Reinforcement Learning with Convex Constraints

Sobhan Miryoosefi
Princeton University
syoosefi@cs.princeton.edu

Kianté Brantley
University of Maryland
kdbrant@cs.umd.edu

Hal Daumé III
Microsoft Research
University of Maryland
me@hal3.name

Miro Dudík
Microsoft Research
mdudik@microsoft.com

Robert E. Schapire
Microsoft Research
schapire@microsoft.com

Abstract

In standard reinforcement learning (RL), a learning agent seeks to optimize the overall reward. However, many key aspects of a desired behavior are more naturally expressed as constraints. For instance, the designer may want to limit the use of unsafe actions, increase the diversity of trajectories to enable exploration, or approximate expert trajectories when rewards are sparse. In this paper, we propose an algorithmic scheme that can handle a wide class of constraints in RL tasks, specifically, any constraints that require expected values of some vector measurements (such as the use of an action) to lie in a convex set. This captures previously studied constraints (such as safety and proximity to an expert), but also enables new classes of constraints (such as diversity). Our approach comes with rigorous theoretical guarantees and only relies on the ability to approximately solve standard RL tasks. As a result, it can be easily adapted to work with any model-free or model-based RL algorithm. In our experiments, we show that it matches previous algorithms that enforce safety via constraints, but can also enforce new properties that these algorithms cannot incorporate, such as diversity.

1 Introduction

Reinforcement learning (RL) typically considers the problem of learning to optimize the behavior of an agent in an unknown environment against a single scalar reward function. For simple tasks, this can be sufficient, but for complex tasks, boiling the learning goal down into a single scalar reward can be challenging. Moreover, some learning objectives are unnatural to state in terms of scalar reward; safety desires, like "avoid dangerous situations," and exploration suggestions, like "maintain as uniform as possible a distribution over visited states," can be awkward to reduce to a scalar. In these, and related settings, it is much more natural to define the goal of learning in terms of a vector of *measurements* over the behavior of the agent, and to learn a policy whose measurement vector is inside a target set (§2).

We derive an algorithm, APPROachability-based Policy Optimization (APPROPO, pronounced like "apropos"), for solving such problems (§3). Given a Markov decision process with vector-valued measurements (§2), and a target constraint set, APPROPO learns a stochastic policy whose expected measurements fall in that target set (akin to Blackwell approachability in single-turn games, Blackwell et al., 1956). We derive our algorithm from a game-theoretic perspective, leveraging recent results in online convex optimization. APPROPO is implemented as a *reduction* to any off-the-shelf reinforcement learning algorithm that can return an approximately optimal policy, and so can be used in conjunction with the algorithms that are the most appropriate for any given domain.

Preprint. Under review.

Our approach builds on prior work for reinforcement learning under constraints, such as the well-studied formulation of constrained Markov Decision Process (CMDP) introduced by Altman (1999). In CMDPs, the agent's goal is to maximize reward while satisfying some linear constraints over auxiliary costs (akin to our *measurements*). Altman (1999) gave an LP-based approach when the MDP is fully known, and more recently, model-free approaches have been developed for CMDPs in high-dimensional settings. For instance, Achiam et al.'s (2017) Constrained Policy Optimization (CPO), which focused on safe exploration, learns a policy while ensuring near-constraint satisfaction during the learning process (based on Schulman et al., 2015). Reward Constrained Policy Optimization (RCPO) (Tessler et al., 2019) follows a two-timescale primal-dual approach, giving guarantees for the convergence to a fixed point. Le et al. (2019) describe a batch off-policy algorithm with PAC-style guarantees for CMDPs using a similar game-theoretic formulation to ours.

Our algorithm builds on several ideas from this literature, extending the space of constraints allowed to arbitrary convex constraints, rather than only *orthant* constraints. This enables APPROPO to work both with constraint sets proposed in these earlier papers, such as safety constraints or inequality constraints that keep the policy's behavior close to that of an expert (Syed and Schapire, 2008), as well as with constraints like the aforementioned "diversity" constraint, implemented as an entropy constraint on the policy's state visitation vector. The entropy of the visitation vector was recently studied as the objective by Hazan et al. (2018), who gave an algorithm capable of maximizing a concave function (e.g., entropy function) over visitation vectors.

Our main contributions are: (1) a new algorithm, APPROPO, for solving reinforcement learning problems with arbitrary convex constraints; (2) a rigorous theoretical analysis that demonstrates that it can achieve sublinear regret under mild assumptions (§3); and (3) a preliminary experimental comparison to RCPO (Tessler et al., 2019), showing that our algorithm satisfies convex constraints with fewer samples than RCPO (Tessler et al., 2019), while also handling a diversity constraint (§4).

2 Setup and Preliminaries: Defining the Feasibility Problem

We begin with a description of our learning setting. A *vector-valued Markov decision process* is a tuple $M = (\mathcal{S}, \mathcal{A}, \beta, P_s, P_z)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions and β is the initial-state distribution. As is usual in such a process, the initial state s_0 is selected according to β. Then on each round $i = 1, 2, \ldots$, the agent observes its current state s_i and takes action $a_i \in \mathcal{A}$ causing the agent to move to the next state $s_{i+1} \sim P_s(\cdot|s_i, a_i)$. However, in our setting, instead of receiving a scalar reward, the agent observes a d-dimensional *measurement* vector $\mathbf{z}_i \in \mathbb{R}^d$, which, like s_{i+1}, is dependent on both the current state s_i and the action a_i, that is, $\mathbf{z}_i \sim P_z(\cdot|s_i, a_i)$. (Although not explicit in our setting, reward could be incorporated in the measurement vector.)

Typically, actions are selected according to a (stationary) policy π so that $a_i \sim \pi(s_i)$, where π maps states to distributions over actions. The space of all stationary policies is denoted by Π. Our aim is to control the MDP so that measurements in the long term satisfy some constraints. Thus, for any policy π, we define the *long-term measurement* $\overline{\mathbf{z}}(\pi)$ to be the expected sum of discounted measurements:

$$\overline{\mathbf{z}}(\pi) \triangleq \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \mathbf{z}_i \;\middle|\; \pi\right] \tag{1}$$

for some discount factor $\gamma \in [0, 1)$, and where expectation is over the random process described above (including randomness inherent in π).

Later, we will also find it useful to consider *mixed policies* μ, which are distributions over finitely many stationary policies. The space of all such mixed policies is denoted $\Delta(\Pi)$. To execute a mixed policy μ, before taking any actions, a single policy π is randomly selected according to μ; then all actions henceforth are chosen from π, for the entire trajectory. The long-term measurement of a mixed policy $\overline{\mathbf{z}}(\mu)$ is defined accordingly:

$$\overline{\mathbf{z}}(\mu) \triangleq \mathbb{E}_{\pi\sim\mu}\left[\overline{\mathbf{z}}(\pi)\right] = \sum_{\pi} \mu(\pi)\overline{\mathbf{z}}(\pi). \tag{2}$$

Our learning problem is specified by a *target set* $\mathcal{C}$, which we assume is convex and compact. The *feasibility problem* is to find some $\mu \in \Delta(\Pi)$ such that $\overline{\mathbf{z}}(\mu) \in \mathcal{C}$. For instance, in our experiments (§4) we consider a simulated robotics setting where the two measurements are progress toward a goal

and torque applied to a (simulated) motor. The feasibility goal is to achieve at least a fixed amount of progress while keeping the maximum torque below a threshold for safety reasons. We can potentially also handle settings where the goal is to maximize one measurement (e.g., "reward") subject to others by performing a binary search over the maximum attainable value of the reward (see §3.4).

