Projection-free Algorithms for Convex Optimization and Online Learning

Dan Garber

Projection-free Algorithms for Convex Optimization and Online Learning

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Dan Garber

Submitted to the Senate of
the Technion — Israel Institute of Technology
Shevat 5776 Haifa January 2016

The research thesis was done under the supervision of Prof. Elad Hazan in the Industrial Engineering and Management Department.

Most results in this thesis are based on the following papers:

Dan Garber and Elad Hazan. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. Under revision for SIAM Journal on Optimization.

Dan Garber and Elad Hazan. Faster rates for the Frank-Wolfe method over strongly-convex sets. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015.

Dan Garber, Elad Hazan, and Tengyu Ma. Online learning of eigenvectors. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015.

Dan Garber and Elad Hazan. Playing non-linear games with linear oracles. 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013.

Acknowledgements

First and foremost, I would like to thank my advisor Elad Hazan. I could not have asked for a better mentor to guide me through grad-school. Working with Elad was always fun and exhilarating. I owe most of what I have learned during these years to him.

Next, I would like to thank the superb group of people I had the fortune of sharing my workspace with: Tomer Koren, Oren and Yael Anava, Kfir Levi, and Alon Cohen. It has been a great joy sharing the wonderful travels to conferences, numerous lunches and talks.

I would like to thank my mother, Shoshana Garber, for her constant support and encouragement throughout the years.

Finally, I would like to thank my family, to whom I also dedicate this thesis. I would like to thank my beloved wife Dana for her love and support throughout these long years, and for always believing in me and encouraging me. I would like to thank the three musketeers: Shira, Harel, and Rom. I still sometimes find it hard to believe that I managed to graduate while having three kids around. These guys are my light, my strength, a constant reminder to push forward and improve.

The generous financial support of the Technion is gratefully acknowledged.

To Dana, Shira, Harel, and Rom.

Contents

List of Figures

Abstract

The problem of minimizing a linear function over a convex set is many times algorithmically simpler and more efficient than its non-linear convex counterpart. Examples include polytopes that arise in combinatorial optimization problems, certain sets of matrices with bounded singular values, and balls induced by p-norms, and generalizations of.

This phenomena motivates the computational model of convex optimization and online learning using a linear optimization oracle. In this computational model we give several new results that improve over the previous state-of-the-art:

1. We present a variant of the *conditional gradient method*, a first-order method for constrained smooth minimization using a linear optimization oracle, that converges exponentially fast when the convex set is a polytope and the objective function is strongly convex. This gives an exponential improvement in convergence rate over previous results.

2. Based on the machinery developed to derive the above result, we derive a linear optimization-based algorithm with optimal regret for online convex optimization, in case the feasible set is a polytope. This resolves open questions posed by Kalai and Vempala, and Hazan and Kale, in this important case. The new online algorithm also gives rise to linear optimization-based algorithms for non-smooth and stochastic convex optimization over polytopes, with the same rates as projection-based first-order methods in terms of the accuracy parameter.

3. We show that the vanilla conditional gradient algorithm converges at an accelerated rate (quadratic improvement with respect to the standard, previously known, rate) when applied to smooth and strongly

convex optimization over strongly convex sets, which include various balls induced by p-norms, and generalizations of.

4. We present a linear optimization-based online algorithm for a natural online extension of the fundamental leading eigenvector problem from numerical linear algebra. In contrast to previous linear oracle-based algorithms, the new algorithm i) enjoys a regret bound with a much more favourable dependency on the dimension, and ii) requires computations that depend only on the sparsity of the data and not explicitly on the dimension. We also present a simpler and more efficient algorithm for the easier stochastic version of the problem.

Chapter 1

Introduction

This thesis presents algorithms for optimization problems that fall into one of the following two paradigms: convex optimization or online learning. Convex optimization requires little introduction. It has a very rich literature that was developed in the past several decades and has seen numerous applications in almost every field of science and engineering. Online learning is a relatively new optimization paradigm that has emerged about two decades ago and has since then established itself as a powerful framework for modeling various prediction, decision-making, and optimization problems of a certain repetitive nature, with deep connections to game theory, statistical learning, and convex optimization.

There are very strong connections between these two paradigms: online learning has borrowed many tools from the rich convex optimization toolbox, and in turn, many algorithms for convex optimization that were developed in recent years, are based on tools and insights from online learning.

In both paradigms, a major body of work was dedicated over the years to develop algorithms that are, at a high-level, optimal in terms of the amount of information they require on the objective function, in order to guarantee a certain bound on their performance. However, in many cases of interest, especially in high-dimensional settings, these algorithms are not practically efficient. The reason for this inefficiency is that while these algorithms are designed to be optimal with respect to one measure (i.e. information on the objective), they overlook certain computational aspects which are many times critical. To make this argument more concrete: these algorithms assume the availability of an oracle that, by it self, solves a certain convex op-

timization problem, known as the *projection*. Implementing this projection oracle is many times computationally prohibitive on large scale problems.

The unifying theme of the algorithms presented in this thesis is that they exchange the computationally-expensive projection oracle, with a potentially much efficiently-implementable oracle, and thus we refer to them as *projection-free* algorithms. In particular, they assume only an oracle for linear optimization over the feasible set.

In the remaining of this chapter we extend the above discussion. The eager reader already familiar with first-order methods for convex optimization and online learning, can skip the remaining of this chapter and go straight ahead to the following chapters since each chapter is self-contained and can be read independently of the others.

1.1 Convex Optimization and First-order Methods

In this thesis we consider convex optimization in the following form:

$$\min_{x \in \mathcal{K}} f(x), \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $\mathcal{K} \subseteq \mathbb{R}^n$ is a convex set. For ease of presentation we are going to assume throughout this introductory chapter, unless stated otherwise, that f is differential everywhere in $\mathcal{K}$ [1].

Given a tolerable-error parameter ϵ, we are looking for a feasible point $x_\epsilon \in \mathcal{K}$ such that

$$f(x_\epsilon) - \min_{x \in \mathcal{K}} f(x) \leq \epsilon.$$

Under relatively mild assumptions, it is possible to find such x_ϵ in polynomial time via methods such as the *Ellipsoid method* [39], *Interior-Point methods* [73] and *Random-Walk* methods [57]. However, these methods use computationally-expensive iterations in terms of run-time and memory requirements which tend to be impractical when the dimension n is very large. Indeed in recent years, due to the data-explosion phenomena, there is a growing interest in solving optimization problems that involve vast amounts of

[1] the case in which f is not differential only affects some of the technical details but not the "big picture" we are trying to convey in this introduction.

data for purposes of massive data-analysis. It is for this reason that in recent years *first-order* methods, i.e. algorithms that access the function f only through an oracle that given a point $x \in \mathbb{R}^n$, returns the function value and its first derivative at x, that is $f(x), \nabla f(x)$, with computationally-cheap iterations became the method-of-choice for coping with large scale convex optimization problems [2].

1.1.1 The unconstrained case

Suppose f is given by a "black-box" first-order oracle $\mathcal{O}_f : \mathcal{K} \to \mathbb{R} \times \mathbb{R}^n$ that given a point $x \in \mathcal{K}$ returns the tuple $(f(x), \nabla f(x))$. In case the feasible set $\mathcal{K}$ is simply the entire space $\mathbb{R}^n$, a natural way to use $\mathcal{O}_f$ is to produce a sequence of iterates $\{x_t\}_{t=1}^{\infty}$ such that

$$x_{t+1} \leftarrow x_t - \eta_t \nabla f(x_t), \tag{1.2}$$

where $\{\eta_t\}_{t=1}^{\infty}$ is a sequence of non-negative real scalars, and x_1 is set to some arbitrary point in $\mathbb{R}^n$. This simple update rule is known as *gradient descent*. Indeed one can show that, under various assumptions on $f(x)$, there exists a choice for $\{\eta_t\}_{t=1}^{\infty}$ and a corresponding value T_ϵ such that for every $t \geq T_\epsilon$ it holds that $f(x_t) - \min_{x \in \mathcal{K}} f(x) \leq \epsilon$. For instance we bring the following theorem, a proof of which can be found for instance in [12].

Theorem 1. *Suppose that $f(x)$ is twice differential everywhere in $\mathbb{R}^n$ and it holds for any $x \in \mathbb{R}^n$ that $\alpha I \preceq \nabla^2 f(x) \preceq \beta I$, where I is the identity matrix in $\mathbb{R}^n$ and $A \preceq B$ means that for any vector v, $v^\top A v \leq v^\top B v$, and $\beta \geq \alpha > 0$. Then, there exists a choice for $\{\eta_t\}_{t=1}^{\infty}$ such that $T_\epsilon = O\left(\frac{\beta}{\alpha} \log((f(x_1) - f(x^*))/\epsilon)\right)$, where $x^* := \arg\min_{x \in \mathcal{K}} f(x)$ (which in this case is unique).*

The fact that f is accessed only through the oracle $\mathcal{O}_f$ has the huge advantage that it encapsulates the difficulty of representing f or computing its derivatives, and hence allows to design universal methods that assume only very little on the function f, which is very different from the ad-hoc

[2] we intentionally add the requirement of computationally-cheap iterations since the Elliposid method can also be viewed as a first-order method, however, both the run-time of a single iteration and the number of iterations to convergence scale like n^2, which justifies its reputation is inefficient in practice.

approach vastly used for combinatorial optimization problems in computer science, where each specific problem usually requires its own algorithm and analysis. The disadvantage is of course that now we cannot analyze the complexity of the algorithm using the standard methodology of computer science, i.e. counting the number of arithmetic operations. Instead, in this oracle model, the natural way to measure the efficiency of a method is according to the maximal number of calls it needs to issue to the oracle $\mathcal{O}_f$ in order to reach a point x_ϵ, which for the update-step given in Eq. (1.2), just translates to a worst-case upper bound on T_ϵ.

Indeed much research was devoted in the past decades to develop first-order methods that optimize this complexity measure, and today we have optimal algorithms (i.e. their worst-case complexity matches known lower-bounds on the complexity of, roughly speaking, any reasonable first order method), under various assumptions on the function $f(x)$ [71]. One such optimal algorithm is the *accelerated gradient method* developed by Nesterov [71] which uses a simple, yet highly non-trivial, modification of the update step in Eq. (1.2), and improves the bound on the number of iterations T_ϵ given in Theorem 1 to $T_\epsilon = \tilde{O}\left(\sqrt{\frac{\beta}{\alpha}}\log((f(x_1) - f(x^*))/\epsilon)\right)$, i.e. the factor β/α is improved to $\sqrt{\beta/\alpha}$, which is critical in certain settings, and in fact tight.

1.1.2 The constrained case

Things become more complicated when the feasible set $\mathcal{K}$ is not the entire space $\mathbb{R}^n$, but only a subset of it, which we will assume is closed and compact. Observe that now the simple gradient descent rule given in Eq. (1.2) cannot be used, since nothing is keeping the iterate x_t from stepping outside of $\mathcal{K}$. There is a conceptually simple solution to this problem given by the following modified update step:

$$x_{t+1} \leftarrow \arg\min_{x\in\mathcal{K}} \|x - [x_t - \eta_t \nabla f(x_t)]\|_2. \tag{1.3}$$

That is, x_{t+1} is produced by applying the update rule (1.2), which might result in an infeasible point, and then taking the next iterate to be the feasible point closest to it in Euclidean distance. The computation $\arg\min_{x\in\mathcal{K}} \|x - y\|_2$ is called *the projection* of the point y onto the set $\mathcal{K}$.

We focus our discussion here on the Euclidean setting, i.e. we measure distances according to the ℓ_2 norm which agrees with the update step in (1.3). There is interest in certain settings to use other norms, which results in update rules that are different from (1.3), such as the celebrated *mirror descent* method [11] however, as a rule of thumb, in these cases the iteration-complexity of the resulting method is worse than in the Euclidean setting. Since our claim is that even computing the Euclidean projection is practically inefficient many times, we restrict our discussion to the Euclidean setting.

Note that in terms of complexity, the rule in Eq. (1.3) amounts to a single call to the oracle $\mathcal{O}_f$ and minimizing a single quadratic function over the feasible set, which is by itself, a convex optimization problem.

As a rule of thumb, the same convergence rates that are attainable for the unconstrained case using the update-rule (1.2) and its variants (such as Nesterov's accelerated method) also apply in the constrained setting when we add the projection, as done in the update-rule given in (1.3). For instance, Theorem 1 still holds when the iterates are generated according to (1.3) instead of (1.2).

Computing projections to convex sets

Since the overall complexity of the method described by (1.3) is affected by the complexity of solving the projection problem onto $\mathcal{K}$, we now survey several important convex sets that arise both in theory and applications and discuss the efficiency of computing the projection.

The Euclidean ball Computing the projection to a Euclidean ball is trivial. Assuming w.l.o.g. that the ball is centered at the origin and has unit radius, the projection of a point $y \in \mathbb{R}^n$ onto the ball is given by $\Pi_{\text{ball}}(y) = \frac{y}{\max\{1, \|y\|_2\}}$, which could be carried out in time that is linear in the number of non-zero entries in y.

The hypercube The n-dimensional hypercube is the convex hull of the set of points $\{0,1\}^n$. Computing the projection to the hypercube is also trivial since it amounts to projecting each coordinate in a vector y independently of the others. That is $[\Pi_{\text{cube}}(y)](i) = \max\{0, \min\{1, y_i\}\}$. Again, the

projection in this case could be carried out in linear time in the number of non-zeros in y.

The unit simplex and the ℓ_1-ball The unit simplex in $\mathbb{R}^n$ is the convex hull of all standard basis vectors in $\mathbb{R}^n$, which is also the set of all distributions over n elements. The ℓ_1-ball, which we assume w.l.o.g. has unit radius, is simply the set $\{x \in \mathbb{R}^n \mid \sum_{i=1}^n |x_i| \leq 1\}$. Computing the projection of a vector y onto the simplex and the ℓ_1-ball, denoted here by $\Pi_{\text{simplex}}(y)$, $\Pi_{\ell_1-\text{ball}}(y)$, admits a non-trivial thresholding scheme which could be implemented in $O(n\log(n))$ time [24].

Polytopes Polytopes are one of the most important family of convex sets. Consider the polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ for some $m \times n$ matrix A and $b \in \mathbb{R}^m$. In general, the problem of projection onto a polytope $\mathcal{P}$ could be formulated as a *conic quadratic programming* problem (CQP) [11], which can in turn be solved via a state-of-the-art interior point method. However, the running time IP for CQP is dominated by $\sqrt{m}n(n^2+m)$, which is highly impractical for large scale problems.

Things get worse for polytopes for which the number of constraints is not polynomial in the dimension n, and thus even IP methods cannot be used. This is for instance the case for the *perfect matching polytope* [77]. In this case other methods such at the Ellipsoid method may be used to compute the projection. However, the inherent complexity of the Ellipsoid method is $O(n^4)$, excluding the time required to generate separating hyperplanes (which could be carried out in polynomial time for this polytope [79]), which is again inefficient for large scale instances. A similar case arises when considering the polytopes of matroids (see [79]), which are naturally described by exponentially many constraints, and thus one needs to use the Ellipsoid method for computing the projection.

For some non-trivial polytopes there exists tailored ad-hoc methods for computing the projection, such as the method presented in [88], which can be used to compute the projection onto flow-related polytopes (i.e. convex hull of fesible flows in a graph) in strongly polynomial time. However, here the running time is $\tilde{O}(n^4)$, where n is the number of edges in the graph, which is again not practical in large scale.

The spectrahedron and the nuclear-ball The spectrahedron, a certain matrix-generalization of the unit simplex, is the set of all positive semidefinite $n \times n$ matrices (i.e. real symmetric matrices with non-negative eigenvalues) with unit trace (i.e. sum of elements along the main diagonal is exactly 1). For $m \times n$ matrices, where w.l.o.g. $m \leq n$, the nuclear norm-ball of unit radius is the set of all $m \times n$ matrices whose vector of singular values has ℓ_1 norm at most 1 (i.e. the singular values vector is in the m-dimensional unit ℓ_1 ball). These sets arises in many problems involving matrices such as *matrix completion* [76, 75, 53] and semidefinite optimization [45]. Projection to the spectrahderon (nuclear-ball) is closely related to the projection onto the simplex (ℓ_1-ball). The resulting matrix retains the eigenvectors (left and right singular vectors) of the original matrix, and the vector of eigenvalues (singular values) is projected onto the simplex (ℓ_1-ball). Since, as follows, one needs the full *singular vector decomposition* of a matrix in order to project it onto either one of these sets, the complexity is $O(n^3)$ for the spectrahedron, and $O(m^2 n)$ for the nuclear-ball. In both cases the run-time is superlinear in the dimension, and hence not practical in very large scale.

The picture that is hopefully clear from the above survey is that, while computing the projection is very efficient for simple convex sets such as the ball, hypercube, simplex (i.e. could be carried out in linear or nearly linear time), it is highly inefficient for more involved, yet highly important, convex sets such as polytopes and certain sets of matrices, when the dimension is high. Thus, even in cases in which implementing the first-order oracle $\mathcal{O}_f$ is very efficient, which is the case in many important applications, computing the projection may render the overall method impractical.

1.2 Online Learning and Online Convex Optimization

Online learning is a general optimization problem that takes the form of the following repeating game: on each round of the game, a decision maker is required to choose an action from a fixed set of actions. Then, a loss function, mapping each possible action to a real scalar, is revealed, and the decision maker incurs the loss of his action. This interaction repeats itself for a finite number of rounds. The emphasis in this game is that on each round,

when choosing his action, the decision maker has basically no knowledge on the loss function that is going to be revealed and assign loss to his action. In particular, it could be the case that the loss function is chosen by an *adversary* according to the action of the decision maker, with the intent of making him suffer a large loss at this round of the game. Thus, this is a worst-case limited-information optimization problem.

An important variant is known as the "bandit" setting (as opposed to the standard "full-information" setting we have just described), in which, on each round, the loss function is not revealed, but only the loss of the action chosen by the decision maker on that round.

Perhaps the most important and well-studied class of problems that fall into the paradigm of online learning, is known as *online convex optimization* (OCO), in which the set of actions of the decision maker is a convex set in some linear space (and thus an action corresponds to a point in the set), and the loss functions revealed are also convex.

Because the decision-maker has no knowledge on the loss function that is about to be reveled when making his choice on each round, intuitively, there is no hope for him to play optimally, i.e. compete in terms of the total loss with the best sequence of T actions in a game of overall T rounds. Instead, the standard goal is to compete with an easier yet still powerful benchmark: the best fixed action in hindsight. This performance measure is known as the *regret*.

The standard text on online learning is [14]. For a more recent one see [80].

1.2.1 Formal definition of OCO

Since in the context of online learning we will mostly consider problems within the subclass of OCO, we are going to focus on the mathematical formulation of OCO.

We consider a repeated game of T rounds, where on each round t, the decision maker is required to choose a point $x_t \in \mathcal{K}$, where $\mathcal{K} \subset \mathbb{R}^n$ is a fixed compact and convex set. After choosing x_t, a loss function $f_t : \mathbb{R}^n \to \mathbb{R}$ is revealed, and the decision maker incurs the loss $f_t(x_t)$. In the full-information setting, at the end of the round, the decison maker gains full knowledge of $f_t(x)$. On the contrary, in the "bandit" setting, the decision

maker only learns the value $f_t(x_t)$ and does not gain any other knowledge on $f_t(x)$. The goal of the decision maker is to minimize the *regret* which is formally given by

$$\text{regret}_T(x_1, x_2, ..., x_T) := \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x).$$

In words: the decision maker wishes to minimize the difference between his cumulative loss and the cumulative loss of the best fixed point in $\mathcal{K}$ (throughout this work we will simply write regret_T, as the sequence of plays $x_1, ..., x_T$ to which it refers will be clear from context).

Most importantly, the decision maker is considered successful if he can guarantee regret that grows as a sublinear function of T, i.e. the ratio regret_T/T is strictly decreasing in T. In this case, indeed when $T \to \infty$, the average loss of the decision maker per round is at most that of the best fixed action in hindsight. It is for this reason, that when designing algorithms for OCO (and online learning in general), the primary concern is the dependence of the worst-case regret on the length of the game T.

1.2.2 Applications

We now briefly overview some examples of optimization problems that fall into the framework of online learning and OCO.

Predicting with expert advise Perhaps the most well-known online learning problem, is the problem of predicting with expert advise. In this problem, a decision maker has access to n experts. On each round of the game the decision maker must choose one of these experts. After making his choice, a loss vector that maps each expert to a loss in the interval $[0, 1]$ is revealed, and the decision maker incurs the loss of the expert he has chosen. The goal here is to perform in the long-run nearly as well as the best fixed expert in hindsight.

This problem is easily modelled in the setting of OCO: each expert i is mapped to the ith standard basis vector in $\mathbb{R}^n$, e_i, and the convex set $\mathcal{K}$ is taken to be the convex-hull of $\{e_i\}_{i=1}^n$ which is also known as the *unit simplex*. Given the vector of losses on round t, denoted by $\ell_t \in \mathbb{R}^n$, the loss function for round t is naturally defined to be $f_t(x) = x \cdot \ell_t$. Note that in this

formalism, the action of the decision maker on round t, x_t, is not a single expert, but rather the distribution over the experts. However, by choosing expert i with probability $x_t(i)$ we can make sure that the decision maker chooses a single expert while ensuring that the *expected* loss on round t will be the same as playing the distribution x_t (it is important to note however that here we assume that the loss vector ℓ_t is chosen independently of the randomness used in order to sample from x_t).

The important "bandit" variant of the problem, known as *the adversarial multi-armed bandit problem*, has also received much attention in the literature. See the seminal work [6].

Online shortest path Let G be a directed graph with n vertices and m edges. Let s, t be two unique vertices which we refer to as the *source* and the *target*. On each round of the online shortest path problem, the decision maker is required to choose a path in G leading from s to t. Then, each edge is assigned a weight in the interval $[0, 1]$, and the decision maker incurs loss that equals the sum of weights along the path he has chosen. The goal is again to perform in the long-run nearly as well as the best fixed $s - t$ path in the graph in hindsight.

This problem could be easily reduced to the problem of predicting with expert advise: we can map each simple $s-t$ path in G to an expert, and the loss of each expert on round t will be the weight of the path that corresponds to this expert. However, since the number of paths may be exponential in the number of edges, this leads to inefficient algorithms.

This problem could be modelled directly in the setting of OCO as follows: we map each simple $s - t$ path to its identifying vector in $\{0, 1\}^m$ (i.e. there is 1 in the ith entry of the vector if and only if the ith edge in the graph is part of this path), and then take the convex set $\mathcal{K}$ to be the convex-hull of all identifying vectors (this is also known as the *unit flow polytope*). If we denote by $\ell_t \in \mathbb{R}^m$ the vector of edge-weights on time t, then again the loss function could be written as $f_t(x) = x \cdot \ell_t$, which is again just a linear function. Note that now both vectors x_t, ℓ_t are in $\mathbb{R}^m$, which is an efficient representation in the natural parameters of the problem m, n. As in the previous example, we need to note that now the decision on time t is not a path, but rather an implicit distribution over paths, which in turn is a unit flow from s to t. Such an implicit distribution could be decomposed

in polynomial time to an explicit distribution over at most m $s-t$ paths, and thus, as before, we can use this distribution to randomly choose a path, while incurring the same loss in expectation.

For a "bandit" variant of the problem see [7].

Online Principal Component Analysis In the fundamental problem of *Principal Component Analysis* (PCA), given a set of vectors $x_1, x_2, ..., x_T$ in $\mathbb{R}^n$ and an integer $k \in [n]$, we are interested in finding a rank-k *projection* matrix, i.e. a matrix $W \in \mathbb{R}^{n \times n}$ such that $W^2 = W$, that minimizes the quantity $\sum_{t=1}^{T} \|x_t - Wx_t\|_2^2$. That is, we wish to find k directions is space such that the projection of the data onto these directions minimizes the reconstruction error.

In the online learning version of this problem [91], the vectors $x_1, ..., x_T$ arrive sequentially. On round t, the decision maker chooses a rank-k projection matrix $W_t \in \mathbb{R}^{n \times n}$, and then the vector x_t is revealed and the loss incurred is $\|x_t - W_t x_t\|_2^2$. The goal here is to compete with the best fixed projection matrix in hindsight, which is just given by the offline PCA over the entire data $x_1, ..., x_T$.

The loss function in this setting is clearly $f_t(W) = \|x_t - Wx_t\|_2^2$, i.e. the reconstruction error of x_t when using the projection matrix W. However, as in previous examples, the set of all rank-k $n \times n$ projection matrices is not convex. Again, this difficulty is resolved by taking $\mathcal{K}$ to be the convex hull of all such matrices. It can be shown that $\mathcal{K}$ is exactly the set of all $n \times n$ positive semidefinite matrices with sum of eigenvalues bounded by k and largest eigenvalue bounded by 1. Moreover, given a matrix in $\mathcal{K}$, it is possible in turn to decompose it into to a convex combination of at most n rank-k projection matrices using its eigendecomposition [91]. Thus, as before, we can use this decomposition to randomly sample a single matrix without changing the loss in expectation.

Online Portfolio Selection In the online portfolio selection problem [20, 1], we are given a set of n financial assets in which we can invest our wealth. At each round of the repeated game, the decision maker is required to spread his wealth among these n assets, formally by choosing a distribution that assigns to each asset the fraction of wealth to be invested in it. After making his choice, a vector of returns that prescribes the return of each asset is

revealed, and the wealth of the decision maker is changed according to his chosen distribution and the vector of returns, and the game continuous to a new round. Two important remarks here are: i) the selected portfolio is *rebalancing*, i.e. the decision maker sells his entire investments at the beginning of each new trading round and then uses all of his wealth to re-buy assets. This is true even if his portfolio does not change from one round to another, and ii) in this simplified model we overlook any transaction costs for buying and selling assets and we just assume they are 0.

Since the decision maker action is a distribution over the n assets, the feasible set in this problem is naturally the unit simplex in $\mathbb{R}^n$. To keep track on the change in the wealth of the decision maker during trading rounds, the loss funtion on round t is taken to be $f_t(x) = -\log(x \cdot r_t)$, where $r_t \in \mathbb{R}^{n+}$ is the vector of returns on round t. To see why this choice makes sense observe that

$$\sum_{t=1}^{T} f_t(x_t) = -\sum_{t=1}^{T} \log(x_t \cdot r_t) = -\log\left(\prod_{t=1}^{T} x_t \cdot r_t\right)$$

If we denote by w_t the total wealth of the decision maker at the beginning of round t, then we arrive at the recursion: $w_{t+1} = w_t x_t \cdot r_t = ... = w_1 \prod_{\tau=1}^{t} x_\tau \cdot r_\tau$. Thus, if we assume without losing generality that the initial wealth w_1 is 1, then $-\sum_{t=1}^{T} \log\left(\prod_{t=1}^{T} x_t \cdot r_t\right)$ is the minus logarithm of the overall wealth after T trading rounds. Moreover, if we denote by x^* the best constant rebalancing portfolio (CRP) in hindsight, then

$$\text{regret}_T = -\log\left(\prod_{t=1}^{T} x_t \cdot r_t\right) + \log\left(\prod_{t=1}^{T} x^* \cdot r_t\right) = \log\left(\frac{w_T^*}{w_T}\right)$$

is the log ratio between the wealth of the best CRP at the end of the trading period, and the wealth of decision maker.

Application to Stochastic Optimization and Statistical Learning OCO has strong connections to the learning and optimization paradigms known as *statistical learning* and *stochastic optimization*. Roughly speaking, in both paradigms the goal is to minimize a function over some feasible set however, the function is not given explicitly, but is defined by the expectation according to some unknown distribution over a (possibly infinite)

set of functions. Hence, in these settings we do not have direct access to the objective function, but only to unbiased estimators of its values, or its derivatives. In these problems, the standard way to measure peformence is according to the *sample complexity*, i.e. number of estimators the algorithm needs to observe in order to guarantee a certain approximation to the optimal value. For more details see [13, 48], and also Section 2.1.3.

Application to Convex Optimization Algorithms for OCO could be applied via a natural and elegant framework, that allows for a robust and flexible analysis, to solve saddle-point optimization problems. As a result, many state-of-the-art algorithms for convex optimization based on OCO algorithms were developed in recent years. See for instance [38, 19, 37, 17], and also Section 4.7 for a concrete application to semidefinite optimization.

1.2.3 Algorithms for OCO

We are going to assume that the feasible set $\mathcal{K}$ is closed and compact (which is standard in OCO) and for ease of presentation, we are going to assume in this chapter that all loss functions $\{f_t\}_{t=1}^T$ are differential everywhere in $\mathcal{K}$.

Following our discussion on first-order methods for convex optimization, it is tempting to consider the following, seemingly simple, modification of the update-rule (1.3):

$$x_{t+1} \leftarrow \arg\min_{x \in \mathcal{K}} \|x - [x_t - \eta_t \nabla f_t(x_t)]\|_2. \tag{1.4}$$

That is, on each iteration t of the OCO game, the next iterate is produced by performing a gradient improvement step with respect to the *current* loss function f_t, and then projecting the resulting point to enforce feasibility. Suprisingly, this update-rule for OCO can indeed guarantee low regret, and in fact optimal regret! The following theorem is basically due to Zinkevich [94].

Theorem 2. *Suppose that there exists a constant G such that for all $t \in \{1, 2, ..., T\}$ and all $x \in \mathcal{K}$, it holds that $\|\nabla f(x)\|_2 \leq G$. Then, there exists a choice for $\{\eta_t\}_{t=1}^T$ such that $\text{regret}_T = O(GD\sqrt{T})$, where D is the diameter of $\mathcal{K}$, i.e. $D := \max_{x,y \in \mathcal{K}} \|x - y\|_2$.*

There is a strong connection to be drawn between the worst-case regret bounds attainable for OCO (under certain assumptions on the set $\mathcal{K}$ and the functions $f_1, ..., f_T$) and the oracle complexity of first-order methods for convex optimization. Consider the function $r(t) := \frac{\text{regret}_t}{t}$, which is just the average regret after t rounds. We can interpret $r(t)$ as describing a worst-case convergence rate, and as in the setting of convex optimization, ask for which T_ϵ it holds that $r(t) \leq \epsilon$ for all $t \geq T_\epsilon$. With this view, and the update rule suggested in Eq. (1.4), we can see that minimizing the worst case regret is equivalent to minimizing the number of gradients of loss functions the algorithm needs to observe to guarantee sufficiently small average regret. Thus in a sense, both optimal first-order methods for convex optimization, and optimal methods for OCO, optimize the same performance measure.

Of course, the same discussion we had on the computational efficiency of first-order methods for constrained convex optimization holds also here, since we also need to deal with computing projections onto the feasible set. The previous paragraph, indeed suggests that the majority of focus in developing algorithms for OCO in recent years has been on minimizing the oracle complexity and not necessarily the computational complexity of algorithms, which may render even regret-optimal algorithms to be inefficient in practice.

1.3 The Linear Optimization Oracle Model and Motivation for this Thesis

In light of the discussion on information-optimal methods for convex optimization and online convex optimization, and their shortcomings in handling feasible sets with involved geometry (due to the need to compute computationally-expensive projections), in this thesis we wish to strike a new balance between the first-order oracle complexity of algorithms and the complexity of enforcing the feasbilility of the iterates. Towards this end, we consider replacing the projection operation, which is equivalent to minimizing a quadratic function over the feasible set, with a potentially much more computationally-efficient procedure, namely minimizing a linear function over the feasible set. That is, we assume that given a vector $c \in \mathbb{R}^n$,

we can efficiently solve:

$$\arg\min_{x \in \mathcal{K}} x \cdot c.$$

Indeed, linear optimization is the simplest non-trivial optimization problem that we can ask to solve given a fixed feasible set $\mathcal{K}$.

Thus, inspired by the simple projection-based schemes (1.3) and (1.4), and with the intent of replacing the projection operation with a call to a linear optimization oracle (LOO), we arrive at the following high-level question that guides this thesis:

what are the convergence rates attainable for convex optimization and online learning by algorithms that access the feasible set only through a linear optimization oracle?

We conclude this section with the following two important remarks. First, while linear optimization could be carried out via generic algorithms for convex optimization, this is not our intention here, since then, the efficiency of solving the linear problem will not be much more favorable than that of solving the projection problem. Indeed the interesting types of convex sets in this context, are those for which the problem of linear optimization has received special treatment in the literature and for which highly efficient ad-hoc algorithms are available. Examples include various polytopes that arise in combinatorial optimization (see Chapter 2 for concrete examples), various balls induced by ℓ_p norms and generalization of (see Chapter 3), and the spectrahedron (see Chapter 4).

Second, it is known that convex optimization is solvable in polynomial time given a LOO in the following way: given an infeasible point $x \notin \mathcal{K}$, it is possible to run the Ellipsoid method, where the LOO is used to generate a separating hyperplane on each iteration, such that the resulting point is in turn a hyperplane separating x from $\mathcal{K}$. Thus, we can use this as a subprocedure for generating separating hyperplanes for the Ellipsoid method to optimize over $\mathcal{K}$. However, this elaborated nested-ellipsoid method will have arithmetic complexity $\tilde{O}(n^6)$ and will require $\tilde{O}(n^4)$ calls to the LOO [3]. Hence, it is clearly not of any practical interest for even medium-sized

[3] recall the the Ellipsoid method performs $\tilde{O}(n^2)$ iterations, each consists of generating a single separating hyperplane and additional $O(n^2)$ arithmetic operations.

problems. We refer the interested reader to the seminal work [40] for morel details.

1.4 Organization of this Thesis and Contribution

The remaining of this thesis is organized in three chapters. Each chapter considers either the problem of convex optimization or online learning (or both) with a different type of feasible set. Each chapter proposes algorithms that access the feasible set only through a LOO and provably improve over the previous state-of-the-art LOO-based results. Interestingly, in each one of the chapters, it is shown that different proprieties of the feasible set enable the derivation of more efficient algorithms.

To make the results in this thesis more accessible, each chapter is self-contained and can be read independently of the others.

Chapter 2 studies convex optimization and OCO under the assumption that the feasible set is a polytope. The main result of this chapter is a novel variant of an LOO-based algorithm, known as *the conditional gradient method*, for smooth and strongly convex optimization, which converges to an ϵ approximated solution after performing roughly $n \log(1/\epsilon)$ calls to the LOO and the first-order oracle. This gives exponential improvement over previous results in this model that were studied since the original introduction of the conditional gradient method by Frank and Wolfe in 1956 [31].

The chapter then continues to harness the machinery developed for the offline algorithm to derive projection-free algorithms for online convex optimization that require only a single call to the LOO per round of the game, and guarantee optimal regret in terms of the game length T. These results resolve a question of Kalai and Vempala [56], and of Hazan and Kale [49].

A consequence of these online algorithms are algorithms for offline convex optimization settings such as *non-smooth* optimization and *stochastic* optimization, to which previous LOO-based algorithms were not applicable in general, and are optimal in terms of the error parameter ϵ.

Chapter 3 studies convex optimization under the assumption that the feasible set is *strongly convex*, which includes various balls induced by ℓ_p norms and generalizations of. The main result in this chapter is that the vanilla

LOO-based algorithm, known as the *conditional gradient method*, converges at an accelerated rate of $O(1/t^2)$ instead of the previously known $O(1/t)$ rate (i.e. quadratic improvement), when the function f is both smooth and strongly convex.

Chapter 4 studies an online learning problem we refer to as *online learning of eigenvectors*, which is a natural extension to the online setting of the classical problem from numerical linear algebra of computing the leading eigenvector of a real symmetric matrix. The main result of this chapter is a LOO-based online algorithm that improves over previous such algorithms by a factor of roughly $\sqrt{n}$ in the regret. Furthermore, the running time of the algorithm per iteration depends only the number of non-zeros in the observed data, and not explicitly on the dimension n^2, as in previous algorithms.

On the technical side, the derivation of the algorithm follows from a novel approach to analyzing an online meta-algorithm, known as *follow the perturbed leader*, which is inspired by *matrix perturbation theory*. Specifically we study certain spectral proprieties of matrices under random rank-one perturbations, which is interesting in its own right and has further applications.

We also present an online algorithms for a slightly easier *stochastic* setting, which guarantees nearly optimal regret and whose overall complexity per iteration is just the number of non-zero entries in the observed data.

Chapter 2

A Linearly Convergent Conditional Gradient Algorithm with Applications to Online and Stochastic Optimization

First-order optimization methods, such as (sub)gradient-descent methods [83, 69, 70] and conditional-gradient methods [31, 27, 18, 45, 52], are often the method of choice for coping with very large scale optimization tasks. While theoretically attaining inferior convergence rate compared to other efficient optimization algorithms (e.g. interior point methods [73]), modern optimization problems are often so large that using second-order information or other super-linear operations becomes practically infeasible.

The computational bottleneck of (sub)gradient descent methods in many settings is the computation of orthogonal projections onto the convex domain. This is also the case with proximal methods [70]. Computing such projections is very efficient for simple domains such as the euclidean ball, the hypercube and the simplex but much more involved for more complicated domains, making these methods impractical for such problems in high-dimensional settings.

On the other hand, for many convex sets of interest, optimizing a linear

objective over the domain could be done by a very efficient and simple combinatorial algorithm. Prominent examples for this phenomena are the matroid polytope for which there is a simple greedy algorithm for linear optimization, and the flow polytope (convex hull of all $s - t$ paths in a directed acyclic graph) for which linear optimization amounts to finding a minimum-weight path [79]. Other important examples include the set of rotations for which linear optimization is very efficient using Wahba's algorithm [89], and the bounded cone of positive semidefinite matrices, for which linear optimization amounts to a leading eigenvector computation whereas projections require cimputing the *singular value decomposition*.

This phenomena motivates the study of optimization algorithms that require only linear optimization steps over the domain and their linear oracle complexity - that is, the number of linear objectives that the algorithm needs to minimize over the domain in order to achieve a desired accuracy with respect to the optimization objective.

The main contribution of this chapter is a conditional gradient (aka Frank-Wolfe) algorithm for offline smooth and strongly convex optimization over polyhedral sets that requires only a single linear optimization step over the domain on each iteration and enjoys a linear convergence rate, an exponential improvement over previous results in this setting.

We also consider the setting of *online convex optimization* [94, 80, 46, 58]. In this setting, a decision maker is iteratively required to choose a point in a fixed convex decision set. After choosing his point, an adversary chooses some convex function and the decision maker incurs a loss that is equal to the function evaluated at the point chosen. In this adversarial setting there is no hope to play as well as an optimal offline algorithm that has the benefit of hindsight. Instead the standard benchmark is an optimal naive offline algorithm that has the benefit of hindsight but must play the same fixed point on each round. The difference between the cumulative loss of the decision maker and that of of this offline benchmark is known as *regret*. Based on our new linearly converging conditional gradient algorithm, we give algorithms for online convex optimization over polyhedral sets that perform only a single linear optimization step over the domain on each iteration while enjoying optimal regret guarantees in terms of the game length, answering an open question of Kalai and Vempala [56], and Hazan and Kale [49]. Using existing techniques we give an extension of this algorithm to the partial

Setting	Previous	This work
Offline, smooth and strongly convex	t^{-1} [52]	$e^{-\Theta(t)}$
Offline, non-smooth and convex	$t^{-1/3}$ [49]	$t^{-1/2}$
Offline, non-smooth and strongly convex	$t^{-1/3}$ [49]	$\log(t)/t$
Stochastic, non-smooth and convex	$t^{-1/3}$ [49]	$t^{-1/2}$
Stochastic, non-smooth and strongly convex	$t^{-1/3}$ [49]	$\log t/t$
Online, convex losses	$T^{3/4}$ [49]	$\sqrt{T}$
Online, strongly convex losses	$T^{3/4}$ [49]	$\log T$

Table 2.1: Comparison between conditional gradient-based methods for optimization over polytopes in various settings. In the offline and stochastic settings we give the approximation error after t linear optimization steps over the domain and t gradient vector evaluations. In the online setting we give the order of the regret in a game of length T, and after at most T linear optimization steps over the domain. In all results we omit the dependencies on constants and the dimension, these dependencies will be fully detailed in the sequel.

information setting which obtains the best known regret bound for this setting.

Finally, our online algorithms also imply conditional gradient-like algorithms for offline non-smooth convex optimization and stochastic convex optimization that enjoys the same convergence rates as projected (sub)gradient methods in terms of the accuracy parameter ϵ(albeit different dependency on constants and the dimension), but replacing the projection step of (sub)gradient methods with a single linear optimization step, again improving over the previous state of the art in these settings.

Our results are summarized in Table 2.

Related work

The conditional gradient method for smooth optimization Conditional gradient methods for offline minimization of convex and smooth functions date back to the work of Frank and Wolfe [31] which presented a method for smooth convex optimization over polyhedral sets whose iteration complexity amounts to a single linear optimization step over the convex domain. More recent works of Clarkson [18], Hazan [45] and Jaggi [52] consider

the conditional gradient method for the cases of smooth convex optimization over the simplex, semidefinite cone and arbitrary convex and compact sets respectively. Despite its relatively slow convergence rate - additive error of the order $1/t$ after t iterations, the benefit of the method is twofold: i) its computational simplicity - each iteration is comprised of optimizing a linear objective over the set and ii) it is known to produce sparse solutions (for the simplex this means only a few non zeros entries, for the semidefinite cone this means that the solution has low rank). Due to these two properties, conditional gradient methods have attracted much attention in the machine learning community in recent years, see [53, 62, 52, 26, 42, 81, 64, 8].

It is known that in general the convergence rate $1/t$ is also optimal for this method without further assumptions, as shown in [18, 45, 52]. In case the objective function is both smooth and strongly convex, there exist extensions of the basic method which achieve faster rates under various assumptions. One such extension of the conditional-gradient algorithm with linear convergence rate was presented by Migdalas [68], however the algorithm requires to solve a regularized linear problem on each iteration which is computationally equivalent to computing projections. This is also the case with the algorithm for smooth and strongly convex optimization in the recent work of Lan [63]. In case the convex set is a polytope, GuéLat and Marcotte [41] has shown that the algorithm of Frank and Wolfe [31] converges in linear rate assuming that the optimal point in the polytope is bounded away from the boundary. The convergence rate is proportional to a quadratic of the distance of the optimal point from the boundary. We note that in case the optimum lies in the interior of the convex domain, then the problem is in fact an unconstrained convex optimization problem and solvable via much more efficient methods. GuéLat and Marcotte [41] also gave an improved algorithm based on the concept of "away steps" with a linear convergence rate that holds under weaker conditions, however this linear rate still depends on the location of the optimum with respect to the boundary of the set which may result in an arbitrarily bad convergence rate. We note that the suggestion of using "away steps" to accelerate the convergence of the FW algorithm for strongly convex objectives was already made by Wolfe himself in [92]. Beck and Taboule [10] gave a linearly converging conditional gradient algorithm for solving convex linear systems, but as in [41], their convergence rate depends on the distance of the optimum from

the boundary of the set. Here we emphasize that in this work we do not make any assumptions on the location of the optimum in the convex domain and our convergence rates are independent of it.

Ahipasaoglu, Sun and Todd [2] gave a variant of the conditional gradient algorithm with *away steps* that achieves a linear convergence rate for the specific case in which the convex domain is the unit simplex. Their work also does not specify the precise dependency of the convergence rate on parameters of the problem such as the dimension, which is of great importance. In this work we derive, as an illustrating example, a linearly converging algorithm for the unit simplex. Our generalization to arbitrary polytopes is highly non-trivial and is indeed the technical heart of this work. We also provide convergence rates with detailed dependencies on natural parameters of the problem.

After our work first appeared [34], Jaggi and Lacoste-Julien [61] presented a refined analysis of a variant of the conditional gradient algorithm with away steps from [41] that achieves a linear convergence rate without the assumption on the location of the optimum as in the original work of [41]. Their algorithm is also shown to be affine invariant. Their convergence rate however is not given explicitly and its dependency on the dimension or other natural parameters of the problem is not clear.

Conditional gradient-like methods for online, stochastic and non-smooth optimization The two closest works to ours are those of Kalai and Vempala[56] and Hazan and Kale [49], both present projection-free algorithms for online convex optimization in which the only optimization carried out by the algorithms on each iteration is the minimization of a single linear objective over the decision set. [56] gives a random algorithm for the online setting in the special case in which all loss functions are linear, also known as *online linear optimization*. In this setting their algorithm achieves regret of $O(\sqrt{T})$ which is optimal [14]. On iteration t their algorithm plays a point in the decision set that minimizes the cumulative loss on all previous iterations plus a vector whose entries are independent random variables. The work of [49] introduces algorithms for stochastic and online optimization which are based on ideas similar to ours - using the conditional gradient update step to approximate the steps a meta-algorithm for online convex optimization known as *Regularized Follow the Leader* (RFTL) [46, 80]. For stochastic op-

timization, in case that all loss functions are smooth they achieve an optimal convergence rate of $1/\sqrt{T}$, however for non-smooth stochastic optimization they only get convergence rate of $T^{-1/3}$ and for the full adversarial setting of online convex optimization they get suboptimal regret that scales like $T^{3/4}$.

