Value Interpretable Dynamic Treatment Regimes

Anonymous authors
Paper under double-blind review

Abstract

Understanding the rationale behind the decision-making process proves invaluable in various applications of AI and ML algorithms, such as in healthcare. In this paper, we propose Value Interpretable Dynamic Treatment Regimes (VIDTR) as a method to identify an optimal decision tree policy when finding the best treatment regime in sequential settings. Our approach works by identifying and integrating rewards over state regions that geometrically form a disjoint union of boxes at each iteration step. We make use of this property by designing an appropriate data structure and performing optimization under suitable tightness assumptions on the rewards and transition probabilities. Finally, through various experiments, we examine how our method performs compared to various existing methods using simulated and real-world data.

1 Introduction

It is no coincidence that widespread adoption of AI algorithms is the requirement for this age. For adoption of these algorithms in human-centric fields such as healthcare, interpretability of the techniques is a necessity. In order for doctors to make quick decisions about the treatment to give to a patient, the output of the algorithm must be interpretable. Recently, there has been a surge in research dedicated to the design of artificial intelligence tools in healthcare ((3), (1), (7)).

(14) posits that we are to save up to 10^8 dollars in 10 years by implementing AI systems in healthcare. To enable the fusion of AI tools in healthcare, there has been an uptake of interest in the development of human-algorithm 'centaurs' ((20), (24), (25)). These centaurs usually work based on the principle of generating algorithmic feedback, which goes through external human validation to generate a final output. As an example in (20), a survey is used to generate expert human output that can guide the decision making provided by the machine learning algorithm.

To further develop this line of research, we wish to develop tools that ease the burden of decision making on doctors. Toward that goal, the creation of algorithms whose output is interpretable is necessary. Reinforcement learning-a branch of machine learning which enables the automation of dynamic decision making is to be deployed in fields such as healthcare. (32) provides a recent account of the different reinforcement learning methods that are being deployed in this field.

(30) gives a deep reinforcement learning approach for real time treatment of sepsis, where the authors propose a Weighted Dueling Deep Q-network with embedded human expertise. In this paper, a Q target value with adaptive dynamic weight is designed to improve the accuracy of the estimate. In (13) the authors come up with a continuous space and action solution to provide recommendations for the treatment of sepsis inspired by the Deep Deterministic Policy Gradient (DDPG) method. In (17) a supervised learning framework is used to determine the actions that are to be taken by human experts. Further deep reinforcement learning is used to estimate the value function for expert actions with highest probabilities and recommendations are made amongst treatment options.

The dynamic treatment regimes literature (6) provides the concept of a DTR from a statistical perspective and lays the groundwork for future developments. (22) work on structured nested models is foundational for the development of statistical methods in dynamic treatment regimes, particularly in handling time-varying treatments and confounders. (15) discusses adaptive strategies in clinical trials which forms the basis for understanding sequential treatment adjustments. (18) discusses the use of deep RL for optimizing treatment regimes in healthcare, addressing challenges like high dimensional data and complex treatment sequences. (21) discusses the development of treatment rules that maximize patient outcomes, providing performance guarantees for the derived policies.

In order to accomplish the task of designing a model based interpretable greedy algorithm to compute interpretable policies, we design the Value Interpretable Dynamic Treatment Regimes (VIDTR) - a technique to compute the optimal interpretable policies when we have knowledge of the transition and the reward models for the Markov Decision Process. The VIDTR algorithm provides inbuilt interpretability by performing backwards induction on regularized versions of the Bellmann equations. Borrowing ideas from (33) we design a model based version of their technique, where we add the volume promotion and complexity penalisation errors as part of the regularization. This is added to the derived L^2 penalty obtained from the Bellman equations. We provide bounds in a simulated setting for the experiments and verify them in the grid based setup, where we perform sensitivity analysis with respect to the parameters of the experimental setup. We also check how our method compares with existing approaches from the literature.

Our contributions in the following paper can be summarized as follows:

1. **Model based interpretable policies for MDPs**: We extend the paper by (34) to derive a model-based version of interpretable dynamic treatment regimes where we compute a greedy model based interpretable policy at the different time steps.
2. **Implementation** : We design a data structure called DisjointBoxUnion in which we can perform geometric set operations on a disjoint union of boxes. Examples of this would include unions, intersections, subtractions, and integration of a function over this region. We also design a data structure called a constraint-condition, from which we can derive the constraints over which we perform set operations over.
3. **Extension of IDTR**: We extend the IDTR paper by (33) to the case where we use a DisjointBoxUnion to integrate functions over. This is added to the comparisons in the experimental setup.[Is this needed?] [How does this method work?]
4. **Theoretical bounds** : We compute theoretical bounds which assess the performance of the derived interpretable policy for the different time and length steps using stochastic approximation techniques for choosing the splits.
5. **Tight Value Interpretable Dynamic Treatment Regimes** : We design a tight version of the algorithm under assumptions of the transitions and rewards being close to each other in time and by taking a convex approximation of the objective.
6. **Convex Envelope Problem** : In order to find a tight approximation of our algorithm, we need to compute a convex approximation of the error functional. We do this by extending the work of (5) by constructing a convex d dimensional approximation to the Q-learning method. Further we develop the corresponding stochastic optimization theory to show error bounds for the method.
7. **Experimental setup** : We design experiments and setups where we test the final policy and review the performance of the derived interpretable policy. We compare our work against (9), (33) and an extension of the (33) work by using disjoint box unions to store regions and integrate functions over.

8. **Real world data** : We check the performance of the derived policy on real world data, where we have to designate treatments to patients at the different timesteps.

2 Literature Review

In this section we review the literature which consists of different ways to derive the optimal interpretable policy for a Markov Decision Process.

The first step to perform is to derive estimates of the transition and reward vectors for the offline reinforcement learning problem. (2) devise an active learning procedure to derive the model parameters for Markov chains and Markov decision processes. The paper presents a model based active learning sampling procedure that chooses examples which are most informative with respect to the current model hypothesis. This is based on active learning of MDPs using the Baum Welch algorithm. The authors of (31) propose two regularized maximum likelihood estimation algorithms for learning the transition probability model and policy. They also provide a sample complexity result that shows that a lower regret can be achieved with a relatively small number of training samples. (27) formalizes the problem of learning an accurate model for an MDP. Here we sample states from regions where we wish to maximize the entropy of the empirical state frequency.

After deriving the transitions and rewards, we move on to find interpretable policies. Here we present approaches in the literature when the same is done. The approach presented in (10) works based on reverse induction of the Bellman equations. After having obtained the transitions and the rewards for the different timesteps to compute the value functions. After deriving the value function, we fit a classification tree T_t which fits the value function at that timestep. Our approach computes the (approximate) solution to the forward RL problem and then performs reverse induction using the mean squared error with regularization on the complexity and volume promotion at that timestep. The approach presented in (8) introduces a fitted Q value iteration procedure where we similarly learn the Q-values by doing a backwards induction on the Bellman equations. This approach is a model free version of the one presented in (10). The authors of (26) introduce parameterized versions of decision trees over which we can differentiate on. Assuming this as the class of policies which we need to optimize over, they introduce a policy gradient method over this set of policies where we differentiate over the splitting parameters. This ensures that we maximize for the total discounted return over the whole space. In our approach we compute the best split in a greedy fashion at each time and lengthstep. In (29) we obtain optimal decision tree policies using an MDP formulation. Here we assume a linear programming formulation to compute the optimal policy for the MDP. To this formulation, we add additional constraints to optimize only over the class of all decision tree policies. Our method differs in the sense that we obtain a greedy solution which is arguably quicker but would yield a larger runtime.

We can derive the optimal interpretable policies directly from the trajectory data. Here we present approaches from the literature where this is done. Our work can be thought of as a model based version of the algorithm presented in (33), where the authors compute list based approximations to the final policy. They perform regression to obtain the estimates of the Q-values at the different timesteps, they pull back the Q-values using Bellman's equations for which they fit a decision tree at each timestep. We directly estimate the model parameters to which we perform reverse induction with the Bellman equations. In the paper by (12), the authors introduce the genetic programming for reinforcement learning approach based on model based reinforcement learning and genetic programming, which autonomously learns policy equations from pre-existing state-action trajectory samples. In (4) the authors construct a verifiable policy from the ones trained using a DNN as in the algorithm presented in (23). This involves coming up with an imitation learning procedure to the algorithm learnt by the DNN in (23).

