Robust Decision-Focused Learning for Reward Transfer

Abhishek Sharma[1], Sonali Parbhoo[2], Omer Gottesman[3], and Finale Doshi-Velez[1]

[1]SEAS, Harvard University, Cambridge
[2]Imperial College London
[3]AWS

April 10, 2023

Abstract

Decision-focused (DF) model-based reinforcement learning has recently been introduced as a powerful algorithm which can focus on learning the MDP dynamics which are most relevant for obtaining high rewards. While this approach increases the performance of agents by focusing the learning towards optimizing for the reward directly, it does so by learning less accurate dynamics (from a MLE standpoint), and may thus be brittle to changes in the reward function. In this work, we develop the robust decision-focused (RDF) algorithm which leverages the non-identifiability of DF solutions to learn models which maximize expected returns while simultaneously learning models which are robust to changes in the reward function. We demonstrate on a variety of toy example and healthcare simulators that RDF significantly increases the robustness of DF to changes in the reward function, without decreasing the overall return the agent obtains.

1 Introduction

In reinforcement learning (RL), model-based approaches focus on learning a transition model of the environment and using this model for planning decisions. Using simple model classes can be necessary for interpretability and data efficiency reasons. It has recently been demonstrated that, since the model's limited representational capacity precludes modeling *everything* about the dynamics, optimizing the model parameters with respect to the *eventual return* of the planned policy may perform better than just minimizing prediction error (Grimm et al., 2020). We refer to this type of model learning as *decision-focused (DF) learning* (Wilder et al., 2019; Sharma et al., 2021) of the model. By construction, DF learning enables learning the MDP dynamics which are only relevant for obtaining high rewards.

The focus of the learned DF model on returns, while desirable, can be thought of as "over-fitting" the model to the reward function. Thus, it can be expected to perform poorly for other reward functions. Changing reward functions are common in real world applications such as healthcare, where clinicians can shift their preferences over how much weight to give different measures of a patient's health (Lizotte et al., 2010), or industry, where costs of different resources may change over time. In this paper we demonstrate that due to this "over-fitting" to a specific reward function, *a DF model is sensitive to changes in the reward function, resulting in non-robust solutions.* Prior work by Futoma et al. (2020) introduced a log-likelihood objective to constrain the maximum likelihood objective to focus on information that is important for a decision. However, this form of regularization may still suffer from the original MLE learning problem of potentially focusing on parts of the transition dynamics that are irrelevant to planning.

To overcome these challenges, in this paper we introduce a Robust Decision-Focused (RDF) model that is robust to changes in the reward function when the transition dynamics of the environment stay the same. Conceptually, we do this by optimizing a model with respect to a set of possible reward functions, characterized by a parameter, w, rather than a single reward function (Wang et al., 2021; Nikishin et al., 2022). We develop a novel algorithm which allows us to perform such optimization efficiently using a sampling based approach.

Our contributions are as follows: First, we demonstrate that a DF model may produce solutions that are not robust to changes in the choice of reward. Next, we introduce the RDF model which allows us to learn models which are robust over a wide range of reward functions. Finally, we empirically demonstrate on a suite of synthetic toy and simulated healthcare domains that, in addition to retaining all the advantages of standard DF learning, our RDF model provides solutions that are robust to different reward preferences. It can utilize all of the innovations in DF learning: it can achieve superior performance to MLE on restricted model classes, and can be optimized efficiently using gradient-based optimization.

2 Related Work

Model-based and Decision-Focused Learning Several works have looked at decision-focused model-based reinforcement learning (Joseph et al., 2013; Farahmand et al., 2017; Wang et al., 2021; Nikishin et al., 2022; Futoma et al., 2020). Grimm et al. (2020) and Nikishin et al. (2022) note that decision-focused models can be non-identifiable for a given reward function. Both suggest this can be a good thing. In contrast, we demonstrate that this non-identifiability can be problematic when the model must be re-used for perturbations in reward.

Multi-objective RL The general idea of multi-objective RL (MORL) is to train agents to perform well on multiple reward functions. The most common setting involves two reward functions, which are linearly combined using a weight called *preference* Abels et al. (2019); Mossalam et al. (2016); Barrett & Narayanan (2008); Yamaguchi et al. (2019); Lizotte et al. (2010). (We refer the reader to Hayes et al. (2022) for a comprehensive survey and overview of the literature.) In this work, we assume a similar setting. In context of the transition dynamics model, while most approaches focus on a model-free setting, there are few exceptions Wiering et al. (2014); Wan et al. (2021); Yamaguchi et al. (2019). These model-based MORL algorithms learn the model using either maximum likelihood or Bayesian inference. Importantly, they ignore the eventual return on the true environment to inform model learning. In contrast, we focus on learning the dynamics model in a decision-focused manner.

Transfer Learning Work by Barreto et al. (2020) introduces the idea of performing transfer learning in situations where only the reward function differs. The authors introduce the idea of using Successor Features to capture a representation of the policy by decoupling its dynamics from expected rewards. This is extended in Reinke & Alameda-Pineda (2021) where the authors relax some of the assumptions that rewards may be decomposed linearly into successor features for knowledge transfer. Unlike both of these, our work makes no assumptions about the form of the reward function though the idea of using successor features to express the reward function is complementary and could also be incorporated into our approach.

Reward Shaping There may be conditions under which modifying the reward function may preserve the optimal policy. Reward Shaping (RS) (Ng et al., 1999) is a technique that leverages exterior knowledge via a potential shaping function (Ng et al., 1999; Wiewiora et al., 2003) to guide an agent's policy learning. Unlike in RS, for Robust DF learning, each choice of reward preference will have a *different optimal policy.*

3 Preliminaries

Markov Decision Process We consider a Markov Decision Process (MDP). An MDP M is characterized by $< S, A, R, T, \gamma >$, defined as follows. S: State space of size N. A: Action space. $R(s, a)$: Reward, as a function of state s and action a. $T(s, a)$: Transition function, mapping each state-action pair to a probability distribution over successor states. γ: Discount factor, $0 \leq \gamma \leq 1$. The optimal policy π^* maximises the expected return $\mathcal{J}_T(\pi^*(\theta, R))$ for model parameters θ and reward R.

Model-based RL Given the parameters θ of the transition model, we can use a planning algorithm to learn an optimal policy $\pi^*(\theta)$. This policy can then be used to run on the true environment to a achieve a return $\mathcal{J}_{T_*}(\pi^*(\theta, R))$.

$$\theta \rightarrow \pi^*(\theta, R) \rightarrow \mathcal{J}_{T_*}(\pi^*(\theta, R)) \tag{1}$$

Model-based RL traditionally estimates the model parameters θ using a dataset of transitions $\{(s_n, a_n, s'_n)\}_{n=1}^N$. This is typically done by maximum-likelihood estimation (MLE), which is equivalent to minimizing the KL divergence between the transition model and the true dynamics:

$$\theta_{\text{MLE}} \leftarrow \arg\min_\theta \mathbb{KL}\left(T_* || T_\theta\right) \tag{2}$$

While model-based RL can provide several statistical advantages, e.g. sample-efficiency, it is often limited in its ability to represent precise details of the dynamics. Since the MLE objective is not tied to the decision-making task, it can fail to find optimal policies when the model class is limited. Importantly, it is leaving money on the table—by using its limited representation capacity on modeling irrelevant information rather than on signal important for the task.