In a recent work, Lan [63] showed how to apply the conditional gradient algorithm to offline non-smooth optimization via a well known smoothing technique (also employed in [49]). His analysis shows that an ϵ additive error is guaranteed after a total of $O(\epsilon^{-2})$ linear optimization steps over the domain and $O(\epsilon^{-4})$ calls to the subgradient oracle of the objective. Our algorithm for the non-smooth setting guarantees an ϵ additive error after $O(\epsilon^{-2})$ linear optimization steps over the domain and $O(\epsilon^{-2})$ calls to the subgradient oracle.

Also relevant to our work is the recent work of Harchaoui, Juditsky and Nemirovski [43] who give methods for i) minimizing a norm over the intersection of a cone and the level set of a convex smooth function and ii) minimizing the sum of a convex smooth function and a multiple of a norm over a cone. Their algorithms are extensions of the conditional gradient method that assume the availability of a stronger oracle that can minimize a linear objective over the intersection of the cone and a unit ball induced by the norm of interest. They present several problems of interest for which such an oracle could be implemented very efficiently, however in general such an oracle could be computationally much less efficient than the linear oracle required by standard conditional gradient methods.

Organization of this chapter

The rest of this chapter is organized as follows. In section 2.1 we give preliminaries, including notation and definitions that will be used throughout this chapter, examples for polytopes of interest, overview of the conditional gradient method, and describe the settings of online convex optimization and stochastic optimization. In section 2.2 we give an informal statement of the main results presented in this chapter. In section 2.3 we present our main result - a new linearly convergent conditional gradient algorithm for offline smooth and strongly convex optimization over polyhedral sets. In section 2.4 we present and analyze our main new algorithmic machinery which we refer to as a *local linear optimization oracle*. In section 2.5 we discuss

an application of our linearly convergent CG algorithm to the problem of Submodular Function Minimization. In section 2.6 we present and analyze our algorithms for online and stochastic optimization, and finally in section 2.7 we discuss a lower bound for the problem of minimizing a smooth and strongly convex function using only linear optimization steps - showing that the oracle complexity of our new algorithm presented in section 2.3 is nearly optimal.

2.1 Preliminaries

We denote by $\mathbb{B}_r(x)$ the euclidean ball of radius r centred at x. We denote by $\|x\|$ the ℓ_2 norm of the vector x and by $\|A\|$ the *spectral norm* of the matrix A, that is $\|A\| = \max_{x\in\mathbb{B}} \|Ax\|$. Given a matrix A, we denote by $A(i)$ the vector that corresponds to the ith row of A.

Definition 1. *We say that a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz with parameter L over the set $\mathcal{K}$ if for all $x, y \in \mathcal{K}$ it holds that*

$$|f(x) - f(y)| \leq L\|x - y\|.$$

Definition 2. *We say that a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is β-smooth over the set $\mathcal{K}$ if for all $x, y \in \mathcal{K}$ it holds that*

$$f(y) \leq f(x) + \nabla f(x) \cdot (y - x) + \frac{\beta}{2}\|x - y\|^2.$$

Definition 3. *We say that a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is σ-strongly convex over the set $\mathcal{K}$ if for all $x, y \in \mathcal{K}$ it holds that*

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x) + \frac{\sigma}{2}\|x - y\|^2.$$

The above definition together with first order optimality conditions imply that for a σ-strongly convex f, if x^* is the unique minimizer of f over $\mathcal{K}$, then for all $x \in \mathcal{K}$ it holds that

$$f(x) - f(x^*) \geq \frac{\sigma}{2}\|x - x^*\|^2. \tag{2.1}$$

Note that a sufficient condition for a twice-differential function f to be

β-smooth and σ-strongly convex over a domain $\mathcal{K}$ is that

$$\forall x \in \mathcal{K}: \quad \beta I \succeq \nabla^2 f(x) \succeq \sigma I.$$

Let $\mathcal{P}$ be a polytope described by linear equations and inequalities, i.e.

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid A_1 x = b_1,\ A_2 x \leq b_2\}, \tag{2.2}$$

where $A_2 \in \mathbb{R}^{m \times n}$. We assume without loss of generality that all rows of A_2 are scaled to have unit ℓ_2 norm.We denote by $\mathcal{V}(\mathcal{P})$ the set of vertices of $\mathcal{P}$, and we let $N = |\mathcal{V}|$. We now define several geometric parameters of $\mathcal{P}$ that will come up naturally in the analysis of our algorithms. We denote the Euclidean diameter of $\mathcal{P}$ by $D(\mathcal{P})$, i.e., $D(\mathcal{P}) = \max_{x,y \in \mathcal{P}} \|x - y\|$. We denote

$$\xi(P) = \min_{v \in \mathcal{V}(\mathcal{P})} \left(\min\{b_2(j) - A_2(j) \cdot v \mid j \in [m], A_2(j) \cdot v < b_2(j)\}\right).$$

That is, given an inequality constraint that defines the polytope and a vertex of the polytope, the vertex either satisfies the constraint with equality or is at least $\xi(P)$-far from satisfying it with equality. Let $r(A_2)$ denote the row-rank of the matrix A_2. Let $\mathbb{A}(\mathcal{P})$ denote the set of all $r(A_2) \times n$ matrices whose rows are linearly independent vectors chosen from the rows of A_2 and denote $\psi(\mathcal{P}) = \max_{M \in \mathbb{A}(\mathcal{P})} \|M\|$. Finally denote $\mu(\mathcal{P}) = \frac{\psi(\mathcal{P}) D(\mathcal{P})}{\xi(\mathcal{P})}$. It is important to note that the quantity $\mu(\mathcal{P})$ is invariant to translation, rotation and scaling of the polytope $\mathcal{P}$. Note also that it always holds that $\mu(\mathcal{P}) \geq 1$ (this follows since by definition $\psi(\mathcal{P}) \geq 1$ and $\xi(\mathcal{P}) \leq D(\mathcal{P})$). Henceforth we shall use the shorthand notation of $\mathcal{V}, D, \xi, \psi, \mu$ when the polytope considered is clear from context. Note that in many settings of interest (problems for which there is indeed an highly-efficient algorithm for linear optimization over the specified polytope), estimating the parameters ξ, ψ is straightforward. For instance, in convex domains that arise in combinatorial optimization such at the flow polytope, matching polytope, matroid polytopes, etc.

Throughout this chapter we will assume that we have access to an oracle that returns a vertex of $\mathcal{P}$ that minimizes the dot product with a given linear objective. That is we are given a procedure $\mathcal{O}_{\mathcal{P}} : \mathcal{V} \to \mathbb{R}$ such that for all $c \in \mathbb{R}^n$, $\mathcal{O}_{\mathcal{P}}(c) \in \arg\min_{v \in \mathcal{V}} v \cdot c$. We call $\mathcal{O}_{\mathcal{P}}$ a linear optimization oracle.

2.1.1 Examples of polytopes

We now turn to briefly discuss some common polyhedral sets. In particular we discuss their geometry in terms of the parameters ξ, ψ, μ, D that we have just defined, and the complexity of performing linear minimization over them. We emphasize that our methods work for *any* polytope and that the polytopes discussed here are brought only as an example.

Probabilistic Simplex The simplex in $\mathbb{R}^n$ is the set of all distributions over n elements, i.e. the set:

$$\mathcal{S}_n = \{x \in \mathbb{R}^n \,|\, \forall i \in [n] : \, x_i \geq 0 \,, \sum_{i=1}^{n} x_i = 1\}.$$

Alternatively, $\mathcal{S}_n$ is the convex hull of all standard basis vectors in $\mathbb{R}^n$.

$\mathcal{S}_n$ could be written in the form of (2.2) as follows:

$$\mathcal{S}_n = \{x \in \mathbb{R}^n \,|\, -Ix \leq \vec{0} \,, \, x \cdot \vec{1} = 1\},$$

where I is the identity matrix and $\vec{0}, \vec{1}$ are the all-zeros and all-ones vectors, respectively.

Since the inequalities defining $\mathcal{S}_n$ are of the form $-Ix \leq \vec{0}$ and all vertices of $\mathcal{S}_n$ are in $\{0,1\}^n$, it immediately follows that $\xi = 1$, $\psi = 1$. Also it is easy to see that $D = \sqrt{2}$. Thus it follows that for the simplex $\mu = \sqrt{2}$.

Linear minimization over the simplex is highly trivial: since the vertices of $\mathcal{S}_n$ are exactly the standard basis vectors of $\mathbb{R}^n$, it follows that given a linear objective $c \in \mathbb{R}^n$, its minimizer over $\mathcal{S}_n$ (generally there could be more than one) is the standard basis vector that corresponds to the smallest (signed) entry in c.

Hypercube The hypercube in $\mathbb{R}^n$ is the set:

$$\mathcal{C}_n = \{x \in \mathbb{R}^n \,|\, \forall i \in [n] : \, 1 \geq x_i \geq 0\}.$$

$\mathcal{C}_n$ could be written in the form of (2.2) as follows:

$$\mathcal{C}_n = \{x \in \mathbb{R}^n \,|\, -Ix \leq \vec{0} \,, \, Ix \leq \vec{1}\}.$$

As in the case of the simplex, since the inequalities are of the form $-Ix \leq \vec{0}, Ix \leq \vec{1}$, and all vertices of $\mathcal{S}_n$ are in $\{0,1\}^n$, it immediately follows that $\xi = 1$, $\psi = 1$. Also it is easy to see that $D = \sqrt{n}$. Thus it follows that for the hypercube $\mu = \sqrt{n}$.

Linear minimization over the hypercube is also highly trivial: since the set of vertices of $\mathcal{C}_n$ is exactly the set $\{0,1\}^n$, it follows that given a linear objective $c \in \mathbb{R}^n$, its minimizer over $\mathcal{C}_n$ (generally there could be more than one) is given by $-\text{sign}(c)$, where $\text{sign} : \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued function that sets each entry in the output to the sign (either 1 or -1) of the corresponding entry in the input.

Flow polytope Let G be a *directed acyclic graph* (DAG) with a set of vertices V such that $|V| = n$, and a set of edges E such that $|E| = m$, and let s, t be two vertices in V which we refer to as the *source* and the *target*, respectively. The $s-t$ flow polytope, denoted here by $\mathcal{F}_{st}$, is the set of all unit $s-t$ flows in G, where for each point $x \in \mathcal{F}_{st}$ and $i \in [m]$, the entry x_i is the amount of flow through edge i according to the flow x. $\mathcal{F}_{st}$ is also known as the $s-t$ *path polytope* since it is the convex hull of all identifying vectors of paths from s to t in the graph G. Since it is the set of all $s-t$ unit flows, the polytope $\mathcal{F}_{st}$ could be described in the following standard way:

$$\mathcal{F}_{st} = \{x \in \mathbb{R}^m \mid -Ix \leq \vec{0},\, x \cdot \vec{1}_{\to t} = 1,\, \forall u \in V \setminus \{s,t\} : (\vec{1}_{\to u} - \vec{1}_{\leftarrow u}) \cdot x = 0\},$$

where for any vertex $u \in V$ we denote by $\vec{1}_{\to u}$ the vector in $\mathbb{R}^m$ for which there is 1 in each entry that corresponds to an edge going into u in the graph, and 0 otherwise. $\vec{1}_{\leftarrow u}$ is defined in the same way for edges going out of u.

As in previous examples, the inequalities defining $\mathcal{F}_{st}$ are of the form $-Ix \leq \vec{0}$ and all vertices of $\mathcal{F}_{st}$ are in $\{0,1\}^m$ (since, as discussed, it is the convex hull of all $s-t$ paths in the graph). Thus, as before, it follows that $\xi = \psi = 1$. Also, it easily follows that $D < \sqrt{2n}$, and thus we have that $\mu \leq \sqrt{2n}$.

Since $\mathcal{F}_{st}$ is the convex hull of paths, linear minimization is straight forward: given a linear objective $c \in \mathbb{R}^m$, we need to find the identifying vector of the lightest $s-t$ path in G with respect to the edge weights induced by c. Since the graph G is a DAG, this could be carried out in $O(m)$ time

[79].

Matching polytope (Birkhoff polytope) Let G be a *bipartite* graph with a set of vertices $V = L \cup R$ such that $|R| = |L| = n$. Assume G is fully connected, i.e. for every two vertices $u \in L, v \in R$, there is an edge in the graph connecting u and v. The matching polytope, denoted here by $\mathcal{M}$, is then the convex hull of all identifying vectors of perfect matchings in G. Note that $\mathcal{M} \subset \mathbb{R}^m$ and the vertices of $\mathcal{M}$ are vectors in $\{0,1\}^m$, where $m = n^2$ is the number of edges in the graph.

The matching polytope is equivalent to the *Birkhoff polytope*, which is the set of all $n \times n$ doubly stochastic matrices, i.e. matrices with non-negative real entries whose entries along any row and any column add up to 1. In correspondence with the vertices of the matching poltyope, the Birkhoff polytope is the convex hull of all $n \times n$ *permutation matrices*, i.e. matrices whose any row and any column contain exactly one non-zero entry which is set to 1 (this is also known as the *Birkhoff-von Neumann Theorem*).

Since the matching polytope is equivalent to the polytope of doubly stochastic matrices, it is easily described in algebraic form:

$$\mathcal{M} = \{x \in \mathbb{R}^m \mid -Ix \leq \vec{0}, \forall u \in L : \vec{1}_u \cdot x = 1, \forall u \in R : \vec{1}_u \cdot x = 1\},$$

where $\vec{1}_u$ denotes the vector in $\{0,1\}^m$ in which all entries that correspond to edges connected to the vertex u are set to 1, and all other edges are set to 0.

The inequalities defining $\mathcal{M}$ are again of the form $-Ix \leq \vec{0}$ and, as discussed, all vertices of $\mathcal{M}$ are in $\{0,1\}^m$. Thus, as before, it follows that $\xi = \psi = 1$. Also, it easily follows that $D \leq \sqrt{2n}$ and thus we have that $\mu \leq \sqrt{2n}$.

In order to do linear minimization over $\mathcal{M}$, given a linear objective $c \in \mathbb{R}^m$, we need to find a minimum-weight perfect matching in a fully connected bipartite graph, where the edge weights are induced by c. There is a well known algorithm for this problem known as the *Hungarian algorithm* which runs in $O(n^3)$ time [79].

Matroid Polytopes Let $E = [n]$ and let $\mathcal{I}$ be a set of subsets of E, also known as the set of *independent sets*, such that $(E, \mathcal{I})$ is a matroid. For a

detailed introduction to matroids and their proprieties the reader is referred to [79]. The polytope assosiated with the matroid $(E, \mathcal{I})$, denoted here by $\mathcal{M}_{\mathcal{I}}$, is the convex hull of all identifying vectors of the subsets in $\mathcal{I}$. It thus holds that $\mathcal{M}_{\mathcal{I}} \subset \mathbb{R}^n$ and that the vertices of $\mathcal{M}_{\mathcal{I}}$ are in $\{0,1\}^n$.

Matroids and their polytopes have important applications in many combinatorial optimization problems. Examples include spanning trees-related problems and *Submodular Function Minimization* (SFM) (see [79] for a more detailed list). For a connection of this work with SFM see also Section 2.5.

In order to describe $\mathcal{M}_{\mathcal{I}}$ in algebraic form we need to introduce the notion of a rank function of a matroid $r : 2^E \to \mathbb{N}$. For any subset $S \subseteq E$, $r(S)$ is the size of the largest independet set $M \in \mathcal{I}$ that is fully contained in S.

The polytope $\mathcal{M}_{\mathcal{I}}$ could be described as follows (see also [79]):

$$\mathcal{M}_{\mathcal{I}} = \{x \in \mathbb{R}^n \mid -Ix \leq \vec{0}, \forall S \in 2^E : \vec{1}_S \cdot x \leq r(S)\},$$

where $\vec{1}_S$ is the identifying vector of the set S. In Section 2.5 we show that for $\mathcal{M}_{\mathcal{I}}$ it holds that $\xi = 1$ and $\psi \leq n$. It also clearly holds that $D \leq \sqrt{n}$. Thus, it follows that $\mu \leq n^{3/2}$.

Given a linear objective $c \in \mathbb{R}^n$, there is a well known *greedy* algorithm for linear minimization over $\mathcal{M}_{\mathcal{I}}$ which amounts to sorting the elements in E according to their weight in c in increasing order, and adding them to the current solution, as long as it remains an independent set in $\mathcal{I}$, and the total weight decreases. This algorithm runs in $O(n \log n)$ time under the standard assumption that $\mathcal{I}$ is given by an *independence oracle* that can answer in $O(1)$ time whether a subset S is in $\mathcal{I}$ or not. See [79] for details.

2.1.2 The conditional gradient method and local linear optimization oracles

The conditional gradient method is a simple algorithm for minimizing a smooth and convex function f over a convex set $\mathcal{P}$ - which in this chapter we assume to be a polytope. The appeal of the method is that it is a first order *feasible point* method, i.e., the iterates always lie inside the convex set and thus no projections are needed. Further more, the update step on each iteration simply requires to minimize a linear objective over the set. The basic algorithm is given below.

Algorithm 1 Conditional Gradient

Input: sequence of step sizes $\{\alpha_t\}_{t=1}^{\infty} \subseteq [0,1]$
Let x_1 be an arbitrary point in $\mathcal{P}$.
for $t = 1, 2, \ldots$ **do**
  $p_t \leftarrow \mathcal{O}_{\mathcal{P}}(\nabla f(x_t))$
  $x_{t+1} \leftarrow x_t + \alpha_t(p_t - x_t)$
end for

Let x^* denote the unique minimizer of f over $\mathcal{P}$ that is, $x^* = \arg\min_{x \in \mathcal{K}} f(x)$. The convergence of algorithm 1 is due to the following simple observations.

$$\begin{aligned} & f(x_{t+1}) - f(x^*) \qquad (2.3) \\ &= f(x_t + \alpha_t(p_t - x_t)) - f(x^*) \\ &\leq f(x_t) - f(x^*) + \alpha_t(p_t - x_t) \cdot \nabla f(x_t) + \tfrac{\alpha_t^2 \beta}{2}\|p_t - x_t\|^2 \quad / \ \beta\text{-smoothness of } f \\ &\leq f(x_t) - f(x^*) + \alpha_t(x^* - x_t) \cdot \nabla f(x_t) + \tfrac{\alpha_t^2 \beta}{2}\|p_t - x_t\|^2 \quad / \ \text{optimality of } p_t \\ &\leq f(x_t) - f(x^*) + \alpha_t(f(x^*) - f(x_t)) + \tfrac{\alpha_t^2 \beta}{2}\|p_t - x_t\|^2 \quad / \ \text{convexity of } f. \end{aligned}$$

Thus for an appropriate choice for the sequence of step sizes $\{\alpha_t\}_{t=1}^{\infty}$, the approximation error strictly decreases on each iteration. This leads to the following theorem. For a proof see for instance the modern survey of [52], or alternatively, see Theorem 11 in Section 3.1.2.

Theorem 3. *There is an explicit choice for the sequence of step sizes $\{\alpha_t\}_{t=1}^{\infty}$ such that for every $t \geq 2$, the iterate x_t of Algorithm 1 satisfies that $f(x_t) - f(x^*) = O\left(\frac{\beta D^2}{t-1}\right)$.*

The relatively slow convergence of the conditional gradient algorithm is due to the term $\|p_t - x_t\|$ in Eq. (2.3), that may remain as large as the diameter of $\mathcal{P}$ while the term $f(x_t) - f(x^*)$ keeps on shrinking, that forces choosing values of α_t that decrease like $\frac{1}{t}$ in order to guarantee convergence [18, 45, 52].

Notice that if $f(x)$ is σ-strongly convex for some $\sigma > 0$ then according to Eq. (2.1), knowing that for some iteration t it holds that $f(x_t) - f(x^*) \leq \epsilon$, implies that $\|x_t - x^*\|^2 \leq \frac{2\epsilon}{\sigma}$. Thus when choosing the point p_t, denoting $r = \sqrt{2\epsilon/\sigma}$, it is enough to consider points that lie in the intersection set

$\mathcal{P} \cap \mathbb{B}_r(x_t)$, i.e., take p_t to be the solution to the optimization problem

$$\min_{p \in \mathcal{P} \cap \mathbb{B}_r(x_t)} p \cdot \nabla f(x_t). \tag{2.4}$$

In this case the term $\|p_t - x_t\|^2$ in Eq. (2.3) will be of the same magnitude as $f(x_t) - f(x^*)$ (or even smaller) and as observable in Eq. (2.3), a linear convergence rate will follow.

However, solving Problem (2.4) is potentially much more difficult than solving the original linear problem $\min_{p \in \mathcal{P}} p \cdot \nabla f(x_t)$, and is not straightforward solvable using the linear optimization oracle of $\mathcal{P}$.

To overcome the problem of solving the linear problem in the intersection $\mathcal{P} \cap \mathbb{B}_r(x_t)$ we introduce the following definition which is a primary ingredient of this chapter.

Definition 4 (Local Linear Optimization Oracle)**.** *We say that a procedure $\mathcal{A}(x, r, c)$, where $x \in \mathcal{P}$, $r \in \mathbb{R}^+$, $c \in \mathbb{R}^n$, is a Local Linear Optimization Oracle with parameter $\rho \geq 1$ for the polytope $\mathcal{P}$, if $\mathcal{A}(x, r, c)$ returns a feasible point $p \in \mathcal{P}$ such that:*

1. *$\forall y \in \mathbb{B}(x, r) \cap \mathcal{P}$ it holds that $y \cdot c \geq p \cdot c$.*

2. *$\|x - p\| \leq \rho \cdot r$.*

The local linear optimization oracle (LLOO) relaxes Problem (2.4) by solving the linear problem on a larger set, but one that still has a diameter that is not much larger than $\sqrt{f(x_t) - f(x^*)}$. Our main contribution is in showing that for a polytope $\mathcal{P}$, a LLOO can be constructed such that the parameter ρ depends only on the dimension n and the quantity $\mu(\mathcal{P})$. Moreover, the algorithmic construction requires only a single call to the original linear optimization oracle $\mathcal{O}_{\mathcal{P}}$. Hence, the complexity per iteration, in terms of the number of calls to the linear optimization oracle $\mathcal{O}_{\mathcal{P}}$, remains the same as the original conditional gradient algorithm (Algorithm 1). Figure 2.1 gives some intuition why a construction of such an oracle is possible for polyhedral sets.

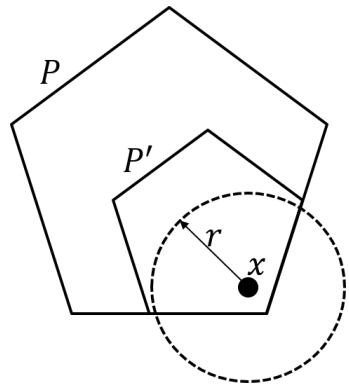

Figure 2.1: Given a polytope $\mathcal{P}$, a feasible point x and a radius $r > 0$, if r is small enough, then it is always possible to fit a scaled-down version of $\mathcal{P}$, denoted by $\mathcal{P}'$, such that on one hand $\mathcal{P}'$ is fully contained in $\mathcal{P}$ and on the other hand $\mathcal{P}'$ fully contains the intersection of $\mathcal{P}$ with the ball of radius r centered at x. Given a linear objective $c \in \mathbb{R}^n$, we can take the vertex of $\mathcal{P}'$ that minimizes the dot product with c to be the output of a LLOO for $\mathcal{P}$ queried with the input (x, r, c).

2.1.3 Online convex optimization and its application to stochastic and offline optimization

The problem of online convex optimization (OCO) [94, 47, 46] takes the form of the following repeated game. A decision maker is required on each iteration t of the game to choose a point $x_t \in \mathcal{K}$, where $\mathcal{K}$ is a fixed convex set. After choosing the point x_t, a convex loss function $f_t(x)$ is reveled, and the decision maker incurs loss $f_t(x_t)$. The emphasis in this model is that the loss function on time t may be chosen completely arbitrarily and even in an adversarial manner given the current and past decisions of the decision maker. In the *full information* setting, after making his decision on time t, the decision maker gets full knowledge of the function f_t. In the *partial information* setting (*bandit*) the decision maker only learns the value $f_t(x_t)$ and does not gain any other knowledge about f_t.

The standard goal in this setting is to have overall loss which is not much larger than that of the best fixed point in $\mathcal{K}$, in hindsight. Formally the goal

is to minimize a quantity known has *regret* which is given by

$$\text{regret}_T := \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x).$$

In certain cases, such as in the bandit setting, the decision maker must use randomness in order to make his decisions. In this case we consider the expected regret, where the expectation is taken over the randomness in the algorithm of the decision maker.

In the *full information setting* and for general convex losses the optimal regret bound attainable scales like $\sqrt{T}$ [14] where T is the length of the game. In the case that all loss functions are strongly convex, the optimal regret bound attainable scales like $\log(T)$ [48].

Algorithms for OCO

A simple algorithm that attains optimal regret of $O(\sqrt{T})$ for general convex losses is known as the Regularized Follows The Leader algorithm (RFTL) [46]. On time t the algorithm predicts according to the following rule.

$$x_t \leftarrow \arg\min_{x \in \mathcal{K}} \left\{ \eta \sum_{\tau=1}^{t-1} \nabla f_\tau(x_\tau) \cdot x + \mathcal{R}(x) \right\}. \tag{2.5}$$

Where η is a parameter known as the learning rate and $\mathcal{R}$ is a strongly convex function known as the regularization. From an offline optimization point of view, achieving low regret is thus equivalent to minimizing a single strongly-convex objective over the feasible set per iteration. In fact, with the popular choice $\mathcal{R}(x) = \|x\|^2$, we get that Problem (2.5) is just the minimization of a function that is both smooth and strongly-convex over the feasible domain $\mathcal{K}$, and is in fact equivalent to computing an Euclidean projection onto $\mathcal{K}$.

In case of strongly-convex losses a slight variant of Eq. (2.5), which also takes the form of minimizing a smooth and strongly convex function when choosing $\mathcal{R}(x) = \|x\|^2$, guarantees optimal $O(\log(T))$ regret.

In the *partial information setting* the RFTL rule (2.5) with the algorithmic conversion of the *bandit* problem to that of the *full information* problem established in [30], yields an algorithm with regret $O(T^{3/4})$, which is the best

to date.

Our algorithms for online optimization are based on iteratively approximating the RFTL objective in Eq. (2.5) using our new linearly convergent CG algorithm for smooth and strongly convex optimization, thus replacing the projection step in (2.5) (in case $\mathcal{R}(x) = \|x\|^2$) with a single linear optimization step over the domain.

We note that while the update rule in Eq. (2.5) uses the gradients of the loss functions which are denoted by ∇f_τ, it is in fact not required to assume that the loss functions are differentiable everywhere in the domain. It suffices to assume that the loss functions only have a sub-gradient everywhere in the domain, making the algorithm suitable also for non-smooth settings. Throughout this chapter we do not differentiate between these two cases and the notation $\nabla f(x)$ should be understood as a gradient of f at the point x in case f is differentiable and as a sub-gradient of f in case f only has a sub-gradient in this point.

Stochastic optimization

In stochastic optimization the goal is to minimize a convex function $F(x)$ given by

$$F(x) = \mathbb{E}_{f\sim\mathcal{D}}[f(x)],$$

where $\mathcal{D}$ is a fixed, yet unknown distribution over convex functions. In this setting we don't have direct access to the function F, instead we assume to have a stochastic oracle for F that when queried, returns a function f sampled from $\mathcal{D}$, independently of previous samples.

The general setting of online convex optimization is harder than stochastic optimization in the sense that an algorithm for OCO could be directly applied to stochastic optimization as follows. We simulate an online game of T rounds for the OCO algorithm, where on each iteration t the loss function $f_t(x)$ is generated by a query to the stochastic oracle of $\mathcal{D}$. Let us denote by regret_T an upper bound on the regret of the online algorithm with respect to *any* sample of T functions from the distribution $\mathcal{D}$. Thus, given such a

sample - $\{f_t\}_{t=1}^T$, it holds that

$$\sum_{t=1}^T f_t(x_t) - \min_{x\in\mathcal{K}} \sum_{t=1}^T f_t(x) \leq \text{regret}_T.$$

Denoting $x^* \in \arg\min_{x\in\mathcal{K}} F(x)$ we thus in particular have that

$$\sum_{t=1}^T f_t(x_t) - \sum_{t=1}^T f_t(x^*) \leq \text{regret}_T.$$

Since for all $t \in [T]$ it holds that $\mathbb{E}[f_t(x_t)|x_t] = F(x_t)$ and $\mathbb{E}[f_t(x^*)] = F(x^*)$ (where in both cases the expectation is with respect to the random choice of f_t), taking expectation over the randomness of the oracle for F we have that

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^T f_t(x_t) - \sum_{t=1}^T f_t(x^*)\right] &= \sum_{t=1}^T \mathbb{E}\left[\mathbb{E}[f_t(x_t)|x_t]\right] - T\cdot F(x^*) \\
&= \sum_{t=1}^T \mathbb{E}[F(x_t)] - T\cdot F(x^*) \\
&= \mathbb{E}\left[\sum_{t=1}^T F(x_t)\right] - T\cdot F(x^*).
\end{aligned}$$

Denoting $\bar{x} = \frac{1}{T}\sum_{t=1}^T x_t$ we have by convexity of F that

$$\mathbb{E}[F(\bar{x})] - F(x^*) \leq \frac{\text{regret}_T}{T}.$$

Thus the same regret rates that are attainable for online convex optimization hold as convergence rates, or sample complexity, for stochastic convex optimization. We note that using standard concentration results for martingales, one can also derive error bounds that hold with high probability and not only in expectation, but these are beyond the scope of this chapter. We refer the interested reader to [13] for more details.

Non-smooth optimization

As in stochastic optimization (see previous subsection), an algorithm for OCO also implies an algorithm for offline convex optimization. Thus a conditional gradient-like algorithm for OCO implies a conditional gradient-like algorithm for non-smooth convex optimization. This is in contrast to the original conditional gradient method which is suitable for smooth optimization only.

Applying an OCO algorithm to the minimization of a, potentially non-smooth, convex function $f(x)$ over a feasible convex set $\mathcal{K}$, is as follows. As in the previous subsection, we simulate a game of length T for the OCO algorithm in which the loss function f_t on each round is just the function to minimize $f(x)$. As in the stochastic case, denoting $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$, i.e., the average of iterates returned by the online algorithm, we have that

$$f(\bar{x}) - f(x^*) \leq \frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^*) = \frac{1}{T}\left(\sum_{t=1}^{T} f_t(x_t) - f_t(x^*)\right) = \frac{\text{regret}_T}{T},$$

where the first inequality follows from convexity of f. Hence the regret bound immediately translates to a convergence rate for offline optimization problem.

2.2 Results in this Chapter

In this section we give an informal presentation of the main results presented in this chapter. In all of the following results we consider optimization (either offline or online) over a polytope, denoted $\mathcal{P}$, and we assume the availability an oracle $\mathcal{O}_{\mathcal{P}}$ that given a linear objective $c \in \mathbb{R}^n$ returns a vertex of $\mathcal{P}$, $v \in \mathcal{V}$ that minimizes the dot product with c over $\mathcal{P}$.

Offline smooth and strongly convex optimization Given a β-smooth, σ-strongly convex function $f(x)$ we present an iterative algorithm that after t iterations returns a point $x_{t+1} \in \mathcal{P}$ such that

$$f(x_{t+1}) - f(x^*) \leq C \exp\left(-\frac{\sigma}{4\beta n \mu^2} t\right), \tag{2.6}$$

where $x^* = \arg\min_{x\in\mathcal{P}} f(x)$ and C satisfies that $C \geq f(x_1) - f(x^*)$. Each iteration is comprised of a single call to the linear optimization oracle of $\mathcal{P}$ and a single evaluation of a gradient vector of f.

As we show in section 2.7, the above convergence rate is nearly tight in certain settings for a conditional gradient-like method.

Online convex optimization We present algorithms for OCO that require only a single call to the linear optimization oracle of $\mathcal{P}$ per iteration of the game. In the following we let G denote an upper bound on the ℓ_2 norm of the (sub)gradients of the loss functions revealed throughout the game. Our results for the online setting are as follows:

1. An algorithm for OCO with arbitrary convex loss functions whose sequence of predictions - $\{x_t\}_{t=1}^T$ satisfies that

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{P}} \sum_{t=1}^{T} f_t(x) = O\left(GD\mu\sqrt{nT}\right). \tag{2.7}$$

   This bound is optimal in terms of T [14].

2. An algorithm for OCO with σ-strongly convex loss functions whose sequence of predictions - $\{x_t\}_{t=1}^T$ satisfies that

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{P}} \sum_{t=1}^{T} f_t(x) = O\left(\sigma D^2\rho^4 + \frac{(G+\sigma D)^2 n\mu^2}{\sigma}\log(T)\right). \tag{2.8}$$

   This bound is also optimal in terms of T [48].

3. A randomized algorithm for the *partial information* setting whose sequence of predictions - $\{x_t\}_{t=1}^T$ satisfies that

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{P}} \sum_{t=1}^{T} f_t(x)\right] = O\left(GD\sqrt{\frac{nD}{r_0}}T^{3/4} + GD\mu\sqrt{nT}\right). \tag{2.9}$$

   Here we assume for simplicity that $\mathcal{P}$ is full-dimensional and we denote by r_0 the size of the largest Euclidean ball enclosed in it. This bound matches the current state-of-the-art in this setting in terms of T [30].

Stochastic and non-smooth optimization Applying our online algorithms to the stochastic setting, as specified in Subsection 2.1.3, yields algorithms that given a stochastic oracle for a function of the form - $F(x) = \mathbb{E}_{f\sim\mathcal{D}}[f(x)]$, and after viewing T i.i.d samples from the distribution $\mathcal{D}$ and making T calls to the linear optimization oracle $\mathcal{O}_{\mathcal{P}}$, return a point $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ such that the following guarantees hold:

1. If $\mathcal{D}$ is a distribution over arbitrary convex functions then

$$\mathbb{E}[F(\bar{x})] - \min_{x\in\mathcal{P}} F(x) = O\left(\frac{GD\sqrt{n}\mu}{\sqrt{T}}\right). \tag{2.10}$$

2. If $\mathcal{D}$ is a distribution over σ-strongly convex functions then

$$\mathbb{E}[F(\bar{x})] - \min_{x\in\mathcal{P}} F(x) = O\left(\frac{\sigma^2 D^2 \rho^4 + (G+\sigma D)^2 n\mu^2 \log(T)}{\sigma T}\right). \tag{2.11}$$

Here again G denotes an upper bound on the ℓ_2 norm of the (sub)gradients of the functions f sampled from the distribution $\mathcal{D}$.

As described in Subsection 2.1.3, the above rates (without the expectation) hold also for non-smooth convex and strongly convex optimization.

2.3 A Linearly Convergent Conditional Gradient Algorithm

In this section we consider the following offline optimization problem.

$$\min_{x\in\mathcal{P}} f(x), \tag{2.12}$$

where we assume that f is β-smooth and σ-strongly convex, and $\mathcal{P}$ is a polytope. We further assume that we have a LLOO oracle for $\mathcal{P}$ - $\mathcal{A}(x, r, c)$, as defined in Subsection 2.1.2 . In section 2.4 we show that given an oracle for linear minimization over $\mathcal{P}$, such a LLOO oracle could be efficiently constructed.

The algorithm is given below.

Algorithm 2 LLOO-based Convex Optimization

Input: $\mathcal{A}(x, r, c)$ - LLOO with parameter $\rho \geq 1$ for polytope $\mathcal{P}$
Let x_1 be an arbitrary vertex of $\mathcal{P}$ and let $C \geq f(x_1) - f(x^*)$
$\alpha \leftarrow \frac{\sigma}{2\beta\rho^2}$
for $t = 1, 2, \ldots$ **do**
  $r_t \leftarrow \sqrt{\frac{2C}{\sigma} e^{-\frac{\alpha}{2}(t-1)}}$
  $p_t \leftarrow \mathcal{A}(x_t, r_t, \nabla f(x_t))$
  $x_{t+1} \leftarrow x_t + \alpha(p_t - x_t)$
end for

Theorem 4. *Algorithm 2, instanciated with the LLOO implementation given in Algorithm 4 (for which $\rho = \sqrt{n}\mu$, see Section 2.4), satisfies that for each $t \geq 1$, the iterate x_{t+1} is feasible ($x_{t+1} \in \mathcal{P}$) and*

$$f(x_{t+1}) - f(x^*) \leq C \exp\left(-\frac{\sigma}{4\beta n \mu^2} t\right),$$

where $x^* = \arg\min_{x \in \mathcal{P}} f(x)$. Furthermore, after t iterations the algorithm has made a total of t calls to the linear optimization oracle of $\mathcal{P}$ and t gradient vector evaluations of $f(x)$.

The theorem is a consequence of the following Lemma 2 and Lemma 8 (see Section 2.4). Lemma 2 proves the convergence rate of the algorithm given a black-box access to a LLOO with some arbitrary parameter ρ. Lemma 8 then gives an explicit construction of a LLOO with parameter $\rho = \sqrt{n}\mu$ that requires only a single call to the linear optimization oracle per invocation.

We now turn to analyze the convergence rate of Algorithm 2. The following lemma is of general interest and will be also used in the section on online optimization.

Lemma 1. *Assume that $f(x)$ is β-smooth and let $x^* \in \arg\min_{x \in \mathcal{P}} f(x)$. Assume that on iteration t it holds that $\|x_t - x^*\| \leq r_t$, and let $x_{t+1} \leftarrow x_t + \alpha(p_t - x_t)$, where p_t is the output of a LLOO with parameter ρ with respect to the input $(x_t, r_t, \nabla f(x_t))$, and let $\alpha \in [0, 1]$. Then it holds that*

$$f(x_{t+1}) - f(x^*) \leq (1 - \alpha)\left(f(x_t) - f(x^*)\right) + \frac{\beta}{2}\alpha^2 \min\{\rho^2 r_t^2, D^2\}.$$

Proof. By the β-smoothness of $f(x)$ and the definition of x_{t+1} we have that

$$\begin{aligned} f(x_{t+1}) &= f(x_t + \alpha(p_t - x_t)) \\ &\leq f(x_t) + \alpha(p_t - x_t) \cdot \nabla f(x_t) + \frac{\beta}{2}\alpha^2 \|p_t - x_t\|^2. \end{aligned}$$

Since $\|x_t - x^*\| \leq r_t$, by the definition of the oracle $\mathcal{A}$ it holds that i) $p_t \cdot \nabla f(x_t) \leq x^* \cdot \nabla f_t(x_t)$ and ii) $\|x_t - p_t\| \leq \min\{\rho r_t, D\}$. Thus we have that

$$f(x_{t+1}) \leq f(x_t) + \alpha(x^* - x_t) \cdot \nabla f(x_t) + \frac{\beta}{2}\alpha^2 \min\{\rho^2 r_t^2, D^2\}.$$

Using the convexity of $f(x)$ and subtracting $f(x^*)$ from both sides we have,

$$f(x_{t+1}) - f(x^*) \leq (1-\alpha)\left(f(x_t) - f(x^*)\right) + \frac{\beta}{2}\alpha^2 \min\{\rho^2 r_t^2, D^2\}.$$

□

Lemma 2. *[Convergence of Algorithm 2] Denote $h_t = f(x^*) - f(x_t)$. Then for all $t \geq 1$ it holds that*

$$h_t \leq C e^{-\frac{\sigma}{4\beta\rho^2}(t-1)}.$$

Proof. The proof is by a simple induction. For $t = 1$ we have by definition that $h_1 = f(x^*) - f(x_1) \leq C$.

Now assume that the lemma holds for $t \geq 1$. This implies via the the strong convexity of $f(x)$ (see Eq. 2.1) that

$$\|x_t - x^*\|^2 \leq \frac{2}{\sigma} h_t \leq \frac{2C}{\sigma} e^{-\frac{\sigma}{4\beta\rho^2}(t-1)},$$

where the second inequality follows from the induction hypothesis.

Thus, for $r_t = \sqrt{\frac{2C}{\sigma} e^{-\frac{\sigma}{4\beta\rho^2}(t-1)}}$ we have that $x^* \in \mathcal{P} \cap \mathbb{B}_{r_t}(x_t)$. Applying Lemma 1 with respect to x_t, r_t and using the induction hypothesis we have

that

$$\begin{aligned} h_{t+1} &\leq (1-\alpha)h_t + \frac{\beta}{2}\alpha^2 \min\{\rho^2 r_t^2, D^2\} \\ &\leq (1-\alpha)Ce^{-\frac{\sigma}{4\beta\rho^2}(t-1)} + \frac{\alpha^2\beta\rho^2}{\sigma}Ce^{-\frac{\sigma}{4\beta\rho^2}(t-1)} \\ &= Ce^{-\frac{\sigma}{4\beta\rho^2}(t-1)}(1-\alpha+\frac{\alpha^2\beta\rho^2}{\sigma}). \end{aligned}$$

By plugging the value of α from Algorithm 2 and using $(1-x) \leq e^{-x}$ we have that

$$h_{t+1} \leq Ce^{-\frac{\sigma}{4\beta\rho^2}t}.$$

□

2.4 Construction of a Local Linear Optimization Oracle

In this section we present an efficient construction of a Local Linear Optimization Oracle for a polytope $\mathcal{P}$, given only an oracle for minimizing a linear objective over $\mathcal{P}$.

As an exposition for our construction for arbitrary polytopes, we first consider the specific case of constructing a LLOO for the probabilistic simplex in $\mathbb{R}^n$, that is the set $\mathcal{S}_n = \{x \in \mathbb{R}^n \,|\, \forall i \in [n] : x_i \geq 0\,,\, \sum_{i=1}^n x_i = 1\}$. Then we show how to generalize the simplex case to an arbitrary polytope.

2.4.1 Construction of a local linear optimization oracle for the probabalistic simplex

The following lemma shows that in the case of the probabilistic simplex, an LLOO could be implemented by minimizing a linear objective over the intersection of the simplex and an ℓ_1 ball. We then show that this problem could be solved optimally by minimizing a single linear objective over the simplex (without the additional ℓ_1 constraint).

Lemma 3. *Given a point $x \in \mathcal{S}_n$, a radius $r > 0$ and a linear objective*

$c \in \mathbb{R}^n$, *consider the optimization problem*

$$\begin{aligned} &\min_{y \in \mathcal{S}_n} y \cdot c \\ s.t. \quad &\|x - y\|_1 \leq d, \end{aligned} \tag{2.13}$$

for some $d > 0$. *Let us denote by* p^* *an optimal solution to Problem* (2.13) *when we set* $d = \sqrt{n}r$. *Then* p^* *is the output of a LLOO with parameter* $\rho = \sqrt{n}$ *for* $\mathcal{S}_n$. *That is,*

1. $\forall y \in \mathcal{S}_n \cap \mathbb{B}_r(x) : p^* \cdot c \leq y \cdot c$.

2. $\|x - p^*\| \leq \sqrt{n}r$.

Proof. The proof follows since for any $x, y \in \mathbb{R}^n$ it holds that $\frac{1}{\sqrt{n}}\|x - y\|_1 \leq \|x - y\| \leq \|x - y\|_1$. □

Problem (2.13) with parameter $d = \sqrt{n}r$ is solved optimally by the following simple algorithm.

Algorithm 3 Local Linear Optimization Oracle for the Simplex

Input: point $x \in \mathcal{S}_n$, radius $r > 0$, linear objective $c \in \mathbb{R}^n$
$d \leftarrow \sqrt{n}r$
$\Delta \leftarrow \min\{d/2, 1\}$
$i^* \leftarrow \arg\min_{i \in [n]} c(i)$
$p_+ \leftarrow \Delta \cdot e_{i^*}$ {e_i denotes the ith standard basis vector}
$p_- \leftarrow \vec{0}$
Let $i_1, ..., i_n$ be a permutation over $[n]$ such that $c(i_1) \geq c(i_2) \geq ... \geq c(i_n)$
Let $k \in [n]$ be the smallest integer such that $\sum_{j=1}^{k} x(i_j) \geq \Delta$
$\forall j \in [k-1] : p_-(i_j) \leftarrow x(i_j)$
$p_-(i_k) \leftarrow \Delta - \sum_{j=1}^{k-1} x(i_j)$
return $p \leftarrow x + p_+ - p_-$

The algorithm basically modifies the input point x by moving the largest amount of mass which will not violate the constraint $\|x - p\|_1 \leq d$ from the entries that correspond to the largest (signed) entries in the objective c to the single entry that corresponds to the smallest (signed) entry in the objective c.

