Apprenticeship Learning via Frank-Wolfe

Tom Zahavy, Alon Cohen, Haim Kaplan and Yishay Mansour
Google research
tomzahavy@gmail.com

Abstract

We consider the applications of the Frank-Wolfe (FW) algorithm for Apprenticeship Learning (AL). In this setting, there is a Markov Decision Process (MDP), but the reward function is not given explicitly. Instead, there is an expert that acts according to some policy, and the goal is to find a policy whose feature expectations are closest to those of the expert policy. We formulate this problem as finding the projection of the feature expectations of the expert on the feature expectations polytope – the convex hull of the feature expectations of all the deterministic policies in the MDP. We show that this formulation is equivalent to the AL objective and that solving this problem using the FW algorithm is equivalent to the most known AL algorithm, the projection method of Abbeel and Ng [2004]. This insight allows us to analyze AL with tools from the convex optimization literature and to derive tighter bounds on AL. Specifically, we show that a variation of the FW method that is based on taking "away steps" achieves a linear rate of convergence when applied to AL. We also show experimentally that this version outperforms the FW baseline. To the best of our knowledge, this is the first work that shows linear convergence rates for AL.

1 Introduction

We consider sequential decision making in the Markov decision process (MDP) formalism. Given an MDP, the optimal policy and its value function are characterized by the Bellman equations and can be computed via value or policy iteration. This makes the MDP model useful in problems where we can specify the MDP model (states, actions, reward, transitions) appropriately. However, in many real-world problems, it is often hard to define a reward function, such that the optimal policy with respect to this reward produces the desired behavior.

In Apprenticeship Learning (AL), instead of manually tweaking the reward to produce the desired behavior, the idea is to observe and mimic an expert. The literature on AL is quite vast and dates back to the work of Abbeel and Ng [2004], who proposed a novel framework for AL. In this setting, the reward function (while unknown to the apprentice) equals to a linear combination of a set of known features. More specifically, there is a weight vector w. The rewards are associated with states, and each state s has a feature vector $\phi(s)$, and its reward is $\phi(s) \cdot w$. The expected return of a policy π is $V^\pi = \Phi(\pi) \cdot w$, where $\Phi(\pi)$ is the feature expectation under policy π. The expert demonstrates a set of trajectories that are used to estimate the feature expectations of its policy π_E, denoted by $\Phi_E \triangleq \Phi(\pi_E)$. The goal is to find a policy ψ, whose feature expectations are close to this estimate, and hence will have a similar return with respect to any weight vector w.

Abbeel and Ng [2004] suggested two algorithms to solve this problem, one that is based on a maximum margin solver and a simpler projection algorithm. The algorithm starts with an arbitrary policy π_0 and computes its feature expectation $\Phi(\pi_0)$. At step t they define a reward function using weight vector $w_t = \Phi_E - \bar{\Phi}_{t-1}$ and find the policy π_t that maximizes it, where $\bar{\Phi}_t$ is a convex combination of feature expectations of previous (deterministic) policies $\bar{\Phi}_t = \sum_{j=1}^{t} \alpha_j \Phi(\pi_j)$. They show that in order to get that $\|\bar{\Phi}_T - \Phi_E\| \leq \epsilon$, it suffices to run the algorithm for $T = O(\frac{k}{(1-\gamma)^2\epsilon^2} \log(\frac{k}{(1-\gamma)\epsilon}))$ iterations.

In this work, we focus on the computational complexity of the problem as a function of ϵ. We show that a small modification to the algorithm of Abbeel and Ng [2004] can lead to a **linear rate of**

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

convergence, i.e., $T = O(\log(1/\epsilon))$. Methods that are based on online convex optimization (like [Syed and Schapire, 2008]), on the other hand, cannot achieve rates better than $T = O(1/\epsilon^2)$ (the O notation hides the dependency in k and γ).

Our result is based on the observation that (a slight modification) of the algorithm of Abbeel and Ng [2004] is, in fact, an instantiation of the Frank-Wolfe (FW) method – a projection free method for convex optimization. To see this, we formulate the AL problem as finding the projection of the (estimated) feature expectations of the expert on the feature expectations polytope — the convex hull of the feature expectations of all the deterministic policies in the MDP (Definition 3). To compute this polytope (and to project to it), one has to compute the feature expectations of the exponentially ($|A|^{|S|}$) many deterministic policies in the MDP. The benefit in applying the FW method to this problem is that it avoids projecting to this polytope (as in projected gradient methods); instead, it minimizes a linear objective function over the polytope, which is equivalent to finding the optimal policy in an MDP. The observation that the algorithm of Abbeel and Ng [2004] is an instantiation of the Frank-Wolfe (FW) method allows to derive the convergence result of Abbeel and Ng [2004] immediately (even with a logarithmic factor improvement) from known analysis of the FW method.

Furthermore, this equivalence leads us to propose a modification to this Frank-Wolfe AL algorithm that is based on taking "away steps." These steps try to remove weight from "bad policies" (policies that were added to the solution in previous iterations but now by removing them we get an improvement). This modification gives the first AL algorithm with a linear rate of convergence. We implemented this algorithm and compared it with the method of Abbeel and Ng [2004]. Our findings suggest that "away steps" indeed give a better empirical performance.

Due to space considerations, related work, experiments, and a lemma on estimating the expert from samples can be found in the supplementary material.

2 Preliminaries

In this section, we provide the relevant background on convex optimization, apprenticeship learning, and the Frank-Wolfe algorithm. In convex analysis, we are interested in solving problems of the form $\min h(\mathbf{x})$ where $x \in \mathcal{K}$ is a convex set and h is a convex function. We briefly define important properties of convex functions and convex sets in the supplementary material. We use on the Euclidean norm in this paper. We will focus on a specific convex optimization problem: finding a particular Euclidean projection on a convex set.

Definition 1 (Euclidean projection)**.** The Euclidean projection onto a convex set $\mathcal{K}$ is given by $\text{Proj}_{\mathcal{K}}(x) = \arg\min_{y \in \mathcal{K}} ||x - y||$.