Decision-focused RL Decision-focused RL instead directly optimizes the following objective:

$$\theta_{\text{DF}} \leftarrow \arg\max_\theta \mathcal{J}_{T_*}(\pi^*(\theta, R)) \tag{3}$$

That is, DF learning directly focuses on yielding a good policy for the true environment. It learns a transition model that outputs this policy. In settings where the model class of T_θ cannot represent T_*, the decision-focused model can outperform the maximum-likelihood model (Joseph et al., 2013; Farahmand et al., 2017). Furthermore, the value equivalence principle by Grimm et al. (2020) states that there can be several model parameters θ with the same optimal policy (and return on the true environment).

Algorithm 1 Robust Decision-focused RL

```
input Initial model parameters $\theta$, reward preference $w_{train}$, reward preferences $\mathcal{W}$, $P(w)$, reward basis functions $r_1$, $r_0$
  Initialize Q-function parameters $\phi_{w_t}$, $\{\phi_w : w \in \mathcal{W}\}$
  repeat
    Using $T_\theta$ and $R_{w_t}$, compute $Q^*_{\phi_{w_t}}$
    for $w \in \mathcal{W}$, in parallel do
      Compute $R_w$ using Eqn 5
      Using $T_\theta$ and $R_w$, Compute $Q^*_{\phi_w}$
    end for
    Update $\theta$ using Eqn 8
  until the end condition
```

4 Method: Robust Decision-Focused RL for Varying Reward Preferences

Our goal is to learn a DF model that is *robust to changes in the reward preferences at test time.*

Reward preferences We begin by assuming a reward function that linearly interpolates between K basis functions r_1 to r_k according to,

$$R_w(s,a) = \sum_{k=1}^{K} w_k r_i(s,a); \quad \sum_k w_k = 1 \tag{4}$$

where w denotes a preference for particular basis, and the r_k's are assumed as given. To simplify exposition, we proceed with two basis functions, r_1 and r_0 with the reward being (we discuss extension to K bases at the end of the section)

$$R_w(s,a) = w r_1(s,a) + (1-w) r_0(s,a). \tag{5}$$

In practice, these are competing reward functions that the practitioner considers important, e.g. drug efficacy and drug side-effects.

Problems with the DF Objective DF learning assumes that the reward preference w is fixed and stays the same for both train and test time. However, when the reward preference w_{train} differs from that at test time (w_{test}), the DF model θ_{DF} may not yield a good policy anymore. Instead, we learn θ which can transfer to the new reward, while remaining performant under the original reward. Moreover, DF learning is known to suffer from non-identifiability: there may be multiple candidate solutions and several local optima as noted in Nikishin et al. (2022). In the single-reward case, this is not a problem, and perhaps beneficial: finding any of the local optima is good enough. But in the setting of multiple rewards, members of this non-identifiable set may have differing abilities with respect to transfer to non-DF rewards.

The Robust Decision-Focused objective we present next however, is also applicable, without loss of generality, to reward functions with more than two basis functions. Our key assumption is that we observe a reward preference w_{train} at train time, which might change at test time. Since we are learning a model, we can re-plan a new policy at test time when we are given a new reward preference. One scenario where this setting is useful is that of precision healthcare: consider again drug efficacy and side-effect. Learning a different model for each patient might not make sense because the underlying dynamics could transfer between them. We can train on a preference w_{train} that the clinician surmises would be a reasonable estimate, but accepts that each patient might have a different preference not exactly the same as w_{train}.

A Robust DF Objective To overcome the deficiencies (i.e. non-identifiability and non-transferability) of DF learning, we re-formulate the DF objective in Eqn 3 as a robust DF (RDF) objective. Unlike DF learning, the RDF objective selects the model that maximises the expected return of the optimal policy over model parameters θ *while being cognisant of possible change in reward preference at test time*:

$$\begin{aligned} &\max_{\theta} \mathop{\mathbb{E}}_{w\sim P(w)} \left[\mathcal{J}_{T_*,R_w}(\pi^*(\theta, R_w))\right] \\ &\text{s.t. } \mathcal{J}_{T_*,R_{w_t}}(\pi^*(\theta_{DF}, R_{w_t})) - \mathcal{J}_{T_*,R_{w_t}}(\pi^*(\theta, R_{w_t})) < \delta \end{aligned} \tag{6}$$

Here, w_t is a shorthand for w_{train}. $P(w)$ encodes the region of test w for which we want our model to be robust to. $\mathcal{J}_{T_*,R_w}(\pi)$ is the return on the true MDP (transition dynamics T_*) and reward function R_w. Finally, $\pi^*(\theta, R_w)$ explicitly states the dependence on the dynamics model θ and the reward function R_w used to learn the optimal policy.

Intuitively, Eqn 6 states that a robust DF model should maximise the expected return of a policy under transition dynamics T_* *for any preference of reward in the region specified by* $P(w)$, whilst yielding a similar return as optimal policy $\pi^*(\theta_{DF}, w_t)$ on the train setting.

Selecting $P(w)$ The distribution $P(w)$ encodes our information about the reward preferences we might encounter at test-time. While a uniform distribution around w_{train} may be a natural choice, we are free to choose a $P(w)$ that is non-uniform, or discontinuous, or even one that excludes w_{train}. We explore some of these choices in the experiments section. We show that RDF is useful when the test w is a perturbation around w_{train} but also when we care about a region far from w_{train}.

Computing the Robustness Objective Since optimising the constrained form of RDF objective in Eqn 6 can be challenging, we rewrite it using a Lagrange multiplier $\lambda > 0$:

$$\mathcal{J}_{RDF}(\theta, \lambda) = \mathop{\mathbb{E}}_{w\sim P(w)} \left[\mathcal{J}_{T_*,R_w}(\pi^*(\theta, R_w))\right] + \lambda \mathcal{J}_{T_*,R_{w_t}}(\pi^*(\theta, R_{w_t})) \tag{7}$$

The limit $\lambda \to \infty$ recovers DF learning, while setting $\lambda = 0$ ignores performance on the train-time reward and focuses on overall robustness with respect to P.

We use a uniform grid of points in the support of P to approximate the expectation in Eqn 7, corresponding to the trapezoid method of approximating the integral.

Selecting λ Although our optimization uses λ as a hyperparameter, a practitioner would find it easier to specify the δ parameter in the constraint-style formulation of Eqn 6. δ has an inverse relationship with λ where $\delta = 0$ would correspond to $\lambda = 0$. A practitioner may prioritize being close to DF performance (low δ/high λ) or being robust to different w's (high δ/low λ). We explore the $\delta - \lambda$ relationship in our experiments.