In Algorithm 3, we fix the value of d to $\sqrt{n}r$ to correspond to Lemma 3. However, as the following lemma shows, the algorithm finds an optimal solution to Problem (2.13) for any $d \geq 0$.

Lemma 4. *Fix $d \geq 0$. Algorithm 3 finds an optimal solution to Problem* (2.13) *with parameter d.*

Proof. Fix an optimal solution p^* to Problem (2.13). We can write p^* in the following way:

$$p^* = x - p_- + p_+, \tag{2.14}$$

where p_-, p_+ are non-negative. Note that without loss of generality we can assume that p_-, p_+ are orthogonal. To see this, assume that there exists an entry i such that $p_-(i) > 0$ and $p_+(i) > 0$. By replacing p_-, p_+ with $p_- - \min\{p_-(i), p_+(i)\}e_i$ and $p_+ - \min\{p_-(i), p_+(i)\}e_i$ respectively, where e_i is the ith standard basis vector in $\mathbb{R}^n$, we have that the new vectors still satisfy Eq. (2.14), both are non-negative but now at least one of them has a value of 0 in the ith entry. By repeating this process for every entry i that is non-zero in both vectors we can make them orthogonal. As a result, it must hold that $x \geq p_-$ (otherwise p^* is not be feasible). Furthermore, since p^* is feasible ($\|p^*\|_1 = 1$), it must hold that $\|p_+\|_1 = \|p_-\|_1$.

Denote $\Delta = \min\{d/2, 1\}$ and assume now that $\|p_+\|_1 < \Delta$ (i.e. the ℓ_1 constraint in Problem (2.13) is not tight for p^*), and denote $w = x - p_-$. It follows that there must exist a vector $y \geq 0$ such that $\|y\|_1 = \Delta - \|p_+\|_1$ and $w \geq y$. Now define

$$\tilde{p}_- := p_- + y, \qquad \tilde{p}_+ := p_+ + y.$$

Note that it holds that $p^* = x - \tilde{p}_- + \tilde{p}_+$, $\|\tilde{p}_+\|_1 = \|\tilde{p}_-\|_1 = \Delta$ and that $x \geq \tilde{p}_-$ (although $\tilde{p}_-, \tilde{p}_+$ are no longer orthogonal).

Thus we have that

$$\tilde{p}_+ \in \Delta \cdot \mathcal{S}_n, \qquad \tilde{p}_- \in (\Delta \cdot \mathcal{S}_n) \cap \{z \in \mathbb{R}^n \mid z \leq x\}.$$

Note also that for any $p_1 \in \Delta \cdot \mathcal{S}_n$ and $p_2 \in (\Delta \cdot \mathcal{S}_n) \cap \{z \in \mathbb{R}^n \mid z \leq x\}$ it holds that $x + p_1 - p_2$ is a feasible solution to Problem (2.13).

Now we can write

$$\begin{aligned}(p^* - x) \cdot c &= \tilde{p}_+ \cdot c - \tilde{p}_- \cdot c \\ &\geq \min_{p_1 \in \Delta \cdot \mathcal{S}_n} p_1 \cdot c - \max_{p_2 \in \Delta \cdot \mathcal{S}_n : p_2 \leq x} p_2 \cdot c. \end{aligned} \tag{2.15}$$

It is now a simple observation that the vectors p_+, p_- computed in Algorithm 3 are exactly solutions to the optimization problems

$$\min_{p_1 \in \Delta \cdot \mathcal{S}_n} p_1 \cdot c, \qquad \max_{p_2 \in \Delta \cdot \mathcal{S}_n : p_2 \leq x} p_2 \cdot c$$

respectively. Hence the lemma follows.

□

Two important observations regarding the implementation of Algorithm 3 are that i) the running time of the algorithm does not explicitly depends on the dimension n but rather on the number of non-zero entries in x and the time to compute the index i^* and ii) computing the index i^* is equivalent to finding a vertex of $\mathcal{S}_n$ that minimizes the dot product with the objective c, and hence is equivalent to a single call to the linear optimization oracle of $\mathcal{S}_n$.

2.4.2 Construction of a local linear optimization oracle for an arbitrary polytope

We now turn to generalize the above simple construction for the simplex to an arbitrary polytope $\mathcal{P}$. A natural approach is to consider the polytope $\mathcal{P}$ as convex hull of its vertices, i.e., we map a point $x \in \mathcal{P}$ to a point $\lambda_x \in \mathcal{S}_N$, where $\mathcal{V} = \{v_1, v_2, ...\}$ denotes the set of vertices of $\mathcal{P}$ and $N = |\mathcal{V}|$. Given a linear objective $c \in \mathbb{R}^n$, consider its extension to $\mathbb{R}^N$ given by the vector $c_{ext} \in \mathbb{R}^N$ such that $c_{ext}(i) = v_i \cdot c$, for all $i \in [N]$. Now we can see that

$$\min_{y \in \mathcal{P}} y \cdot c \equiv \min_{\lambda \in \mathcal{S}_N} \lambda \cdot c_{ext}. \tag{2.16}$$

Thus, following our approach for the probabilistic simplex, it is tempting to consider as the output of a LLOO for $\mathcal{P}$, the point $p = \sum_{i=1}^N \lambda_i^* v_i$, where λ^* is an optimal solution to the following optimization problem:

$$\begin{aligned} &\min_{\lambda \in \mathcal{S}_N} \lambda \cdot c_{ext} \\ \text{s.t.} \quad &\|\lambda - \lambda_x\|_1 \leq d, \end{aligned} \tag{2.17}$$

where $\lambda_x \in \mathcal{S}_N$ is a mapping of the LLOO input point - x to $\mathcal{S}_N$ and d is a positive scalar. Note that since $\lambda^* \in \mathcal{S}_N$, the solution p is always a feasible point of the polytope $\mathcal{P}$.

The main question is whether we can find a value of d such that a solution to Problem (2.17) indeed corresponds to the output of a LLOO for $\mathcal{P}$ with a reasonable parameter ρ, as in the case of the simplex.

Our implementation of a LLOO for an arbitrary polytope $\mathcal{P}$ based on solving Problem (2.17) and outputting the corresponding point in $\mathcal{P}$ is given below (Algorithm 4). The algorithm is a clear extension of Algorithm 3 for the simplex, and basically moves mass from vertices in the support of the input point x (that is, vertices with non-zero weight in the convex decomposition of x) which have large (signed) product with the linear objective c, to a single vertex (possibly not in the support of the input point x) which minimizes the dot product with c. The latter is just the result of calling the linear optimization oracle of the polytope with respect to the linear objective c.

Note that the algorithm assumes that the input point x is given in the form of a convex combination of vertices of the polytope. Later on we show that maintaining such a decomposition of the input point x is straightforward and efficient when the LLOO is used with any of the optimization algorithms considered in this chapter. Note also that in the algorithm we implicitly fix the value d in Problem (2.17) to $d = 2\frac{\sqrt{n}\psi}{\xi}r$ (recall that ψ, ξ are geometric quantities of the polytope at hand, defined formally in Section 2.1), which is justified by our analysis.

It is important to note that, as in the case of Algorithm 3 for the simplex, the running time of Algorithm 4 does not explicitly depends on the number of vertices N, but only on the number of non-zeros in the vector λ (the mapping of the input point x to $\mathcal{S}_N$), the natural dimension of $\mathcal{P}$ - n and the time to complete a single call to the linear optimization oracle of the polytope - $\mathcal{O}_{\mathcal{P}}(\cdot)$. In particular, observe that in the computations in lines 3-10 of the algorithm, one needs to consider only the vertices v_i for which

$\lambda_i > 0$.

Algorithm 4 Local Linear Optimization Oracle for Polytope $\mathcal{P}$

Input: point $x \in \mathcal{P}$ such that $x = \sum_{i=1}^{N} \lambda_i v_i$, $\lambda \in \mathcal{S}_N$, radius $r > 0$, linear objective $c \in \mathbb{R}^n$
$\Delta \leftarrow \min\{\frac{\sqrt{n}\psi}{\xi} r, 1\}$
$\forall j \in [N]$: $\ell_i \leftarrow v_i \cdot c$
Let $i_1, ...i_N$ be a permutation over $[N]$ such that $\ell_{i_1} \geq \ell_{i_2} \geq ...$
Let k be the smallest integer such that $\sum_{i=1}^{j} \lambda_i \geq \Delta$
$p_- \leftarrow \vec{0}$
for $j = 1...k-1$ **do**
  $p_- \leftarrow p_- + \lambda_{i_j} v_{i_j}$
end for
$p_- \leftarrow p_- + \left(\Delta - \sum_{j=1}^{k-1} \lambda_{i_j}\right) v_{i_k}$
$v^* \leftarrow \mathcal{O}_{\mathcal{P}}(c)$
$p_+ \leftarrow \Delta \cdot v^*$
return $p \leftarrow x + p_+ - p_-$

We turn to prove that there is indeed a choice for the parameter d in Problem (2.17) (the one used to set Δ in Algorithm 4) such that Algorithm 4 is indeed a LLOO for $\mathcal{P}$. Towards this end, the main step is to show that there exists a constant $c(\mathcal{P})$, such that given a query point $x \in \mathcal{P}$ in the form $x = \sum_{i=1}^{N} \lambda_x(i) v_i$ where $\lambda_x \in \mathcal{S}_N$, and a point $y \in \mathcal{P}$, there exists a mapping of y to $\mathcal{S}_N$, i.e., a point $\lambda_y \in \mathcal{S}_N$ satisfying $y = \sum_{i=1}^{N} \lambda_y(i) v_i$, such that

$$\|\lambda_x - \lambda_y\|_1 \leq c(\mathcal{P}) \|x - y\|. \tag{2.18}$$

This fact is a consequence of Lemmas 5, 7. Lemma 5 considers a certain way to map a point $y \in \mathcal{P}$ to $\lambda_y \in \mathcal{S}_N$ which has useful properties. Lemma 7 then builds on these properties to give a consequence in the spirit of Eq. (2.18) by considering the projection of the vector $(x - y)$ onto a certain set of constraints defining the polytope $\mathcal{P}$.

Lemma 5. *Let $x \in \mathcal{P}$ and $\lambda \in \mathcal{S}_N$ such that $x = \sum_{i=1}^{N} \lambda_i v_i$, and let $y \in \mathcal{P}$. Write $y = \sum_{i=1}^{N} (\lambda_i - \Delta_i) v_i + (\sum_{i=1}^{N} \Delta_i) z$ for values $\Delta_i \in [0, \lambda_i] \, \forall i \in [N]$ and $z \in \mathcal{P}$, such that the sum $\Delta = \sum_{i=1}^{N} \Delta_i$ is minimized. Then, for all $i \in [N]$ for which $\Delta_i > 0$, there exists an index $j_i \in [m]$ such that $A_2(j_i) \cdot v_i < b_2(j_i)$*

and $A_2(j_i) \cdot z = b_2(j_i)$.

Proof. By way of contradiction, suppose the lemma is false and let $i' \in [N]$ such that $\Delta_{i'} > 0$ and $\forall j \in [m]$ it holds that if $A_2(j) \cdot v_{i'} < b_2(j)$ then $A_2(j) \cdot z < b_2(j)$. Fixing some $j \in [m]$ we consider two cases. If $A_2(j) \cdot v_{i'} = b_2(j)$ then we have that

$$\forall \gamma \geq 0 : \qquad A_2(j) \cdot (z - \gamma v_{i'}) \leq b_2(j) - \gamma b_2(j) = (1-\gamma) b_2(j). \tag{2.19}$$

On the other hand, if $A_2(j) \cdot v_{i'} < b_2(j)$, then by the assumption we have that $A_2(j) \cdot z < b_2(j)$. Denote

$$\delta_j := b_2(j) - A_2(j) \cdot v_{i'}, \quad \epsilon_j := b_2(j) - A_2(j) \cdot z,$$

and note that $\delta_j > 0$ and $\epsilon_j > 0$.

Now it holds that

$$\begin{aligned} \forall \gamma \in [0, \frac{\epsilon_j}{\delta_j}] : \qquad A_2(j) \cdot (z - \gamma v_{i'}) &= b_2(j) - \epsilon_j - \gamma(b_2(j) - \delta_j) \\ &= (1-\gamma) b_2(j) - (\epsilon_j - \gamma \delta_j) \\ &\leq (1-\gamma) b_2(j). \end{aligned} \tag{2.20}$$

Let $\tilde{\gamma} = \min\{\frac{\epsilon_j}{\delta_j} \mid j \in [m],\ A_2(j) \cdot v_{i'} < b_2(j)\}$ (note that by definition $\tilde{\gamma} > 0$, since it is the minimum over a set of strictly positive scalars).

Combining Eq. (2.19), (2.20) for all $j \in [m]$, we have that

$$\forall \gamma \in [0, \min\{\tilde{\gamma}, 1\}] : \qquad A_2(z - \gamma v_{i'}) \leq (1-\gamma) b_2.$$

Since $v_{i'}, z$ are both feasible, it also holds that $A_1(z - \gamma v_{i'}) = (1-\gamma) b_1$ and thus we arrive at the conclusion that

$$\forall \gamma \in [0, \min\{\tilde{\gamma}, 1\}] : \qquad z - \gamma v_{i'} \in (1-\gamma)\mathcal{P}. \tag{2.21}$$

Thus in particular, by choosing $\gamma \in (0, \min\{\tilde{\gamma}, 1\}] \cap (0, \frac{\Delta_{i'}}{\Delta}]$ (recall that $\Delta_{i'} > 0$ and $\tilde{\gamma} > 0$), we have that there exists $w \in \mathcal{P}$ such that $z =$

$(1-\gamma)w + \gamma v_{i'}$, and

$$\begin{aligned} y &= \sum_{i=1}^{N}(\lambda_i - \Delta_i)v_i + \Delta z \\ &= \sum_{i=1}^{N}(\lambda_i - \Delta_i)v_i + \Delta((1-\gamma)w + \gamma v_{i'}) \\ &= \left(\sum_{i=1, i\neq i'}^{N}(\lambda_i - \Delta_i)v_i\right) + (\lambda_{i'} - (\Delta_{i'} - \gamma\Delta) - \gamma\Delta)v_{i'} + \Delta(1-\gamma)w \\ &\quad +\gamma\Delta v_{i'} \\ &= \left(\sum_{i=1, i\neq i'}^{N}(\lambda_i - \Delta_i)v_i\right) + (\lambda_{i'} - (\Delta_{i'} - \gamma\Delta)v_{i'} + \Delta(1-\gamma)w. \end{aligned}$$

Thus, by defining $\forall i \in [N], i \neq i'$: $\tilde{\Delta}_i = \Delta_i$ and $\tilde{\Delta}_{i'} = \Delta_{i'} - \gamma\Delta$, we have that $y = \sum_{i=1}^{N}(\lambda_i - \tilde{\Delta}_i)v_i + (\sum_{i=1}^{N}\tilde{\Delta}_i)w$ with $\sum_{i=1}^{N}\tilde{\Delta}_i < \sum_{i=1}^{N}\Delta_i$, which contradicts the minimality of $\sum_{i=1}^{N}\Delta_i$. □

In Lemma 7 we are going to examine the projection of a vector $(x-y)$ onto a set of constraints of $\mathcal{P}$ satisfied by a certain feasible point $z \in \mathcal{P}$. However, we would like that this set will not be too large. The following simple lemma shows that it suffices to consider a basis for the set of constraints satisfied by z.

Lemma 6. *Let $z \in \mathcal{P}$ and denote $C(z) = \{i \in [m] \mid A_2(i) \cdot z = b_2(i)\}$ and let $C_0(z) \subseteq C(z)$ be such that the set $\{A_2(i)\}_{i\in C_0(z)}$ is a basis for the set $\{A_2(i)\}_{i\in C(z)}$. Then given a point $y \in \mathcal{P}$, if there exists $i \in C(z)$ such that $A_2(i) \cdot y < b_2(i)$, then there exists $i_0 \in C_0(z)$ such that $A_2(i_0) \cdot y < b_2(i_0)$.*

Proof. Fix $z \in \mathcal{P}$ and let $C(z), C_0(z)$ be as in the lemma. Assume by way of contradiction that there exists $y \in \mathcal{P}$ and $i \in C(z)$ such that $A_2(i) \cdot y < b_2(i)$ and for any $j \in C_0(z)$ it holds that $A_2(j) \cdot y = b_2(j)$. Since $A_2(i)$ is a linear combination of vectors from $\{A_2(j)\}_{j\in C_0(z)}$, there exists scalars $\{\alpha_j\}_{j\in C_0(z)}$, not all zeros, such that $A_2(i) = \sum_{j\in C_0(z)} \alpha_j A_2(j)$. From our assumption on y it follows that

$$b_2(i) > A_2(i) \cdot y = \sum_{j\in C_0(z)} \alpha_j A_2(j) \cdot y = \sum_{j\in C_0(z)} \alpha_j b_2(j).$$

However, since for all $j \in C(z)$ it holds that $A_2(j) \cdot z = b_2(j)$, we have that

$$b_2(i) = A_2(i) \cdot z = \sum_{j \in C_0(z)} \alpha_j A_2(j) \cdot z = \sum_{j \in C_0(z)} \alpha_j b_2(j).$$

Thus we arrive at a contradiction and the lemma follows. □

Lemma 7. *Let $x \in \mathcal{P}$ and $\lambda \in \mathcal{S}_N$ such that $x = \sum_{i=1}^N \lambda_i x_i$, and let $y \in \mathcal{P}$. Write $y = \sum_{i=1}^N (\lambda_i - \Delta_i) v_i + (\sum_{i=1}^N \Delta_i) z$, where $\forall i \in [N] : \Delta_i \in [0, \lambda_i]$ and $z \in \mathcal{P}$, such that the sum $\sum_{i=1}^N \Delta_i$ is minimized (as in Lemma 5). Then it holds that*

$$\sum_{i=1}^N \Delta_i \leq \frac{\sqrt{n}\psi}{\xi} \|x - y\|.$$

As a consequence, y could be mapped to a point $\lambda_y \in \mathcal{S}_N$ such that

$$\|\lambda - \lambda_y\|_1 \leq 2 \sum_{i=1}^N \Delta_i \leq 2 \frac{\sqrt{n}\psi}{\xi} \|x - y\|.$$

Proof. Denote $C(z) = \{j \in [m] \mid A_2(j) z = b_2(j)\}$ and note that according to Lemma 5 it holds that $C(z) \neq \emptyset$. Let $C_0(z) \subseteq C(z)$ such that the set of vectors $\{A_2(i)\}_{i \in C_0(z)}$ is a basis for the set $\{A_2(i)\}_{i \in C(z)}$. Denote by $A_{2,z} \in \mathbb{R}^{|C_0(z)| \times n}$ the matrix A_2 after deleting every row $i \notin C_0(z)$ and recall that by definition $\|A_{2,z}\| \leq \psi$. Then it holds that

$$\begin{aligned}
\|x - y\|^2 &= \|\sum_{i \in [N]: \Delta_i > 0} \Delta_i (v_i - z)\|^2 \geq \frac{1}{\|A_{2,z}\|^2} \|A_{2,z} \sum_{i \in [N]: \Delta_i > 0} \Delta_i (v_i - z)\|^2 \\
&\geq \frac{1}{\psi^2} \|\sum_{i \in [N]: \Delta_i > 0} \Delta_i A_{2,z} (v_i - z)\|^2 \\
&= \frac{1}{\psi^2} \sum_{j \in C_0(z)} \left(\sum_{i \in [N]: \Delta_i > 0} \Delta_i (A_2(j) \cdot v_i - b_2(j)) \right)^2 .
\end{aligned}$$

Note that $|C_0(z)| \leq n$ and that for any vector $x \in \mathbb{R}^{|C_0(z)|}$ it holds that

$\|x\| \geq \frac{1}{\sqrt{|C_0(z)|}}\|x\|_1$. Thus we have that

$$\begin{aligned}\|x-y\|^2 &\geq \frac{1}{n\psi^2}\left(\sum_{j\in C_0(z)}\left|\sum_{i\in[N]:\Delta_i>0}\Delta_i(A_2(j)\cdot v_i - b_2(j))\right|\right)^2 \\ &= \frac{1}{n\psi^2}\left(\sum_{j\in C_0(z)}\sum_{i\in[N]:\Delta_i>0}\Delta_i(b_2(j)-A_2(j)\cdot v_i)\right)^2.\end{aligned}$$

Combining Lemma 5 and Lemma 6, we have that for all $i \in [N]$ such that $\Delta_i > 0$ there exists $j \in C_0(z)$ such that $A_2(j)\cdot v_i \leq b_2(j) - \xi$. Hence,

$$\|x-y\|^2 \geq \frac{1}{n\psi^2}\left(\sum_{i\in[N]:\Delta_i>0}\Delta_i\xi\right)^2 = \frac{\xi^2}{n\psi^2}\left(\sum_{i\in[N]:\Delta_i>0}\Delta_i\right)^2.$$

Thus we conclude that $\sum_{i=1}^N \Delta_i \leq \frac{\sqrt{n}\psi}{\xi}\|x-y\|$. □

The following lemma establishes that Algorithm 4 is indeed a local linear optimization oracle for $\mathcal{P}$ with parameter $\rho = \sqrt{n}\mu$ (μ is a geometric parameter of $\mathcal{P}$ that was formally defined in Section 2.1).

Lemma 8. *Let p be the point returned by algorithm 4 when called with the input $x = \sum_{i=1}^N \lambda_i v_i$, r, c. Then the following conditions hold:*

1. *$p \in \mathcal{P}$.*
2. *$\|x-p\| \leq \sqrt{n}\mu r$.*
3. *$\forall y \in \mathbb{B}_r(x)\cap\mathcal{P}$ it holds that $c\cdot y \geq c\cdot p$.*

Proof. Condition 1. holds since p is clearly given as a convex combination of points in $\mathcal{V}$. For conditions 2,3, note that we can write the returned point p as $p = \sum_{i=1}^N(\lambda_i - \Delta_i)v_i + \Delta v^*$, where Δ is as in Algorithm 4, for all $i \in [N] : \Delta_i \in [0, \lambda_i]$, $\sum_{i=1}^N \Delta_i = \Delta$ and $v^* \in \mathcal{V}$. Thus we have that

$$\begin{aligned}\|x-p\| &= \|\sum_{i=1}^N \Delta_i v_i - \Delta v^*\| = \|\sum_{i=1}^N \Delta_i(v_i - v^*)\| \\ &\leq \sum_{i=1}^N \Delta_i\|v_i - v^*\| \leq \Delta\mathcal{D} \leq \frac{\sqrt{n}\psi\mathcal{D}}{\xi} = \sqrt{n}\mu,\end{aligned}$$

which gives condition 2.

Finally, for condition 3, note that from Algorithm 4 and Lemma 4 it follows that the returned point p could be written as $p = \sum_{i=1}^{N} \lambda^*(i) v_i$ such that λ^* is an optimal solution to Problem (2.17) with parameter $d = 2\frac{\sqrt{n}\psi}{\xi} r$. From Lemma 7 we have that for any point $y \in \mathbb{B}_r(x) \cap \mathcal{P}$, y could be mapped to a point $\lambda_y \in \mathcal{S}_N$ such that $\|\lambda - \lambda_y\|_1 \leq 2\frac{\sqrt{n}\psi}{\xi} r$. Thus we have that

$$\begin{aligned} p \cdot c &= \sum_{i=1}^{N} \lambda^*(i) v_i \cdot c = \lambda^* \cdot c_{ext} = \min_{\lambda_z \in \mathcal{S}_N : \|\lambda - \lambda_z\|_1 \leq 2\frac{\sqrt{n}\psi}{\xi} r} \lambda_z \cdot c_{ext} \\ &\leq \lambda_y \cdot c_{ext} = \sum_{i=1}^{N} \lambda_y(i) v_i \cdot c = y \cdot c. \end{aligned}$$

□

2.4.3 Maintaining a small decomposition of the input point and efficient implementation of Algorithm 4

Algorithm 4 assumes that the input point x is given by its convex decomposition into vertices. All optimization algorithms in this chapter use Algorithm 4 in the following way: they give as input to Algrotihm 4 the current feasible iterate $x_t \in \mathcal{P}$, and then given the output of Algorithm 4, denoted in all algorithms by p_t, they produce the next iterate x_{t+1} by taking a convex combination $x_{t+1} \leftarrow (1-\alpha)x_t + \alpha p_t$, for some parameter $\alpha \in [0,1]$. Note that Algorithm 4 implicitly produces the convex decomposition of the returned point p_t and thus, given the convex decomposition of x_t, updating it to the convex decomposition of x_{t+1} is straightforward.

Moreover, denoting $\mathcal{V}_t \subseteq \mathcal{V}$ the set of vertices that forms the convex decomposition of x_t (i.e. the vertices with non-zero weight in the decomposition), it is clear from Algorithm 4 and the discussion above that $|\mathcal{V}_{t+1} \setminus \mathcal{V}_t| \leq 1$, since at most a single vertex (v^* in Algorithm 4) is added to the decomposition.

This brings us to the following lemma.

Lemma 9. *Algorithm 4 admits an implementation such that each invocation of the algorithm requires a single call to the oracle $\mathcal{O}_{\mathcal{P}}$ and additional $O(T(n+\log T))$ time, where T is the overall number of calls to the algorithm.*

Proof. Clearly Algorithm 4 calls the oracle $\mathcal{O}_{\mathcal{P}}$ only once per invocation. The complexity of all other operations depends on the number of non-zeros in the vector λ, i.e., the number of vertices in the convex decomposition of the input point x. As discussed above, if we denote by x_t, x_{t+1} the inputs to the t and $t+1$ times Algorithm 4 was invoked respectively, and by N_t, N_{t+1} the number of vertices in the convex decomposition of x_t, x_{t+1} respectively, then $N_{t+1} \leq N_t + 1$. Thus, if the algorithm is invoked for a total number of T times and the initial point - x_1 is a vertex of $\mathcal{P}$, then at any time $t \in [T]$ it holds that $N_t \leq T$. Since all other operations except for calling $\mathcal{O}_{\mathcal{P}}$ consist of computing N_t inner products between vectors in $\mathbb{R}^n$ and sorting N_t scalars, the lemma follows. □

Note that we can get rid of the linear dependence on T in the bound in lemma 9 by decomposing the iterate x_t into a convex sum of fewer vertices in case the number of vertices in the current decomposition - N_t becomes too large. From Carathéodory's theorem we know that we can find such a decomposition with at most $n+1$ vertices. Moreover, for many polytopes of interest (such as the flow polytope), there is an even more efficient algorithm for computing such a decomposition (however these are beyond the scope of this work). It follows from previous discussions that we will need to invoke such a decomposition procedure only every $O(n)$ iterations which will keep the amortized iteration complexity low.

Another generic approach to the above problem that relies only on the use of the linear optimization oracle - $\mathcal{O}_{\mathcal{P}}$ (and which might also be more efficient), is to "bootstrap" Algorithm 2 to compute a more compact decomposition of x_t. If N_t is too large we can compute a new decomposition of the input point x_t by solving the optimization problem $\min_{y \in \mathcal{P}} \|x_t - y\|^2$ up to some precision r_t^2, where r_t is the current radius parameter of the LLOO. Using Theorem 4, the result will be a point $\tilde{x}_t$ given by a decomposition into $O(n\mu^2 \log(1/r_t))$ vertices of $\mathcal{P}$ such that $\|\tilde{x}_t - x_t\| \leq r_t$. Now, by using the point $\tilde{x}_t$ to maintain the input to the LLOO and replacing the input r_t to LLOO with $\tilde{r}_t = 2r_t$ we get (via the triangle inequality) a modified LLOO with parameter $\tilde{\rho} = 2\rho + 1 = 2\sqrt{n}\mu + 1$. As discussed above, we will need to invoke this decomposition procedure only every $O(n\mu^2 \log(1/r_t))$ iterations which leads to the following lemma.

Lemma 10. *Assume that on every invocation of the LLOO algorithm, the*

input r to the LLOO is lower-bounded by some $r_0 > 0$. Then there exists an implementation for a LLOO with parameter $\rho = 2\sqrt{n}\mu + 1$, such that the amortized linear optimization oracle complexity per iteration is 2, and the additional amortized complexity per iteration is

$$O\left(n\mu^2 \log(1/r_0)\left(n + \log(n\mu^2 \log(1/r_0))\right)\right).$$

The proof follows the same lines as that of Lemma 9.

We note that in our online algorithms the lower bound r_0 in Lemma 10 will always satisfy: $\log(1/r_0) = O(\log(T))$, where T is the overall length of the game, and thus the running time per iteration will depend only logarithmically on T.

It is also worth mentioning that we can significantly accelerate Algorithm 4 by using parallel computations. Note that all dot product computations in line 3 of the algorithm (recall again that in practice we need to carry out these computations only for vertices v_i for which $\lambda_i > 0$) are independent of each other and could be computed in parallel.

2.4.4 Linear-space implementation of the local linear optimization oracle

It is clear from the description of Algorithm 4 and from the discussion in subsection 2.4.3 that our construction of a generic LLOO requires to store in memory and maintain at all times a convex decomposition of the query point for the LLOO algorithm. Indeed this requirement could be impractical in certain scenarios due to the possible large memory needed, and the time required to compute a smaller decomposition from time to time. Here we show that for a large and important family of polytopes, Algorithm 4 could be implemented using only $O(n)$ memory, not including the memory possibly required to implement the linear optimization oracle of the polytope. That is, we present an implementation that avoids the need to store an explicit convex decomposition of the query point altogether. This improvement comes however at the price of increasing the number-of-calls-per-iteration to the linear optimization oracle.

Towards this end, here we assume that the polytope $\mathcal{P}$ is naturally given in the following algebraic form[1]:

[1] It is important to note that while *any* polytope could be described according to (2.22),

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ix \geq \vec{0},\ Ax = b\}, \tag{2.22}$$

In particular, note that almost all examples given in Section 2.1.1 fall into this case.

Our linear-space LLOO implementation is given in Algorithm 5. The idea is that we don't have to compute an explicit convex-decomposition of the query point x in order to move mass from vertices with large product with the objective c to the vertex that minimizes the product with c. We can instead do this iteratively, "pilling off" one "bed vertex" at a time from x.

Algorithm 5 Linear-space Local Linear Optimization Oracle

Input: point $x \in \mathcal{P}$ such that $x = \sum_{i=1}^{N} \lambda_i v_i$, $\lambda \in \mathcal{S}_N$, radius $r > 0$, linear objective $c \in \mathbb{R}^n$
$\Delta \leftarrow \min\{\frac{\sqrt{n}\psi}{\xi} r, 1\}$
$\tilde{\Delta} \leftarrow 0$
$\tilde{x} \leftarrow x$
Let ω be a scalar such that $\omega > 2\frac{\|c\|}{\xi} \max_{u \in \mathcal{P}} \|u\|$ {think of ω as ∞}
repeat
    Let $\tilde{c} \in \mathbb{R}^n$ be such that $\forall i \in [n]$

$$\tilde{c}(i) = \begin{cases} -c(i) & \tilde{x}(i) > 0 \\ \omega & \tilde{x}(i) = 0 \end{cases}$$

    $v \leftarrow \mathcal{O}_{\mathcal{P}}(\tilde{c})$
    Let η be the largest scalar in $[0, \Delta - \tilde{\Delta}]$ such that $\tilde{x} - \eta v \geq \vec{0}$
    $\tilde{x} \leftarrow \tilde{x} - \eta v$
    $\tilde{\Delta} \leftarrow \Delta + \eta$
until $\tilde{\Delta} = \Delta$
$v^* \leftarrow \mathcal{O}_{\mathcal{P}}(c)$
return $p \leftarrow \tilde{x} + \Delta v^*$

Before proving the correctness of Algorithm 5, we first prove the following

altering the natural description of the polytope also affects the proprities of the objective function $f(x)$ which we wish to minimize over the polytope. In particular, changing the description may turn a strongly-convex function f into a non-strongly-convex function.

auxiliary lemma.

Lemma 11. *The loop in Algorithm 5 is executed at most n times, and at the beginning of the ith iteration of the loop it holds that $\tilde{x} \in (1-\sum_{j=1}^{i-1}\eta_j)\mathcal{P}$ and $x = \tilde{x} + \sum_{j=1}^{i-1}\eta_j v_j$, where $v_1, ..., v_{i-1}$ are the values of v in the previous iterations and $\eta_1, ..., \eta_{i-1} > 0$ are the corresponding values of η.*

Proof. We prove the second part of the lemma by induction and on the way derive the proof of the first part. Observe that the claim clearly holds for $i = 1$. Consider now an iteration $i \geq 1$ and assume that the claim holds. We are going to show that at the end of the current iteration of the loop the claim still holds.

First we show that $\eta_i > 0$. This follows since $\eta_i = 0$ if and only if there exists an entry $j \in [n]$ such that $\tilde{x}(j) = 0$ and $v_i(j) > 0$, which in turn, according to the definition of $\tilde{c}$, implies that

$$\begin{aligned} v_i \cdot \tilde{c} &= v_i(j)\tilde{c}(j) + \sum_{l \neq j} v_i(l)\tilde{c}(l) \geq v_i(j)\tilde{c}(j) + \sum_{l \in [n]:\, \tilde{x}(l) > 0} v_i(l)c(l) \\ &\geq \xi\omega - \|v\|\|c\| > \|c\| \max_{u \in \mathcal{P}} \|u\|, \end{aligned} \tag{2.23}$$

where the first inequality follows from the definition of $\tilde{c}$ and the fact that $\mathcal{P}$ contains only non-negative vectors, the second inequality follows from the Cauchy-Schwartz inequality and since by definition of ξ, for all $y \in \mathcal{P}$ and $l \in [n]$ it holds that if $y(l) \neq 0$ then $y(l) \geq \xi$, and the last inequality follows from the definition of ω. However, on the other hand, according to the induction hypothesis we have that $\tilde{x}$ could be decomposed to vertices of $\mathcal{P}$ (with non-negative weights that add up to $1 - \tilde{\Delta}$). It thus follows that there exist some vertex v of $\mathcal{P}$ and weight $\lambda > 0$ such that $\tilde{x} \geq \lambda v$, which in turn implies that $v \cdot \tilde{c} = v \cdot c \leq \|v\|\|c\|$, hence, when contrasted with Eq. (2.23), contradicts the optimality of v_i.

Now, let us denote $\tilde{x}' = \tilde{x} - \eta_i v_i$, i.e. $\tilde{x}'$ is the value of $\tilde{x}$ at the end of the ith iteration. By the definition of η_i it follows that $\tilde{x}' \geq 0$. Moreover, since according to the induction hypothesis it holds that $\tilde{x} \in (1 - \sum_{j=1}^{i-1}\eta_j)\mathcal{P}$, it follows that $A\tilde{x} = (1 - \sum_{j=1}^{i-1}\eta_j)b$. Since $v_i \in \mathcal{P}$, it follows that $A\tilde{x}' = (1 - \sum_{j=1}^{i}\eta_i)b$. Thus we have that $\tilde{x}' \in (1 - \sum_{j=1}^{i})\mathcal{P}$. By "rolling" the induction up to the first iteration it follows that $x = \tilde{x}' + \sum_{j=1}^{i}\eta_i v_i$, and the claim follows.

In order to bound the number of iterations performed by the loop let us consider the value of η on some iteration. if η obtains its maximal allowed value $\Delta - \tilde{\Delta}$, then the loop terminates after this iteration. Otherwise, it follows that for any $\tilde{\eta} > \eta$, there exists some $j \in [n]$ such that $\tilde{x}(j) - \tilde{\eta} v(j) < 0$. Thus, we have that the number of zeros in $\tilde{x}$ at the end of the iteration is larger by at least one than its value at the beginning of the iteration, and thus the loop may execute at most n times. □

The following lemma establishes the correctness and complexity of Algorithm 5.

Lemma 12. *Algorithm 5 correctly implements Algorithm 4, uses only $O(n)$ space, makes at most $n+1$ calls to the linear optimization oracle of $\mathcal{P}$, and uses $O(n^2)$ additional time.*

Proof. First we note that the bound on the space used by the algorithm is clear from the description of the algorithm. The bounds on the number of calls to the oracle $\mathcal{O}_{\mathcal{P}}$ and on the additional time complexity follows immidiately from the bound on the number of iterations performed by the loop, given in Lemma 11.

Note that in order to prove the correctness of the algorithm it suffices to show that there exists a convex decomposition for x, $x = \sum_{i=1}^{k} \lambda_i v_i$ that is monotonically decreasing with respect to c, i.e. for any two vertices v_i, v_j in the decomposition it holds that $i \leq j$ if and only if $c \cdot v_i \geq c \cdot v_j$, and that Algorithm 5 "pills off" the vertices from x according to the order $v_1, v_2, ..., v_k$ until a total "probability mass" of Δ has been removed.

Lemma 11 indeed shows that the loop in Algorithm 5 pills off vertices from the convex-decomposition of x until a mass of Δ has been removed. It only remains to show that these vertices and the convex-decomposition of the remaining vector $\tilde{x}$ are indeed monotonically-ordered according to c.

Let $\sum_{i=1}^{n} \eta_i v_i$ be a convex decomposition of x such that $v_1, ..., v_k$,$\eta_1, ..., \eta_k$ correspond to the values computed by the iterations of the loop and $v_{k+1}, ..., v_n$, $\eta_{k+1}, ...\eta_n$ correspond to the decomposition of the residue $\tilde{x}$ (not computed explicitly by Algorithm 5).

Suppose now that there exists some $i \in \{1, .., k\}$ and $j \in \{i+1, ..., n\}$ such that $c \cdot v_i < c \cdot v_j$ (hence a contradiction to our monotonic-order claim). Let us denote by $\tilde{c}_i$ and $\tilde{x}_i$ the values of $\tilde{c}$ and $\tilde{x}$ on iteration i (before the update of $\tilde{x}$), respectively. It follows from Lemma 11 that $\text{supp}(v_j) \subseteq \text{supp}(\tilde{x}_i)$, where

$\text{supp}(y) := \{i \in [n] \,|\, y(i) \neq 0\}$ (this follows since, as a consequence of the lemma, we can write the decomposition of $\tilde{x}_i$ using v_j with positive weight, and since $v_j \geq \vec{0}$). Thus, it follows from the definition of $\tilde{c}$ that

$$v_j \cdot \tilde{c}_i = -v_j \cdot c < -v_i \cdot c = v_i \cdot \tilde{c}_i,$$

which contradicts the optimality of v_i. □

2.5 Application to Submodular Function Minimization

In this section we consider the problem of *Submodular Function Minimization* (SFM). While SFM is solvable in polynomial time [67], it was observed that in practice, an algorithm known as the *Minimum-Norm-Point Algorithm* or *Wolfe's Algorithm*, which was originally suggested by Wolfe in 1976 [93], exhibits much better running times than all currently known polynomial time algorithms for SFM [67, 78]. On the other-hand, the complexity of Wolfe's algorithms is not well understood and it is currently not known if it runs in polynomial time [67].

In this section, we first show how Algorithm 2 and the corresponding Theorem 4 could be used to solve the SFM problem in pseudo-polynomial time (Theorem 6), and then draw a strong connection to Wolfe's algorithm.

Consider a ground set E of n elements. Without loss of generality we can think of E as $E = \{1, 2, .., n\}$.

Definition 5 (Submodular function)**.** *A function $f : 2^E \to \mathbb{R}$ is called submodular if for any two subsets $X, Y \subseteq E$ it holds that*

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

The problem of SFM is the following optimization problem:

$$\min_{X \in 2^E} f(X), \tag{2.24}$$

where f is submodular over the ground set E. Throughout this section we are going to assume without loss of generality that f is integral and that $f(\emptyset) = 0$. Furthermore, as standard, we assume that the function f is given

by a *value oracle* $\mathcal{O}_f : 2^E \to \mathbb{Z}$, that given a subset $S \subseteq E$ returns the value $f(S)$. For purposes of measuring the complexity of algorithms for solving Problem (2.24), we denote by T_f the worst case time required to evaluate f on any subset $S \subseteq E$, and we denote by F the maximal value of f over any subset of E, that is $F := \max_{S \in 2^E} |f(S)|$.

Instead of solving Problem (2.24) directly, we are going to consider convex optimization over a related polytope known as the *base polyhedron* of f.

Definition 6 (Base polyhedron)**.** *The base polyhedron associated with a submodular function $f : 2^E \to \mathbb{R}$ is given by*

$$\mathcal{P}_f := \{x \in \mathbb{R}^n \,|\, \forall S \in 2^E : \sum_{i \in S} x_i \leq f(S), \quad \sum_{i \in E} x_i = f(E)\}.$$

The following theorem, originally due to Fujishige, establishes that Problem (2.24) is equivalent to minimizing a certain *continuous* smooth and strongly convex function over the base polyhedron of f, $\mathcal{P}_f$.

Theorem 5 ([32], Sec. 7.1.(a))**.** *Let $f : 2^E \to \mathbb{R}$ be submodular. Let $x^* = \arg\min_{x \in \mathcal{P}_f} \|x\|^2$, and let $X^* = \{e \in E \,|\, x^*(e) < 0\}$. Then X^* is a minimizer of f over 2^E.*

Since the function $g(x) = \|x\|^2$ is smooth and strongly convex, we can solve SFM by applying Algorithm 2 and the corresponding Theorem 4 to finding the minimum-norm point in $\mathcal{P}_f$.

Theorem 6. *Fix $\epsilon > 0$. Algorithm 2, applied to the minimum-norm-point problem over the base polyhedron $\mathcal{P}_f$ given by an integral submodular functions, finds a point $x \in \mathcal{P}_f$ such that*

$$\|x\| \leq \min_{y \in \mathcal{P}_f} \|y\| - \epsilon$$

after making $O\left(n^4 F^2 \log(1/\epsilon)\right)$ iterations and $O\left(n^5 F^2 \log(1/\epsilon)\right)$ calls to the value oracle of f.

In order to formally derive Theorem 6 from Theorem 4, we require two missing ingredients: i) an efficient implementation of a linear optimization oracle for $\mathcal{P}_f$, and ii)bounding the parameter $\mu(\mathcal{P}_f)$.

Luckily, the first issue is easily addressed since there is a well known and simple greedy algorithm for linear optimization over the base polyhedron $\mathcal{P}_f$.

Theorem 7 ([67], Theorem 2.1)**.** *Let $w \in \mathbb{R}^n$ be a linear objective and let $e_1, ..., e_n$ be a permutation over $[n]$ such that $w(e_1) \leq ... \leq w(e_n)$. Then the point $y \in \mathbb{R}^n$ given by*

$$y(e_i) = f(\{e_1, e_2, ..., e_i\}) - f(\{e_1, e_2, ..., e_{i-1}\}) \quad \forall i \in [n],$$

is a vertex of $\mathcal{P}_f$ and a minimizer of the dot product $x \cdot w$ over $\mathcal{P}_f$. Moreover, y can be computed in $O(nT_f + n \log n)$ time.

In order to bound $\mu(\mathcal{P}_f)$ we have the following lemma.

Lemma 13. *Given a submodular function $f : 2^E \to \mathbb{R}$ that takes integral values, it holds that $\mu(\mathcal{P}_f) = O(n^{3/2}F)$.*

Proof. Note that according to Definition 6 and our assumption that f is integral, $\mathcal{P}_f$ could be described in the following algebraic form:

$$\mathcal{P}_f := \{x \in \mathbb{R}^n \mid Ax \leq b, \vec{1} \cdot x = f(E)\},$$

where $A \in \{0,1\}^{2^{|E|} \times n}$, $b \in \mathbb{Z}^{2^{|E|}}$ and $\vec{1}$ denotes the all-ones vector.

Note that A is not normalized to have unit-length rows as assumed in Section 2.1. However, we can still bound $\psi(\mathcal{P}_f), \xi(\mathcal{P}_f)$ and $\mu(\mathcal{P}_f)$ as described in Section 2.1 and nothing in Algorithm 2 or its analysis will change.

