UNIVERSITY OF CALIFORNIA
SANTA CRUZ

OPTIMAL ONLINE LEARNING WITH MATRIX PARAMETERS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Jiazhong Nie

September 2015

The Dissertation of Jiazhong Nie
is approved:

Professor Manfred K. Warmuth, Chair

Professor S.V.N. Vishwanathan

Professor Wojciech Kotłowski
Poznań University of Technology, Poland

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Optimal Online Learning with Matrix Parameters

by

Jiazhong Nie

This thesis considers two online matrix learning problems: the problem of online Principle Component Analysis (PCA) and the problem of learning rotations online. Previous papers proposed two online algorithms for these problems, which are the Matrix Exponentiated Gradient (MEG) algorithm and the Gradient Descent (GD) algorithm, respectively [WK08, HKW10a]. This thesis evaluates these two algorithms by their regret, which is the additional total loss of the online algorithm over the total loss of the best offline comparator (chosen in hindsight).

In Chapter 2, we show that for online PCA, both algorithms achieve essentially the same and optimal (within a constant factor) regret bound when the bound is expressed as a function of the number of trials. However, when considering regret bounds as functions of the loss of the best comparator, MEG remains optimal and strictly outperforms GD. Chapter 3 considers a generalization of online PCA in which we use the combination of the compression gain of PCA and the cosine similarity to measure the closeness between two directions (i.e. unit length vectors). Such a combined measurement involves both the first and second moments of the learner's randomized prediction, and therefore, we propose an online algorithm that maintains both a vector and a matrix as its parameter simultaneously. In particular, in each trial of the game, this algorithm uses a novel iterative procedure to decompose its parameter pair into a mixture of at most n pure directions. Chapter 4 considers the problem of learning rotations online. We show that for this problem, the GD algorithm is optimal (up to a constant) for both the regret bounds that are functions of the number of trials and the regret bounds that are functions of the loss of best comparator.

Acknowledgments

My most gratitude goes to my advisor, Manfred Warmuth, who provided me both mentorship and friendship. He offered me insights and inspiration that allowed me to grow as a theoretical researcher in online learning. I benefit immensely from his curiosity, open mind and rigorous attitude. This work is heavily influenced by his previous work and this dissertation will not be completed without his tireless editing and guidance. I cannot thank him enough.

I am lucky to work with many excellent professors and scholars, Professor David Helmbold, Professor Wojciech Kotłowski, Dr. Wouter M. Koolen and Professor S.V.N. Vishwanathan. They provided insightful intellectual simulation through discussions and brainstorming sessions. I thank them for their help, feedback and advice regarding my research. You will be my guiding examples in my future career.

I gained a huge amount of help from the professors and colleagues from the computer science department at UCSC. I specially thank Professor Yi Zhang for mentoring me for two years and serving as my committee member in my advancement. Special thanks to the students from machine learning group: Pei Jiang, Holakou Rahmanian, Michał Dereziński. They provided valuable feedback on my work. I also would like to thank Tracie Tucker for her patience with all my administrative problems.

Finally, I would like to thank my family for their support. My parents offered me selfless love in these many years. I feel so happy to share this achievement with them. I greatly thank my beautiful and smart wife, Dr. Xiao, who understands me and supports me all the time. Thank my son, Oscar, for bringing so much joy to my life.

Chapter 1

Introduction

This thesis presents some significant new results in the online learning subfield of Machine Learning. Online learning distinguishes itself from the common batch learning by its interactive learning protocol and its adversarial data assumption [Lit87, LW94, Vov90, CBFH$^+$97, CBL06]. In this thesis, we consider online learning with matrix parameters. In particular, we are interested in the online version of two classical machine learning problems: $\underline{\text{P}}$rinciple $\underline{\text{C}}$omponent $\underline{\text{A}}$nalysis (PCA) and learning rotation matrices [WK08, HKW10b]. Previous papers proposed two online algorithms for these problems, which are the Matrix Exponentiated Gradient (MEG) algorithm and the Gradient Descent (GD) algorithm, respectively [WK08, HKW10a]. Each of these algorithms aims at suffering small regret, that is the maximum extra loss the algorithm could suffer against the best comparator chosen in hindsight. In this thesis, we evaluate these two algorithms by upper and lower bounds on their regrets. For each problem setting, an "optimal" algorithm is given, which is always a version of GD or MEG. Here optimal means having regret no more than a constant factor times a regret lower bound for the problem.

We start in this chapter with an introduction to online learning as well as an overview of the problems discussed in the thesis. This chapter itself is organized as follows: Section 1.1 explains basic concepts of online learning, such as, the interactive learning protocol, the adversarial data setting and worst case regret bounds. These concepts are further demonstrated in Section 1.1.1 with a basic online learning problem, which is the so called expert setting in the online learning literature. In Section 1.2, we introduce the two learning problems considered in this thesis, which are the problem of online Principle Component Analysis (PCA) and the problem of learning rotations online. Section 1.3 describes the mirror descent algorithm framework, which, when configured with different Bregman divergences, specializes to the MEG and GD algorithms discussed in the thesis. Finally, in Section 1.4, we give an overview of the main chapters of the thesis (Chapter 2, 3 and 4).

1.1 Basic Concepts of Online Learning

In a common batch learning problem, the learning algorithm receives a batch of training examples (labeled instances) as input to find a single hypothesis that generalizes well on the testing examples. An online learning problem, on the other hand, uses an interactive protocol that only processes one example in each trial. In particular, in each trial, the algorithm first commits to some hypothesis, usually by choosing a parameter from a fixed parameter set, then it receives an instance and predicts the label of that instance. After that, the true label is revealed and some loss is incurred, which measures the discrepancy between the true and the predicted label. The performance of an algorithm is measured by the difference between the total loss of the algorithm and the total loss of the best offline comparator which knows the entire example sequence in advance. This difference is called the algorithm's *regret* and our goal is to design efficient algorithms whose regret is guaranteed to be small.

The core assumption of the online learning is the *adversarial* data setting, which means the algorithms are required to perform well against *any* example sequence, possibly generated by an adversary. Therefore, we need an upper bound on the largest regret any sequence can produce for the algorithm. We call such bounds *worst case* regret upper bounds. These bounds are further categorized into two types: the *time dependent* bounds, which are functions of the number of the trials (i.e. length of the example sequence) [Zin03, SST11], and the *loss dependent* bounds which are functions of the loss of the best comparator on the example sequence [LW94, CBLW96, CBFH$^+$97, KW97]. Although most previous work focused on time dependent bounds, in this thesis, we prove both types of bounds for our learning algorithms, and show that in many cases, they lead to significantly different conclusion about which algorithm is optimal.

The optimality of algorithms is defined in terms of their worst case regret bound, that is, given any learning problem, we are interested in algorithms that have the least worst case regret bound among all the algorithms following the problem's protocol. To this end, we lower bound the regret any algorithm will suffer from an adversarial example sequence and call such a bound the regret lower bound of the problem. If an algorithm suffers worst case regret no more than a constant factor times the lower bound, then we say the algorithm is *optimal* for the problem. Notice that both the discussion of optimal algorithms and the regret lower bounds depend on the parameterization of the bounds, i.e. if they are time dependent bounds or the loss dependent bounds.

1.1.1 A Basic Online Learning Problem: the Expert Setting

Now we describe in detail a basic online learning problem, the so called expert setting. The problem is generalized in various ways later in this thesis. Assume that we need to solve a problem in repeated trials with the help of n experts. At the beginning of each trial, the experts propose their solutions to the problem, and a learning algorithm chooses one of them to follow. After that, each expert incurs a loss, which depends on the quality of his solution, and we suffer the loss of the chosen expert. Notice that although we solve a problem repeatedly, the quality of each expert's solution may vary from trial to trial, e.g., the accuracy of TV weather predictions may change across seasons. Also notice that the expert setting is decision-theoretic – a learning algorithm chooses experts not by evaluating the proposed solutions directly but by its confidence in the experts, which is derived from the losses revealed throughout trials. An algorithm's confidence in the experts at trial t is specified by an n dimensional probability vector $\boldsymbol{w}_t$: expert i is chosen with probability $w_{t,i}$. We denote the losses of n experts at trial t by another n-dimensional vector $\boldsymbol{\ell}_t \in [0,1]^n$. Then, the expected loss of the algorithm is

$\boldsymbol{w}_t \cdot \boldsymbol{\ell}_t$, the inner product between $\boldsymbol{w}_t$ and $\boldsymbol{\ell}_t$[1]. For a problem of T trials, the algorithm's total loss is $\sum_{t=1}^{T} \boldsymbol{\ell}_t \cdot \boldsymbol{w}_t$ and when using the best expert in hindsight as the comparator, the algorithm's regret is defined as

$$\text{REG} = \sum_{t=1}^{T} (\boldsymbol{\ell}_t \cdot \boldsymbol{w}_t) - \underbrace{\min_{1 \le i \le n} \sum_{t=1}^{T} l_{t,i}}_{=\mathcal{L}_c} . \tag{1.1}$$

If all the experts receive losses between zero and one, then the well known Hedge algorithm achieves a loss dependent regret bound of $\text{REG} \le \sqrt{2\mathcal{L}_c \ln n} + \ln n$, where $\mathcal{L}_c$ is the loss of comparator, i.e. the best expert in hindsight (see definition in (1.1)) [FS95][2]. In this thesis, we prove for the expert setting a regret lower bound of $\Omega(\sqrt{2\ln(n)\mathcal{L}_c} + \ln(n))$[3]. This shows that for the expert setting, the Hedge algorithm is optimal up to a constant factor.

1.2 Learning Matrix Parameters

In most common machine learning problems, batch or online, an algorithm's decision/hypothesis is parameterized by a real parameter vector. For example, in the expert setting, an algorithm's confidence in the experts is parameterized by a probability vector whereas in the linear classification, a separating plane is parameterized by its normal vector. This thesis considers two learning problems in which hypotheses are naturally parameterized by matrices. First, the online PCA problem concerns learning

[1]Note that in the expert setting, the loss vectors can be viewed as non-negative instance vectors with a fixed yet hidden label 0. For any weight vector $\boldsymbol{w}$, the loss between the prediction $\boldsymbol{w} \cdot \boldsymbol{\ell}$ and the hidden label is $|0 - \boldsymbol{w} \cdot \boldsymbol{\ell}| = \boldsymbol{w} \cdot \boldsymbol{\ell}$.

[2]This requires the learning rate of the algorithm to be tuned as a function of $\mathcal{L}_c$

[3]An asymptotic version of the lower bound, which only holds in the limit $(T, n \to \infty)$, was previously proved in [CBFH$^+$97].

$n \times n$ projection matrices of rank $k \leq n$ and the corresponding parameter set is the convex hull of these projection matrices, which is the set of symmetric matrices with eigenvalues in $[0, 1]$ and summing to k; Second, the rotation learning problem concerns learning $n \times n$ rotation matrices and the corresponding parameter set is convex hull of rotation matrices, which is the set of matrices with singular values not larger than one. Note that these matrix learning problems cannot be simply reduced to learning vectors of length $n \times n$, since their eigenvalue/singular value structure is lost in such a vectorized form, and more importantly, because learning these matrices is much *easier* than learning a vector of length $n \times n$. In fact, consider the special cases where all the data matrices are diagonal. In these cases, since all the data matrices have a fixed eigensystem/singular system, the algorithm only learns n eigenvalues. Nevertheless, [WK08] and [HKW10a] prove for the matrix learning problems the same regret upper bounds as those proved for the diagonal cases. Moreover, we show in this thesis that these regret upper bounds are also matched in a constant factor by the *lower bounds* proved for the diagonal cases, which means that the diagonal cases are as hard as these matrix learning problems up to a constant factor.

1.3 Learning Algorithms

Most of the learning algorithms discussed in this thesis are instances of the Mirror Descent algorithm framework[NY78, KW97]. Under this framework, a learning algorithm is motivated by a Bregman divergence and updates its parameters from trial to trial as the following: the parameter used for next trial is obtained by minimizing a trade-off between the motivating Bregman divergence to the old parameter and the loss on the current example. The intuition is that we want to stay somewhat close to the old parameter, since it summarizes our knowledge attained so far, but we also want to learn something new by improving the performance on the current example[KW97].

Different motivating Bregman divergences lead to different families of learning algorithms. We discuss in this thesis two families of algorithms, the Exponentiated Gradient (EG) family and the Gradient Descent (GD) family. When learning vector parameters, the GD family is motivated by squared Euclidean distance, which leads to an additive update: the parameter is updated by subtracting from it a scaled version of the loss function's gradient. On the other hand, the EG family for the vector case is motivated by relative entropy, which leads to a multiplicative update: the parameter is updated by multiplying its i-th element w_i with $e^{-\eta l_i}$, where l_i is loss function's derivative w.r.t. w_i and η is a learning rate that scales the derivatives.

When learning matrix parameters, the EG family is motivated by the quantum relative entropy and the resulted algorithm is referred as the Matrix Exponentiated Gradient (MEG) algorithm, while the GD family is motivated by squared Frobenius norm and the resulted algorithm is still referred the GD algorithm, since the Frobenius norm of a matrix equals to the Euclidean norm of the vectorized matrix. In both vector and matrix versions of the GD algorithm, the parameters are updated by subtracting from them the scaled gradients of the loss function, where the positive learning rate is the scaling factor.

One central topic of this thesis is to compare the regret bound of MEG and GD for the two matrix learning problems we considered, and to find for each problem an algorithm whose regret bound is optimal (within a constant factor). We show in the following chapters that such a comparison depends on three factors. First, the set of concepts we are learning, second, whether the instance matrices[4] are of rank one or of full rank, and finally whether we are discussing the time dependent bounds or the loss dependent bounds. Nevertheless, we are able to show that for the problem of online PCA, MEG is the optimal algorithm in all cases and in some cases is strictly better than the GD algorithm. For the problem of learning rotations online, the GD algorithm is optimal[5].

1.4 Overview of the Chapters

We now sketch the contents of the main chapters of this dissertation. Chapter 2 and Chapter 3 have been published at COLT and ALT conferences [NKW13, KNW13], respectively. Chapter 4 is based on the unpublished manuscript [Nie14].

1.4.1 Chapter 2: Online PCA with Optimal Regrets

In Principal Component Analysis (PCA), the data points $\boldsymbol{x}_t \in \mathrm{R}^n$ are projected / compressed onto a k-dimensional subspace. Such a subspace can be represented by its projection matrix $\boldsymbol{P}$ which is a symmetric matrix in $\mathrm{R}^{n\times n}$ with k eigenvalues equal

[4]Note that in a matrix learning problem, both the parameters and the instances are matrices.

[5]Since quantum relative entropy is a function of positive semi-definite matrices, MEG is not directly applicable to the rotation learning problem. For partial results, see [Aro09].

1 and $n-k$ eigenvalues equal 0. The goal of PCA is to minimize the total *compression loss* $\sum_t \|\boldsymbol{P}\boldsymbol{x}_t - \boldsymbol{x}_t\|^2$, which is solved by the eigenvectors of the k largest eigenvalues of the covariance matrix $\boldsymbol{C} = \sum_t \boldsymbol{x}_t\boldsymbol{x}_t^\intercal$.

In this chapter, we consider the online version of PCA [WK08], where in each trial $t = 1, \ldots, T$, the algorithm chooses (based on the previously observed points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$) a randomized subspace of dimension k, which is described by a randomized projection matrix $\boldsymbol{P}_t$ of rank k. Then a next point $\boldsymbol{x}_t$ is revealed and the algorithm incurs the expected *compression loss*: $\mathrm{E}[\|\boldsymbol{x}_t - \boldsymbol{P}_t\boldsymbol{x}_t\|_2^2]$. The regret of an algorithm is the difference between the its cumulative loss and the loss of the off-line PCA solution to data points $\boldsymbol{x}_1 \ldots \boldsymbol{x}_T$.

We evaluate the GD and MEG algorithms for online PCA in terms of their worst-case regret bounds. We show that for the time dependent regret bounds, both algorithms are essentially optimal. This is surprising because MEG is believed to perform sub-optimally when the instances are sparse. However, in the case of online PCA, the loss function is linear with a non-negative gradient, because when viewed in the right way, the compression loss is a dot product between the complementary projection matrix $\boldsymbol{I} - \boldsymbol{P}_t$ and an instance matrix $\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$ [WK08]. Furthermore, we show that when considering loss dependent regret bounds which are functions of the loss of comparator, then MEG remains optimal and strictly outperforms GD.

Next, we study a generalization of the online PCA problem, in which Nature is allowed to present the algorithm with dense instance matrices (i.e. positive semi-definite matrices with bounded largest eigenvalue). Again we can show that MEG is optimal and strictly better than GD in this setting.