(16) presents a parametric gradient based policy update where we have access to the self driving car environment to perform the updates. Here we compute the return for a parametric policy, where interpretability is defined based on the values taken by the parameters. For the tree based setup, one fixes a tree-based structure based on expert knowledge and in the RL algorithm we optimize

over the parameter space to tune the parameters. Our approach works based on greedy splits and the tree structure is also optimized for. In (28) we wrap a base MDP to form an IBMDP and solve using a modified RL algorithm. The solution is a decision tree policy for the base environment. In (11), the authors represent a decison tree where the leaves represent parametric policies and the various parameters represent the splits. With the total return over the MDP as the objective, we seek to optimize over the given parameters. (19) defines the notion of interpretability as given by simple closed form formulas in the policy space using which we rank actions. In this case, the search for the optimal policy turns out to be a multi-armed bandit problem where each arm represents a policy in the shortest expression to express it.

3 Problem Formulation

We aim to answer the following question, given knowledge of the modelling parameters, how does one come up with an optimal interpretable policy?

3.1 Notation

1. T : Time Horizon of the MDP
2. $(S_t)_{t=1}^T$: The state spaces for the different time steps for the MDP.
3. $\mathcal{A}$: Action space for the MDP
4. $P_t : S_{t+1} \times S_t \times \mathcal{A} \to [0,1], \forall t \in [0,T]$: The probability transition kernels for the different timesteps for the MDP.
5. $r_t : S_t \times \mathcal{A} \to \mathbb{R}, \forall t \in [0,T]$: The reward maps for the different timesteps in the MDP.
6. $\gamma \in (0,1)$: Discount factor for the Markov Decision Process
7. $(L_t)_{t=1}^T$: Maximum allowed depths for the different lengthsteps.
8. k: Maximum complexity of the final policy.
9. $(\rho_t)_{t=1}^T$: The complexity constants in the constraint condition for the different timesteps.
10. $(\eta_t)_{t=1}^T$: Volume split promotion constant in the optimization problem for the different timesteps.
11. $[(R_t^l, a_t^l)] \forall t \in [0,T]$ and $\forall l \in [0, l_t^*]$: The obtained interpretable policy at the different time and lengthsteps.

In this problem setting, we are given an MDP $((P_t)_{t=1}^T, (r_t)_{t=1}^T)$ over a finite horizon T and a discount factor γ. The algorithm proceeds by performing a backwards induction on the regularized Bellman equation. The parameters we tune in the algorithm are the maximum complexity k, maximum lengths per time $(L_t)_{t=1}^T$, η which is the volume promotion constant and ρ which is the complexity constant. The goal is to find interpretable polices $(\pi_t^I)_{t=1}^T$ such that an interpretable policy at the tth timestep is characterized by $[(R_t^l, a_t^l)_{l=1}^{L_t}]$ where R_t^l are the regions that are described by the conditions and a_t^l are the actions chosen at that length and timestep.

4 Algorithm Design

Since we perform reverse induction on the Bellman Equations, we start the procedure from the last timestep. At the last timestep, we perform the following analysis -:

4.1 Timestep T

At timestep T, we find a policy $\pi_T^I \in \mathcal{I}$ which approximates $\max_a r_T(s,a)$. At the first length-step, we perform

$$\pi_T^I(s) = \begin{cases} a_1 \text{ if } s \in R_1 \\ \arg\max_a r_T(s,a) \text{ otherwise} \end{cases}$$

Here $R_1 \in \mathcal{R}_1$ which given a maximum complexity k look like:

$$\{x : \exists i_1, i_2, ..., i_k s.t \pm x_{i_1} \leq c_1, ..., \pm x_{i_k} \leq c_k\}$$

Here we assume $i_1, i_2, ..., i_k \in \{1, 2, 3..., d\}$.

When compared to the optimal policy, we get $||V_{\pi^*} - V_{\pi_I}||$, which is given by:

$$\frac{1}{|S|}\int_S \max_a [r_T(s,a)]ds - \int_{R_1} r_T(s,a_1)ds - \int_{S-R_1} \max_a [r_T(s,a)]ds$$

We now wish to add regularization terms that penalize complexity and promote splitting, towards this goal we get:

This results in the following optimization problem:

$$\Psi_1^{T*} := \min_{R_1,a_1} \Psi(R_1,a_1) = \frac{1}{|S|}\int_{R_1} [\max_a [r_T(s,a)] - r_T(s,a_1)]ds - \eta_T V(R_1) + \rho_T c(R_1)$$

Here, since we try to compute $\min_{R\in\mathcal{R}}(-V(R))$, we maximize over $\eta V(R)$ at each lengthstep thereby promoting the volume splitting at each lengthstep. $c(R)$ is a parameter that would measure the complexity of the partitioning set. In the example of lists from the IDTR paper, we could measure the total number of parameters involved in the split. For example

$$c(\{x : \exists j_1, j_2 \text{ s.t } \pm x_{j_1} \leq c_1, \pm x_{j_2} \leq c_2\}) = 2 - I(c_1 = \pm\infty) - I(c_2 = \pm\infty)$$

In the next lengthstep we have:

$$\pi_I(s) = \begin{cases} a_1 \text{ if } s \in R_1 \\ a_2 \text{ if } s \in R_2 - R_1 \\ \arg\max_a r_T(s,a) \text{ otherwise} \end{cases}$$

We now assume $G_1 = R_1$ The optimization problem is given by:

$$\Psi_2^T := \min_{R_2,a_2} \Psi(R_2,a_2) = \frac{1}{|S|}\int_{R_2-G_1} [\max_a [r_T(s,a)] - r_T(s,a_1)]ds - \eta_T V(R_2 - G_1) + \rho_T c(R_2)$$

For the general lengthstep $l < l_T^*$ and timestep T, we have that the optimization problem is given by:

$$\Psi_l^{T*} := \min_{R_l,a_l} \Psi^T(R_l,a_l) = \int_{R_l-G_{l-1}} [\max_a (r_T(s,a)) - r_T(s,a_l)]ds - \eta_T V(R_l - G_{l-1}) + \rho_T c(R_l)$$

Then we have

$$G_l = G_{l-1} \cup R_l$$

When $l = l_T^*$ we have:

$$\pi_I(s) = a_{l_T^*} \text{ if } s \in S - G_{l_T - 1}$$

$$\min_{a_l} \frac{1}{|S|} \int_{S-G_{l^*_T-1}} [\max_a(r_T(s,a)) - r_T(s,a_l)]ds$$

To perform backward induction on the V-values we would need access to the optimal V values which we now compute using the Bellman's equation; we have

$$V_T(s) = \max_a r_T(s,a)$$

4.2 Timestep $t < T$

For lengthstep $l < l^*_t$, we have:

$$\pi_I(s) = \begin{cases} a_1 \text{ if } s \in R_1 \\ \arg_{a\in A} \max[r_t(s,a) + \gamma P^a_t V_{t+1}(s)] \text{ otherwise} \end{cases}$$

Here the optimizing equation is given by:

$$\Psi_t(R_l, a_l) := \int_{R_l - G_{l-1}} ([\max_a [r_t(s,a) + \gamma P^a_t V_{t+1}(s)]$$
$$-[r_t(s,a_l) + \gamma P^a_t V_{t+1}(s)]])ds - \eta_t V(R_l - G_{l-1}) + \rho_t c(R_l)$$

Here we have $\Psi^{t*}_l := \min_{R_l, a_l} \Psi_t(R_l, a_l)$
The optimization equation for $l = l_{max}$ is given by:

$$\Psi^{t*}_l := \min_{a_l} \Psi_t(a_l) = \int_{S-G_{l_{\max}-1}} [\max_{a_l \in A} [(r_t(s,a) - r_t(s,a_l)] + \gamma(P^a_t V^{t+1}(s) - P^{a_l}_t V^{t+1}_I(s)]$$

We also note the expressions for $V_t(s)$:

$$V_t(s) = \max_a [r_t(s,a) + \gamma P^a_t V_{t+1}(s)]$$

We have the following expression for $V^I_t(s)$:

$$V^I_t(s) = \begin{cases} r_t(s, a^t_1) + \gamma P^{a^t_1}_t V^I_{t+1}(s) \text{ if } s \in R^t_1 \\ r_t(s, a^t_2) + \gamma P^{a^t_2}_t V^I_{t+1}(s) \text{ if } s \in R^t_2 \\ ... \\ r_t(s, a^t_{l_t}) + \gamma P^{a^t_{l_t}}_t V^I_{t+1}(s) \text{ if } s \in R^t_{l_t} \end{cases}$$

When we are in the discrete setting, the integrals may be replaced by summations to yield an appropriate result.