2.1 Inverse reinforcement learning and apprenticeship learning

For consistency with prior work, we consider the discounted infinite horizon scenario. We emphasize here that all the results in this paper can be easily extended to the episodic finite horizon and the average reward criteria. We are given an MDP\R, (MDP without a reward) denoted $M \triangleq \{S, A, P, \gamma, D\}$, where S is the set of states, A is the set of actions, $P = \{P^a \mid a \in A\}$ is the set of transition matrices, γ is the discount factor, and D is the distribution of the initial state.

Each state s is represented by an observable low-dimensional vector of features $\phi(s) \in [0,1]^k$, and the reward function, while unknown to the apprentice, is assumed to be equal to a linear combination of the features; i.e., $r_w(s) = w \cdot \phi(s)$, for some $w \in \mathcal{W}$ where $\mathcal{W}$ is a convex set. For example, $\mathcal{W}$ can be chosen to be the simplex Syed and Schapire [2008], or the L_2 ball Abbeel and Ng [2004]. We further assume the existence of an *expert policy*, denoted by π_E, such that we can observe its execution in M. We define the feature expectations of a policy π in M as$\Phi(\pi) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi, P, D\right]$. With this feature representation, the value of a policy π may be written as $V^\pi = w \cdot \Phi(\pi)$. In addition, the feature expectations are bounded: $||\Phi(\pi)||_\infty \leq 1/(1-\gamma)$.

Like Abbeel and Ng [2004], and Syed and Schapire [2008], the policy that we find is not necessarily deterministic, but a mixed policy. A mixed policy ψ is a distribution over Π, the set of all deterministic policies in M. Because Π is finite (though extremely large), we can fix an ordering $\pi_1, \pi_2, \ldots$ of the policies in Π. This allows us to treat ψ as a vector, where $\psi(i)$ is the probability assigned to π_i. A mixed policy ψ is executed by randomly selecting the policy $\pi_i \in \Pi$ at time 0 with probability $\psi(i)$, and exclusively following π_i thereafter. The definitions of the value function and the feature expectations are naturally extended to mixed policies as follows: $V(\psi) = \mathbb{E}_{i \sim \psi} V^{\pi_i}$ and $\Phi(\psi) = \mathbb{E}_{i \sim \psi} \Phi(\pi_i)$. The objective of AL is to find a policy π that does at least as well as the expert with respect to any reward function of the form $r(s) = w \cdot \phi(s), w \in \mathcal{W}$. That is we solve

$$\max_{\psi \in \Psi} \min_{w \in \mathcal{W}} [w \cdot \Phi(\psi) - w \cdot \Phi_E] \tag{1}$$

If we denote the value of Eq. (1) by f^*, then, due to the von-Neumann minimax theorem we also have that $f^* = \min_{w \in \mathcal{W}} \max_{\psi \in \Psi} [w \cdot \Phi(\psi) - w \cdot \Phi_E]$. We will refer to approximately solving the last equation as IRL, i.e., finding $w \in \mathcal{W}$ such that $\forall \psi \in \Psi : w \cdot \Phi_E \geq w \cdot \Phi(\psi) - \epsilon - f^*$; and to the problem of approximately solving Eq. (1) as AL, i.e., finding ψ such that

$$\forall w \in \mathcal{W} : w \cdot \Phi(\psi) \geq w \cdot \Phi_E + f^* - \epsilon, \tag{2}$$

Notice that due to [Syed et al., 2008, Theorem 3], it is equivalent to solve AL over Π and Ψ (the sets of deterministic and mixed policies).

The most famous AL algorithm for solving Eq. (2) is the projection algorithm of Abbeel and Ng [2004] (Algorithm 1). Notice that we slightly changed the notation and the order of indices in Algorithm 1 w.r.t [Abbeel and Ng, 2004]; it is immediate to verify that these algorithms are equivalent. In addition, line 8 in Algorithm 1 was not part of the original algorithm. The role of this step is to replace the post-processing procedure in [Abbeel and Ng, 2004] by maintaining a policy ψ_t with feature expectations $\bar{\Phi}_t$ (in Section 3, we show explicitly that $\Phi(\psi_t) = \bar{\Phi}_t$). Somewhat ironically, Abbeel and Ng [2004] termed their algorithm the "projection algorithm", while we will soon see that it is actually a projection-free algorithm (CG method) with respect to the feature expectations polytope.

Algorithm 1 The projection method Abbeel and Ng [2004]

1: Input: feature expectations of the expert Φ_E, T number of iterations
2: Initialize: choose π_0, set $\psi_0 = e_{\pi_0}, \bar{\Phi}_0 = \Phi(\psi_0)$
3: **for** $t = 1, \ldots, T$ **do**
4: Set $w_t = \Phi_E - \bar{\Phi}_{t-1}$
5: Compute $\pi_t = \pi^*_{w_t}, \Phi_t = \Phi(\pi_t)$
6: $\alpha_t = \frac{(\Phi_t - \bar{\Phi}_{t-1}) \cdot (\Phi_E - \bar{\Phi}_{t-1})}{(\Phi_t - \bar{\Phi}_{t-1}) \cdot (\Phi_t - \bar{\Phi}_{t-1})}$
7: $\bar{\Phi}_t = \bar{\Phi}_{t-1} + \alpha_t(\Phi_t - \bar{\Phi}_{t-1})$
8: $\psi_t = \psi_{t-1} + \alpha_t(e_{\pi_t} - \psi_{t-1})$
9: **end for**
10: Return ψ_T

The algorithm begins by estimating the feature expectation Φ_0 of some arbitrary policy π_0. Then, for iterations $t = 1, \ldots, T$ it finds the optimal policy π_t w.r.t reward $w_t = \Phi_E - \bar{\Phi}_{t-1}$. The feature expectation Φ_t of the policy π_t are computed and added to the solution $\bar{\Phi}_t$, such that $\bar{\Phi}_t = \bar{\Phi}_{t-1} + \alpha_t(\Phi_t - \bar{\Phi}_{t-1})$. The parameter α_t is chosen via a line search, i.e., $\alpha_t = \min_\alpha \|\bar{\Phi}_{t-1} + \alpha(\Phi_t - \bar{\Phi}_{t-1}) - \Phi_E\|^2$. For ψ to be a mixed policy, α_t must be in the range $[0, 1]$. In the case that Φ_E is given exactly, it is guaranteed that $\alpha_t \in [0, 1]$. When Φ_E is estimated from samples, for ψ to be a mixed policy, α_t must be truncated to $[0, 1]$.Abbeel and Ng [2004] proved directly that the features expectations of $\bar{\Phi}_t$ converge to the features of the expert.