3 Approach, Algorithm, and Analysis

Before giving details of our approach, we overview the main ideas, which, to a large degree, follow the work of Abernethy et al. (2011) who considered the problem of solving two-player games; we extend these results to the reinforcement learning setting.

Although feasibility is our main focus, we actually solve the stronger problem of finding a policy μ that minimizes the distance between $\overline{\mathbf{z}}(\mu)$ and $\mathcal{C}$, denoted $\mathrm{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C})$, where dist is the Euclidean distance between a point and a set. That is, we want to solve:

$$\min_{\mu \in \Delta(\Pi)} \mathrm{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}). \tag{3}$$

Our main idea is to take a game-theoretic approach, formulating this problem as a game and solving it. Specifically, suppose we can express the distance function in Eq. (3) as a maximization of the form:

$$\mathrm{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) = \max_{\boldsymbol{\lambda} \in \Lambda} \boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu) \tag{4}$$

for some convex, compact set Λ (achieving this is described in §3.2). Then Eq. (3) becomes:

$$\min_{\mu \in \Delta(\Pi)} \max_{\boldsymbol{\lambda} \in \Lambda} \boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu), \tag{5}$$

whose min-max form immediately evokes interpretation as a two-person zero-sum game. Assuming this game satisfies certain conditions, we can write it equivalently with min and max reversed:

$$\max_{\boldsymbol{\lambda} \in \Lambda} \min_{\mu \in \Delta(\Pi)} \boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu). \tag{6}$$

Note that the policy μ we are seeking is the *solution* of this game, that is, the policy realizing the minimum in Eq. (5). Therefore, to find that policy, we can apply general techniques for solving a game, namely, to let a no-regret learning algorithm play the game repeatedly against a best-response player. When played in this way, it can be shown that the averages of their plays converge to the solution of the game (details in §3.1).

In our case, we can use a no-regret player for the $\boldsymbol{\lambda}$-player, and best response for the μ-player. Importantly, in our context, computing best response turns out to be an especially convenient task. Given $\boldsymbol{\lambda}$, best response means finding the mixed policy μ minimizing $\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu)$. As we show below, this can be solved by treating the problem as a standard reinforcement learning task with scalar reward r that is defined in terms of $\boldsymbol{\lambda}$ and the measurement vectors $\mathbf{z}$, namely, by letting $r = -\boldsymbol{\lambda} \cdot \mathbf{z}$.

In the remainder of this section, we provide the details of our approach, leading to our main algorithm and its analysis, and conclude with a discussion of steps for making a practical implementation. We begin by discussing game-playing techniques in general, which we then apply to our setting.

3.1 Solving zero-sum game using online learning

At the core of our approach, we use the general technique of Freund and Schapire (1999) for solving a game by repeatedly playing a no-regret online learning algorithm against best response.

For this purpose, we first briefly review the framework of online convex optimization (OCO), which we will soon use for one of the players. In this setting, at time $t = 1, \ldots, T$, the learner makes a decision $\boldsymbol{\lambda}_t$ in some convex decision set Λ, then receives the convex loss function $\ell_t : \Lambda \to \mathbb{R}$, and incurs loss $\ell_t(\boldsymbol{\lambda}_t)$. The goal of a learning algorithm $\mathcal{O}$ is to minimize its *regret*, defined as

$$\mathrm{Regret}_T(\mathcal{O}) = \sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}_t) - \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}). \tag{7}$$

Algorithm 1 Solving a game with repeated play

input concave-convex function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, online convex optimization algorithm $\mathcal{O}_\mathcal{X}$
for $t = 1$ **to** T **do**
  $\mathcal{O}_\mathcal{X}$ makes a decision $x_t \in \mathcal{X}$
  $y_t \leftarrow \operatorname{argmin}_{y \in \mathcal{Y}} g(x_t, y)$
  $\mathcal{O}_\mathcal{X}$ observes loss function $\ell_t(x) = -g(x, y_t)$
end for
return $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ and $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$

An algorithm is *no-regret* if $\text{Regret}_T(\mathcal{O}) = o(T)$, meaning its average loss approaches the best in hindsight. An example of such an algorithm is *Online Gradient Descent (OGD)* by Zinkevich (2003) (see Appendix A). If Λ is bounded with diameter D, and also the norm of the gradients of loss functions are bounded by G, then its regret is at most $DG\sqrt{T}$.

Now consider a two-player zero-sum game in which Player 1 and Player 2 select, respectively, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ (where both spaces are assumed convex and compact) resulting in the outcome $g(x, y)$, which Player 1 wants to maximize and Player 2 wants to minimize. By the minimax theorem (von Neumann, 1928; Sion, 1958), assuming g is concave in x and convex in y, it is known that

$$\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y) = \min_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} g(x, y). \tag{8}$$

This means Player 1 (the "x-player") has an "optimal" strategy which realizes the maximum on the left and guarantees payoff of at least the *value* of the game, i.e., the value given by this expression. Similarly for Player 2 (the "y-player").

We can solve this game (find these optimal strategies) by playing it repeatedly. Specifically, we use an OCO learning algorithm $\mathcal{O}_\mathcal{X}$ as the x-player. At each timestep $t = 1, \ldots, T$, $\mathcal{O}_\mathcal{X}$ chooses $x_t \in \mathcal{X}$. In response, the y-player, who in this setting is permitted knowledge of x_t, selects y_t to minimize the resulting outcome $g(x_t, y_t)$, that is, $y_t = \operatorname{argmin}_{y \in \mathcal{Y}} g(x_t, y)$. This is called *best response*. To update $\mathcal{O}_\mathcal{X}$, we set the loss function it observes to be $\ell_t(x) = -g(x, y_t)$. See Algorithm 1.

As stated in Theorem 3.1, $\bar{x}$ and $\bar{y}$, the averages of the players' decisions, converge to the solution of the game, assuming the online learning algorithm guarantees sublinear regret. The proof of Theorem 3.1 is given in Appendix B.

Theorem 3.1. *Let v be the value of the game in Eq. (8). Then for $\bar{x}$ and $\bar{y}$ we have*

$$\min_{y \in \mathcal{Y}} g(\bar{x}, y) \geq v - \delta_T^\mathcal{X} \quad \textit{and} \quad \max_{x \in \mathcal{X}} g(x, \bar{y}) \leq v + \delta_T^\mathcal{X}, \quad \textit{where } \delta_T^\mathcal{X} = \frac{1}{T}\text{Regret}_T(\mathcal{O}_\mathcal{X}). \tag{9}$$

3.2 Algorithm and main result

We can now apply this game-playing framework to the approach outlined at the beginning of this section. First, we show how to write distance as a maximization, as in Eq. (4). For now, we assume that our target set $\mathcal{C}$ is a *convex cone*, that is, closed under summation and also multiplication by non-negative scalars (we will remove this assumption in §3.3). With this assumption, we can apply the following lemma, from Lemma 13 in (Abernethy et al., 2011), in which distance to a convex cone $\mathcal{K} \subseteq \mathbb{R}^d$ is written as a maximization over a dual convex cone $\mathcal{K}^\circ$ called the *polar cone*:

$$\mathcal{K}^\circ \triangleq \{\boldsymbol{\lambda} \in \mathbb{R}^d : \boldsymbol{\lambda} \cdot \mathbf{x} \leq 0 \text{ for all } \mathbf{x} \in \mathcal{K}\}. \tag{10}$$

Lemma 3.2. *For a convex cone $\mathcal{K} \subseteq \mathbb{R}^d$ and any point $\mathbf{x} \in \mathbb{R}^d$*

$$\text{dist}(\mathbf{x}, \mathcal{K}) = \max_{\boldsymbol{\lambda} \in \mathcal{K}^\circ \cap \mathcal{B}(1)} \boldsymbol{\lambda} \cdot \mathbf{x}, \tag{11}$$

where $\mathcal{B}(r)$ is the l_2-ball of radius r about the origin.

