06	0.3 ± 0.04	0.31 ± 0.02
Time	114.66 ± 27.54	14267.26 ± 529.69	0.75 ± 0.53	**0.01 ± 0.00**	7.82 ± 0.32
Total Variation	**137.81 ± 3.79**	160.22 ± 3.89	138.71 ± 3.5	141.67 ± 3.11	160.4 ± 3.62
Value CVaR 1%	17.94 ± 7.28	**103.22 ± 7.96**	102.83 ± 33.76	-5.23 ± 0.56	-3.49 ± 0.56
Value CVaR 2%	54.96 ± 16.37	**106.6 ± 7.27**	104.7 ± 25.52	-5.04 ± 0.57	-2.9 ± 0.63
Value CVaR 5%	106.58 ± 42.1	**109.18 ± 5.81**	107.58 ± 16.89	-4.74 ± 0.5	-1.65 ± 0.92
Nbr constraints violated	0.82 ± 2.53	**0.0 ± 0.0**	**0.0 ± 0.0**	20.6 ± 5.73	14.19 ± 2.53
	Randomworlds				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.5 ± 0.13	0.72 ± 0.13	**0.73 ± 0.15**	0.38 ± 0.13	0.24 ± 0.05
ϵ matching transfer	0.42 ± 0.12	0.34 ± 0.11	**0.47 ± 0.13**	0.42 ± 0.12	0.25 ± 0.05
Best matching	0.45 ± 0.12	0.66 ± 0.13	**0.68 ± 0.15**	0.34 ± 0.12	0.21 ± 0.05
Best matching transfer	0.39 ± 0.13	0.31 ± 0.1	**0.43 ± 0.14**	0.39 ± 0.13	0.22 ± 0.05
Time	31.96 ± 25.36	5493 ± 271.62	0.68 ± 0.69	**0.0 ± 0.0**	2.24 ± 0.28
Total Variation	112.12 ± 2.43	124.38 ± 4.55	**103.32 ± 4.91**	112.07 ± 2.23	128.34 ± 2.51
Value CVaR 1%	-520.09 ± 1.89	**-401.79 ± 23.07**	-408.13 ± 0.14	-523.71 ± 1.51	-428.06 ± 0.79
Value CVaR 2%	-519.89 ± 1.72	**-388.77 ± 29.86**	-396.53 ± 5.59	-520.58 ± 1.96	-426.44 ± 1.25
Value CVaR 5%	-450.23 ± 25.6	-356.37 ± 33.32	**-352.47 ± 21.57**	-451.59 ± 23.83	-421.73 ± 2.52
Nbr constraints violated	7.03 ± 3.94	**0.0 ± 0.0**	**0.0 ± 0.0**	9.5 ± 3.95	9.22 ± 2.52

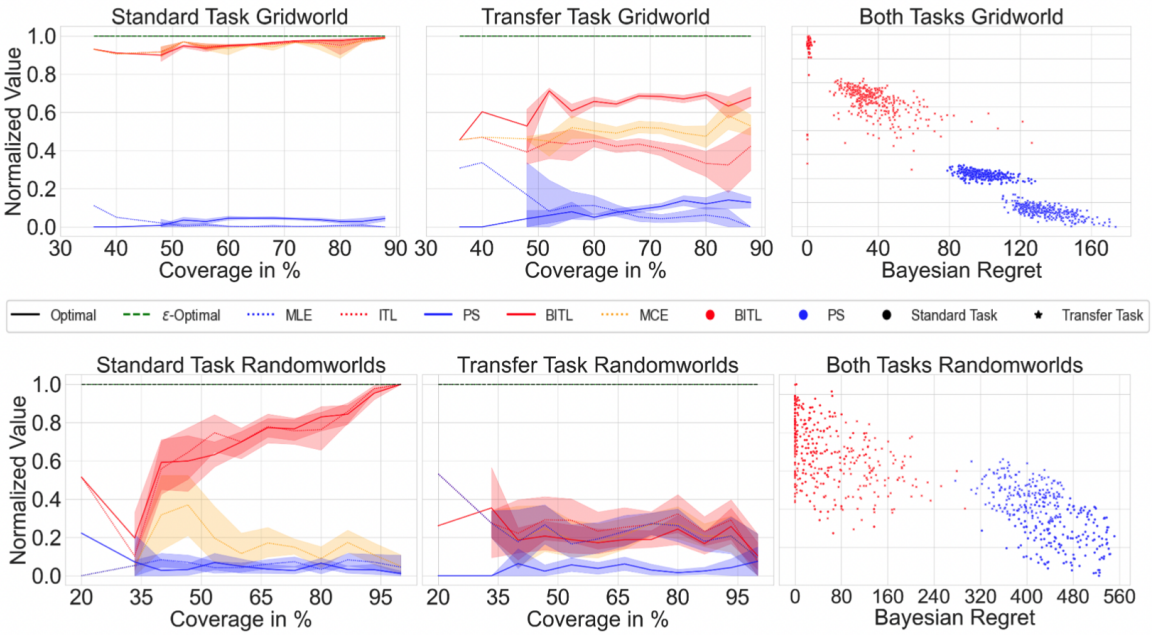


Figure 9: (0% stochastic-policy states, ϵ Gridworld = 0.0, ϵs RandomWorlds = 0.0) Top row: Normalized Value vs. Coverage for Gridworld (left: Standard Task, middle: Transfer Task), Bottom row: Normalized Value vs. Coverage for Randomworlds (left: Standard Task, middle: Transfer Task). Rightmost plots: Normalized Value vs. Bayesian Regret of both Tasks (top: Gridworld, bottom: Randomworlds).