2.2 The conditional gradient (CG) method

A common algorithm to minimize a convex function over a convex set $\mathcal{K}$ is projected gradient descent. This algorithm takes a step in the reverse gradient direction $z_{t+1} = x_t + \alpha_t \nabla_h(x_t)$, and then projects z_{t+1} back into $\mathcal{K}$ to obtain x_{t+1}. Computing this projection may be expensive for some convex sets. The CG algorithm of Frank and Wolfe [1956] (Algorithm 2) avoids this projection. It finds a point $y_t \in \mathcal{K}$ that has the largest correlation with the negative gradient, and updates $x_{t+1} = (1 - \alpha_t)x_t + \alpha_t y_t$, which by convexity guarantees to be in $\mathcal{K}$.

Algorithm 2 The CG method Frank and Wolfe [1956]

1: Input: a convex set $\mathcal{K}$, a convex function h, learning rate schedule α_t.
2: Initiation: let $x_0 \in \mathcal{K}$
3: **for** $t = 1, \ldots, T$ **do**
4: $y_t = \arg\min_{y \in \mathcal{K}} \nabla_h(x_{t-1}) \cdot y$
5: $x_t = (1 - \alpha_t)x_{t-1} + \alpha_t y_t$
6: **end for**

To find y_t, the algorithm has to minimize a linear objective function over the feasible set $\mathcal{K}$. We assume that this optimization is performed by an oracle (which we call *linear-oracle*). If $\mathcal{K}$ is a polyhedron (given by its facets), then an oracle call is a linear programming problem. The CG method is useful for problems where implementing such a linear-oracle is easier than computing a projection onto $\mathcal{K}$.

To give some context, in AL, the linear-oracle will be an algorithm that finds the optimal policy in an MDP with known reward and dynamics, e.g., Policy Iteration (PI). The polyhedral set will be the set of feature expectations of all the deterministic policies in this MDP, which is of size $|A|^{|S|}$. The computational complexity of computing this set explicitly (and hence projecting onto it) is therefore exponential in the size of the state space. On the other hand, it is known that PI converges to the optimal policy in a finite number of iterations [Puterman, 1984, Theorem 8.6.6]. A trivial upper bound on the number of iterations is the total number of deterministic policies, which is $|A|^{|S|}$. In discounted MDPs, it was shown that PI runs in strongly polynomial time [Ye, 2011]; therefore the CG algorithm has a computational advantage over projection-based algorithms.

The original paper of Frank and Wolfe contains a proof of an $O(1/t)$ rate of convergence (Theorem 1, extracted from Jaggi [2013]). Canon and Cullum [1968] prove that for functions that are not strongly convex, this rate is tight.

Theorem 1. *Let h be a convex and β-smooth function. Let $D_{\mathcal{K}}$ be the diameter of $\mathcal{K}$, and let $\alpha_t = \frac{2}{t+1}$ for $t \geq 1$. Then for any $t \geq 2$, (Algorithm 2) computes x_t such that $h(x_t) - h(x^*) \leq \frac{2\beta D_{\mathcal{K}}^2}{t+1}$, where x^* is a minimizer of h over $\mathcal{K}$.*

Fast rates. In this paper, we focus on minimizing a strongly convex function. In this case, if the optimal solution is in the interior of the feasible set, then CG converges in a linear rate Beck and Teboulle [2004], Guélat and Marcotte [1986]. Another setting in which a faster rate of convergence can be derived is when the feasible set is strongly convex. (A strongly convex set is a set where each convex combination of two points in the set is in the interior of the set.) In this case, the convergence rate is $O(1/t^2)$ Garber and Hazan [2015]. Alternatively, if the norm of the gradient of the objective function is bounded away from zero everywhere in $\mathcal{K}$, then the rate of convergence is linear [Levitin and Polyak, 1966] (even if the objective is only convex and not strongly convex). Unfortunately, for reasons that we will see later on, none of these cases is relevant for AL. A different approach to speed up the convergence is to modify the algorithm, as we describe next.

2.3 Frank-Wolfe with away steps (ASCG)

Away steps conditional gradient (ASCG) is a variation of the CG method, proposed by Wolfe [1970] for polyhedral sets. By Carathéodory theorem, the iterate x_t can always be represented as a sparse convex combination of at most $k+1$ vertices of $\mathcal{K}$, i.e., $x_t = \sum_{i=1}^{k+1} \alpha_{y_i} y_i$. ASCG uses this fact and removes weight from "bad" elements y_i (not needed to represent the final solution) by taking "away steps." These steps decrease the weight of the "bad" elements faster then they would have decayed via the standard CG iterates. Explicitly, ASCG (Algorithm 3) maintains the list of vertices $S^{(t)} = \{y_{i_1}, \ldots, y_{i_{\ell_t}}\}$, where t is the iteration index, $\ell_t = |S^{(t)}|$, and $i_j \leq t$ for every $j = 1, \ldots, \ell_t$, and a corresponding list of coefficients $\{\alpha_{y_{i_j}}\}_{j=1}^{\ell_t}$ such that $x_t = \sum_{j=1}^{\ell_t} \alpha_{y_{i_j}} y_{i_j}$.