1.4.2 Chapter 3: Learning a Set of Directions

The previous chapter discussed online PCA in which data instances are compressed by a k-dimensional subspace minimizing the compression loss, that is, the sum of the squared distances between the data instances and their projections. Alternatively, this subspace can be identified by maximizing the compression gain, which is the sum of the squared length of the projected data instances. In this chapter, we combine the compression gain with the cosine similarity and use the combined measure to measure the closeness between directions, i.e. vectors of length 1. We refer this combined measure as the *directional gain*. Formally, this chapter considers the following online problem of learning a set of directions: in each trial the learner predicts a set of k randomized directions $\boldsymbol{u}_1 \dots \boldsymbol{u}_k$ which are orthogonal to each other and Nature responds with an instance direction $\boldsymbol{x}$, then the learner receives the expected directional gain, which is the expectation of $(\sum_{i=1}^k \boldsymbol{u}_i^\intercal \boldsymbol{x} + c)^2$. Since $\boldsymbol{u}_1 \dots \boldsymbol{u}_k$ are orthogonal to each other, the directional gain can be expanded as the following:

$$(\sum_{i=1}^k \boldsymbol{u}_i^\intercal \boldsymbol{x} + c)^2 = \boldsymbol{x}^\intercal \sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{u}_i^\intercal \boldsymbol{x} + 2c \sum_{i=1}^k \boldsymbol{u}_i^\intercal \boldsymbol{x} + c^2. \tag{1.2}$$

Hence, the gain is in fact a trade-off between $\boldsymbol{x}^\intercal \sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{u}_i^\intercal \boldsymbol{x}$, which is the compression gain of the projection matrix $\sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{u}_i^\intercal$ w.r.t. instance $\boldsymbol{x}$, and the cosine similarity $\sum_{i=1}^k \boldsymbol{u}_i^\intercal \boldsymbol{x} = \sum_{i=1}^k \cos(\boldsymbol{u}_i, \boldsymbol{x})$.

Taking expectation of (1.2) w.r.t. learner's randomized prediction $\boldsymbol{u}_1 \dots \boldsymbol{u}_k$, one would see that the expected directional gain depends on both the first and the second moments of the randomized prediction ($\mathrm{E}[\sum_{i=1}^{k} \boldsymbol{u}_i]$ and $\mathrm{E}[\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\intercal]$), whereas in online PCA, only the second order moment is involved. This fact poses two challenges to a learning algorithm: First, the algorithm needs to maintain uncertainty via both the first and the second moments of its prediction simultaneously, and second, when the algorithm predicts, such uncertainty needs to be efficiently decomposed into a convex combination of direction sets.

The first challenge is addressed by maintaining a parameter pair $(\boldsymbol{\mu}, \boldsymbol{D})$, where vector $\boldsymbol{\mu}$ and matrix $\boldsymbol{D}$ represent the first and the second moments of the algorithms prediction, respectively. Moreover, we constraint $(\boldsymbol{\mu}, \boldsymbol{D})$ with the condition $\boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D} \preceq \boldsymbol{I}$. We will show that such a condition is necessary and sufficient for $(\boldsymbol{\mu}, \boldsymbol{D})$ to be the first and second moments of a single distribution on orthogonal direction sets of size k. For the second challenge, we propose an efficient procedure which decomposes the parameter pair $(\boldsymbol{\mu}, \boldsymbol{D})$ into a mixture of at most n sets of pure directions. This procedure consists of at most n iterative steps: Each of these steps reduces the rank of parameter matrix $\boldsymbol{D}$ by removing from the parameter pair $(\boldsymbol{\mu}, \boldsymbol{D})$ a set of pure directions $\boldsymbol{u}_1 \ldots \boldsymbol{u}_k$ with a mixing coefficient p. Unlike the decomposition procedure of online PCA which always decomposes with parameter matrix's eigendirections, our procedure chooses $\boldsymbol{u}_1 \ldots \boldsymbol{u}_k$ and p in a special way such that after the removal, the new parameter pair $\boldsymbol{\mu} - p\sum_{i=1}^{k} \boldsymbol{u}_i$ and $\boldsymbol{D} - p\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\intercal$ still satisfy the aforementioned necessary and sufficient condition.

1.4.3 Chapter 4: Learning Rotations with Optimal Regret

In this chapter, we consider the problem of learning rotations online [HKW10b]. In each trial of this problem, the learning algorithm is first given a unit vector $\boldsymbol{x}_t$, then it predicts (deterministically or randomly) with a unit vector $\widehat{\boldsymbol{y}}_t$. Finally, Nature reveals true rotated vector $\boldsymbol{y}_t$, which is also a unit vector. The loss to the algorithm is half the expected squared norm of the difference between her predicted vector $\widehat{\boldsymbol{y}}_t$ and the true rotated vector $\boldsymbol{y}_t$: $\mathrm{E}\left[\,\frac{1}{2}\|\boldsymbol{y}_t - \widehat{\boldsymbol{y}}_t\|^2\,\right] = 1 - \boldsymbol{y}_t^\intercal \mathrm{E}\left[\,\widehat{\boldsymbol{y}}_t\,\right]$. We define the offline comparator of the problem as the rotation matrix $\boldsymbol{R}_c$ that minimizes its loss on data points $(\boldsymbol{x}_1, \boldsymbol{y}_1) \dots (\boldsymbol{x}_T, \boldsymbol{y}_T)$ in hindsight, i.e.,

$$\boldsymbol{R}_c = \min_{\substack{\boldsymbol{R} \text{ is a rotation} \\ \text{matrix of dimension } n}} \sum_{t=1}^{T} \|\boldsymbol{x}_t - \boldsymbol{R}\boldsymbol{y}_t\|_2^2.$$

The regret of an algorithm is then the difference between its cumulative loss and the loss of the comparator $\boldsymbol{R}_c$. [HKW10a] applied the GD algorithm to this problem and showed that when considering the time dependent regret bounds, the algorithm is optimal up to a constant factor. However, for the loss dependent bounds, there has not been any result on upper bounding the regret. In this chapter, we prove a loss dependent upper bound on the regret of the GD algorithm by introducing two new techniques to the its analysis: First, a refined application of Pythagorean Theorem and second, a new linear term is added to our algorithm's measure of progress. Note that the loss dependent regret bound obtained for GD is also optimal up to a constant factor in contrast to the case of online PCA where GD is only optimal for time dependent bounds and is sub-optimal for loss dependent bounds.

Chapter 2

Online PCA with Optimal Regrets

2.1 Introduction

In $\underline{\text{P}}$rincipal $\underline{\text{C}}$omponent $\underline{\text{A}}$nalysis (PCA), the data points $\boldsymbol{x}_t \in \mathrm{R}^n$ are projected / compressed onto a k-dimensional subspace. Such a subspace can be represented by its projection matrix $\boldsymbol{P}$ which is a symmetric matrix in $\mathrm{R}^{n\times n}$ with k eigenvalues equal 1 and $n-k$ eigenvalues equal 0. The goal of *uncentered PCA* is to find the rank k projection matrix that minimizes the total *compression loss* $\sum_t \|\boldsymbol{P}\boldsymbol{x}_t - \boldsymbol{x}_t\|^2$, i.e. the sum of the squared Euclidean distances between the original and the projected data points.[1] Surprisingly, this loss can be written as a linear loss [WK08]:

$$\sum_t \|\boldsymbol{P}\boldsymbol{x}_t - \boldsymbol{x}_t\|^2 \;=\; \sum_t \|(\boldsymbol{P}-\boldsymbol{I})\boldsymbol{x}_t\|^2 \;=\; \sum_t \boldsymbol{x}_t^\intercal(\boldsymbol{I}-\boldsymbol{P})^{\cancel{2}}\boldsymbol{x}_t \;=\; \operatorname{tr}\big((\boldsymbol{I}-\boldsymbol{P})\underbrace{\sum_t \boldsymbol{x}_t\boldsymbol{x}_t^\intercal}_{\boldsymbol{C}}\big),$$

[1]In *centered PCA* the goal is to minimize $\sum_t \|\boldsymbol{P}(\boldsymbol{x}_t-\boldsymbol{\mu})-(\boldsymbol{x}_t-\boldsymbol{\mu})\|^2$ where $\boldsymbol{P}$ is a projection matrix of rank k and $\boldsymbol{\mu}\in\mathrm{R}^n$ is a second mean parameter. For the sake of simplicity we focus on the optimal algorithms for uncentered PCA. However we believe that our results will essentially carry over to the centered case (as was already partially done in [WK08]).

where in the 3rd equality we used the fact that $\boldsymbol{I}-\boldsymbol{P}$ is a projection matrix and therefore $(\boldsymbol{I}-\boldsymbol{P})^2 = \boldsymbol{I}-\boldsymbol{P}$. The final expression of the compression loss is linear in the projection matrix $\boldsymbol{P}-\boldsymbol{I}$ as well as the covariance matrix $\boldsymbol{C} = \sum_t \boldsymbol{x}_t\boldsymbol{x}_t^\intercal$ which is the sum of the outer products. The crucial point to note here is that the compression loss is *linear* in the outer products $\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$ but not in the original points $\boldsymbol{x}_t$.

The batch version of uncentered PCA is equivalent to finding the eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ belonging to the k largest eigenvalues of the covariance matrix $\boldsymbol{C}$: if $\boldsymbol{P} = \sum_{i=1}^k \boldsymbol{u}_i\boldsymbol{u}_i^\intercal$ is the k dimensional projection matrix formed from these k eigenvectors, then $\boldsymbol{I}-\boldsymbol{P}$ is the complimentary $n-k$ dimensional projection matrix minimizing the linear loss $\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P})\boldsymbol{C})$.

In this chapter we consider the online version of uncentered PCA [WK08], where in each trial $t = 1, \ldots, T$, the algorithm chooses (based on the previously observed points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$) a subspace of dimension k described by its projection matrix $\boldsymbol{P}_t$ of rank k. Then a next point $\boldsymbol{x}_t$ (or instance matrix $\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$) is revealed and the algorithm suffers the *compression loss*:

$$\|\boldsymbol{x}_t - \boldsymbol{P}_t\boldsymbol{x}_t\|^2 \quad = \quad \operatorname{tr}\big((\boldsymbol{I}-\boldsymbol{P}_t)\,\boldsymbol{x}_t\boldsymbol{x}_t^\intercal\big). \tag{2.1}$$

The goal here is to obtain an online algorithm whose cumulative loss over trials $t = 1, \ldots, T$ is close to the cumulative loss of the best rank k projection matrix chosen in hindsight after seeing all T instances. The maximum difference between the cumulative loss of the algorithm and the best off-line comparator is called the (worst-case) *regret.* This regret naturally scales with the maximum square 2-norm of the data points $\boldsymbol{x}_t$. For the sake of simplicity we assume that all points have 2-norm equal 1. In the chapter we find the optimal algorithm for online PCA (and some generalizations), where optimal here means that the upper bounds we prove for the regret of the algorithm is at most a constant factor larger the lower bound we can prove for the learning problem.

There are two main families of algorithms in online learning, which differ in how the parameter vector/matrix is updated: the Gradient Descent (GD) family [CBLW96, Zin03] and the Exponentiated Gradient (EG) family [KW97]. Both are motivated by trading off a divergence against the loss. The GD family is motivated by the squared Euclidean distance divergence, and the Exponentiated Gradient (EG) family by the the relative entropy divergence [KW97]. The first family leads to *additive updates* of the parameter vector/matrix. When there are no constraints on the parameter space, then the parameter vector/matrix of the GD family is a linear combination of the instances. However when there are constraints, then after the update the parameter is projected onto the constraints (by a Bregman projection with respect to the squared Euclidean distance). The second family leads to *multiplicative update* algorithms. For that family, the components of the parameter are non-negative and if the parameter space consists of probability vectors, then the non-negativity is already enforced by the relative entropy divergence and less projections are needed.

What is the best parameter space for uncentered PCA? The compression loss (2.1) is linear in the projection matrix matrix $\boldsymbol{I} - \boldsymbol{P}_t$ which is of rank $n-k$. An online algorithm has uncertainty over the best projection matrix. Therefore the parameter matrix $\boldsymbol{W}_t$ of the algorithm is a mixture of such matrices [WK08] which must be a positive semi-definite matrix of trace $n-k$ whose eigenvalues are capped at 1. The algorithm chooses its projection matrix $\boldsymbol{I}-\boldsymbol{P}_t$ by sampling from this mixture $\boldsymbol{W}_t$, i.e. $\mathrm{E}[\boldsymbol{I}-\boldsymbol{P}_t]=\boldsymbol{W}_t$. The loss of the algorithm is $\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\,\boldsymbol{x}_t\boldsymbol{x}_t^\intercal)$ and its expected loss $\operatorname{tr}(\boldsymbol{W}_t\,\boldsymbol{x}_t\boldsymbol{x}_t^\intercal)$.

In [WK08], a matrix version of the multiplicative update was applied to PCA, whose regret bound is logarithmic in the dimension n. This algorithm uses the quantum relative entropy in its motivation and is called the *Matrix Exponentiated Gradient* (MEG) algorithm [TRW05]. It does a matrix version of a multiplicative update and then projects onto the "trace equal $n-k$" and the "capping" constraints (Here the projections are with respect to the quantum relative entropy).

For the PCA problem the (expected) loss of the algorithm at trial t is $\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{x}_t\boldsymbol{x}_t^\intercal)$. Consider the generalization to the loss $\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$ where now $\boldsymbol{X}_t$ is any positive semi-definite symmetric instance matrix and the parameter $\boldsymbol{W}_t$ is still a convex combination of rank $n-k$ density matrices. In PCA the instance matrices are the outer products, i.e. $\boldsymbol{X}_t=\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$. Such instances (also called *dyads*) are *s*parse in the sense that their trace norm is one, independent of the dimension n of the instance matrix. Beginning with some of the early work on linear regression [KW97], it is known that multiplicative updates are especially useful when the instances are allowed to be *dense*. In the matrix context this means that the symmetric positive semi-definite instance matrices $\boldsymbol{X}_t$ have maximum eigenvalue (i.e. spectral norm) of say one. Thus for PCA, one may suspect that MEG is not able to fully exploit the sparsity of the instance matrices. On the other hand for linear regression, GD is known to have the advantage when the instances are sparse [KW97] and consistently with that, when GD is used for PCA, then its regret is bounded by a term that is *independent* of the dimension of the instances. The advantage of GD in the sparse case is also supported by a general survey of Mirror Descent algorithms (to which GD and MEG belong) for the case when the gradients of the convex loss functions (which may have negative components) lie in certain symmetric norm balls [SST11]. Again when the gradient vectors of the losses are sparse then GD has the advantage.

Surprisingly, the situation is quite different for PCA: We show that MEG achieves the same regret bound as GD for online PCA (despite the sparseness of the instance matrices) and the regret bounds for both algorithms are within a constant factor of a new lower bound proved in this chapter that holds for any online PCA algorithm. This surprising performance of MEG seems to come from the fact that gradients of the losses in the PCA case are restricted to be non-negative. Therefore our results are qualitatively different from the cases studied in [SST11] where the gradients of the loss functions are within a $p-$norm ball, i.e. symmetric around zero.

Actually, there are two kinds of regret bounds in the literature: bounds expressed as a function of the time horizon T and bounds that depend on an *upper bound* on the loss of the best comparator (which we call a *loss budget* following [AWY08]). In typical applications for PCA, there exists a low dimensional subspace which captures most of the variance in the data and the compression loss is small. Therefore, guarding against the worst-case loss that grows with the number of trials T is overly pessimistic. We can show that when considering regret bounds as a function of a loss budget, MEG is optimal and strictly better than GD by a factor of $\sqrt{k}$. This suggests that the multiplicative updates algorithm is the best choice for prediction problems in which the parameters are mixture of projection matrices and the gradients of the losses are non-negative. Note that in this chapter we call an algorithm *optimal* for a particular problem if we can prove an upper bound on its worst-case regret that is within a constant factor of the lower bound for the problem (which must holds for any algorithm).

Related Work and Our Contribution:

The comparison of the GD and MEG algorithms has an extensively history (see, e.g. [KW97, WV05, ST10, SST11]). It is simplest to compare algorithms in the case when the loss is linear. Linear losses are the least convex losses and in the regret bounds, convex losses are often approximated by first-order Taylor approximations which are linear, and the gradient of the loss functions as the "loss/gain vector" [KW97, Zin03]. In this case it is often assumed that the gradient of the loss lies in an L_p ball (which is a symmetric constraint) and the results are as expected: EG is optimal when the parameter space is 1-norm bounded and the gradient vectors are infinity norm bounded, and GD is optimal when the both spaces are 2-norm bounded [ST10, SST11].

In contrast for PCA, the gradient of the loss $\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$ is the instance matrix $\boldsymbol{X}_t$ which is assumed to be positive semi-definite. None of the previous work exploits this special property of the PCA setup, where the gradient of the loss satisfies some non-negativity property. In this chapter we carefully study this case and show that MEG is optimal.