As a consequence, Eq. (4) is immediately achieved by setting $\Lambda = \mathcal{C}^\circ \cap \mathcal{B}(1)$.

Next, to see that Eqs. (5) and (6) are equal, it can be checked that the Sion (1958) conditions hold since both Λ and $\Delta(\Pi)$ are convex and compact, and the payoff $\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu)$ is affine in both $\boldsymbol{\lambda}$ and μ.

Now, if we instantiate Algorithm 1 for our case, using an OCO algorithm $\mathcal{O}_\Lambda$ as the $\boldsymbol{\lambda}$-player and best response as the μ-player, then Theorem 3.1 immediately implies

$$\operatorname{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) \leq \min_{\mu \in \Delta(\Pi)} \operatorname{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) + \frac{1}{T}\operatorname{Regret}(\mathcal{O}_\Lambda), \tag{12}$$

where μ is the strategy returned for the μ-player. This means we can get arbitrarily close to the target set if $\mathcal{O}_\Lambda$ is no-regret. If the problem is feasible, then $\min_{\mu \in \Delta(\Pi)} \operatorname{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) = 0$ which means our long-term measurement $\overline{\mathbf{z}}(\mu)$ converges to the target set.

To apply this game-playing approach, we need to implement the two players in our setting, one an OCO algorithm, and the other playing best response. We discuss these next, beginning with the latter.

The best-response player, for a given $\boldsymbol{\lambda}$, aims to minimize $\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu)$ over mixed policies μ, or equivalently (by Eq. (2)), to minimize $\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\pi)$ over stationary policies π. The key point, as already mentioned, is that this is the same as finding a policy that maximizes long-term reward in a standard reinforcement learning task if we define the scalar reward to be $r_i = -\boldsymbol{\lambda} \cdot \mathbf{z}_i$. This is because the reward of a policy π is given by

$$R(\pi) \triangleq \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_i \;\middle|\; \pi\right] = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i (-\boldsymbol{\lambda} \cdot \mathbf{z}_i) \;\middle|\; \pi\right] = -\boldsymbol{\lambda} \cdot \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \mathbf{z}_i \;\middle|\; \pi\right] = -\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\pi). \tag{13}$$

Therefore, maximizing $R(\pi)$, as in standard RL, is equivalent to minimizing $\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\pi)$.

Thus, to implement best response, we assume access to a *planning oracle* for finding a policy that maximizes reward in an entirely standard way, using any one of the many well-studied techniques for this problem. For robustness, we allow this oracle to return an approximately optimal policy.

Planning Oracle: $\operatorname{Plan}(\boldsymbol{\lambda})$. Given $\boldsymbol{\lambda} \in \mathbb{R}^d$, return policy $\pi \in \Pi$ with $R(\pi) \geq \max_{\pi' \in \Pi} R(\pi') - \epsilon_0$, where $R(\pi)$ is the long-term reward of policy π with scalar reward $r = -\boldsymbol{\lambda} \cdot \mathbf{z}$.

For the OCO player, we do our analysis using online gradient descent (Zinkevich, 2003) (in experiments, we use OGD with momentum; see §4), an effective no-regret learner. For its update, OGD needs the gradient of the loss functions $\ell_t(\boldsymbol{\lambda}) = -\boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\pi_t)$, which is just $-\overline{\mathbf{z}}(\pi_t)$. With access to the MDP, $\overline{\mathbf{z}}(\pi)$ can be estimated simply by taking multiple trajectories using π and averaging the observed measurements appropriately. We formalize this by assuming access to an *estimation oracle* for estimating $\overline{\mathbf{z}}(\pi)$.

Estimation Oracle: $\operatorname{Est}(\pi)$. Given policy π, return $\hat{\mathbf{z}}$ satisfying $\|\hat{\mathbf{z}} - \overline{\mathbf{z}}(\pi)\|_2 \leq \epsilon_1$.

OGD also requires projection to the set $\Lambda = \mathcal{C}^\circ \cap \mathcal{B}(1)$. In fact, if we can simply project onto the target set $\mathcal{C}$, which is more natural, then it is possible to also project onto Λ. Consider an arbitrary x and its projection, into $\mathcal{C}$ denoted by $\Gamma_{\mathcal{C}}(x)$, it is known that projection of x into the polar cone is $\Gamma_{\mathcal{C}^\circ}(x) = x - \Gamma_{\mathcal{C}}(x)$ (Ingram and Marsh, 1991). Knowing projection into both $\mathcal{C}^\circ$ and $\mathcal{B}(1)$, it can be checked that projection into Λ is $\Gamma_\Lambda(\mathbf{x}) = (\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x}))/\max\{1, \|\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x})\|_2\}$ (e.g. Dykstra's projection algorithm converges to this point after 2 steps Boyle and Dykstra, 1986). We therefore also assume such a *projection oracle* for $\mathcal{C}$.

Projection Oracle: $\Gamma_{\mathcal{C}}(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}' \in \mathcal{C}} \|\mathbf{x} - \mathbf{x}'\|_2$.

Pulling these ideas together and plugging into Algorithm 1, we obtain our main algorithm, called APPROPO (Algorithm 2), for APPROachability-based Policy Optimization. The algorithm provably yields a mixed policy that approximately minimizes distance to the set $\mathcal{C}$, as shown in Theorem 3.3, whose proof is given in Appendix C.

Theorem 3.3. *Assume that for all measurements we have $\|\mathbf{z}\| \leq B$ and also $\mathcal{C}$ is a convex cone. Suppose we run Algorithm 2 for T rounds with $\eta = \left(\frac{B}{1-\gamma} + \epsilon_1\right)^{-1} T^{-1/2}$. Then*

$$\operatorname{dist}(\overline{\mathbf{z}}(\bar{\mu}), \mathcal{C}) \leq \min_{\mu \in \Delta(\Pi)} \operatorname{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) + \left(\tfrac{B}{1-\gamma} + \epsilon_1\right) T^{-1/2} + \epsilon_0 + 2\epsilon_1, \tag{14}$$

where $\bar{\mu}$ is the mixed policy returned by the algorithm.