Algorithm 3 Frank-Wolfe with away steps (ASCG)

Input: a convex set $\mathcal{L}$, and a convex function h
Initiation: let $x_1 \in \mathcal{K}$, $S^{(1)} = \{x_1\}$, $\alpha_{x_1} = 1$
for $t = 1, \ldots, T$ **do**
  $y_t = \arg\max_{y \in \mathcal{K}} -\nabla_h(x_t) \cdot y$, $d^{FW} = y_t - x_t$
  $z_t = \arg\max_{z \in S^{(k)}} \nabla_h(x_t) \cdot z$, $d^{AS} = x_t - z_t$
  if $\nabla_h(x_t) \cdot d^{FW} < \nabla_h(x_t) \cdot d^{AS}$ **then**
    Frank-Wolfe step: $d = d^{FW}$, $\gamma_{\max} = 1$
  else
    Away step: $d = d^{AS}$, $\gamma_{\max} = \alpha_{z_t}/(1 - \alpha_{z_t})$
  end if
  Line-search: $\gamma_t = \arg\min_{\gamma \in [0, \gamma_{\max}]} h(x_t + \gamma d)$
  Update: $x_{t+1} = x_t + \gamma_t d$
  Update representation:
  if Frank-Wolfe step **then**
    if $(\gamma_t = 1)$ **then**
      $S^{(t+1)} = \{y_t\}, \alpha_{y_t} = 1$
    else
      $\alpha_{y_t} = (1 - \gamma_t)\alpha_{y_t} + \gamma_t$
      $\forall y \in S^{(t)} : \alpha_y = (1 - \gamma_t)\alpha_y$
      $S^{(t+1)} = S^{(t)} \cup \{y_t\}$
    end if
  else if Away step **then**
    if $(\gamma_t = \gamma_{\max})$ **then**
      Drop step: $S^{(t+1)} = S^{(t)} \setminus \{z_t\}$
    else
      $a_{z_t} = (1 - \gamma_t)\alpha_{z_t} - \gamma_t$
      $\forall y \in S^{(t)} : \alpha_y = (1 + \gamma_t)\alpha_y$
      $S^{(t+1)} = S^{(t)}$
    end if
  end if
end for

At each iteration, the algorithm computes a regular CG step (d^{FW}); In addition, it checks the possibility of decreasing the coefficient α_{z_t} of some $z_t \in S^{(t)}$ in the representation of x_t as a convex combination of $S^{(t)}$ by taking a so-called "away step" in the direction $d^{AS} = x_t - z_t$. The z_t that has the largest correlation with the gradient is chosen, and the learning rate is set via a line search procedure. In addition, it is guaranteed that x_{t+1} remains in $\mathcal{K}$. Once the step is taken the coefficients of the remaining members in S are updated such that their combination remains convex (all coefficients are positive and sum to 1). As a result, members in S that are not part of the solution are removed as their coefficient decreases to 0.

In contrast to CG, ASCG maintains the coefficients of x_t as a convex combination of the vertices in $S^{(t)}$ explicitly. This is required to guarantee that the learning rate of the away step is chosen such that x_{t+1} remains in $\mathcal{K}$. In general, the size of the list $S^{(t)}$ at time t is bounded by t, however, we know that x_{t+1} can be written as a convex combination of at most $k+1$ points in $\mathcal{K}$. Beck and Shtern [2017] propose an improved update representation procedure, based on the Carathéodory theorem, that guarantees that $S^{(t)}$ is of size at most $k+1$ for all t.

Guélat and Marcotte [1986] were the first to suggest that ASCG attains a linear rate of convergence when the set is a polytope. Garber and Hazan [2013] provided the first official proof that a variant of CG (that is similar to ASCG) convergence in linear rate; Jaggi [2013] proved this for ASCG.

Theorem 2 below, due to Lacoste-Julien and Jaggi [2014], specifies the convergence rate of ASCG in terms of a constant $C(\mathcal{K})$ called the *pyramidal width* of $\mathcal{K}$ that depends on the geometry of $\mathcal{K}$. Here we will use a characterization of $C(\mathcal{K})$ (which we found to be more intuitive) that is called the *facial distance* Pena and Rodriguez [2018] of $\mathcal{K}$.

Definition 2 (The facial distance, Pena and Rodriguez [2018])**.** Let A be a set of points in $\mathbb{R}^k$ and let $\mathcal{K} = \text{conv}(A)$, The facial distance of $\mathcal{K}$ is $C(\mathcal{K}) = \min_{\substack{F \in \text{faces}(\mathcal{K}) \\ 0 \notin F \notin \mathcal{K}}} \min_{\substack{u \in F \\ v \in \text{conv}(A \setminus F)}} \|u - v\|_2$.

By $\text{conv}(B)$ we denote the convex hull of the points in B, and by $\text{faces}(\mathcal{K})$ we denote the set of faces (convex hulls of sets of pairwise adjacent vertices) of the polytope $\mathcal{K}$.

Theorem 2 (Linear convergence of ASCG; [Lacoste-Julien and Jaggi, 2014])**.** *Suppose that h is a $\beta-$smooth $\sigma-$strongly convex function over a convex set with diameter $D_{\mathcal{K}}$. Then the error of ASCG decreases geometrically as $h(x_t) \leq h(x_1)\exp(-\frac{\rho t}{2})$, where $\rho = \frac{\sigma C(\mathcal{K})^2}{4D_{\mathcal{K}}^2 \beta}$.*

We remark that Garber and Hazan [2016] also give a variant of Frank-Wolfe that converges linearly on polyhedral sets when the objective function smooth and strongly convex (as it is in our case). In their result, the convergence rate is dominated by a constant different from the facial distance, that equals to the minimum distance between a vertex v and a hyperplane supporting a facet which does not contain v. Nonetheless, we conjecture that this constant is strongly related to facial distance. In this work, we focus on ASCG for two reasons: it incorporates line search, which is important in practice, and it is simpler to implement.

3 Convex formulation of AL

In this section, we further assume that $\mathcal{W}$ is the L_2 ball with a unit radius Abbeel and Ng [2004]. Since scaling of the reward by a constant does not affect the resulting policy, this assumption is without loss of generality. Thus, we can rewrite Eq. (1) as follows:

$$\max_{\psi \in \Psi} \min_{w \in \mathcal{W}} [w \cdot \Phi(\psi) - w \cdot \Phi_E] = \max_{\psi \in \Psi} -||\Phi(\psi) - \Phi_E|| = -\min_{\psi \in \Psi} ||\Phi(\psi) - \Phi_E||, \tag{3}$$

where in Eq. (3), we use the fact that a unit vector in the direction of $\Phi(\psi) - \Phi_E$ is the minimizer when $\mathcal{W}$ is the unit L_2 ball. Next, we define the feature expectations polytope $\mathcal{K}$ as the convex hull of the feature expectations of all the deterministic policies in M:

Definition 3 (The feature expectations polytope)**.** $\mathcal{K} = \left\{ x : \sum_{i=1}^{k+1} a_i \Phi(\pi_i), a_i \geq 0, \sum_{i=1}^{k+1} a_i = 1, \pi_i \in \Pi \right\}$.