Table 8: Comparison of Gridworld and Randomworlds (0% stochastic-policy states, ϵ Gridworld = 0.0, ϵs RandomWorlds = 0.0)

	Gridworld				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.68 ± 0.09	**0.75 ± 0.07**	0.68 ± 0.09	0.37 ± 0.06	0.29 ± 0.01
ϵ matching transfer	0.4 ± 0.06	**0.49 ± 0.05**	0.38 ± 0.06	0.29 ± 0.04	0.31 ± 0.01
Best matching	0.68 ± 0.09	**0.75 ± 0.07**	0.68 ± 0.09	0.37 ± 0.06	0.29 ± 0.01
Best matching transfer	0.4 ± 0.06	**0.49 ± 0.05**	0.38 ± 0.06	0.29 ± 0.04	0.31 ± 0.01
Time	119.86 ± 22.01	15284.26 ± 391.76	4.36 ± 1.46	**0.01 ± 0.0**	11.62 ± 0.55
Total Variation	**137.38 ± 3.3**	158.41 ± 3.37	138.6 ± 3.06	141.52 ± 2.55	160.22 ± 2.95
Value CVaR 1%	75.34 ± 19.3	**104.24 ± 7.94**	104.24 ± 37.65	-5.23 ± 0.48	-4.11 ± 0.22
Value CVaR 2%	102.59 ± 21.97	**108.18 ± 6.74**	105.64 ± 26.95	-5.2 ± 0.43	-3.65 ± 0.34
Value CVaR 5%	106.74 ± 23.68	**110.83 ± 5.93**	108.95 ± 18.49	-4.94 ± 0.35	-1.88 ± 0.84
Nbr constraints violated	0.08 ± 0.59	**0.0 ± 0.0**	**0.0 ± 0.0**	22.4 ± 5.23	13.15 ± 1.83
	Randomworlds				
Method	MCE	BITL	ITL	MLE	PS
ϵ matching	0.46 ± 0.13	0.77 ± 0.14	**0.78 ± 0.15**	0.32 ± 0.13	0.19 ± 0.05
ϵ matching transfer	0.37 ± 0.14	0.34 ± 0.13	**0.39 ± 0.16**	0.36 ± 0.15	0.23 ± 0.05
Best matching	0.46 ± 0.13	0.77 ± 0.14	**0.78 ± 0.15**	0.32 ± 0.13	0.19 ± 0.05
Best matching transfer	0.37 ± 0.14	0.34 ± 0.13	**0.39 ± 0.16**	0.36 ± 0.15	0.23 ± 0.05
Time	32.11 ± 27.16	5493 ± 216.83	0.91 ± 0.55	**0.0 ± 0.0**	2.35 ± 0.46
Total Variation	111.47 ± 2.27	117.66 ± 11.76	**101.44 ± 2.87**	111.84 ± 2.26	128.14 ± 2.54
Value CVaR 1%	-520.09 ± 3.18	**-343.11 ± 0.56**	-350.2 ± 11.25	-521.78 ± 0.0	-428.28 ± 0.05
Value CVaR 2%	-517.71 ± 2.71	**-278.91 ± 36.29**	-343.9 ± 20.59	-520.09 ± 2.39	-427.96 ± 0.4
Value CVaR 5%	-446.29 ± 4.08	**-246.36 ± 41.54**	-254.84 ± 47.72	-447.26 ± 3.11	-425.21 ± 1.24
Nbr constraints violated	4.9 ± 1.71	**0.0 ± 0.0**	**0.0 ± 0.0**	6.9 ± 1.91	8.59 ± 2.03

Table 9: Healthcare dataset results ($\epsilon = 10$)

	Standard Task					Transfer Task				
Method	**MLE**	ITL	MCE	BITL	PS	MLE	ITL	MCE	BITL	PS
Best matching	0.33	**0.47**	0.38	0.46	0.31	0.34	**0.50**	0.10	0.47	0.34
ϵ matching	0.52	**1**	0.68	**1**	0.51	0.58	**0.94**	0.16	**0.94**	0.58
Nbr Constraints	49	**0**	43	**0**	52	-	-	-	-	-
Time	-	**2.31**	180	-	-	-	-	-	-	-
Bayesian Regret	-	-	-	**0.55**	10	-	-	-	**0.44**	5.40

Table 10: Healthcare dataset results ($\epsilon = 15$)

	Standard Task					Transfer Task				
Method	**MLE**	ITL	MCE	BITL	PS	MLE	ITL	MCE	BITL	PS
Best matching	0.33	**0.48**	0.25	0.49	0.31	0.34	**0.47**	0.15	**0.47**	0.34
ϵ matching	0.52	**1**	0.43	**1**	0.51	0.58	**0.97**	0.31	**0.97**	0.58
Nbr Constraints	49	**0**	51	**0**	52	-	-	-	-	-
Time	-	**1.78**	94	-	-	-	-	-	-	-
Bayesian Regret	-	-	-	**1.02**	10	-	-	-	**0.56**	5.40