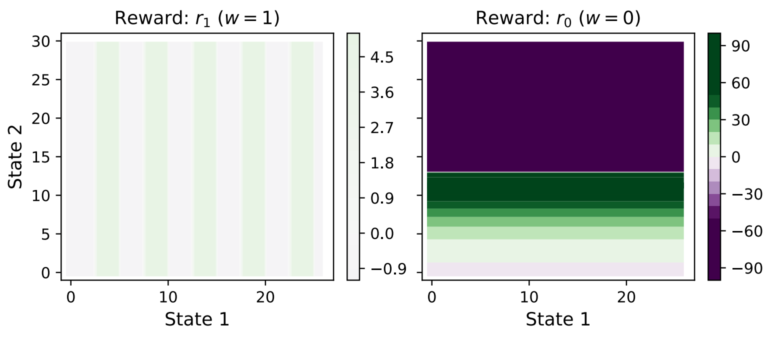

Figure 1: Reward functions for the Synthetic domain. Each reward function only depends on one of the states when $w = 0$ and $w = 1$ respectively

Optimizing the Robust DF Objective Taking gradients of the RDF objective gives

$$\nabla_\theta \mathcal{J}_{RDF}(\theta, \lambda) = \mathop{\mathbb{E}}_{w \in P(w)} \left[\nabla_\theta \mathcal{J}_{T_*, R(w)}(\pi^*(\theta))\right] + \lambda \nabla_\theta \mathcal{J}_{T_*, R(w_t)}(\pi^*(\theta)) \tag{8}$$

where we drop the dependence of π^* on R_w for brevity.

We employ implicit differentiation through the policy learning step and policy gradients to compute the gradients with respect to each w:

$$\nabla_\theta \mathcal{J}_{T_*}(\pi^*(\theta)) = -\underbrace{\nabla_{\pi^*} \mathcal{J}_{T_*}(\pi^*)}_{\text{Policy Grad}} \underbrace{\left[\nabla^2_\pi \mathcal{J}_{T_\theta}(\pi^*(\theta))\right]^{-1} \nabla^2_{\pi,\theta} \mathcal{J}_{T_\theta}(\pi^*(\theta))}_{\text{Implicit Grad of} \pi^* \text{ w.r.t } \theta} \tag{9}$$

where we have dropped dependence of $R(w)$ in all terms for clarity. Since each of these gradients can be computed parallely, the run-time cost is not any worse than a DF method.

Based on Eqn 8, we propose a new algorithm to optimise this objective. The algorithm assumes access to a parameterization of the Q-function ϕ, model parameters θ, a reward preference at train time w_{train}, the robustness distribution $P(w)$, and reward basis functions r_1, r_0.

Approximating the expectation We again approximate the expectation in Eqn 8 using the trapezoid method. Specifically, we construct $\mathcal{W}$, a set of uniformly-spaced w values in the support of P and use it to approximate the expectation:

$$\mathop{\mathbb{E}}_{w \sim P(w)} \left[\mathcal{J}_{T_*, R_w}(\pi^*(\theta, R_w))\right] = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} \mathcal{J}_{T_*, R_w}(\pi^*(\theta, R_w)) \tag{10}$$

In principle, RDF requires computing a separate Q-function per w in the grid, which might suggest that runtime of the algorithm increases linearly with $|\mathcal{W}|$. However, we can exploit the fact that each of the Q-functions can be computed independently. We leverage parallelization to learn the Q-function, which makes the time complexity of RDF learning to be the same as that of DF learning. If storing a separate Q-function for each w is expensive, we recommend sampling w from a uniform grid $\mathcal{W}$ used to approximate $P(w)$.

Choosing of policy planner Depending on the model parameterization, we can use an appropriate model-based learning. We note that choosing specific simple model classes enable fast planning, e.g. Value Iteration with discretized states and LQR controller for linear dynamics. We demonstrate the use of both in our experiments. Each approach has its limitations though—while discretization does not scale to higher dimensional-states, the LQR-based planning worked best for short-horizon problems (< 50) for us.

Generalization to Multiple Reward Bases Our formulation presented a reward function that is a linear interpolation of two reward basis functions mainly for simplicity of exposition. In principle, the approach is limited to neither linearity nor two rewards. Extending to non-linear preferences is straightforward: just replace

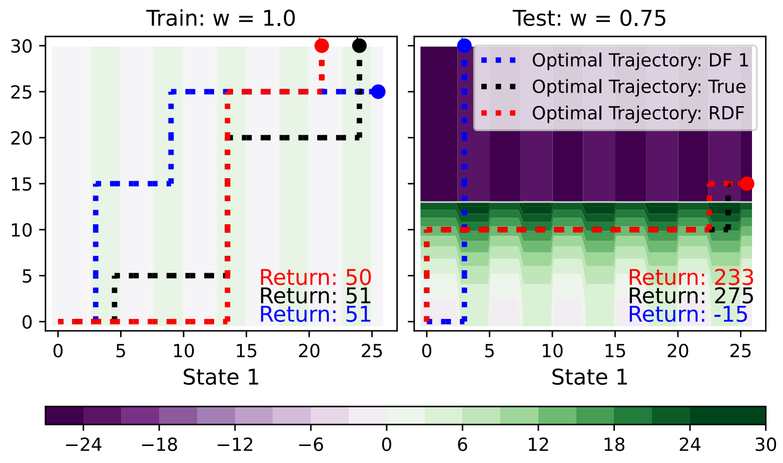

Figure 2: (Left) RDF and DF policies on the training reward. At train time, the DF policy (blue) and the optimal (true) policy (black) achieve the highest return. The RDF policy (red) performs suboptimally on the train reward. (Right) RDF and DF policies on the test reward. At test time, the RDF policy (red) significantly outperforms the DF policy (blue), while achieving near optimal performance.

Solution	$\mathcal{J}(w_{train})$±std	$\mathcal{J}$(avg)±std	$\mathcal{J}$ (w_{train})±std	$\mathcal{J}$(avg)±std
w_{train}	0.25	0.25	0.75	0.75
$w \in$	[0.0, 0.5]	[0.0, 0.5]	[0.5, 1.0]	[0.5, 1.0]
DF	**-6.4±1.0**	-6.9±0.9	**12.4±0.2**	8.0±0.9
MLE	-55.4±0.2	-50.6±0.1	-12.2±0.6	-11.7±0.2
RDF	**-5.4±0.7**	**-5.2±0.7**	11.0±0.2	**10.1±0.3**

Table 1: Results on the Cancer Domain: DF achieves optimal performance on the training reward when $w \in [0.5, 1.0]$, RDF is however robust to varying reward preferences at test time and outperforms both DF and MLE in terms of cumulative returns.

Eqn 5 with a different version without any change in time- or space-complexity. Extending to K rewards basis functions is also similar except that w would lie on a $(K-1)$-dimensional probability simplex with an appropriate $P(w)$ (Eqn 4). However, approximating this integral would require much more samples as K increases.

5 Experimental Setup

We provide an empirical demonstration of the advantages of our RDF approach in comparison to a DF model on a synthetic toy domain, mountain car and a cancer simulator. The aim of our experiments on each of these domains is to show that while a DF model focuses on modelling those dynamics relevant for higher rewards, it is not robust to changing reward preferences at test time, resulting in poorer performance under model misspecification. We can overcome this and other issues with our RDF model.

Baselines Our main focus is to compare the performance of our RDF approach to the DF approach under the setting where the true dynamics can not be completely represented by the chosen model class. We also compare to the maximum likelihood (MLE) solution for reference.