It is straightforward to verify that the bounded features assumption ($|\phi(s)| \leq 1$ for all s) implies that the diameter of the polytope (see the supplementary for definition) is $D_{\mathcal{K}} = \sqrt{k}/(1-\gamma)$. Definition 3 also implies that $\forall \psi \in \Psi$, we have that $\Phi(\psi) \in \mathcal{K}$. Therefore, solving Eq. (3) is equivalent to finding the mixed policy ψ, whose feature expectations are $\text{Proj}_{\mathcal{K}}(\Phi_E)$, i.e., the euclidean projection (Definition 1) of the feature expectations of the expert onto $\mathcal{K}$ (if we know Φ_E exactly then it is in $\mathcal{K}$ but typically we have only an estimate). The challenge is that $\mathcal{K}$ has $|A|^{|S|}$ vertices (feature expectations of deterministic policies), thus, computing the projection explicitly and then finding ψ whose feature expectations are close to this projection, is computationally prohibitive. This makes the CG method appealing for solving this projection problem. In particular, since we are able to compute a mixed policy $\bar{\psi}$ whose feature expectation are equal to those of the projection (via line 8, that we added in Algorithm 1).

We are now ready to define the CG method for AL explicitly and to show that it is indeed equivalent to the projection algorithm of Abbeel and Ng [2004]. Consider the square of Eq. (3) as the objective function for the CG method, where we take the feature expectation as the argument rather than the policy ψ. I.e., we define a function h over $x \in \mathcal{K}$ as $h(x) = \frac{1}{2}\|x - \Phi_E\|^2$.

Clearly, $\nabla_h(x) = x - \Phi_E$, and therefore line 4 in Algorithm 2 is equivalent to finding the feature expectations of the optimal policy in an MDP with reward given by $w = -\nabla_h(x_t)$. It follows that lines (4-5) in Algorithm 1 are equivalent to line 4 in Algorithm 2 and line 5 in Algorithm 2 is equivalent to line 7 in Algorithm 1, if we substitute $\bar{\Phi}_t = x_t$ and $\Phi_t = y_t$.

As we already mentioned in Section 2.1, line 6 of Algorithm 1 is equivalent to setting $\alpha_t = \min_\alpha \|\bar{\Phi}_{t-1} + \alpha(\Phi_t - \bar{\Phi}_{t-1}) - \Phi_E\|^2$ (line search). For the CG method to maintain $\bar{\Phi}$ as a convex combination of feature expectations, α_t must be restricted to $[0, 1]$. This holds automatically if

$\Phi_E \in \mathcal{K}$. When $\Phi_E \notin \mathcal{K}$, (e.g., when it is estimated from samples), we should restrict the line search and set $\alpha_t = \min_{\alpha \in [0,1]} \|\bar{\Phi}_{t-1} + \alpha(\Phi_t - \bar{\Phi}_{t-1}) - \Phi_E\|^2$. We also note that by Theorem 1, we can also set $\alpha_t = \frac{2}{t+1}$ and get the same convergence rate. We focus on line search since it is known to work better empirically. Notice that $h(\Phi_t)$ is a $1-$smooth $1-$strongly convex function. In addition, it has the same unique minimizer as the original objective (Eq. (3)), which is not $\beta-$smooth. For smooth functions, CG converges at a rate of $O(D_{\mathcal{K}}^2/t) = O(k/t(1-\gamma)^2)$ (Theorem 1). Thus, after $O(k/(1-\gamma)^2\epsilon^2)$ iterations, the CG method finds an $\epsilon-$optimal solution to Eq. (3). This gives a logarithmic improvment on the result of Abbeel and Ng [2004].

4 Linear rate of convergence for AL

In the preliminaries section, we described the conditions for the CG method to achieve a linear rate of convergence. Unfortunately, as we now explain, these conditions do not hold for AL, despite the fact that $h(\Phi_t)$ is a strongly convex function. First of all, since $\mathcal{K}$ is a polytope, it is not a strongly convex set. Secondly, Φ_E cannot be guaranteed to be an interior point of $\mathcal{K}$. If π_E is an optimal policy w.r.t some reward, and Φ_E is given explicitly (and not via sampled trajectories), then it is located on the boundary of $\mathcal{K}$ and is not an interior point. It is perhaps possible to compute a direction into the interior of the set (i.e., by mixing the feature expectations of the expert policy with those of a random policy) to modify Φ_E by an $\epsilon-$small step in this direction such that it will be an interior point. The problem is that when Φ_E is approximated from samples (which is the case of interest), it is not guaranteed that it is located inside $\mathcal{K}$. Since we do not know at what distance and at what direction it from $\mathcal{K}$ it is, we cannot guarantee that an $\epsilon-$step will take us to the interior. Since CG cannot attain a linear rate of convergence for AL, we now turn to analyze ASCG for AL.

4.1 ASCG for AL

Recall that in each iteration, the ASCG algorithm chooses between two alternative steps: an FW step and an away step. The FW step finds the feature expectations of the optimal policy in an MDP whose reward is the negative gradient. This is a standard RL (planning) problem and can be solved, for example, with policy iteration. We also know that there exists at least one optimal deterministic policy for it and that PI will return a solution that is a deterministic policy. Thus, the list of elements that ASCG maintains (S) is composed of feature expectations of deterministic policies $\{\pi_1, \pi_2, \dots, \}$. Since the associated coefficients $\{\alpha_{\pi_1}, \alpha_{\pi_2}, \dots\}$ are a convex combination, the mixed policy $\psi(\pi_i) = \alpha_{\pi_i}$ is guaranteed to have the same feature expectations as $\bar{\Phi}_t$. The away step in AL checks each one of the deterministic policies in the list and tries to reduce its coefficient. If a policy that is not part of the final solution was added to the solution during the run of the algorithm, then the away step will reduce its coefficient faster then it would have decreased via FW steps.