Since all entries of A are in $\{0,1\}$, given a matrix $M^{n \times n}$ whose rows are linearly-independent rows of A, it follows that $\|M\| \leq n$ (by taking all entries of M to equal 1). Thus we can upper bound $\psi(\mathcal{P}_f)$ by n. Note that this bound is tight up to a small constant since M might be the following matrix:

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots \\ 0 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix}.$$

In order to bound the quantity $\xi(\mathcal{P}_f)$, we note that since f is integral, for any row A_i of A it holds that $b(i)$ is also integral. Moreover, according to to structure of the vertices of $\mathcal{P}_f$ described in Theorem 7, it follows that if f is integral, then the vertices of $\mathcal{P}_f$ are also integral (note that for any vertex v of $\mathcal{P}_f$ there exists a vector $w \in \mathbb{R}^n$, such that v is the only vertex that minimizes the dot product with w. Hence, Theorem 7 exactly characterizes the structure of the vertices of $\mathcal{P}_f$). Since the row A_i is also integral, it follows that for any vertex v, either $v \cdot A_i = b(i)$ or $v \cdot A_i \leq b(i) - 1$. Thus, we conclude that $\xi(\mathcal{P}_f) \geq 1$.

In order to bound the diameter of $\mathcal{P}_f$ we use:

$$\begin{aligned} D(\mathcal{P}_f) &\leq 2 \max_{v \in \mathcal{V}(\mathcal{P}_f)} \|v\| \leq 2\sqrt{n} \max_{v \in \mathcal{V}(\mathcal{P}_f)} \|v\|_\infty \\ &\leq 4\sqrt{n} \max_{v \in \mathcal{V}(\mathcal{P}_f)} \max_{S \in 2^E} |f(S)| = 4\sqrt{n}F, \end{aligned}$$

where the last inequality follows from the structure of the vertices of $\mathcal{P}_f$ given in Theorem 7, and the last equality follows from the definition of F.

Thus, we conclude that $\mu(\mathcal{P}_f) = O(n^{3/2}F)$. □

2.5.1 Connection to Wolfe's algorithm

Wolfe's algorithm [93] is an algorithm for finding the point of minimal ℓ_2 norm in a given polytope $\mathcal{P}$, i.e. it solves the optimization problem:

$$\min_{x \in \mathcal{P}} \{g(x) := \|x\|^2\}. \tag{2.25}$$

The algorithm is closely related to the conditional gradient method: on each iteration t, Wolfe's algorithm finds a vertex p_t of $\mathcal{P}$ that minimizes the dot product $v \cdot x_t$, where x_t is the current *feasible* iterate of the algorithm. Indeed this step is equivalent to the invocation of the linear optimization oracle in the conditional gradient method, since $x_t \propto \nabla g(x_t)$.

Wolfe's algorithm differs from the standard conditional gradient method in the following way: instead of producing the next iterate by taking a convex combination: $x_{t+1} \leftarrow x_t + \alpha(p_t - x_t)$ for some $\alpha \in [0, 1]$, as in the CG method, it produces x_{t+1} by a applying a sophisticated secondary optimization step over the convex hull of all vertices in the convex decomposition of x_t and the new vertex p_t. Thus, both methods perform a single call to the linear

optimization oracle of the polytope on each iteration and produce the next iterate by combining the new and old vertices via some convex combination.

Until recently, no sub-exponential-time analysis was known for Wolfe's algorithm. Recently, the authors in [16] proved that Wolfe's algorithm converges with rate $O(nD^2/t)$, which has the same dependence on D, t as the CG method (Theorem 3), but is worse by a factor of n. Moreover, the secondary optimization step in Wolfe's algorithm comes with a computational price: it requires $O(n^3)$ time per iteration. The authors in [16] also proved a *robust* version of Fujishige's theorem (Theorem 5) that shows that in order to find a minimizer of a submodular function f, it suffices to approximate Problem (2.25) up to precision $\epsilon = \Theta(n^{-1})$. Thus, Wolfe's algorithm finds a minimizer of f after making at most $O(n^5F^2)$ calls to the value oracle of f, which is the same, up to a logarithmic factor, as our result in Theorem 6.

Theorem 4 shows that using only a linear optimization oracle, it is possible to solve the minimum-norm-point problem exponentially faster than the most advanced analysis of Wolfe's algorithm.

We conclude this section with the following open question which we believe is of interest: is it possible, by exploiting the rich proprieties of submodular functions and the base polyhedron, to modify the analysis of Algorithm 2, so it solves the SFM problem in fully polynomial time? Due to the practical success of Wolfe's algorithm for SFM, such a result may be of true practical interest.

2.6 Algorithms for Online and Stochastic Convex Optimization

In this section we present algorithms for the general setting of online convex optimization that are suitable when the decision set is a polytope. We present regret bounds for both general convex losses and for strongly convex losses. These regret bounds imply convergence rates for stochastic convex optimization and non-smooth convex optimization over polyhedral sets as described in subsections 2.1.3, 2.1.3. In the sequel we also present an algorithm for the bandit setting.

Our algorithm for online convex optimization in the full information setting is given below (Algorithm 6) . The algorithm is based on the ideas presented in Subsection 2.1.3, i.e., iteratively approximating the steps of

a regret-optimal algorithm known as *Regularize Follow the Leader* using the update step of our Algorithm 2, which amounts to a single call a local linear optimization oracle (which in turn, given the construction presented in Section 2.4, amounts to a single call to the linear optimization oracle of the polytope).

For ease of presentation, we use a standard assumption that the algorithm has knowledge on several parameters of the problem including the length of the game - T, an upper bound on the magnitude of the gradients of the observed loss functions - G, and a lower bound on the strong convexity of the observed functions - σ (which may also be zero) [2] .

Algorithm 6 LLOO-based Online Convex Optimization

Input: horizon T, upper bound on gradients G, strong convexity parameter σ, $\mathcal{A}(x, r, c)$ - LLOO with parameter ρ for $\mathcal{P}$

Set: $\alpha \leftarrow \begin{cases} (3\rho^2)^{-1} & \sigma = 0 \\ (5\rho^2)^{-1} & \sigma > 0 \end{cases}$

Set: $\eta \leftarrow \frac{D}{18G\rho\sqrt{T}}, \qquad T_0 \leftarrow (25\rho^2)^2$

Let x_1 be an arbitrary vertex in $\mathcal{V}$

for $t = 1...T$ **do**

  Play x_t

  Receive $f_t(x)$

  Define the function:

$$F_t(x) = \begin{cases} \eta\left(\sum_{\tau=1}^{t} \nabla f_\tau(x_\tau) \cdot x\right) + \|x - x_1\|^2 & \sigma = 0 \\ \left(\sum_{\tau=1}^{t} \nabla f_\tau(x_\tau) \cdot x + \frac{\sigma}{2}\|x - x_\tau\|^2\right) + T_0 \frac{\sigma}{2}\|x - x_1\|^2 & \sigma > 0 \end{cases}$$

  Set:

$$r_t \leftarrow \begin{cases} \frac{D}{\sqrt{T}}\left(\rho + \frac{1}{18\rho}\right) & \sigma = 0 \\ \frac{2(G+\sigma D)}{\sigma(t+T_0)}(60\rho^2 + 1) & \sigma > 0 \end{cases}$$

  $p_t \leftarrow \mathcal{A}(x_t, r_t, \nabla F_t(x_t))$

  $x_{t+1} \leftarrow x_t + \alpha(p_t - x_t)$

end for

We prove the following two main theorems.

[2] in case one of these bounds is unknown, one can use standard techniques such as the well known "doubling trick", which increases the overall regret only by a log factor.

Denote $G = \sup_{x \in \mathcal{P}, t \in [T]} \|\nabla f_t(x)\|$ and recall that we have a construction for a local linear optimization oracle with parameter $\rho = O(\sqrt{n}\mu)$ for the decision set $\mathcal{P}$.

Theorem 8. *In case Algorithm 6 is instanciated with the LLOO described in Section 2.4 (Algorithm 4), then for arbitrary convex loss fundtions, the regret of the algorithm is $O(GD\mu\sqrt{nT})$.*

Theorem 9. *In case Algorithm 6 is instanciated with the LLOO described in Section 2.4 (Algorithm 4), then for σ-strongly convex loss functions, the regret of the algorithm is $O(\sigma D^2 \rho^4 + (G + \sigma D)^2 n\mu^2/\sigma) \log T)$.*

Applying the above two theorems with the reduction of stochastic optimization to online optimization described in Subsection 2.1.3, yields the following two corollaries.

Corollary 1. *Let $F(x) = \mathbb{E}_{f \sim \mathcal{D}}[f(x)]$, where $\mathcal{D}$ is a distribution over arbitrary convex functions, and assume the availability of an oracle $\mathcal{O}_{\mathcal{D}}$ for sampling functions from the distribution $\mathcal{D}$. Then running Algorithm 6, instanciated with the LLOO described in Section 2.4, with a sequence of T loss functions sampled i.i.d. using $\mathcal{O}_{\mathcal{D}}$ and denoting $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ (the average of iterates), we have that*

$$\mathbb{E}[F(\bar{x}_T)] - \min_{x^* \in \mathcal{P}} F(x^*) = O\left(\frac{GD\sqrt{n}\mu}{\sqrt{T}}\right).$$

Corollary 2. *Let $F(x) = \mathbb{E}_{f \sim \mathcal{D}}[f(x)]$, where $\mathcal{D}$ is a distribution over σ-strongly convex functions, and assume the availability of an oracle $\mathcal{O}_{\mathcal{D}}$ for sampling functions from the distribution $\mathcal{D}$. Then running Algorithm 6, instanciated with the LLOO described in Section 2.4, with a sequence of T loss functions sampled i.i.d. using $\mathcal{O}_{\mathcal{D}}$ and denoting $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ (average of iterates), we have that*

$$\mathbb{E}[F(\bar{x}_T)] - \min_{x^* \in \mathcal{P}} F(x^*) = O\left(\frac{\sigma^2 D^2 \rho^4 + (G + \sigma D)^2 n\mu^2 \log(T)}{\sigma T}\right).$$

In the following two subsections we prove Theorems 8, 9.

2.6.1 Analysis for general convex losses

In this subsection we analyze the regret of Algorithm 6 in case the observed loss functions are all convex but not necessarily strongly convex, that is, $\sigma = 0$.

Consider the sequence of points $\{x_t^*\}_{t=1}^{T+1}$ such that for all $t \in [T+1]$, $x_t^* = \arg\min_{x \in \mathcal{P}} F_{t-1}(x)$, where for all $t \in [T]$, $F_t(x)$ is as defined in Algorithm 6 (for $\sigma = 0$) and for $t = 0$ we define $F_0(x) := \|x - x_1\|^2$. The regret analysis is comprised of two parts. Part 1 shows that on any time t, the point x_t played by Algorithm 6 is close to the corresponding point x_t^*. Thus by a Lipschitz argument, the cumulative loss of the sequence $\{x_t\}_{t=1}^T$ is close to that of $\{x_t^*\}_{t=1}^T$. Part 2 then follows the analysis of an algorithm known as *Regularized Follow the Leader* (see [46]) to claim that the sequence of points $\{x_t^*\}_{t=1}^T$ achieves low regret with respect to the sequence of observed loss functions.

Lemma 14. *Fix $\epsilon > 0$. Let*

$$\eta = \frac{\sqrt{\epsilon}}{18G\rho^2}, \qquad \alpha = \frac{1}{3\rho^2}, \qquad r_t = \sqrt{\epsilon} + \eta G \quad \forall t \in [T].$$

Then, the sequence of points $\{x_t\}_{t=1}^T$ produced by Algorithm 6 satisfies that for all $t \in [T]$, $\|x_t - x_t^*\| \leq \sqrt{\epsilon}$.

Proof. Observe that on any time $t \in \{0, 1, ..., T\}$ it holds that the function $F_t(x)$ is 2-strongly convex and 2-smooth.

We prove by induction that for all $t \in [T]$ it holds that $F_{t-1}(x_t) - F_{t-1}(x_t^*) \leq \epsilon$. By the strong-convexity of F_{t-1} (Eq. 2.1) this yields that $\|x_t - x_t^*\| \leq \sqrt{\epsilon}$.

The proof is by induction on t. For $t = 1$ it holds that $x_1 = x_1^*$ and thus the claim holds. Assume now that for time $t \geq 1$ it holds that $F_{t-1}(x_t) - F_{t-1}(x_t^*) \leq \epsilon$. By the strong-convexity of $F_{t-1}(x)$ and the induction hypothesis we have that

$$\|x_t - x_t^*\| \leq \sqrt{\epsilon}. \tag{2.26}$$

By the definition of $F_t(x)$ and the optimality of x_t^* we have that

$$\begin{aligned} F_t(x_t^*) - F_t(x_{t+1}^*) &= F_{t-1}(x_t^*) - F_{t-1}(x_{t+1}^*) + \eta \nabla f_t(x_t) \cdot (x_t^* - x_{t+1}^*) \\ &\leq \eta G \|x_{t+1}^* - x_t^*\|, \end{aligned}$$

and thus again by the strong convexity of $F_t(x)$ we have that

$$\|x_{t+1}^* - x_t^*\| \leq \eta G. \tag{2.27}$$

Combining Eq. (2.26), (2.27) we have that

$$\|x_t - x_{t+1}^*\| \leq \sqrt{\epsilon} + \eta G.$$

Using again the induction hypothesis, we have that

$$\begin{aligned} F_t(x_t) - F_t(x_{t+1}^*) &= F_{t-1}(x_t) - F_{t-1}(x_{t+1}^*) + \eta \nabla f_t(x_t) \cdot (x_t - x_{t+1}^*) \\ &\leq \epsilon + \eta G \|x_t - x_{t+1}^*\| \leq \epsilon + \eta G \sqrt{\epsilon} + \eta^2 G^2. \end{aligned} \tag{2.28}$$

Setting $r_t = \sqrt{\epsilon} + \eta G$, we can apply Lemma 1 with respect to $F_t(x), x_t, r_t$ and get,

$$F_t(x_{t+1}) - F_t(x_{t+1}^*) \leq (1-\alpha)(F_t(x_t) - F_t(x_{t+1}^*)) + \alpha^2 \rho^2 \left(\sqrt{\epsilon} + \eta G\right)^2.$$

Plugging Eq. (2.28) we have that

$$\begin{aligned} F_t(x_{t+1}) - F_t(x_{t+1}^*) &\leq (1-\alpha)\left(\epsilon + \eta G \sqrt{\epsilon} + \eta^2 G^2\right) \\ &\quad + 2\alpha^2 \rho^2 \left(\epsilon + \eta G \sqrt{\epsilon} + \eta^2 G^2\right) \\ &= \left(\epsilon + \eta G \sqrt{\epsilon} + \eta^2 G^2\right)(1 - \alpha + 2\alpha^2 \rho^2). \end{aligned}$$

Setting $\alpha = \frac{1}{3\rho^2}$ we get that

$$F_t(x_{t+1}) - F_t(x_{t+1}^*) \leq \left(\epsilon + \eta G \sqrt{\epsilon} + \eta^2 G^2\right)\left(1 - \frac{1}{9\rho^2}\right).$$

Finally, plugging $\eta = \frac{\sqrt{\epsilon}}{18 G \rho^2}$ gives

$$F_t(x_{t+1}) - F_t(x_{t+1}^*) \leq \epsilon \left(1 + \frac{1}{9\rho^2}\right)\left(1 - \frac{1}{9\rho^2}\right) < \epsilon.$$

□

We also need the following lemma, originally proved in [56], that states that playing on each time t the point in $\mathcal{P}$ that minimizes the loss up to time t (including), yields zero regret. A proof is given in Section 2.9 for completeness.

Lemma 15. *Let $\{f_t(x)\}_{t=1}^T$ be a sequence of loss functions and let $\{w_t^*\}_{t=1}^T$ be a sequence of points such that for all $t \in [T]$, $w_t^* \in \arg\min_{w\in\mathcal{P}} \sum_{\tau=1}^t f_\tau(w)$. Then it holds that*

$$\sum_{t=1}^T f_t(w_t^*) - \min_{w\in\mathcal{P}} \sum_{t=1}^T f_t(w) \leq 0.$$

We are now ready to prove Theorem 8.

Proof. Denote $x^* \in \arg\min_{x\in\mathcal{P}} \sum_{t=1}^T f_t(x)$. We define a sequence of functions $\{\tilde{f}_t(x)\}_{t=1}^T$ as follows: $\tilde{f}_1(x) = \nabla f_1(x_1)\cdot x + \frac{1}{\eta}\|x - x_1\|^2$ and $\tilde{f}_t(x) = \nabla f_t(x_t)\cdot x$ for all $t \geq 2$. Note that for all $t \in [T]$ we have that $F_t(x) = \eta \sum_{\tau=1}^t \tilde{f}_\tau(x)$. Recall that the sequence of points $\{x_t^*\}_{t=1}^{T+1}$ satisfies that $x_t^* = \arg\min_{x\in\mathcal{P}} F_{t-1}(x)$. Hence, for all $t \in [T]$, $x_{t+1}^* = \arg\min_{x\in\mathcal{P}} F_t(x)$. Thus, by Lemma 15 we have that

$$\begin{aligned}\sum_{t=1}^T \tilde{f}_t(x_{t+1}^*) - \sum_{t=1}^T \tilde{f}_t(x^*) &= \sum_{t=1}^T \nabla f(x_t)\cdot(x_{t+1}^* - x^*) \\ &\quad + \frac{1}{\eta}(\|x_2^* - x_1\|^2 - \|x^* - x_1\|^2) \leq 0.\end{aligned}$$

Rearranging and using $\|x^* - x_1\| \leq D$ we have that

$$\sum_{t=1}^T \nabla f_t(x_t)\cdot(x_{t+1}^* - x^*) \leq \frac{D^2}{\eta}. \tag{2.29}$$

Fix $t \in [T]$. Since $F_t(x)$ is 2-strongly convex, using Eq. (2.1) we have

that

$$\begin{aligned}\|x_t^* - x_{t+1}^*\|^2 &\leq F_t(x_t^*) - F_t(x_{t+1}^*) \\ &= F_{t-1}(x_t^*) - F_{t-1}(x_{t+1}^*) + \eta \nabla f_t(x_t) \cdot (x_t^* - x_{t+1}^*) \\ &\leq \eta G \|x_t^* - x_{t+1}^*\|, \end{aligned} \tag{2.30}$$

where the last inequality follows from the optimality of x_t^* with respect to $F_{t-1}(x)$ and the Cauchy-Schwartz inequality.

Combining Eq. (2.29) and Eq. (2.30) for all $t \in [T]$ via the Cauchy-Schwartz inequality we have that

$$\sum_{t=1}^{T} \nabla f_t(x_t) \cdot (x_t^* - x^*) \leq \frac{D^2}{\eta} + T\eta G^2.$$

Rearranging and using the Cauchy-Schwartz inequality again we have that

$$\sum_{t=1}^{T} \nabla f_t(x_t) \cdot (x_t - x^*) \leq \frac{D^2}{\eta} + T\eta G^2 + G \sum_{t=1}^{T} \|x_t - x_t^*\|.$$

Fix $\epsilon = \frac{(D\rho)^2}{T}$. Applying Lemma 14 with respect to our choice of ϵ and setting η accordingly (and recalling that $\rho \geq 1$), we have that

$$\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \sum_{t=1}^{T} \nabla f_t(x_t) \cdot (x_t - x^*) = O(GD\rho\sqrt{T}),$$

where the first inequality follows from convexity of each $f_t(x)$. The theorem now follows since according to our results from Section 2.4 we can assume that $\rho = \sqrt{n}\mu$.

□

2.6.2 Analysis for strongly convex losses

Here we analyze the regret of Algorithm 6 in case all loss function are at least σ-strongly convex for some $\sigma > 0$. The analysis goes along the same lines as the analysis for the non-strongly convex case, but requires a few modifications.

As in the previous subsection we define the sequence $\{x_t^*\}_{t=1}^{T+1}$ such that $x_t^* = \arg\min_{x\in\mathcal{P}} F_{t-1}(x)$, where $F_t(x)$ for $t \in \{1, ..., T\}$ is defined as in Algorithm 6 (for $\sigma > 0$) and in addition we define $F_0 := T_0 \frac{\sigma}{2}\|x - x_1\|^2$.

Lemma 16. *For any $t \in [T]$, the function $\tilde{f}_t(x) = \nabla f_t(x_t) \cdot x + \frac{\sigma}{2}\|x - x_t\|^2$ is $L = G + \sigma D$ Lipschitz over $\mathcal{P}$.*

Proof. Fix two points $y, z \in \mathcal{P}$. Since $\tilde{f}_t(x)$ is convex we have that

$$\begin{aligned}\tilde{f}_t(y) - \tilde{f}_t(z) &\leq \nabla \tilde{f}_t(y) \cdot (y - z) = (\nabla f_t(x_t) + \sigma(y - x_t)) \cdot (y - z) \\ &\leq \|\nabla f_t(x_t) + \sigma(y - x_t)\| \cdot \|y - z\| \leq (G + \sigma D)\|y - z\|,\end{aligned}$$

where the last inequality uses the triangle inequality and the upper bounds G, D for $\|\nabla f_t(x_t)\|$ and $\|y - x_t\|$ respectively. Since the above inequality is symmetric in y, z, the lemma follows. □

The following Lemma is analogues to Lemma 14 for the non-strongly convex case.

Lemma 17. *Let $L = G + \sigma D$, $\alpha = \frac{1}{5\rho^2}$ and $T_0 = (25\rho^2)^2$. Let $\{\epsilon_t\}_{t=1}^T$ be a sequence of positive reals such that $\epsilon_t = \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}$ $\forall t \in [T]$. Let*

$$r_t = \sqrt{\frac{4\epsilon_t}{\sigma(t + T_0)}} + \frac{2L}{\sigma(t + T_0)} \quad \forall t \in [T].$$

Then, for any $t \in [T]$ it holds that $\|x_t - x_t^*\| \leq \sqrt{\frac{\epsilon_t}{\sigma(t-1+T_0)}}$.

Proof. The proof is similar to that of Lemma 14. Observe that on any time $t \in \{0, 1, ..., T\}$ it holds that the function $F_t(x)$ is $\sigma(t + T_0)$-strongly convex and $\sigma(t + T_0)$-smooth.

We prove that for any time $t \in [T]$ it holds that $F_{t-1}(x_t) - F_{t-1}(x_t^*) \leq \epsilon_t$, which by the strong convexity of $F_{t-1}(x)$ (see Eq. (2.1)) implies that $\|x_t - x_t^*\| \leq \sqrt{\frac{2\epsilon_t}{\sigma(t-1+T_0)}}$.

Clearly for time $t = 1$ the claim holds since $x_1 = x_1^*$. Assume that on time $t \geq 1$ it holds that $F_{t-1}(x_t) - F_{t-1}(x_t^*) \leq \epsilon_t$. By the strong convexity of $F_{t-1}(x)$ we again have that

$$\|x_t - x_t^*\| \leq \sqrt{\frac{2\epsilon_t}{\sigma(t - 1 + T_0)}}. \tag{2.31}$$

Define the function $\tilde{f}_t(x) = \nabla f_t(x_t) \cdot x + \frac{\sigma}{2}\|x - x_t\|^2$. It holds that

$$\begin{aligned} F_t(x_t^*) - F_t(x_{t+1}^*) &= F_{t-1}(x_t^*) - F_{t-1}(x_{t+1}^*) + \tilde{f}_t(x_t^*) - \tilde{f}_t(x_{t+1}^*) \\ &\leq \tilde{f}_t(x_t^*) - \tilde{f}_t(x_{t+1}^*) \leq L\|x_t^* - x_{t+1}^*\|, \end{aligned}$$

where the first inequality follows from the optimality of x_t^* with respect to $F_{t-1}(x)$ and the second inequality follows from Lemma 16.

By the strong convexity of $F_t(x)$ we thus have that

$$\|x_t^* - x_{t+1}^*\| \leq \frac{2L}{\sigma(t + T_0)}. \tag{2.32}$$

Combining Eq. (2.31), (2.32) via the triangle inequality we have that

$$\begin{aligned} \|x_t - x_{t+1}^*\| &\leq \sqrt{\frac{2\epsilon_t}{\sigma(t - 1 + T_0)}} + \frac{2L}{\sigma(t + T_0)} \\ &\leq \sqrt{\frac{4\epsilon_t}{\sigma(t + T_0)}} + \frac{2L}{\sigma(t + T_0)}, \end{aligned} \tag{2.33}$$

where the second inequality holds since $T_0 \geq 1$.

Using the induction hypothesis we have that

$$\begin{aligned} F_t(x_t) - F_t(x_{t+1}^*) &= F_{t-1}(x_t) - F_{t-1}(x_{t+1}^*) + \tilde{f}_t(x_t) - \tilde{f}_t(x_{t+1}^*) \\ &\leq \epsilon_t + \sqrt{\frac{4L^2\epsilon_t}{\sigma(t + T_0)}} + \frac{2L^2}{\sigma(t + T_0)}, \end{aligned} \tag{2.34}$$

where the inequality follows from Eq. (2.33) and Lemma 16.

Setting r_t to equal the RHS of Eq. (2.33), and applying Lemma 1 with

respect to $F_t(x)$ we have that

$$\begin{aligned}
F_t(x_{t+1}) - F_t(x^*_{t+1}) &\leq (1-\alpha)(F_t(x_t) - F_t(x^*_{t+1})) + \frac{\sigma}{2}(t+T_0)\alpha^2\rho^2 r_t^2 \\
&\leq (1-\alpha)\left(\epsilon_t + \sqrt{\frac{4L^2\epsilon_t}{\sigma(t+T_0)}} + \frac{2L^2}{\sigma(t+T_0)}\right) \\
&+ \sigma(t+T_0)\alpha^2\rho^2\left(\frac{4\epsilon_t}{\sigma(t+T_0)} + \frac{4L^2}{\sigma^2(t+T_0)^2}\right) \\
&\leq (1-\alpha)\left(\epsilon_t + \sqrt{\frac{4L^2\epsilon_t}{\sigma(t+T_0)}} + \frac{2L^2}{\sigma(t+T_0)}\right) \\
&+ 4\alpha^2\rho^2\left(\epsilon_t + \frac{L^2}{\sigma(t+T_0)}\right) \\
&\leq \left(\epsilon_t + \sqrt{\frac{4L^2\epsilon_t}{\sigma(t+T_0)}} + \frac{2L^2}{\sigma(t+T_0)}\right)\left(1-\alpha+4\alpha^2\rho^2\right),
\end{aligned}$$

where the second inequality follows from Eq. (2.34), the value of r_t, and using $(a+b)^2 \leq 2a^2 + 2b^2$ to upper bound r_t^2. The rest of the inequalities follows from simple algebraic manipulations.

Setting $\alpha = \frac{1}{5\rho^2}$ we have that

$$F_t(x_{t+1}) - F_t(x^*_{t+1}) \leq \left(\epsilon_t + \sqrt{\frac{4L^2\epsilon_t}{\sigma(t+T_0)}} + \frac{2L^2}{\sigma(t+T_0)}\right)\left(1 - \frac{1}{25\rho^2}\right).$$

Plugging in our choice $\epsilon_t = \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}$ we have that

$$\begin{aligned}
F_t(x_{t+1}) - F_t(x^*_{t+1}) &\leq \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}\left(1 + \frac{1}{30\rho^2} + \frac{1}{1800(\rho^2)^2}\right)\left(1 - \frac{1}{25\rho^2}\right) \\
&< \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}\left(1 + \frac{1}{25\rho^2}\right)\left(1 - \frac{1}{25\rho^2}\right) \\
&= \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}\left(1 - \frac{1}{(25\rho^2)^2}\right).
\end{aligned}$$

Finally, setting $T_0 = (25\rho^2)^2$ we have that

$$\begin{aligned} F_t(x_{t+1}) - F_t(x^*_{t+1}) &\leq \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}\left(1 - \frac{1}{T_0}\right) < \frac{(60\rho^2 L)^2}{\sigma(t+T_0)}\left(1 - \frac{1}{t+1+T_0}\right) \\ &= \frac{(60\rho^2 L)^2}{\sigma(t+T_0)} \cdot \frac{t+T_0}{t+1+T_0} = \frac{(60\rho^2 L)^2}{\sigma(t+1+T_0)} = \epsilon_{t+1}. \end{aligned}$$

□

We are now ready to prove Theorem 9.

Proof. The proof follows the lines of the proof for Theorem 8. Define the sequence of functions $\{\tilde{f}_t(x)\}_{t=0}^T$ in the following way:

$$\tilde{f}_0(x) = T_0 \frac{\sigma}{2}\|x - x_1\|^2; \quad \tilde{f}_t(x) = \nabla f_t(x_t) \cdot x + \frac{\sigma}{2}\|x - x_t\|^2 \quad \forall t \in \{1, 2, ..., T\}.$$

Note that for all $t \in \{0, 1, ..., T\}$ it holds that $F_t(x) = \sum_{\tau=0}^t \tilde{f}_t(x)$, where $F_t(x)$ is as defined in Algorithm 6 for the case $\sigma > 0$, and recall that we define $F_0(x) := T_0 \frac{\sigma}{2}\|x - x_1\|^2$. Recall that we define a sequence of points $\{x^*_t\}_{t=1}^{T+1}$ such that for all $t \in [T+1]$, $x^*_t = \arg\min_{x \in \mathcal{P}} F_{t-1}(x)$. Let $x^* = \arg\min_{x \in \mathcal{P}} \sum_{t=1}^T f_t(x)$.

According to Lemma 15 it holds that

$$\begin{aligned} \sum_{t=0}^T \tilde{f}_t(x^*_{t+1}) - \tilde{f}_t(x^*) &= \sum_{t=1}^T \tilde{f}_t(x^*_{t+1}) - \tilde{f}_t(x^*) \\ &\quad + T_0 \frac{\sigma}{2}\left(\|x^*_1 - x_1\|^2 - \|x^* - x_1\|^2\right) \leq 0. \end{aligned}$$

Using the upper bound $\|x^* - x_1\| \leq D$ and plugging the value of T_0 in Algorithm 6 we have that

$$\sum_{t=1}^T \tilde{f}_t(x^*_{t+1}) - \tilde{f}_t(x^*) \leq \frac{T_0 \sigma D^2}{2} = O(\sigma D^2 \rho^4). \tag{2.35}$$

Fix $t \in [T]$. It holds that

$$\begin{aligned} F_t(x^*_t) - F_t(x^*_{t+1}) &= F_{t-1}(x^*_t) - F_{t-1}(x^*_{t+1}) + \tilde{f}_t(x^*_t) - \tilde{f}_t(x^*_{t+1}) \\ &\leq \tilde{f}_t(x^*_t) - \tilde{f}_t(x^*_{t+1}) \leq L\|x^*_t - x^*_{t+1}\|, \end{aligned}$$

where the first inequality follows from the optimality of x_t^* with respect to $F_{t-1}(x)$, and the second inequality follows from Lemma 16 and using $L = G + \sigma D$. Since $F_t(x)$ is $\sigma(t+T_0)$-strongly convex, this implies via Eq. (2.1) that

$$\|x_t^* - x_{t+1}^*\| \leq \frac{2L}{\sigma(t+T_0)}.$$

Applying Lemma 17 with the triangle inequality we have that

$$\|x_t - x_{t+1}^*\| \leq \frac{2L}{\sigma(t+T_0)} + \frac{60\rho^2 L}{\sigma\sqrt{(t+T_0)(t-1+T_0)}} = O\left(\frac{\rho^2 L}{\sigma t}\right),$$

where the equality follows since $T_0 \geq 1$ and by definition $\rho \geq 1$.

Thus, using Lemma 16 again we have that

$$\tilde{f}_t(x_t) - \tilde{f}_t(x_{t+1}^*) = O\left(\frac{\rho^2 L^2}{\sigma t}\right).$$

Plugging the above for all $t \in [T]$ into Eq. (2.35) we have that

$$\begin{aligned}\sum_{t=1}^{T} \tilde{f}_t(x_t) - \tilde{f}_t(x^*) &= O(\sigma D^2 \rho^4) + \sum_{t=1}^{T} O\left(\frac{\rho^2 L^2}{\sigma t}\right) \\ &= O\left(\sigma D^2 \rho^4 + \frac{\rho^2 L^2}{\sigma} \log T\right).\end{aligned}$$

The theorem now follows from the observation that since for all $t \in [T]$, $f_t(x)$ is σ-strongly convex it holds that

$$\begin{aligned}f_t(x_t) - f_t(x^*) &\leq \nabla f_t(x_t) \cdot (x_t - x^*) - \frac{\sigma}{2}\|x_t - x^*\|^2 \\ &= \nabla f_t(x_t) \cdot (x_t - x^*) - \frac{\sigma}{2}(\|x_t - x^*\|^2 - \|x_t - x_t\|^2) \\ &= \tilde{f}_t(x_t) - \tilde{f}_t(x^*)\end{aligned}$$

□

2.6.3 Bandit algorithm

In this section we give an online algorithm for the *partial information* setting (bandits). The derivation is basically straightforward using our algorithm for the *full information* setting (Algorithm 6) and the technique of [30].

For this section we assume that the feasible set $\mathcal{P}$ (again a polytope) is a full dimensional. We assume without loss of generality that the origin lies in the interior of $\mathcal{P}$ (note that our Algorithms and the complexity measure μ are invariant to translation) and we denote by r_0 the largest scalar such that $\mathbb{B}_{r_0}(0) \subset \mathcal{P}$.

We assume that the loss function $f_t(x)$ chosen by the adversary on time t is chosen with knowledge of the history of the game but without any knowledge of possible randomization used by the decision maker on time t to produce his prediction. We further assume without loss of generality that for each function $f_t(x)$ it holds that $f_t(0) = 0$.

Note that since we assume that the gradients of each $f_t(x)$ are bounded in magnitude by G, it holds that $f_t(x)$ is G-Lipschitz. This follows since,

$$\forall x, y \in \mathcal{P}: \quad f_t(x) - f_t(y) \leq (x - y) \cdot \nabla f_t(x) \leq G\|x - y\|. \tag{2.36}$$

Also, since $f_t(0) = 0$ and $0 \in \mathcal{P}$ we have that

$$\forall x \in \mathcal{P}: \quad f_t(x) = f_t(x) - f_t(0) \leq (x - 0) \cdot \nabla f_t(x) \leq GD. \tag{2.37}$$

The algorithm is given below. The algorithm uses our full-information algorithm - Algorithm 6 as a black box in the following way: on each round of the game, the bandit algorithm asks Algorithm 6 for his prediction for time t - x_t, and then it generates a loss function $\tilde{f}_t(x)$ based on the received bandit feedback, and sends it to Algorithm 6 as the loss function for time t. In this way, the bandit algorithm "simulates" a full-information game for Algorithm 6.

Algorithm 7 LLOO-based Bandit Optimization

Input: time horizon T, upper bound on magnitude of gradients - G
Init Algorithm 6 with the LLOO from Section 2.4 and input parameters $T, G, \sigma = 0$
Set: $\gamma \leftarrow \sqrt{\frac{nD}{r_0}} T^{-1/4}, \qquad \delta \leftarrow r_0 \gamma$
for $t = 1...T$ **do**
  Receive point x_t from Algorithm 6
  Sample a unit vector u_t uniformly at random
  Play $y_t = (1-\gamma)x_t + \delta u_t$
  Receive $f_t(y_t)$
  $g_t \leftarrow \frac{n}{\delta} f_t(y_t) u_t$
  Define the loss function $\tilde{f}_t(x) := g_t \cdot x$
  Feed $\tilde{f}_t(x)$ to Algorithm 6
end for

The regret analysis of Algorithm 7 closely follows the analysis in [30], but instead of using Zinkevich's algorithm [94] for the reduction from bandit feedback to full feedback, we use our Algorithm 6.

For a proof of the following Lemma see Lemma 2.1 in [30].

Lemma 18. *Fix $t \in [T]$ and define the function $\hat{f}_t(x) = \mathbb{E}_v[f_t(x + \delta v)]$, where v is a vector sampled uniformly at random from the unit ball. Let $z_t = (1-\gamma)x_t$. Then, $\mathbb{E}_{u_t}[g_t] = \nabla \hat{f}_t(z_t)$.*

In order to derive the regret bound for Algorithm 7 we need the following technical lemma.

Lemma 19. *For all $t \in [T]$, let $\hat{f}_t(x)$ be as in Lemma 18. It holds that*

$$\mathbb{E}\left[\max_{x \in \mathcal{P}} \sum_{t=1}^{T} (z_t - x) \cdot (\nabla \hat{f}_t(z_t) - g_t)\right] \leq \frac{\sqrt{T} n G D^2}{\delta}.$$

Proof. It holds that

$$\begin{aligned}\mathbb{E}\left[\max_{x \in \mathcal{P}} \sum_{t=1}^{T} (z_t - x) \cdot (\nabla \hat{f}_t(z_t) - g_t)\right] &= \mathbb{E}\left[\sum_{t=1}^{T} z_t \cdot (\nabla \hat{f}_t(z_t) - g_t)\right] \\ &+ \mathbb{E}\left[\max_{x \in \mathcal{P}} \sum_{t=1}^{T} x \cdot (g_t - \nabla \hat{f}_t(z_t))\right].\end{aligned}$$

Note that by Lemma 18 we have that for all $t \in [T]$, $\mathbb{E}[z_t \cdot (\nabla \hat{f}_t(z_t) - g_t) \,|\, z_t] = 0$, and thus,

$$\mathbb{E}\left[\max_{x\in\mathcal{P}} \sum_{t=1}^{T} (z_t - x) \cdot (\nabla \hat{f}_t(z_t) - g_t)\right] = \mathbb{E}\left[\max_{x\in\mathcal{P}} \sum_{t=1}^{T} x \cdot (g_t - \nabla \hat{f}_t(z_t))\right].$$

Using the Cauchy-Schwartz inequality and the bound $\max_{x\in\mathcal{P}} \|x\| \leq D$ we have that

$$\begin{aligned}
\mathbb{E}\left[\max_{x\in\mathcal{P}} x \cdot \left(\sum_{t=1}^{T} g_t - \nabla \hat{f}_t(z_t)\right)\right] &\leq \mathbb{E}\left[D \cdot \|\sum_{t=1}^{T} \nabla \hat{f}_t(z_t) - g_t\|\right] \\
&= D \cdot \mathbb{E}\left[\|\sum_{t=1}^{T} \nabla \hat{f}_t(z_t) - g_t\|\right] \quad (2.38)
\end{aligned}$$

It now holds that

$$\begin{aligned}
&\mathbb{E}\left[\|\sum_{t=1}^{T} \nabla \hat{f}_t(z_t) - g_t\|\right]^2 \leq \mathbb{E}\left[\|\sum_{t=1}^{T} \nabla \hat{f}_t(z_t) - g_t\|^2\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T} \|\nabla \hat{f}_t(z_t) - g_t\|^2 + 2 \sum_{1\leq i<j\leq T} (\nabla \hat{f}_i(z_i) - g_i) \cdot (\nabla \hat{f}_j(z_j) - g_j)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\|\nabla \hat{f}_t(z_t) - g_t\|^2\right] + 2 \sum_{1\leq i<j\leq T} \mathbb{E}\left[(\nabla \hat{f}_i(z_i) - g_i) \cdot (\nabla \hat{f}_j(z_j) - g_j)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\|g_t - \mathbb{E}_{u_t}[g_t]\|^2\right] + 2 \sum_{1\leq i<j\leq T} \mathbb{E}\left[(\nabla \hat{f}_i(z_i) - g_i) \cdot (\nabla \hat{f}_j(z_j) - g_j)\right] \\
&\leq \sum_{t=1}^{T} \mathbb{E}\left[\|g_t\|^2\right] + 2 \sum_{1\leq i<j\leq T} \mathbb{E}\left[(\nabla \hat{f}_i(z_i) - g_i) \cdot (\nabla \hat{f}_j(z_j) - g_j)\right], \quad (2.39)
\end{aligned}$$

where the last equality follows from Lemma 18 and the last inequality follows since for any random vector w it holds that $\mathbb{E}[\|w - \mathbb{E}[w]\|^2] \leq \mathbb{E}[\|w\|^2]$.

For all $j > i$ it holds that

$$\begin{aligned}
&\mathbb{E}\left[(\nabla \hat{f}_i(z_i) - g_i) \cdot (\nabla \hat{f}_j(z_j) - g_j)\right] = \quad (2.40) \\
&\mathbb{E}_{\{u_t\}_{t=1}^{i}}\left[(\nabla \hat{f}_i(z_i) - g_i) \cdot \mathbb{E}_{\{u_\tau\}_{\tau=i+1}^{j}}\left[(\nabla \hat{f}_j(z_j) - g_j) \,|\, \{u_t\}_{t=1}^{i}\right]\right] = 0,
\end{aligned}$$

where the last equality follows since according to Lemma 18, the inner expectation is zero.

Plugging Eq. (2.40) into Eq. (2.39) for all $i < j$ we have that

$$\mathbb{E}\left[\|\sum_{t=1}^{T}\nabla\hat{f}_t(z_t)-g_t\|\right]^2 \leq \sum_{t=1}^{T}\|g_t\|^2 \leq T\left(\frac{nGD}{\delta}\right)^2, \tag{2.41}$$

where we have used Eq. (2.37) in the last inequality.

Plugging Eq. (2.41) into Eq. (2.38) we finally have that

$$\mathbb{E}\left[\max_{x\in\mathcal{P}} x\cdot\left(\sum_{t=1}^{T}\nabla\hat{f}_t(z_t)-g_t\right)\right] \leq \frac{\sqrt{T}nGD^2}{\delta}.$$

□

Theorem 10. *For $T \geq \left(\frac{Dn}{r_0}\right)^2$ it holds that the sequence of points $\{y_t\}_{t=1}^T$ produced by Algorithm 7 is feasible and satisfies*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(y_t) - \min_{x^*\in\mathcal{P}}\sum_{t=1}^{T} f_t(x^*)\right] = O\left(GD\sqrt{\frac{nD}{r_0}}T^{3/4} + GD\mu\sqrt{nT}\right),$$

where the expectation is with respect to the randomness in choosing the vectors $\{u_t\}_{t=1}^T$.

Proof. First, we prove the feasibility of the sequence of points $\{y_t\}_{t=1}^T$. Fix $t \in [T]$. By Algorithm 7 we have that $y_t = (1-\gamma)x_t + \delta u_t$, where $x_t \in \mathcal{P}$ and u_t is a unit vector. Since by our assumption it holds that $\mathbb{B}_{r_0}(0) \subset \mathcal{P}$, we have that $r_0 u_t \in \mathcal{P}$. Thus, for any $\gamma \in [0,1]$ and $\delta \in [0, \gamma r_0]$, it follows that $y_t \in \mathcal{P}$. Clearly, the values of δ, γ in Algorithm 7 satisfy these requirements.

We move to prove the regret bound.

By applying Theorem 8 with respect to the loss functions $\{\tilde{f}_t(x) = g_t \cdot x\}_{t=1}^T$ we have that

$$\sum_{t=1}^{T} x_t\cdot g_t - \min_{x^*\in\mathcal{P}}\sum_{t=1}^{T} x^*\cdot g_t = O\left(GD\rho\sqrt{T}\right),$$

where we recall that $\rho = \sqrt{n}\mu$.