We also made significant technical progress on the lower bounds for online PCA. The previous lower bounds ([WK08] and [KWK10]) were incomplete in the following three ways: First, the lower bounds require dense instance matrices, whereas for standard PCA, the instance matrices (the outer products $\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$) are sparse. Second, the previous lower bounds assume that the dimension k of target subspace is at least $\frac{n}{2}$ and in common PCA problems k is much smaller than $\frac{n}{2}$. Third, the proofs rely on the Central Limit Theorem and therefore the resulting lower bounds only hold in the limit as T and n go to infinity (See [CBFH+97, CBL06, AABR09] for details). In this chapter, we circumvent all three weak aspects of the previous proofs: We give lower bounds for all four combinations of sparse or dense instance matrices versus $k \leq \frac{n}{2}$ or $k \geq \frac{n}{2}$, respectively. All our lower bounds are non-asymptotic i.e. they hold for all values of the variables T and n. The new lower bounds use a novel probabilistic bounding argument for the minimum of n random variables. Alternate methods for obtaining non-asymptotic lower bound for label efficient learning problems in the expert setting were given in [AB10]. However those techniques are more complicated and it is not clear how to adapt them to the online PCA problem.

In summary, our contribution consists of proving tight upper bounds on the regret of the two main online PCA algorithms, as well as proving lower bounds on the regret of any algorithm for solving online PCA. For the case when the regret is expressed as a function of the number of trials T, we show that MEG's and GD's regret bounds are independent of the dimension n of the problem and are within a constant factor of the lower bound on the regret of any online PCA algorithm. This means the both algorithms are optimal in this case. For the case when the regret is a function of the loss budget, we prove that MEG remains optimal, while we show that the regret of GD is suboptimal by a $\sqrt{k}$ factor.

Furthermore, for a generalization of the PCA setting to the dense instance case, we improve the known regret bound significantly by switching from a loss version to a gain version of MEG depending on the dimension k of the subspace. If $k \geq \frac{n}{2}$ then the gain version of MEG is optimal in the dense instance case and when $k \leq \frac{n}{2}$ then the loss version is optimal. On the other hand, GD is non-optimal for both ranges of k.

A much shorter preliminary version of this manuscript appeared in the 24th International Conference on Algorithmic Learning Theory (2013) [NKW13]. In this more detailed journal version we give more background and complete proofs of all of our results (mostly omitted or only sketched in the conference version). this chapter also has the following additional material: A proof of the budget bound (2.13) for the gain version of MEG; an extension of the lower bound on the regret of GD (Theorem 2.2) to the case of small budgets; the analysis of the Follow the Regularized Leader variant of GD (Section 2.4.2) and a discussion of its final parameter matrix (Appendix 2.E); lower bounds on the regret when the number of trials is small (Appendix 2.G).

Outline of the Chapter:

In Section 2.2, we start with describing the MEG and GD algorithms for online PCA. In particular, we present two versions of the MEG algorithm: the Loss MEG algorithm introduced in [WK08], and the Gain MEG algorithm, which is the same as

Loss MEG except for a sign change in the exponential. Following the description of each algorithm, we then derive in Section 2.3 their regret bounds expressed as functions of the number of trials T. These bounds are compared in Section 2.3.2 for all four combinations of sparse or dense instance matrices versus $k \leq \frac{n}{2}$ or $k \geq \frac{n}{2}$, respectively (see Table 2.1). Next we consider regret bounds expressed as functions of the loss budget. In Section 2.4, we prove a lower bound on GD's regret which shows that the regret of GD is at least $\sqrt{k}$ times larger than the regret of Loss EG. A similar lower bound is proved for the Follow the Regularized Leader variant of GD in Section 2.4.2. In Section 2.5 we prove lower bounds for online PCA and its dense generalization which hold for any online algorithm, and in Section 2.6 we conclude with a summary of which algorithms are optimal.

2.2 The Online Algorithms

Online uncentered PCA uses the following protocol in each trial $t = 1, \ldots, T$: the algorithm probabilistically chooses a projection matrix $\boldsymbol{P}_t \in \mathrm{R}^{n\times n}$ of rank k. Then a point $\boldsymbol{x}_t \in \mathrm{R}^n$ is received and the algorithm suffers the *loss* $\mathrm{tr}((\boldsymbol{I} - \boldsymbol{P}_t)\boldsymbol{x}_t\boldsymbol{x}_t^\intercal)$.

We also consider the generalization where the instance matrix is any positive definite matrix $\boldsymbol{X}_t$ instead of an outer product $\boldsymbol{x}_t\boldsymbol{x}_t^\intercal$. In that case the loss of the algorithm is $\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\boldsymbol{X}_t)$.[2] The loss is "complementary" to the *gain* $\operatorname{tr}(\boldsymbol{P}_t\boldsymbol{X}_t)$, i.e.

$$\underbrace{\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\boldsymbol{X}_t)}_{\text{loss}} \;=\; \underbrace{\operatorname{tr}(\boldsymbol{X}_t)}_{\text{constant}} \;-\; \underbrace{\operatorname{tr}(\boldsymbol{P}_t\boldsymbol{X}_t)}_{\text{gain}},$$

and the $n-k$ dimensional projection matrix $\boldsymbol{I}-\boldsymbol{P}_t$ is "complementary" to the k dimensional projection matrix $\boldsymbol{P}_t$. These two complementations are inherent to our problem and will be present throughout the chapter.

In the above protocol, the algorithm is allowed to chooses its k dimensional subspace $\boldsymbol{P}_t$ probabilistically. Therefore we use the expected compression loss $\mathrm{E}[\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\boldsymbol{X}_t)]$ as the loss of the algorithm. The regret of the algorithm is then the difference between its cumulative loss and the loss of the best k subspace:

$$\mathrm{REG} = \sum_{t=1}^{T} \mathrm{E}[\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\boldsymbol{X}_t)] \;-\; \min_{\substack{\boldsymbol{P} \text{ projection} \\ \text{matrix of rank } k}} \sum_{t=1}^{T} \operatorname{tr}((\boldsymbol{I}-\boldsymbol{P})\boldsymbol{X}_t).$$

The regret can also be rewritten in terms of gain, but this gives the same value of the regret. Therefore, throughout the chapter we use (expected) losses and "loss" regrets (as defined above) to evaluate the algorithms.

Now we rewrite the loss of the algorithm as $\operatorname{tr}(\mathrm{E}[\boldsymbol{I}-\boldsymbol{P}_t]\boldsymbol{X}_t)$ which shows that for any random prediction $\boldsymbol{P}_t$ of rank k, this loss is fully determined by $\mathrm{E}[\boldsymbol{I}-\boldsymbol{P}_t]$, a convex combination of rank $m=n-k$ projection matrices. Hence it is natural to choose the set $\boldsymbol{\mathcal{W}}_m$ of convex combinations of rank m projection matrices as the parameter set of

[2] If the instance matrix $\boldsymbol{X}_t$ has the eigendecomposition $\sum_i \xi_i\, \boldsymbol{x}_t^i \boldsymbol{x}_t^{i\top}$, then we can re-express the loss as a weighted compression loss

$$\operatorname{tr}((\boldsymbol{I}-\boldsymbol{P}_t)\boldsymbol{X}_t) = \sum_i \xi_i\, ||\boldsymbol{P}_t\boldsymbol{x}_t^i - \boldsymbol{x}_t^i||_2^2.$$

the algorithm. By the definition of projection matrices, $\boldsymbol{\mathcal{W}}_m$ is the set of positive semi-definite matrices of trace m and eigenvalues not larger than 1. The current parameter $\boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_m$ of the online algorithm expresses its "uncertainty" about which subspace of rank m is best for the online data stream seen so far and the (expected) loss in trial t becomes $\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$. Alternatively, the complementary set $\boldsymbol{\mathcal{W}}_k$ of rank k projection matrices can be used as the parameter set (In that case the loss is $\operatorname{tr}((\boldsymbol{I}-\boldsymbol{W}_t)\boldsymbol{X}_t)$). As discussed, there is a one-to-one correspondence between the two parameter sets: Given $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_k$, then $\boldsymbol{I} - \boldsymbol{W}$ is the corresponding convex combination in $\boldsymbol{\mathcal{W}}_m$. However as we shall see, we will motivate online algorithms with divergences between the new and old parameters and for the multiplicative algorithms the choice of the parameter set leads to different algorithms.

The second reason why convex combinations are natural parameter spaces is that since the loss is linear, the convex combination with the minimum loss occurs at a "pure" projection matrix, i.e.

$$\begin{aligned}
\min_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_m} \sum_{t=1}^{T} \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t) &= \min_{\substack{\boldsymbol{P}\text{ projection}\\ \text{matrix of rank } k}} \sum_{t=1}^{T} \operatorname{tr}((\boldsymbol{I}-\boldsymbol{P})\boldsymbol{X}_t) \quad \text{and} \\
\min_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_k} \sum_{t=1}^{T} \operatorname{tr}((\boldsymbol{I}-\boldsymbol{W})\boldsymbol{X}_t) &= \min_{\substack{\boldsymbol{P}\text{ projection}\\ \text{matrix of rank } k}} \sum_{t=1}^{T} \operatorname{tr}((\boldsymbol{I}-\boldsymbol{P})\boldsymbol{X}_t). \qquad (2.2)
\end{aligned}$$

Our protocol requires the algorithm to predict with a rank k projection matrix. Therefore given a parameter matrix $\boldsymbol{W}_t$ in say $\boldsymbol{\mathcal{W}}_m$, the online algorithm still needs to produce a random projection matrix $\boldsymbol{P}_t$ of rank k at the beginning of trial t such that $\mathrm{E}[\boldsymbol{I} - \boldsymbol{P}_t] = \boldsymbol{W}_t$. A simple greedy algorithm for achieving this is given in [WK08]

(Algorithm 2) which efficiently decomposes $\boldsymbol{W}_t$ into a convex combination of up to n projection matrices of rank m. Using the mixture coefficients it is now easy to sample a projection matrix $\boldsymbol{I} - \boldsymbol{P}_t$ from parameter matrix $\boldsymbol{W}_t$.

We now motivate the main two online algorithms used in this chapter: the GD and MEG algorithms. The GD algorithm is straight forward and the MEG algorithm was introduced in [TRW05]. Both are examples of the *Mirror Descent* family of algorithms developed much earlier in the area of convex optimization [NY78]. The Mirror Descent algorithms update their parameter by minimizing a trade-off between a divergence of the new and old parameter and the loss of the new parameter on the current instance, while constraining the new parameter to lie in the parameter set.

For the problem of online PCA, the update specializes into the following two versions depending on the choice of the parameter set:

Loss update on parameter set $\boldsymbol{\mathcal{W}}_m$ (i.e. $\boldsymbol{W}_{t+1}, \boldsymbol{W}, \boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_m$):

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_m}{\operatorname{argmin}} \left(\Delta(\boldsymbol{W}, \boldsymbol{W}_t) + \eta \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)\right). \tag{2.3}$$

Gain update on parameter set $\boldsymbol{\mathcal{W}}_k$ (i.e. $\boldsymbol{W}_{t+1}, \boldsymbol{W}, \boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_k$):

$$\begin{aligned} \boldsymbol{W}_{t+1} &= \underset{\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_k}{\operatorname{argmin}} \left(\Delta(\boldsymbol{W}, \boldsymbol{W}_t) + \eta \ \operatorname{tr}((\boldsymbol{I} - \boldsymbol{W})\boldsymbol{X}_t)\right), \\ &= \underset{\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_k}{\operatorname{argmin}} \left(\Delta(\boldsymbol{W}, \boldsymbol{W}_t) - \eta \ \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)\right). \end{aligned} \tag{2.4}$$

Here $\Delta(\boldsymbol{W}, \boldsymbol{W}_t)$ is the motivating Bregman divergence that will be different for the MEG and GD algorithms. The *Loss update* minimizes a trade-off with the expected loss $\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)$ which is a matrix version of the dot loss used for motivating the Hedge algorithm [FS95]. Note that in the *gain* version, minimizing the loss $-\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)$ is the same as maximizing the gain $\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)$. Recall that there is a one-to-one correspondence between $\boldsymbol{\mathcal{W}}_m$ and $\boldsymbol{\mathcal{W}}_k$, i.e. $\boldsymbol{I}$ minus a parameter in $\boldsymbol{\mathcal{W}}_m$ gives the corresponding parameter in $\boldsymbol{\mathcal{W}}_k$ and vice versa. Therefore, one can for example rewrite the Gain update (2.4) with the parameter set $\boldsymbol{\mathcal{W}}_m$ as well:

$$\widetilde{\boldsymbol{W}}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_m} \Big(\Delta(\boldsymbol{I}-\boldsymbol{W}, \boldsymbol{I}-\widetilde{\boldsymbol{W}}_t) + \eta\ \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)\Big), \tag{2.5}$$

where the above solutions $\widetilde{\boldsymbol{W}}_{t+1} \in \boldsymbol{\mathcal{W}}_m$ of the Gain update is related to the solutions $\boldsymbol{W}_{t+1} \in \boldsymbol{\mathcal{W}}_k$ of (2.4) by the same complimentary relationship, i.e. $\widetilde{\boldsymbol{W}}_{t+1} = \boldsymbol{I} - \boldsymbol{W}_{t+1}$, for $t = 1, \ldots, T$. Notice that the Loss update is motivated by the divergence $\Delta(\boldsymbol{W}, \boldsymbol{W}_t)$ on parameter space $\boldsymbol{\mathcal{W}}_m$ (2.3). On the other hand, when the Gain update is formulated with parameter $\boldsymbol{\mathcal{W}}_m$, then it is motivated by the divergence $\Delta(\boldsymbol{I}-\boldsymbol{W}, \boldsymbol{I}-\widetilde{\boldsymbol{W}}_t)$ (2.5).

Now we define the GD and MEG algorithms for online PCA. For the GD algorithm, the motivating Bregman divergence is the squared Frobenius norm between the old and new parameters: $\Delta(\boldsymbol{W}, \boldsymbol{W}_t) = \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2$ [KW97, Zin03]. With this divergence, the Loss update is solved in the following two steps:

$$\text{GD update:} \qquad \begin{array}{ll} \text{Descent step:} & \widehat{\boldsymbol{W}}_{t+1} = \boldsymbol{W}_t - \eta\boldsymbol{X}_t, \\ \text{Projection step:} & \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_m} \|\boldsymbol{W} - \widehat{\boldsymbol{W}}_{t+1}\|_F^2. \end{array} \tag{2.6}$$

Note, that the split into two steps happens whenever a Bregman divergence is traded off against a linear loss and domain is convex (See [HW09], Section 5.2, for a discussion). For the squared Frobenius norm, the Gain update is equivalent to the Loss update, since when formulating both updates on parameter set $\boldsymbol{\mathcal{W}}_m$, then the divergence $\|\boldsymbol{W} - \boldsymbol{W}_t\|_F^2$

of the Loss update (2.3) and the divergence $\|(\boldsymbol{I}-\boldsymbol{W})-(\boldsymbol{I}-\boldsymbol{W}_t)\|_F^2$ of the Gain update (2.5) are the same.

The MEG algorithm uses the (un-normalized) quantum relative entropy $\Delta(\boldsymbol{W},\boldsymbol{W}_t) = \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W}-\mathbf{log}\,\boldsymbol{W}_t)+\boldsymbol{W}_t-\boldsymbol{W})$ as its motivating Bregman divergence [TRW05] which is based on the matrix logarithm **log**. With this divergence the solutions to the Loss update (2.3) and Gain update (2.4) are the following updates which make use of the matrix exponential **exp** (the inverse of **log**):

$$\text{Loss MEG update:}\qquad \begin{array}{ll} \text{Descent step:} & \widehat{\boldsymbol{W}}_{t+1} = \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \eta \boldsymbol{X}_t), \\ \text{Projection step:} & \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_m} \Delta(\boldsymbol{W},\widehat{\boldsymbol{W}}_{t+1}). \end{array} \tag{2.7}$$

$$\text{Gain MEG update:}\qquad \begin{array}{ll} \text{Descent step:} & \widehat{\boldsymbol{W}}_{t+1} = \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t + \eta \boldsymbol{X}_t), \\ \text{Projection step:} & \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_k} \Delta(\boldsymbol{W},\widehat{\boldsymbol{W}}_{t+1}). \end{array} \tag{2.8}$$

Note that the only difference between the gain and loss versions of MEG is a sign flip in the exponential. The projection steps in the algorithms are with respect to the quantum relative entropy. An efficient procedure for solving such projections is given in Algorithm 4 of [WK08]: it does a projection with respect to the standard relative entropy on the vector of eigenvalues of the parameter matrix.

2.3 Upper Bounds on the Regret

In this section, we present regret upper bounds for the three online algorithms introduced in the previous section, which are Loss MEG, Gain MEG and GD. All three algorithms are examples from the Mirror Descent family of algorithms. Our proof techniques require us to use different restrictions on the worst-case sequences that the adversary can produce. For the Loss MEG algorithm, we give the adversary a *loss budget*, i.e. the adversary must produce a sequence of instances $\boldsymbol{X}_1 \ldots \boldsymbol{X}_T$ for which the loss of the best subspace is upper bounded by the loss budget B_L:

$$\min_{\substack{\boldsymbol{P} \text{ projection} \\ \text{matrix of rank } k}} \sum_{t=1}^{T} \operatorname{tr}((\boldsymbol{I} - \boldsymbol{P})\boldsymbol{X}_t) \leq B_L. \tag{2.9}$$

We call a regret bound that depends on this parameter a *loss budget dependent* bound. A bound of this type was first proved for Loss MEG in [WK08]. The latter paper is the precursor of this work in which the analysis of online algorithms for PCA was started.