Algorithm 2 APPROPO

input projection oracle $\Gamma_{\mathcal{C}}(\cdot)$ for target set $\mathcal{C}$ which is a convex cone,
planning oracle $\mathrm{Plan}(\cdot)$, estimation oracle $\mathrm{Est}(\cdot)$,
step size η, number of iterations T
let $\Lambda := \mathcal{C}^\circ \cap \mathcal{B}(1)$, and define its projection operator $\Gamma_\Lambda(\mathbf{x}) \triangleq (\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x}))/\max\{1, \|\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x})\|_2\}$
initialize $\boldsymbol{\lambda}_1$ arbitrarily in Λ
for $t = 1$ **to** T **do**
  Compute an approximately optimal policy for standard RL with scalar reward $r = -\boldsymbol{\lambda}_t \cdot \mathbf{z}$:
  $\pi_t \leftarrow \mathrm{Plan}(\boldsymbol{\lambda}_t)$
  Call the estimation oracle to approximate long-term measurement for π_t:
  $\hat{\mathbf{z}}_t \leftarrow \mathrm{Est}(\pi_t)$
  Update using online gradient descent with the loss function $\ell_t(\boldsymbol{\lambda}) = -\boldsymbol{\lambda} \cdot \hat{\mathbf{z}}_t$:
  $\boldsymbol{\lambda}_{t+1} \leftarrow \Gamma_\Lambda\big(\boldsymbol{\lambda}_t + \eta \hat{\mathbf{z}}_t\big)$
end for
return $\bar{\mu}$, a uniform mixture over $\pi_1, \ldots, \pi_T$

When, as in the original formulation of the problem, the goal is to find a mixed policy in $\mathcal{C}$, we can make use of a weaker planning oracle, and we also can sometimes detect when a problem is in fact infeasible. Specifically, we only need a *positive response* oracle that finds a policy that is "good enough" in the sense of providing long-term reward above some threshold:

Positive Response Oracle: $\mathrm{PosPlan}(\boldsymbol{\lambda})$. Given $\boldsymbol{\lambda} \in \mathbb{R}^d$, return policy $\pi \in \Pi$ with $R(\pi) \geq -\epsilon_0$ if $\max_{\pi' \in \Pi} R(\pi') \geq 0$ (and arbitrary π otherwise), where $R(\pi)$ is the long-term reward of policy π with scalar reward $r = -\boldsymbol{\lambda} \cdot \mathbf{z}$.

When the problem is feasible, it can be shown that there must exist $\pi \in \Pi$ with $R(\pi) \geq 0$, and furthermore, that $\ell_t(\boldsymbol{\lambda}_t) \geq -(\epsilon_0 + \epsilon_1)$ (from Lemma C.1 in Appendix C). This means, if the goal is feasibility, we can modify Algorithm 2, replacing Plan with $\mathrm{PosPlan}$, and adding a test at the end of each iteration to report infeasibility if $\ell_t(\boldsymbol{\lambda}_t) < -(\epsilon_0 + \epsilon_1)$. The pseudocode is given in Algorithm 4 in Appendix D along with the proof of the following convergence bound:

Theorem 3.4. *Assume that for all measurements we have $\|\mathbf{z}\| \leq B$ and also $\mathcal{C}$ is a convex cone. Suppose we run Algorithm 4 for T rounds with $\eta = \big(\frac{B}{1-\gamma} + \epsilon_1\big)^{-1} T^{-1/2}$. Then either the algorithm reports infeasibility or returns $\bar{\mu}$ such that*

$$\mathrm{dist}(\overline{\mathbf{z}}(\bar{\mu}), \mathcal{C}) \leq \big(\tfrac{B}{1-\gamma} + \epsilon_1\big) T^{-1/2} + \epsilon_0 + 2\epsilon_1. \tag{15}$$

3.3 Removing the Cone Assumption

Our results so far have assumed the target set $\mathcal{C}$ is a convex cone. If instead $\mathcal{C}$ is an arbitrary convex, compact set, we can use a technique like the one from Abernethy et al. (2011) in which we apply our algorithm to some convex cone $\tilde{\mathcal{C}}$ constructed from $\mathcal{C}$ to obtain a solution with provable guarantees.

In more detail, given a compact, convex target set $\mathcal{C} \subseteq \mathbb{R}^d$, we augment every vector in $\mathcal{C}$ with a new coordinate held fixed to some value $\kappa > 0$, and then let $\tilde{\mathcal{C}}$ be its conic hull. Thus,

$$\tilde{\mathcal{C}} = \mathrm{cone}(\mathcal{C} \times \{\kappa\}), \qquad \text{where } \mathrm{cone}(\mathcal{X}) = \{\alpha \mathbf{x} \mid \mathbf{x} \in \mathcal{X}, \alpha \geq 0\}. \tag{16}$$

Given our original vector-valued MDP $M = (\mathcal{S}, \mathcal{A}, \beta, P_s, P_z)$, we define a new MDP $M' = (\mathcal{S}, \mathcal{A}, \beta, P_s, P'_{z'})$ with $(d+1)$-dimensional measurement $\mathbf{z}' \in \mathbb{R}^{d+1}$, defined (and generated) by

$$\mathbf{z}'_i = \mathbf{z}_i \oplus \langle (1-\gamma)\kappa \rangle \qquad \mathbf{z}_i \sim P_z(\cdot \mid s_i, a_i) \tag{17}$$

where $\oplus$ denotes vector concatenation. Writing long-term measurement for M and M' as $\overline{\mathbf{z}}$ and $\overline{\mathbf{z}}'$ respectively, $\overline{\mathbf{z}}'(\pi) = \overline{\mathbf{z}}(\pi) \oplus \langle \kappa \rangle$, for any policy $\pi \in \Pi$, and similarly for any mixed policy μ.

The main idea of our reduction is to apply the algorithms described above to the modified MDP M' using the cone $\tilde{\mathcal{C}}$ as target set. For an appropriate choice of $\kappa > 0$, we claim that the resulting mixed policy will approximately minimize distance to $\mathcal{C}$ for the original MDP M. This is a consequence of the following lemma, an extension of Lemma 14 in (Abernethy et al., 2011), which shows that distances are largely preserved in a controllable way under this construction. The proof is given in Appendix E.

Lemma 3.5. *Consider a compact convex set $\mathcal{K}$ in $\mathbb{R}^d$ and $\mathbf{x} \in \mathbb{R}^d$. For any $\delta > 0$, let $\tilde{\mathcal{K}} = \mathrm{cone}(\mathcal{K} \times \{\kappa\})$, where $\kappa = \frac{\max_{\mathbf{x} \in \mathcal{K}} \|\mathbf{x}\|_2}{\sqrt{2\delta}}$. Then $\mathrm{dist}(\mathbf{x}, \mathcal{K}) \leq (1+\delta)\mathrm{dist}(\mathbf{x} \oplus \langle \kappa \rangle, \tilde{\mathcal{K}})$.*

Corollary 3.6. *Assume that for all measurements we have $\|\mathbf{z}\| \leq B$ and also $\mathcal{C}$ is convex and compact. Then by putting $\eta = \left(\frac{B+\kappa}{1-\gamma} + \epsilon_1\right)^{-1} T^{-1/2}$ and running Algorithm 2 for T rounds with M' as the MDP and $\tilde{\mathcal{C}}$ as the target set, the mixed policy $\bar{\mu}$ returned by the algorithm satisfies*

$$\mathrm{dist}(\overline{\mathbf{z}}(\bar{\mu}), \mathcal{C}) \leq (1+\delta)\left(\min_{\mu \in \Delta(\Pi)} \mathrm{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) + \left(\tfrac{B+\kappa}{1-\gamma} + \epsilon_1\right) T^{-1/2} + \epsilon_0 + 2\epsilon_1\right), \tag{18}$$

where $\kappa = \frac{\max_{\mathbf{x} \in \mathcal{C}} \|\mathbf{x}\|_2}{\sqrt{2\delta}}$ for an arbitrary $\delta > 0$. Similarly for Algorithm 4, we either have

$$\mathrm{dist}(\overline{\mathbf{z}}(\bar{\mu}), \mathcal{C}) \leq (1+\delta)\left(\left(\tfrac{B+\kappa}{1-\gamma} + \epsilon_1\right) T^{-1/2} + \epsilon_0 + 2\epsilon_1\right) \tag{19}$$

or it reports infeasibility.