For $t \in [T]$, denote $z_t = (1-\gamma)x_t$. Since $(1-\gamma) \in [0,1]$, it holds that

$$\begin{aligned}\sum_{t=1}^{T} z_t \cdot g_t - (1-\gamma) \min_{x^* \in \mathcal{P}} \sum_{t=1}^{T} x^* \cdot g_t &= \sum_{t=1}^{T} z_t \cdot g_t - \min_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} x^* \cdot g_t \\ &= O\left(GD\rho\sqrt{T}\right). \end{aligned} \tag{2.42}$$

Ranging we have that

$$\max_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} (z_t - x^*) \cdot g_t = O\left(GD\rho\sqrt{T}\right).$$

Plugging in Lemma 19 and taking expectation we have that

$$\mathbb{E}\left[\max_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} (z_t - x^*) \cdot \nabla \hat{f}_t(z_t)\right] = O\left(\frac{\sqrt{T}nGD^2}{\delta} + GD\rho\sqrt{T}\right).$$

Note that since $f_t(x)$ is a convex function, so is $\hat{f}_t(x)$ and thus we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} \hat{f}_t(z_t) - \min_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} \hat{f}_t(x^*)\right] = O\left(\frac{\sqrt{T}nGD^2}{\delta} + GD\rho\sqrt{T}\right). \tag{2.43}$$

Fix $t \in [T]$. Note that since $f_t(x)$ is G-Lipschitz (see Eq. (2.36)), for all $x \in \mathcal{P}$ it holds that

$$\begin{aligned}|\hat{f}_t(x) - f_t(x)| &= |\mathbb{E}_{v \in \mathbb{B}}[f_t(x+\delta v) - f_t(x)]| \\ &\leq \mathbb{E}_{v \in \mathbb{B}}[|f_t(x+\delta v) - f_t(x)|] \leq G\delta. \end{aligned} \tag{2.44}$$

Plugging Eq. (2.44) for all $t \in [T]$ into Eq. (2.43) we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(z_t) - \min_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} f_t(x^*)\right] = O\left(TG\delta + \frac{\sqrt{T}nGD^2}{\delta} + GD\rho\sqrt{T}\right).$$

Since for all $t \in [T]$, $\|y_t - z_t\| \leq \delta$, using the Lipschits property of $f_t(x)$

we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(y_t) - \min_{x^* \in (1-\gamma)\mathcal{P}} \sum_{t=1}^{T} f_t(x^*)\right] = O\left(TG\delta + \frac{\sqrt{T}nGD^2}{\delta} + GD\rho\sqrt{T}\right).$$

Using again Eq. (2.36), (2.37) and the fact that $0 \in \mathcal{P}$, $f_t(0) = 0$, we have that for all $x \in \mathcal{P}$ and $t \in [T]$, $f_t((1-\gamma)x) \leq (1-\gamma)f_t(x) \leq f_t(x) + \gamma GD$. Thus we have that

$$\begin{aligned}\mathbb{E}[\text{regret}_T] &:= \mathbb{E}\left[\sum_{t=1}^{T} f_t(y_t) - \min_{x^* \in \mathcal{P}} \sum_{t=1}^{T} f_t(x^*)\right] \\ &= O\left(T\gamma GD + TG\delta + \frac{\sqrt{T}nGD^2}{\delta} + GD\rho\sqrt{T}\right).\end{aligned}$$

Now, setting $\delta = \gamma r_0$ as in Algorithm 7, and recalling that $D \geq r_0$, we have that

$$\mathbb{E}[\text{regret}_T] = O\left(T\gamma GD + \frac{\sqrt{T}nGD^2}{\gamma r_0} + GD\rho\sqrt{T}\right).$$

Finally, setting $\gamma = \sqrt{\frac{Dn}{r_0}}T^{-1/4}$ we have that

$$\mathbb{E}[\text{regret}_T] = O\left(GD\sqrt{\frac{nD}{r_0}}T^{3/4} + GD\rho\sqrt{T}\right).$$

□

2.7 Lower Bound

In this section we revisit our main result from Section 2.3, that is, our linearly converging algorithm for smooth and strongly convex optimization over polytopes. We show that in certain settings our convergence rate (i.e., number of calls to the linear optimization oracle to reach a certain approximation error) is in fact nearly tight and cannot be improved beyond constants and logarithmic terms for conditional gradient-like algorithms, i.e., algorithms that can request a vertex of the polytope that minimizes the dot product

with a certain linear objective and take linear combinations of these vertices. Similar arguments appear in [52, 63].

Towards this end, consider the following optimization problem:

$$\min_{x \in \mathcal{S}_n} \{ f(x) := \frac{1}{2} \| x - \frac{1}{n} \vec{1} \|^2 \}, \tag{2.45}$$

where $\mathcal{S}_n$ is the probabilistic simplex in $\mathbb{R}^n$ and $\vec{1}$ is the all-ones vector in $\mathbb{R}^n$. Note that the feasible point $x^* = \frac{1}{n}\vec{1}$ is the optimal solution to Problem (2.45) with value $f(x^*) = 0$. Note also that $f(x)$ is 1-smooth and 1-strongly convex.

Given a conditional gradient-like algorithm, as described above, and assuming without losing generality that the initial iterate of the algorithm x_1 is a vertex of $\mathcal{S}_n$, let x_t denote the last feasible iterate that the algorithm produced by making no more than $t-1$ queries to the linear optimization oracle of $\mathcal{S}_n$. Since, as we assume, each call to the linear optimization oracle returns a vertex, i.e., a member of the standard basis in $\mathbb{R}^n$, it follows that $\|x_t\|_0 \leq t$ (here $\|\cdot\|_0$ measures the number of non-zeros). It follows from a simple calculation that

$$f(x_t) - f(x^*) = \frac{1}{2} \| x_t - \frac{1}{n} \vec{1} \|^2 \geq \frac{1}{2}(n-t) \cdot \frac{1}{n^2} \geq \frac{1}{4n} \quad \forall t \leq \frac{n}{2}.$$

It thus follows that in order for a conditional gradient-like method to solve Problem (2.45) up to an error of at most $\frac{1}{4n}$, it requires $\Omega(n)$ calls to the linear optimization oracle of $\mathcal{S}_n$.

We now show that for Problem (2.45) our algorithm, Algorithm 2, nearly matches the above lower bound. To see this, note that we can write $\mathcal{S}_n$ in the following way: $\mathcal{S}_n = \{x \in \mathbb{R}^n \mid -Ix \leq \vec{0},\, x \cdot \vec{1} = 1\}$. It thus follows that for the simplex we have that $\psi = 1, \xi = 1, D = \sqrt{2}$ which implies that $\mu = O(1)$. Plugging this into Theorem 4 we have that Algorithm 2 requires $O(n \log(1/\epsilon))$ calls to the linear optimization oracle in order to produce an ϵ-approximated solution to Problem (2.45) for any $\epsilon > 0$, which matches the above lower bound up to a logarithmic factor.

2.8 Open Questions

We conclude this chapter with some open questions. The lower bound presented in Section 2.7 shows that in general, the explicit dependence of our algorithms on the dimension could not be further improved. However, the situation is not clear for the geometric parameter μ that appears in all of our bounds. The main question is whether the dependence on the geometrical proprieties of the polytope could be improved, perhaps by considering some other quantities? or, on the other side, is it possible to show a lower bound, at list for the offline smooth and strongly convex case, that this dependence on μ is in fact tight?

Another interesting question is whether for specific polytopes of interest, we can exploit the specific structure, and give a tighter analysis for the local linear oracle construction presented in Section 2.4, or give a different, more efficient construction of such an oracle?

Finally, it is interesting whether results of the same flavour as presented in this chapter, i.e. convergence and regret bounds, could be derived for a conditional gradient-like method for other families of convex sets.

2.9 Proof of Lemma 15

For clarity, we first restate the lemma and then prove it.

Lemma 20. *Let $\{f_t(x)\}_{t=1}^T$ be a sequence of loss functions and let $\{x_t^*\}_{t=1}^T$ be a sequence of points such that for all $t \in [T]$, $x_t^* \in \arg\min_{x\in\mathcal{P}} \sum_{\tau=1}^t f_\tau(x)$. Then it holds that*

$$\sum_{t=1}^T f_t(x_t^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^T f_t(x) \leq 0.$$

Proof. We prove by induction that for any $\tau \in [T]$ it holds that

$$\sum_{t=1}^\tau f_t(x_t^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^\tau f_t(x) \leq 0.$$

For the base case $\tau = 1$ the claim clearly holds since $x_1^* \in \arg\min_{x\in\mathcal{P}} f_1(x)$. Assume now that the claim holds for some $\tau \geq 1$. On time $\tau + 1$ it holds

that

$$\begin{aligned}
\sum_{t=1}^{\tau+1} f_t(x_t^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^{\tau+1} f_t(x) &\leq \min_{y\in\mathcal{P}} \sum_{t=1}^{\tau} f_t(y) + f_{\tau+1}(x_{\tau+1}^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^{\tau+1} f_t(x) \\
&\leq \sum_{t=1}^{\tau} f_t(x_{\tau+1}^*) + f_{\tau+1}(x_{\tau+1}^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^{\tau+1} f_t(x) \\
&= \sum_{t=1}^{\tau+1} f_t(x_{\tau+1}^*) - \min_{x\in\mathcal{P}} \sum_{t=1}^{\tau+1} f_t(x) = 0,
\end{aligned}$$

where the first inequality follows from the induction hypothesis and the third one from the optimality of $x_{\tau+1}^*$. $\square$

Chapter 3

Faster Rates for the Conditional Gradient Method over Strongly-convex Sets

The conditional gradient method (CG), also known as the Frank-Wolfe algorithm, originally introduced by Frank and Wolfe in the 1950's [31], is a *first order* method for the minimization of a smooth convex function over a convex set. Its main advantage in large-scale problems is that it is a first-order and projection-free method - i.e. the algorithm proceeds by iteratively solving a linear optimization problem and remaining inside the feasible domain. For matrix completion problems, metric learning, sparse PCA, structural SVM and other large-scale machine learning problems, this feature enabled the derivation of algorithms that are practical on one hand and come with provable convergence rates on the other [53, 62, 26, 42, 49, 81, 64].

Despite its empirical success, the main drawback of the method is its relatively slow convergence rate in comparison to optimal first order methods. The convergence rate of the method is on the order of $1/t$ where t is the number of iterations, and this is known to be tight. In contrast, Nesterov's accelerated gradient descent method gives a rate of $1/t^2$ for general convex smooth problems and a rate $e^{-\Theta(t)}$ is known for smooth and strongly convex problems. The following question arises: are there projection-free methods

with convergence rates matching that of projected gradient-descent and its extensions?

Motivated by this question, in this chapter we advance the line of research for faster convergence rates of projection free methods. We prove that in case both the objective function and the feasible set are strongly convex (in fact a slightly weaker assumption than strong convexity of the objective is required), the vanilla conditional gradient method converges at an accelerated rate of $1/t^2$. The improved convergence rate is independent of the dimension. This is also the first convergence result for the CG method that we are aware of that achieves a rate that is between the standard $1/t$ rate and a linear rate. We further show how the analysis used to prove the latter result enables to easily derive previous fast convergence rates for the CG method.

We motivate the study of optimization over strongly convex sets by demonstrating that various norms that serve as popular regularizes in machine learning problems, including ℓ_p norms, matrix Schatten norms and matrix group norms, give rise to strongly convex sets. We further show that indeed linear optimization over these sets is straightforward to implement and admits a closed-form solution. Hence the CG method is appealing for solving optimization problems with such constraints, such as regularized linear regression.

Related work

The conditional gradient method dates back to the original work of Frank and Wolfe [31] which presented an algorithm for minimizing a quadratic function over a polytope using only linear optimization steps over the feasible set. Recent results by Clarkson [18], Hazan [45] and Jaggi [52] extend the method to smooth convex optimization over the *simplex*, *spectrahedron* and arbitrary convex and compact sets respectively.

It was shown in numerous works that the convergence rate of the method is on the order of $1/t$ and that it could not be improved in general, even if the objective function is strongly convex for instance, as shown in [18, 45, 52], even though it is known that in this case, the projected gradient method achieves an exponentially fast convergence rate.

Over the past years, several results tried to improve the convergence rate

of the conditional gradient method under various assumptions. GuéLat and Marcotte [41] showed that in case the objective function is strongly convex and the feasible set is a polytope, then in case the optimal solution is located in the interior of the set, the CG method converges exponentially fast. A similar result was presented in the work of Beck and Teboulle [10] who considered a specific problem they refer to a *the convex feasibility problem* over an arbitrary convex set. They also obtained a linear convergence rate under the assumption that an optimal solution that is far enough from the boundary of the set exists.

Recently, Garber and Hazan [35] gave the first natural linearly-converging CG variant without any restricting assumptions on the location of the optimum. They showed that a variant of the Frank Wolfe method with the addition of *away steps* converges exponentially fast in case the objective function is strongly convex and the feasible set is a polytope. In follow-up work, Jaggi and Lacoste-Julien [61] gave a refined analysis of an algorithm presented in [41] which also uses away steps and showed that it also converges exponentially fast in the same setting as the Garber-Hazan result. Also relevant in this context is the work of Ahipasaoglu, Sun and Todd [2] who showed that in the specific case of minimizing a smooth and strongly convex function over the unit simplex, a variant of the conditional gradient method that also uses away steps converges with a linear rate.

In a different line of work, Migdalas and recently Lan [68, 63] considered the conditional gradient algorithm with a stronger optimization oracle that is able to solve quadratic problems over the feasible domain. They show that in case the objective function is strongly convex then exponentially fast convergence is attainable. However, in most settings of interest, an implementation of such a non-linear oracle is computationally much more expensive than the linear oracle, and the key benefit of the conditional gradient method is lost.

In the specific case that the feasible set is strongly convex, an assumption also made in this chapter, Levitin and Polyak showed in their classical work [65] that under the restrictive assumption that the norm of the gradient of the objective function is lower bounded by a constant everywhere in the feasible set, the CG method converges with an exponential rate. The same result appeared in following works by Demyanov and Rubinov [23] and Dunn [28], both also requiring that the magnitude of the gradients is lower

Ref.	Set $\mathcal{K}$	Function f	Location of x^*	Conv. rate
[52]	convex	convex	unrestricted	t^{-1}
[41]	polytope	strongly convex	interior	$e^{-\Theta(t)}$
[10]	convex	$f(x) = \|Ax - b\|_2^2$	interior	$e^{-\Theta(t)}$
[65] [23] [28]	s.convex	$\|\nabla f(x)\| \geq c > 0$	unrestricted	$e^{-\Theta(t)}$
Thm. 12	s.convex	strongly convex	unrestricted	t^{-2}

Table 3.1: Comparison of convergence rates for the conditional gradient method under different assumptions. We denote the optimal solution by x^*. Note that since all results assume smoothness of the function we omit it from column 3.

bounded by a constant everywhere in the feasible set. As we later show, the lower bound requirement on the gradients is in a sense much stronger than requiring that the objective function is strongly convex. Under our assumption however, which is slightly weaker than strong convexity of the objective, the gradient may become arbitrarily small on the feasible set.

We summarize previous convergence rate results for the standard CG method in Table 3.

3.1 Preliminaries

3.1.1 Smoothness and strong convexity

For the following definitions let $\mathbf{E}$ be a finite vector space and $\|\cdot\|$, $\|\cdot\|_*$ be a pair of dual norms over $\mathbf{E}$.

Definition 7 (smooth function)**.** *We say that a function $f : \boldsymbol{E} \to \mathbb{R}$ is β smooth over a convex set $\mathcal{K} \subset \boldsymbol{E}$ with respect to $\|\cdot\|$ if for all $x, y \in \mathcal{K}$ it holds that*

$$f(y) \leq f(x) + \nabla f(x) \cdot (y - x) + \frac{\beta}{2}\|x - y\|^2.$$

Definition 8 (strongly convex function)**.** *We say that a function $f : \boldsymbol{E} \to \mathbb{R}$ is α-strongly convex over a convex set $\mathcal{K} \subset \boldsymbol{E}$ with respect to $\|\cdot\|$ if it satisfies the following two equivalent conditions*

1. $\forall x, y \in \mathcal{K}$:

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x) + \frac{\alpha}{2}\|x - y\|^2.$$

2. $\forall x, y \in \mathcal{K}, \gamma \in [0, 1]$:

$$f(\gamma x + (1-\gamma)y) \quad \leq \quad \gamma f(x) + (1-\gamma)f(y) - \frac{\alpha}{2}\gamma(1-\gamma)\|x - y\|^2.$$

The above definition (part 1) combined with first order optimality conditions imply that for a α-strongly convex function f, if $x^* = \arg\min_{x \in \mathcal{K}} f(x)$, then for any $x \in \mathcal{K}$

$$f(x) - f(x^*) \geq \frac{\alpha}{2}\|x - x^*\|^2. \tag{3.1}$$

Eq. (3.1) further implies that the magnitude of the gradient of f at point x, $\|\nabla f(x)\|_*$ is at least of the order of the square-root of the approximation error at x, $f(x) - f(x^*)$. This follows since

$$\begin{aligned}\sqrt{\frac{2}{\alpha}\left(f(x) - f(x^*)\right)} \cdot \|\nabla f(x)\|_* &\geq \|x - x^*\| \cdot \|\nabla f(x)\|_* \\ &\geq (x - x^*) \cdot \nabla f(x) \\ &\geq f(x) - f(x^*),\end{aligned}$$

where the first inequality follows from (3.1), the second from Holder's inequality and the third from convexity of f. Thus we have that at any point $x \in \mathcal{K}$ it holds that

$$\|\nabla f(x)\|_* \geq \sqrt{\frac{\alpha}{2}} \cdot \sqrt{f(x) - f(x^*)}. \tag{3.2}$$

We will show that this property, that is in fact weaker than strong convexity, combined with an additional property of the convex set that we define next, allows to obtain the faster rates [1].

[1] In this work we assume that the convex set $\mathcal{K}$ is full-dimensional. In case this assumption does not hold, e.g. if the convex set is the unit simplex, then Eq. (3.2) holds even if we replace $\nabla f(x)$ with $P_{S(\mathcal{K})}[\nabla f(x)]$ where $P_{S(\mathcal{K})}$ denotes the projection operator onto the smallest subspace that contains $\mathcal{K}$.

Definition 9 (strongly convex set)**.** *We say that a convex set $\mathcal{K} \subset \boldsymbol{E}$ is α-strongly convex with respect to $\|\cdot\|$ if for any $x, y \in \mathcal{K}$, any $\gamma \in [0,1]$ and any vector $z \in \boldsymbol{E}$ such that $\|z\| = 1$, it holds that*

$$\gamma x + (1-\gamma)y + \gamma(1-\gamma)\frac{\alpha}{2}\|x-y\|^2 z \in \mathcal{K}.$$

That is, $\mathcal{K}$ contains a ball of of radius $\gamma(1-\gamma)\frac{\alpha}{2}\|x-y\|^2$ induced by the norm $\|\cdot\|$ centered at $\gamma x + (1-\gamma)y$.

3.1.2 The conditional gradient algorithm

The conditional gradient algorithm, also known as the Frank-Wolfe algorithm, is an algorithm for the minimization of a convex function $f : \mathbf{E} \to \mathbb{R}$ which is assumed to be β_f-smooth with respect to a norm $\|\cdot\|$, over a convex and compact set $\mathcal{K} \subset \mathbf{E}$. The algorithm implicitly assumes that the convex set $\mathcal{K}$ is given in terms of a linear optimization oracle $\mathcal{O}_{\mathcal{K}} : \mathbf{E} \to \mathcal{K}$ which given a linear objective $c \in \mathbf{E}$ returns a point $x = \mathcal{O}_{\mathcal{K}}(c) \in \mathcal{K}$ such that $x \in \arg\min_{y \in \mathcal{K}} y \cdot c$. The algorithm is given below. The algorithm proceeds in iterations, taking on each iteration t the new iterate x_{t+1} to be a convex combination between the previous feasible iterate x_t and a feasible point that minimizes the dot product with the gradient direction at x_t, which is generated by invoking the oracle $\mathcal{O}_{\mathcal{K}}$ with the input vector $\nabla f(x_t)$. There are various ways to set the parameter that controls the convex combination η_t in order to guarantee convergence of the method. The option that we choose here is the optimization of η_t via a simple line search rule, which is straightforward and computationally cheap to implement.

Algorithm 8 Conditional Gradient Algorithm with Line Search

```
Let x_0 be an arbitrary point in K
for t = 0, 1, ... do
    p_t ← O_K(∇f(x_t))
    η_t ← arg min_{η∈[0,1]} η(p_t − x_t) · ∇f(x_t) + η^2 (β_f/2) ||p_t − x_t||^2
    x_{t+1} ← x_t + η_t(p_t − x_t)
end for
```

The following theorem states the well-known convergence rate of the conditional gradient algorithm for smooth convex minimization over a compact and convex set, without any further assumptions. A proof is given in

Section 3.5 for completeness though similar proofs could also be found in [65, 52].

Theorem 11. *Let $x^* \in \arg\min_{x\in\mathcal{K}} f(x)$ and denote $D_{\mathcal{K}} = \max_{x,y\in\mathcal{K}} \|x-y\|$ (the diameter of the set with respect to $\|\cdot\|$). For every $t \geq 1$ the iterate x_t of Algorithm 8 satisfies*

$$f(x_t) - f(x^*) \leq \frac{8\beta_f D_{\mathcal{K}}^2}{t} = O\left(\frac{1}{t}\right).$$

3.1.3 Results in this chapter

In this work, we consider the case in which the function to optimize f is not only β_f-smooth with respect to $\|\cdot\|$ but also α_f-strongly convex with respect to $\|\cdot\|$ (we relax this assumption a bit in subsection 3.3.3). We further assume that the feasible set $\mathcal{K}$ is $\alpha_{\mathcal{K}}$-strongly convex with respect to $\|\cdot\|$. Under these two additional assumptions alone we prove the following theorem.

Theorem 12. *Let $x^* = \arg\min_{x\in\mathcal{K}} f(x)$ and let $M = \frac{\sqrt{\alpha_f}\alpha_K}{8\sqrt{2}\beta_f}$. Denote $D_{\mathcal{K}} = \max_{x,y\in\mathcal{K}} \|x-y\|$. For every $t \geq 1$ the iterate x_t of Algorithm 8 satisfies*

$$f(x_t) - f(x^*) \leq \frac{\max\{\frac{9}{2}\beta_f D_{\mathcal{K}}^2, 18M^{-2}\}}{(t+2)^2} = O\left(\frac{1}{t^2}\right).$$

3.2 Proof of Theorem 12

We denote the approximation error of the iterate x_t produced by the algorithm by h_t. That is $h_t = f(x_t) - f(x^*)$ where $x^* = \arg\min_{x\in\mathcal{K}} f(x)$.

To better illustrate our results, we first shortly revisit the proof technique of Theorem 11. The main observation to be made is the following:

$$\begin{aligned} h_{t+1} &= f(x_t + \eta_t(p_t - x_t)) - f(x^*) \\ &\leq h_t + \eta_t(p_t - x_t)\cdot\nabla f(x_t) + \frac{\eta_t^2\beta_f}{2}\|p_t - x_t\|^2 \\ &\leq h_t + \eta_t(x^* - x_t)\cdot\nabla f(x_t) + \frac{\eta_t^2\beta_f}{2}\|p_t - x_t\|^2 \\ &\leq (1-\eta_t)h_t + \frac{\eta_t^2\beta_f}{2}\|p_t - x_t\|^2, \end{aligned} \tag{3.3}$$

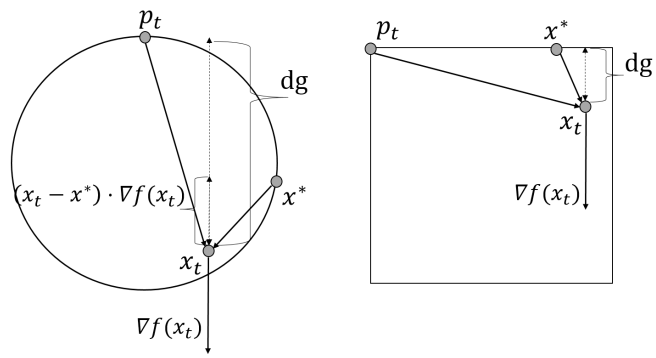

Figure 3.1: For strongly convex sets, as in the left picture, the duality gap (denoted dg) increases with $\|p_t - x_t\|^2$, which accelerates the convergence of the conditional gradient method. As shown in the picture on the right, this property clearly does not hold for arbitrary convex sets.

where the the first inequality follows from the smoothness of f, the second from the optimality of p_t and the third from convexity of f. Choosing η_t to be roughly $1/t$ yields the convergence rate of $1/t$ stated in Theorem 11. This rate cannot be improved in general since while the so-called *duality gap* $(x_t - p_t) \cdot \nabla f(x_t)$ could be arbitrarily small (as small as $(x_t - x^*) \cdot \nabla f(x_t)$), the quantity $\|p_t - x_t\|$ may remain as large as the diameter of the set. Note that in case f is strongly-convex, then according to Eq. (3.1) it holds that x_t converges to x^* and thus according to Eq. (3.3) it suffices to solve the inner linear optimization problem in Algorithm 8 on the intersection of $\mathcal{K}$ and a small ball centered at x_t. As a result the quantity $\|p_t - x_t\|^2$ will be proportional to the approximation error at time t, and a linear convergence rate will be attained. However in general, under the linear oracle assumption, we have no way to solve the linear optimization problem over the intersection of $\mathcal{K}$ and a ball without greatly increasing the number of calls to the linear oracle, which is the most expensive step in many settings.

In case the feasible set $\mathcal{K}$ is strongly convex, then the main observation to be made is that while the quantity $\|p_t - x_t\|$ may still be much larger than $\|x^* - x_t\|$ (the distance to the optimum), in this case, the *duality gap* must also be large, which results in faster convergence. This observation is illustrated in Figure 3.1 and given formally in Lemma 21.

Lemma 21. *On any iteration t of Algorithm 8 it holds that*

$$h_{t+1} \leq h_t \cdot \max\{\frac{1}{2}, 1 - \frac{\alpha_K \|\nabla f(x_t)\|_*}{8\beta_f}\}.$$

Proof. By the optimality of the point p_t we have that

$$(p_t - x_t) \cdot \nabla f(x_t) \quad \leq \quad (x^* - x_t) \cdot \nabla f(x_t) \leq f(x^*) - f(x_t) = -h_t, \tag{3.4}$$

where the second inequality follows from convexity of f. Denote $c_t = \frac{1}{2}(x_t + p_t)$ and $w_t \in \arg\min_{w \in \mathbf{E}, \|w\| \leq 1} w \cdot \nabla f(x_t)$. Note that from Holder's inequality we have that $w_t \cdot \nabla f(x_t) = -\|\nabla f(x_t)\|_*$. Using the strong convexity of the set $\mathcal{K}$ we have that the point $\tilde{p}_t = c_t + \frac{\alpha_K}{8}\|x_t - p_t\|^2 w_t$ is in $\mathcal{K}$. Again using the optimality of p_t we have that

$$\begin{aligned}
(p_t - x_t) \cdot \nabla f(x_t) &\leq (\tilde{p}_t - x_t) \cdot \nabla f(x_t) \\
&= \frac{1}{2}(p_t - x_t) \cdot \nabla f(x_t) + \frac{\alpha_K \|x_t - p_t\|^2}{8} w_t \cdot \nabla f(x_t) \\
&\leq -\frac{1}{2} h_t - \frac{\alpha_K \|x_t - p_t\|^2}{8} \|\nabla f(x_t)\|_*, \qquad (3.5)
\end{aligned}$$

where the last inequality follows from Eq. (3.4).

We now analyze the decrease in the approximation error h_{t+1}. By smoothness of f we have

$$f(x_{t+1}) \quad \leq \quad f(x_t) + \eta_t (p_t - x_t) \cdot \nabla f(x_t) + \frac{\beta_f}{2} \eta_t^2 \|p_t - x_t\|^2.$$

Subtracting $f(x^*)$ from both sides we have

$$h_{t+1} \leq h_t + \eta_t (p_t - x_t) \cdot \nabla f(x_t) + \frac{\beta_f}{2} \eta_t^2 \|p_t - x_t\|^2. \tag{3.6}$$

Plugging Eq. (3.5) we have

$$\begin{aligned}
h_{t+1} \quad &\leq \quad h_t \left(1 - \frac{\eta_t}{2}\right) - \eta_t \frac{\alpha_K \|x_t - p_t\|^2}{8} \|\nabla f(x_t)\|_* + \frac{\beta_f}{2} \eta_t^2 \|p_t - x_t\|^2 \\
&= \quad h_t \left(1 - \frac{\eta_t}{2}\right) + \frac{\|x_t - p_t\|^2}{2} \left(\eta_t^2 \beta_f - \eta_t \frac{\alpha_K \|\nabla f(x_t)\|_*}{4}\right).
\end{aligned}$$

In case $\frac{\alpha_K \|\nabla f(x_t)\|_*}{4} \geq \beta_f$, by the optimal choice of η_t in Algorithm 8, we can set $\eta_t = 1$ and get

$$h_{t+1} \leq \frac{h_t}{2}.$$

Otherwise, we can set $\eta_t = \frac{\alpha_K \|\nabla f(x_t)\|_*}{4\beta_f}$ and get

$$h_{t+1} \quad \leq \quad h_t \left(1 - \frac{\alpha_K \|\nabla f(x_t)\|_*}{8\beta_f}\right).$$

□

Note that Lemma 21 only relies on the strong convexity of the set $\mathcal{K}$ and did not assume anything regrading f beyond convexity and smoothness. In particular it does not require f to be strongly convex.

We can now prove Theorem 12.

Proof. Let $M = \frac{\sqrt{\alpha_f}\alpha_K}{8\sqrt{2}\beta_f}$ and $C = \max\{\frac{9}{2}\beta_f D_{\mathcal{K}}^2, 18M^{-2}\}$. We prove by induction that for all $t \geq 1$, $h_t \leq \frac{C}{(t+2)^2}$.

Since we assume that the objective function f satisfies Eq. (3.2), we have from Lemma 21 that on any iteration t,

$$\begin{aligned} h_{t+1} &\leq h_t \cdot \max\{\frac{1}{2}, 1 - \frac{\alpha_K\sqrt{\alpha_f}}{8\sqrt{2}\beta_f}\sqrt{h_t}\} \\ &= h_t \cdot \max\{\frac{1}{2}, 1 - M h_t^{1/2}\}. \end{aligned} \tag{3.7}$$

For the base case $t = 1$ we need to prove that $h_1 = f(x_1) - f(x^*) \leq C/4$. By β_f smoothness of f we have

$$\begin{aligned} f(x_1) - f(x^*) &= f(x_0 + \eta_0(p_0 - x_0)) - f(x^*) \\ &\leq h_0 + \eta_0(p_0 - x_0) \cdot \nabla f(x_0) + \frac{\beta_f \eta_0^2}{2} D_{\mathcal{K}}^2 \\ &\leq h_0(1 - \eta_0) + \frac{\beta_f \eta_0^2}{2} D_{\mathcal{K}}^2, \end{aligned}$$

where the last inequality follows from convexity of f. By the optimal choice of η_0 we can in particular set $\eta_0 = 1$ which gives $h_1 \leq \frac{\beta_f}{2} D_{\mathcal{K}}^2 \leq C/9$.

Assume now that the induction holds for time $t \geq 1$, that is $h_t \leq \frac{C}{(t+2)^2}$.

If the result of the max operation in Eq. (3.7) is the first argument, that is 1/2, we have that

$$h_{t+1} \quad \leq \quad \frac{h_t}{2} \leq \frac{C}{2(t+2)^2} = \frac{C}{(t+3)^2} \cdot \frac{(t+3)^2}{2(t+2)^2} \leq \frac{C}{(t+3)^2}, \tag{3.8}$$

where the last inequality holds for any $t \geq 1$.

We now turn to the case in which the result of the max operation in Eq. (3.7) is the second argument. We consider two cases.

If $h_t \leq \frac{C}{2(t+2)^2}$ then as in Eq. (3.8) it holds for any $t \geq 1$ that

$$h_{t+1} \leq h_t \leq \frac{C}{2(t+2)^2} \leq \frac{C}{(t+3)^2},$$

where the first inequality follows from Eq. (3.7).

Otherwise, $h_t > \frac{C}{2(t+2)^2}$. By Eq. (3.7) and the induction assumption we have

$$\begin{aligned}
h_{t+1} &\leq h_t \left(1 - M h_t^{1/2}\right) \\
&< \frac{C}{(t+2)^2} \left(1 - M\sqrt{\frac{C}{2}} \frac{1}{t+2}\right) \\
&= \frac{C}{(t+3)^2} \cdot \frac{(t+3)^2}{(t+2)^2} \left(1 - M\sqrt{\frac{C}{2}} \frac{1}{t+2}\right) \\
&= \frac{C}{(t+3)^2} \cdot \frac{(t+2)^2 + 2t + 5}{(t+2)^2} \left(1 - M\sqrt{\frac{C}{2}} \frac{1}{t+2}\right) \\
&< \frac{C}{(t+3)^2} \left(1 + \frac{3(t+2)}{(t+2)^2}\right) \left(1 - M\sqrt{\frac{C}{2}} \frac{1}{t+2}\right) \\
&= \frac{C}{(t+3)^2} \left(1 + \frac{3}{t+2}\right) \left(1 - M\sqrt{\frac{C}{2}} \frac{1}{t+2}\right).
\end{aligned}$$

Thus for $C \geq \frac{18}{M^2}$ we have that

$$h_{t+1} \leq \frac{C}{(t+3)^2} \left(1 + \frac{3}{t+2}\right) \left(1 - \frac{3}{t+2}\right) < \frac{C}{(t+3)^2}.$$

□

3.3 Derivation of Previous Fast Rates Results and Extensions

3.3.1 Deriving the linear rate of Polayk & Levitin

Polyak & Levitin considered in [65] the case in which the feasible set is strongly convex, the objective function is smooth and there exists a constant $g > 0$ such that

$$\forall x \in \mathcal{K}: \quad \|\nabla f(x)\|_* \geq g. \tag{3.9}$$

They showed that under the lower-bounded gradient assumption, Algorithm 8 converges with a linear rate, that is $e^{-\Theta(t)}$. Clearly by plugging Eq. (3.9) into Lemma 21 we have that on each iteration t

$$h_{t+1} \leq h_t \cdot \max\{\frac{1}{2}, 1 - \frac{\alpha_k g}{8\beta_f}\}.$$

which results in the same exponentially fast convergence rate as in [65] and following works such as [23, 28].

3.3.2 Deriving a linear rate for arbitrary convex sets in case the optimum is in the interior

Assume now that the feasible set $\mathcal{K}$ is convex but not necessarily strongly convex. We assume that the objective function f is smooth, convex, satisfies Eq. (3.2) with some constant α_f and admits a minimizer (not necessarily unique) x^* that lies in the interior of $\mathcal{K}$, i.e. there exists a parameter $r > 0$ such that the ball of radius r with respect to norm $\|\cdot\|$ centered at x^* is fully contained in $\mathcal{K}$ [2]. GuéLat and Marcotte [41] showed the under the above conditions, the conditional gradient algorithm converges with a linear rate. We now show how a slight modification in the proof of Lemma 21 yields this linear convergence result.

Let w_t be as in the proof of Lemma 21, that is $w_t \in \arg\min_{w \in \mathbf{E}, \|w\| \leq 1} w \cdot \nabla f(x_t)$. Instead of defining the point $\tilde{p}_t$ as in the proof of Lemma 21 we

[2] We assume here that $\mathcal{K}$ is full-dimensional. In any other case, we can assume instead that the intersection of the ball centered at x^* with the smallest subspace containing $\mathcal{K}$ is fully contained in $\mathcal{K}$. In this case we also need to replace the gradient $\nabla f(x)$ with its projection onto this subspace, see also footnote 1.

define it to be $\tilde{p}_t = x^* + rw_t$. Because of our assumption on the location of x^*, it holds that $\tilde{p}_t \in \mathcal{K}$. We thus have that

$$(\tilde{p}_t - x_t) \cdot \nabla f(x_t) = (x_t^* - x_t) \cdot \nabla f(x_t) + rw_t \cdot \nabla f(x_t) \leq -r\|\nabla f(x_t)\|_*.$$

Plugging this into Eq. (3.6) we have

$$\begin{aligned} h_{t+1} &\leq h_t - \eta_t r\|\nabla f(x_t)\|_* + \frac{\beta_f \eta_t^2 D_{\mathcal{K}}^2}{2} \\ &\leq h_t - \eta_t r\sqrt{\frac{\alpha_f}{2}}\sqrt{h_t} + \frac{\beta_f \eta_t^2 D_{\mathcal{K}}^2}{2}. \end{aligned}$$

where $D_{\mathcal{K}}$ denotes the diameter of $\mathcal{K}$ with respect to norm $\|\cdot\|$ and the second inequality follows from Eq. (3.2). By the optimal choice of η_t, we can set $\eta_t = \frac{r\sqrt{\alpha_f}\sqrt{h_t}}{\sqrt{2}\beta_f D_{\mathcal{K}}^2}$ and get

$$h_{t+1} \leq h_t - \frac{r^2 \alpha_f}{4\beta_f D_{\mathcal{K}}^2} h_t,$$

which results in a linear convergence result.

3.3.3 Relaxing the strong convexity of the objective function

So far we have considered the case in which the objective function f is strongly convex. Notice however that our main instrument for proving the accelerated convergence rate, i.e. Lemma 21, did not rely directly on strong convexity of f, but on the magnitude of the gradient, $\|\nabla f(x_t)\|_*$. We have seen in Eq. (3.2) that indeed if f is strongly convex than the gradient is at least of the order of $\sqrt{h_t}$. We now show that there exists functions which are not strongly convex but still satisfy Eq. (3.2) and hence our results apply also for them.

Consider a function f of the following form:

$$f(x) = g(Ax),$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $g : \mathbb{R}^m \to \mathbb{R}$ is β_g-smooth and α_g-strongly convex over $\mathbb{R}^m$ with respect to the ℓ_2 norm. Assume further that $m < n$ and all rows of A are linearly independent. Note that since A maps x to a low-

dimensional subspace, the function $f(x)$ is not strongly convex. However, $f(x)$ is smooth. To see this, note that the gradient vector of $f(x)$ is given by $\nabla f(x) = A^\top g(Ax)$, and thus, for any $x, y \in \mathbb{R}^n$ it holds that

$$\begin{aligned}
\|\nabla f(x) - \nabla f(y)\|_2^2 &= \|A^\top(\nabla g(Ax) - \nabla g(Ay))\|_2^2 \\
&\leq \lambda_{\max}(AA^\top) \cdot \|\nabla g(Ax) - \nabla g(Ay)\|_2^2 \\
&\leq \lambda_{\max}(AA^\top)\beta_g^2 \cdot \|Ax - Ay\|_2^2 \\
&\leq \lambda_{\max}(AA^\top)\lambda_{\max}(A^\top A)\beta_g^2 \cdot \|x - y\|_2^2,
\end{aligned}$$

where the second inequality follows from the smoothness of g, and the notation $\lambda_{\max}(M)$ is used to denote the largest eigenvalue of a matrix M. Thus we have that $f(x)$ has a $\sqrt{\lambda_{\max}(AA^\top)\lambda_{\max}(A^\top A)}\beta_g$-Lipshitz gradient and thus it is smooth. Moreover, we now show that $f(x)$ indeed satisfies the condition in Eq. (3.2):

$$\begin{aligned}
\|\nabla f(x)\|_2^2 &= \|A^\top \nabla g(Ax)\|_2^2 \geq \lambda_{\min}(AA^\top) \cdot \|\nabla g(Ax)\|_2^2 \\
&\geq \lambda_{\min}(AA^\top) \cdot \frac{\alpha_g}{2}\left(g(Ax) - g(Ax^*)\right) \\
&= \frac{\lambda_{\min}(AA^\top)\alpha_g}{2}\left(f(x) - f(x^*)\right),
\end{aligned}$$

where the inequality follows from applying Eq. (3.2) with respect to the function g, and $\lambda_{\min}(AA^\top)$ is used to denote the smallest eigenvalue of the matrix $AA^\top$. Note that by our assumption on the matrix A, it indeed follows that $\lambda_{\min}(AA^\top) > 0$.

For instance, a case of interest that fits the above assumptions is when $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, where A is a matrix as discussed above. In this case the optimization problem $\min_{x \in \mathcal{K}} f(x)$ is the problem of finding a point in $\mathcal{K}$ that best satisfies an under-determined linear system in terms of the mean square error. An application of the conditional gradient method to this problem was studied in [10]. Combining the result of this subsection for this choice of $f(x)$ with the result of the previous subsection yields the linear convergence rate of the conditional gradient method applied to the convex feasibility problem studied in [10].

$\mathbf{E}$	Domain	S.C. parameter	Lin. opt. comp.
$\mathbb{R}^n$	ℓ_p ball, $p \in (1,2]$	$\frac{p-1}{r}$	$O(\text{nnz})$
$\mathbb{R}^{m\times n}$	Schatten ℓ_p ball, $p \in (1,2]$	$\frac{p-1}{r}$	$O(n^3)$ (SVD)
$\mathbb{R}^{m\times n}$	Group $\ell_{s,p}$ ball, $s,p \in (1,2]$	$\frac{(s-1)(p-1)}{(s+p-2)r}$	$O(\text{nnz})$

Table 3.2: Examples of strongly convex sets with corresponding strong convexity parameter and complexity of a linear optimization oracle implementation . nnz denotes the number of non-zero entries in the linear objective and $\sigma(X)$ denotes the vector of singular values.

3.4 Examples of Strongly Convex Sets

In this section we explore convex sets for which Theorem 12 is applicable. That is, convex sets which on one hand are strongly convex, and on the other, admit a simple and efficient implementation of a linear optimization oracle. We show that various norms that give rise to natural regularization functions in machine learning, induce convex sets that fit both of the above requirements. A summary of our findings is given in Table 3.4. We note that in all cases in which the norm parameter p is smaller than 2 (or one of the parameters s, p in case of group norms), we are not aware of a practical algorithm for computing the projection.

3.4.1 Partial characterization of strongly convex sets

The following lemma is taken from [54] (Theorem 12).

Lemma 22. *Let $\boldsymbol{E}$ be a finite vector space and let $f : \boldsymbol{E} \to \mathbb{R}$ be non-negative, α-strongly convex and β-smooth. Then the set $\mathcal{K} = \{x \mid f(x) \leq r\}$ is $\frac{\alpha}{\sqrt{2\beta r}}$-strongly convex.*

This lemma for instance shows that the Euclidean ball of radius r is $1/r$-strongly convex (by applying the lemma with $f = \|x\|_2^2$).

The following lemma will be useful to prove that convex sets that are induced by certain norms, which do not correspond to a smooth function as in the previous lemma, are strongly convex. The proof is given in Section 3.6.

Lemma 23. *Let $\boldsymbol{E}$ be a finite vector space, let $\|\cdot\|$ be a norm over $\boldsymbol{E}$ and assume that the function $f(x) = \|x\|^2$ is α-strongly convex over $\boldsymbol{E}$ with respect to the norm $\|\cdot\|$. Then for any $r > 0$, the set $\mathbb{B}_{\|\cdot\|}(r) = \{x \in \boldsymbol{E} \mid \|x\| \leq r\}$ is $\frac{\alpha}{2r}$-strongly convex with respect to $\|\cdot\|$.*

3.4.2 ℓ_p balls for $p \in (1, 2]$

Given a parameter $p \geq 1$, consider the ℓ_p ball of radius r,

$$\mathbb{B}_p(r) = \{x \in \mathbb{R}^n \mid \|x\|_p \leq r\}.$$

The following lemma is proved in [84].

Lemma 24. *Fix $p \in (1, 2]$. The function $\frac{1}{2}\|x\|_p^2$ is $(p-1)$-strongly convex w.r.t. the norm $\|\cdot\|_p$.*

The following corollary is a consequence of combining Lemma 24 and Lemma 23. The proof is given in Section 3.6.

Corollary 3. *Fix $p \in (1, 2]$. The set $\mathbb{B}_p(r)$ is $\frac{p-1}{r}$-strongly convex with respect to the norm $\|\cdot\|_p$ and $\frac{(p-1)n^{\frac{1}{2}-\frac{1}{p}}}{r}$-strongly convex with respect to the norm $\|\cdot\|_2$.*

The following lemma establishes that linear optimization over $\mathbb{B}_p(r)$ admits a simple closed-form solution that can be computed in time that is linear in the number of non-zeros in the linear objective. The proof is given in Section 3.6.

Lemma 25. *Fix $p \in (1, 2]$, $r > 0$ and a linear objective $c \in \mathbb{R}^n$. Let $x \in \mathbb{R}^n$ such that $x_i = -\frac{r}{\|c\|_q^{q-1}} sgn(c_i)|c_i|^{q-1}$ where q satisfies: $1/q + 1/p = 1$, and $sgn(\cdot)$ is the sign function. Then $x = \arg\min_{y \in \mathbb{B}_p(r)} y \cdot c$*

3.4.3 Schatten ℓ_p balls for $p \in (1, 2]$

Given a matrix $X \in \mathbb{R}^{m \times n}$ we denote by $\sigma(X)$ the vector of singular values of X in descending order, that is $\sigma(X)_1 \geq \sigma(X)_2 \geq ...\sigma(X)_{\min(m,n)}$. The Schatten ℓ_p norm is given by

$$\|X\|_{S(p)} = \|\sigma(X)\|_p = \left(\sum_{i=1}^{\min(m,n)} \sigma(X)_i^p \right)^{1/p}.$$

Consider the Schatten ℓ_p ball of radius r,

$$\mathbb{B}_{S(p)}(r) = \{X \in \mathbb{R}^{m\times n} \mid \|X\|_{S(p)} \leq r\}.$$

The following lemma is taken from [55].