For the algorithm of Gain MEG, we give the adversary a *gain budget* B_G, i.e. an upper bound on the gain of the best subspace:

$$\max_{\substack{\boldsymbol{P} \text{ projection} \\ \text{matrix of rank } k}} \sum_{t=1}^{T} \operatorname{tr}(\boldsymbol{P}\boldsymbol{X}_t) \leq B_G. \tag{2.10}$$

Now the adversary can only produce sequences for which all subspaces have gain at most B_G. We call this type of bound a *gain budget dependent* bound. Finally we prove regret bounds of a third type for the GD algorithm. For this type the regret is a function of the number of trials T, and we call such a regret bound a *time dependent* regret bound.

We present the three regret bounds in the next subsection and compare them in the following subsection. As we shall see, upper bounds of the regret in terms of a budget imply time dependent bounds, and for lower bounds the implication is reversed.

2.3.1 Upper Bounds on the Regret of Loss MEG, Gain MEG, and GD

The Loss MEG algorithm (2.7) is the original MEG algorithm developed in the precursor paper [WK08] for online PCA. this chapter proves a loss budget dependent upper bound on the regret of Loss MEG by lifting a bound developed for learning well compared to the best subset of m experts to the matrix case where subsets of size m generalize to projection matrices of rank m (which are complementary to rank k projection matrices (2.2)).

Loss budget dependent bound of Loss MEG:

$$\mathrm{REG}_{\text{Loss MEG}} \;\leq\; \sqrt{2B_L\, m \log\frac{n}{m}} + m\log\frac{n}{m}. \tag{2.11}$$

This bound holds for any sequence of instance matrices (dense as well as sparse) for which the total compression loss of the best rank m subspace does not exceed the loss budget B_L (Condition (2.9)).

We begin by showing that the right-hand side of (2.11) is bounded above by an expression that does not depend on the dimension n of the data points:

$$\mathrm{REG}_{\text{Loss MEG}} \;\leq\; \sqrt{2B_L\, k} + k. \tag{2.12}$$

This follows immediately from the following inequality and the relationship $m = n - k$ $(n = m + k)$:

$$m\log\frac{n}{m} = m\log\left(\frac{k+m}{m}\right) = m\log\left(1+\frac{k}{m}\right) \leq m\frac{k}{m} = k.$$

As mentioned at the beginning of this subsection (and discussed in more detail later in Section 2.4), online PCA specializes to the problem of learning well compared to the best set of $m = n - k$ experts. Regret bounds for the expert setting typically depend logarithmically on the number of experts n. Therefore the above dimension free regret bound might seem puzzling at first. However there is no contradiction. Using the setup here, we have $m = 1$ and $k = n - m = n - 1$ for the vanilla single expert case and the above dimension free bound (2.12) becomes $\sqrt{2B_L(n-1)}$. This bound is not close to the optimum loss budget dependent regret bound for the single expert case which is $O(\sqrt{B_L \log n} + \log n)$. This latter bound is obtained by plugging $m = 1$ into *the original* regret bound (2.11). Thus for $m = 1$, the above dimension free approximation (2.12) of the original bound is loose. However when $k \leq \frac{n}{2}$, then as we shall see in Section 2.5, the dimension free approximation actually is tight. In the precursor paper [WK08], a different but weaker approximation of the original bound was proved that still has an additional logarithmic dependence when $k \leq \frac{n}{2}$: $O(\sqrt{B_L k \log \frac{n}{k}} + k \log \frac{n}{k})$.

We next develop a regret bound for Gain MEG (2.8). The proof technique is a variation of the original regret bound for Loss MEG (and is given for the sake of completeness in Appendix 2.A).

Gain budget dependent bound of Gain MEG:

$$\mathrm{REG}_{\text{Gain MEG}} \;\leq\; \sqrt{2B_G\; k \log \frac{n}{k}}. \tag{2.13}$$

This bound holds for any sequence of instance matrices (dense as well as sparse) for which the total gain of the best rank k subspace does not exceed the gain budget B_G (Condition (2.10)).

Finally, we give a simple regret bound for the GD algorithm. This bound (also observed in [ACS13] and proved for the sake of completeness in Appendix 2.B) is based on two standard techniques: the use of the drop of divergence (squared Frobenius norm) as a measure of progress and the use of the Pythagorean Theorem for handling the projection step [HW01].

Time dependent regret bound of GD:

$$\mathrm{REG}_{\mathrm{GD}} \leq \begin{cases} \sqrt{T\,\frac{km}{n}} & \text{for sparse instances} \\ \\ \sqrt{T\,km} & \text{for dense instances} \end{cases} . \tag{2.14}$$

Note that each regret bound is expressed as a function of a loss budget, a gain budget or a time bound. They are obtained by setting the fixed learning rate of the algorithm as a function of one of these three parameters. The resulting basic algorithms can be used as sub-modules: For example the algorithm can be stopped as soon as the loss budget is reached and restarted with twice the budget and the corresponding re-tuned learning rate. This heuristic is known as the "doubling trick" [CBFH+97]. Much fancier tuning schemes are explored in [vEGKdR11, dRvEGK14] and are not the focus of this chapter.

2.3.2 Comparison of the Regret Upper Bounds

Our goal is to find algorithms that achieve the optimal loss budget dependent and time dependent regret bounds where optimal means that the bound is within a constant factor of optimum. We are not interested in "gain dependent" regret bounds per se, i.e. bounds in terms of a gain budget B_G, because gains are typically much larger than losses. However when the gain budget restricted regret bounds are converted to time bounds, then for some setting (discussed below) the resulting algorithm becomes the only optimal algorithm we are aware of.

The only known *loss budget dependent* regret bound is bound (2.11) for Loss MEG obtained in the original paper for online learning of PCA [WK08]. We will show later in Section 2.5 that this upper bound on the regret is optimal. There are no known loss budget dependent upper bounds on the regret of GD. However in Section 2.4 we prove a lower bound on GD's regret in terms of the loss budget which shows that GD's regret is suboptimal by at least a factor of $\sqrt{k}$ when the regret is expressed as a function of the loss budget.

The discussion of the *time dependent* regret upper bounds is more involved. We first convert the budget dependent regret bounds of the MEG algorithms into time dependent bounds. We shall see later, for lower bounds on the regret, time dependent bounds lead to budget dependent bounds (see Corollary 2.5). Before we do this, recall that the instance matrices $\boldsymbol{X}_t$ are sparse, if the trace of $\boldsymbol{X}_t$ is a most 1, and the instances are allowed to be dense, if the maximum eigenvalues of the $\boldsymbol{X}_t$ is at most 1. Note that for any unit length vector $\boldsymbol{x}_t$, $\operatorname{tr}(\boldsymbol{x}_t\boldsymbol{x}_t^\intercal) = 1$, and therefore PCA belongs to the sparse instance matrix case.

Theorem 2.1. *For the problem of online PCA with T trials, the following regret bounds* *hold for the Loss MEG and Gain MEG algorithms, respectively:*

$$\mathrm{REG}_{\text{Loss MEG}} \;\le\; m\sqrt{\frac{2T}{n}\log\frac{n}{m}} + m\log\frac{n}{m}, \qquad\qquad \mathrm{REG}_{\text{Gain MEG}} \le \sqrt{2T\,k\log\frac{n}{k}}. \tag{2.15}$$

Similarly, for the generalized problem where the instance matrices can be dense, the following regret bounds hold:

$$\mathrm{REG}_{\text{Loss MEG}} \le m\sqrt{2T\log\frac{n}{m}} + m\log\frac{n}{m}, \qquad\qquad \mathrm{REG}_{\text{Gain MEG}} \le k\sqrt{2T\log\frac{n}{k}}. \tag{2.16}$$

Proof. The theorem will be proved by developing simple upper bounds on the loss/gain of the best rank k subspace that depend on the sequence length T. These upper bounds are then used as budgets in the previously obtained budget dependent bounds.

The best rank k subspace picks k eigenvectors of the covariance matrix $\boldsymbol{C} = \sum_{t=1}^{T}\boldsymbol{X}_t$ with the largest eigenvalues. Hence the total compression loss equals the sum of the smallest m eigenvalues of $\boldsymbol{C}$. If $\omega_1,\ldots,\omega_n$ denote all the eigenvalues of $\boldsymbol{C}$, then:

$$\sum_{i=1}^{n}\omega_i = \mathrm{tr}(\boldsymbol{C}) = \sum_{t=1}^{T}\mathrm{tr}\left(\boldsymbol{X}_t\right) \le \begin{cases} T & \text{for sparse instances} \\ Tn & \text{for dense instances}\end{cases}.$$

where the inequality follows from our definition of sparse and dense instance matrices. This implies that the sum of the m smallest eigenvalues is upper bounded by $\frac{Tm}{n}$ and Tm, respectively. By using these two bounds as the loss budget B_L in (2.11), we get the time dependent bound for Loss MEG for sparse and dense instances, respectively.

For the regret bounds of Gain MEG, we use the fact that B_G is upper bounded by T when instances are sparse and upper bounded by kT when the instances are dense,

	sparse instances		dense instances	
	$k \le \frac{n}{2}$	$k \ge \frac{n}{2}$	$k \le \frac{n}{2}$	$k \ge \frac{n}{2}$
Loss MEG	$\boldsymbol{\sqrt{Tk}}$	$\boldsymbol{\sqrt{Tm\left(\log\frac{n}{m}\right)/\frac{n}{m}}}$	$\sqrt{Tkm}$	$\boldsymbol{\sqrt{Tm^2\log\frac{n}{m}}}$
Gain MEG	$\sqrt{Tk\log\frac{n}{k}}$	$\sqrt{Tm}$	$\boldsymbol{\sqrt{Tk^2\ln\frac{n}{k}}}$	$\sqrt{Tkm}$
GD	$\boldsymbol{\sqrt{Tk}}$	$\sqrt{Tm}$	$\sqrt{Tkm}$	$\sqrt{Tkm}$

Table 2.1: Comparison of the time dependent upper bounds on the regret of the Loss MEG, Gain MEG, and GD algorithms. Each column corresponds to one of the four combinations of sparse or dense instance matrices versus $k \le \frac{n}{2}$ or $k \ge \frac{n}{2}$, respectively. All bounds were given in Section 2.3.1 and Section 2.3.2: constants are omitted, we only show the leading term of each bound, and when we compare Loss and Gain MEG bounds, we use $m\ln\frac{n}{m} = \Theta(k)$ when $k \le \frac{n}{2}$ and $k\ln\frac{n}{k} = \Theta(m)$ when $k \ge \frac{n}{2}$. Recall that m is shorthand for $n-k$. The best (smallest) bound for each case (column) is shown in bold. In Section 2.5, all bold bounds will be show to be optimal (within constant factors).

and plug these values for B_G into (2.13). $\square$

Table 2.1 compares time dependent upper bounds for each of the three algorithms (Loss MEG, Gain MEG, GD) where we consider each of the 4 variants of the problem: sparse or dense instance matrices versus $k \le \frac{n}{2}$ or $k \ge \frac{n}{2}$.

As far as time dependent bounds are concerned, no single algorithm is optimal in all cases. In Table 2.1, the optimum bounds are shown in bold. The lower bounds matching these bold bounds within a constant factor will be proved in Section 2.5. Note that one version of MEG (either the loss or gain version) is optimal in each case, while GD is optimal only in first case (This is the most important case in practice: online PCA with $k \ll n$). For the remaining three cases, consider the ratio between the GD's bound and the better of the two MEG bounds, which is

- $\sqrt{\frac{n}{m} / \left(\log \frac{n}{m}\right)}$, when the instances are sparse and $k \geq \frac{n}{2}$,
- $\sqrt{\frac{n}{k} / \left(\log \frac{n}{k}\right)}$, when the instances are dense and $k \leq \frac{n}{2}$ and
- $\sqrt{\frac{n}{m} / \left(\log \frac{n}{m}\right)}$, when the instances are dense and $k \geq \frac{n}{2}$.

Since none of these three ratios can be upper bounded by a constant, GD is clearly suboptimal in each of the remaining three cases.

2.4 Lower Bounds on the Regret of GD

Recall that for the case of online PCA, the instances are sparse and the subspace dimension k is typically at most $\frac{n}{2}$. In this case Loss MEG has regret $O(\sqrt{Tk})$ and the regret of GD is $O(\sqrt{Tk})$ as well. As for loss budget dependent regret bounds, Loss MEG has regret $O(\sqrt{B_L k} + k)$ and we initially conjectured that GD has the same bound. However, this is not true: we will now show in this section an $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$ *lower bound* on the regret of GD for sparse instance sequences when $k \leq \frac{n}{2}$. In contrast, Loss MEG's regret bound of $O(\sqrt{B_L k} + k)$ will be shown to be optimal in Section 2.5 for this case. It follows that GD is suboptimal by at least a factor of $\sqrt{k}$ when $B_L = \Omega(k^2)$. A detailed comparison of the lower bound for GD and the optimum upper bound is given in Table 2.2.

It suffices to prove lower bounds on GD's regret on a restricted class of instance matrices: We assume that all instance matrices are in the same eigensystem, i.e. they are diagonal matrices $\boldsymbol{X} = \operatorname{diag}(\boldsymbol{\ell})$ with $\boldsymbol{\ell} \in \mathrm{R}^n_{\geq 0}$. We call the diagonals $\boldsymbol{\ell}$ the *loss vectors*. In the sparse instance case, the loss vectors are further restricted to be one of n unit vectors $\boldsymbol{e}_i$, i.e. $\boldsymbol{X} = \operatorname{diag}(\boldsymbol{e}_i) = \boldsymbol{e}_i\boldsymbol{e}_i^\intercal$. In the dense instance case, the loss vectors $\boldsymbol{\ell}$ are restricted to lie in $[0,1]^n$.

When all instance matrices are diagonal, the covariance matrix is always diagonal as well. The off-diagonal elements in a parameter matrix $\boldsymbol{W}$ are irrelevant and therefore the algorithm's loss and regret is determined by the diagonals of the parameter matrices $\boldsymbol{W}$ of trace m. Therefore without loss of generality we can assume that the parameter matrices are diagonal as well, i.e. $\boldsymbol{W} = \operatorname{diag}(\boldsymbol{w})$ where $\boldsymbol{w}$ is a *weight vector* in $[0,1]^n$ with total weight m. Note that the loss becomes a dot product between the weight vector and the loss vector:

$$\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}) = \operatorname{tr}(\operatorname{diag}(\boldsymbol{w})\operatorname{diag}(\boldsymbol{\ell})) = \boldsymbol{w} \cdot \boldsymbol{\ell}.$$

What is the prediction of the algorithm with a diagonal parameter matrix $\boldsymbol{W} = \operatorname{diag}(\boldsymbol{w})$? It probabilistically predicts with an m dimensional projection matrix $\boldsymbol{P}$ s.t. $\mathrm{E}[\boldsymbol{P}] = \operatorname{diag}(\boldsymbol{w})$. This means $\boldsymbol{P}$ is a subset of size m from $\{\boldsymbol{e}_1\boldsymbol{e}_1^\intercal, \boldsymbol{e}_2\boldsymbol{e}_2^\intercal, \ldots, \boldsymbol{e}_n\boldsymbol{e}_n^\intercal\}$. The diagonals of such projection matrices consists of exactly m ones and $n - m = k$ zeros. In other words the diagonals are indicator vectors of the chosen *subsets of size* m and the expected indicator vector equals the weight vector $\boldsymbol{w}$.

We just outlined one of the main insights of [WK08]: The restriction of the

PCA problem to diagonal matrices corresponds to learning a subset of size m. The n components of the vectors are usually called *experts*. At trail t the algorithm chooses a subset of m experts. It then receives a loss vector $\boldsymbol{\ell} \in \mathrm{R}^n_{\geq 0}$ for the experts and incurs the total loss of the chosen m experts. The algorithm maintains its uncertainty over the m-sets by means of a parameter vector $\boldsymbol{w} \in [0,1]^n$ with total weight m, and it chooses the subset of size m probabilistically so that the expected indicator vector equal $\boldsymbol{w}$. We denote the set of such parameter vectors as $\boldsymbol{\mathcal{S}}_m$. In the sparse instance case, the loss vector is a unit vector (only one expert incurs a unit of loss). In the dense instance case $\boldsymbol{\ell} \in [0,1]^n$, i.e. each expert has bounded loss in $[0,1]$.

2.4.1 Lower Bound on the Regret of GD Algorithm

The GD algorithm for online PCA (2.6) specializes to the following update of the parameter vector for learning sets:

$$\begin{aligned} &\text{Descent step:} && \hat{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{\ell}_t, \\ &\text{Projection step:} && \boldsymbol{w}_{t+1} = \operatorname{argmin}_{\boldsymbol{w} \in \boldsymbol{\mathcal{S}}_m} \|\boldsymbol{w} - \hat{\boldsymbol{w}}_{t+1}\|^2. \end{aligned} \tag{2.17}$$

We now give a lower bound on the regret of the GD algorithm for the m-set problem. This lower bounds is expressed as a function of the loss budget.