Metrics Across all of our experiments, we compare the performance of RDF to each of the baselines by computing the return of the models on the train reward, denoted as $\mathcal{J}(w_{train})$, and the average return on the test region, denoted as $\mathcal{J}$(avg). We report the means and standard deviations of the quantities. To compare the distribution of return over all w values in the synthetic environment, we also compute the “percentile of return”

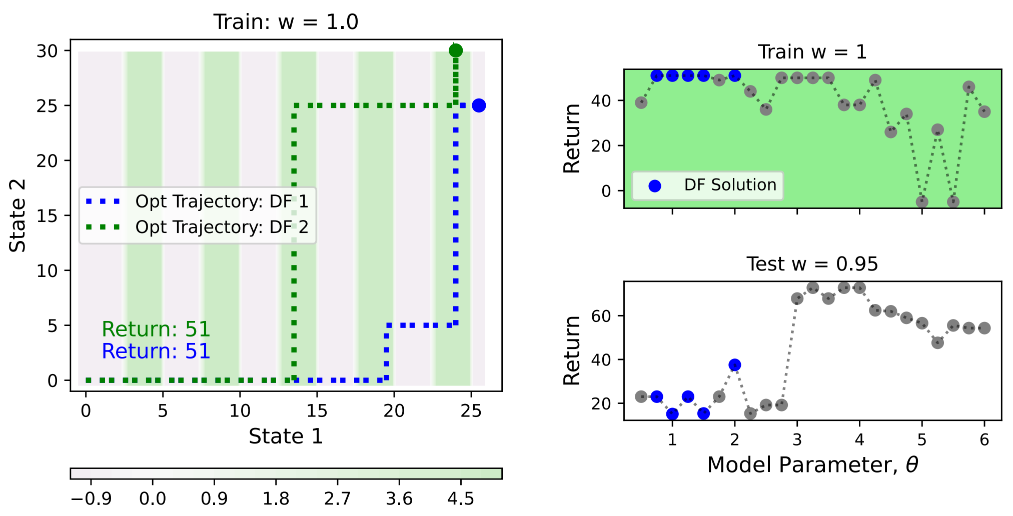

Figure 3: Not all DF models transfer equally well. Top: Optimal trajectories of two DF solutions (θ_1 = DF 1 and θ_2 = DF 2) visualized. Both solutions have a return of 51 on the train reward. Bottom: The two panels show the return on true environment for two reward preferences. The green panel corresponds to w_{train} which is also what the DF objective optimizes against. All DF solutions are marked with blue markers. At test time (white panel), these points have different values (θ_1 has a return of around 15 in the white panel (test) and θ_2 has a return of around 40), i.e. *the non-identifiable set of DF models do not transfer equally well* to test reward.

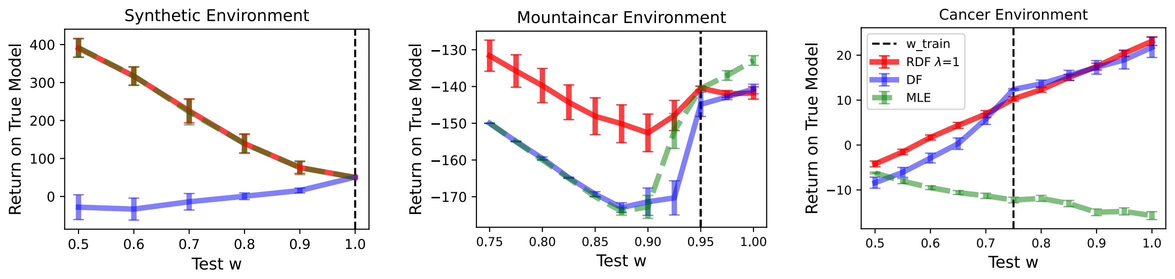

Figure 4: RDF approach outperforms DF and MLE across a variety of domains. While DF has comparable performance on w_{train}, its performance drops as we move away from it. While the MLE does well on the synthetic environment, there are no guarantees: it fails fully on the Cancer simulator. Note that the x-axis is different for different plots: we are plotting test w's that we wish to be robust to.

Solution $((\theta_1, \theta_2))$	$\mathcal{J}(w_{train})$		$\mathcal{J}$ (avg)	
$w \in$	[.5, 1]	[.75, 1]	[.5, 1]	[.75, 1]
DF $(1.5, 1.5)$	**51.0**	**51.0**	-11.0	5.1
MLE $(3.25, 3.25)$	50.0	50.0	224.8	125.1
RDF $(3.5, 3.5)$	50.0	50.0	**225.2**	**126.6**
True $(1.5, 5)$	51.0	51.0	269.9	154.9

Table 2: Results on the Synthetic domain: Though DF achieves optimal performance on the training reward, RDF is robust to varying reward preferences at test time and thus outperforms both DF and MLE in terms of cumulative returns (although MLE does come close). We do not report standard deviations because they are practically zero.

Solution	$\mathcal{J}(w_{train})$±std	$\mathcal{J}$(avg)±std
w_{train}	0.95	0.95
$w \in$	[0.75, 1.0]	[0.75, 1.0]
DF	-144.8±3.3	-158.3±0.9
MLE	**-140.5±0.6**	-155.4±0.5
RDF	**-140.6±0.5**	**-144.0±3.0**

Table 3: Results on the Mountain Car Domain: MLE achieves optimal performance when $w \in [0.75, 1.0]$, RDF is however robust to varying reward preferences at test time and outperforms both DF and MLE in terms of cumulative returns.

computed as follows. For each w, we run all the methods (DF, RDF, MLE) for multiple trials. These returns are ranked according to percentile of return range they belong to. Finally, we aggregate all the percentile scores for each method and obtain the histogram in Figure 5. For reference, the optimal model should achieve the 100th percentile return for each w, and thus appear as aspike over 100 in the figure.

5.1 Synthetic Toy

Consider an MDP with continuous states in $\mathbf{R}^2_+$ and actions in $\{[0,1]^\top, [1,0]^\top\}$. The agent starts at state $[0,0]$ and the episode ends if any of the states crosses 25.1. Assume the true dynamics of the model are given by,

$$\mathbf{s}' \leftarrow \mathbf{s} + \theta^* \odot \mathbf{a}, \tag{11}$$

where $\odot$ denotes element-wise multiplication of two vectors. We set $\theta^* = [1.5, 5]^\top$ and restrict the model class to $\theta = [1, 1]^\top$. The first reward basis function r_1 (corresponding to $w = 1$) only depends on the first state variable and alternates between -1 and 5: it is square wave which changes sign every 2.5 distance. The second reward basis function r_0 is only a function of s_2:

$$r_0(s_2) = \begin{cases} 200\left(\frac{s_2}{13}\right)^2 - 1, & s_2 \leq 13 \\ -201, & \text{otherwise.} \end{cases}$$

The reward function is a linear combination of r_1 and r_0 as specified by Eqn 5. Fig 1 shows the reward functions for $w = 1$ and $w = 0$ respectively.