3.4 Practical implementation of the positive response and estimation oracles

We next briefly describe a few techniques for the practical implementation of our algorithm.

As discussed in §3.2, when our aim is to solve a feasibility problem, we can use a variant (Algorithm 4) which only relies on access to a positive response oracle. In episodic environments, it is straightforward to use any standard iterative reinforcement learning approach as a positive response oracle: As the reinforcement learning algorithm runs, we track its accrued rewards, and when the trailing average of the last n trajectory-level rewards goes above some level $-\epsilon$, we return the current policy (possibly specified implicitly as a Q function).[1] Furthermore, the average of the measurement vectors $\mathbf{z}$ collected over the last n trajectories can serve as the estimate $\hat{\mathbf{z}}_t$ of the long-term measurement required by the algorithm, side-stepping the need for an additional estimation oracle. The hyperparameters ϵ and n influence the oracle quality; specifically, assuming that the rewards are bounded and the overall number of trajectories until the oracle terminates is at most polynomial in n, we have $\epsilon_0 = \epsilon - O(\sqrt{(\log n)/n})$ and $\epsilon_1 = O(\sqrt{(\log n)/n})$. In theory we could use Theorem 3.4 to select a value T at which to stop; in practice we run until the distance to the constraint set $\mathcal{C}$ of the mixed policy is zero (or negligible). If the reinforcement learning algorithm runs for too long without moving to non-negative rewards, we can stop and declare that the underlying problem is "empirically infeasible." (Actual infeasibility would hold if it is truly not possible to reach non-negative expected reward.)

We can further speed up our algorithm by maintaining a list, or "cache," of all the policies returned by the positive response oracle so far. Each of the stored policies π is annotated with the estimate of its expected measurement vector $\hat{\mathbf{z}}(\pi) \approx \bar{\mathbf{z}}(\pi)$, based on the last n iterations of the RL oracle as discussed above. On each iteration of our algorithm, we first check if the cache contains a policy that already achieves a reward at least $-\epsilon$, which can be determined from $\hat{\mathbf{z}}(\pi)$ since the reward is just a linear function of the measurement vector. If such a policy is found, we return it, alongside $\hat{\mathbf{z}}(\pi)$, instead of calling the oracle. Otherwise, we pick the policy with the largest reward (necessarily below $-\epsilon$) and use it to warm-start the RL algorithm implementing the oracle. The cache can be initialized with a few random policies (as we do in our experiments), effectively implementing randomized weight initialization.

In addition, the cache interplays well with a straightforward binary-search scheme that can be used when the goal is to maximize some reward (possibly subject to additional constraints). In this case, the feasibility problems corresponding to iterates of binary search only differ in the constraint values, but use the same measurements, so that the same cache can be reused across all iterations.

4 Experimental Setup and Results

We evaluated APPROPO on the "Mars Rover" navigation grid domain described below to show that our method is competitive with previous approaches, and to show empirically that it can solve

[1] This assumes that the last n trajectories accurately estimate the performance of the final iterate. If that is not the case, the oracle can instead return the mixture of the policies corresponding to the last n iterates.

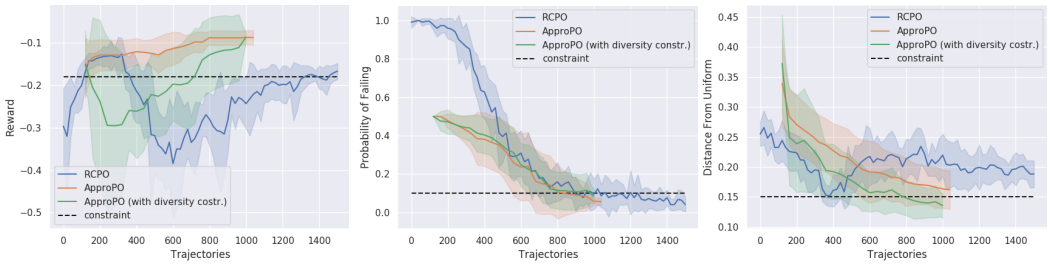


Figure 1: Experimental results comparing RCPO to APPROPO (with and without a diversity constraint) on Mars Rover. In all figures, the x-axis is number of trajectories sampled. **(Left)** reward of the learned policies; **(Middle)** probability with which the learned policies fail to meet the constraints (hitting rocks) with the threshold as a dashed line; **(Right)** diversity of states visited with the threshold (for APPROPO with diversity constraint) as a dashed line.

convex constraints. We compared to RCPO (Tessler et al., 2019), the most natural alternative technique, which is a primal-dual approach guaranteed to find the fixed point of the Lagrangian of a constrained policy optimization problem. Our online convex optimizer was online gradient descent with momentum. We considered reward, constraint satisfaction, and diversity, which we plotted as functions of the number of trajectories run. We experimented with APPROPO both with and without a diversity constraint on the expected state visitation vector. For the two variants of our method, as well as RCPO, we report mean and standard deviation over ten random restarts.[2]

Mars Rover. We considered a small version of the Mars Rover grid-world environment that Tessler et al. (2019) evaluated RCPO on, in which the agent must move from the starting position to a goal without crashing into rocks (both of which terminate the episode). The environment is stochastic: with probability $\delta = 0.05$ the agent will take a random action. The agent receives small negative reward each time step and a reward of zero for terminating. This environment is challenging because the agent is incentivized to end the episode as soon as possible. We used the same safety constraint as Tessler et al. (2019): ensure that the probability of hitting a rock is at most a fixed threshold (set to 0.1). For generating our best response player, to facilitate comparison to RCPO, we used Actor-Critic (A2C) to train the policy (Mnih et al., 2016) (with the learning rate of 10^{-2}). For APPROPO, the constant κ (§3.3) is set to be 10. Our algorithm solves a feasibility problem with the same safety constraint on the probability of hitting a rock and the average reward constrained to be at least -0.18 (this equals the final reward achieved by RCPO). In addition, we also experimented with including the diversity constraint stating that the Euclidean distance between our visitation probability vector (across the cells of the grid) and the uniform distribution over the upper-triangle cells of the grid (excluding rocks) is at most 0.15.

Experimental Results. All results are shown in Figure 1. Both variants of our algorithm were able to achieve a larger reward faster than the baseline, while also satisfying the constraint on the probability of failure. Furthermore, including the diversity constraint allowed our method to reach a more diverse policy. The presence of the additional diversity constraint somewhat slowed the convergence in terms of reward, but our approach still achieved the same high reward as without the constraint. While there was substantial variance among the runs of our algorithm (due to its batch nature), the final solutions attained similar narrow values across all three measurements (reward, probability of failure, and diversity).

[2]Because APPROPO starts by estimating expected measurement vectors for the initial policies in the cache (§3.4), it does not start receiving any reward until after 120 trajectories are executed. Furthermore, because APPROPO operates in a batch-like fashion—and often makes substantial progress using the cache without requiring any more trajectories during its best response phase—it makes computing mean performance (and deviations) difficult because different runs do not "align." To address this, for each of the ten runs, we first linearly interpolate the reward over time, and then compute means and deviations over those interpolations.