Theorem 2.2. *Consider the $m = n - k$ set problem with $k \leq n/2$ and unit loss vectors. Then for any fixed learning rate $\eta \geq 0$, the GD algorithm* (2.17) *can be forced to have regret* $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.

We prove this theorem in Appendix 2.C. From the fact that m-set problem is

Regret bounds for sparse instances and $k \leq \frac{n}{2}$	$B_L \leq k$	$k \leq B_L \leq k^2$	$k^2 \leq B_L$
Upper bound on regret of Loss MEG (see (2.11))	$\boldsymbol{O(k)}$	$\boldsymbol{O(\sqrt{B_L k})}$	$\boldsymbol{O(\sqrt{B_L k})}$
Lower bound on regret of GD (see Theorem 2.2)	$\Omega(k)$	$\Omega(B_L)$	$\Omega(k\sqrt{B_L})$

Table 2.2: Comparison of the loss budget dependent regret bounds for online PCA with $k \leq \frac{n}{2}$. Given dimension k of the subspace, each column shows the values of the two bounds for a specific range of the loss budget B_L. The first row gives the upper bound on the regret of Loss MEG in bold, which will be shown to be optimal in Section 2.5. The second row gives the lower bound on the regret of GD, which is sub-optimal whenever $B_L \geq k$.

a special case of PCA problem, we get the following corollary, which shows that the GD algorithm is suboptimal (see Table 2.2 for an overview):

Corollary 2.1. *Consider the PCA problem with $k \leq n/2$ and sparse instance matrices. Then for any fixed learning rate $\eta \geq 0$, the GD algoirthm* (2.6) *can be forced to have regret* $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.

2.4.2 Lower Bound on the Regret of the Follow the Regularized Leader GD Algorithm (FRL-GD)

In the previous section, we showed that for online PCA with sparse instance matrices and $k \leq \frac{n}{2}$, the GD algorithm is sub-optimal for loss budget dependent regret bounds. However, our lower bounds are only for the Mirror Descent version of GD given in (2.6). This algorithm is prone to "forgetting" lots of information about the past losses when projections with respect to inequality constraints are involved. Recall that at the end of each trial t, the mirror descent algorithm uses the last parameter $\boldsymbol{W}_t$ as a summary of the knowledge attained so far, and minimizes a trade-off between a divergence to the $\boldsymbol{W}_t$ and the loss on the last data point $\boldsymbol{x}_t$ to determine the next parameter $\boldsymbol{W}_{t+1}$. When the parameter resulting from the trade-off lies outside the parameter set, then it is projected back into the parameter set (see update (2.6)). In the case when the projection enforces inequality constraints on the parameters, information about the past losses may be lost. This issue was first discussed in Section 5.5 of [HW09]. Curiously enough, Bregman projections with respect to only equality constraints do not loose information.

We now demonstrate in more detail the "forgetting" issue for the Mirror Descent GD algorithm when applied to online PCA. First recall that the batch PCA solution consists of the subspace spanned by the k eigenvectors belonging to the k largest values of the covariance matrix $\boldsymbol{C} = \sum_{t=1}^{T} \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$. The complementary space is the $m = n - k$ dimensional subspace formed by the m eigenvectors of m largest eigenvalues of $-\boldsymbol{C}$. Hence, the final parameter $\boldsymbol{W}_{T+1}$ of the on-line algorithm should have the same eigenvectors as $-\boldsymbol{C}$, as well as the order of their corresponding eigenvalues. The descent step of (2.6) accumulates the scaled negated instance matrices $\boldsymbol{X}_t = \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$, i.e. $\widehat{\boldsymbol{W}}_{t+1} = \boldsymbol{W}_t - \eta \boldsymbol{X}_t$. In the projection step of (2.6), the parameter matrix $\widehat{\boldsymbol{W}}_{t+1}$ is projected back to the parameter set $\boldsymbol{\mathcal{W}}_m$ by enforcing an equality constraint $\operatorname{tr}(\boldsymbol{W}_{t+1}) = m$ and inequality constraints that keep all the eigenvalues of $\boldsymbol{W}_{t+1}$ are in the range $[0, 1]$. The equality constraint on $\widehat{\boldsymbol{W}}_{t+1}$ results in adding to $\widehat{\boldsymbol{W}}_{t+1}$ a scaled version of the identity matrix $\boldsymbol{I}$ (See Appendix 2.C). These iterated shifts do not affect either the eigenvectors or the order of their corresponding eigenvalues. However, when the inequality constraints are enforced, then at trial t the eigenvalues of $\widehat{\boldsymbol{W}}_{t+1}$ that are larger than 1 or less than 0 are capped at 1 and 0, respectively. Performing such a non-uniform capping of $\widehat{\boldsymbol{W}}_{t+1}$'s eigenvalues in each trial will results in a final parameter $\boldsymbol{W}_{T+1}$ with an eigensystem that is typically different from $-\boldsymbol{C}$. Therefore the PCA solution extracted from $\boldsymbol{W}_{T+1}$ and the covariance matrix $\boldsymbol{C}$ will not be the same.

There is another version of the GD algorithm that does not "forget": The Follow the Regularized Leader GD (FRL-GD) algorithm (see, e.g., [SS07][3]) trades off the total loss on all data points against the Frobenius norm of the parameter matrix:

Follow the regularized leader:

$$\widehat{\boldsymbol{W}}_{t+1} = \operatorname{argmin}\left(\|\boldsymbol{W}\|_F^2 + \eta \sum_{q=1}^{t} \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_q) \right) = -\eta \sum_{q=1}^{t} \boldsymbol{X}_q,$$

Projection step:

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_m}{\operatorname{argmin}} \|\boldsymbol{W} - \widehat{\boldsymbol{W}}_{t+1}\|_F^2 = \underset{\substack{\text{Eigenvalues of } \boldsymbol{W} \text{ in} \\ [0,1] \text{ and } \operatorname{tr}(\boldsymbol{W}) = m}}{\operatorname{argmin}} \|\boldsymbol{W} - \widehat{\boldsymbol{W}}_{t+1}\|_F^2. \tag{2.18}$$

Note that in each trial, the update (2.18) projects a parameter $\widehat{\boldsymbol{W}}_{t+1}$ that accumulates all the past scaled negated instance matrices $(-\eta\boldsymbol{X}_t)$ back to trial one. In contrast, the Mirror Descent update in (2.6) performs projection iteratively, i.e. it projects parameter matrices of previous trails that are projections themselves. Therefore, the FRL-GD

[3]This algorithm is also called as the Incremental Off-line Algorithm in [AW01].

algorithm circumvents the forgetting issue introduced by iterative projections with respect to inequality constraints. In fact the final parameter $\boldsymbol{W}_{T+1}$ of the FRL-GD is the projection of the scaled negated covariance matrix $\widehat{\boldsymbol{W}}_{T+1} = -\eta \sum_{t=1}^{T} \boldsymbol{x}_t \boldsymbol{x}_t^\intercal = -\eta \boldsymbol{C}$. We will show essentially in Appendix 2.E that a single projection operation does not change the set of eigenvectors belonging to the m largest eigenvalues. This means that the eigenvectors belonging to the k smallest eigenvalues of $\boldsymbol{W}_{T+1}$ agree with the eigenvectors of $\boldsymbol{C}$ belonging to the k largest eigenvalues of $\boldsymbol{C}$.

Encouraged by this observation, we initially conjectured that the FRL-GD is strictly better than the commonly studied Mirror Descent version. More concretely, we conjectured that the FRL-GD has the optimal loss budget dependent regret bound for online PCA (as Mirror Descent MEG does which enforces the non-negativity constraints with its divergence). Unfortunately, we are able to show the opposite: The $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$ lower bound we showed for (Mirror Descent) GD in Theorem 2.2 also holds for FRL-GD. To be precise, we have the following theorem and corollary:

Theorem 2.3. *Consider the $m = n - k$ set problem with $k \leq n/2$ and unit loss vectors. Then for any fixed learning rate $\eta \geq 0$, the vector version of the FRL-GD algorithm* (2.18) *can be forced to have regret $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.*

The proof is given in Appendix 2.D. Theorem 2.3 immediately gives the lower bound on the regret of FRL-GD algorithm for the online PCA:

Corollary 2.2. *Consider the PCA problem with $k \leq n/2$ and sparse instance matrices.*

Then for any fixed learning rate $\eta \geq 0$, the FRL-GD algorithm (2.18) *can be forced to have regret* $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.

This shows that the worst case regret of the FRL-GD algorithm is the same as that of (Mirror Descent) GD, and hence suboptimal.

2.5 General Lower Bounds and Optimal Algorithms

In the previous section, we presented lower bounds on the regret of the GD algorithms. In this section we present lower bounds on the regret of *any* algorithm that solves the online PCA problem and its generalization to the dense instance matrix case. More importantly, these lower bounds match all our upper bounds on the regret of the MEG algorithms within a constant factor (See bold entries in Table 2.1 and Table 2.2) To be precise, we will prove in this section a series of regret lower bounds that match our loss budget dependent upper bound (2.11) on the regret of Loss MEG, and our time dependent upper bounds (Theorem 2.1) on the regret of Loss MEG and Gain MEG, respectively. For the time dependent bounds, our lower bounds will match the lower of the two MEG bounds in each of the four sub-cases of the problem, i.e. sparse or dense instance matrices versus $k \leq \frac{n}{2}$ or $k \geq \frac{n}{2}$ (See Table 2.1 for a summary). Note in one case the GD algorithm is also optimal: time dependent regret bounds for PCA when $k \leq \frac{n}{2}$.

We begin with an overview of our proof techniques for the regret lower bounds that hold for any algorithm solving online PCA and its dense generalization. When proving *upper bounds* on the regret (in Section 2.3), we first proved upper bounds as a function of the loss budget B_L and then converted them into time dependent upper bounds. For *lower* bounds on the regret, the order is reversed: we first will show time dependent lower bounds and then convert them into loss budget dependent lower bounds. As discussed in Section 2.4, it suffices to prove lower bounds for the m-set problem, which is the hard special case when all instances are diagonal.

Let $\boldsymbol{\mathcal{A}}$ be the set of all online algorithms for the m-set problem. Such algorithms maintain a weight vector in $\boldsymbol{\mathcal{S}}_m$ (consisting of all vectors in $[0,1]^n$ of total weight m). For an algorithm $A \in \boldsymbol{\mathcal{A}}$, we denote its regret by $\mathrm{REG}(A, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T)$ where $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$ is a sequence of T loss vectors. The loss vectors $\boldsymbol{\ell}_t$ lie in a constraint set $\boldsymbol{\mathcal{L}}$. The constraint set $\boldsymbol{\mathcal{L}}$ either consists of all n dimensional unit vectors (the sparse case), or $\boldsymbol{\mathcal{L}} = [0,1]^n$ (the dense case). We use the standard method of lower bounding the regret for worst case loss sequences from $\boldsymbol{\mathcal{L}}$ by the expected regret when the loss vectors are generated i.i.d. with respect to a distribution $\mathcal{P}$ on $\boldsymbol{\mathcal{L}}$:

$$\begin{aligned} &\underset{\substack{\text{over any}\\ \text{alg. } A \in \boldsymbol{\mathcal{A}}}}{\min} \left\{ \underset{\substack{\text{over loss vectors}\\ \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T \in \boldsymbol{\mathcal{L}}}}{\max} \mathrm{REG}(A, \boldsymbol{\ell}_{1,\ldots,T}) \right\} \\ &\qquad\qquad\qquad \geq \underset{\substack{\text{over any}\\ \text{alg. } A \in \boldsymbol{\mathcal{A}}}}{\min} \left\{ \mathrm{E}_{\boldsymbol{\ell}_1,\ldots,\boldsymbol{\ell}_T \sim \mathcal{P}^T} \left[\, \mathrm{REG}(A, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T) \,\right] \right\}. \end{aligned}$$

Each lower bound is proved as follows: Choose a distribution $\mathcal{P}$ on $\boldsymbol{\mathcal{L}}$, and then show a lower bound on the expected regret of any algorithm $A \in \boldsymbol{\mathcal{A}}$. Note that this expectation becomes the expected loss of A minus the expected loss of the best comparator (i.e. the best m-set). We first prove time dependent regret lower bounds with sparse and dense instance matrices in sections 2.5.1 and 2.5.2, respectively. Finally we convert these lower bounds into loss budget dependent lower bounds (in Section 2.5.3).

2.5.1 Time Dependent Lower Bounds for Online PCA

Recall that $m = n - k$. First, we give a lower bound on the regret of any algorithm for the m-set problem, when $k \leq \frac{n}{2}$:

Theorem 2.4. *Consider the m-set problem with unit loss vectors. Then for $k \leq \frac{n}{2}$ and $T \geq k$, any online algorithm suffers worst case regret at least $\Omega(\sqrt{Tk})$.*

The proof is given in Appendix 2.F. We lower bound the expected loss w.r.t. the distribution $\mathcal{P}$ which is uniform on the first $2k$ unit vectors. Note that Theorem 2.4 requires the condition $T \geq k$. For the case $T < k$, there is a lower bounds of $\Omega(T)$ (See Theorem 2.8 in Appendix 2.G). For unit loss vectors, any algorithm has loss (and regret) $O(T)$. Therefore when $T < k$, any algorithm achieves the minimax regret up to a constant factor.

We now consider the uncommon case when $k \geq \frac{n}{2}$:

Theorem 2.5. *Consider the m-set problem with unit loss vectors. Then for $k \geq \frac{n}{2}$ and $T \geq n \log_2(n/m)$, any online algorithm suffers worst case regret of at least $\Omega(m\sqrt{\frac{T}{n} \ln \frac{n}{m}})$.*

We now set $\mathcal{P}$ to the uniform distribution on all n unit vectors (See Appendix 2.F). The small T case (here $T < n \log_2(n/m)$) is slightly more involved. There is a lower bound of $\Omega(\frac{m}{n}T)$ regret for any algorithm (see Theorem 2.9 in Appendix 2.G).

Also the algorithm which predicts with the uniform weight $\frac{m}{n}$ on all experts achieves the matching regret of $O(\frac{m}{n}T)$.

Recall that the m-set problem with unit loss vectors is a special case of the online PCA problem. Combining the above two lower bounds for different ranges of k with our upper bound (Inequality (2.15)) on the regret of Loss MEG for online PCA gives the following corollary:

Corollary 2.3. *Consider the problem of online PCA. Then for $T \geq n \log_2(n/m)$, the $\Theta(m\sqrt{\frac{T}{n} \ln \frac{n}{m}})$ regret of Loss MEG is within a constant factor of the minimax regret.*

Note that we do not use the condition $T \geq k$ of Theorem 2.4, since when $k \leq \frac{n}{2}$, $k = \Theta(n \log_2(n/m))$.

2.5.2 Time Dependent Lower Bound for the Generalization with Dense Instance Matrices

We first give the time dependent lower bound for the m-set problem with dense loss vectors.

Theorem 2.6. *Consider the m-set problem with dense loss vectors. Then for $T \geq \log_2 \frac{n}{\min\{k,m\}}$, any online algorithm suffers worst case regret of at least*

$$\Omega(k\sqrt{T \ln \frac{n}{k}}) \text{ when } k \leq \frac{n}{2} \quad \text{or} \quad \Omega(m\sqrt{T \ln \frac{n}{m}}) \text{ when } k \geq \frac{n}{2}.$$

The proof is given in Appendix 2.F. The distribution $\mathcal{P}$ is such that each expert incurs a unit of loss with probability 1/2 independently from the other experts.

For the small T case ($T < \log_2 \frac{n}{\min\{k,m\}}$), there is a lower bound of $\Omega(\min\{Tm, Tk\})$ (See Theorem 2.10 and Theorem 2.11 in Appendix 2.G). A matching upper bound of $O(\min\{Tm, Tk\})$ on the regret of any algorithm can be reasoned as follows: Recall that at each trial, the algorithm suffers loss $\boldsymbol{\ell}_t \cdot \boldsymbol{w}_t$, where $\boldsymbol{\ell}_t$ is a dense loss vectors in $[0,1]^n$ and $\boldsymbol{w}_t$ is a weight vector that always sums to m. Hence, any algorithm suffers loss at most m per trial and for T trials and the cumulative loss (and regret) is at most Tm. The Tk upper bound can be showed similarly by considering the "gain" of the best m set $\boldsymbol{w}^*$, which is $\sum_{t=1}^{T} \sum_{i=1}^{n} l_{t,i}(1 - w_i^*) \leq Tk$.

Combining the lower bounds of Theorem 2.6 with the upper bounds on the regret of Loss MEG and Gain MEG when the instance matrices can be dense (inequalities (2.16)), results in the following corollary, which states that the Gain MEG is optimal for $k \leq \frac{n}{2}$ while the Loss MEG is optimal for $k \geq \frac{n}{2}$.