Training details Given a model parameter θ and reward preference w, we use Fitted Q-Iteration to learn the optimal policy. We search over $\lambda \in \{0, 1, 10\}$ and choose the model with the highest return. We set train reward

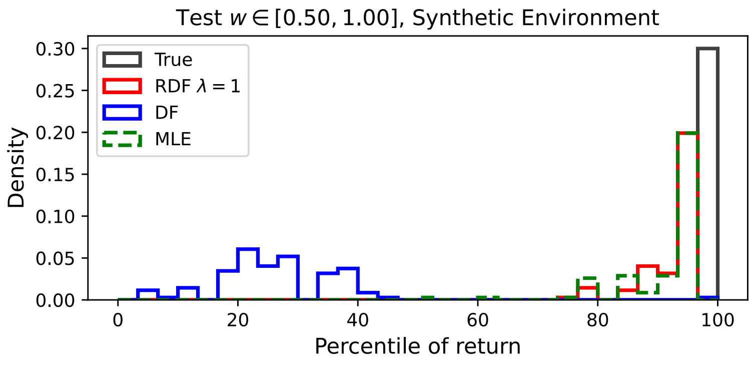

Figure 5: RDF model achieves more probability mass over higher returns in the synthetic environment. As expected, the true model always gets the highest returns.

preference w_{train} as 1 and then test the reward in $[0.5, 1]$ and $[0.75, 1]$. We report the results in Table 2. By construction, the DF model only maximizes the expected return with respect to s_1, and we would not expect it to be robust to changes in w at test time.

5.2 Cancer MDP

We also test the performance of our RDF model using a Cancer environment as described in Yauney & Shah (2018). The environment consists of 5-dimensional continuous state variables based on the mean size of a patient's tumor, drug concentration, and time-step to ensure the Markovian assumption is met, as well as the recent trajectory of a patient's tumor dimensions. At each time step, there are two possible choices of action corresponding to the dosage of drug administered. Overall, a reduction in the mean tumor size and possible side-effects is incentivized through a reward. That is, the reward is comprised of two basis functions: one that promotes a reduction in overall tumor size, and another that penalizes possible side-effects from using high concentrations of drugs. Specifically,

$$r_1 = \begin{cases} \frac{(MTD_t - MTD_{t'})}{10}, & \text{if } t' \neq T. \\ \frac{(MTD_t - MTD_{t'})}{10} + (MTD_0 - MTD_T), & \text{otherwise.} \end{cases}$$
$$r_0 = -concentration \tag{12}$$

where MTD_t denotes the mean tumor dimensions at a particular time point t.

Training details The transition dynamics are modeled using a linear model and our discount factor is 1. We train on reward preferences of $w = 0.25$ and $w = 0.75$ respectively and set the test w region to $[0, 0.5]$ and $[0.5, 1]$ respectively.

These results are shown in Table 1. We also demonstrate the effect of the λ parameter and experiment with other distributions for $P(w)$. These results are presented in Figs 7. We can also see how the choice of δ hyperparameter in the original formulation 6 translates into a choice of λ in Fig 6.

5.3 Mountain Car

In the basic mountain car problem, an underpowered car is positioned in a valley between two mountains on a one-dimensional track. The aim of the problem is to drive the car to the top of the mountain on the right-hand side, but the engine power available is insufficient to simply accelerate and power through to the top. A player must build momentum by going back and forth between the two mountains until the car has sufficient momentum to reach its goal. The states are the current position and velocity, and there are three actions: accelerate forward, accelerate backward and zero throttle. The single objective version has a -1 penalty for each time step the agent spends before reaching the goal (r_1). Vamplew et al. (2011) propose a multi-objective version of mountain car problem where another penalty of -0.1 is given each time the car accelerates (r_0). The goal is to motivate the car to reach the goal by applying least throttle as possible.

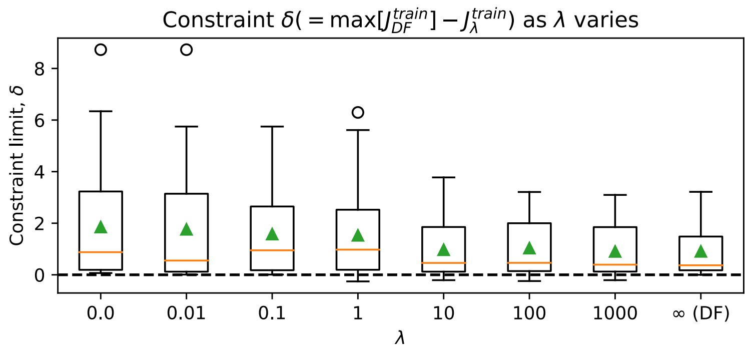

Figure 6: (Cancer domain) How a practioner would select λ: decide the δ in Eqn 6 and select the smallest λ that achieves this δ.