4.3 Algorithm

In this section we describe the pseudocode for the implemented algorithms.

4.4 Implementation

We provide the code to implement this method. We note that to perform the integration step in the algorithm, we compute the regions over which we integrate over. Assuming we start out with a box to perform the initial integration, we notice that when we remove or add conditions, we are left either with a box or a disjoint union of boxes (DBU). We create a data structure under the same name and store this structure using the centres and the lengths of the same. To the data structure we ensure it has the following functionalities:

Algorithm 1 Value Interpretable Dynamic Treatment Regimes

Data: $S, A, T, \gamma, (P_t)_{t=1}^T, (r_t)_{t=1}^T, (l_t^*)_{t=1}^T, \eta, \rho, \kappa, st, M$
Output: $\pi_t^I, T\epsilon$ bounds = **bounds**(S) $R = S, T\epsilon = 0, \mathcal{C} =$ **conditions**(S)
while $t \in \{T-1, T-2, ..., 0\}$ **do**
while $l \in [1, l_t^* - 1]$ **do** $\epsilon_{min} = \infty, c^* = None, a^* = None$
$M = \min\{M, |\mathcal{C}|\}$
while $i \in [M]$ **do**
$\epsilon = 0$
$C[i] = i$th condition in $\mathcal{C}$
while $a \in A$ **do**
$bf(s) = \max_a(r_t(s,a) + \gamma P_t^a V_{t+1}(s))$ - $(r_t(s,a) + \gamma P_t V_{t+1}(s))$
$\epsilon = \int_{C[i]} (bf(s) + \eta_t) ds + \rho_t$ Complexity$(C[i])$
if $\epsilon < \epsilon_{min}$ **then**
$c^* = C[i], a^* = a, \epsilon_{min} = \epsilon$
$R = R - c^*, \mathcal{C} = \mathcal{C} - c^*$
$T\epsilon += \epsilon_{min}; \pi_t^I|_{c^* \cap R} := a^*$

Algorithm 2 Value Interpretable Dynamic Treatment Regime - Last lengthstep

while $t \in [T-1, T-2, ..., 0]$ **do**
At $l = l_t^*$
$\epsilon_{min} := \infty, a^* := None$
while $a \in A$ **do**
$bf(s) = \max_a(r_t(s,a) + \gamma P_t^a V_{t+1}(s))$ - $(r_t(s,a) + \gamma P_t V_{t+1}(s))$
$\epsilon = \int_R (bf(s) + \eta_t) ds$
if $\epsilon < \epsilon_{min}$ **then**
$a^* = a, \epsilon_{min} = \epsilon$
$T\epsilon += \epsilon_{min}; \pi_t^I|_R := a^*$

Algorithm 3 Optimal policy computation - Value Iteration

$S, A, T, \gamma, (P_t)_{t=1}^T, (r_t)_{t=1}^T (\pi_t)_{t=1}^T, (V_t)_{t=1}^T$ $\pi_t(s) \leftarrow a_0 \forall t \in [0,T], \forall s \in S$ $V_t(s) \leftarrow 0 \forall t \in [0,T], \forall s \in S$
while $t \in [0,T] \cap \mathcal{Z}$ **do**
while $s \in S$ **do** $\bar{v} = \inf$
while $a \in A$ **do** `b_val` $= r_t(s,a) + \gamma P_t^a V_{t+1}(s)$
if `b_val` $\geq \bar{v}$ **then** $\bar{v} =$ `b_val` $\bar{a} = a$ $\pi_t(s) = \bar{a}$ $V_t(s) = \bar{v}$

1. **Subtraction of DBUs**: This works by performing disjoint unions over subtracted rectangles. Given rectangles $A_1, B_1 \in \mathbb{R}^{d_1}$ and rectangles $A, B \in \mathbb{R}^{d_2}$. On a rectangle level the logic we employ is the following:

$$(A_1 \times A) - (B_1 \times B) := [(A_1 - B_1) \times A] \sqcup [(A_1 \cap B_1) \times (A - B)]$$

   We perform the above operation until we reach the dimension of 1, here we can manually compute the subtraction of two intervals.

2. **Union of DBUs**: This works by performing the following logic on the level of rectangles:

$$A \cup B := A \sqcup (B - A)$$

   The subtraction of two rectangles can be performed as above.

3. **Intersection of DBUs**: This works by performing a disjoint union over the set of all the rectangles that we take a union of. The logic on the level of rectangles in this scenario is given by:

$$(A \times A_1) \cap (B \times B_1) := (A \cap B) \times (A_1 \cap B_1)$$

4. **Integration of a function**: Given a function f, we perform the Riemannian integration of the function over the DBU. Given a $DBU := A_1 \sqcup A_2 ... \sqcup A_n$. This is given by:

$$\int_{DBU} f := \sum_{i=1}^{n} \int_{A_i} f$$

   The integral of f over a box is the discrete Riemannian sum of the function. We are given stepsizes $(s_d)_{d=1}^{D}$ and we sum up the function by spacing points over the grid using the given stepsizes. This is done by performing the sum of f with points spaced out by s_d in the corresponding dth dimension.

4.5 Empirical estimation of the rewards and transitions

We empirically estimate the transitions and the rewards for an MDP. In this algorithm, we compute the transition and reward probabilities as empirical estimates of what is observed in the trajectory data. The algorithm works by scanning over the trajectories while storing the empirical estimates of the transition probabilities and the rewards.

Algorithm 4 Empirical estimation of the transitions and the rewards

$(s_t^i)_{t=1}^T, (a_t^i)_{t=1}^T, (r_t^i)_{t=1}^T \forall i \in [1, N]$ $(P_t)_{t=1}^T, (r_t)_{t=1}^T$: The estimated transitions and rewards $T(s'|s, a) = 0 \forall s, s' \in S, a \in A$ $O(s, a) = 0 \forall s \in S, a \in A$ $T_r(s, a) = 0 \forall s, s' \in S, a \in A$ $P_{emp}(s'|s, a) = 0 \forall s, s' \in S, a \in A$ $r_{emp}(s, a) = 0 \forall s \in S, a \in A$

while $t \in [T - 1]$ **do**

  while $i \in [1, N]$ **do** $T((s_{t+1}^i)|(s_t^i), (a_t^i)) += 1$ $T_r((s_t^i), (a_t^i)) += r_t^i$ $O((s_t^i), (a_t^i)) += 1$

  while $t \in [T - 1]$ **do**

    while $i \in [1, N]$ **do** $P_{emp}((s^i)_{t+1}|(s^i)_t, (a^i)_t) = T((s^i)_{t+1}|(s^i)_t, (a^i)_t)/O((s^i)_t, (a^i)_t)$

$r_{emp}(s_t^i, a_t^i) = T_r(s_t^i, a_t^i)/O((s_t^i), (a_t^i))$

An issue with using the empirical estimation to compute the transitions and the rewards is that we would need to discretize the state space beforehand. Since the data we have from the IDTR paper is of continous data, we need to an additional step of preprocessing to get the algorithm to work. We do this by employing the following procedure - given a point in the DBU, we find a closest point in the

4.6 Ambiguity

In this section we introduce the notion of ambiguity for a Markov Decision Process. Here we define ambiguity as the weighted MEU - α sum of the rewards obtained in the final policy.

4.7 Relation with the different time and lengthsteps

For time-step t, we have that the maximum possible length is l_t^*. We assume that the optimization

5 Experimental Results

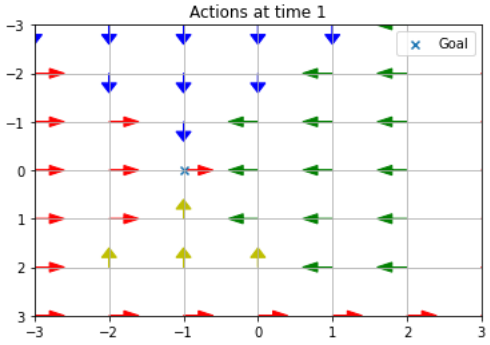
We observe how our method compares against existing methods in the literature for a 2-D grid based setup.