The AL problem satisfies all the requirements in Theorem 2, thus, it attains a linear rate of convergence, and we have that $h(x_t) \leq h(x_1)\exp(-\frac{\rho t}{2})$. Since h is $1-$strongly convex, $1-$smooth, and $D_{\mathcal{K}} \leq \sqrt{k}/(1-\gamma)$, we have that $\rho = \frac{\sigma C(\mathcal{K})^2}{4D_{\mathcal{K}}^2\beta} = \frac{(1-\gamma)^2 C(\mathcal{K})^2}{4k}$. The facial distance $C(\mathcal{K})$ is defined in (Definition 2) and depends on the dynamics of the MDP and the features. Intuitively, ASCG converge faster than CG, since it chooses away steps only when they lead to steeper descent. In the experiments section, we observed that ASCG indeed enjoys faster convergence than CG in practice.

5 Discussion

We presented a convex optimization formulation for AL and showed that the CG method is equivalent to the projection algorithm of Abbeel and Ng [2004]. This revelation allowed us to leverage known results on the CG method for AL. We showed that a version of the CG method that is taking away steps gives improved performance empirically and has a provable linear rate of convergence.

We believe that our findings will help to improve AL algorithms further. One direction is to try and find a relaxation of the problem, where instead of optimizing over the polytope, we optimize over a large strongly convex set that is contained in the polytope. Such a set can be obtained, for example, by mixing each deterministic policy with a random policy. If the distance between the sets is guaranteed to be small, then, it should be possible to obtain faster rates. Another direction is to try and bound the facial distance of the polytope using parameters of the MDP. Finally, it should be possible to use a stochastic version of FW [Hazan and Kale, 2012] for AL. This will allow the agent to act after each demonstration of the expert, instead of the current scheme that first approximates the expert and only then uses it for AL.

References

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

Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming*, 164(1-2):1–27, 2017.

Amir Beck and Marc Teboulle. A conditional gradient method with linear rate of convergence for solving convex linear systems. *Mathematical Methods of Operations Research*, 59(2):235–247, 2004.

Michael D Canon and Clifton D Cullum. A tight upper bound on the rate of convergence of frank-wolfe algorithm. *SIAM Journal on Control*, 6(4):509–516, 1968.

Eyal Even-Dar and Yishay Mansour. Learning rates for q-learning. *Journal of machine learning Research*, 5(Dec):1–25, 2003.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Dan Garber and Elad Hazan. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *arXiv preprint arXiv:1301.4666*, 2013.

Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *32nd International Conference on Machine Learning, ICML 2015*, 2015.

Dan Garber and Elad Hazan. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization*, 26(3):1493–1528, 2016.

Jacques Guélat and Patrice Marcotte. Some comments on wolfe's away step. *Mathematical Programming*, 35(1):110–119, 1986.

Elad E Hazan and Satyen Kale. Projection-free online learning. In *29th International Conference on Machine Learning, ICML 2012*, pages 521–528, 2012.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. 2013.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International conference on Machine learning*, pages 267–274, 2002.

S. Lacoste-Julien and M. Jaggi. An affine invariant linear convergence analysis for frank-wolfe algorithms. *NIPS 2013 Workshop on Greedy Algorithms, Frank-Wolfe and Friends*, 2014.

Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

Arkadii Semenovich Nemirovsky and David Borisovich Yudin. In *Problem complexity and method efficiency in optimization*. Wiley, New York, 1983.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. *International Conference on Machine Learning (ICML)*, 2000.

Javier Pena and Daniel Rodriguez. Polytope conditioning and linear convergence of the frank–wolfe algorithm. *Mathematics of Operations Research*, 44(1):1–18, 2018.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1984.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

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

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039. ACM, 2008.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Philip Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pages 1–36, 1970.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

Tom Zahavy, Alon Cohen, Haim Kaplan, and Yishay Mansour. Average reward reinforcement learning with unknown mixing times. *arXiv preprint arXiv:1905.09704*, 2019.

A Definitions for convex analysis

Definition 4 (Convex set)**.** A set $\mathcal{K}$ is convex if $\forall x_1, x_2 \in \mathcal{K}, \forall \lambda \in [0,1] : \lambda x_1 + (1-\lambda)x_2 \in \mathcal{K}$.

Definition 5 (Diameter of a set)**.** The diameter of a set $\mathcal{K}$ is given by $D_{\mathcal{K}} = \max_{x_1,x_2 \in \mathcal{K}} ||x_1 - x_2||$.

Definition 6 (Convex function)**.** A function $h : \mathcal{K} \to \mathbb{R}$ is convex if $\mathcal{K}$ is a convex set and $\forall x_1, x_2 \in \mathcal{K}, \forall \lambda \in [0,1] : h(\lambda x_1 + (1-\lambda)x_2) \leq \lambda h(x_1) + (1-\lambda)h(x_2)$.

Definition 7 (Properties of convex functions)**.** A convex function h over a convex set $\mathcal{K}$, i.e., $h : \mathcal{K} \to \mathbb{R}$ w.r.t $||\cdot||$ is:

1. **Strongly convex** with strong convexity parameter $\sigma > 0$ if $\forall x_1, x_2 \in \mathcal{K} : (\nabla_h(x_1) - \nabla_h(x_2)) \cdot (x_1 - x_2) \geq \sigma ||x_1 - x_2||^2$.
2. **Smooth** with parameter β if $\forall x_1, x_2 \in \mathcal{K}, |\nabla h(x_1) - \nabla h(x_2)| \leq \beta ||x_1 - x_2||$.
3. **Lipschitz continous** with parameter L_h if $\forall x \in \mathcal{K}, \|\nabla h(x)\| \leq L_h$.

B Estimating Φ_E from samples

In most practical cases, it is unrealistic to assume that the feature expectations of the expert are given explicitly. In such cases, AL algorithms estimate the feature expectations by querying the expert for trajectories and then run AL on the estimated feature expectations. Lemma 3 bounds the number of samples needed from the expert to get a good approximation of its feature expectations.