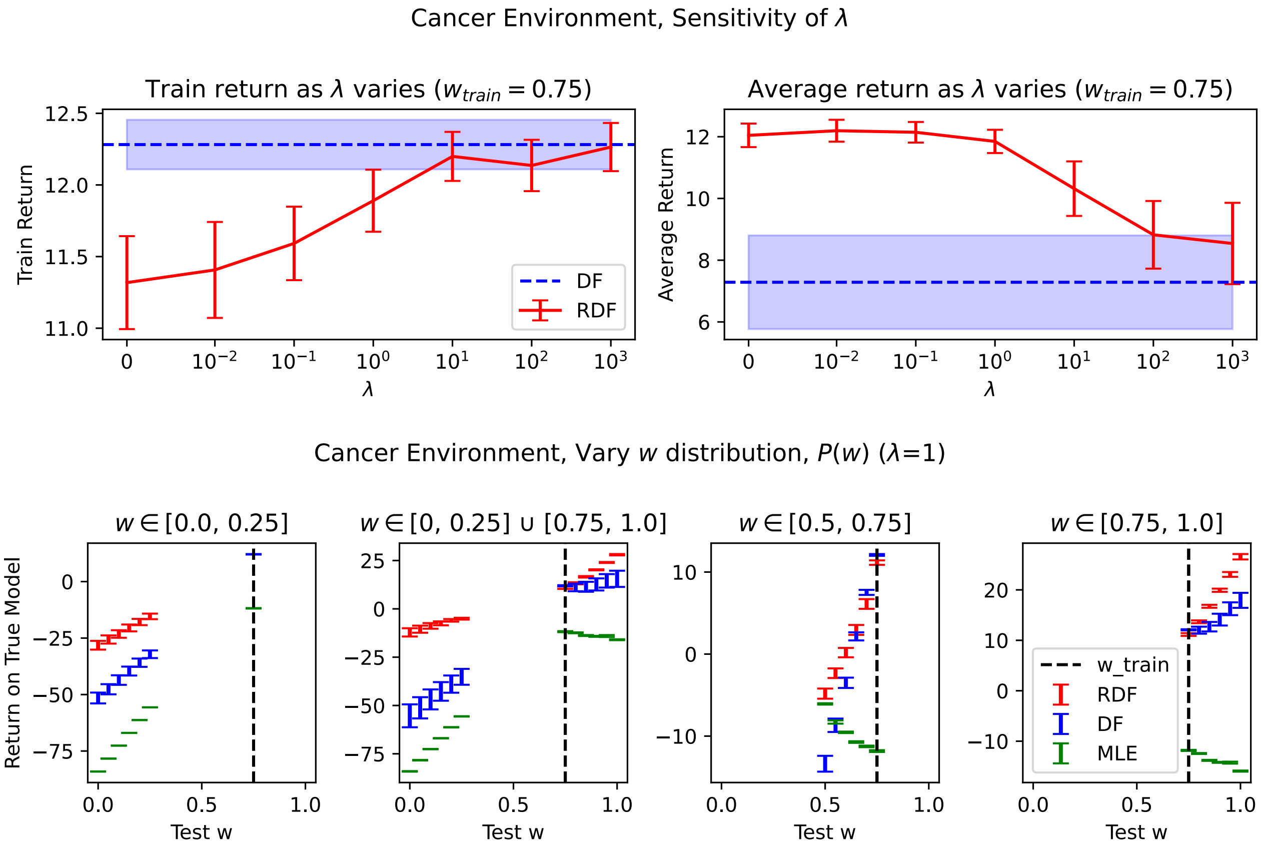

Figure 7: Analysing the sensitivity of RDF hyperparameters for the cancer environment. Top: RDF gives performance improvements over DF and MLE across a variety of λ values, and converges towards the DF behavior as we increase λ. Bottom: We can see the versatility of RDF approach: we can set $P(w)$ to encode a variety test-time preferences (we set $\lambda = 1$). The RDF solution upper bounds the DF solution in all the cases.

Training details In this domain, we use a common form of simplifying model dynamics: discretization. We use a 15×15 grid to represent the state space. We use Value Iteration to learn the optimal Q-function. We train on reward preference of $w_{train} = 0.95$, and set the test w region to be $[0.75, 1]$. The discount factor is set as 1. We report the results in Table 3.

6 Results and Discussion

The DF solution can be non-identifiable which poses a problem when re-using the model with perturbations to the reward. For the synthetic domain, Fig 3 (Top) shows two possible DF solutions θ_1 and θ_2 on the reward at train time, both yielding optimal returns of 51. Without any changes to the reward, both solutions are indistinguishable. Yet at test time if we perturb the reward, the models can have very different performances. The white panel of Fig 3 (Bottom) shows the returns w.r.t the choice of model parameter θ. Here, we observe that θ_1 has a return of around 15, while θ_2 has a return of around 40.

DF solutions do not transfer well across different reward preferences. Because of DF's sensitivity to reward preferences, although the DF model works really well for the reward it is trained on, its performance can quickly degrade if the reward function changes. In contrast, RDF leverages the DF approach to focus only on the dynamics relevant for high rewards, while *simultaneously learning a model robust to changes in reward preferences.* By doing so, the RDF policy (red) significantly outperforms the DF policy (blue) at test time, achieving near optimal performance across all the domains (Fig 2 for synthetic and Fig 4 for all domains). For the synthetic environment, since the reward at train time only depends on s_1, the DF solution θ_{DF} can achieve the optimal policy by learning θ_1^* as its value. Note that MLE only sometimes transfers better than DF (See Tables 1, 2, and 3 for detailed results on all domains), while RDF consistently transfers better than DF (Fig 4).

The RDF solution produces the highest cumulative returns in comparison to the other baselines. From Tables 1, 2, and 3, our RDF solution achieves higher returns at test time across all domains. For the Toy domain, we visualise the probability mass of returns for each method in comparison to the true model in the test environment. These results are shown in Fig 5. Overall, RDF captures better solutions over different ranges of w and thus achieves more probability mass over higher returns

RDF outperforms DF and MLE across a variety of λ and $P(w)$ values. Our results in Fig 7 (Top) show that RDF gives performance improvements over DF and MLE across a variety of λ values: by trading off train-time performance a little (left plot), we achieve robust as measured by average return across the test w's. We omit MLE because it achieves a much worse return of -11. As expected, RDF converges to DF as we increase the value of λ. Fig 7 (Bottom) and Table 1 shows that RDF achieves better overall performance for different values of $P(w)$ (i.e. different ranges of w).

RDF learning can be used to obtain performance guarantees when there is inherent uncertainty about the choice of reward. The reward function in the original cancer simulator MDP Yauney & Shah (2018) used arbitrary constants to combine the two reward basis functions, which were possibly informed by the authors'/consulting clincian's judgements. While informed, these coefficients cannot work for all the patients. RDF enables the clinicians to encode this knowledge about the patients in the distribution $P(w)$ instead of choosing a single point estimate. This serves a larger population of patients without forgoing the nice properties DF learning guarantees for simple models. RDF can thus be used to guide choices of reward functions that lead to robust performance.

Limitations A limitation of the current algorithm is scaling to more reward basis functions or to finer grids to approximate $P(w)$. This is because we can use parallelization to keep computational complexity of Eq 8 constant only upto a limit. Beyond this limit, the cost of optimizing the RDF objective would be more expensive than just computing the DF objective. Another limitation is that the algorithm requires the user to specify $P(w)$. It might not always be obvious as to what the right bounds and distribution should be.

7 Conclusion and Future Work

In this paper we extended the DF learning algorithm to produce the RDF algorithm which leads to learning of models which are more robust to changes of the reward function. We have empirically demonstrated the RDF allows for better generalization across reward functions, at the expense of minimal performance loss on the original reward function. These results suggest that there is often some degeneracy of the model parameters with respect to the returns obtained when planing with such a model. RDF learning leverages this degeneracy to select the parameters which lead the model to perform best on a specified range of tasks. Thus, a natural avenue for future work will be to investigate the relationship between the size of the set of "equally" good solution, determined by δ, and the range of changes to reward functions the agent can be robust to. Closely related is the question of how the size of the set of reward functions considered by RDF affect how well we can perform on each one - i.e. tradeoff performing very well on a small number of reward functions, or performing reasonably well on a wide range of reward functions.

References

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., and Steckelmacher, D. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 11–20. PMLR, May 2019. URL `https://proceedings.mlr.press/v97/abels19a.html`. ISSN: 2640-3498.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020. doi: 10.1073/pnas.1907370117. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1907370117`.

Barrett, L. and Narayanan, S. Learning All Optimal Policies with Multiple Criteria. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 41–47, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390162. URL `https://doi.org/10.1145/1390156.1390162`. event-place: Helsinki, Finland.

Farahmand, A.-M., Barreto, A., and Nikovski, D. Value-Aware Loss Function for Model-based Reinforcement Learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, April 2017. URL `https://proceedings.mlr.press/v54/farahmand17a.html`. ISSN: 2640-3498.

Futoma, J., Hughes, M., and Doshi-Velez, F. POPCORN: Partially Observed Prediction Constrained Reinforcement Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pp. 3578–3588. PMLR, June 2020. URL `https://proceedings.mlr.press/v108/futoma20a.html`. ISSN: 2640-3498.

Grimm, C., Barreto, A., Singh, S., and Silver, D. The Value Equivalence Principle for Model-Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5541–5552. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/3bb585ea00014b0e3ebe4c6dd165a358-Abstract.html`.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., Howley, E., Irissappane, A. A., Mannion, P., Nowé, A., Ramos, G., Restelli, M., Vamplew, P., and Roijers, D. M. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, April 2022. ISSN 1573-7454. doi: 10.1007/s10458-022-09552-y. URL `https://doi.org/10.1007/s10458-022-09552-y`.