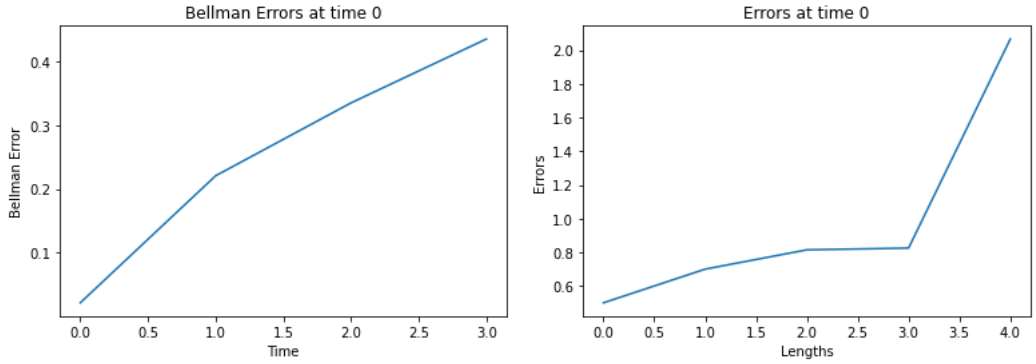
5.1 Grid Environment

We run experiments in a grid based environment where we have a goal at a certain location and the agent must choose one of the following actions : {up, down, left, right} in order to get to the goal. We describe the experimental setup as follows:

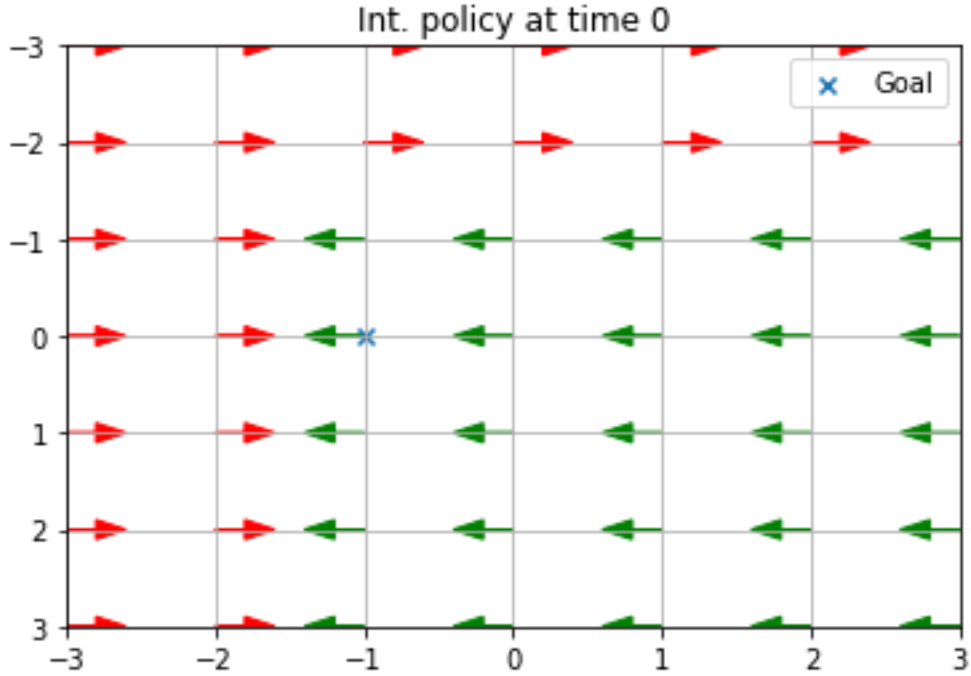
1. S: Points on the grid that have an appropriate spacing are given by points in $\mathbb{R}^2$.
2. A: The coordinate actions $\{(0,1),(1,0),(-1,0),(0,-1)\}$.
3. T: Time horizon for the MDP
4. g: The goal for the described setup, here it is at $(-1,0)$.
5. γ: Discount factor for the MDP
6. $r_t(s,a) := (||(s+a)-g||+1)^{-1} \forall t > 0$, these are the rewards
7. $P_t(s'|s,a) := \mathbf{I}(s+a=s')$ for $s,s' \in S, a \in A$. Here $\mathbf{I}$ is the indicator function.

In the setup below the goal is situated at the $(-1,0)$ coordinate. The optimal policy for the agent to choose in this setting is the direction which leads it to the goal.



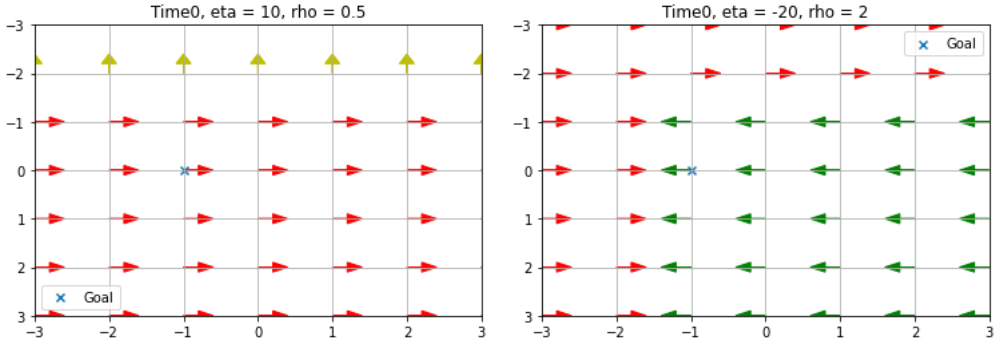


We insert plots of the Bellman errors obtained and the total errors above. The Bellman error is the approximation error obtained from the Bellman equations. The total error plots have the regularization terms added to the plots.



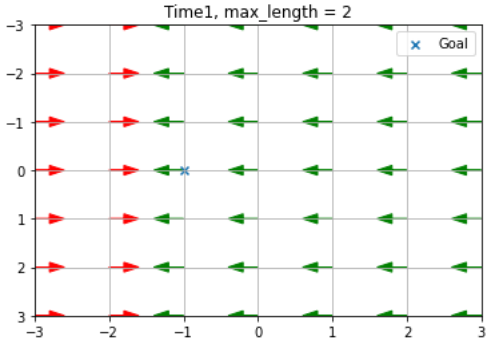
We also look at the plot of the optimal policy obtained above.

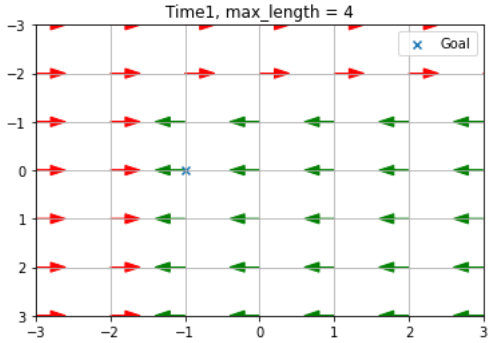
We also run experiments for different values of ρ and η to check the behaviour of the optimal policy under these values. We get the same plots for the values of η being $-20, -50$ and ρ being $0.5, 1, 2$



For the following values of $\eta, \rho := \{(-50, 0.5), (-50, 1), (-50, 2), (-20, 0.5), (-20, 1), (-20, 2)\}$ we get the plots as in the RHS.

For the following values of $\eta, \rho := \{(10, 0.5), (10, 2), (30, 0.5), (30, 1), (30, 2)\}$ we get the plot as in the LHS.



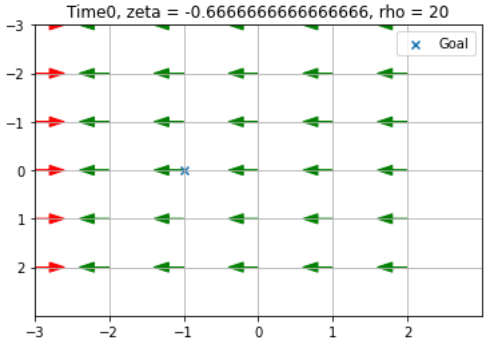


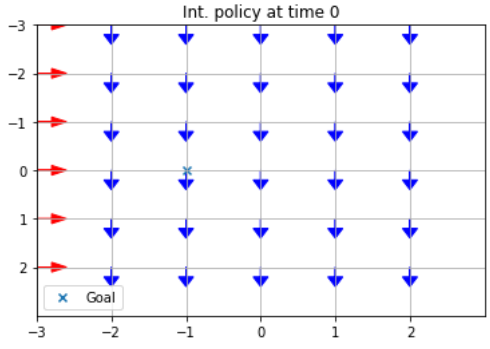
For maximum length from 2 to 3 we have the same plot as the one on the left, for lengths of 4 upto 7 and more we get the same plots as the one on the right.