5 Conclusion

In this paper, we introduced APPROPO, an algorithm based on a game-theoretic approach for solving reinforcement learning problems with arbitrary convex constraints. APPROPO can operate with any no-regret online learner, and any positive response reinforcement learning algorithm (i.e., one which returns any policy with non-negative reward whenever such a policy exists). Theoretically, we showed that for the specific case of online gradient descent, APPROPO learns to approach the constraint set at a rate of $1/\sqrt{T}$, with an additive non-vanishing term that measures the optimality gap of the reinforcement learner. Experimentally, we demonstrated that APPROPO can be applied with well-known reinforcement learning algorithms for discrete domains (like actor-critic), and achieved strong performance using a small number of trajectories in comparison to RCPO (Tessler et al., 2019), while additionally being able to satisfy a diversity constraint. In sum, this yields a theoretically justified, practical algorithm for solving the approachability problem in reinforcement learning.

References

Abernethy, J., Bartlett, P. L., and Hazan, E. (2011). Blackwell approachability and no-regret learning are equivalent. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 27–46.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31.

Altman, E. (1999). *Constrained Markov decision processes*, volume 7. CRC Press.

Blackwell, D. et al. (1956). An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8.

Boyle, J. P. and Dykstra, R. L. (1986). A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer.

Freund, Y. and Schapire, R. E. (1999). Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103.

Hazan, E., Kakade, S. M., Singh, K., and Van Soest, A. (2018). Provably efficient maximum entropy exploration. *arXiv preprint arXiv:1812.02690*.

Ingram, J. M. and Marsh, M. (1991). Projections onto convex cones in hilbert space. *Journal of approximation theory*, 64(3):343–350.

Le, H. M., Voloshin, C., and Yue, Y. (2019). Batch policy learning under constraints. *arXiv preprint arXiv:1903.08738*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Sion, M. (1958). On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176.

Syed, U. and Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tessler, C., Mankowitz, D. J., and Mannor, S. (2019). Reward constrained policy optimization. In *International Conference on Learning Representations*.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Supplementary Material For: Reinforcement Learning with Convex Constraints

A Online Gradient Descent (OGD)

Algorithm 3 Online Gradient Descent (OGD)

input: projection oracle Γ_Λ $\{\Gamma_\Lambda(\boldsymbol{\lambda}) = \operatorname{argmin}_{\boldsymbol{\lambda}' \in \Lambda} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|_2\}$
init: $\boldsymbol{\lambda}_1$ arbitrarily
parameters: step size η_t
for $t = 1$ **to** T **do**
    $\boldsymbol{\lambda}'_{t+1} = \boldsymbol{\lambda}_t - \eta_t \nabla \ell_t(\boldsymbol{\lambda}_t)$
    $\boldsymbol{\lambda}_{t+1} = \Gamma_\Lambda(\boldsymbol{\lambda}'_{t+1})$
end for

Theorem A.1. *(Zinkevich, 2003) Assume that for any $\boldsymbol{\lambda}, \boldsymbol{\lambda}' \in \Lambda$ we have $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|_2 \leq D$ and also $\|\nabla \ell_t(\boldsymbol{\lambda})\|_2 \leq G$. Let $\eta_t = \eta = \frac{D}{G\sqrt{T}}$. Then the regret of OGD is*

$$\operatorname{Regret}_T(\textit{OGD}) \leq DG\sqrt{T}.$$

B Proof of Theorem 3.1

We have that

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} g(x_t, y_t) &= \frac{1}{T}\sum_{t=1}^{T} \min_{y \in \mathcal{Y}} g(x_t, y) && (20)\\
&\leq \frac{1}{T} \min_{y \in \mathcal{Y}} \sum_{t=1}^{T} g(x_t, y) && (21)\\
&\leq \min_{y \in \mathcal{Y}} g\left(\frac{1}{T}\sum_{t=1}^{T} x_t, y\right) && (22)\\
&\leq \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y). && (23)
\end{aligned}$$

Eq. (20) is because the y-player is playing best response so that $y_t = \operatorname{argmin}_{y \in \mathcal{Y}} g(x_t, y)$. Eq. (21) is because taking the minimum of each term of a sum cannot exceed the minimum of the sum as a whole. Eqs. (22) and (23) use the concavity of g with respect to x, and the definition of $\max$, respectively. Writing the definition of regret for the x-player and using $\ell_t(x) = -g(x, y_t)$, we have

$$\frac{1}{T}\sum_{t=1}^{T} g(x_t, y_t) + \delta_T^{\mathcal{X}} \geq \frac{1}{T} \max_{x \in \mathcal{X}} \sum_{t=1}^{T} g(x, y_t) \geq \max_{x \in \mathcal{X}} g\left(x, \frac{1}{T}\sum_{t=1}^{T} y_t\right) \geq \min_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} g(x, y),$$

where the second and third inequalities use convexity of g with respect to y and definition of $\min$, respectively. Combining yields

$$\min_{y \in \mathcal{Y}} g\left(\frac{1}{T}\sum_{t=1}^{T} x_t, y\right) \geq \min_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} g(x, y) - \delta_T^{\mathcal{X}},$$

and also

$$\max_{x \in \mathcal{X}} g\left(x, \frac{1}{T}\sum_{t=1}^{T} y_t\right) \leq \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y) + \delta_T^{\mathcal{X}},$$

completing the proof.

C Proof of Theorem 3.3

Let v be the value of the game in Eq. (6):

$$v = \min_{\mu \in \Delta(\Pi)} \operatorname{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}), \tag{24}$$

and let $\ell_t(\boldsymbol{\lambda}) = -\boldsymbol{\lambda} \cdot \hat{\mathbf{z}}_t$ (i.e., the loss function that OGD observes).

Lemma C.1. *For $t = 1, 2, \ldots, T$ we have*

$$\ell_t(\boldsymbol{\lambda}_t) = -\boldsymbol{\lambda}_t \cdot \hat{\mathbf{z}}_t \geq -v - (\epsilon_0 + \epsilon_1).$$

Proof. By Eq. (4) (which must hold by Lemma 3.2), and by Eq. (24), there exists $\mu^* \in \Delta(\Pi)$ such that

$$v = \operatorname{dist}(\overline{\mathbf{z}}(\mu^*), \mathcal{C}) = \max_{\boldsymbol{\lambda} \in \Lambda} \boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\mu^*).$$

Thus, $\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\mu^*) \leq v$ since $\boldsymbol{\lambda}_t \in \Lambda$ for all t. By our assumed guarantee for the policy π_t returned by the planning oracle, we have

$$-\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\pi_t) \geq -\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\mu^*) - \epsilon_0 \geq -v - \epsilon_0.$$

Now using the error bound of the estimation oracle,

$$\|\overline{\mathbf{z}}(\pi_t) - \hat{\mathbf{z}}_t\|_2 \leq \epsilon_1, \tag{25}$$

and the fact that $\|\boldsymbol{\lambda}_t\|_2 \leq 1$, we have

$$(-\boldsymbol{\lambda}_t \cdot \hat{\mathbf{z}}_t) + \epsilon_1 \geq -\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\pi_t).$$