Corollary 2.4. *Consider the generalization of online PCA where the instance matrices can be dense.*

- *When $k \leq \frac{n}{2}$ and $T \geq \log_2 \frac{n}{k}$, then the regret $\Theta(k\sqrt{T \log \frac{n}{k}})$ of Gain MEG is within a constant factor of the minimax regret,*
- *When $k \geq \frac{n}{2}$ and $T \geq \log_2 \frac{n}{m}$, then the regret $\Theta(m\sqrt{T \log \frac{n}{m}})$ of Loss MEG is within a constant factor of the minimax regret.*

2.5.3 Loss Budget Dependent Lower Bounds

In this subsection, we give regret lower bounds that are functions of the loss budget B_L (defined in (2.9)). Similar to our loss budget dependent upper bound (2.11) on the regret of Loss MEG, the loss dependent lower bounds are the same for both unit and dense loss vectors:

Theorem 2.7. *For the m-set problem with either unit or dense loss vectors, any online algorithm suffers worst case regret at least $\Omega(\sqrt{B_L m \ln \frac{n}{m}}) + m \ln \frac{n}{m})$.*

The proof of the theorem is given in Appendix 2.H. We convert the time dependent lower bounds given in Theorem 2.4 and Theorem 2.5 into loss budget dependent ones. Note that unlike our time dependent lower bounds, Theorem 2.7 is stated for the full range of the loss budget parameter B_L. The proof also distinguishes between a small and a large budget case depending on whether $B_L \leq m \ln \frac{n}{m}$. The lower bound of $\Theta(m \ln \frac{n}{m})$ follows from a conversion. However the upper bound of $O(m \ln \frac{n}{m})$ for the small budget case is non-trivial. Incidentally, this upper bound is achieved by Loss MEG.

Finally, combining this lower bound with the upper bounds (2.11) on the regret of Loss MEG, gives the following corollary, which establishes the optimality of Loss MEG no matter if the instance matrices are dense or sparse.

Corollary 2.5. *Consider both the problem of online PCA and its generalization to the dense instance matrices case. Then the regret $\Theta(\sqrt{B_L m \ln \frac{n}{m}} + m \ln \frac{n}{m})$ of Loss MEG is within a constant factor of the minimax regret.*

2.6 Conclusion

In this chapter, we carefully studied two popular online algorithms for PCA: the Gradient Descent (GD) and Matrix Exponentiated Gradient (MEG) algorithms. Contrary to the popular belief that the Exponentiated Gradient family is suboptimal when the instances are sparse (see, e.g., [KW97]), we showed that both algorithms are optimal within a constant factor on worst-case sequences of sparse instances, when the regret is expressed as a function of the number of trials. Furthermore, when considering regret bounds as a function of a loss budget, then MEG remains optimal and strictly outperforms GD for sparse instances.

We also studied a generalization of the online PCA problem, in which the adversary is allowed to present the algorithm with dense instance matrices. Again we showed that MEG is optimal and strictly better than GD in this setting. It follows that MEG is the algorithm of choice for online PCA as well as for its generalization to dense matrices.

In this chapter we focused on obtaining online algorithms with optimal regret and we ignored efficiency concerns. Straight forward implementations of both the GD and MEG online PCA updates required $O(n^3)$ computation per trial (because they require an eigendecomposition of the parameter matrices). This leads to a major open problem for online PCA [HKW10c]: Is there any algorithm that can achieve optimal regret with $O(n^2)$ computation per trial. To this end, [ACS13] considers the Gain version of GD (Equation (2.4), with the squared Euclidean distance as the divergence) where the projection enforces the additional constraint that the parameter matrix $\boldsymbol{W}_t$ has rank $\widehat{k}$. Encouraging experimental results are provided for the choice $\widehat{k} = k + 1$. However, as we shall see immediately, when the unit length data points are chosen by an adversary, then any algorithm that uses parameter matrices of rank $\widehat{k}$ less than n suffers worst case regret linear in T. Recall that the parameter matrix $\boldsymbol{W}_t$ at trial t is simply the expected projection matrix of rank k chosen by the algorithm and this matrix is defined for any (deterministic or randomized) algorithm. We give an adversary argument for any algorithm for which the rank of the parameter matrix $\boldsymbol{W}_t$ at any trial t is at most $\widehat{k}$. The parameter matrices are known to the adversary. Also the initial parameter matrix $\boldsymbol{W}_1$ must have rank $\widehat{k}$ and be known to the adversary. For any algorithm following this setup the adversary argument proceeds as follows: At the beginning of the game the adversary fixes any subspace $\mathcal{Q}$ of dimension $\widehat{k} + 1$. In each trial, the adversary picks a unit length vector $\boldsymbol{x}_t \in \mathcal{Q}$, which is in the null space of the parameter matrix $\boldsymbol{W}_t$ of the algorithm (This is always possible, because the dimension of $\mathcal{Q}$ is larger than the rank of $\boldsymbol{W}_t$). After T trials, the algorithm has zero gain, while the total gain T is accumulated within subspace $\mathcal{Q}$. This means that there are k orthogonal directions within $\mathcal{Q}$ with the total gain at least $\frac{k}{\widehat{k}+1}T$ and therefore, the algorithm suffers regret at least $\frac{k}{\widehat{k}+1}T$.

Besides restricting the rank of the parameter matrix, a second approach is to add perturbations to the current covariance matrix and then find the eigenvectors of the k-largest eigenvalues [HKW10c]. So far this approach has not led to algorithms with optimal regret bounds and $O(n^2)$ update time. Some partial results recently appeared in [GHM15] and [KW15].

Appendix

2.A Proof of Upper Bound (2.13) on the Regret of Gain MEG

Proof. The proof is based on the by now standard proof techniques of [TRW05]. Let $\boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_k$ be the parameter of the Gain MEG algorithm at trial t and $\boldsymbol{X}_t$ be the instance matrix at this trial. Now plugging the (un-normalized) relative entropy $\Delta(\boldsymbol{W}, \boldsymbol{W}_t) = \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_t) + \boldsymbol{W}_t - \boldsymbol{W})$ into the descent step of the Gain MEG algorithm (2.8) gives:

$$\widehat{\boldsymbol{W}}_{t+1} = \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t + \eta \boldsymbol{X}_t) \quad \text{where } \eta \geq 0 \text{ is the learning rate.}$$

Take any projection matrix $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_k$ as a comparator and use $\Delta(\boldsymbol{W}, \boldsymbol{W}_t) -$

$\Delta(\boldsymbol{W}, \boldsymbol{W}_{t+1})$ as a measure of progress towards $\boldsymbol{W}$:

$$\begin{aligned}
\Delta(\boldsymbol{W}, \boldsymbol{W}_t) - \Delta(\boldsymbol{W}, \boldsymbol{W}_{t+1}) &\geq \Delta(\boldsymbol{W}, \boldsymbol{W}_t) - \Delta(\boldsymbol{W}, \widehat{\boldsymbol{W}}_{t+1}) \\
&= \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\widehat{\boldsymbol{W}}_{t+1} - \mathbf{log}\,\boldsymbol{W}_t) + \boldsymbol{W}_t - \widehat{\boldsymbol{W}}_{t+1}) \\
&= \operatorname{tr}(\eta \boldsymbol{W}\boldsymbol{X}_t) + \operatorname{tr}(\boldsymbol{W}_t - \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t + \eta\boldsymbol{X}_t)) \qquad (2.19) \\
&\geq \operatorname{tr}(\eta \boldsymbol{W}\boldsymbol{X}_t) + \operatorname{tr}(\boldsymbol{W}_t - \boldsymbol{W}_t\,\mathbf{exp}(\eta\boldsymbol{X}_t)) \\
&= \operatorname{tr}(\eta \boldsymbol{W}\boldsymbol{X}_t) + \operatorname{tr}(\boldsymbol{W}_t(\boldsymbol{I} - \mathbf{exp}(\eta\boldsymbol{X}_t)),
\end{aligned}$$

where the first inequality follows from the Pythagorean Theorem and the second from the Golden-Thompson inequality: $\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t + \eta\boldsymbol{X}_t) \leq \operatorname{tr}(\boldsymbol{W}_t\,\mathbf{exp}(\eta\boldsymbol{X}_t))$. By Lemma 2.1 of [TRW05],

$$\operatorname{tr}(\boldsymbol{W}_t(\boldsymbol{I} - \mathbf{exp}(\eta\boldsymbol{X}_t))) \geq (1 - e^\eta)\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t),$$

and therefore

$$\Delta(\boldsymbol{W}, \boldsymbol{W}_t) - \Delta(\boldsymbol{W}, \boldsymbol{W}_{t+1}) \geq \eta \underbrace{\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)}_{\substack{\text{gain of the}\\\text{comparator}}} + (1 - e^\eta) \underbrace{\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)}_{\substack{\text{gain of the}\\\text{algorithm}}}.$$

Summing over trials gives:

$$\begin{aligned}
&\eta \overbrace{\sum_{t=1}^T \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t)}^{\substack{\text{total gain } G_{\boldsymbol{W}} \text{ of}\\\text{the comparator } \boldsymbol{W}}} + (1 - e^\eta) \overbrace{\sum_{t=1}^T \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)}^{\substack{\text{total gain } G_A\\\text{of Gain MEG}}} \\
&\qquad\leq \underbrace{\Delta(\boldsymbol{W}, \boldsymbol{W}_1)}_{\substack{\leq k \log \frac{k}{n}\\\text{with initialization}\\\boldsymbol{W}_1 = \frac{k}{n}\boldsymbol{I}}} - \underbrace{\Delta(\boldsymbol{W}, \boldsymbol{W}_{T+1})}_{\geq 0}.
\end{aligned}$$

We now rearrange the terms to bound the regret of Gain MEG:

$$G_{\boldsymbol{W}} - G_A \quad \leq \quad \frac{1}{e^\eta - 1}\, k \log \frac{k}{n} + \left(1 - \frac{\eta}{e^\eta - 1}\right) G_{\boldsymbol{W}}. \tag{2.20}$$

Since $e^\eta \geq 1 + \eta$, the coefficient $\frac{1}{e^\eta - 1}$ of the first term on the RHS is upper bounded by $\frac{1}{\eta}$. Next we upper bound the coefficient of the second term by η:

$$1 - \frac{\eta}{e^\eta - 1} = 1 - \frac{\eta e^{-\eta}}{1 - e^{-\eta}} \leq 1 - \frac{\eta e^{-\eta}}{\eta} = 1 - e^{-\eta} \leq \eta.$$

The inequality (2.13) on the regret of Gain MEG now follows from these two upper bounds, the budget inequality $G_{\boldsymbol{W}} \leq B_G$ and from tuning the learning rate as a function of B_G:

$$\mathrm{REG}_{\text{Gain EG}} \quad \leq \quad \frac{k \log \frac{k}{n}}{\eta} + \eta B_G \quad \overset{\eta = \sqrt{\frac{\log \frac{k}{n}}{B_G}}}{=} \quad \sqrt{2 B_G\, k \log \frac{k}{n}}.$$

$\square$

2.B Proof of Upper Bound (2.14) on the Regret of GD

Proof. This proof is also standard [HW01]. Minor alterations are needed because we have matrix parameters. Let $\boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_m$ be the parameter of the GD algorithm at trial t and $\boldsymbol{X}_t$ be the instance matrix at this trial. Then for the best comparator $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_m$ and any learning rate $\eta \geq 0$, the following holds

$$\|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2 \leq \|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}\|_F^2 = \|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - 2\eta \operatorname{tr}((\boldsymbol{W}_t - \boldsymbol{W})\boldsymbol{X}_t^\intercal) + \eta^2 \|\boldsymbol{X}_t\|_F^2,$$

where the inequality follows from the Pythagorean Theorem [HW01] and the equality follows from the descent step of the GD algorithm (see (2.6)). By rearranging terms, we have

$$\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t^\intercal) - \operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t^\intercal) \;\le\; \frac{\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2}{2\eta} + \frac{\eta\|\boldsymbol{X}_t\|_F^2}{2}.$$

Note that the LHS is the regret in trial t w.r.t. $\boldsymbol{W}$. By summing all trials, we have that the (total) regret $\mathrm{REG}_{GD} = \sum_{t=1}^T \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t^\intercal)$ is upper bounded by

$$\frac{\|\boldsymbol{W}_1 - \boldsymbol{W}\|_F^2 - \cancel{\|\boldsymbol{W}_{T+1} - \boldsymbol{W}\|_F^2}}{2\eta} + \frac{\eta\sum_{t=1}^T\|\boldsymbol{X}_t\|_F^2}{2} \;\le\; \frac{k(n-k)}{2n\eta} + \frac{\eta\sum_{t=1}^T\|\boldsymbol{X}_t\|_F^2}{2}, \quad (2.21)$$

where we used $\|\boldsymbol{W}_1 - \boldsymbol{W}\|_F^2 \le \frac{k(n-k)}{n}$ since $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_m$ and $\boldsymbol{W}_1 = \frac{n-k}{n}\boldsymbol{I}$. In the sparse instance matrix case (when $\|\boldsymbol{X}\|_F^2 \le 1$), (2.21) can be further simplified as

$$\mathrm{REG}_{GD} \;\le\; \frac{k(n-k)}{2n\eta} + \frac{\eta T}{2}.$$

By setting $\eta = \frac{k(n-k)}{nT}$, we obtain the $\sqrt{\frac{k(n-k)}{n}T}$ regret bound for the sparse instance case. In the dense instance matrix case, $\|\boldsymbol{X}_t\|_F^2 \le n$ and hence, $\mathrm{REG}_{GD} \le \sqrt{k(n-k)T}$ with $\eta = \frac{k(n-k)}{T}$. □

2.C Proof of Theorem 2.2

Theorem 2.2 gives a lower bound on the regret of the GD algorithm for the m-set problem with unit loss vectors. At each trial of the m-set problem, the online algorithm first predicts with a weight vector $\boldsymbol{w}_t \in [0,1]^n$, the coordinates of which sum to m. Then the algorithm receives a unit loss vector $\boldsymbol{\ell}_t$ and suffers loss $\boldsymbol{w}_t \cdot \boldsymbol{\ell}_t$. The GD algorithm for online PCA (2.6) specializes to the following updates of the parameter vector for learning m-sets:

$$\begin{aligned} &\text{Descent step:} \quad && \hat{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{\ell}_t, \\ &\text{Projection step:} \quad && \boldsymbol{w}_{t+1} = \operatorname{argmin}_{\boldsymbol{w} \in \boldsymbol{\mathcal{S}}_m} \|\boldsymbol{w} - \hat{\boldsymbol{w}}_{t+1}\|^2, \end{aligned} \tag{2.22}$$

where $\eta > 0$ is the learning rate and $\boldsymbol{\mathcal{S}}_m = \{\boldsymbol{w} \in [0,1]^n : \sum_{i=1}^n w_i = m\}$.

Since our lower bound for GD must hold no matter what the *fixed* learning rate η is, we construct two adversarial loss sequences: The first causes the GD algorithm to suffer large regret when η is small and the second causes large regret when η is large. Specifically, we will show that the GD algorithm suffers regret at least $\Omega(k/\eta)$ on the first sequence, and at least $\Omega(\min\{B_L, kB_L\eta\})$ on the second sequence. We will then show that the lower bound of the theorem follows by taking the maximum of these two bounds and by solving for the learning rate that minimizes this maximum. The first sequence consists of unit losses assigned to the first k experts. At each trial, the adversary gives a unit of loss to the expert (out of the first k) with the largest current weight. If the learning rate η is small, then the weights assigned to the first k experts decrease too slowly (Lemma 2.2). This causes the algorithm to suffer a substantial amount of loss on the first sequence, while the loss of the remaining m experts remains zero. The second sequence consists of unit losses assigned to the first $k+1$ experts. As before, the adversary always gives the expert with the largest weight (now out of the first $k+1$) a unit of loss. Intuitively, the GD algorithm will give high weight to the $m-1 = n-(k+1)$ loss free experts and the best out of the first $k+1$ experts. As the η gets larger, the algorithm puts more and more weight on the *current* best out of the $k+1$ experts instead of hedging its bets over all $k+1$ experts. So the algorithm becomes more and more deterministic and the adversary strategy of hitting the expert with the largest weight (out of the first $k+1$) causes the algorithm to suffer a substantial loss (Lemma 2.3). Formalizing these findings is not simple as the projection step of the GD algorithm does not have a closed form. Hence, we need to resort to the Karush-Kuhn-Tucker optimality conditions and prove a sequence of lemmas before assembling all the pieces for proving Theorem 2.2.

Let α_i be a dual variable for the constraint $w_{t+1,i} \geq 0$ $(i = 1, \ldots, n)$, β_i be a dual variable for the constraint $w_{t+1,i} \leq 1$ $(i = 1, \ldots, n)$, and γ be a dual variable for the constraint $\sum_{i=1}^n w_{t+1,i} = m$. Then the KKT conditions on the projection step of (2.22) have the following form: For $i = 1, \ldots, n$,

$$\begin{array}{ll} \text{Stationarity:} & w_{t+1,i} = -w_{t,i} - \eta \ell_{t,i} + \gamma + \alpha_i - \beta_i, \\ \text{Complementary slackness:} & w_{t+1,i}\, \alpha_i = 0, \qquad (w_{t+1,i} - 1)\beta_i = 0, \\ \text{Primal feasibility:} & \sum_{i=1}^n w_{t+1,i} = m, \qquad 0 \leq w_{t+1,i} \leq 1, \\ \text{Dual feasibility:} & \alpha_i \geq 0, \qquad \beta_i \geq 0. \end{array} \tag{2.23}$$

Note that since the projection step of (2.22) is a convex optimization problem, these conditions are necessary and sufficient for the optimality of a solution. Hence, for any intermediate weight vector $\hat{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{\ell}_t$, if a set of primal and dual variables $\boldsymbol{w}_{t+1}, \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n), \boldsymbol{\beta} = (\beta_1, \ldots, \beta_n), \gamma$ satisfy all the conditions (2.23), then they are the unique primal and dual solutions of the projection step.