Lemma 3. *Given m samples $\{\tau_i\}_{i=1}^m$ whose expectation is Φ_E, define the estimator $\hat{\Phi}(\pi_E) = \frac{1}{m}\sum_{i=1}^m \tau_i$. Assume that we performed AL w.r.t $\hat{\Phi}(\pi_E)$ and found a solution π such that $\|\Phi(\pi) - \hat{\Phi}(\pi_E)\| \leq \epsilon$. Then, for any ϵ_m, δ it is enough to have $m = 2k\ln(2k/\delta)/\epsilon_m^2$ samples in order to have that $\|\Phi(\pi) - \Phi_E\| \leq \epsilon + \epsilon_m$ with probability $1-\delta$.*

Proof. By Hoeffding's inequality we get that

$$\forall i \in [1,..,k] \;\; \Pr(|\hat{\Phi}_E(i) - \Phi_E(i)| \geq \epsilon) \leq 2\exp(-m\epsilon^2/2).$$

Applying the union bound over the features we get that

$$\Pr(\exists i \in [1,..,k], s.t., |\hat{\Phi}_E(i) - \Phi_E(i)| \geq \epsilon) \leq 2k\exp(-m\epsilon^2/2).$$

This is equivalent to

$$\Pr(\forall i \in [1,..,k] \;\; |\hat{\Phi}_E(i) - \Phi_E(i)| \leq \epsilon) \geq 1 - 2k\exp(-m\epsilon^2/2).$$

and to

$$\Pr(\|\hat{\Phi}_E - \Phi_E\|_\infty \leq \epsilon) \geq 1 - 2k\exp(-m\epsilon^2/2).$$

Thus, we got after collecting $m = \frac{2\ln(2k/\delta)}{(\epsilon_m/\sqrt{k})^2}$ samples of $\Phi(E)$, then with probability $1-\delta$, we have that $\|\hat{\Phi}_E - \Phi_E\|_\infty \leq \frac{\epsilon_m}{\sqrt{k}}$. Therefore, with probability $1-\delta$ we have that

$$\|\hat{\Phi}_E - \Phi_E\|_2 \leq \sqrt{k}\|\hat{\Phi}_E - \Phi_E\|_\infty \leq \epsilon_m. \tag{4}$$

Now, by the assumption, we have that $\|\Phi(\pi) - \hat{\Phi}(\pi_E)\|_2 \leq \epsilon$ and that $\|\Phi_E - \hat{\Phi}(\pi_E)\|_2 \leq \epsilon_m$. Thus, we have that with probability $1-\delta$

$$\begin{aligned}
\|\Phi(\pi) - \Phi_E\|_2 &= \|\Phi(\pi) - \hat{\Phi}(\pi_E) + \hat{\Phi}(\pi_E) - \Phi_E\|_2 \\
&\leq \|\Phi(\pi) - \hat{\Phi}(\pi_E)\|_2 + \|\Phi_E - \hat{\Phi}(\pi_E)\|_2 \qquad (5)\\
&\leq \epsilon + \epsilon_m, \qquad (6)
\end{aligned}$$

where Eq. (5) follows from the triangle inequality, and Eq. (6) follows from Eq. (4). □

The proof (based on Abbeel and Ng [2004]) can be found in the supplementary material. In the proof, we assumed that the samples in Lemma 3 are bounded and unbiased. For finite horizon, each sample can be obtained by observing the expert executing a trajectory. In the discounted case, we can follow Syed and Schapire [2008], limit the trajectories to be of length $H \geq (1/(1-\gamma))\log(1/(\epsilon_H(1-\gamma)))$ and show that this comes at an additional cost of ϵ_H. Alternatively, we can follow Kakade and Langford [2002], execute the expert trajectory online, and terminate it at each time step with probability $1-\gamma$. This will make the estimate unbiased and bounded with high probability. Thus, it is possible to follow Zahavy et al. [2019] to obtain a concentration result.

C Experiments

In this section, we compare the CG and ASCG methods for AL in two AL domains: an autonomous driving simulation [Abbeel and Ng, 2004, Syed and Schapire, 2008], and a grid world domain. The results in each experiment are averaged over 10 runs of each algorithm (random seeds). The mean is presented in a solid line; around it, the colored area shows the mean plus/minus the standard deviation.

Setting. In each domain, there are fixed dynamics and initial state distribution that are given as a simulator (and not explicitly in a matrix form). Computation of feature expectations are done through Monte Carlo simulations: Given a policy π, (that could be the expert or the agent) we execute $N_{\text{Estimation}}$ trajectories of length H from a state drawn by the initial state distribution. The average of the cumulative discounted sum of the features along these trajectories is our estimated feature expectations of π. Given a reward function, the optimal policy is computed by running Q learning [Watkins and Dayan, 1992] for N_{RL} steps. Our implementation of Q learning is standard and includes an $\epsilon-$greedy exploration with $\epsilon = 0.05$ and a learning rate of $\alpha_t = 0.2/t^{0.75}$ (following Even-Dar and Mansour [2003]). It is important to note that both of these procedures do not necessarily return accurate solutions and that these solutions become more accurate as we increase the computation resources, and specifically, $N_{\text{Estimation}}, H, N_{\text{RL}}$. Empirically we found that as we increased these resources (making our implementation closer to the theoretical framework), the differences between the two methods become more significant.

Gridworld domain. In this domain, we place an agent in a 5×5 grid world domain. The agent can move up, down, left, and right. At each point in the grid, there is a reward (negative, zero), and there is a single point with a positive reward. Once the agent collected a positive reward, it starts again from the initial state. We used $N_{\text{Estimation}} = 300, H = 50, N_{\text{RL}} = 300$ and run both CG and ASCG for $N_{iter} = 100$ steps. In Appendix C, we can see the error of each algorithm as a function of the iteration number. The error measures the distance (in a logarithmic scale) between the feature expectations of the expert and those of the agent at time t, i.e., $\|\Phi_E - \Phi_t\|$. We can see that ASCG has a clear advantage over CG.