Combining completes the proof. □

Now we are ready to prove Theorem 3.3. Using the definition of mixed policy $\bar{\mu}$ returned by the algorithm we have

$$\begin{aligned}
\operatorname{dist}(\overline{\mathbf{z}}(\bar{\mu}), \mathcal{C}) &= \operatorname{dist}\left(\frac{1}{T}\sum_{t=1}^{T} \overline{\mathbf{z}}(\pi_t), \mathcal{C}\right) \\
&= \max_{\boldsymbol{\lambda} \in \Lambda} \boldsymbol{\lambda} \cdot \left(\frac{1}{T}\sum_{t=1}^{T} \overline{\mathbf{z}}(\pi_t)\right) && (26) \\
&= \frac{1}{T} \max_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \boldsymbol{\lambda} \cdot \overline{\mathbf{z}}(\pi_t) \\
&\leq \frac{1}{T} \max_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \boldsymbol{\lambda} \cdot \hat{\mathbf{z}}_t + \epsilon_1 && (27) \\
&= -\frac{1}{T} \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}) + \epsilon_1 && (28) \\
&\leq -\frac{1}{T} \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}) + \epsilon_1 + \frac{1}{T}\sum_{t=1}^{T} (\ell_t(\boldsymbol{\lambda}_t) + \epsilon_1 + \epsilon_0 + v) && (29) \\
&= v + \left(-\frac{1}{T} \min_{\boldsymbol{\lambda} \in \Lambda} \sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}) + \frac{1}{T}\sum_{t=1}^{T} \ell_t(\boldsymbol{\lambda}_t)\right) + 2\epsilon_1 + \epsilon_0 \\
&= v + \frac{\operatorname{Regret}_T(\mathrm{OGD})}{T} + 2\epsilon_1 + \epsilon_0.
\end{aligned}$$

Here, Eq. (26) is by Eq. (4). Eq. (27) uses Eq. (25) and the fact that $\|\boldsymbol{\lambda}\|_2 \leq 1$. Eq. (30) uses Lemma C.1.

The diameter of decision set $\Lambda = \mathcal{C}^\circ \cap \mathcal{B}(1)$ is at most 1. The gradient of the loss function $\nabla(\ell_t(\boldsymbol{\lambda})) = -\hat{\mathbf{z}}_t$ has norm at most $\|\overline{\mathbf{z}}(\pi_t)\|_2 + \epsilon_1 \leq \frac{B}{1-\gamma} + \epsilon_1$. Therefore, setting $\eta = \left(\left(\frac{B}{1-\gamma} + \epsilon_1\right)\sqrt{T}\right)^{-1}$ based on Theorem A.1, we get

$$\frac{\operatorname{Regret}_T(\mathrm{OGD})}{T} \leq \left(\frac{B}{1-\gamma} + \epsilon_1\right) T^{-1/2}$$

D APPROPO for feasibility

Algorithm 4 APPROPO- Feasibility

input projection oracle $\Gamma_{\mathcal{C}}(\cdot)$ for target set $\mathcal{C}$ which is a convex cone,
positive response oracle $\mathrm{PosPlan}(\cdot)$, estimation oracle $\mathrm{Est}(\cdot)$,
step size η, number of iterations T
let $\Lambda := \mathcal{C}^\circ \cap \mathcal{B}(1)$, and define its projection operator $\Gamma_\Lambda(\mathbf{x}) \triangleq (\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x}))/\max\{1, \|\mathbf{x} - \Gamma_{\mathcal{C}}(\mathbf{x})\|_2\}$
initialize $\boldsymbol{\lambda}_1$ arbitrarily in Λ
for $t = 1$ **to** T **do**
  Call positive response oracle for the standard RL with scalar reward $r = -\boldsymbol{\lambda}_t \cdot \mathbf{z}$:
  $\pi_t \leftarrow \mathrm{PosPlan}(\boldsymbol{\lambda}_t)$
  Call the estimation oracle to approximate long-term measurement for π_t:
  $\hat{\mathbf{z}}_t \leftarrow \mathrm{Est}(\pi_t)$
  Update using online gradient descent with the loss function $\ell_t(\boldsymbol{\lambda}) = -\boldsymbol{\lambda} \cdot \hat{\mathbf{z}}_t$:
  $\boldsymbol{\lambda}_{t+1} \leftarrow \Gamma_\Lambda\big(\boldsymbol{\lambda}_t + \eta \hat{\mathbf{z}}_t\big)$
  if $\ell_t(\boldsymbol{\lambda}_t) < -(\epsilon_0 + \epsilon_1)$ **then**
    return *problem is infeasible*
  end if
end for
return $\bar{\mu}$, a uniform mixture over $\pi_1, \ldots, \pi_T$

D.1 Proof of Theorem 3.4

Lemma D.1. *If the problem is feasible, then for $t = 1, 2, \ldots, T$ we have*

$$\ell_t(\boldsymbol{\lambda}_t) = -\boldsymbol{\lambda}_t \cdot \hat{\mathbf{z}}_t \geq -(\epsilon_0 + \epsilon_1).$$

Proof. If the problem is feasible, then there exists μ^* such that $\overline{\mathbf{z}}(\mu^*) \in \mathcal{C}$. Since all $\boldsymbol{\lambda}_t \in \mathcal{C}^\circ$, they all have non-positive inner product with every point in $\mathcal{C}$ including $\overline{\mathbf{z}}(\mu^*)$. Since $-\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\mu^*) \geq 0$, we can conclude that $\max_{\pi \in \Pi} R(\pi) = \max_{\pi \in \Pi} -\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\pi) \geq 0$. Therefore, by our guarantee for the positive response oracle,

$$R(\pi_t) = -\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\pi) \geq -\epsilon_0.$$

Now using Eq. (25) and the fact that $\|\boldsymbol{\lambda}_t\|_2 \leq 1$, we have

$$(-\boldsymbol{\lambda}_t \cdot \hat{\mathbf{z}}_t) + \epsilon_1 \geq -\boldsymbol{\lambda}_t \cdot \overline{\mathbf{z}}(\pi_t).$$

Combining completes the proof. □The proof of Theorem 3.4 is similar to that of Theorem 3.3. If the algorithm reports infeasibility then the problem is infeasible as a result of Lemma D.1. Otherwise, we have

$$\frac{1}{T} \sum_{t=1}^{T} (\ell_t(\boldsymbol{\lambda}_t) + \epsilon_1 + \epsilon_0) \geq 0,$$

which can be combined with Eq. (28) as before. Continuing this argument as before yields

$$\mathrm{dist}(\overline{\mathbf{z}}(\mu), \mathcal{C}) \leq \left(\frac{B}{1-\gamma} + \epsilon_1 \right) T^{-1/2} + 2\epsilon_1 + \epsilon_0,$$

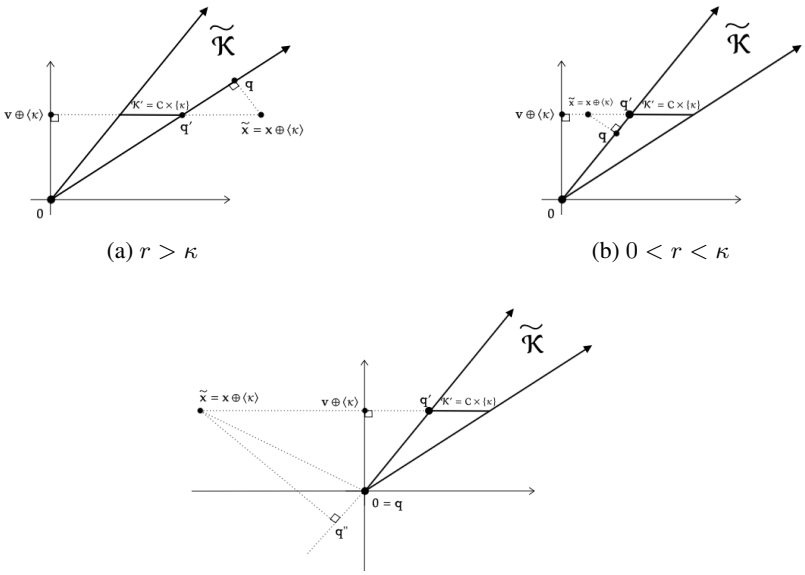
completing the proof.