Joseph, J., Geramifard, A., Roberts, J. W., How, J. P., and Roy, N. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, pp. 939–946, May 2013. doi: 10.1109/ICRA.2013.6630686. ISSN: 1050-4729.

Lizotte, D. J., Bowling, M., and Murphy, S. A. Efficient Reinforcement Learning with Multiple Reward Functions for Randomized Controlled Trial Analysis. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 695–702, Madison, WI, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. event-place: Haifa, Israel.

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. Multi-Objective Deep Reinforcement Learning, October 2016. URL `http://arxiv.org/abs/1610.02707`. arXiv:1610.02707 [cs].

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Nikishin, E., Abachi, R., Agarwal, R., and Bacon, P.-L. Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7886–7894, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i7.20758. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20758`. Number: 7.

Reinke, C. and Alameda-Pineda, X. Xi-learning: Successor feature transfer learning for general reward functions. *arXiv preprint arXiv:2110.15701*, 2021.

Sharma, A., Zeng, C., Narayanan, S., Parbhoo, S., and Doshi-Velez, F. On learning prediction-focused mixtures. *arXiv preprint arXiv:2110.13221*, 2021.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1-2):51–80, July 2011. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-010-5232-5. URL `http://link.springer.com/10.1007/s10994-010-5232-5`.

Wan, R., Zhang, X., and Song, R. Multi-Objective Model-based Reinforcement Learning for Infectious Disease Control. KDD '21, pp. 1634–1644, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467303. URL `https://doi.org/10.1145/3447548.3467303`.

Wang, K., Shah, S., Chen, H., Perrault, A., Doshi-Velez, F., and Tambe, M. Learning MDPs from Features: Predict-Then-Optimize for Sequential Decision Making by Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8795–8806. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/49e863b146f3b5470ee222ee84669b1c-Abstract.html`.

Wiering, M. A., Withagen, M., and Drugan, M. M. Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–6, December 2014. doi: 10.1109/ADPRL.2014.7010622. ISSN: 2325-1867.

Wiewiora, E., Cottrell, G. W., and Elkan, C. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 792–799, 2003.

Wilder, B., Ewing, E., Dilkina, B., and Tambe, M. End to end learning and optimization on graphs. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/8bd39eae38511daad6152e84545e504d-Abstract.html`.

Yamaguchi, T., Nagahama, S., Ichikawa, Y., and Takadama, K. Model-Based Multi-objective Reinforcement Learning with Unknown Weights. In Yamamoto, S. and Mori, H. (eds.), *Human Interface and the Management of Information. Information in Intelligent Systems*, Lecture Notes in Computer Science, pp. 311–321, Cham, 2019. Springer International Publishing. ISBN 978-3-030-22649-7. doi: 10.1007/978-3-030-22649-7_25.

Yauney, G. and Shah, P. Reinforcement Learning with Action-Derived Rewards for Chemotherapy and Clinical Trial Dosing Regimen Selection. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, pp. 161–226. PMLR, November 2018. URL `https://proceedings.mlr.press/v85/yauney18a.html`. ISSN: 2640-3498.

A Alternate RDF Objective Formulation

The RDF objective can also be formulated in the integral terms

$$\theta_{RDF} \leftarrow \arg\max_{\theta} \int_{w \in \mathcal{U}} J_{T_*,R_w}(\pi^*(\theta, R_w))dw$$
$$\text{s.t. } J_{T_*,R_{w_t}}(\pi^*(\theta_{DF}, R_{w_t})) - J_{T_*,R_{w_t}}(\pi^*(\theta, R_{w_t})) < \delta \quad (13)$$

where $\mathcal{U}$ specifies the region over which we wish to be robust over. While this objective is equivalent to the objective in Eqn 6 under the assumption that $P(w)$ is a uniform measure on the domain of $\mathcal{U}$, it has an intuitive interpretation: we wish to maximize the volume of the return achieved by our model θ. This intuition also motivates why we choose a uniform measure for $P(w)$.

B Additional Plots

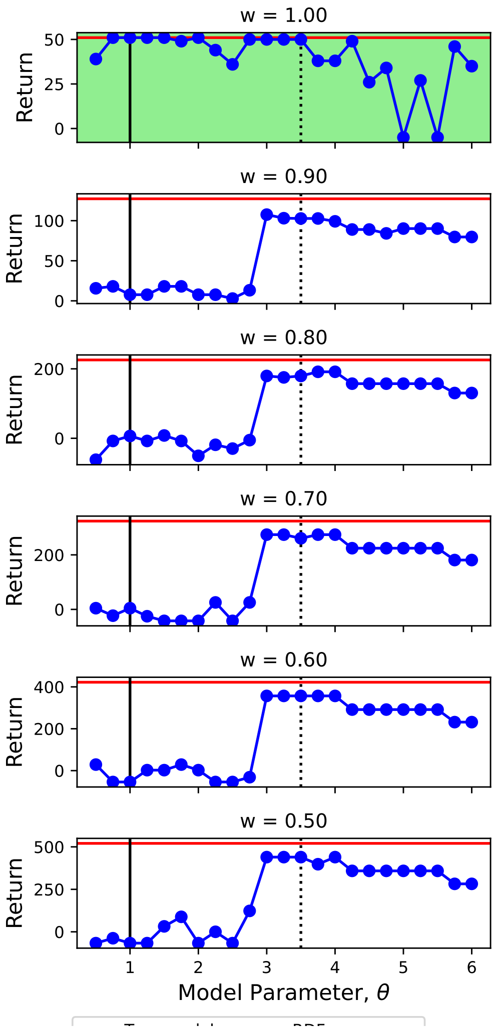

Figure 8: Model performance on synthetic Toy environment for different choice of w ($w_{train} = 1$, $w \in [.5, 1]$)