Lemma 26. *Fix $p \in (1,2]$. The function $\frac{1}{2}\|X\|^2_{S(p)}$ is $(p-1)$-strongly convex w.r.t. the norm $\|\cdot\|_{S(p)}$.*

The proof of the following corollary follows the exact same lines as the proof of Corollary 3 by using Lemma 26 instead of Lemma 24.

Corollary 4. *Fix $p \in (1,2]$. The set $\mathbb{B}_{S(p)}(r)$ is $\frac{p-1}{r}$-strongly convex with respect to the norm $\|\cdot\|_{S(p)}$ and $\frac{(p-1)\min(m,n)^{\frac{1}{2}-\frac{1}{p}}}{r}$-strongly convex with respect to the frobenius norm $\|\cdot\|_F$.*

The following lemma establishes that linear optimization over $\mathbb{B}_{S(p)}(r)$ admits a simple closed-form solution given the *singular value decomposition* of the linear objective. The proof is given in Section 3.6.

Lemma 27. *Fix $p \in (1,2]$, $r > 0$ and a linear objective $C \in \mathbb{R}^{m\times n}$. Let $C = U\Sigma V^\top$ be the singular value decomposition of C. Let σ be a vector such that $\sigma_i = -\frac{r}{\|\sigma(C)\|_q^{q-1}}\sigma(C)_i^{q-1}$ where q satisfies: $1/q + 1/p = 1$. Finally, let $X = UDiag(\sigma)V^\top$ where $Diag(\sigma)$ is an $m \times n$ diagonal matrix with the vector σ as the main diagonal. Then $X = \arg\min_{Y\in\mathbb{B}_{S(p)}(r)} Y \bullet C$, where $\bullet$ denotes the standard inner product for matrices.*

3.4.4 Group $\ell_{s,p}$ balls for $s, p \in (1,2]$

Given a matrix $X \in \mathbb{R}^{m\times n}$ denote by $X_i \in \mathbb{R}^n$ the ith row of X. That is $X = (X_1, X_2, ..., X_m)^\top$.

The $\ell_{s,p}$ norm of X is given by,

$$\|X\|_{s,p} = \|(\|X_1\|_s, \|X_2\|_s, ..., \|X_m\|_s)\|_p.$$

We define the $\ell_{s,p}$ ball as follows:

$$\mathbb{B}_{s,p}(r) = \{X \in \mathbb{R}^{m\times n} \mid \|X\|_{s,p} \leq r\}.$$

The proof of the following lemma is given in Section 3.6.

Lemma 28. *Let $s, p \in (1, 2]$. The set $\mathbb{B}_{s,p}(r)$ is $\frac{(s-1)(p-1)}{(s+p-2)r}$-strongly convex with respect to the norm $\|\cdot\|_{s,p}$ and $n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} \frac{(s-1)(p-1)}{(s+p-2)r}$-strongly convex with respect to the frobenius norm $\|\cdot\|_F$.*

The following lemma establishes that linear optimization over $\mathbb{B}_{s,p}(r)$ admits a simple closed-form solution that can be computed in time that is linear in the number of non-zeros in the linear objective. The proof is given in Section 3.6.

Lemma 29. *Fix $s, p \in (1, 2]$, $r > 0$ and a linear objective $C \in \mathbb{R}^{m \times n}$. Let $X \in \mathbb{R}^{m \times n}$ be such that $X_{i,j} = -\frac{r}{\|C\|_{z,q}^{q-1} \|C_i\|_z^{z-q}} sgn(C_{i,j}) |C_{i,j}|^{z-1}$ where z satisfies: $1/s + 1/z = 1$, q satisfies: $1/p + 1/q = 1$ and C_i denotes the ith row of C. Then $X = \arg\min_{Y \in \mathbb{B}_{s,p}(r)} Y \bullet C$, where $\bullet$ denotes the standard inner product for matrices.*

3.5 Proof of Theorem 11

Proof. Fix an iteration t. By the β_f-smoothness of f we have that

$$\begin{aligned} h_{t+1} &= f(x_t + \eta_t(p_t - x_t)) - f(x^*) \\ &\leq f(x_t) - f(x^*) + \eta_t(p_t - x_t) \cdot \nabla f(x_t) + \frac{\eta_t^2 \beta_f}{2} \|p_t - x_t\|^2 \\ &\leq h_t - \eta_t h_t + \frac{\eta_t^2 \beta_f D_{\mathcal{K}}^2}{2}, \end{aligned} \tag{3.10}$$

where the last inequality follows from convexity of f. Notice that by the optimal choice of η_t in Algorithm 8, it holds in particular that $h_{t+1} \leq h_t$ (by setting $\eta_t = 0$ in Eq. (3.10)).

Fix $C = 8\beta_f D_{\mathcal{K}}^2$. We now prove by induction on t that $h_t \leq \frac{C}{t}$.

For the base case $t = 1$ we notice that by the optimal choice of η_0 in Algorithm 8 we can in particular set $\eta_0 = 1$ and thus it follows from Eq. (3.10) that $h_1 \leq \frac{\beta_f D_{\mathcal{K}}^2}{2} < C$ as needed.

Assume now that the induction holds for $t \geq 1$. That is $h_t \leq \frac{C}{t}$. We consider two cases.

If $h_t \leq \frac{C}{2t}$ then we have

$$h_{t+1} \leq h_t \leq \frac{C}{2t} = \frac{C}{t+1} \cdot \frac{t+1}{2t} \leq \frac{C}{t+1},$$

where the last inequality holds for any $t \geq 1$.

Otherwise it holds that $h_t > \frac{C}{2t}$. Using Eq. (3.10) again we have

$$h_{t+1} \leq h_t - \eta_t h_t + \frac{\eta_t^2 \beta_f D_{\mathcal{K}}^2}{2}.$$

By the optimal choice of η_t in Algorithm 8 we can set $\eta_t = \frac{h_t}{\beta_f D_{\mathcal{K}}^2}$ and get

$$\begin{aligned} h_{t+1} &\leq h_t - \frac{1}{2\beta_f D_{\mathcal{K}}^2} h_t^2 < \frac{C}{t} - \frac{C^2}{8\beta_f D_{\mathcal{K}}^2 t^2} \\ &= \frac{C}{t+1}\left(\frac{t+1}{t} - \frac{C(t+1)}{8\beta_f D_{\mathcal{K}}^2 t^2}\right) \\ &< \frac{C}{t+1}\left(1 + \frac{1}{t} - \frac{Ct}{8\beta_f D_{\mathcal{K}}^2 t^2}\right). \end{aligned}$$

Thus for $C \geq 8\beta_f D_{\mathcal{K}}^2$ we have that $h_{t+1} \leq \frac{C}{t+1}$. □

3.6 Proofs of Lemmas and Corollaries from Section 3.4

Proof of Lemma 23

Proof. It suffices to show that given $x, y \in \mathbf{E}$ such that $f(x) \leq r^2, f(y) \leq r^2$, a scalar $\gamma \in [0,1]$ and $z \in \mathbf{E}$ such that $\|z\| \leq \frac{\alpha}{4r}\gamma(1-\gamma)\|x-y\|^2$, it holds that, $f(\gamma x + (1-\gamma)y + z) \leq r^2$.

By the definition of f and the triangle inequality for $\|\cdot\|$ we have

$$\begin{aligned} f(\gamma x + (1-\gamma)y + z) &= \|\gamma x + (1-\gamma)y + z\|^2 \\ &\leq (\|\gamma x + (1-\gamma)y\| + \|z\|)^2 \\ &= \left(\sqrt{f(\gamma x + (1-\gamma)y)} + \|z\|\right)^2. \end{aligned} \tag{3.11}$$

Since f is α strongly convex with respect to $\|\cdot\|$ we have that

$$\begin{aligned} f(\gamma x + (1-\gamma)y) &\leq \gamma f(x) + (1-\gamma)f(y) - \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|^2 \\ &\leq r^2 - \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|^2. \end{aligned}$$

The function $g(t) = \sqrt{t}$ is concave, meaning $\sqrt{a-b} = g(a-b) \leq g(a) - g'(a) \cdot b = \sqrt{a} - \frac{b}{2\sqrt{a}}$. Thus,

$$\begin{aligned}\sqrt{f(\gamma x + (1-\gamma)y)} &\leq \sqrt{r^2 - \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|^2} \\ &\leq r - \frac{\alpha\gamma(1-\gamma)\|x-y\|^2}{4r}.\end{aligned}$$

Plugging back in Eq. (3.11) we have

$$f(\gamma x + (1-\gamma)y + z) \leq \left(r - \frac{\alpha\gamma(1-\gamma)\|x-y\|^2}{4r} + \|z\|\right)^2.$$

By our assumption on $\|z\|$ we have

$$\begin{aligned}f(\gamma x + (1-\gamma)y + z) &\leq \left(r - \frac{\alpha\gamma(1-\gamma)\|x-y\|^2}{4r} + \frac{\alpha}{4r}\gamma(1-\gamma)\|x-y\|^2\right)^2 \\ &= r^2.\end{aligned}$$

□

Proof of Corollary 3

Proof. The strong convexity of the set w.r.t. $\|\cdot\|_p$ is an immediate consequence of Lemma 23.

Since $\mathbb{B}_p(r)$ is $\alpha = (p-1)/r$ strongly convex w.r.t. the norm $\|\cdot\|_p$, we have that given $x, y \in \mathbb{B}_p(r)$, $\gamma \in [0,1]$ and $z \in \mathbb{R}^n$ such that $\|z\|_p \leq 1$ it holds that

$$\gamma x + (1-\gamma)y + \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_p^2 z \in \mathbb{B}_p(r).$$

For any $p \in (1,2]$ and vector $v \in \mathbb{R}^n$ it holds that

$$\|v\|_2 \leq \|v\|_p \leq n^{\frac{1}{p}-\frac{1}{2}}\|v\|_2. \tag{3.12}$$

Given a vector $z' \in \mathbb{R}^n$ such that $\|z'\|_F \leq 1$ we have that

$$\|\frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_2^2 z'\|_p = \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_2^2\|z'\|_p.$$

Using Eq. (3.12) we have

$$\begin{aligned}\|\frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_2^2 z'\|_p &\leq \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_p^2 n^{\frac{1}{p}-\frac{1}{2}}\|z'\|_2 \\ &\leq \frac{\alpha n^{\frac{1}{p}-\frac{1}{2}}}{2}\gamma(1-\gamma)\|x-y\|_p^2.\end{aligned}$$

Hence, $\mathbb{B}_p(r)$ is $\alpha n^{\frac{1}{2}-\frac{1}{p}} = \frac{(p-1)n^{\frac{1}{2}-\frac{1}{p}}}{r}$-strongly convex with respect to $\|\cdot\|_2$. □

Proof of Lemma 25

Proof. Since $\|\cdot\|_p$ and $\|\cdot\|_q$ are dual norms, we have using Holder's inequality that for all $x \in \mathbb{B}_p(r)$,

$$x \cdot c \geq -\|x\|_p\|c\|_q \geq -r\|c\|_q.$$

Thus choosing $x_i = -\frac{r}{\|c\|_q^{q-1}}\text{sgn}(c_i)|c_i|^{q-1}$ we have that

$$\begin{aligned}x \cdot c &= -\sum_{i=1}^{n}\frac{r}{\|c\|_q^{q-1}}\text{sgn}(c_i)|c_i|^{q-1}\cdot c_i \\ &= -\sum_{i=1}^{n}\frac{r}{\|c\|_q^{q-1}}|c_i|^q = -\frac{r}{\|c\|_q^{q-1}}\|c\|_q^q \\ &= -r\|c\|_q.\end{aligned}$$

Moreover,

$$\|x\|_p^p = \frac{r^p}{\left(\|c\|_q^{q-1}\right)^p}\sum_{i=1}^{n}\left(|c_i|^{q-1}\right)^p.$$

Since $p = q/(q-1)$ we have that

$$\|x\|_p^p = \frac{r^p}{\|c\|_q^q}\sum_{i=1}^{n}|c_i|^q = r^p.$$

Thus we have that $x \in \mathbb{B}_p(r)$. □

Proof of Lemma 27

Proof. Since $\|\cdot\|_{S(p)}$ and $\|\cdot\|_{S(q)}$ are dual norms we have from Holder's inequality that for all $X \in \mathbb{B}_{S(p)}(r)$,

$$X \bullet C \geq -\|X\|_{S(p)}\|C\|_{S(q)} \geq -r\|C\|_{S(q)} = r\|\sigma(C)\|_q.$$

By our choice of X we have that

$$\begin{aligned}
X \bullet C &= \mathrm{Tr}(X^\top C) = \mathrm{Tr}(V Diag(\sigma)^\top U^\top U \Sigma V^\top) \\
&= \mathrm{Tr}(V Diag(\sigma)^\top \Sigma V^\top) \\
&= \mathrm{Tr}(V^\top V Diag(\sigma)^\top \Sigma) = \mathrm{Tr}(Diag(\sigma)^\top \Sigma) \\
&= \sum_{i=1}^{\min(m,n)} -\frac{r}{\|\sigma(C)\|_q^{q-1}} \sigma(C)_i^{q-1} \cdot \sigma(C)_i \\
&= -\frac{r}{\|\sigma(C)\|_q^{q-1}} \sum_{i=1}^{\min(m,n)} \sigma(C)_i^q \\
&= -r\|\sigma(C)\|_q.
\end{aligned}$$

Moreover,

$$\|X\|_{S(p)}^p = \|\sigma(X)\|_p^p = \frac{r^p}{\left(\|\sigma(C)\|_q^{q-1}\right)^p} \sum_{i=1}^n \left(\sigma(C)_i^{q-1}\right)^p.$$

Since $p = q/(q-1)$ we have that

$$\|X\|_{S(p)}^p = \frac{r^p}{\|\sigma(C)\|_q^q} \sum_{i=1}^n |\sigma(C)_i|^q = r^p.$$

Thus we have that $X \in \mathbb{B}_{S(p)}(r)$. □

Proof of Lemma 28

The following lemma will be of use in the proof.

Lemma 30. *for any matrix $A \in \mathbb{R}^{m\times n}$ and $s, p \in (1, 2]$ it holds that*

$$\|A\|_F \leq \|A\|_{s,p} \leq n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} \|A\|_F.$$

Proof. For any vector $v \in \mathbb{R}^n$ and $p \in (1, 2]$ it holds that

$$\|v\|_2 \leq \|v\|_p \leq n^{\frac{1}{p}-\frac{1}{2}} \|v\|_2. \tag{3.13}$$

Denote by A_i the ith row of A. For any $i \in [m]$ and $p \in (1, 2]$ it holds that

$$\|A_i\|_2 \leq \|A_i\|_p \leq n^{\frac{1}{p}-\frac{1}{2}} \|A_i\|_2. \tag{3.14}$$

Note that by definition $\|\cdot\|_F \equiv \|\cdot\|_{2,2}$. Applying Eq. (3.13) and (3.14) we have,

$$\begin{aligned}
\|A\|_F &= \|A\|_{2,2} = \|(\|A_1\|_2, \|A_2\|_2, ..., \|A_m\|_2)\|_2 \\
&\leq \|(\|A_1\|_s, \|A_2\|_s, ..., \|A_m\|_s)\|_p \\
&\leq n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} \|(\|A_1\|_2, \|A_2\|_2, ..., \|A_m\|_2)\|_2 \\
&= n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} \|A\|_F.
\end{aligned}$$

□

We can now prove Lemma 28.

Proof. Let z, q be such that $1/z + 1/s = 1$ and $1/q + 1/p = 1$. Note that $z, q \in [2, \infty)$. The norm $\|\cdot\|_{z,q}$ is the dual norm to $\|\cdot\|_{s,p}$ (see [55] for instance).

According to Lemma 24, the functions $\|x\|_s^2$ and $\|x\|_p^2$ are $\alpha_s = 2(s-1)$-strongly convex w.r.t. $\|\cdot\|_p$ and $\alpha_p = 2(p-1)$-strongly convex w.r.t. $\|\cdot\|_q$ respectively. Hence by the *strong convexity / smoothness duality* (see Theorem 3 in [55]) we have that the functions $\|x\|_z^2$ and $\|x\|_q^2$ are α_s^{-1}-smooth w.r.t. $\|\cdot\|_z$ and α_p^{-1}-smooth w.r.t. $\|\cdot\|_q$ respectively.

By Theorem 13 in [55] we have that the function $\|X\|_{z,q}^2$ is $(\alpha_p^{-1} + \alpha_s^{-1})$-smooth with respect to the norm $\|\cdot\|_{z,q}$. Again using the *strong convexity / smoothness duality* we have that $\|X\|_{s,p}^2$ is $\left(\alpha_p^{-1} + \alpha_s^{-1}\right)^{-1} = \frac{\alpha_p \alpha_s}{\alpha_p + \alpha_s}$ strongly convex with respect to the norm $\|\cdot\|_{s,p}$. The first part of the lemma now follows from applying Lemma 23.

Since $\mathbb{B}_{s,p}(r)$ is $\alpha = \frac{(s-1)(p-1)}{(s+p-2)r}$ strongly convex w.r.t. the norm $\|\cdot\|_{s,p}$, we have that given $X, Y \in \mathbb{B}_{s,p}(r)$, $\gamma \in [0,1]$ and $Z \in \mathbb{R}^{m\times n}$ such that $\|Z\|_{s,p} \leq 1$ it holds that

$$\gamma X + (1-\gamma)Y + \frac{\alpha}{2}\gamma(1-\gamma)\|X-Y\|_{s,p}^2 Z \in \mathbb{B}_{s,p}(r).$$

Given a matrix $Z' \in \mathbb{R}^{m\times n}$ such that $\|Z'\|_F \leq 1$ we have that

$$\|\frac{\alpha}{2}\gamma(1-\gamma)\|X-Y\|_F^2 Z'\|_{s,p} = \frac{\alpha}{2}\gamma(1-\gamma)\|x-y\|_F^2\|Z'\|_{s,p}.$$

Using Lemma 30 we have

$$\begin{aligned}\|\frac{\alpha}{2}\gamma(1-\gamma)\|X-Y\|_F^2 Z'\|_{s,p} &\leq \frac{\alpha}{2}\gamma(1-\gamma)\|X-Y\|_{s,p}^2 n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}}\|Z'\|_F \\ &\leq \frac{\alpha n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}}}{2}\gamma(1-\gamma)\|X-Y\|_{s,p}^2.\end{aligned}$$

Hence, $\mathbb{B}_{s,p}(r)$ is $\alpha n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} = n^{\frac{1}{s}-\frac{1}{2}} m^{\frac{1}{p}-\frac{1}{2}} \frac{(s-1)(p-1)}{(s+p-2)r}$ strongly convex with respect to $\|\cdot\|_F$. □

Proof of Lemma 29

Proof. Since by choice of z, q it holds that $\|\cdot\|_{s,p}, \|\cdot\|_{z,q}$ are dual norms, we have by Holder's inequality that

$$X \bullet C \geq -\|X\|_{s,p}\|C\|_{z,q} \geq -r\|C\|_{z,q}.$$

Thus, choosing

$$X_{i,j} = -\frac{r}{\|C\|_{z,q}^{q-1}\|C_i\|_z^{z-q}}\text{sgn}(C_{i,j})|C_{i,j}|^{z-1},$$

we have that

$$\begin{aligned}
X \bullet C &= \sum_{i\in[m],j\in[n]} X_{i,j}C_{i,j} \\
&= \sum_{i\in[m],j\in[n]} -\frac{r}{\|C\|_{z,q}^{q-1}\|C_i\|_z^{z-q}}\mathrm{sgn}(C_{i,j})|C_{i,j}|^{z-1}\cdot C_{i,j} \\
&= \sum_{i\in[m],j\in[n]} -\frac{r}{\|C\|_{z,q}^{q-1}\|C_i\|_z^{z-q}}|C_{i,j}|^{z} \\
&= \sum_{i\in[m]} -\frac{r}{\|C\|_{z,q}^{q-1}\|C_i\|_z^{z-q}}\sum_{j\in[n]}|C_{i,j}|^{z} \\
&= \sum_{i\in[m]} -\frac{r}{\|C\|_{z,q}^{q-1}\|C_i\|_z^{z-q}}\|C_i\|_z^{z} = \sum_{i\in[m]} -\frac{r}{\|C\|_{z,q}^{q-1}}\|C_i\|_z^{q} \\
&= -\frac{r}{\|C\|_{z,q}^{q-1}}\sum_{i\in[m]}\|C_i\|_z^{q} = -\frac{r}{\|C\|_{z,q}^{q-1}}\|C\|_{z,q}^{q} = -r\|C\|_{z,q}.
\end{aligned}$$

Moreover, for all $i \in [m]$ it holds that

$$\|X_i\|_s^s = \sum_{j=1}^{n}|X_{i,j}|^s = \frac{r^s}{\|C\|_{z,q}^{s(q-1)}\|C_i\|_z^{s(z-q)}}\sum_{i=j}^{n}|C_{i,j}|^{s(z-1)}.$$

Since $s = z/(z-1)$ we have

$$\|X_i\|_s^s = \frac{r^s}{\|C\|_{z,q}^{s(q-1)}\|C_i\|_z^{s(z-q)}}\|C_i\|_z^z = \frac{\|C_i\|_z^{sq-z(s-1)}}{\|C\|_{z,q}^{s(q-1)}}r^s$$

Using $z = s/(s-1)$ we have that

$$\|X_i\|_s^s = \frac{\|C_i\|_z^{s(q-1)}}{\|C\|_{z,q}^{s(q-1)}}r^s.$$

Thus,

$$\|X_i\|_s = \left(\frac{\|C_i\|_z}{\|C\|_{z,q}}\right)^{q-1} r.$$

Finally, we have that

$$
\begin{aligned}
\|X\|_{s,p}^p &= \sum_{i\in[m]} \|X_i\|_s^p = \sum_{i\in[m]} \left(\frac{\|C_i\|_z}{\|C\|_{z,q}}\right)^{p(q-1)} r^p \\
&= \sum_{i\in[m]} \left(\frac{\|C_i\|_z}{\|C\|_{z,q}}\right)^{q} r^p = \frac{r^p}{\|C\|_{z,q}^q} \sum_{i\in[m]} \|C_i\|_z^q = r^p.
\end{aligned}
$$

Thus, $X \in \mathbb{B}_{s,p}(r)$. $\square$

3.7 Open Problems

The following questions naturally arise. It is known that in case the objective function is both smooth and strongly convex, projection/prox-based methods achieve a convergence rate of $O(\log(1/\epsilon))$. Is it possible to achieve such a rate for a conditional gradient-like method in case the feasible set is strongly convex?

We have shown that it is possible to obtain faster rates for optimization over balls induced by norms that give rise to strongly convex functions. Is it possible to obtain faster rates for balls induced by norms that do not give rise to strongly convex functions, but rather to smooth functions? e.g. is it possible to obtain faster rates for ℓ_p balls (and generalizations of) for $p > 2$?

Finally, the most intriguing question is to give a linear optimization oracle-based method that enjoys the same convergence rate, at least in terms of the approximation error, as optimal projection/prox-based gradient methods, in any regime (including non-smooth problems).

Chapter 4

Online Learning of Eigenvectors

Computing the leading eigenvector of a symmetric real matrix is one of the most important problems in numerical linear algebra and an important primitive in many algorithms. Perhaps the best known application of this problem is the *Principal Component Analysis* problem, in which, roughly speaking, given a set of high-dimensional vectors, the problem is to find a low-dimensional subspace such that the projection of the vectors onto this low-dimensional subspace is close on average to the original vectors. It is well known that the optimal solution to this problem is to project the high-dimensional vectors over the several leading eigenvectors of the covariance matrix of the data.

In this chapter we consider an *online learning* problem that is a natural extension of the leading eigenvector problem. A decision maker observes a sequence of matrices. Before a new matrix is revealed, the decision maker must commit to a unit vector. Once the matrix is revealed the decision maker gains the quadratic product of the selected unit vector with the revealed matrix, and his overall goal is to maximize the total reward. As standard in such settings, the performance of the decision maker is measured via the regret which is given by the difference between the total reward of the best fixed unit vector in hindsight and the total reward of the decision maker. Indeed the best fixed unit vector in hindsight is simply given by the leading eigenvector of the sum of revealed matrices and the associated total reward

is the corresponding leading eigenvalue. This problem captures as a special case the problem known as *Online Principal Component Analysis* that was studied in [90], [91], [74], in case of a single principal component.

From an optimization theory point of view, the offline leading eigenvector problem is a non-convex quadratic optimization problem, since w.l.o.g. it requires to maximize a convex function over a non convex set, i.e. the unit sphere. However, it could be solved to high precision via eigendecomposition which takes $O(n^3)$ time where n is the size of the matrix, or via iterative approximation algorithms such as the *Power* or *Lanczos* algorithms whose running time for well-conditioned matrices is roughly $O(\text{nnz})$ where nnz denotes the number of non-zeros entries in the input matrix. When considering the online problem, three different approaches come to mind which we now detail.

The Convexification Approach The first approach is to convexify the problem by *lifiting* the decision variable from a unit vector to a matrix, more specifically a positive semidefinite matrix with unit trace. This approach corresponds to the problem of *Online Linear Optimization* over the *Spectrahedron* [1]. This is also the approach taken in previous works on Online PCA [90], [91], [74]. While this approach leads to theoretically efficient algorithms with nearly optimal regret bounds, such as the *Matrix Multiplicative Weights* algorithm [87, 5], their major drawback is that they require super-linear computation per iteration, i.e. they require to compute a full eigendcomposition which amounts to $O(n^3)$ arithmetic operations per iteration . The latter is true even if the support of the sequence of matrices is sparse.

The Oracle-based Approach A natural approach is to try to reduce the online problem to the offline one. That is, assume that we are given an oracle for the offline problem that given a query matrix returns its leading eigenvector, and derive an online algorithm that is based on making queries to the oracle. Such a reduction is possible either via the *Follow the Perturbed Leader* meta-algorithm (FPL) [56] or by the recent *Online Frank-Wolfe* algorithm presented in [49]. Both of these algorithms require on each iteration of the online problem to make a single query to the eigenvector

[1] formally defined as $\{X \in \mathbb{R}^{n\times n} \mid X \succeq 0,\ \mathrm{Tr}(X) = 1\}$.

oracle which could be implemented using the Power or Lancsoz algorithms mentioned above with complexity of roughly $O(\text{nnz})$ arithmetic operations. While the per-iteration complexity of these methods is potentially much more favorable than the convexification approach and despite the fact that the regret bound guaranteed by FPL is optimal in terms of the length of the sequence, it comes with the price of a large dependence on the dimension. Indeed when designing online learning algorithms, usually the primary goal is optimal dependence on the sequence length, however favorable dependence on the dimension is crucial in order for the proposed method to be of any practical significance.

The Iterative Approach A third approach is to design online algorithms that directly tackle the non-convex optimization problem, i.e. online analogues of iterative algorithms for the offline problem such as the Power Method with an update step that roughly amounts to computing a single matrix-vector product, which is much more efficient than both previous approaches. Such an approach is reminiscent of the *Online Gradient Decent* method presented in [94] which is an online analogue of the gradient descent method for offline convex optimization. For the specific problem of Stochastic Principal Component Analysis, such algorithms with provable guarantees exist [9], [82], however we are not aware of any such method for the online setting considered here.

Our interest in this chapter is to study algorithms for the online eigenvector problem that may be of use for large scale instances. Towards this end we part from the convexification approach that was the main approach studied in previous related problems and requires super-linear computations, and focus on the oracle-based and iterative approaches which allow for more efficient implementations and may leverage sparsity in the data.

4.1 Results in this Chapter

Our main result is an online algorithm that takes the so called oracle approach and is based on the *Follow the Perturbed Leader* meta-algorithm. The algorithm requires on each iteration to perform only a single call to an offline eigenvector oracle and attains near-optimal regret in terms of the sequence length. In contrast to previous such algorithms, the dependence

Method	Regret	It. time
Matrix-MW [87, 5]	$\sqrt{T}$	n^3 (SVD)
FPL - entry-wise uniform noise (sec. 4.4.1) [56]	$n^{5/4}\sqrt{T}$	EV (dense)
FPL - spectral exp.-dist. noise (sec. 4.4.2)*	$n\sqrt{T}$	EV (dense)
FPL - gauss. noise ($\mathbf{Rank}(A_t) = 1$) [29]	$n^{1/4}\sqrt{T}$	EV (dense)
FPL - gauss. noise ($\mathbf{Rank}(A_t)$ unrestrict.) [59]*	$\sqrt{nT}$	EV (dense)
Online Frank-Wolfe [49]	$n^{1/2}T^{3/4}$	EV (dense)
This work, online (section 4.5)	$\sqrt{nT}$	EV (nnz)
This work, stochastic (section 4.3)	$\sqrt{T}$	nnz

Table 4.1: Comparison between algorithms in terms of the regret bound and run-time of a single iteration (dependence on log factors in n, T is omitted). EV denotes the computation of the leading eigenvector of a given matrix, where we write "(dense)" in case the EV computation is always carried out with a dense matrix ($\approx n^2$ non-zero entries), or write "(nnz)" when the EV computation is carried out in time that depends only on the number of non-zeros in the data observed so far. Results denoted with * refer to results that were not available when this work was published. The result in [29] applies only when all observed matrices are rank-one.

of the obtained regret bound on the dimension is much more favourable. Moreover, as opposed to previous oracle-based approaches, our algorithm admits an implementation that may leverage sparsity in the data to further reduce computation. On the technical side, our algorithm is based on a novel analysis of FPL. While previous approaches to analyzing the FPL algorithm are geometric in nature, which seems to inevitably introduce a large dependence on the dimension, our approach exploits the specific structure of the problem at hand and is algebraic in nature. More precisely, we study the spectrum of symmetric real matrices under Gaussian rank-one perturbations and apply tools from *matrix perturbation theory* to derive the regret bound.

We also consider a somewhat easier stochastic setting, in which we assume that the sequence of matrices is sampled from a fixed and unknown distribution. We present an algorithm that takes the so called iterative approach and is analogues to the Power algorithm for the offline setting, i.e. it computes a single matrix-vector product on each iteration. The regret of

the algorithm is nearly optimal in terms of the sequence length and depends on the dimension only through a logarithmic factor. The analysis of the algorithm is especially accessible and requires only a black-box application of the offline Power method and a, by now standard, matrix concentration inequality.

A comparison of our results to previous related work is detailed in Table 4.1. It can be seen that all previously-known eigenvector computation-based methods that apply to the full generality of the problem (observed matrices have arbitrary rank) have a worse dependence on the dimension n and perform eigenvector computations on inherently dense matrices, whereas our method applies eigenvector computations with run-time that depends on the sparsity of the observed data.

A performance measure that seems natural for comparing between online algorithms with different regret bounds and iteration complexity is the worst case overall time complexity to achieve ϵ average (expected) regret. This measure is important for instance when considering the application of online learning algorithms to saddle-point optimization problems [38, 19, 33]. The overall complexity required for the Matrix Mul. Weights method to achieve ϵ average regret is $\tilde{O}(n^3\epsilon^{-2})$. Our online algorithm on the other hand admits an implementation with overall running time $\tilde{O}(n^{3/2}\epsilon^{-7/2}\text{nnz})$, where nnz denotes the joint-sparsity of observed matrices (see subsection 4.5.1 for details). Hence in case $\epsilon^{-3/2}\text{nnz} = \tilde{O}(n^{3/2})$, it is overall faster.

Finally, we extend our results to handle non-symmetric matrices as well, and draw some interesting connections to semidefinite optimization and smoothing of spectral functions.

The rest of this chapter is organized as follows. In Section 4.2 we give notation, formally define problem of *online learning of eigenvectos*, and show how a generalization of the problem - online learning of singular-vectors from not-necessarily-square matrices, could be reduced to our original problem. In Section 4.3 we present our algorithm for the stochastic setting and analyze its performance. In Section 4.4 we overview the *follow the perturbed leader* (FPL) meta-algorithm for the online adversarial setting and present old and new guarantees for it for the online eigenvector problem. In Section 4.5 we present our main result - a new noise distribution for FPL and a novel regret bound and analysis that are based on a matrix perturbation theory approach. In section 4.6 we show how to extend these results to also

handle an adaptive adversary. In Section 4.7 we give applications of our main technique - eigengap inducing rank-one perturbations to semidefinite programming and smoothing of spectral functions. Finally, in Section 4.9 we give complete proofs of supporting lemmas and theorems used throughout the chapter.

4.2 Preliminaries

4.2.1 Notation

We denote by $\mathbb{S}_n$ the linear space of all $n \times n$ real symmetric matrices. We denote by $\mathcal{S}$ the Euclidean sphere in $\mathbb{R}^n$, that is $\mathcal{S} = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$. Given a matrix $A \in \mathbb{S}_n$ we denote its eigenvalues by $\lambda_1(A) \geq \lambda_2(A) \geq ...\lambda_n(A)$. We also refer to $\lambda_1(A)$ as $\lambda_{\max}$ which is given by $\lambda_{\max} = \max_{x \in \mathcal{S}} x^\top Ax$. We denote by $\delta(A)$ the *eigengap* of the matrix A which is given by $\delta(A) = \lambda_1(A) - \lambda_2(A)$. Unless specified else, given a vector $x \in \mathbb{R}^n$ we denote by $\|x\|$ its standard Euclidean norm and given a matrix $A \in \mathbb{S}_n$ we denote by $\|A\|$ its spectral norm. We also denote by $\|A\|_F$ and $\|A\|_*$ the Frobenius and nuclear norms of A respectively. Recall that $\|A\|, \|A\|_*$ are dual norms and thus according to Holder's inequality, it holds for any two matrices $A, B \in \mathbb{S}_n$ that $A \bullet B \leq \|A\| \cdot \|B\|_*$ where $\bullet$ denotes the standard inner product for matrices, that is $A \bullet B = \sum_{i,j} A_{i,j} \cdot B_{i,j}$.

4.2.2 Formal definition of the setting

In this chapter we consider the following repeated game: an adversary chooses a sequence of matrices $A_1, A_2, ..., A_T \in \mathbb{S}_n$. Then, for T rounds, the player is required on each round $t \in [T]$ to choose a vector $x_t \in \mathcal{S}$. After making his choice, the matrix A_t is revealed and the player gains the profit $x_t^\top A_t x_t$. Such an adversary is referred to in the literature as *oblivious* since he chooses the sequence of matrices without any knowledge of the actual actions of the player. A stronger type of adversary, known as *adaptive adversary*, need not commit in advance to the entire sequence of matrices, but may choose on time t the matrix A_t to depend on the entire history of the game, that is on $A_1, ..., A_{t-1}$ and $x_1, ..., x_{t-1}$. Throughout the chapter (especially in Sections 4.4, 4.5 and unless stated otherwise) we consider only

an oblivious adversary. In section 4.6 we detail how our results could be easily extended to also handle an adaptive adversary.

We measure the overall performance of the player according to the *regret* which is given by.

$$\begin{aligned}\text{regret}_T &= \max_{x\in\mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \sum_{t=1}^{T} x_t^\top A_t x_t \\ &= \lambda_{\max}\left(\sum_{t=1}^{T} A_t\right) - \sum_{t=1}^{T} x_t^\top A_t x_t.\end{aligned}$$

In case the decision maker uses randomization to choose is actions it also makes sense to consider the expected regret which is given by

$$\mathbb{E}[\text{regret}_T] = \max_{x\in\mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \mathbb{E}[\sum_{t=1}^{T} x_t^\top A_t x_t],$$

where the expectation is taken over the randomness introduced by the decision maker.

We assume without losing generality that $\|A_t\| \leq 1$ and that A_t is positive definite (note that adding a multiplicity of the identity matrix to A_t does not change the regret).

4.2.3 The asymmetric case

It makes sense to also consider the asymmetric case in which the input matrices $A_1, ..., A_T$ are not necessarily symmetric but are $m \times n$ real matrices for fixed m, n. In this case a prediction is a rank-one matrix $uv^\top$ where $u \in \mathbb{R}^m, v \in \mathbb{R}^n$ and both are unit vectors. In this case the regret is given by

$$\begin{aligned}\text{regret}_T &= \max_{\substack{u\in\mathbb{R}^m,\ \|u\|=1 \\ v\in\mathbb{R}^n,\ \|v\|=1}} \sum_{t=1}^{T} u^\top A_t v - \sum_{t=1}^{T} u_t^\top A_t v_t \\ &= \sigma_{\max}\left(\sum_{t=1}^{T} A_t\right) - \sum_{t=1}^{T} u_t^\top A_t v_t,\end{aligned}$$

where $\sigma_{\max}(\cdot)$ denotes the largest singular value.

We now show how given a low-regret algorithm for the symmetric problem we can use it to achieve low-regret on the asymmetric problem via a randomized conversion. The algorithm is given below. The following lemma

Algorithm 9 Asymmetric to Symmetric Conversion Algorithm

Input: Algorithm $\mathcal{A}$ for the online eigenvector problem in dimension $m \times n$.
for $t = 1, ..., T$ **do**
  Receive prediction $x_t \in \mathbb{R}^{m+n}$ from $\mathcal{A}$.
  Decompose x_t into $x_t = (\tilde{u}_t, \tilde{v}_t)$ for $\tilde{u}_t \in \mathbb{R}^m, \tilde{v}_t \in \mathbb{R}^n$.
  With probability $2\|\tilde{u}_t\|\|\tilde{v}_t\|$ set $(u_t, v_t) = \left(\frac{\tilde{u}_t}{\|\tilde{u}_t\|}, \frac{\tilde{v}_t}{\|\tilde{v}_t\|}\right)$. and with remaining probability set $(u_t, v_t) = (u, v)$, for uniformly chosen unit vectors $u \in \mathbb{R}^m, v \in \mathbb{R}^n$.
  Observe A_t.
  Feed the matrix $\tilde{A}_t = \begin{pmatrix} 0_{m\times m} & A_t \\ A_t^\top & 0_{n\times n} \end{pmatrix}$ to $\mathcal{A}$.
end for

bounds the regret of Algorithm 9 in terms of the regret of an algorithm for the symmetric problem [2]. The proof is given in the Section 4.9.

Lemma 31. *Assume that for all $t \in [T]$ it holds that $\|A_t\| \leq 1$. Then it holds for all $t \in [T]$ that $\tilde{A}_t$ is symmetric, $\|\tilde{A}_t\| \leq 1$ and*

$$\mathbb{E}[\sigma_{\max}\left(\sum_{t=1}^{T} A_t\right) - \sum_{t=1}^{T} u_t^\top A_t v_t] = \lambda_{\max}\left(\sum_{t=1}^{T} \tilde{A}_t\right) - \sum_{t=1}^{T} x_t^\top \tilde{A}_t x_t,$$

where the expectation is taken over the randomness in choosing u_t, v_t.

4.3 The Stochastic Setting

In this section we consider a stochastic setting which is somewhat easier than the online adversarial setting. In the stochastic setting we assume

[2] Note that the matrix $\tilde{A}_t$ defined in the algorithm is not positive definite as we assumed in the eigenvector problem, however this could be easily fixed by adding a multiplicity of the identity matrix and scaling accordingly to keep the unit upper bound on the spectral norm.

that all matrices $A_1, A_2, ..., A_T$ are sampled i.i.d. from a fixed but unknown distribution $\mathcal{D}$ over matrices in $\mathbb{S}_n$ with spectral norm bounded by one and w.l.o.g. we assume that these matrices are positive definite . We denote the distribution mean by $A = \mathbb{E}_{M\sim\mathcal{D}}[M]$.

Our algorithm titled *Epoch Power Method* for the stochastic setting is given below. It works by dividing the sequence into disjoint epochs, each of length ℓ which is a parameter that will be determined in the analysis. The algorithm predicts using a single unit vector throughout each epoch, while applying Power method update steps in order to compute the prediction for the following epoch.

We refer to an epoch by τ and denote by $x_{(\tau)}$ the point played throughout epoch τ. We also denote $\bar{A}_{\to(\tau)} = \frac{1}{\tau\ell}\sum_{t=1}^{\tau\ell} A_t$, that is the empirical mean of all matrices observed until the end of epoch τ.

In this section we prove the following theorem.

Theorem 13. *Let $x_1, x_2, ..., x_T$ denote the unit vectors played by Algorithm 10 throughout rounds $1, 2, ..., T$. The following guarantees hold.*

1. *Given $\delta > 0$, choosing block length $\ell = \lceil \frac{1}{4}\sqrt{T \log \frac{2nT}{\delta}} \rceil$ guarantees that with probability at least $1 - \delta$*

$$\sum_{t=1}^{T} \left(\lambda_{\max}(A) - x_t^\top A x_t\right) = O\left(\sqrt{T \log \frac{nT}{\delta}}\right).$$

2. *Choosing block length $\ell = \lceil \frac{1}{4}\sqrt{2T \log 2nT} \rceil$ guarantees that*

$$\mathbb{E}[\lambda_{\max}\left(\sum_{t=1}^{T} A_t\right) - \sum_{t=1}^{T} x_t^\top A_t x_t] = O\left(\sqrt{T \log nT}\right).$$

We note that both results in the theorem are optimal up to log factors.

In what follows we always refer to $x_{(\tau)}$ as the final value of this vector, that is, its value at the end of epoch $\tau - 1$.

In order to prove Theorem 13 we need the following two lemmas.

The following lemma is a straightforward application of a Bernstein concentration inequality for symmetric matrices (see [86], Theorem 1.4).

Algorithm 10 Epoch Power Method

```
Input: block length parameter ℓ.
Let v be a unit vector chosen uniformly at random from the sphere.
for t = 1, ..., ℓ do
    Play arbitrarily & observe A_t.
end for
x_(2) ← v.
for τ = 2...⌈T/ℓ⌉ do
    x_(τ+1) ← v.
    Let Ā_→(τ−1) = 1/((τ−1)ℓ) Σ_{t=1}^{(τ−1)ℓ} A_t.
    for t = (τ − 1)ℓ + 1... min{τℓ, T} do
        Play x_(τ) & observe A_t.
        x_(τ+1) ← Ā_→(τ−1) x_(τ+1) / ‖Ā_→(τ−1) x_(τ+1)‖.
    end for
end for
```

Lemma 32. *For any block τ and $\epsilon > 0$ it holds that*

$$\Pr\left(\|A - \bar{A}_{\to(\tau)}\| \geq \epsilon\right) \leq n \exp\left(\frac{-\epsilon^2 \min\{\tau\ell, T\}}{16}\right).$$

The following lemma is based on an analysis of the Power Method for computing the leading eigenvector of a positive definite matrix and gives a guarantee on the quality of the vectors $x_{(\tau)}$. For details see Theorem 4.1 in [60].

Lemma 33. *For any $\lfloor \frac{T}{\ell} \rfloor \geq \tau \geq 2$ and $\epsilon > 0$, it holds with probability at least $1 - n\exp(-\ell\epsilon)$ that*

$$x_{(\tau+1)}^{\top} \bar{A}_{\to(\tau-1)} x_{(\tau+1)} \geq (1-\epsilon)\lambda_{\max}(\bar{A}_{\to(\tau-1)}).$$

We can now prove Theorem 13.

Proof. We first prove part 1 of the theorem, part 2 follows as a corollary.

Fix a block number $\tau \geq 3$ and an error tolerance ϵ_τ. Using Lemmas 32, 33 we have that with probability at least $1 - n\exp(-\ell\epsilon_\tau) - n\exp\left(\frac{\epsilon_\tau^2(\tau-2)\ell}{16}\right)$

that

$$\begin{aligned} x_{(\tau)}Ax_{(\tau)} &\geq x_{(\tau)}\bar{A}_{\to(\tau-2)}x_{(\tau)} - \epsilon_\tau \\ &\geq \lambda_{\max}(\bar{A}_{\to(\tau-2)}) - 2\epsilon_\tau \\ &\geq \lambda_{\max}(A) - 3\epsilon_\tau, \end{aligned}$$

where the first and last inequalities follow from Lemma 32 and the second inequity follows from Lemma 33 and the observation that $\lambda_{\max}(\bar{A}_{\to(\tau-2)}) \leq 1$.