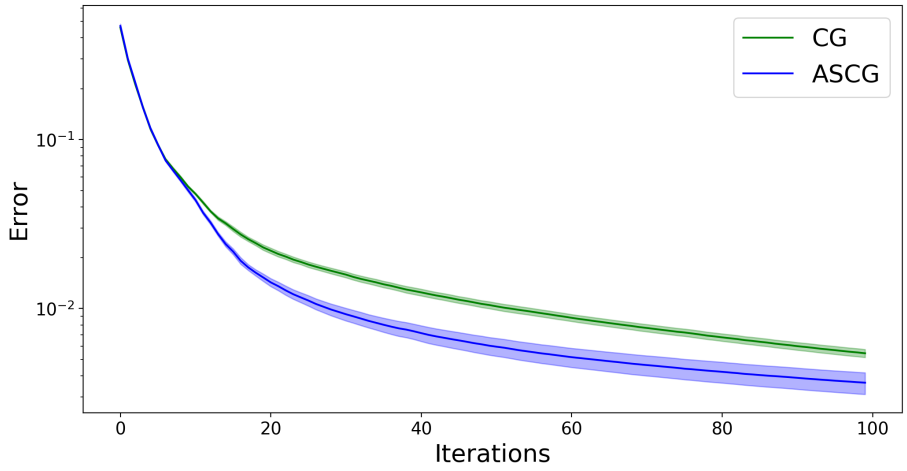

Figure 1: Comparison of CG with ASCG on a 5×5 gridworld domain

Car simulator. The driving task simulates a three-lane highway, in which there are two visible cars - cars A and B. The agent, car A, can drive both on the highway and off-road. Car B drives on a fixed lane, at a slower speed than car A. Upon leaving the frame, car B is replaced by a new car, appearing in a random lane at the top of the screen. The reward is a linear combination of driving features: speed, collisions, and off-road driving. The goal of the agent is to find a driving policy that balances between these features based on expert preferences.

We used $N_{\text{Estimation}} = 1000, H = 40, N_{\text{RL}} = 1000$ and run both algorithms for $N_{iter} = 50$ steps. Similar to the greed domain, we can see that ASCG has a clear advantage over CG.

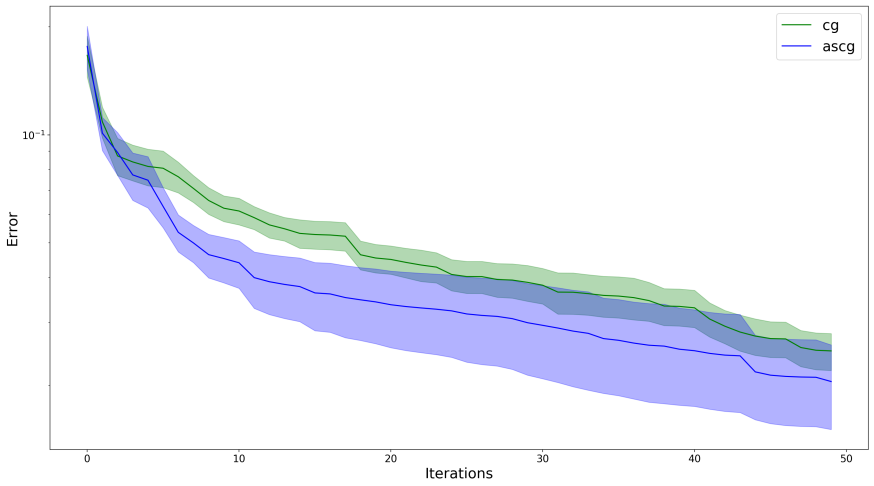

Figure 2: Comparison of CG with ASCG on the car driving simulator

D Related work

Another type of AL algorithms was proposed by Syed and Schapire [2008]. The idea is to solve Eq. (2) in the following manner. In each round the "reward player" plays an online convex optimization algorithm on losses $l_t(w_t) = w_t \cdot (\Phi_E - \Phi(\pi_t))$; and the "policy player" plays the best response, i.e, the policy π_t that maximizes the return $\Phi(\pi_t) \cdot w_t$ at time t. The algorithm runs for T steps and returns a mixed policy ψ that draws with probability $1/T$ a policy $\pi_t, t = 1, \ldots, T$. Thus, we have that

$$f^* \leq \frac{1}{T} \sum_{t=1}^{T} \max_{\pi \in \Pi} w_t \left[\Phi(\pi) - \Phi_E\right] = \frac{1}{T} \sum_{t=1}^{T} w_t \left[\Phi(\pi_t) - \Phi_E\right] \leq \min_{w \in \mathcal{W}} \frac{1}{T} \sum_{t=1}^{T} w \left[\Phi(\pi_t) - \Phi_E\right] \tag{7}$$

$$+ O\left(\frac{\sqrt{\log(k)}}{(1-\gamma)\sqrt{T}}\right) = \min_{w \in \mathcal{W}} w \left(\Phi(\psi) - \Phi_E\right) + O\left(\frac{\sqrt{\log(k)}}{(1-\gamma)\sqrt{T}}\right), \tag{8}$$

where Eq. (7) follows from the fact that the policy player plays the best response, that is, π_t is the optimal policy w.r.t the reward w_t; Eq. (8) follows from the fact that the reward player plays a no-regret algorithm, e.g., online MDA.Thus, we obtain from Eq. (8) that $\forall w \in \mathcal{W} : w \cdot \Phi(\psi) \geq w \cdot \Phi_E + f^* - O\left(\frac{1}{\sqrt{T}}\right)$[1]. Since this technique runs a no regret algorithm, it cannot obtain a convergence rate faster than $T = O(1/\epsilon^2)$[1].

Syed and Schapire [2008] proved that their scheme is faster by a factor of k and requires only $T = O(\log(k)/(1-\gamma)^2\epsilon^2)$ iterations. This improvement is closely related to the analysis of the mirror descent algorithm (MDA, Nemirovsky and Yudin [1983]). That is, by choosing the norm of the space (and projecting w.r.t this norm), a dimension-free rate of convergence (up to logarithmic factor) is achieved. The results in [Syed and Schapire, 2008] use a specific instance of MDA where the optimization set is the simplex and distances are measured w.r.t $\|\cdot\|_1$. This version of MDA is known as multiplicative weights or Hedge.

Finally, IRL can also be formulated as a convex optimization problem, but it is not differentiable [Ratliff et al., 2006]. IRL is also not strongly convex, as it does not have a unique solution, as was observed in Ng and Russell [2000]. For these reasons, convex optimization methods for IRL did not achieve a linear rate of convergence.

[5]The O notation hides the dependency in k and γ .