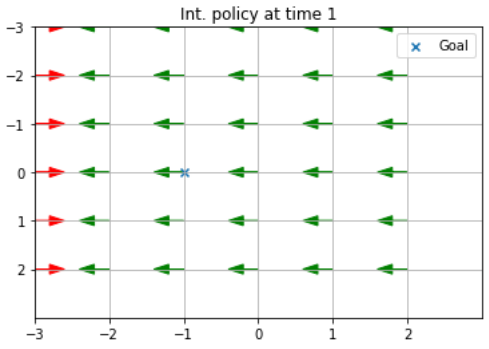
We run experiments to check the performance of the error with the different parameters of the MDP. The parameters we test include:

5.2 IDTR results

We get the following results when we run the DBU version of the IDTR method.







The split on the y - axis seems to be off by one mark on the y-axis.

5.3 IDTR data

We simulate the scenarios according to the Interpretable Dynamic Treatment Regimes paper by Zhang. Here we have the following scenarios:

5.3.1 Scenario 1

This consists of 2 timesteps. At the first timestep we have $S_1 = (S_{1,1}, S_{1,2}, ..., S_{1,50})^T$ are iid standard normal variables, $A_1 \sim Uni(\{-1, 1\})$, $Y_1 \sim \mathcal{N}(\mu_1, 1)$, where $\mu_1 = 0.5S_{1,3}A_1$. Here S_2 is empty, $A_2 \sim Uni(\{-1, 1\})$, $Y_2 \sim \mathcal{N}(\mu_2, 1)$, where $\mu_2 = \{(S_{1,1}^2 + S_{1,2}^2 - 0.2)(0.5 - S_{1,1}^2 - S_{1,2}^2) + Y_1\}A_2$.

5.3.2 Scenario 2

Here we have time varying covariates - At the first timestep we have $S_1 = (S_{1,1}, S_{1,2}, ..., S_{1,50})^T$ are iid standard normal variables, $A_1 \sim Uni(\{-1, 1\})$, $A_2 \sim Uni(\{-1, 1\})$, $Y_1 \sim N(\mu_1, 1)$, where $\mu_1 = (1 + 1.5S_{1,3})A_1$. $\mathcal{S}_2 = (S_{2,1}, S_{2,2})^T$, where $S_{2,1} \sim \text{Ber}(1 - \Phi(1.25S_{1,1}A_1))$; $S_{2,2} \sim \text{Ber}(1 - \Phi(-1.75S_{1,2}A_1))$; $Y_2 \sim \mathcal{N}(\mu_2, 1)$, here $\mu_2 = (0.5 + Y_1 + 0.5A_1 + 0.5S_{2,1} - 0.5S_{2,2})A_2$.

5.3.3 Scenario 3

Here we have 3 timesteps such that $A_1, A_2, A_3 \sim \text{Uniform}(\{-1, 1\}^3)$. $S_{1,1}, S_{1,2}, S_{1,3} \sim N(45, 15^2)$; $S_2 \sim N(1.5S_{1,1}, 10^2)$; $S_3 \sim N(0.5S_2, 10^2)$; $Y_1 = Y_2 = 0$

5.4 Real world data

We get the following results when we run the DBU version of the IDTR method.

The split on the y - axis seems to be off by one mark on the y-axis.

5.5 IDTR data

We simulate the scenarios according to the Interpretable Dynamic Treatment Regimes paper by Zhang. Here we have the following scenarios:

5.5.1 Scenario 1

This consists of 2 timesteps. At the first timestep we have $S_1 = (S_{1,1}, S_{1,2}, ..., S_{1,50})^T$ are iid standard normal variables, $A_1 \sim Uni(\{-1, 1\})$, $Y_1 \sim \mathcal{N}(\mu_1, 1)$, where $\mu_1 = 0.5S_{1,3}A_1$. Here S_2 is empty, $A_2 \sim Uni(\{-1, 1\})$, $Y_2 \sim \mathcal{N}(\mu_2, 1)$, where $\mu_2 = \{(S_{1,1}^2 + S_{1,2}^2 - 0.2)(0.5 - S_{1,1}^2 - S_{1,2}^2) + Y_1\}A_2$.

To be completed

5.5.2 Scenario 2

Here we have time varying covariates - At the first timestep we have $S_1 = (S_{1,1}, S_{1,2}, ..., S_{1,50})^T$ are iid standard normal variables, $A_1 \sim Uni(\{-1, 1\})$, $A_2 \sim Uni(\{-1, 1\})$, $Y_1 \sim N(\mu_1, 1)$, where $\mu_1 = (1 + 1.5S_{1,3})A_1$. $\mathcal{S}_2 = (S_{2,1}, S_{2,2})^T$, where $S_{2,1} \sim \text{Ber}(1 - \Phi(1.25S_{1,1}A_1))$; $S_{2,2} \sim \text{Ber}(1 - \Phi(-1.75S_{1,2}A_1))$; $Y_2 \sim \mathcal{N}(\mu_2, 1)$, here $\mu_2 = (0.5 + Y_1 + 0.5A_1 + 0.5S_{2,1} - 0.5S_{2,2})A_2$.

To be completed

5.5.3 Scenario 3

Here we have 3 timesteps such that $A_1, A_2, A_3 \sim \text{Uniform}(\{-1, 1\}^3)$. $S_{1,1}, S_{1,2}, S_{1,3} \sim N(45, 15^2)$; $S_2 \sim N(1.5S_{1,1}, 10^2)$; $S_3 \sim N(0.5S_2, 10^2)$; $Y_1 = Y_2 = 0$

To be completed

5.6 Real world data

To be completed

6 Theoretical Results

In this section we establish the theoretical results that establish the convergence of the value interpretable dynamic treatment regimes method.

6.1 Bounds on the errors

Assumption 1 *We list the assumptions we use for proving the error bounds*

1. *Lipschitz continuity of rewards in the second coordinate:* $|r_T(s,b) - r_T(s,a)| \leq C_r d_A(a,b)$
2. *Distance between actions is bounded:* $d_A(a,b) \leq D$
3. *Smoothness in the kernels:* $|P_t(s'|s,a) - P_t(s'|s,b)| \leq C_p d_A(a,b)$
4. *Bounds on the value functions:* $V_{t+1}(s) \leq W_{t+1}$
5. $S = \prod_{i=1}^{N}(b_i, a_i)$, *here we assume* $|b_i - a_i| \leq S^{1/d}$

For each t, we measure the cost of intepretablity to be the sum of the quantities that we aim to minimize, which is the adjusted value metric. Here for different timesteps t it is given by:

$$C_t = \sum_{l=1}^{l_{max}} \Psi^*_{tl}$$

So the final cost we obtain is given by:

$$C = \sum_{t=1}^{T} C_t$$

Recall that for time T and lengthstep 1, we have:

$$\Psi^*_{T1} = \min_{R,a_1} \int_R [\max_a(r_T(s,a)) - r_T(s,a_1)]ds - \eta_T V(R) + \rho_T c(R_1)$$

We have the following to be true:

$$|r_T(s,a^*(s)) - r_T(s,a_1)| \leq C_r d_A(a^*(s), a_1)$$

Assuming the rewards are Lipschitz and that the action space is bounded we get the below inequality:

$$\Psi^*_{T1} \leq \min_{R_1}[|R|(C_r D - \eta) + \rho_T c(R_1)]$$

We have that $G_l = \cup_{i=1}^{l} R_i$ this would imply for a general l, we get the following equation:

$$\Psi^*_{Tl} \leq \min_{R_l}(|R_l - G_l|(C_r D - \eta) + \rho c(R_l))$$

For $l = l_{max}$, we get:

$$\Psi^*_{Tl_{max}} \leq C_r D|S - G_{l_{max}-1}|$$

The total error would be the sum of ([6.1] + [6.1] (for different lengths l) + [6.1]).