Setting $\epsilon_\tau = 4\sqrt{\frac{\log \frac{2nT}{\delta}}{(\tau-2)\ell}}$ we have that with probability at least

$$\begin{aligned} &1 - n\exp\left(-4\sqrt{\frac{\ell \log \frac{2nT}{\delta}}{(\tau-2)}}\right) - \frac{\delta}{2T} \\ &> 1 - n\exp\left(-4\sqrt{\frac{\ell^2 \log \frac{2nT}{\delta}}{T}}\right) - \frac{\delta}{2T} \end{aligned}$$

it holds that

$$x_{(\tau)}Ax_{(\tau)} \geq \lambda_{\max}(A) - 12\sqrt{\frac{\log \frac{2nT}{\delta}}{(\tau-2)\ell}}.$$

Thus setting the block length to $\ell = \lceil \frac{1}{4}\sqrt{T \log \frac{2nT}{\delta}} \rceil$ we have that with probability at least $1 - \frac{\delta}{T}$ it holds that

$$x_{(\tau)}Ax_{(\tau)} \geq \lambda_{\max}(A) - 24\sqrt{\frac{\sqrt{\log \frac{2nT}{\delta}}}{(\tau-2)\sqrt{T}}}.$$

Summing over $\tau \geq 3$ we have that with probability at least $1-\delta$,

$$\begin{aligned}
\sum_{\tau=3}^{\lceil T/\ell \rceil} \left(x_{(\tau)}^\top A x_{(\tau)} - \lambda_{\max}(A) \right) &\leq 24 \left(\frac{\log \frac{2nT}{\delta}}{T} \right)^{1/4} \sum_{\tau=3}^{\lceil T/\ell \rceil} \frac{1}{\sqrt{\tau-2}} \\
&< 24 \left(\frac{\log \frac{2nT}{\delta}}{T} \right)^{1/4} \int_1^{T/\ell} \frac{1}{\sqrt{\tau-1}} d\tau \\
&< 48 \left(\frac{\log \frac{2nT}{\delta}}{T} \right)^{1/4} \sqrt{\frac{T}{\ell}} \\
&= 48 \left(\frac{T \log \frac{2nT}{\delta}}{\ell^2} \right)^{1/4} = 24.
\end{aligned}$$

Since each block accounts for ℓ iterations and in worst case the algorithm suffers loss of 1 on all first 2ℓ iterations we have that with probability at least $1-\delta$

$$\sum_{t=1}^{T} \left(\lambda_{\max}(A) - x_t^\top A x_t \right) \leq 26\ell = O\left(\sqrt{T \log \frac{2nT}{\delta}} \right). \tag{4.1}$$

We now prove part 2 of the theorem.

Since on any time t, x_t is independent of A_t we have that

$$\begin{aligned}
&\mathbb{E}[\max_{x \in \mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \sum_{t=1}^{T} x_t^\top A_t x_t] \leq \\
&\mathbb{E}[\max_{x \in \mathcal{S}} x^\top (T \cdot A) x + \|\sum_{t=1}^{T} (A_t - A)\| - \sum_{t=1}^{T} x_t^\top A x_t] = \\
&\mathbb{E}[T \cdot \lambda_{\max}(A) - \sum_{t=1}^{T} x_t^\top A x_t + \|\sum_{t=1}^{T} (A_t - A)\|]. \qquad (4.2)
\end{aligned}$$

Let us denote by τ_{end} the index of the last block. Note that $\|\sum_{t=1}^{T} A_t - A\| = T \cdot \|\bar{A}_{\to(\tau_{end})} - A\|$. By applying Lemma 32 we have that with probability at least $1-\delta$ it holds that

$$\frac{1}{T} \|\sum_{t=1}^{T} A_t - A\| = \|\bar{A}_{\to(\tau_{end})} - A\| \leq 4\sqrt{\frac{\ln \frac{n}{\delta}}{T}}. \tag{4.3}$$

Plugging Eq. (4.1) and Eq. (4.3) into Eq. (4.2) we have that

$$\mathbb{E}[\max_{x\in\mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \sum_{t=1}^{T} x_t^\top A_t x_t] \leq$$
$$(1-2\delta)\cdot(4+13/2)\sqrt{T\log\frac{2nT}{\delta}} + 2\delta\cdot 2T.$$

Thus setting $\delta = T^{-1}$, which corresponds to setting the block length to $\ell = \lceil \frac{1}{4}\sqrt{2T\log 2nT}\rceil$, gives the second part of the theorem. □

4.4 The FPL Meta-algorithm for the Online Setting

In this section we overview the Follow the Perturbed Leader meta-algorithm for online learning and its application to the problem of online learning of eigenvectors. The meta-algorithm is given below. The algorithm relies on the availability of an oracle for the offline eigenvector problem, that is an oracle that given a matrix A, returns a leading eigenvector of A. We denote a call to this oracle by $\mathbf{EV}(A)$. Additionally, the algorithm relies on the availability of a distribution $\mathcal{D}$ over $\mathbb{S}_n$ from which it is possible to sample a perturbation matrix (efficiently). Different such distributions give rise to different instances of the algorithm with different regret guarantees.

Algorithm 11 Follow the Perturbed Leader

```
Input: distribution D over S_n.
Sample a matrix N ~ D.
x_1 ← EV(N).
for t = 1, 2, ... do
    Play x_t & observe A_t.
    x_{t+1} ← EV(Σ_{τ=1}^{t} A_τ + N).
end for
```

Theorem 14. *The expected regret of algorithm 11 is upper bounded as*

follows.

$$\mathbb{E}[regret_T(FPL(\mathcal{D}))] \leq \sum_{t=1}^{T} \mathbb{E}_{N\sim\mathcal{D}}[x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t] + \mathbb{E}_{N\sim\mathcal{D}}[x_1^\top N x_1 - x^{*\top} N x^*]$$

where $x^* = \arg\max_{x\in\mathcal{S}} x^\top \left(\sum_{t=1}^{T} A_t\right) x$.

The proof is given in Section 4.9.

We now turn to survey two choices for the perturbation-generating distribution $\mathcal{D}$ and their corresponding regret bounds. We show that even though these distributions give rise to optimal algorithms in terms of the sequence length T, they suffer from a large dependence on the problem's dimension n.

4.4.1 FPL with entry-wise uniform perturbation

Following [56] we consider the following entry-wise uniform distribution $\mathcal{D}_{uni}$. Each coordinate $i \geq j$ in the perturbation matrix N is sampled $\mathrm{U}[0, 1/\epsilon]$ and for each coordinate $i < j$ we set $N_{i,j} \leftarrow N_{j,i}$ (in order for the resulting perturbation to be symmetric).

In [56] it was shown that with such noise distribution, one can bound the expected regret of a single round t as follows (see proof of Theorem 1.1. in [56]).

$$\mathbb{E}_{N\sim\mathcal{D}_{uni}}[x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t] \leq \epsilon \|A_t\|_1,$$

where $\|A\|_1 = \sum_{i,j} |A_{i,j}|$.

Furthermore, since the sampled perturbation is bounded in ℓ_∞, using Holder's inequality we have that

$$\begin{aligned}\mathbb{E}_{N\sim\mathcal{D}_{uni}}[x_1^\top N x_1 - x^{*\top} N x^*] &\leq \mathbb{E}_{N\sim\mathcal{D}_{uni}}[\|x_1 x_1^\top - x^* x^{*\top}\|_1 \cdot \|N\|_\infty] \\ &\leq \frac{D_1}{\epsilon},\end{aligned}$$

where D_1 denotes the ℓ_1 diameter of the set $\{xx^\top \mid x \in \mathcal{S}\}$.

In order to derive the precise regret bound we need to bound both quantities $\|A_t\|_1, D_1$. The proof of the following lemma is given Section 4.9.

Lemma 34. *It holds that $D_1 = O(n)$ and for all t it holds that $\|A_t\|_1 = O(n^{3/2})$. These bounds are also tight.*

Plugging the result of the lemma into Theorem 14 and optimizing over ϵ we have that

$$\mathbb{E}[\text{regret}_T(\text{FPL}(\mathcal{D}_{uni}))] = O\left(n^{5/4}\sqrt{T}\right). \tag{4.4}$$

4.4.2 FPL with exponentially-distributed perturbation

A second distribution that we consider, $\mathcal{D}_{exp}$, samples from $\mathbb{S}_n$ according to the following density.

$$d\mu(N) \propto \exp(-\epsilon\|N\|), \tag{4.5}$$

for appropriately chosen parameter ϵ.

A similar distribution was used for the problem of learning rotations [51] (altough in [51] the density was proportional to the nuclear norm and not the spectral as in Eq. (4.5)). For more details on how to sample from the distribution specified by Eq. (4.5) as well as a proof of the following lemma, the reader is referred to [50].

Lemma 35. *It holds that $\|N\| \sim$ Gamma$(n^2, 1/\epsilon)$ and in particular $\mathbb{E}[\|N\|] = \frac{n^2}{\epsilon}$.*

The following lemma upper bounds the expected regret on a single round t.

Lemma 36. *On any time $t \in [T]$ it holds that*

$$\mathbb{E}_{N\sim\mathcal{D}_{exp}}[(x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t^\top] \leq e\epsilon.$$

The proof is given in Section 4.9.

Plugging Lemmas 35, 36 into Theorem 14 and optimizing over ϵ we have that

$$\mathbb{E}[\text{regret}_T(\text{FPL}(\mathcal{D}_{exp}))] = O\left(n\sqrt{T}\right). \tag{4.6}$$

4.5 New Perturbation and Analysis for FPL via Matrix Perturbation Theory

In this section we present our main result - a new noise distribution for the FPL algorithm and a corresponding analysis. In contrast to the analysis used in order to derive the regret bounds in Subsections 4.4.1, 4.4.2, which relies on geometric considerations and for which a large dependence of the regret bound on the problem's dimension n seems unavoidable, here we use a new analysis idea that is algebraic in nature and relies on tools from matrix perturbation theory which results in a much more moderate dependence on the dimension.

The new distribution, denoted $\mathcal{D}_{new}$, is based on a single parameter c and sampling from it is done as follows. We draw a vector $v \in \mathbb{R}^n$ whose entries are i.i.d. $\mathcal{N}(0,1)$ random variables and set the perturbation matrix to $N = c \cdot vv^\top$.

We prove the following theorem.

Theorem 15. *Let* $c = \sqrt{\frac{T}{n}\max\{1, \ln(T/n)\}}$. *Then*

$$\mathbb{E}[\mathit{regret}_T(\mathit{FPL}(\mathcal{D}_{new}))] = O(\sqrt{nT\max\{1, \ln(T/n)\}}).$$

Aside from the important improvement in the dependence on the dimension ($\sqrt{n}$ in Theorem 15 vs. at least n in Section 4.4), a key difference between the perturbations is in the efficiency of the resulting implementations. The key feature of the FPL algorithm for the online eigenvector problem is that the $\mathbf{EV}(\cdot)$ oracle could be implemented using iterative methods such as the Power or Lanczos methods, that only require to compute matrix-vector products, to run in time that is typically $O(\text{nnz})$ where nnz denotes the sparsity of the input matrix. However, since the perturbations considered in Subsections 4.4.1. 4.4.2 are dense with high probability, even in case the support of all matrices A_t is sparse, the call to the oracle $\mathbf{EV}$ in Algorithm 11 will be with a dense matrix In contrast, the perturbation considered in this section is rank-one. Hence, while the perturbation matrix $N = cvv^\top$ is still dense with high probability, computing the product of N with a vector requires only $O(n)$ time which allows for an oracle implementation that could still benefit computationally from sparsity in the data.

We now turn to prove Theorem 15.

The following classic result in matrix perturbation theory is known as the Davis-Kahan *Sine Theroem*, see [22]. For ease of presentation, we restate the theorem and give a self-contained proof in Section 4.9.

Theorem 16 (Davis Kahan sine theorem)**.** *Let $A, B, E \in \mathbb{S}_n$ such that $B = A + E$ and assume that $\delta(A) > 0$. Denote by u_A the top eigenvector of A and by u_B the top eigenvector of B. It holds that*

$$\|u_A u_A^\top - u_B u_B^\top\|_F \leq 2\sqrt{2}\frac{\|E\|}{\delta(A)}.$$

The following theorem constitutes the key technical ingredient in the proof of Theorem 15. It upper bounds the cumulative distribution function of the eigengap of the perturbed matrix $A + cvv^\top$ for a given matrix A.

Theorem 17. *Let $A \in \mathbb{S}_n$ and let v be a vector of independent $\mathcal{N}(0,1)$ random variables and let c be a positive scalar. Denote $B = A + cvv^\top$. Then for any $\epsilon > 0$*

$$\Pr(\delta(B) \leq \epsilon) \leq \min\{\frac{2\sqrt{2}\epsilon}{\pi c}, 1\}.$$

The proof is based on anti-concentration results for the leading eigenvalue of the perturbed matrix. Due to its length and technical detail it is deferred to Section 4.9. Here for some intuition, we prove a weaker version of Theorem 17, which captures some of the key ideas.

Lemma 37. *[Weaker version of Theorem 17] Let $A \in \mathbb{S}_n$ and let v be a vector of independent $\mathcal{N}(0,1)$ random variables and let c be a positive scalar. Denote $B = A + cvv^\top$. Then for any $\epsilon > 0$*

$$\Pr(\delta(B) \leq \epsilon) \leq \sqrt{\frac{2\epsilon}{\pi c}}.$$

Proof. Weyl's eigenvalues inequality states that for any two matrices $X, Y \in \mathbb{S}_n$ with eigenvalues $\lambda_1(X) \geq \lambda_2(X) \geq ... \geq \lambda_n(X)$ and $\lambda_1(Y) \geq \lambda_2(Y) \geq ... \geq \lambda_n(Y)$ it holds for any $i \in [n]$ that

$$\lambda_i(X) + \lambda_n(Y) \leq \lambda_i(X + Y) \leq \lambda_i(X) + \lambda_1(Y). \tag{4.7}$$

Applying Eq. (4.7) with $X = cvv^\top$, $Y = A$ and $i = 2$ gives us that

$$\lambda_2(B) \leq \lambda_2(cvv^\top) + \lambda_1(A) = \lambda_1(A),$$

where the equality follows since $vv^\top$ is rank-one.

Thus we have that

$$\delta(B) = \lambda_1(B) - \lambda_2(B) \geq \lambda_1(A + cvv^\top) - \lambda_1(A).$$

Let us denote by u_1 the eigenvector of A that corresponds to eigenvalue $\lambda_1(A)$. We continue to lower bound $\delta(B)$ as follows.

$$\delta(B) \geq u_1^\top \left(A + cvv^\top\right) u_1 - \lambda_1(A) = c(u_1^\top v)^2.$$

Since v is a vector of independent $\mathcal{N}(0,1)$ random variables we have that $(u_1^\top v)$ is also distributed as a $\mathcal{N}(0,1)$ random variable. Thus $(u_1^\top v)^2$ is a Chi-squared random variable with a single degree of freedom, also denoted as χ_1^2. If R is a χ_1^2 random variable then it holds that for any $\epsilon > 0$

$$\Pr(R \leq \epsilon) = \int_0^\epsilon \frac{e^{-t/2}}{\sqrt{2\pi t}} dt \leq \frac{1}{\sqrt{2\pi}} \int_0^\epsilon \frac{1}{\sqrt{t}} dt = \sqrt{\frac{2}{\pi}\epsilon}.$$

Thus we have that for any $\epsilon > 0$ it holds that

$$\Pr(\delta(B) \leq \epsilon) \leq \Pr\left((u_1^\top v)^2 \leq \frac{\epsilon}{c}\right) \leq \sqrt{\frac{2\epsilon}{\pi c}}.$$

□

The following lemma is a consequence of Theorem 17. The proof is given in Section 4.9.

Lemma 38. *Given $A \in \mathbb{S}_n$ let v be a random vector whose entries are independent $\mathcal{N}(0,1)$ random variables and let c be a positive scalar. Denote $\delta = \delta(A + cvv^\top)$. Define the random variable $X = \min(a, \delta^{-1})$ where a is a given positive constant. Then we have $\mathbb{E}[X] \leq \frac{2\sqrt{2}}{\pi c} \max\{\ln(\frac{\pi e a c}{2\sqrt{2}}), 1\}$.*

We are now ready to prove Theorem 15.

Proof. Starting from Theorem 14 we have that

$$\begin{aligned}
\mathbb{E}[\text{regret}_T(\text{FPL}(\mathcal{D}_{new}))] &\leq \sum_{t=1}^{T} \mathbb{E}_{N\sim\mathcal{D}_{new}}[x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t] \\
&\quad + \mathbb{E}[x_1^\top N x_1 - x^{*\top} N x^*] \\
&\leq \sum_{t=1}^{T} \mathbb{E}[\|x_{t+1}x_{t+1}^\top - x_t x_t^\top\|_* \|A_t\|] + \mathbb{E}[\|N\|] \\
&\leq \sqrt{2}\sum_{t=1}^{T} \mathbb{E}[\|x_{t+1}x_{t+1}^\top - x_t x_t^\top\|_F] + c\mathbb{E}[\|v\|^2],
\end{aligned}$$

where the second inequality follows from Holder's inequality and the third follows from the fact that for any matrix in $\mathbb{S}_n$ with k non-zero eigenvalues $\lambda_1, \lambda_2, ..., \lambda_k$ it holds that $\|A\|_F^2 = \sum_{i=1}^k \lambda_i^2 \geq \frac{1}{k}\left(\sum_{i=1}^k |\lambda_i|\right)^2 = \frac{1}{k}\|A\|_*^2$, and from our choice of the noise matrix N.

Denote $S_t = \sum_{\tau=1}^{t-1} A_\tau$. Since for any t, x_t is the leading eigenvector of the matrix $(S_t + N)$ and x_{t+1} is the leading eigenvector of the matrix $(S_{t+1} + N) = (S_t + N) + A_t$, we have by applying Theorem 16 that

$$\begin{aligned}
\mathbb{E}[\|x_{t+1}x_{t+1}^\top - x_t x_t^\top\|_F] &\leq \mathbb{E}[\min\{\sqrt{2}, 2\sqrt{2}\frac{\|A_t\|}{\delta(S_t+N)}\}] \\
&\leq 2\sqrt{2}\mathbb{E}[\min\{\frac{1}{2}, \frac{1}{\delta(S_t+N)}\}],
\end{aligned}$$

where the min term is used since obviously $\|x_t x_t^\top - x_{t+1}x_{t+1}^\top\|$ is upper bounded by $\sqrt{2}$. The second inequality follows since $\|A_t\| \leq 1$.

Since all entries of v are $\mathcal{N}(0,1)$ random variables it holds that $\mathbb{E}[\|v\|^2] = n$ and thus

$$\mathbb{E}[\text{regret}_T(\text{FPL}(\mathcal{D}_{new}))] \leq \sqrt{2}\sum_{t=1}^{T} \mathbb{E}[\min\{\frac{1}{2}, \frac{1}{\delta(S_t + cvv^\top)}\}] + cn.$$

Applying the result of Lemma 38 with $a = \frac{1}{2}$ for every t gives

$$\mathbb{E}[\text{regret}_T(\text{FPL}(\mathcal{D}_{new}))] \leq 2\sqrt{2}T \cdot \frac{2\sqrt{2}}{\pi c}\max\{\ln(\frac{\pi e c}{4\sqrt{2}}), 1\} + cn.$$

Thus setting $c = \sqrt{\frac{T}{n}\max\{\ln(T/n), 1\}}$ yields the theorem. $\square$

4.5.1 Using approximate eigenvector computations

So far we have assumed that the eigenvector oracle used in Algorithm 11 finds an exact leading eigenvector. In practice, it is much more efficient to use iterative methods such as the Power and Lanczos algorithms to find an approximate eigenvector. The following theorem states that indeed Algorithm 11 admits such an efficient implementation, without sacrificing the regret guarantee given in Theorem 15. The proof is given in Section 4.9.

Theorem 18. *Algorithm 11, instantiated with noise distribution $\mathcal{D}_{new}$, admits an implementation such that the statement of Theorem 15 holds, and the worst-case time complexity of each iteration is $\tilde{O}(n^{-1/4}T^{3/4}\min\{nnz, n^2\})$, where nnz denotes the joint-sparsity of all observed matrices $A_1, ..., A_T$.*

4.6 Extending the Result of Section 4.5 to Adaptive Adversaries

We now turn to consider a slightly more challenging online setting than considered so far, in which the adversary is not *oblivious* but *adaptive*. That is the sequence of matrices $\{A_t\}_{t=1}^T$ is no longer chosen in advanced and remains fixed throughout the game, instead, on iteration t, the matrix A_t may depend on the the choices of the algorithm on iterations $1, 2, ...t-1$ (that is on the vectors $x_1, ..., x_{t-1}$), but not on any fresh randomness possibly used by the algorithm on time t.

Unfortunately Theorem 15 does not directly apply to handle an adaptive adversary since in the analysis we have used the fact that on iteration t, the matrix A_t does not depend on the random perturbation N. Luckily, we now show that it is not hard to modify Algorithm 11 and the analysis of Theorem 15 to also handle an adaptive adversary.

Towards this end, we now consider a slight modification of Algorithm 11 in which on each iteration t we sample a new perturbation matrix $N_t = cv_tv_t^\top$ from the distribution $\mathcal{D}_{new}$, which is chosen independently of previous perturbations. That is, now on iteration t the algorithm predicts according

to

$$\tilde{x}_t \leftarrow \mathbf{EV}\left(\sum_{\tau=1}^{t-1} A_\tau + N_t\right), \qquad N_t \sim \mathcal{D}_{new}. \tag{4.8}$$

The following theorem bounds the regret of the modified algorithm.

Theorem 19. *With probability at least* $1-\delta$ *it holds that*

$$\max_{x\in\mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \sum_{t=1}^{T} \tilde{x}_t^\top A_t \tilde{x}_t = O\left(\sqrt{nT\max\{1, \ln(T/n)\}} + \sqrt{T\ln\frac{1}{\delta}}\right).$$

The proof of the theorem follows a known analysis that can be found for instance in [15]. For the sake completeness we provide one with full detail here.

Proof. Consider the sequence of vectors $\tilde{x}_1, ..., \tilde{x}_T$ generated according to Eq. (4.8) with respect to a sequence of matrices $A_1, ..., A_T$, chosen by an adaptive adversary. Consider also a second sequence of unit vectors $x_1, ..., x_T$ generated by Algorithm 11, such that only a single sample N is drawn from the distribution $\mathcal{D}_{new}$, when applied to the sequence of matrices $A_1, ..., A_T$. An important observation is that the points $x_1, ..., x_T$ are chosen as if the input sequence $A_1, ..., A_T$ was chosen by an oblivious adversary and in particular, the single perturbation N used to generate $x_1, ..., x_T$ is independent of the matrices$A_1, ..., A_T, N_1, ..., N_T$, where $N_1, ..., N_T$ are the perturbations used to generate the vectors $\tilde{x}_1, ..., \tilde{x}_T$ according to Eq. (4.8).

By applying Theorem 15 with respect to the sequence of vectors $x_1, ..., x_T$ we have that

$$\max_{x\in\mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \mathbb{E}_N[\sum_{t=1}^{T} x_t^\top A_t x_t] = O(\sqrt{nT\max\{1, \ln(T/n)\}}). \tag{4.9}$$

Further note that for any iteration t, the random matrices N, N_t are identically distributed and independent of $N_1, ..., N_{t-1}$, and thus it holds

that

$$\mathbb{E}_{N_t}[\tilde{x}_t^\top A_t \tilde{x}_t] = \mathbb{E}_N[x_t^\top A_t x_t]. \tag{4.10}$$

Let us define for any iteration t the random variable $Z_t = \tilde{x}_t^\top A_t \tilde{x}_t - \mathbb{E}[\tilde{x}_t^\top A_t \tilde{x}_t \,|\, N_1, ..., N_{t-1}]$. Note that $\mathbb{E}[Z_t \,|\, N_1, ..., N_{t-1}] = 0$. Thus we have that

$$\mathbb{E}[\sum_{\tau=1}^{t} Z_\tau \,|\, N_1, ..., N_{t-1}] = \mathbb{E}[\sum_{\tau=1}^{t-1} Z_\tau \,|N_1, ..., N_{t-1}] = \sum_{\tau=1}^{t-1} Z_\tau,$$

and thus the sequence $S_t := \sum_{\tau=1}^{t} Z_\tau$ for $t \in [T]$ forms a martingale sequence with respect to the random variables $N_1, ..., N_T$. This martingale also satisfies the bounded difference property, since for any t,

$$|S_t - S_{t-1}| = |Z_t| = |\tilde{x}_t^\top A_t \tilde{x}_t - \mathbb{E}[\tilde{x}_t^\top A_t \tilde{x}_t \,|\, N_1, ..., N_{t-1}]| \leq 1,$$

where the last inequality follows from our assumption that $A_t \succ 0$ and $\|A_t\| \leq 1$.

Thus by applying Azuma's concentration inequality for martingales, we have that for any $C > 0$

$$\Pr(\sum_{t=1}^{T} \mathbb{E}[\tilde{x}_t^\top A_t \tilde{x}_t \,|\, N_1, ..., N_{t-1}] - \tilde{x}_t^\top A_t \tilde{x}_t \geq C) \leq e^{-\frac{C^2}{2T}}.$$

By applying Eq. (4.10) for all t we have that

$$\Pr(\sum_{t=1}^{T} \mathbb{E}_N[x_t^\top A_t x_t] - \tilde{x}_t^\top A_t \tilde{x}_t \geq C) =$$
$$\Pr(\mathbb{E}_N[\sum_{t=1}^{T} x_t^\top A_t x_t] - \sum_{t=1}^{T} \tilde{x}_t^\top A_t \tilde{x}_t \geq C) \leq 2e^{-\frac{C^2}{2T}},$$

and thus by Eq. (4.9) we have that

$$\Pr(\max_{x \in \mathcal{S}} \sum_{t=1}^{T} x^\top A_t x - \sum_{t=1}^{T} x^\top A_t x \geq C + R_T) \leq 2e^{-\frac{C^2}{2T}}$$

where $R_T = O(\sqrt{nT \max\{1, \ln(T/n)\}})$.

The theorem follows from setting $C = \sqrt{2T \ln \frac{1}{\delta}}$. $\square$

4.7 Applications to Semidefinite Programming and Smoothing of Spectral Functions

In this section we briefly detail strong connections between the results presented in Section 4.5 and semidefinite optimization. In particular we show that our online algorithm, described in Section 4.5, could be applied to the problem of Semidefinite Programming (SDP), and that our rank-one perturbation and the corresponding Theorem 17 could be applied to *smooth* the largest eigenvalue function which shows up in the dual problem to SDP.

Towards this end, let $M_1, ..., M_m$ be matrices in $\mathbb{S}_n$ and $b_1, ..., b_m$ be scalars in $\mathbb{R}$. Consider the following optimization problem:

$$\max_{X \succeq 0, \mathrm{Tr}(X)=1} \min_{i \in [m]} M_i \bullet X - b_i, \tag{4.11}$$

where $X \succeq 0$ means that for any vector $v \in \mathbb{R}^n$ it holds that $v^\top X v \geq 0$, and $\mathrm{Tr}(\cdot)$ is the trace function.

Clearly, Problem (4.11) captures the problem of finding a matrix $X \succeq 0$ with unit trace that satisfies all constraints $\{M_i \bullet Y \geq b_i\}_{i=1}^m$. Note also that, by standard reductions, w.l.o.g. the constraint $\mathrm{Tr}(X) = 1$ could be replaced with an arbitrary upper bound on the trace.

The dual of Problem (4.11) is given by the following optimization problem (see [37] for more details):

$$\min_{p \in \Delta_m} \lambda_{\max} \left(\sum_{i=1}^m p_i \left(M_i - b_i I \right) \right), \tag{4.12}$$

where $\lambda_{\max}$ is the largest (signed) eigenvalue function, i.e. $\lambda_{\max}(A) := \max_{x \in \mathcal{S}} x^\top A x$, and Δ_m denotes the probabilistic simplex in $\mathbb{R}^m$, i.e. $\Delta_m = \{x \in \mathbb{R}^m \,|\, \forall i \in [m] : x_i \geq 0, \sum_{i=1}^m x_i = 1\}$. The duality between Problems (4.11) and (4.12) is strong, in the sense that the optimum of (4.11) equals the optimum of (4.12).

4.7.1 Semidefinite programming via online learning

There is a well-known generic approach to apply online-learning algorithms to solve offline $\max - \min$ concave-convex optimization problems such as Problem (4.11). Relevant examples include [44, 3, 37]. This framework, when applied with the FPL algorithm for the online eigenvector problem (Algorithm (11)), leads to Algorithm 12, given below. In short, the algorithm applies two online-algorithms for solving Problem 4.11. The *primal* algorithm tries to maximize $\mathcal{L}(X,p) := \sum_{i=1}^{m} p_i(M_i \bullet X - b_i)$ as a function of X (subject to the constraints on X detailed above) whereas, the *dual* algorithm tries to minimize $\mathcal{L}(X,p)$ as a function of p, which is a distribution over the m constraints. Formally, this is done by applying the primal algorithm to the sequence of linear gain functions $\mathcal{L}_1^{(p)}(X), \mathcal{L}_2^{(p)}(X), ..., \mathcal{L}_T^{(p)}(X)$, where $\mathcal{L}_t^{(p)}(X) := \sum_{i=1}^{m} p_t(i)(M_i - b_i I) \bullet X$ and p_t is the play of the dual algorithm on iteration t, and the dual algorithm to the sequence of linear loss functions $\mathcal{L}_1^{(d)}(p), ..., \mathcal{L}_T^{(d)}(p)$, where $\mathcal{L}_t^{(d)} := \sum_{i=1}^{m} p_i(M_i \bullet X_t - b_i)$ and X_t is the play of the primal algorithm on iteration t. Note that this formulation requires that both algorithms can handle an *adaptive adversary* (as discussed in Section 4.6), and that the primal algorithm produces plays in the set $\{X \in \mathbb{R}^{n\times n} \,|\, X \succeq 0, \operatorname{Tr}(X) = 1\}$, which is in accordance with the algorithms studied in this chapter. The dual algorithm is implemented via the well known *multiplicative weights method* (MW) (see [4] and also Definition 1 and Lemma 1 in [37]).

Theorem 20. *Suppose that for all $i \in [m]$ it holds that $\|M_i\| \leq 1$ and $|b_i| \leq 1$. For $T = \tilde{O}(n\epsilon^{-2})$, there exists a choice of η_p, η_d such that Algorithm 12 produces with probability at least $1 - 1/n$ a solution $\bar{X}$ which satisfies*

$$\min_{i\in[m]} (M_i - b_i I) \bullet \bar{X} \geq OPT - \epsilon,$$

where OPT it the optimal value of Problem (4.11). The algorithm runs in total time $\tilde{O}\left(\frac{n}{\epsilon^2}\left(nnz + \frac{\sqrt{n}}{\epsilon^{3/2}} \min\{nnz, n^2\}\right)\right)$, where nnz denotes the overall number of non-zeros in the matrices $M_1, ..., M_m$.

Proof. According to Theorem 19 we have that there exists a choice for η_p

Algorithm 12 Primal-Dual SDP Solver via Online Learning

Input: T, η_p, η_d
$w_1 \leftarrow \mathbf{1}_m$
Let x_1 be a unit vector chosen uniformly at random
for $t = 1$ to T **do**
  $p_t(i) \leftarrow w_t(i)/\|w_t\|_1$
  $v_t \sim \mathcal{N}(0_{n\times n}, I)$
  $x_{t+1} \leftarrow \mathbf{EV}\left(\sum_{\tau=1}^{t}\sum_{i=1}^{m} p_\tau(i)(M_i - b_i I) + \eta_p v_t v_t^\top\right)$ {**EV** returns the largest eigenvector; $x_{t+1}x_{t+1}^\top$ corresponds to the matrix X_{t+1}}
  for $i = 1$ to m **do**
    $w_{t+1}(i) \leftarrow w_t(i)\left(1 - \eta_d(x_{t+1}^\top M_i x_{t+1} - b_i)\right)$ {multiplicative weights update rule}
  end for
end for
return $\bar{X} = \frac{1}{T}\sum_{t=1}^{T} x_t x_t^\top$

such that with probability at least $1 - 1/n$ it holds that

$$\begin{aligned}
\tilde{O}(\sqrt{nT}) &= \text{regret}_T(FPL) = \max_{x\in\mathcal{S}} x^\top \sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)(M_i - b_i I)x \\
&\quad - \sum_{t=1}^{T} x_t^\top \sum_{i=1}^{m} p_t(i)(M_i - b_i I)x_t \\
&\geq T\cdot OPT - \sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)(x_t^\top M_i x_t - b_i), \qquad (4.13)
\end{aligned}$$

where the inequality follows since

$$\begin{aligned}
&\max_{x\in\mathcal{S}} x^\top \sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)(M_i - b_i I)x = \max_{X\succeq 0,\, \mathrm{Tr}(X)=1} \sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)(M_i - b_i I)\bullet X \\
&= T \max_{X\succeq 0,\, \mathrm{Tr}(X)=1} \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)(M_i - b_i I)\bullet X \\
&= T \max_{X\succeq 0,\, \mathrm{Tr}(X)=1} \sum_{i=1}^{m} \bar{p}(i)(M_i - b_i I)\bullet X \geq T\cdot OPT \quad / \ \bar{p} := \frac{1}{T}\sum_{t=1}^{T} p_t
\end{aligned}$$

According to the MW Lemma (see for instance Lemma 1 in [37]), there

exists a choice for η_d such that

$$\begin{aligned}\text{regret}_T(MW) &:= \sum_{t=1}^{T}\sum_{i=1}^{m} p_t(i)\left(x_t^\top M_i x_t - b_i\right) \\ &\quad - \min_{i\in[m]} \sum_{t=1}^{T} x_t^\top M_i x_t - b_i = O(\sqrt{T\log(m)}). \quad (4.14)\end{aligned}$$

Thus, by combining Eq. (4.13), (4.14) and rearranging we have that with probability at least $1 - 1/n$ it holds that

$$\begin{aligned}\min_{i\in[m]} \sum_{t=1}^{T} x_t^\top (M_i - b_i I) x_t &\geq T\cdot OPT - \text{regret}_T(FPL) - \text{regret}_T(MW) \\ &= T\cdot OPT - \tilde{O}(\sqrt{nT}).\end{aligned}$$

The bound on the approximation error follows now from dividing through by T, setting $T = \tilde{O}(n/\epsilon^2)$, and using the definition of $\bar{X}$.

To bound the run-time, note that all computations required to update p_t on each iteration could be carried out in $O(\text{nnz})$ time. According to Theorem 18, the update of the primal variable x_t on each iteration could be carried out in $\tilde{O}(n^{-1/4}T^{3/4}\min\{\text{nnz}, n^2\})$ time. Plugging in the number of iterations T, gives the total run-time. □

While the run-time stated in Theorem 20 falls behind state-of-the-art methods such as [37, 21], it already beats eigen-decomposition-based methods, such as Nesterov's smoothing approach [72], when nnz is small and ϵ is large (say a small constant).

The run-time of Algorithm 12 could be further accelerated using random sampling techniques such as those used in [37] and state-of-the-art stochastic algorithms for eigenvector computations [36]. Such improvements however are beyond the scope of this chapter.

4.7.2 Smoothing the dual SDP problem via rank-one perturbations

Consider now the dual SDP problem given in (4.12) which can be also written as follows:

$$\min_{p\in\Delta_m}\{g(p) := \lambda_{\max}\left(\sum_{i=1}^{m} p(i)(M_i - b_i I)\right)\}. \tag{4.15}$$

Suppose we wish to apply first-order gradient methods, such as the well known *projected gradient descent* method, to directly solve Problem (4.15). Unfortunately the function $\lambda_{\max}(\cdot)$ is a non-smooth convex function (and thus $g(p)$ is not smooth) which means that, using standard analysis, we can only hope to achieve a convergence rate of roughly $1/\sqrt{\epsilon}$ for it. That is, it will require $O(\epsilon^{-2})$ subgradient descent iterations to achieve an ϵ approximation to the optimal value. A well known approach to overcome this shortcoming is by *smoothing* the function $\lambda_{\max}$, which in turn, will enable to apply *Nesterov's accelerated gradient method*, which converges to ϵ-optimal solution after roughly $\sqrt{\beta/\epsilon}$ iterations, where β is the smoothness parameter.

Thus, for the remaining of this section we focus on smoothing techniques for the function $\lambda_{\max}(X)$.

We first recall the definition of a smooth function.

Definition 10. *Let $\boldsymbol{E}$ be a finite-dimensional vector space. We say that a function $f : \boldsymbol{E} \to \mathbb{R}$ is β-smooth with respect to norm $\|\cdot\|$ if for all $x, y \in \boldsymbol{E}$ it holds that*

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|.$$

In order to better understand why $\lambda_{\max}(\cdot)$ is not smooth and how to overcome this shortcoming, we have the following lemma, which follows from Theorem 1.1 and Theorem 1.3 in [66], and shows that the derivative of $\lambda_{\max}(\cdot)$ is strongly connected to the eigengap of the queried matrix.

Lemma 39. *Let $X \in \mathbb{S}_n$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ and denote by $u_1, ..., u_n$ the corresponding eigenvectors.*

1. *The function $\lambda_{\max}$ is differential at the point X if and only if $\delta(X) = \lambda_1(X) - \lambda_2(X) > 0$, and then the gradient is given by $\nabla\lambda_{\max}(X) = u_1 u_1^\top$.*

2. *The differential set of* $\lambda_{\max}(\cdot)$ *at point* $X \in \mathbb{S}_n$ [3]*, denoted* $\partial\lambda_{\max}(X)$*, is the convex hull of all rank-one matrices* $uu^\top$ *such that* u *is a unit vector, and* $u^\top X u = \lambda_{\max}(X)$*.*

The following lemma gives a well known and popular generic technique for smoothing based on Gaussian perturbations.

Lemma 40 (Folklore, see for instance Lemma 9 in [25])**.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a convex function with subgradients bounded by* L *with respect to the Euclidean norm, i.e.* $\forall x \in \mathbb{R}^n\ \forall y \in \partial f(x)$*:* $\|y\|_2 \leq L$*. Consider the function* $\tilde{f}(x) = \mathbb{E}_{v\sim\mathcal{N}(0_n,I)}[f(x+cv)]$*, where* c *is some positive scalar. Then, if we set* $c = \frac{\epsilon}{L\sqrt{n}}$*, the following claims hold:*

1. $\forall x \in \mathbb{R}^n$*:* $f(x) \leq \tilde{f}(x) \leq f(x) + \epsilon$*.*
2. $\tilde{f}(x)$ *is differentiable and* $O(\sqrt{n}/\epsilon)$*-smooth.*
3. *If* $v \sim \mathcal{N}(0_n, I)$ *then* $\mathbb{E}_v[\nabla f(x+cv)] = \nabla\tilde{f}(x)$*, i.e.* $\nabla f(x+cv)$ *is an unbiased estimator for* $\nabla\tilde{f}(x)$*.*

Using the second part of Lemma 39, we can see that for any $X \in \mathbb{S}_n$ and any $Y \in \partial\lambda_{\max}(X)$ it holds that $\|Y\|_F \leq 1$ (since Y is a convex combination of unit-length rank one matrices). Thus, Lemma 40 gives a smoothing technique for $\lambda_{\max}$ with smoothness parameter n/ϵ (recall that our linear space is $\mathbb{S}_n$ and not $\mathbb{R}^n$). A problem with the above generic smoothing technique is that, as Lemma 39 suggests, in our case, in order to compute derivatives of $\tilde{f}$ (or at least estimate them), we are going to need to perform eigenvector computations on matrices of the form $X + cN$ where N is sampled uniformly from the unit frobenius norm ball in $\mathbb{S}_n$, and is thus dense with high probability. Thus, even if X is sparse, the eigenvector computations will be performed on a dense matrix which is highly unfavorable.

We note here that there is also a *deterministic* smoothing technique for $\lambda_{\max}$ due to Nesterov [72], but computing the gradient of the smoothed function requires eigen-decomposition and thus inherently runs in time $O(n^3)$.

We are now going to show how using our rank-one perturbation technique and corresponding analysis, introduced in Section 4.5, can yield a smoothing

[3] for a non-smooth convex function f, the subdifferential set of f at point x, denoted by $\partial f(x)$, is the set of all vectors y such that for all $z \in \mathbb{R}^n$ it holds that $(x-z)\cdot y \geq f(x) - f(z)$. In case f is differentiable at x, it holds that $\partial f(x) = \{\nabla f(x)\}$.

of $\lambda_{\max}$ (with roughly the same smoothness parameter), such that computing derivatives of the smoothed function will depend on the sparsity of the matrix X, and not explicitly on the dimension.

Towards this end, consider the function

$$f(X) := \mathbb{E}_{v\sim\mathcal{N}(0_{n\times n},I)}[\lambda_{\max}(X + cvv^\top)], \tag{4.16}$$

where $c > 0$ is a parameter.

Lemma 41. *Consider the function $f(X)$ as defined in* (4.16) *with parameter $c = \epsilon/n$. Then the following claims hold:*

1. $\forall X \in \mathbb{S}_n$: $0 \leq f(X) - \lambda_{\max}(X) \leq \epsilon$.

2. *With probability 1, $f(X)$ is differentiable and $\tilde{O}(n/\epsilon)$-smooth with respect to the frobenius norm.*

3. *If $v \sim \mathcal{N}(0_{n\times n}, I)$ and $u_{X,v}$ is the leading eigenvector of $X + cvv^\top$ then, with probability 1, $\mathbb{E}_v[u_{X,v}u_{X,v}^\top] = \nabla f(X)$. Moreover, computing $u_{X,v}$ could be carried out in time that depends explicitly only on $\max\{nnz(X), n\}$ and not on n^2.*

Proof. The first claim holds since for any $X \in \mathbb{S}_n$ it follows using simple calculations that

$$\begin{aligned} 0 &\leq f(x) - \lambda_{\max}(X) = \mathbb{E}_v\left[\lambda_{\max}(X + cvv^\top) - \lambda_{\max}(X)\right] \\ &\leq \mathbb{E}_v\left[\lambda_{\max}(cvv^\top)\right] = c \cdot n. \end{aligned}$$

For the second claim, using Lemma 6.4 in [21], we can exchange the order of differentiation and expectation and thus we have that

$$\nabla f(X) = \mathbb{E}_{v\sim\mathcal{N}(0_{n\times n},I)}[\nabla\lambda_{\max}(X + cvv^\top)]. \tag{4.17}$$

According to Theorem 17, given a random vector $v \sim \mathcal{N}(0_{n\times n}, I)$, it holds with probability 1 that $\delta(X + cvv^\top) > 0$. Thus, by Lemma 39 we have that with probability 1, $\nabla\lambda_{\max}(X + cvv^\top)$ exists, and thus it follows form Eq. (4.17) that the derivative exists with probability 1. Thus, we have that indeed $f(X)$ is differentiable with probability 1.

To bound the smoothness parameter we proceed as follows. Let $X, Y \in \mathbb{S}_n$. Using Eq. (4.17) again, we have that

$$\begin{aligned}
\|\nabla f(X) - \nabla f(Y)\|_F &= \|\mathbb{E}_v \left[\nabla \lambda_{\max}(X + cvv^\top) - \nabla \lambda_{\max}(Y + cvv^\top)\right]\|_F \\
&= \|\mathbb{E}_v \left[u_{X,v} u_{X,v}^\top - u_{Y,v} u_{Y,v}^\top\right]\|_F \\
&\leq \mathbb{E}_v \left[\|u_{X,v} u_{X,v}^\top - u_{Y,v} u_{Y,v}^\top\|_F\right],
\end{aligned}$$

where $u_{X,v}, u_{Y,v}$ denote the largest eignenvectors of $X + cvv^\top, Y + cvv^\top$, respectively (which as discussed before, are unique with probability 1). Note that the second equality follows again from Lemma 39.

Using the Davis-Kahan Sine Theorem (Theorem 16), we thus have that

$$\begin{aligned}
\|\nabla f(X) - \nabla f(Y)\|_F &\leq O(1) \cdot \mathbb{E}_v \left[\frac{\|X - Y\|}{\delta(X + cvv^\top)}\right] \\
&\leq O(1) \cdot \mathbb{E}_v \left[\frac{1}{\delta(X + cvv^\top)}\right] \|X - Y\|_F \\
&= \tilde{O}(c^{-1}) \cdot \|X - Y\|_F,
\end{aligned}$$

where the last inequality follows from Lemma 38. Thus, we have proved the second claim.

Finally, in order to prove the third claim, we can proceed along the same lines of the proof of the second one: we have that with probability 1,

$$\nabla f(X) = \mathbb{E}_{v \sim \mathcal{N}(0_{n \times n}, I)}[\nabla \lambda_{\max}(X + cvv^\top)] = \mathbb{E}_v[u_{X,v} u_{X,v}^\top],$$

and thus $u_{X,v} u_{X,v}^\top$ is an unbiased estimator for $\nabla f(X)$.