We start with a special case where where the GD update (2.22) actually has a closed form solution:

Lemma 2.1. *Consider a trial of the m-set problem with n experts, when only one expert* *incurs a unit of loss. If this expert has weight w and all remaining experts have weight at most $1 - \min\{\frac{\eta}{n}, \frac{w}{n-1}\}$, then the GD algorithm with learning rate $\eta > 0$ will decrease w by $\min\{\frac{(n-1)\eta}{n}, w\}$ and increase all the other weights by $\min\{\frac{\eta}{n}, \frac{w}{n-1}\}$.*

Proof. W.l.o.g., the first expert incurs a unit of loss in trial t, i.e. $w_{t,1} = w$ and $\boldsymbol{\ell}_t = \boldsymbol{e}_1$, where $\boldsymbol{e}_1$ is the unit vector with first coordinate equal to 1 and all other coordinates equal to 0. To solve the projection step of the GD update (2.22), we distinguish two cases based on the value of $w_{t,1}$. In each case we propose a solution to the projection step and show that it is a valid solution by verifying the KKT conditions (2.23).

Case $w_{t,1} = w \geq \frac{n-1}{n}\eta$: The proposed solution is $\gamma = \frac{\eta}{n}$ and for $1 \leq i \leq n$, $\alpha_i = \beta_i = 0$,

$$w_{t+1,i} = \begin{cases} w_{t,1} - \frac{n-1}{n}\eta & \text{for } i = 1 \\ w_{t,i} + \frac{\eta}{n} & \text{for } i \geq 2 \end{cases}.$$

All KKT conditions are easy to check, except for the primal feasibility condition: $w_{t+1,i} \leq 1$, for $i \geq 2$. By the assumption of the lemma, $w_{t,i} \leq 1 - \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\}$. Since we are in the case $w_{t,1} \geq \frac{n-1}{n}\eta$, we have $w_{t,i} \leq 1 - \frac{\eta}{n}$ and therefore

$$w_{t+1,i} = w_{t,i} + \frac{\eta}{n} \leq 1 - \frac{\eta}{n} + \frac{\eta}{n} = 1.$$

We conclude that in this case, the first weight decreases by $\frac{n-1}{n}\eta$ and all the other weights increase by $\frac{\eta}{n}$.

Case $w_{t,1} = w < \frac{n-1}{n}\eta$: The proposed solution is $\gamma = \frac{w_{t,1}}{n-1}$ and for $1 \leq i \leq n$,

$\beta_i = 0$,

$$\alpha_i = \begin{cases} \eta - \frac{n}{n-1} w_{t,1} & \text{for } i = 1 \\ 0 & \text{for } i \geq 2 \end{cases}, \qquad w_{t+1,i} = \begin{cases} 0 & \text{for } i = 1 \\ w_{t,i} + \frac{w_{t,1}}{n-1} & \text{for } i \geq 2 \end{cases}.$$

Again, all KKT conditions are easy to check, except for the primal feasibility condition $w_{t+1,i} \leq 1$ for $i \geq 2$. By the assumption of the lemma $w_{t,i} \leq 1 - \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\}$. Since we are in the case $w_{t,1} < \frac{n-1}{n}\eta$, we have $w_{t,i} \leq 1 - \frac{w_{t,1}}{n-1}$ and therefore

$$w_{t+1,i} = w_{t,i} + \frac{w_{t,1}}{n-1} \leq 1 - \frac{w_{t,1}}{n-1} + \frac{w_{t,1}}{n-1} = 1.$$

We conclude that in this case, the first weight decreases by $w_{t,1}$ and all the other weights increase by $\frac{w_{t,1}}{n-1}$. Combining the above two cases proves the lemma. $\square$

Our next lemma considers the general case when the weight vector before update does not necessarily satisfy the assumption in Lemma 2.1, i.e. the weights of the experts not incurring loss may be larger than $1 - \min\{\frac{\eta}{n}, \frac{w}{n-1}\}$ (where w is the weight of the only expert incurring loss).

Lemma 2.2. *Consider a trial of the m-set problem with n experts, when only one expert incurs a unit of loss. If this expert has weight w, then the GD algorithm with learning rate $\eta > 0$ will decrease w by at most η and will not decrease the weights of any other experts. Furthermore, if any expert not incurring loss has weight at least $1 - \min\{\frac{\eta}{n}, \frac{w}{n-1}\}$, then its weight will be set to 1 by the capping constraint.*

Proof. Let $\boldsymbol{w}_t$ be the weight vector at the beginning of the trial and assume w.l.o.g. that the first expert incurs one unit of loss, i.e. $\boldsymbol{\ell}_t = \boldsymbol{e}_1$. Let $\boldsymbol{w}_{t+1}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ and γ denote the variables satisfying the KKT conditions (2.23). The lemma now states that:

$$w_{t+1,1} \geq w_{t,1} - \eta \qquad \text{and} \qquad w_{t+1,i} \geq w_{t,i}, \quad \text{for } 2 \leq i \leq n, \tag{2.24}$$

and furthermore

$$w_{t+1,i} = 1, \quad \text{for any } 2 \leq i \leq n \text{ such that } w_{t,i} \geq 1 - \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\}. \tag{2.25}$$

We first prove (2.24). By the stationarity condition of (2.23) and the assumption $\boldsymbol{\ell}_t = \boldsymbol{e}_1$, we have that

$$w_{t+1,1} - w_{t,1} \ = \ \cancel{w_{t,1}} - \eta + \alpha_1 - \beta_1 + \gamma - \cancel{w_{t,1}} \ = \ -\eta + \alpha_1 - \beta_1 + \gamma,$$

$$\text{and for } 2 \leq i \leq n\text{:} \quad w_{t+1,i} - w_{t,i} \ = \ \cancel{w_{t,1}} + \alpha_i - \beta_i + \gamma - \cancel{w_{t,1}} \ = \ \alpha_i - \beta_i + \gamma.$$

Therefore, to prove (2.24), it suffices to show $\alpha_i - \beta_i + \gamma \geq 0$ for $1 \leq i \leq n$. By the dual feasibility condition of (2.23), $\alpha_i \geq 0$ but $-\beta_i \leq 0$. However, when $-\beta_i < 0$, we have $w_{t+1,i} = 1$ by the complementary slackness condition, and therefore (2.24) holds trivially in this case (noting that $w_{t,i} \leq 1$). Now we only need to show $\gamma \geq 0$. We do this by summing $w_{t,i} - \eta\ell_{t,i} + \gamma$ over indices i such that $w_{t+1,i} > 0$:

$$\begin{aligned} \sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i} - \eta\ell_{t,i} + \gamma) \ &\overset{\substack{\alpha_i = 0 \text{ since} \\ w_{t+1,i} > 0}}{=} \ \sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i} - \eta\ell_{t,i} + \gamma + \alpha_i) \\ &\geq \ \sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i} - \eta\ell_{t,i} + \gamma + \alpha_i - \beta_i) \\ &\geq \ \sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i}) \ = \ m. \end{aligned} \tag{2.26}$$

Furthermore, since both the learning rate η and the loss vector $\boldsymbol{\ell}_t$ are non-negative, we have that for all $1 \leq i \leq n$,

$$\sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i} - \eta\ell_{t,i}) \quad \leq \quad \sum_{i:1\leq i\leq n, w_{t+1,i}>0} (w_{t,i}) \quad \leq \quad m.$$

Combining the above inequality with (2.26) implies that $\gamma \geq 0$, which completes our proof of (2.24).

Next we prove (2.25). By the stationarity condition of (2.23) and the assumption $\boldsymbol{\ell}_t = \boldsymbol{e}_1$, we have that for $2 \leq i \leq n$,

$$w_{t+1,i} \;=\; w_{t,i} - \eta \ell_{t,i} + \alpha_i - \beta_i + \gamma \;=\; w_{t,i} + \alpha_i - \beta_i + \gamma. \tag{2.27}$$

Now if we further assume that $w_{t,i} \geq 1 - \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\}$, then (2.27) is lower bounded by

$$w_{t+1,i} \;=\; w_{t,i} + \alpha_i - \beta_i + \gamma \;\geq\; 1 - \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\} + \alpha_i - \beta_i + \gamma.$$

Thus to prove (2.25), it suffices to show that $-\min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\} + \alpha_i - \beta_i + \gamma \geq 0$. By the dual feasibility condition of (2.23), $\alpha_i \geq 0$ but $-\beta_i \leq 0$. However, when $-\beta_i < 0$, then $w_{t+1,i} = 1$ follows directly from the complementary slackness condition. Therefore w.l.o.g., we assume $\beta_i = 0$. Now all that remains is to show $\gamma \geq \min\{\frac{\eta}{n}, \frac{w_{t,1}}{n-1}\}$, for which we distinguish the following 2 cases.

Case $w_{t+1,1} > 0$: We will show $\gamma \geq \frac{\eta}{n}$ for this case. First note that

$$m \overset{(2.26)}{\leq} \sum_{i:1 \leq i \leq n, w_{t+1,i} > 0} (w_{t,i} - \eta \ell_{t,i} + \gamma) \overset{\gamma \geq 0}{\leq} \sum_{i:1 \leq i \leq n, w_{t+1,i} > 0} (w_{t,i} - \eta \ell_{t,i}) + n\gamma. \tag{2.28}$$

Now since we assume $w_{t+1,1} > 0$ and $\boldsymbol{\ell}_t = \boldsymbol{e}_1$, the first term on RHS of (2.28) is upper bounded by:

$$\sum_{i:1 \leq i \leq n, w_{t+1,i} > 0} (w_{t,i} - \eta \ell_{t,i}) \quad = \quad \sum_{i:1 \leq i \leq n, w_{t+1,i} > 0} (w_{t,i}) - \eta \quad \leq \quad m - \eta.$$

Together, we get $m \leq n\gamma - \eta + m$, and this gives $\gamma \geq \frac{\eta}{n}$.

Case $w_{t+1,1} = 0$: We will show $\gamma \geq \frac{w_{t,1}}{n-1}$ for this case. Since $w_{t+1,1} = 0$, the summation $\sum_{i:1 \leq i \leq n, w_{t+1,i}>0} (w_{t,i} - \eta \ell_{t,i} + \gamma)$ does not include the case $i = 1$, i.e.

$$\sum_{i:1 \leq i \leq n, w_{t+1,i}>0} (\hat{w}_{t,i} - \eta \ell_{t,i} + \gamma) \quad = \quad \sum_{i:2 \leq i \leq n, w_{t+1,i}>0} (\hat{w}_{t,i} - \eta \ell_{t,i} + \gamma).$$

Therefore, (2.28) can be tightened as follows:

$$m \overset{(2.26)}{\leq} \sum_{i:2 \leq i \leq n, w_{t+1,i}>0} (w_{t,i} - \eta \ell_{t,i} + \gamma) \overset{\gamma \geq 0}{\leq} \sum_{i:1 \leq i \leq n, w_{t+1,i}>0} (w_{t,i} - \eta \ell_{t,i}) + (n-1)\gamma.$$

Again, by the assumption $\boldsymbol{\ell}_t = \boldsymbol{e}_1$, we have

$$\sum_{i:2 \leq i \leq n, w_{t+1,i}>0} (w_{t,i} - \eta \ell_{t,i}) \quad = \sum_{i:2 \leq i \leq n, w_{t+1,i}>0} (w_{t,i}) \quad \leq \quad m - w_{t,1}.$$

Together, we get $m \leq (n-1)\gamma + m - w_{t,1}$, which gives $\gamma \geq \frac{w_{t,1}}{n-1}$ and completes the proof. $\square$

Our third lemma lower bounds the loss of the GD algorithm with respect to a particular adversarial loss sequence of n trials (instead of the above lower bounds for single trials). We argue this lower bound for the special case of the m-set problem when $m = 1$, i.e. the vanilla expert setting. As we shall see shortly in the main proof of Theorem 2.2, the lower bound the general m-set problem degenerates into this special case for a certain loss sequence. Note that the assumptions of Lemma 2.1 are always met when $m = 1$, because in this case any expert not incurring loss has weight at most $1 - w$, where w is the weight of the expert incurring loss.

Lemma 2.3. *Consider the m-set problem with n experts, and $m = 1$. If at each trial, only the expert with the largest weight incurs a unit of loss, then after n consecutive such trials, the GD algorithm with learning rate $\eta > 0$ suffers loss at least $1 + \frac{1}{32}\min\{n\eta, 1\}$.*

Proof. First notice that when $m = 1$, the largest of the n expert weights at each trial is at least $\frac{1}{n}$. Therefore, any algorithm suffers total loss at least 1 in n trials. To show the extra loss of $\frac{1}{32}\min\{n\eta, 1\}$, we claim that in at least $\frac{n}{4}$ of these n trials, the largest expert weight assigned by the GD algorithm is at least $\frac{1}{n} + \frac{1}{8}\min\{\eta, \frac{1}{n}\}$. This claim is proved as follows.

Let $\eta' = \min\{\eta, \frac{1}{n}\}$ and t_0 be the first trial that the largest expert weight of the trial is less than $\frac{1}{n} + \frac{1}{8}\eta'$. If $t_0 > \frac{n}{4}$, the claim holds trivially. Hence, we assume $t_0 \leq \frac{n}{4}$. Now call any expert with weight at least $\frac{1}{n} - \frac{1}{8}\eta'$ at trial t_0 a *candidate*. We will show that the number of candidates s is at least $\frac{n}{2}$. To show this we first upper bound the expert weights at trial t_0 as follows:

$$\begin{aligned}\text{sum of non-candidates' weights} \quad &\leq \quad (n-s)\left(\frac{1}{n} - \frac{1}{8}\eta'\right),\\ \text{sum of candidates' weights} \quad &\leq \quad s\left(\frac{1}{n} + \frac{1}{8}\eta'\right).\end{aligned}$$

The first inequality follows from the fact that non-candidates have weight at most $\frac{1}{n} - \frac{1}{8}\eta'$ and the second inequality follows from the definition of t_0, i.e. the maximum weight at that trial is less than $\frac{1}{n} + \frac{1}{8}\eta'$. Now, since all the expert weights at a trial sum to 1, we have

$$1 \leq s\left(\frac{1}{n} + \frac{1}{8}\eta'\right) + (n-s)\left(\frac{1}{n} - \frac{1}{8}\eta'\right) = 1 + \frac{s}{4}\eta' - \frac{n}{8}\eta',$$

which gives $s \geq \frac{n}{2}$ since $\eta' \geq \eta > 0$.

Next, we show that at trial $t_0 + \frac{n}{4}$, there will be a subset of at least $\frac{n}{4}$ candidates whose weight will be at least the larger value of $\frac{1}{n} + \frac{1}{8}\eta'$. First recall that at each trial, only one expert incurs a unit of loss. Therefore, in the $\frac{n}{4}$ trials from t_0 to $t_0 + \frac{n}{4} - 1$, there will be at least $\frac{n}{2} - \frac{n}{4} = \frac{n}{4}$ candidates that do not incur any loss. By Lemma 2.1, the weight of an expert not incurring loss is increased at each trial by $\min\{\frac{\eta}{n}, \frac{w}{n-1}\}$, where w is the weight of the expert incurring loss at that trial. Note that $w \geq \frac{1}{n}$ always hold since the expert incurring loss has the largest weight among the n experts. Therefore, at trial $t_0 + \frac{n}{4}$, each of the $\frac{n}{4}$ candidates that do incur any loss from trial t_0 to trial $t_0 + \frac{n}{4} - 1$ has weight at least:

$$\overbrace{\frac{1}{n} - \frac{1}{8}\eta'}^{\substack{\text{lower bound on the}\\ \text{weight at trial } t_0}} \qquad + \qquad \overbrace{\frac{n}{4}\min\{\frac{\eta}{n}, \frac{w_t}{n-1}\}}^{\substack{\text{lower bound on the increase}\\ \text{from trial } t_0 \text{ to trial } t_0 + \frac{n}{4} - 1}}$$

$$\overset{\frac{w_t}{n-1} \geq \frac{1}{n^2}}{\geq} \quad \frac{1}{n} - \frac{1}{8}\eta' + \frac{n}{4}\min\{\frac{\eta}{n}, \frac{1}{n^2}\} = \frac{1}{n} + \frac{\eta'}{8}.$$

Finally, consider the next $\frac{n}{4}$ trials from $t_0 + \frac{n}{4}$ to $t_0 + \frac{n}{2} - 1$. (The game must have more than $t_0 + \frac{n}{2}$ trials, since we assume $t_0 \leq \frac{n}{4}$.) The maximum weights at these trials are always at least $\frac{1}{n} + \frac{1}{8}\eta'$, because only one expert incurs loss at a time, and the weights of the remaining experts are never decreased. This completes the proof of the claim and the lemma. $\square$

Now we are ready to give the lower bound on the regret of the GD algorithm for the m-set problem. For the sake of readability, we repeat the statement of Theorem 2.2 below:

Theorem 2.2 *Consider the m-set problem with $k \leq n/2$ and unit loss vectors. Then for any fixed learning rate η, the GD algorithm* (2.22) *can be forced to have regret at* *least* $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.