For a general $t = t$ and $l = 1$, we have that the error is given by:

$$\Psi^*_{t1} \leq \int_R [\max_a (r_t(s,a) + \gamma P^a_t V_{t+1}(s)) - (r_t(s,a) + \gamma P^a_t V_{t+1}(s))] ds - \eta_t V(R) + \rho_t c(R)$$

If $a^*(s) := \max_a (r_t(s,a) + \gamma P^a_t V_{t+1}(s))$, then the following is true:

$$\Psi^*_{t1} \leq \int_R [r_t(s,a^*(s)) + \gamma P^{a^*(s)}_t V^*_t(s) - r_t(s,a) - \gamma P^a_t V_{t+1}(s)] ds - \eta_t V(R) + \rho_t c(R)$$

We can then split the rewards term as:

$$[r_t(s,a^*(s)) - r_t(s,a)] \leq C_r D d_A(a^*, a)$$

We then see that we can bound the kernel term using a triangle inequality, this works as:

$$(P^{a^*(s)}_t V^*_{t+1}(s) - P^a_t V_{t+1}(s)) \leq (P^{a^*(s)}_t V^*_{t+1}(s) - P^a_t V^*_{t+1}(s) + P^a_t V^*_{t+1}(s) - P^a_t V_{t+1}(s))$$

The above then simplifies as follows:

$$|P^{a^*(s)}_t - P^a_t| V^*_{t+1}(s) \leq C_P d_A(a^*(s), a) W_{t+1}$$

We can simplify the other term:

$$|P^a_t V^*_{t+1}(s) - P^a_t V_{t+1}(s)| \leq |P^a_t| |V^*_{t+1}(s) - V_{t+1}(s)|$$

We can the bound the kernel as $|P^a_t| \leq 1$. This gives us the below inequality:

$$\Psi^*_{t1} \leq \min_{R_1} (C_r D |R_1| + \gamma C_p D W_{t+1} |R_1| + \int_{R_1} |V_{t+1}(s') - V^I_{t+1}(s')| ds' - \eta |R_1| + \rho_t c(R_1))$$

We can simplify the computation of $\int_{R_1} [V_{t+1}(s') - V^I_{t+1}(s')] ds'$ for different values of t. We notice that for $t = T$, we have $V^*_T(s) := \max_a (r_T(s,a))$, and that $V^I_T(s) := r_T(s, a_i)$ for $s \in R_i$. This implies that

$$|V^*_T(s) - V^I_T(s)| \leq \max_a (r_T(s,a)) - r_T(s, a_i))$$

As done previously the above can be bounded by the Lipschitz constant on the rewards and the distance between the actions. When the above gets integrated over a region R, we see:

$$\int_R |V^*_T(s) - V^I_T(s)| \leq C_r D |R|$$

We now simplify the calculation of $\int_R |V^*_t(s) - V^I_t(s)| ds$ in terms of the future timesteps. This gives us the following:

$$|(V^*_t - V^I_t)(s)| \leq \max_a (r_t(s,a) + \gamma P^a_t V_{t+1}(s)) + \gamma (P^a_t V^*_{t+1}(s) - P^{a_i}_t V^I_{t+1}(s))$$

The rewards term can now be bouded by $C_r D$ and we see that the kernel terms split as:

$$|(V^*_t - V^I_t)(s)| \leq C_r D + \gamma (P^a_t V^*_{t+1}(s) - P^{a_i}_t V^I_{t+1}(s))$$

We add and subtract $P^{a_i}_t V^*_{t+1}$ to the above transition terms to get:

$$\leq C_r D + \gamma (P^a_t V^*_{t+1} - P^{a_i}_t V^*_{t+1} + P^{a_i}_t V^*_{t+1} - P^{a_i}_t V^I_{t+1})$$

We can simplify the above as follows:

$$\leq C_r D + \gamma[(P_t^a - P_t^{a_i})V_{t+1}^* + P_t^{a_i}(V_{t+1}^* - V_{t+1}^I)]$$

The above is then simplified using the fact that the norm of the transitions are bounded by 1.

$$\leq C_r D + 2\gamma||V_{t+1}^*|| + \gamma||V_{t+1}^* - V_{t+1}^I||$$

If we assume that $d_t := ||V_t^*(s) - V_{t+1}^I(s)||$, we then get the following recursive inequality:

$$d_t \leq C_r D + 2\gamma||V_{t+1}^*|| + \gamma d_{t+1}$$

Before solving the above recursively, we establish bounds on $||V_t^*||$.

$$V_t(s) = \max_a(r_t(s,a) + \gamma P_t^a V_{t+1}(s))$$

We can simplify the above as follows:

$$||V_t(s)|| \leq ||r_t|| + \gamma||V_{t+1}(s)||$$

We also have that $||V_T(s)|| \leq \bar{r}$. Then we can see that the following is true:

$$||V_{T-1}(s)|| \leq \bar{r} + \gamma\bar{r}$$

We make the claim that :

$$||V_t(s)|| \leq \bar{r}[1 + \gamma + \gamma^2 + ... + \gamma^{T-t}]$$

This gives us:

$$||V_t(s)|| \leq \frac{1 - \gamma^{T-t+1}}{1-\gamma}\bar{r}$$

Returning back to the simplification of d_t, we see that for $t = T$:

$$|V_T^* - V_T^I|(s) \leq C_r D$$

We also have $||V_T(s)|| \leq \bar{r}$.
For $t = T - 1$, we have the following:

$$||V_{T-1} - V_{T-1}^I||(s) \leq C_r D(1+\gamma) + 2\gamma\bar{r}$$

For $t = T - 2$, we have:

$$d_{T-2} \leq (C_r D + 2\gamma\bar{r})(1 + \gamma + \gamma^2) + 2\gamma^2\bar{r}(1+\gamma)$$

We then make the following lemma, which we show inductively:

$$d_{T-t} \leq (C_r D + 2\gamma\bar{r})(1 + \gamma + ... + \gamma^t) + 2\gamma^2\bar{r}(1 + 2\gamma + 3\gamma^2 + ... + (t-1)\gamma^{t-2} + (t-1)\gamma^{t-1})$$

We show that the above holds by inserting the recursion equation 6.1. This gives us:

$$\begin{aligned} d_{T-t-1} \leq C_r D + 2\gamma\bar{r}\frac{1-\gamma^{t+2}}{1-\gamma} + \\ +(C_r D + 2\gamma\bar{r})[\gamma + \gamma^2 + ... + \gamma^{t+1}] + \\ 2\gamma^3\bar{r}(1 + 2\gamma + 3\gamma^2 + ... + (t-1)\gamma^{t-2} + (t-1)\gamma^{t-1}) \end{aligned}$$

The above is then simplified as follows:

$$d_{T-t-1} \leq (C_r D + 2\gamma\bar{r})[1 + \gamma + ... + \gamma^{t+1}] + 2\gamma\bar{r}(\gamma + \gamma^2 + ... + \gamma^{t+1}) + 2\gamma^3\bar{r}(1 + 2\gamma + 3\gamma^2 + ... + (t-1)\gamma^{t-2} + (t-1)\gamma^{t-1})$$

By taking γ inside the above is then simplified as:

$$d_{T-t-1} \leq (C_r D + 2\gamma\bar{r})[1 + \gamma + ... + \gamma^{t+1}] + 2\gamma^2\bar{r}[1 + \gamma + \gamma^2 + ... + \gamma^t] + 2\gamma^2\bar{r}(\gamma + 2\gamma^2 + ... + (t-1)\gamma^{t-1} + (t-1)\gamma^t)$$

We can simplify the above as follows:

$$d_{T-t-1} \leq (C_r D + 2\gamma\bar{r})[1 + \gamma + ... + \gamma^{t+1}] + 2\gamma^2\bar{r}(1 + 2\gamma + 3\gamma^2 + ... + t\gamma^{t-1} + t\gamma^t)$$

We now list how we simplify $(1 + 2\gamma + 3\gamma^2 + ... + t\gamma^{t-1} + t\gamma^t)$. We define the following:

$$S_{t-1} := 1 + 2\gamma + 3\gamma^2 + ... + (t-1)\gamma^{t-2}$$

We then have that:

$$\gamma S_{t-1} := \gamma + 2\gamma^2 + 3\gamma^3 + ... + (t-1)\gamma^{t-1}$$

We then get that 6.1 - 6.1 is given by:

$$S_{t-1}(1-\gamma) = 1 + \gamma + ... + \gamma^{t-2} - (t-1)\gamma^{t-1}$$

This then simplifies as:

$$S_{t-1} = \frac{1 - \gamma^{t-1}}{(1-\gamma)^2} - \frac{(t-1)\gamma^{t-1}(1-\gamma)}{(1-\gamma)^2}$$