Since computing the leading eigenvector $u_{X,v}$ could be carried out via iterative methods in time that depends on the complexity of multiplying a vector with the matrix $X + cvv^\top$ (and not explicitly n^2), we have that it could be carried out in time proportional to $\max\{\text{nnz}(X), n\}$. $\square$

We conclude this section by mentioning that a smoothing technique for $\lambda_{\max}$ which is also based on rank-one perturbations was recently proposed in [21] (and our proof of Lemma 41 use a certain component of their analysis).

They consider the following smoothed function:

$$\tilde{f}_k(x) := \mathbb{E}_{v_1,v_2,\ldots,v_k \sim \mathcal{N}(0_{n\times n},I)} \left[\max_{i\in[k]} \lambda_{\max}(X + cv_iv_i^\top) \right],$$

where k is a positive integer and c is a positive scalar. That is, they consider using k independent rank-one perturbations and taking the one that yields the largest eigenvalue. They show (see Proposition 3.7 in [21]) that for $k \geq 3$ and $c = \epsilon/n$, $\tilde{f}$ is indeed a smoothing of $\lambda_{\max}$ with roughly the same smoothing parameter as given in Lemma 41. Quite interestingly, we show that this smoothing technique works already for $k = 1$ which greatly simplifies both the analysis and possible implementations, and is more straightforward.

4.8 Open Questions

The following questions seem of interest. Is it possible to generalize the algorithms presented in this chapter to handle the case in which the prediction on each iteration is not a rank-one matrix, but a rank-k projection matrix, for some integer $k \geq 1$? A special case of this more general settings setting is the *online PCA* problem, studied in [91].

Is it possible to improve the dependence of the regret of the algorithm presented in Section 4.5 in terms of the dimension? We get a dependence of $\sqrt{n}$, while the algorithm in [29] manages to get a dependence of $n^{1/4}$, but only in case the observed matrices are rank-one. Also, the algorithm in [29] uses a fully-dense perturbation and hence the eigenvector computations are potentially much less efficient than in our algorithms. Thus, the interesting question is: is it possible to improve the dependence on the dimension in the regret of our algorithm to $n^{1/4}$, in the case that the rank of the observed matrices is arbitrary, while still having the property that the eigenvector computations depend only on the sparsity of the observed data?

It is also interesting to come up with a lower bound on the regret of FPL-based algorithms for this problem (which is the natural approach for a eigenvector computation-based method), so we know exactly how further down it is possible to push the dependence on the dimension.

4.9 Proofs of Supporting Lemmas

4.9.1 Proofs of Lemmas from section 4.2

Proof of Lemma 31

Proof. Fix a time $t \in [T]$. Clearly the matrix $\tilde{A}_t$ is symmetric. Moreover, given a unit vector $x \in \mathbb{R}^{m\times n}$ write x as $x = (u, v)$ where $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$. It holds that

$$\begin{aligned}
\|\tilde{A}_t x\|^2 &= \|(A_t v, A_t^\top u)\|^2 = \|A_t v\|^2 + \|A_t^\top u\|^2 \\
&= \|v\|^2 \cdot \|A_t \frac{v}{\|v\|}\|^2 + \|u\|^2 \cdot \|A_t^\top \frac{u}{\|u\|}\|^2 \\
&\leq \|v\|^2 + \|u\|^2 = 1,
\end{aligned}$$

where the last inequality follows since $\|A_t\| \leq 1$. Hence $\|\tilde{A}_t\| \leq 1$. We now turn to bound the regret. It holds that $\sum_{t=1}^T \tilde{A}_t = \begin{pmatrix} 0_{m\times m} & \sum_{t=1}^T A_t \\ \sum_{t=1}^T A_t^\top & 0_{n\times n} \end{pmatrix}$. Given a unit vector $x \in \mathbb{R}^{m+n}$, let us write it as before as $x = (u, v)$ where $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$. It holds that

$$\begin{aligned}
x^\top \left(\sum_{t=1}^T \tilde{A}_t\right) x &= u^\top \left(\sum_{t=1}^T A_t\right) v + v^\top \left(\sum_{t=1}^T A_t^\top\right) u \\
&= 2u^\top \left(\sum_{t=1}^T A_t\right) v \\
&= 2\|u\|\|v\| \frac{u^\top}{\|u\|} \left(\sum_{t=1}^T A_t\right) \frac{v}{\|v\|}.
\end{aligned}$$

Clearly the RHS is maximized when u, v are in the direction of the top left and right singular vectors of $\sum_{t=1}^T A_t$ respectively. Moreover, under the constraint that $\|x\| = 1$, the rhs is further maximized when $\|u\| = \|v\| = \frac{1}{\sqrt{2}}$, in which case it equals exactly $\sigma_{\max}\left(\sum_{t=1}^T A_t\right)$. The maximum of the LHS with respect to $x \in \mathcal{S}$ is by definition $\lambda_{\max}\left(\sum_{t=1}^T \tilde{A}_t\right)$. Thus we have that

$$\lambda_{\max}\left(\sum_{t=1}^T \tilde{A}_t\right) = \sigma_{\max}\left(\sum_{t=1}^T A_t\right). \tag{4.18}$$

Note that for every matrix $A \in \mathbb{R}^{m\times n}$, given two unit vectors $u \in \mathbb{R}^m, v \in \mathbb{R}^n$ chosen uniformly from the unit spheres in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively, it holds that $\mathbb{E}_{u,v}[u^\top Av] = 0$. Thus on any time t it holds that

$$\begin{aligned}\mathbb{E}_{u_t,v_t}[u_t^\top A_t v_t] &= 2\|\tilde{u}_t\|\|\tilde{v}_t\|\frac{\tilde{u}_t^\top}{\|\tilde{u}_t\|}A_t\frac{\tilde{v}_t}{\|\tilde{v}_t\|} + (1-2\|\tilde{u}_t\|\|\tilde{v}_t\|)\mathbb{E}_{u,v}[u^\top A_t v] \\ &= 2\tilde{u}_t^\top A_t\tilde{v}_t + (1-2\|\tilde{u}_t\|\|\tilde{v}_t\|)\cdot 0 \\ &= 2\cdot\frac{1}{2}(\tilde{u}_t^\top A_t\tilde{v}_t + \tilde{v}_t^\top A_t^\top\tilde{u}_t) = x_t^\top\tilde{A}_t x_t.\end{aligned} \tag{4.19}$$

Combining Eq. (4.18) and Eq. (4.19) we have that

$$\mathbb{E}\left[\sigma_{\max}\left(\sum_{t=1}^T A_t\right) - \sum_{t=1}^T u_t^\top A_t v_t\right] = \lambda_{\max}\left(\sum_{t=1}^T \tilde{A}_t\right) - \sum_{t=1}^T x_t^\top\tilde{A}_t x_t.$$

□

4.9.2 Proofs of Theorems and Lemmas from section 4.4

Proof of Theorem 14

Proof. Let us consider the noise matrix N as an additional round of the game, that is assume that $A_0 = N$. In this case, it is clear that playing the point x_{t+1} on time t is equivalent to playing the optimal choice with respect to the cumulative matrix $\sum_{\tau=0}^t A_\tau$. In [56] it was shown that this policy, of playing on each time t the optimal choice with respect to sum of rewards up to time t (included) achieves over all zero regret. Thus we have that

$$\begin{aligned}x_1^\top Nx_1 + \sum_{t=1}^T x_{t+1}A_t x_{t+1} &\geq \max_{x\in\mathcal{S}} x^\top Nx + \sum_{t=1}^T x^\top A_t x \\ &\geq x^{*\top}Nx^* + \sum_{t=1}^T x^{*\top}A_t x^*.\end{aligned}$$

Subtracting the player's gain from both sides and rearranging leads to

$$\sum_{t=1}^{T} x^{*\top} A_t x^* - \sum_{t=1}^{T} x_t^\top A_t x_t \leq$$
$$\sum_{t=1}^{T} \left(x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t \right) + x_1^\top N x_1 - x^* N x^*.$$

The result follows from taking expectation on both sides. □

Proof of Lemma 34

Proof. In order to upper bound D_1 we observe that for any two unit vectors x, y it holds that

$$\|xx^\top - yy^\top\|_1 \leq \sqrt{n^2}\|xx^\top - yy^\top\|_F = O(n).$$

This upper bound on D_1 is also tight. To see this, take the vectors $x = \mathbf{e}_1$ (the first n-dimensional standard basis vector) and $y = \frac{1}{\sqrt{n}}\mathbf{1}$ (the uniform unit vector). It holds that $\|xx^\top - yy^\top\|_1 = \Omega(n)$.

To upper bound $\|A_t\|_1$ and recalling that $\|A_t\| \leq 1$ we do the following.

$$\|A_t\|_1 \leq \sqrt{n^2}\|A_t\|_F \leq \sqrt{n^2}\sqrt{n\|A_t\|^2} \leq n^{3/2},$$

where the second inequality follows since for any symmetric matrix A, $\|A\|_F^2 = \sum_{i=1}^n \lambda_i^2(A) \leq n\|A\|^2$. We now show that the upper-bound on $\|A_t\|_1$ is also tight. To see this, assume with out loss of generality that n is a power of 2 and consider the famous deterministic construction of an Hadamard matrix $\mathcal{H}$ due to Sylvester. $\mathcal{H}$ has the following two properties: 1) $\forall i, j : |\mathcal{H}_{i,j}| = 1$ and 2) $\|\mathcal{H}\| \leq \sqrt{n}$. Thus by considering the matrix $A_t = \frac{1}{\sqrt{n}}\mathcal{H}$ we have that $\|A_t\|_1 = \frac{1}{\sqrt{n}}n^2 = n^{3/2}$. There is a slight technical issue that by defining A_t this way it is not positive semidefinite. However this could be easily remedied by adding to it the identity matrix and multiplying the result by a factor of 1/2 to keep the upper bound on the spectral norm. This changes still ensure that the ℓ_1 norm of the matrix is $\Theta(n^{3/2})$. □

Proof of Lemma 36

Proof. Let us denote by $F(t, M)$ the function

$$F(t, M) = [\mathbf{EV}(M)]^\top A_t [\mathbf{EV}(M)].$$

Using the above notation and denoting $S_t = \sum_{\tau=1}^{t-1} A_\tau$, we have that

$$\begin{aligned}
&\mathbb{E}_{N\sim\mathcal{D}_{exp}}[(x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t^\top] = \\
&\int_{\mathbb{S}_n} F(t, S_{t+1} + N')d\mu(N') - \int_{\mathbb{S}_n} F(t, S_t + N)d\mu(N) = \\
&\int_{\mathbb{S}_n} F(t, S_t + N)d\mu(N - A_t) - \int_{\mathbb{S}_n} F(t, S_t + N)d\mu(N).
\end{aligned} \tag{4.20}$$

It holds that

$$\begin{aligned}
\frac{d\mu(N - A_t)}{d\mu(N)} &= \frac{\exp(-\epsilon\|N - A_t\|)}{\exp(-\epsilon\|N\|)} \\
&= \exp(-\epsilon\|N - A_t\| + \epsilon\|N\|) \\
&\leq \exp(-\epsilon\|N - A_t\| + \epsilon\|N - A_t\| + \epsilon\|A_t\|) \\
&= \exp(\epsilon\|A_t\|) \\
&\leq \exp(\epsilon) \leq 1 + e\epsilon,
\end{aligned}$$

where the last inequality holds for all $\epsilon \in [0, 1]$.

Plugging the above into Eq. (4.20) we have that

$$\begin{aligned}
\mathbb{E}_{N\sim\mathcal{D}_{exp}}[(x_{t+1}^\top A_t x_{t+1} - x_t^\top A_t x_t^\top] &\leq e\epsilon \int_{\mathbb{S}_n} F(t, S_n + N)d\mu(N) \\
&= e\epsilon \cdot \mathbb{E}[x_t^\top A_t x_t].
\end{aligned}$$

The lemma follows since $\|A_t\| \leq 1$. □

4.9.3 Proofs of Theorems and Lemmas from section 4.5

Proof of Theorem 16

Proof. Write u_B as $u_B = \alpha u_A + \beta w$ for some unit vector $w \in \mathbb{R}^n$ and scalars α, β that satisfy: $u_A^\top w = 0$, $\alpha^2 + \beta^2 = 1$. Note that the following two

conditions are also satisfied:

1. $w^\top A w \leq \lambda_2(A) = \lambda_1(A) - \delta(A)$.

2. $u_A^\top A w = w^\top A u_A = 0$.

It thus holds that

$$\begin{aligned} u_B^\top B u_B &= u_B^\top A u_B + u_B^\top E u_B \\ &\leq \alpha^2 \lambda_1(A) + \beta^2(\lambda_1(A) - \delta(A)) + u_B^\top E u_B \\ &= \lambda_1(A) - \beta^2 \delta(A) + u_B^\top E u_B. \end{aligned} \tag{4.21}$$

On the other hand it holds that

$$\begin{aligned} u_B^\top B u_B^\top \geq u_A^\top B u_A &= u_A^\top A u_A + u_A^\top E u_A \\ &= \lambda_1(A) + u_A^\top E u_A. \end{aligned} \tag{4.22}$$

Combining Eq. (4.21) and Eq. (4.22) we have that

$$\begin{aligned} \|u_A u_A^\top - u_B u_B^\top\|_F^2 &= 2(1 - (u_A^\top u_B)^2) = 2\beta^2 \\ &\leq 2\frac{u_B^\top E u_B - u_A^\top E u_A}{\delta(A)} \\ &\leq 2\frac{\|E\| \cdot \|u_A u_A^\top - u_B u_B^\top\|_*}{\delta(A)} \\ &\leq 2\sqrt{2}\frac{\|E\| \cdot \|u_A u_A^\top - u_B u_B^\top\|_F}{\delta(A)}, \end{aligned}$$

where the second inequality follows from Holder's inequality and the last one follows since $u_A u_A^\top - u_B u_B^\top$ is a rank-two matrix.

Thus we have that

$$\|u_A u_A^\top - u_B u_B^\top\|_F \leq 2\sqrt{2}\frac{\|E\|}{\delta(A)}.$$

□

Proof of Lemma 38

Proof. Since X takes only non-negative values we have that

$$\begin{aligned}\mathbb{E}[X] &= \int_0^\infty \Pr[X \geq t]dt = \int_0^\infty \Pr[\min(a, \delta^{-1}) \geq t]dt \\ &= \int_0^a \Pr[\min(a, \delta^{-1}) \geq t]dt = \int_0^a \Pr[\delta^{-1} \geq t]dt \\ &= \int_0^a \Pr[\delta \leq 1/t]dt \leq \int_0^a \min\left\{\frac{2\sqrt{2}}{\pi ct}, 1\right\} dt, \end{aligned} \tag{4.23}$$

where the last inequality follows from applying Theorem 17.
If $a \leq \frac{2\sqrt{2}}{\pi c}$, we have that

$$\text{RHS of } (4.23) \leq a \leq \frac{2\sqrt{2}}{\pi c}.$$

On the other hand, if $a > \frac{2\sqrt{2}}{\pi c}$, we have that

$$\begin{aligned}\text{RHS of } (4.23) &\leq \int_0^{\frac{2\sqrt{2}}{\pi c}} 1dt + \int_{\frac{2\sqrt{2}}{\pi c}}^a \frac{2\sqrt{2}}{\pi ct}dt \\ &= \frac{2\sqrt{2}}{\pi c} + \frac{2\sqrt{2}}{\pi c}\ln(a\pi c/2\sqrt{2}) \\ &= \frac{2\sqrt{2}}{\pi c}\ln(\frac{\pi eac}{2\sqrt{2}}).\end{aligned}$$

Therefore under both cases, we have

$$\mathbb{E}[X] \leq \text{ RHS of } (4.23) \leq \frac{2\sqrt{2}}{\pi c}\max\{\ln(\frac{\pi eac}{2\sqrt{2}}), 1\},$$

as desired. □

Proof of Theorem 17

We prove an equivalent version of Theorem 17 that has a more convenient scaling.

Theorem 21. *For any $M \in \mathbb{S}_n$, the perturbed matrix $\widetilde{M} = M + vv^T$ with $v \sim \mathcal{N}(0, \boldsymbol{I}_{n\times n})$ satisfies that for any $t > 0$*

$$\Pr\left[\delta(\widetilde{M}) < t\right] \leq \frac{2\sqrt{2}}{\pi} t. \tag{4.24}$$

Theorem 17 follows from invoking Theorem 21 with $M = \frac{1}{c}A$ and $t = \frac{\epsilon}{c}$.

Although seemingly we need to prove some kind of concentration result, actually the key idea behind it is the following anti-concentration result for the top eigenvalue.

Theorem 22. *For any $M \in \mathbb{S}_n$, any $\lambda > \lambda_1(M)$ and $t > 0$, the perturbed matrix $\widetilde{M} = M + vv^T$ with $v \sim \mathcal{N}(0, \boldsymbol{I}_{n\times n})$ satisfies,*

$$\Pr\left[\lambda_1(\widetilde{M}) \in [\lambda, \lambda + t]\right] \leq \sqrt{\frac{2t}{\pi}}. \tag{4.25}$$

We first give a high level sketch for the proof of Theorem 21. The basic idea is as follows: Let $\mathcal{E}$ be the bad event that $\delta(\widetilde{M}) = \lambda_1(\widetilde{M}) - \lambda_2(\widetilde{M})$ is smaller than t. We upperbound the event $\mathcal{E}$ by the intersection of two independent events $\mathcal{E}_1$ and $\mathcal{E}_2$. Events $\mathcal{E}_1$ and $\mathcal{E}_2$ come from two consequences of $\lambda_1(\widetilde{M})$ and $\lambda_2(\widetilde{M})$ being t-close to each other. Note that by eigenvalue interlacing, we have that $\lambda_2(\widetilde{M}) \leq \lambda_1(M) \leq \lambda_1(\widetilde{M})$. Therefore if $\mathcal{E}$ happens, then we have event (a) $\sigma_1(\widetilde{M}) \leq \lambda_1(M)+t$, and event (b) $\lambda_2(\widetilde{M}) \geq \sigma_1(M)-t$.

Though events (a) and (b) are not independent, we can carefully design two independent events $\mathcal{E}_1$ and $\mathcal{E}_2$ that are closely related to events (a) and (b), respectively. Event $\mathcal{E}_1$ will only depend on correlation between v and the top eigenvector of M, and event $\mathcal{E}_2$ will only depend on correlation of v between the rest of eigenvectors of M, and therefore they are independent (see the proof for details), and then we conclude that $\Pr[\mathcal{E}] \leq \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2]$. Hence it suffices to upperbound the probabilities of $\mathcal{E}_1$ and $\mathcal{E}_2$ by $O(\sqrt{t})$ for the theorem to follow. The bound on $\Pr[\mathcal{E}_1]$ follows from the same arguments used to prove Lemma 37, i.e. a simple anti-concentration result for the dot product of a fixed unit vector with a random vector and $\Pr[\mathcal{E}_2]$ is upper bounded using Theorem 22.

Proof of Theorem 21. Since v has a distribution that is rotational invariant, we can assume, without loss of generality, that M is diagonal. Let $M = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues and $e_1, \ldots, e_n$ are the corresponding eigenvectors. Define $\mathcal{E}$ to be the event that $\delta(\widetilde{M}) < t$.

By eigenvalue interlacing, we have that $\lambda_2(\widetilde{M}) \leq \lambda_1(M) = \lambda_1$. Therefore $\delta(\widetilde{M}) \leq t$ implies that $\lambda_1(\widetilde{M}) \leq \lambda_1 + t$. Note that $\lambda_1(\widetilde{M}) \geq e_1\widetilde{M}e_1 = \lambda_1 + \langle e_1, v\rangle^2 = \lambda_1 + v_1^2$. Therefore, $\lambda_1(\widetilde{M}) \leq \lambda_1 + t$ further implies that $v_1^2 \leq t$. Let $\mathcal{E}_1$ be the event that $v_1^2 \leq t$. Therefore we have just proved that $\mathcal{E} \subset \mathcal{E}_1$. Note that v_1^2 is a Chi-squared random variable with a single degree of freedom, also denoted as χ_1^2, and thus

$$\Pr(v_1^2 < t) = \int_0^t \frac{e^{-x/2}}{\sqrt{2\pi x}} dx \leq \frac{1}{\sqrt{2\pi}} \int_0^t \frac{1}{\sqrt{x}} dx = \sqrt{\frac{2}{\pi}} t. \tag{4.26}$$

We consider defining another event $\mathcal{E}_2$ that involves a delicate treatment for $\lambda_2(\widetilde{M})$. Observe that $\lambda_1(\widetilde{M}) \geq \lambda_1$, and therefore when $\mathcal{E}$ happens, it must be $\lambda_2(\widetilde{M}) \geq \lambda_1(\widetilde{M}) - t \geq \lambda_1 - t$. Using the variational characterization of eigenvalues (see Lemma 43), by choosing $u = e_1$, we have that

$$\begin{aligned}
\lambda_2(\widetilde{M}) \leq & \max_{\substack{x \perp e_1 \\ \|x\| = 1}} x^T \widetilde{M} x \\
= & \max_{\substack{x' \in \mathbb{R}^{n-1} \\ \|x'\| = 1}} x'^T \left(\operatorname{diag}(\lambda_2, \ldots, \lambda_n) + v'v'^T\right) x' \\
= & \lambda_1 \left(\operatorname{diag}(\lambda_2, \ldots, \lambda_n) + v'v'^T\right),
\end{aligned} \tag{4.27}$$

where v' is the $n-1$-dimensional vector obtained by restricting v to the support $\{2, \ldots, n\}$. Let $M' = \operatorname{diag}(\lambda_2, \ldots, \lambda_d)$ and $\widetilde{M}' = \operatorname{diag}(\lambda_2, \ldots, \lambda_n) + v'v'^T$. By Eq. (4.27) we have that $\lambda_1(\widetilde{M}') \geq \lambda_2(\widetilde{M})$ and therefore when $\mathcal{E}$ happens, we have that

$$\lambda_1(\widetilde{M}') \geq \lambda_2(\widetilde{M}) \geq \lambda_1(\widetilde{M}) - t \geq \lambda_1 - t. \tag{4.28}$$

On the other hand, note that $\lambda_1(\widetilde{M}') = \max_{x \perp e_1, \|x\|=1} x^T \widetilde{M} x \leq \lambda_1(\widetilde{M})$, therefore when $\mathcal{E}$ happens, we also have

$$\lambda_1(\widetilde{M}') \leq \lambda_1(\widetilde{M}) \leq \lambda_1 + t. \tag{4.29}$$

Combining Eq. (4.28) and Eq. (4.29), we now define $\mathcal{E}_2$ be the event that $\lambda_1(\widetilde{M'}) \in [\lambda_1 - t, \lambda_1 + t]$, and by the arguments above we have $\mathcal{E} \subset \mathcal{E}_2$.

In case $\lambda_1 - t \leq \lambda_2$ we have, as in the analysis of $\Pr[\mathcal{E}_1]$, that

$$\begin{aligned}
\Pr[\lambda_1(\widetilde{M'}) \in [\lambda_1 - t, \lambda_1 + t]] &\leq \Pr[\lambda_1(\widetilde{M'}) \leq \lambda_1 + t] \\
&\leq \Pr[e_2^\top \widetilde{M} e_2 \leq \lambda_1 + t] \\
&\leq \Pr[\lambda_2 + v_2^2 \leq \lambda_1 + t] \\
&\leq \Pr[v_2^2 \leq 2t] \leq \sqrt{\frac{4}{\pi} t},
\end{aligned}$$

where the forth inequality follows since $\lambda_1 - t \leq \lambda_2$ and the last inequality follows from Eq. (4.26).

In case $\lambda_1 - t > \lambda_2$ we can upper bound $\Pr[\lambda_1(\widetilde{M'}) \in [\lambda_1 - t, \lambda_1 + t]]$ by invoking Theorem 22 with parameters $M', \lambda_1 - t, 2t$. In both cases we conclude that $\Pr[\mathcal{E}_2] = \Pr[\sigma_1(\widetilde{M'}) \in [\lambda_1 - t, \lambda_1 + t]] \leq \sqrt{\frac{4t}{\pi}}$.

Finally, note that $\mathcal{E}_1$ is an event that only depends on v_1 and $\mathcal{E}_2$ is an event that only depends on $v_2, \ldots, v_n$. Thus $\mathcal{E}_1$ and $\mathcal{E}_2$ are two independent events and we conclude $\Pr[\mathcal{E}] \leq \Pr[\mathcal{E}_1 \cap \mathcal{E}_2] = \Pr[\mathcal{E}_1] \Pr[\mathcal{E}_2]$ and the theorem follows. □

Now we prove Theorem 22. Before going into detail, let's sketch the key idea. We first use the fact that the top eigenvector of a matrix A is the max root of the polynomial $p_A(z) = \det(zI - A)$ and obtain that $\sigma_1(\widetilde{M})$ is the max root of the polynomial:

$$\sum_{i=1}^{n} \frac{v_i^2}{z - \lambda_i} = 1, \tag{4.30}$$

where we assumed as in the proof of Theorem 21, $M = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$. We use $z^*(v_1, \ldots, v_n)$ to denote the maximum root of (4.30). We are going to show that z^* is pretty sensitive to the choice of v_1 in the following sense: If we fix $v_2, \ldots, v_n$, and moving v_1^2 from u to $u + t$, then z^* must move from some place z_0 to $z_0 + t'$ with $t' > t$. In other words, z^* should have at least the same anti-concentration property as v_1^2.

Proof of Theorem 22. As in the proof of Theorem 21, we assume w.l.o.g. that $M = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. We are going to use the fact that top eigenvector

of a matrix A is the max root of the polynomial $p_A(z) = \det(zI - A)$. We calculate the characteristic polynomial for $\widetilde{M}$:

$$\begin{aligned} p_{\widetilde{M}}(z) &= \det(zI - \widetilde{M}) = \det(zI - M - vv^T) \\ &= \det\Big((zI - M)(I - (zI - M)^{-1}vv^\top)\Big) \\ &= \det(zI - M) \cdot \det(I - (zI - M)^{-1}vv^T), \end{aligned}$$

where the last equality follows since for any two square matrices A, B it holds that $\det(AB) = \det(A) \cdot \det(B)$. Using Sylvester's determinant identity (Lemma 42), we have that

$$\begin{aligned} p_{\widetilde{M}}(z) &= \det(zI - M) \cdot (1 - v^T(zI - M)^{-1}v) \\ &= \left(\prod_{j=1}^{n}(z - \lambda_j)\right) \cdot (1 - v^T \operatorname{diag}(\frac{1}{z - \lambda_i})v) \\ &= \left(\prod_{j=1}^{n}(z - \lambda_j)\right) \cdot \left(1 - \sum_{i=1}^{n} \frac{v_i^2}{z - \lambda_i}\right) \end{aligned}$$

Let $f(z) = \sum_{i=1}^{d} \frac{v_i^2}{z-\lambda_i}$. Since with probability 1 non of the roots of $p_{\widetilde{M}}(z)$ are in $\{\lambda_1, \lambda_2, ..., \lambda_n\}$, we have that $\lambda_1(\widetilde{M})$ is the largest solution of $f(z) = 1$. Note that $f(z)$ is well-defined and decreasing on (λ_1, ∞). Thus, fixing a value $\lambda \in (\lambda_1, \infty)$, we have that the event that $\lambda_1(\widetilde{M}) \in [\lambda, \lambda + t]$ is equivalent to both of the following conditions,

$$f(\lambda) = \sum_{i=1}^{d} \frac{v_i^2}{\lambda - \lambda_i} \geq 1, \tag{4.31}$$

$$f(\lambda + t) = \sum_{i=1}^{d} \frac{v_i^2}{\lambda - \lambda_i + t} \leq 1. \tag{4.32}$$

Let

$$A_1 := \sum_{i=2}^{d} \frac{v_i^2}{\lambda - \lambda_i}$$

and

$$A_2 := \sum_{i=2}^{d} \frac{v_i^2}{\lambda - \lambda_i + t}$$

Therefore Eq. (4.31) implies that $\frac{v_1^2}{\lambda-\lambda_1} \geq 1 - A_1$ and Eq. (4.32) implies that $\frac{v_1^2}{\lambda-\lambda_1+t} \leq 1-A_2$. Let $L = (\lambda-\lambda_1)(1-A_1)$ and $U = (\lambda-\lambda_1+t)(1-A_2)$. Then it holds that $v_1^2 \in [L, U]$ and $U - L = (A_1 - A_2)(\lambda - \lambda_1) + t(1 - A_2)$.

Our goal will be to show that $U - L \leq t$. We bound $U - L$ as follows:

$$\begin{aligned}
U - L &= t(1 - A_2) + (A_1 - A_2)(\lambda - \lambda_1) \\
&= t(1 - A_2) + (\lambda - \lambda_1) \sum_{i=2}^{d} \frac{v_i^2 t}{(\lambda - \lambda_i)(\lambda - \lambda_i + t)} \\
&\leq t(1 - A_2) + \sum_{i=2}^{d} \frac{v_i^2 t}{\lambda - \lambda_i + t} \\
&\leq t(1 - A_2) + tA_2 = t.
\end{aligned}$$

Thus we have that $\lambda_1(\widetilde{M}) \in [\lambda, \lambda + t]$ implies that $v_1^2 \in [L, L + t]$ where L is a random variable that depends on $v_2, \ldots, v_d$. Since v_1 is independent of $v_2, ..., v_n$ and v_1^2 is a Chi-squared random variable with a single degree of freedom, have that

$$\begin{aligned}
\Pr(v_1^2 \in [L, L + t]) &\leq \sup_{a \in [0,\infty)} \Pr(v_1^2 \in [a, a + t]) \\
&= \sup_{a \in [0,\infty)} \int_a^{a+t} \frac{e^{-t/2}}{\sqrt{2\pi t}} dt \\
&\leq \sup_{a \in [0,\infty)} \frac{1}{\sqrt{2\pi}} \int_a^{a+t} \frac{1}{\sqrt{t}} dt \\
&= \frac{1}{\sqrt{2\pi}} \int_0^t \frac{1}{\sqrt{t}} dt = \sqrt{\frac{2}{\pi} t}.
\end{aligned}$$

□

Lemma 42 (Sylvester's Determinant Identity [85])**.** *For any two vectors*

$u, v \in \mathbb{R}^n$ *it holds that*

$$\det(I + uv^T) = 1 + v^T u.$$

Lemma 43 (Variational Characterization of Eigenvalues (Courant-Fischer))**.** *For any symmetric matrix M, it holds that*

$$\lambda_2(M) = \inf_{u:\, \|u\|=1} \sup_{x \perp u:\, \|x\|=1} x^T M x$$

Proof of Theorem 18

Before proving the theorem we need to prove the following lemma.

Lemma 44. *Assume that on each iteration t it holds that*

$$x_t^\top \tilde{S}_t x_t \geq \lambda_{\max}(\tilde{S}_t) - \epsilon,$$

where $\tilde{S}_t = \sum_{\tau=1}^{t-1} A_\tau + N$ and $\epsilon \leq \frac{2\sqrt{2}}{\pi c}$, for the value of c specified in Theorem 15. Then the statement of Theorem 15 still holds.

Proof. For each iteration t let us denote by $\tilde{x}_t$ an exact leading eigenvector of the matrix $\tilde{S}_t$. In order to derive the lemma, it suffices to show that

$$\mathbb{E}[\sum_{t=1}^{T} \tilde{x}_t^\top A_t \tilde{x}_t - x_t^\top A_t x_t] = O(\sqrt{nT \max\{1, \ln(T/n)\}}),$$

and apply the result of Theorem 15 with respect to the vectors $\tilde{x}_1, ..., \tilde{x}_T$.

Let $\delta_t := \delta(\tilde{S}_t)$. Let us now write $x_t = \alpha \tilde{x}_t + \sqrt{1-\alpha^2} z_t$ for some unit vector $z_t \perp \tilde{x}_t$ and $\alpha \in [-1, 1]$. On the one hand it holds that $x_t^\top \tilde{S}_t x_t \geq \lambda_{\max}(\tilde{S}_t) - \epsilon$. On the other hand it holds that

$$\begin{aligned} x_t^\top \tilde{S}_t x_t &= (\alpha \tilde{x}_t + \sqrt{1-\alpha^2} z_t)^\top \tilde{S}_t (\alpha \tilde{x}_t + \sqrt{1-\alpha^2} z_t) \\ &= \alpha^2 \tilde{x}_t^\top \tilde{S}_t \tilde{x}_t + (1-\alpha^2) z_t^\top \tilde{S}_t z_t \\ &\leq \lambda_{\max}(\tilde{S}_t) - (1-\alpha^2)\delta_t. \end{aligned}$$

Thus we have that

$$\sqrt{1-\alpha^2} \leq \min\{1, \sqrt{\frac{\epsilon}{\delta_t}}\},$$

which means that

$$\begin{aligned}\|\tilde{x}_t\tilde{x}_t^\top - x_t x_t^\top\|_F^2 &= (1-\alpha^2) + 2|\alpha|\sqrt{1-\alpha^2} + (1-\alpha^2)\\ &\le 4\sqrt{1-\alpha^2} \le 4\min\{1, \sqrt{\frac{\epsilon}{\delta_t}}\}\\ &= 4\sqrt{\epsilon}\min\{\epsilon^{-1/2}, \delta_t^{-1/2}\}.\end{aligned}$$

Denote $X_t := 4\sqrt{\epsilon}\min\{\epsilon^{-1/2}, \delta_t^{-1/2}\}$. Going along the same lines as in the proof of Lemma 38, using Theorem 17, we have that

$$\mathbb{E}[X_t] \le 4\sqrt{\epsilon}\int_0^{\epsilon^{-1/2}} \Pr[\delta_t \le 1/t^2]dt \le 4\sqrt{\epsilon}\int_0^{\epsilon^{-1/2}} \min\{\frac{2\sqrt{2}}{\pi c t^2}, 1\}dt,$$

where c is set according to Theorem 15.

For $\epsilon^{-1/2} \ge \sqrt{\frac{2\sqrt{2}}{\pi c}}$, we have that

$$\begin{aligned}\mathbb{E}[X_t] &\le 4\sqrt{\epsilon}\left(\sqrt{\frac{2\sqrt{2}}{\pi c}} + \int_{\sqrt{\frac{2\sqrt{2}}{\pi c}}}^{\epsilon^{-1/2}} \frac{2\sqrt{2}}{\pi c t^2}dt\right)\\ &= 4\sqrt{\epsilon}\left(\sqrt{\frac{2\sqrt{2}}{\pi c}} - \frac{2\sqrt{2\epsilon}}{\pi c} + \frac{2\sqrt{2}}{\pi c}\cdot\sqrt{\frac{\pi c}{2\sqrt{2}}}\right)\\ &< \frac{8\sqrt{2\sqrt{2}\epsilon}}{\sqrt{\pi c}}.\end{aligned}$$

It now follows that,

$$\begin{aligned}\mathbb{E}[\sum_{t=1}^T \tilde{x}_t^\top A_t\tilde{x}_t - x_t^\top A_t x_t] &\le \sum_{t=1}^T \mathbb{E}[\|\tilde{x}_t\tilde{x}_t^\top - x_t x_t^\top\|_*\|A_t\|]\\ &\le \sqrt{2}\sum_{t=1}^T \mathbb{E}[\|\tilde{x}_t\tilde{x}_t^\top - x_t x_t^\top\|_F]\\ &= \sqrt{2}\sum_{t=1}^T \mathbb{E}[X_t] = O\left(T\sqrt{\frac{\epsilon}{c}}\right)\\ &= O\left(\frac{T}{c}\right),\end{aligned}$$

where the first inequality follows from Holder's inequality, the second inequality follows from the connection between the nuclear and frobenius norms and the fact that $\tilde{x}_t\tilde{x}_t^\top - x_t x_t^\top$ is a rank-two matrix, and the last equality follows from plugging the upper bound on ϵ.

The lemma now follows from plugging the value of c stated in Theorem 15. □

We can now prove Theorem 18.

Proof. According to Lemma 44 it suffices to approximate the leading eigenvector of the matrix $\tilde{S}_t$ on each iteration t up to an error of $O(1/c)$, where c is as stated in Theorem 15. Using the Lanczos algorithm, such an approximated eigenvector could be computed in time $\tilde{O}\left(\sqrt{\lambda_{\max}(\tilde{S}_t)c} \cdot \min\{\mathrm{nnz}(\tilde{S}_t), n^2\}\right)$ (see Theorem 4.2 in [60]).

Since $\lambda_{\max}(\tilde{S}_t) = O(T)$ and $c = \tilde{O}(\sqrt{T/n})$ we conclude that each iteration could be carried out in total time of $\tilde{O}(n^{-1/4}T^{3/4}\min\{\mathrm{nnz}, n^2\})$. □

Bibliography

[1] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the newton method. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 9–16, 2006.

[2] S. Damla Ahipasaoglu, Peng Sun, and Michael J. Todd. Linear convergence of a modified Frank-Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software*, 23(1):5–19, 2008.

[3] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semide.nite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 339–348, 2005.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[5] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC*, 2007.

[6] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[7] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97 – 114, 2008. Learning Theory 2004.

[8] Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1359–1366, 2012.

[9] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The fast convergence of incremental PCA. In *27th Annual Conference on Neural Information Processing Systems, NIPS*, 2013.

[10] Amir Beck and Marc Teboulle. A conditional gradient method with linear rate of convergence for solving convex linear systems. *Meth. of OR*, 59(2):235–247, 2004.

[11] Ahron Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. Siam, 2001.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[13] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[14] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[15] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[16] Deeparnab Chakrabarty, Prateek Jain, and Pravesh Kothari. Provable submodular minimization using wolfe's algorithm. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 802–809. Curran Associates, Inc., 2014.

[17] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282, 2011.

[18] Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In *Symposium on Discrete Algorithms (SODA)*, pages 922–931, 2008.

[19] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. *Journal of the ACM*, 59(5):23, 2012.

[20] Thomas M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991.

[21] Alexandre d'Aspremont and Noureddine El Karoui. A stochastic smoothing algorithm for semidefinite programming. *SIAM Journal on Optimization*, 24(3):1138–1177, 2014.

[22] C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation, III. *SIAM J. Numer. Anal.*, 7, March 1970.

[23] Vladimir F. Demyanov and Aleksandr M. Rubinov. *Approximate methods in optimization problems*. Elsevier Publishing Company, 1970.

[24] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[25] John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[26] Miroslav Dudík, Zaïd Harchaoui, and Jérôme Malick. Lifted coordinate descent for learning with trace-norm regularization. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 327–336, 2012.

[27] J.C Dunn. Convergence rates for conditional gradient sequences generated by implicit step length rules. *SIAM Journal on Control and Optimization*, 18:473–487, 1980.

[28] Joseph C. Dunn. Rates of Convergence for Conditional Gradient Algorithms Near Singular and Nonsingular Extremals. *SIAM Journal on Control and Optimization*, 17(2), 1979.

[29] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 11–20. ACM, 2014.

[30] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Symposium on Discrete Algorithms (SODA)*, pages 385–394, 2005.

[31] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:149–154, 1956.

[32] S. Fujishige. *Submodular Functions and Optimization: Second Edition.* Annals of Discrete Mathematics. Elsevier Science, 2005.

[33] Dan Garber and Elad Hazan. Approximating semidefinite programs in sublinear time. In *25th Annual Conference on Neural Information Processing Systems 201*, pages 1080–1088, 2011.

[34] Dan Garber and Elad Hazan. A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization. *CoRR*, abs/1301.4666, 2013.

[35] Dan Garber and Elad Hazan. Playing non-linear games with linear oracles. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2013.

[36] Dan Garber and Elad Hazan. Fast and simple PCA via convex optimization. *CoRR*, abs/1509.05647, 2015.

[37] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, pages 1–33, 2015.

[38] Michael D. Grigoriadis and Leonid G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Oper. Res. Lett.*, 18(2):53–58, 1995.

[39] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[40] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[41] Jacques GuéLat and Patrice Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1), 1986.

[42] Zaïd Harchaoui, Matthijs Douze, Mattis Paulin, Miroslav Dudík, and Jérôme Malick. Large-scale image classification with trace-norm regularization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3386–3393, 2012.

[43] Zaïd Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Math. Program.*, 152(1-2):75–112, 2015.

[44] Elad Hazan. Approximate convex optimization by online game playing. *CoRR*, abs/cs/0610119, 2006.

[45] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American Theoretical Informatics Symposium (LATIN)*, pages 306–316, 2008.

[46] Elad Hazan. A survey: The convex optimization approach to regret minimization. In *Optimization for Machine Learning*, pages 287–302. MIT Press, 2011.

[47] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[48] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. *Journal of Machine Learning Research - Proceedings Track*, 19:421–436, 2011.

[49] Elad Hazan and Satyen Kale. Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

[50] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Corrigendum to learning rotations with little regret". 2010.

[51] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Learning rotations with little regret. In *23rd Conference on Learning Theory, COLT*, 2010.

[52] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning (ICML)*, 2013.

[53] Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *International Conference on Machine Learning (ICML)*, pages 471–478, 2010.

[54] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11:517–553, 2010.

[55] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Regularization techniques for learning with matrices. *Journal of Machine Learning Research*, 13:1865–1890, 2012.

[56] Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.

[57] Adam Tauman Kalai and Santosh Vempala. Simulated annealing for convex optimization. *Mathematics of Operations Research*, 31(2):253–266, 2006.

[58] Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In *Conference on Learning Theory (COLT)*, pages 93–105, 2010.

[59] Wojciech Kotlowski and Manfred K Warmuth. Pca with gaussian perturbations. *arXiv preprint arXiv:1506.04855*, 2015.

[60] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalues by the power and lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.*, 13:1094–1122, October 1992.

[61] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for Frank-Wolfe algorithms. *NIPS Workshop on Greedy Algorithms, Frank-Wolfe and Friends*, 2013.

[62] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate Frank-Wolfe optimization for structural svms. In *International Conference on Machine Learning (ICML)*, 2013.

[63] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *CoRR*, abs/1309.5550, 2013.

[64] Sören Laue. A hybrid algorithm for convex semidefinite optimization. In *International Conference on Machine Learning (ICML)*, 2012.

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

[66] Adrian Stephen Lewis. Derivatives of spectral functions. *Mathematics of Operations Research*, 21(3):576–588, 1996.

[67] S Thomas McCormick. Submodular function minimization. *Handbooks in operations research and management science*, 12:321–391, 2005.

[68] Athanasios Migdalas. A regularization of the Frank—Wolfe method and unification of certain nonlinear programming methods. *Mathematical Programming*, 65:331–345, 1994.

[69] A. S. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization*. John Wiley, 1983.

[70] Y. Nesterov. Gradient methods for minimizing composite objective function. *Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain*, 2007.

[71] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[72] Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007.

[73] Yurii Nesterov and Arkadii Nemirovskii. interior point polynomial methods in convex programming: Theory and applications. *SIAM*, 1994.

[74] Jiazhong Nie, Wojciech Kotlowski, and Manfred K. Warmuth. Online PCA with optimal regrets. In *24th International Conference on Algorithmic Learning Theory, ALT*, 2013.

[75] Benjamin Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011.

[76] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[77] Thomas Rothvoß. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 263–272. ACM, 2014.

[78] Takumi Hayashi Satoru Fujishige and Sigueo Isotani. The minimum-norm-point algorithm applied to submodular function minimization and linear programming. *Preprint*, 2006.

[79] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[80] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[81] Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. Large-scale convex minimization with a low-rank constraint. In *International Conference on Machine Learning (ICML)*, pages 329–336, 2011.

[82] Ohad Shamir. A stochastic PCA algorithm with an exponential convergence rate. *CoRR*, abs/1409.2848, 2014.

[83] N. Z. Shor, Krzysztof C. Kiwiel, and Andrzej Ruszcayǹski. *Minimization methods for non-differentiable functions.* Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[84] Shay Sahlev Shwartz. Phd thesis. 2007.

[85] J.J. Sylvester. On the relation between the minor determinants of linearly equivalent quadratic functions. *Philosophical Magazine Series 4*, 1(4):295–305, 1851.

[86] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

[87] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

[88] László A Végh. Strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 27–40. ACM, 2012.

[89] G. Wahba. A least squares estimate of satellite attitude. *SIAM Rev.*, 7(3), 1965.

[90] Manfred K. Warmuth and Dima Kuzmin. Online variance minimization. In *19th Annual Conference on Learning Theory, COLT*, 2006.

[91] Manfred K. Warmuth and Dima Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, NIPS*, 2006.

[92] Philip Wolfe. Convergence theory in nonlinear programming. *in: J.Abadie, ed., Integer and Nonlinear Progrmming (North-Holland Publishing Company)*, 1970.

[93] Philip Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

[94] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, pages 928–936, 2003.