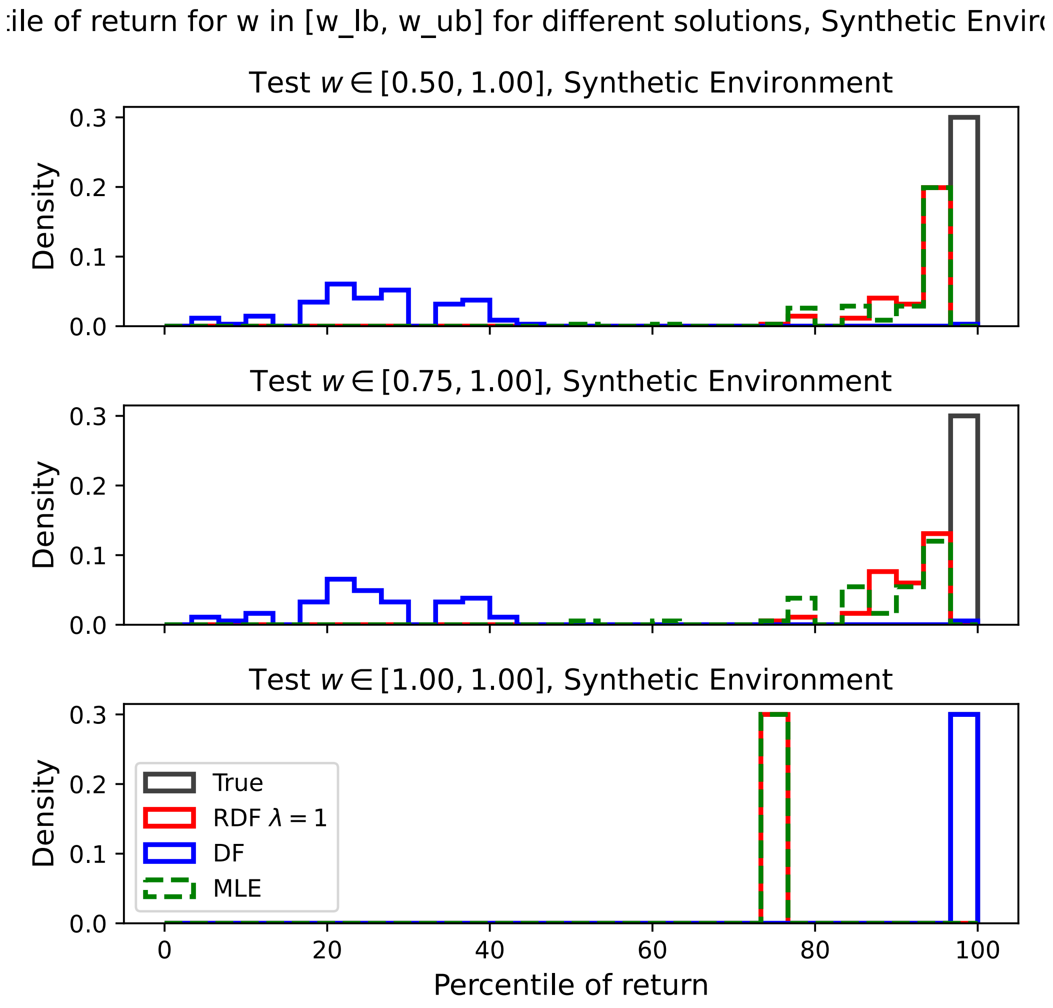

Figure 9: Comparison of DF, RDF, MLE and True solutions across w's in different ranges. RDF was optimized for $w \in [0.5, 1]$

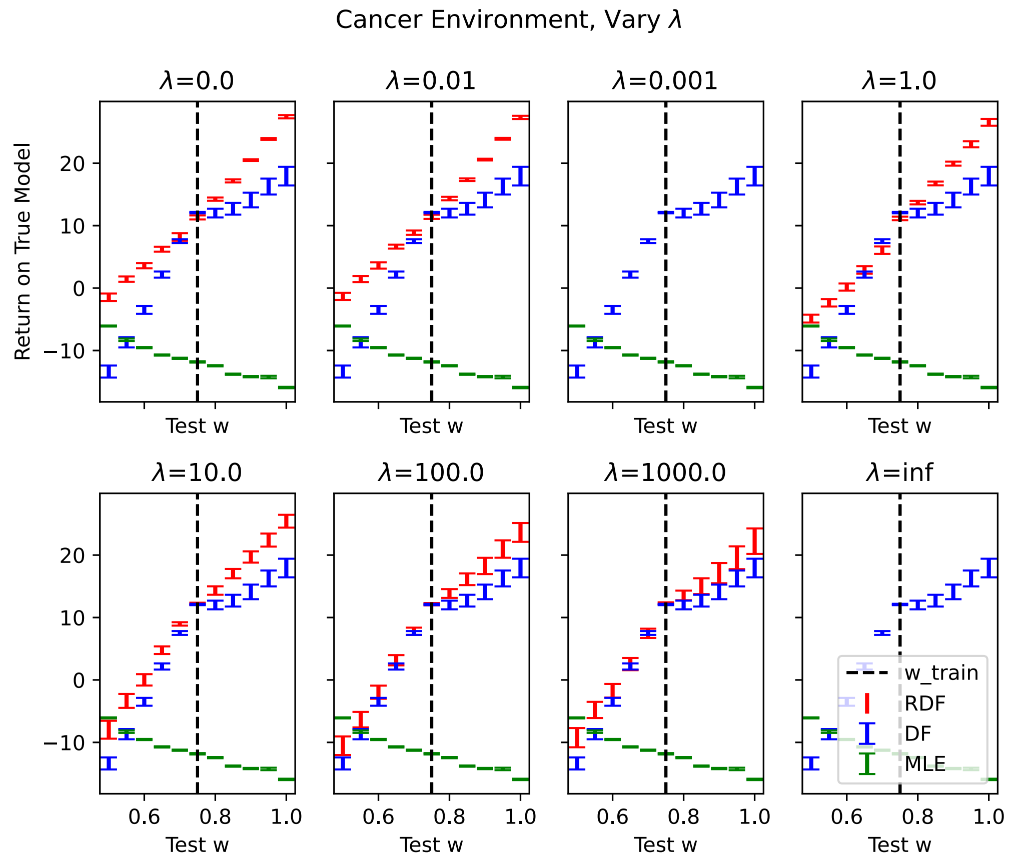

Figure 10: Effect of varying λ on Cancer domain: Increasing λ makes w_{train} performance similar to that of DF model but the average performance degrades ($w_{train} = 0.75$, $w \in [0.5, 1]$).

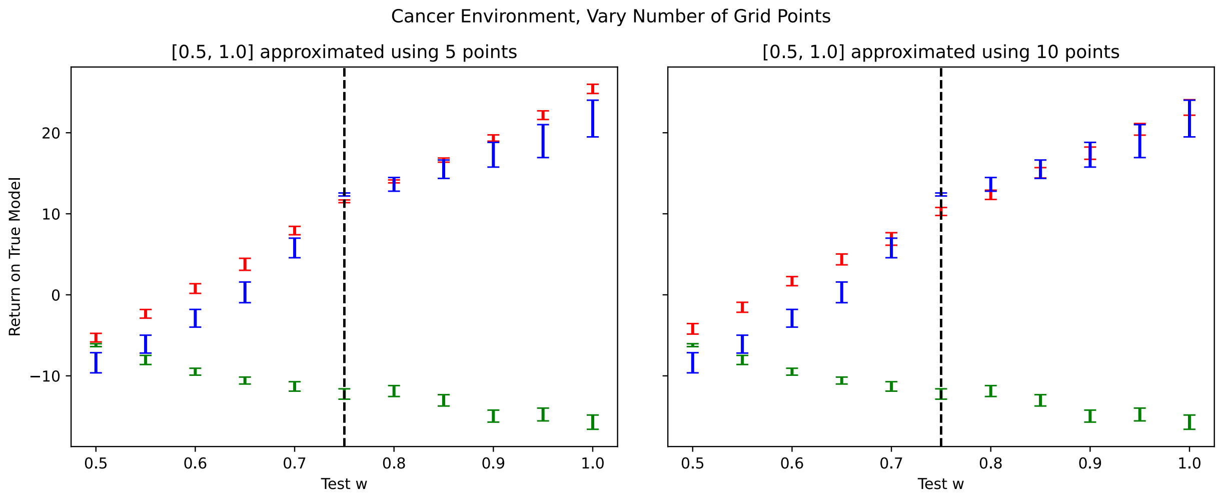

Figure 11: Effect of varying the number of point does not considerably affect the approximation on Cancer domain ($w_{train} = 0.75$, $w \in [0.5, 1]$, $\lambda = 1$)