E Proof of Lemma 3.5

Let $\mathcal{K}' = \mathcal{K} \times \{\kappa\}$ and $\mathbf{q}$ be the projection of $\tilde{\mathbf{x}} = \mathbf{x} \oplus \langle \kappa \rangle$ onto $\tilde{\mathcal{K}} = \mathrm{cone}(\mathcal{K}')$, i.e.,

$$\mathbf{q} = \arg\min_{\mathbf{y} \in \tilde{\mathcal{K}}} \|\tilde{\mathbf{x}} - \mathbf{y}\|_2.$$

Let r be the last coordinate of $\mathbf{q}$. We prove the lemma in cases based on the value of r (which cannot be negative by construction).



(a) $r > \kappa$ (b) $0 < r < \kappa$

(c) $r = 0$

Figure 2: Geometric Interpretation of the proof of Lemma 3.5

Case 1 ($r > \kappa$)**:** Since $\mathbf{q} \in \text{cone}(\mathcal{K}')$ with $r > 0$, there exists $\alpha > 0$ and $\mathbf{q}' \in \mathcal{K}'$ so that $\mathbf{q} = \alpha\mathbf{q}'$. See Figure 2a. Consider the plane defined by the three points $\tilde{\mathbf{x}}, \mathbf{q}, \mathbf{q}'$. Since the origin $\mathbf{0}$ is on the line passing through $\mathbf{q}$ and $\mathbf{q}'$, it must also be in this plane. Now consider the line that passes through $\tilde{\mathbf{x}}$ and $\mathbf{q}'$. Note that all points on this line have last coordinate equal to κ, and they are all also in the aforementioned plane. Let $\mathbf{v} \oplus \langle\kappa\rangle$ be the projection of $\mathbf{0}$ onto this line ($\mathbf{v} \in \mathbb{R}^d$).

Note that the two triangles $\Delta(\tilde{\mathbf{x}}, \mathbf{q}, \mathbf{q}')$ and $\Delta(\mathbf{0}, \mathbf{v} \oplus \langle\kappa\rangle, \mathbf{q}')$ are similar since they are right triangles with opposite angles at $\mathbf{q}'$. Therefore, by triangle similarity,

$$\frac{\|\mathbf{q}'\|_2}{\|\mathbf{v} \oplus \langle\kappa\rangle\|_2} = \frac{\|\tilde{\mathbf{x}} - \mathbf{q}'\|_2}{\|\tilde{\mathbf{x}} - \mathbf{q}\|_2} \geq \frac{\text{dist}(\tilde{\mathbf{x}}, \mathcal{K}')}{\text{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}})} = \frac{\text{dist}(\mathbf{x}, \mathcal{K})}{\text{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}})}.$$

Since $\mathbf{q}' \in \mathcal{K}'$, we have $\|\mathbf{q}'\|_2 \leq \sqrt{(\max_{\mathbf{x} \in \mathcal{K}} \|\mathbf{x}\|_2)^2 + \kappa^2}$, resulting in

$$\frac{\|\mathbf{q}'\|_2}{\|\mathbf{v} \oplus \langle\kappa\rangle\|_2} \leq \frac{\sqrt{(\max_{\mathbf{x} \in \mathcal{K}} \|\mathbf{x}\|_2)^2 + \kappa^2}}{\kappa} = \sqrt{1 + 2\delta} \leq 1 + \delta$$

by the choice of κ given in the lemma. Combining completes the proof for this case.

Case 2 ($r = \kappa$)**:** Since $\mathbf{q} \in \text{cone}(\mathcal{K}')$ with κ as last coordinate, we have $\mathbf{q} \in \mathcal{K}'$. Thus,

$$\text{dist}(\mathbf{x}, \mathcal{K}) = \text{dist}(\tilde{\mathbf{x}}, \mathcal{K}') \leq \|\tilde{\mathbf{x}} - \mathbf{q}\|_2 = \text{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}})$$

which completes the proof for this case.

Case 3 ($0 < r < \kappa$)**:** The proof for this case is formally identical to that of Case 1, except that, in this case, the two triangles $\Delta(\tilde{\mathbf{x}}, \mathbf{q}, \mathbf{q}')$ and $\Delta(\mathbf{0}, \mathbf{v} \oplus \langle\kappa\rangle, \mathbf{q}')$ are now similar as a result of being right triangles with a shared angle at $\mathbf{q}'$. See Figure 2b.

Case 4 ($r = 0$)**:** Since $\mathbf{q} \in \text{cone}(\mathcal{K}')$, $\mathbf{q}$ must have been generated by multiplying some $\alpha \geq 0$ by some point in $\mathcal{K}'$. Since all points in $\mathcal{K}'$ have last coordinate equal to $\kappa > 0$, and since $r = 0$, it must be the case that $\alpha = 0$, and thus, $\mathbf{q} = \mathbf{0}$. Let $\mathbf{q}'$ be the projection of $\tilde{\mathbf{x}}$ onto $\mathcal{K}'$. See Figure 2c. Consider the plane defined by the three points $\tilde{\mathbf{x}}, \mathbf{q} = \mathbf{0}, \mathbf{q}'$. Let $\mathbf{q}''$ be the projection of $\tilde{\mathbf{x}}$ onto the line passing through $\mathbf{q}$ and $\mathbf{q}'$. Then

$$\|\tilde{\mathbf{x}} - \mathbf{q}''\|_2 \leq \|\tilde{\mathbf{x}}\|_2 = \text{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}}).$$

Now consider the line passing through $\tilde{\mathbf{x}}$ and $\mathbf{q}'$. Note that all points on this line have last coordinate equal to κ and are also in the aforementioned plane. Let $\mathbf{v} \oplus \langle \kappa \rangle$ be the projection of $\mathbf{0}$ onto this line ($\mathbf{v} \in \mathbb{R}^d$). Note that the two triangles $\Delta(\tilde{\mathbf{x}}, \mathbf{q}'', \mathbf{q}')$ and $\Delta(\mathbf{0}, \mathbf{v} \oplus \langle \kappa \rangle, \mathbf{q}')$ are similar since they are right triangles with a shared angle at $\mathbf{q}'$. Therefore, by triangle similarity,

$$\frac{\|\mathbf{q}'\|_2}{\|\mathbf{v} \oplus \langle \kappa \rangle\|_2} = \frac{\|\tilde{\mathbf{x}} - \mathbf{q}'\|_2}{\|\tilde{\mathbf{x}} - \mathbf{q}''\|_2} \geq \frac{\operatorname{dist}(\tilde{\mathbf{x}}, \mathcal{K}')}{\operatorname{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}})} = \frac{\operatorname{dist}(\mathbf{x}, \mathcal{K})}{\operatorname{dist}(\tilde{\mathbf{x}}, \tilde{\mathcal{K}})}.$$

The rest of the proof for this case is exactly as in Case 1.