The above can then be simplified as:

$$S_{t-1} := \frac{1 - \gamma^{t-1}(-t\gamma^{t-1} + \gamma^{t-1})(1-\gamma)}{(1-\gamma)^2}$$

We can then simplify the above as:

$$S_{t-1} = \frac{1 - \gamma^{t-1} - t\gamma^{t-1} + \gamma^{t-1} - t\gamma^t - \gamma^t}{(1-\gamma)^2}$$

This is then simplified as:

$$S_{t-1} = \frac{1 - \gamma^{t-1}[\gamma + t(1+\gamma)]}{(1-\gamma)^2}$$

The above can be simplified as:

$$d_{T-t-1} \leq (C_r D + 2\gamma\bar{r})\frac{1 - \gamma^{t+2}}{1-\gamma} + 2\gamma^2\bar{r}[\frac{1 - \gamma^{t-1}[\gamma + t(1+\gamma)]}{(1-\gamma)^2} - (t-1)\gamma^{t-1}]$$

We recall the following bounds for W_t from above:

$$W_t \leq \frac{1-\gamma^{T-t+1}}{1-\gamma}\bar{r}$$

Recall how the errors were computed below:

$$\Psi^*_{t1} \leq \min_{R_1}(C_r D|R_1| + \gamma C_p D W_{t+1}|R_1| + \int_{R_1} |V_{t+1}(s') - V^I_{t+1}(s')|ds' - \eta|R_1| + \rho_t c(R_1))$$

In the above equation, we insert the values of W_t and bounding $|V_{t+1}(s) - V^I_{t+1}(s)|$, we see that:

$$\begin{aligned}\Psi^*_{t1} \leq \min_{R_1}(C_r D|R_1| + \gamma C_p D\frac{1-\gamma^{T-t}}{1-\gamma}\bar{r}|R_1| + \\ (C_r D + 2\gamma\bar{r})\frac{1-\gamma^{T-t}}{1-\gamma}|R_1| + \\ 2\gamma^2\bar{r}\frac{1-\gamma^{T-t-3}[\gamma^3 - 1(T-t-2)(1+\gamma^2(1+\gamma))]}{(1-\gamma)^2}|R_1|) + \\ -\eta|R_1| + \rho_t c(R_1)\end{aligned}$$

For different time-steps t and length-steps l_t, we see that the optimizing function Ψ^*_{tl}. We assume that $G_{t,l-1} = \cup_{j=1}^{l-1} R_{tj}$. This implies:

$$\begin{aligned}\Psi^*_{tl} \leq \min_{R_l}(C_r D|R_l - G_{t,l-1}| + \gamma C_p D\frac{1-\gamma^{T-t}}{1-\gamma}\bar{r}|R_l - G_{t,l-1}| + \\ (C_r D + 2\gamma\bar{r})\frac{1-\gamma^{T-t}}{1-\gamma}|R_l - G_{t,l-1}| + \\ 2\gamma^2\bar{r}\frac{1-\gamma^{T-t-3}[\gamma^3 - 1(T-t-2)(1+\gamma^2(1+\gamma))]}{(1-\gamma)^2}|R_l - G_{t,l-1}|) + \\ -\eta|R_l - G_{t,l-1}| + \rho_t c(R_l)\end{aligned}$$

Lemma 1 *Let $d_1, d_2, ..., d_K$ be numbers such that we wish to find the indices i, j such that $\prod_{j=1}^K d_1 d_2 ...(d_i - 1)...(d_j - 1)...d_K$ is minimized for all $i, j \in [K]$. The indices given by the least 2 elements in the set $d_1, d_2, ..., d_K$, i.e. $d^{(1)}, d^{(2)}$, where $d^{(1)} \leq d^{(2)} \leq ... \leq d^{(K)}$ are the order statistics of the elements.*

We see that the product is given by $\prod_{k=1}^K d_1 d_2 ..(d_i - 1)...(d_j - 1)...d_K$ for all $k \in [K]$. We can multiply and divide by d_i and d_j to get:

$$S = (\prod_{k=1}^K d_k)\frac{d_i - 1}{d_i}\frac{d_j - 1}{d_j}$$

We notice the behavior of the function $\frac{d-1}{d}$, this is monotonically increasing in d. This can be seen by taking the derivative which is $1/d^2$. This implies that the function 6.1 is minimized when $d_i = d^{(1)}, d_j = d^{(2)}$.

Theorem 1 *The optimization function below where we have $S = \Pi_{i=1}^D [a_i, b_i]$ and choose R from one of the following list sets with k leaves. The sets are:*

$$\begin{aligned}R = \{S, \emptyset, \{x|\exists j \in [N],\ s.t\ x_j \leq c_j\}, \\ \{x|\exists j, k\ s.t\ x_j \leq c_j, x_k \leq c_k\}, ..., \{x|\exists j_1, .., j_k \in [N],\ s.t\ x_{j_i} \leq c_{j_i} \forall i \in [K]\}\}\end{aligned}$$

If $\rho > 0$, we observe the following:

$$min_R[|R|A + \rho c(R)]$$

The problem [2] is upper bounded by:

1. $A > 0 \implies$ *We choose $R = \emptyset$, which gives us a 0 valued objective.*
2. $A < 0 \implies$ *We choose $R = S$, which then gives us $c(R) = 0$ and we see that the objective is bounded by $A|S|$.*

Theorem 2 *The optimization function below where we have $S = \Pi_{i=1}^{D}[a_i, b_i]$ and choose R from one of the following list sets with k leaves. The sets are:*

$$R = \{S, \emptyset, \{x|\exists j \in [N],\ s.t\ x_j \leq c_j\},$$
$$\{x|\exists j,k\ s.t\ x_j \leq c_j, x_k \leq c_k\}, ..., \{x|\exists j_1, .., j_k \in [N],\ s.t\ x_{j_i} \leq c_{j_i} \forall i \in [K]\}\}$$

If $\rho < 0$, we observe the following:

$$min_R[|R|A - \rho c(R)]$$

The problem [2] is upper bounded by:

1. $A > 0 \implies$ *We choose R to be a point, i.e $R = c_1 \times c_2 \times ....c_D$ or $R = \emptyset$, which would imply the objective is* $\min(A - \rho k, 0)$.
2. $A < 0 \implies$ *We choose R from $\{S, \Pi_{i=1}^{k}(\{c_{j_i}^{(1)} \leq x_{j_i} \leq c_{j_i}^{(2)} for\ j_i \in [K])\}\}$, this implies that the objectives are given by $Max\{A|S|, A\Pi_{i=1}^{k}(c_{j_i}^{(2)} - c_{j_i}^{(1)})\Pi_{i \neq [j_1,..,j_k]}(b_i - a_i) + k\rho \forall k \in [N], c_{j_i}^{(2)} \leq b_{j_i}, c_{j_i}^{(1)} \geq a_i\}$.*

Theorem 3 *The error in the last timestep is given by:*

$$O(S^{1+1/d})(C_r D - \eta) + O(S^{d/3})C_r/l^* D$$

We can see that the $O(S^{d/3})$ term goes to 0 as l^* becomes large. We would be left with only the errors from the non-Lipschitzness of r.

Theorem 4 *The error in the other timesteps is given by:*

$$O(S^{1+1/d})(2C_r D - \eta) + O(S^{d/3})2C_r/l^* D$$

We can see that the $O(S^{d/3})$ term goes to 0 as l^* becomes large. We would be left with only the errors from the non-Lipschitzness of r.

6.2 Ambiguity

6.3 Algorithm improvement

6.4 Algorithm

In this section, we describe the pseudo-code for the algorithms implemented.

In the step where the maximum computation is performed, improvement is required. Currently, we calculate the objective for each condition R and each action a, this requires us to iterate over each and every condition R. We find an approximation as follows : assume that we know the optimal action a_t^* and condition R_t^* at time t. Can we find R_{t+1}^* and a_{t+1}^* from this information?

Algorithm 5 Value Interpretable Dynamic Treatment Regimes

Data: $S, A, T, \gamma, (P_t)_{t=1}^{T}, (r_t)_{t=1}^{T}, (l_t^*)_{t=1}^{T}, \eta, \rho, \kappa, st, M$
Output: $\pi_t^I, T\epsilon$ bounds = **bounds**(S) $R = S, T\epsilon = 0, \mathcal{C} =$ **conditions**(S)
while $t \in \{T-1, T-2, ..., 0\}$ **do**
  while $l \in [1, l_t^* - 1]$ **do** $\epsilon_{min} = \infty, c^* = None, a^* = None$ $M = \min\{M, |\mathcal{C}|\}$
    while $i \in [M]$ **do** $\epsilon = 0$ $C[i] = i$th condition in $\mathcal{C}$
      while $a \in A$ **do** $bf(s) = \max_a(r_t(s,a) + \gamma P_t^a V_{t+1}(s))$ - $(r_t(s,a) + \gamma P_t V_{t+1}(s))$ $\epsilon = \int_{C[i]} (bf(s) + \eta_t) ds + \rho_t$ Complexity$(C[i])$
        if $\epsilon < \epsilon_{min}$ **then** $c^* = C[i], a^* = a, \epsilon_{min} = \epsilon$ $R = R - c^*, \mathcal{C} = \mathcal{C} - c^*$ $T\epsilon += \epsilon_{min}; \pi_t^I|_{c^* \cap R} := a^*$

7 Acknowledgements

References

[1] S. Alowais, S. Alghamdi, N. Alsuhebany, T. Alqahtani, A. Alshaya, S. Almohareb, A. Aldairem, M. Alrashed, K. Saleh, H. Badreldin, M. Al Yami, S. Al Harbi, and A. Albekairy, *Revolutionizing healthcare: the role of artificial intelligence in clinical practice*, BMC Medical Education, 23 (2023).

[2] G. Bacci, A. Ingólfsdóttir, K. Larsen, and R. Reynouard, *Active learning of markov decision processes using baum-welch algorithm (extended)*, arXiv preprint arXiv:2110.03014, (2021).

[3] J. Bajwa, U. Munir, A. Nori, and B. Williams, *Artificial intelligence in healthcare: transforming the practice of medicine*, Future Healthcare Journal, 8 (2021), pp. e188–e194.

[4] O. Bastani, Y. Pu, and A. Solar-Lezama, *Verifiable reinforcement learning via policy extraction*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Red Hook, NY, USA, 2018, Curran Associates Inc., p. 2499–2509.

[5] V. S. Borkar and A. Akarsh, *Approximation of convex envelope using reinforcement learning*, 2023.

[6] B. Chakraborty and S. A. Murphy, *Dynamic treatment regimes*, Annu. Rev. Stat. Appl., 1 (2014), pp. 447–464.

[7] T. H. Davenport and R. Kalakota, *The potential for artificial intelligence in healthcare*, Future Healthcare Journal, 6 (2019), pp. 94 – 98.

[8] D. Ernst, P. Geurts, and L. Wehenkel, *Tree-based batch mode reinforcement learning*, Journal of Machine Learning Research, 6 (2005).

[9] J. Grand-Clément, C. Chan, V. Goyal, and E. Chuang, *Interpretable machine learning for resource allocation with application to ventilator triage*, arXiv preprint arXiv:2110.10994, (2021).

[10] ——, *Interpretable machine learning for resource allocation with application to ventilator triage*, arXiv preprint arXiv:2110.10994, (2021).

[11] U. D. Gupta, E. Talvitie, and M. Bowling, *Policy tree: adaptive representation for policy gradient*, in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, AAAI Press, 2015, p. 2547–2553.

[12] D. Hein, S. Udluft, and T. A. Runkler, *Interpretable policies for reinforcement learning by genetic programming*, Engineering Applications of Artificial Intelligence, 76 (2018), pp. 158–169.

[13] Y. Huang, R. Cao, and A. Rahmani, *Reinforcement learning for sepsis treatment: A continuous action space solution*, in Proceedings of the 7th Machine Learning for Healthcare Conference, Z. Lipton, R. Ranganath, M. Sendak, M. Sjoding, and S. Yeung, eds., vol. 182 of Proceedings of Machine Learning Research, PMLR, 05–06 Aug 2022, pp. 631–647.

[14] N. N. Khanna, M. A. Maindarkar, V. Viswanathan, J. F. E. Fernandes, S. Paul, M. Bhagawati, P. Ahluwalia, Z. Ruzsa, A. Sharma, R. Kolluri, I. M. Singh, J. R. Laird, M. Fatemi, A. Alizad, L. Saba, V. Agarwal, A. Sharma, J. S. Teji, M. Al-Maini, V. Rathore, S. Naidu, K. Liblik, A. M. Johri, M. Turk, L. Mohanty, D. W. Sobel, M. Miner, K. Viskovic, G. Tsoulfas, A. D. Protogerou, G. D. Kitas, M. M. Fouda, S. Chaturvedi, M. K. Kalra, and J. S. Suri, *Economics of artificial intelligence in healthcare: Diagnosis vs. treatment*, Healthcare (Basel), 10 (2022), p. 2493.

[15] P. W. Lavori, R. Dawson, and D. Shera, *A multiple imputation strategy for clinical trials with truncation of patient data*, Stat. Med., 14 (1995), pp. 1913–1925.

[16] A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano, *Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving*, Robotics Auton. Syst., 131 (2020), p. 103568.

[17] Y. Liu, B. Logan, N. Liu, Z. Xu, J. Tang, and Y. Wang, *Deep reinforcement learning for dynamic treatment regimes on medical registry data*, in 2017 IEEE International Conference on Healthcare Informatics (ICHI), 2017, pp. 380–385.

[18] ——, *Deep reinforcement learning for dynamic treatment regimes on medical registry data*, in 2017 IEEE international conference on healthcare informatics (ICHI), IEEE, 2017, pp. 380–385.

[19] F. Maes, R. Fonteneau, L. Wehenkel, and D. Ernst, *Policy search in a space of simple closed-form formulas: Towards interpretability of reinforcement learning*, in Discovery Science, J.-G. Ganascia, P. Lenca, and J.-M. Petit, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 37–51.

[20] A. Orfanoudaki, S. Saghafian, K. Song, H. A. Chakkera, and C. Cook, *Algorithm, human, or the centaur: How to enhance clinical care?*, (2022).

[21] M. Qian and S. Murphy, *Performance guarantees for individualized treatment rules*, Annals of statistics, 39 (2011), pp. 1180–1210.

[22] J. M. Robins, *Optimal structural nested models for optimal sequential decisions*, in Proceedings of the Second Seattle Symposium in Biostatistics: analysis of correlated data, Springer, 2004, pp. 189–326.

[23] S. Ross, G. Gordon, and D. Bagnell, *A reduction of imitation learning and structured prediction to no-regret online learning*, in Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[24] S. Saghafian and L. Idan, *Effective generative ai: The human-algorithm centaur*, arXiv preprint arXiv:2406.10942, (2024).

[25] ——, *Effective generative ai: The human-algorithm centaur*, arXiv preprint arXiv:2406.10942, (2024).

[26] A. Silva, T. Killian, I. D. J. Rodriguez, S.-H. Son, and M. Gombolay, *Optimization methods for interpretable differentiable decision trees in reinforcement learning*, arXiv preprint arXiv:1903.09338, (2019).

[27] J. Tarbouriech, S. Shekhar, M. Pirotta, M. Ghavamzadeh, and A. Lazaric, *Active model estimation in markov decision processes*, in Conference on Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 1019–1028.

[28] N. Topin, S. Milani, F. Fang, and M. Veloso, *Iterative bounding mdps: Learning interpretable policies via non-interpretable methods*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 9923–9931.

[29] D. Vos and S. Verwer, *Optimal decision tree policies for markov decision processes*, arXiv preprint arXiv:2301.13185, (2023).

[30] X. Wu, R. Li, Z. He, T. Yu, and C. Cheng, *A value-based deep reinforcement learning model with human expertise in optimal treatment of sepsis*, npj Digital Medicine, 6 (2023).

[31] T. Xu, H. Zhu, and I. C. Paschalidis, *Learning parametric policies and transition probability models of markov decision processes from data*, European Journal of Control, 57 (2021), pp. 68–75.

[32] C. Yu, J. Liu, S. Nemati, and G. Yin, *Reinforcement learning in healthcare: A survey*, ACM Computing Surveys (CSUR), 55 (2021), pp. 1–36.

[33] Y. Zhang, E. B. Laber, M. Davidian, and A. A. Tsiatis, *Interpretable dynamic treatment regimes*, Journal of the American Statistical Association, 113 (2018), pp. 1541–1549.

[34] ——, *Interpretable dynamic treatment regimes*, J. Am. Stat. Assoc., 113 (2018), pp. 1541–1549.