Proof. Theorem 2.7 gives a lower bound of $\Omega(\sqrt{B_L m \log \frac{n}{m}} + m \log \frac{n}{m})$ that holds for any algorithm. This lower bound is at least $\Omega(k)$ since $m \log \frac{n}{m} = m \log(\frac{k}{m} + 1) \geq k$. Hence to prove this theorem, we only need to show a lower bound of $\Omega(\{\min\{B_L, k\sqrt{B_L}\})$, where B_L is the loss budget (defined in (2.9)). Also, w.l.o.g., assume $B_L \geq 4k$ since when $B_L \leq 4k$, the claimed bound is in fact $\Omega(k)$, which always holds as we just argued.

The hard part (deferred to later) in proving the $\Omega(\{\min\{B_L, k\sqrt{B_L}\})$ lower bound for GD is to show that the algorithm suffers regret at least $\Omega(k/\eta)$ and $\Omega(\min\{B_L, kB_L\eta\})$ on two different loss sequences, respectively. Clearly, it follows that the regret of GD is then at least the maximum of these two bounds. By a case analysis, one can show that $\max\{a, \min\{b, c\}\} \geq \min\{b, \max\{a, c\}\}$ for any $a, b, c \in$ R. (We prove this as Lemma 2.11 in Appendix 2.I.) Therefore we get the lower bound of $\Omega(\min\{B_L, \max\{k/\eta, kB_L\eta\}\})$. The lower bound for GD with any fixed learning rate now follows from fact that $\max\{k/\eta, kB_L\eta\}$ is minimized at $\eta = \Theta(1/\sqrt{B_L})$. The value of the lower bound with this choice of η is the target lower bound of $\Omega(k\sqrt{B_L})$.

We still need to describe the two loss sequences and prove the claimed lower bounds on the regret. The first loss sequence forces GD to suffer regret $\Omega(k/\eta)$. It consists of $\left\lfloor \frac{km}{n\eta} \right\rfloor + 1$ trials in which only the first k experts incur losses. More precisely, at each trial, the expert with the largest weight (within the first k experts) incurs one unit of loss (In the case of tied weights, only the expert with the smallest index incurs loss). The last m experts have loss 0. Therefore the regret is simply the total loss of the GD algorithm. The loss of the algorithm at each trial is equal to the largest weight of the first k experts. Therefore the loss is lower bounded by the average of the first k weights. With a uniform initial weight vector, this average is $\frac{m}{n}$ at the beginning of the first trail, and by Lemma 2.2, it is decreased by at most $\frac{\eta}{k}$ after each of the following $\left\lfloor \frac{km}{n\eta} \right\rfloor + 1$ trials. Therefore, at the beginning of trial t, the average is at least $\frac{m}{n} - (t-1)\frac{\eta}{k}$. Summing up the arithmetic series from trial 1 to trial $\left\lfloor \frac{km}{n\eta} \right\rfloor + 1$ gives the following lower bound on the total loss of GD:

$$\frac{1}{2}\left(\left\lfloor \frac{km}{n\eta} \right\rfloor + 1\right)\left(\frac{m}{n} + \frac{m}{n} - \left(\left\lfloor \frac{km}{n\eta} \right\rfloor + 1 - 1\right)\frac{\eta}{k}\right) \overset{\frac{m}{n} \geq \frac{1}{2}}{\geq} \frac{1}{4}\left(\left\lfloor \frac{k}{2\eta} \right\rfloor + 1\right) = \Omega\left(\frac{k}{\eta}\right).$$

Now we describe the second loss sequence which forces the GD algorithm to suffer regret $\Omega(\min\{B_L, kB_L\eta\})$. The sequence consists of $(k+1)B_L$ trials, where the expert with the largest weight among first $k+1$ incurs a unit of loss. The best comparator of this sequence consists of the last $m-1$ experts that have 0 total loss and the best of the first $k+1$ experts which has total loss at most B_L.

Next we lower bound the loss of GD with respect to this loss sequence. First observe, that the last $m-1$ experts do not incur any loss in the $(k+1)B_L$ trials. Therefore their weight may increase (from their initial value of $\frac{m}{n}$), but at any trial the weight of these experts always have the same value. The value of this block of equal weights is always the maximum weight of any expert, since the weight value of the block is never decreased by the algorithm. More precisely, at each trial the block's value is increased as given in Lemma 2.1, until the values becomes 1 at trial t_{cap} and stay at 1 till the end of the game. If no such trial t_{cap} exists (i.e. the value of the block remains less than 1 at the end of the game), then let $t_{cap} = \infty$. In the degenerate case when $m = 1$ (i.e. the block has size $m - 1 = 0$), we simply set $t_{cap} = 1$ from the beginning.

Depending on the value of t_{cap}, we distinguish two cases in which GD suffers loss at least $B_L + \Omega(B_L)$ and $B_L + \Omega(\min\{B_L, kB_L\eta\})$, respectively.

Case $t_{cap} > (k+1)B_L/2$: We will show that GD suffers loss at least $B_L+\Omega(B_L)$ in this case. First recall that at the beginning of the proof we assumed $B_L \geq 4k$. Therefore in the case $t_{cap} > (k+1)B_L/2$ we have $t_{cap} > 4$. From our definition of t_{cap} this means that $m \geq 2$. Next we argue that since $t_{cap} > (k+1)B_L/2$, we have $\eta \leq \frac{1}{k+1}$. Let W_t denote the sum of the first $k+1$ weights at trial t and w_t be their maximum. By Lemma 2.1, we know in each trial prior to t_{cap} the weight w_t of the expert incurring loss is decreased by $\min\{\frac{(n-1)\eta}{n}, w_t\}$ and all other weights are increased by $\min\{\frac{\eta}{n}, \frac{w_t}{n-1}\}$. Since the expert incurring loss is always one of the first $k+1$ experts, we have that in each trial prior to t_{cap} the total weight W_t is decreased by at least

$$\min\left\{\frac{(n-1)\eta}{n}, w_t\right\} - k\min\left\{\frac{\eta}{n}, \frac{w_t}{n-1}\right\} \;\geq\; \frac{m-1}{n}\min\{\eta, w_t\} \;\geq\; \frac{m-1}{n}\min\left\{\eta, \frac{1}{k+1}\right\}.$$

The second inequality follows from the fact that since w_t is the largest of the first $k+1$ expert weights, it must be at least $\frac{1}{k+1}$. Together with the fact that $W_1 = \frac{(k+1)m}{n}$, we have

$$W_{(k+1)B_L/2} \leq \frac{(k+1)m}{n} - \frac{(k+1)B_L}{2}\frac{m-1}{n}\min\left\{\eta, \frac{1}{k+1}\right\}. \tag{2.29}$$

Now if $\eta \geq \frac{1}{k+1}$, the upper bound (2.29) becomes $\frac{(k+1)m}{n} - \frac{(m-1)B_L}{2n}$, which can be further upper bounded by $\frac{m}{n}$ using the fact $m \geq 2$ and the assumption $B_L \geq 4k$. However, the upper bound of $W_{(k+1)B_L/2} \leq \frac{m}{n}$ is less than 1 and all W_t are at least 1 since $m - W_t$

is the total weight of the last $m-1$ experts which is at most $m-1$. Therefore we have $\eta < \frac{1}{k+1}$ in this case.

Now we lower bound the loss of GD by lower bounding the average weight $W_t/(k+1)$. We have $\eta < \frac{1}{k+1}$ and $t_{cap} > (k+1)B_L/2$. Also by Lemma 2.1, W_t decreases by exactly $\frac{(m-1)\eta}{n}$ at each trial for $1 \le t \le (k+1)B_L/2$. Therefore the total average weight in trials 1 through $(k+1)B_L/2$ is at least

$$\frac{1}{2}\frac{1}{k+1}\left(\frac{(k+1)m}{n}+1\right)\frac{(k+1)B_L}{2} = \left(\frac{(k+1)m}{n}+1\right)\frac{B_L}{4}. \tag{2.30}$$

Now with $m \ge 2$, $k \ge 1$ and $n = m+k$, it is easy to verify that $\frac{(k+1)m}{n}$ is at least $1+\Omega(1)$, which along with (2.30) results in a $\frac{B_L}{2}+\Omega(B_L)$ lower bound on the loss of GD for $1 \le t \le (k+1)B_L/2$. In trials $(k+1)B_L/2 < t \le (k+1)B_L$, GD suffers loss at least $\frac{(k+1)B_L}{2}\frac{1}{k+1} = \frac{B_L}{2}$ since the weight of the expert incurring loss is at least $\frac{1}{k+1}$. Thus in trial 1 through $(k+1)B_L$ the loss of GD is at least $B_L + \Omega(B_L)$ which concludes the proof of the case $t_{cap} \ge (k+1)B_L/2$.

Case $t_{cap} \le (k+1)B_L/2$: We will show that GD suffers total loss at least $B_L+\Omega(\min\{B_L, kB_L\eta\})$ in this case. Since GD suffers loss at least $B_L/2$ in the first $(k+1)B_L/2$ trials, it suffices to show that GD suffers loss at least $B_L/2+\Omega(\min\{B_L, kB_L\eta\})$ in trials $(k+1)B_L/2+1$ through $(k+1)B_L$. First note that since $t_{cap} \le (k+1)B_L/2$, in each of these trials, the weights of the $m-1$ loss free experts have reached the cap 1. This means that GD updates the weights of the first $k+1$ experts as in the vanilla expert setting (i.e. $m=1$). Therefore by Lemma 2.3, the loss of GD in the second $(k+1)B_L/2$ trials is at least $\frac{B_L}{2}(1+\frac{1}{32}\min\{(k+1)\eta, 1\}) = \frac{B_L}{2}+\Omega(\min\{B_L, kB_L\eta\})$.

We conclude that for the second loss sequence, the loss of the best comparator is at most B_L and the loss of GD is at least $B_L + \Omega(\min\{B_L, kB_L\eta\})$. Therefore, the regret of GD is at least $\Omega(\min\{B_L, kB_L\eta\})$ for the second loss sequence and this completes our proof of the theorem. $\square$

2.D Proof of Theorem 2.3

Theorem 2.3 gives a lower bound on the regret of the FRL-GD algorithm for the m-set problem with unit loss vectors. In this case, the FRL-GD algorithm (2.18) specializes to the following:

$$\begin{aligned} \text{Follow the regularized leader:} \quad & \hat{\boldsymbol{w}}_{t+1} = -\eta \sum_{q=1}^{t} \boldsymbol{\ell}_q, \\ \text{Projection step:} \quad & \boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w} \in \boldsymbol{\mathcal{S}}_m} \|\boldsymbol{w} - \hat{\boldsymbol{w}}_{t+1}\|^2. \end{aligned} \tag{2.31}$$

The proof has the same structure as the lower bound for the GD algorithm (Appendix 2.C). Again we use two adversarial loss sequences (one for low and high learning rates) and give three technical lemmas that reason with the KKT conditions. The details are different because the intermediate weight vector $\hat{\boldsymbol{w}}_{t+1}$ has a different form than for vanilla GD. The KKT conditions are the same as the KKT condition for GD (2.22) except for a slight change in the stationarity condition. For $i = 1, \dots, n$,

$$\begin{aligned} &\text{Stationarity:} && w_{t+1,i} = -\eta \ell_{\le t,i} + \gamma + \alpha_i - \beta_i, \\ &\text{Complementary slackness:} && w_{t+1,i}\alpha_i = 0, \quad (w_{t+1,i} - 1)\beta_i = 0, \\ &\text{Primal feasibility:} && \textstyle\sum_{i=1}^{n} w_{t+1,i} = m, \qquad 0 \le w_{t+1,i} \le 1, \\ &\text{Dual feasibility:} && \alpha_i \ge 0, \qquad \beta_i \ge 0, \end{aligned} \tag{2.32}$$

where $\ell_{\le t,i} = \sum_{q=1}^{t} \ell_{q,i}$ is the cumulative loss of expert i up to trail t. Again we prove three technical lemmas before assembling them into the main proof.

Lemma 2.4. *Consider the m-set problem with n experts, where at the beginning of trial $t+1$, each of the first $k+1$ experts (where $k = n-m$) has incurred the same cumulative loss ℓ, and all the remaining experts are loss free, i.e.*

$$\ell_{\le t,i} = \begin{cases} \ell & \textit{for } i \le k+1 \\ 0 & \textit{for } i > k+1 \end{cases}.$$

Now the FRL-GD algorithm predicts at trail $t+1$ with the weight vector $\boldsymbol{w}_{t+1}$ given by:

$$w_{t+1,i} = \begin{cases} \textit{if } \ \eta\ell < \frac{k}{k+1} \textit{ then } \begin{cases} \frac{m-\eta\ell(m-1)}{n} & \textit{for } i \le k+1 \\ \frac{m+\eta\ell(k+1)}{n} & \textit{for } i > k+1 \end{cases} \\ \textit{if } \ \eta\ell \ge \frac{k}{k+1} \textit{ then } \begin{cases} \frac{1}{k+1} & \textit{for } i \le k+1 \\ 1 & \textit{for } i > k+1 \end{cases} \end{cases}.$$

Proof. We prove this lemma by verifying the KKT conditions (2.32). If $\eta\ell < \frac{k}{k+1}$, we have:

$$1 \; > \; \frac{m-\eta\ell(m-1)}{n} \; > \; 0, \quad \text{and} \quad 0 \; < \; \frac{m+\eta\ell(k+1)}{n} \; < \; 1.$$

Therefore $0 < w_{t+1,i} < 1$, for all i. By taking $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$, and $\gamma = \frac{m+\eta\ell(k+1)}{n}$, the KKT conditions can easily be verified to hold. If $\eta\ell \ge \frac{k}{k+1}$, the KKT conditions are satisfied by taking $\alpha_i = 0$ for $i \le k+1$ and $\alpha_i = \frac{k}{k+1} - \eta\ell$ for $i > k+1$, $\boldsymbol{\beta} = \mathbf{0}$ and $\gamma = \frac{1}{k+1} + \eta\ell$. □

Lemma 2.5. *Consider a trial of the m-set problem with n experts, when only one expert incurs a unit of loss. Then the FRL-GD algorithm with learning rate $\eta > 0$ decreases the weight of this expert by at most η and none of the other weights are decreased in this trial.*

Proof. Let $\boldsymbol{\ell}_{\leq t-1}$ be the cumulative loss vector at the beginning of the trial, and let $\boldsymbol{w}_t, \boldsymbol{\alpha}_t, \boldsymbol{\beta}_t$ and γ_t be the corresponding primal and dual variables satisfying KKT conditions (2.32) with respect to $\boldsymbol{\ell}_{\leq t-1}$. W.l.o.g., we assume the first expert incurs a unit of loss, i.e. $\boldsymbol{\ell}_{\leq t} = \boldsymbol{\ell}_{\leq t-1} + \boldsymbol{e}_1$. Let $\boldsymbol{w}_{t+1}, \boldsymbol{\alpha}_{t+1}, \boldsymbol{\beta}_{t+1}$ and γ_{t+1} denote the variables satisfying the KKT conditions with respect to the updated loss vector $\boldsymbol{\ell}_{\leq t}$. The lemma now states that $w_{t+1,1} - w_{t,1} \geq -\eta$.

The lemma holds trivially when $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$. When $\boldsymbol{w}_{t+1} \neq \boldsymbol{w}_t$, we first show that $\gamma_{t+1} \geq \gamma_t$. Since both $\boldsymbol{w}_t$ and $\boldsymbol{w}_{t+1}$ sum to m, there must be an expert j, such that $w_{t,j} < w_{t+1,j}$. By the stationarity condition of (2.32), we have:

$$0 \quad < \quad w_{t+1,j} - w_{t,j} \quad = \quad (-\eta\ell_{\leq t,j} + \alpha_{t+1,j} - \beta_{t+1,j} + \gamma_{t+1}) - (-\eta\ell_{\leq t-1,j} + \alpha_{t,j} - \beta_{t,j} + \gamma_t),$$

or, equivalently,

$$\gamma_{t+1} - \gamma_t \quad > \quad \eta(\ell_{\leq t,j} - \ell_{\leq t-1,j}) \ + \ (\alpha_{t,j} - \alpha_{t+1,j}) \ + \ (\beta_{t+1,j} - \beta_{t,j}). \tag{2.33}$$

Since $w_{t+1,j} > w_{t,j}$, and the weights must be nonnegative, we have $w_{t+1,j} > 0$, and thus $\alpha_{t+1,j} = 0$ due to the complementary slackness condition of (2.32). Since $\alpha_{t,j}$ must be nonnegative due to the dual feasibility condition of (2.32), we have $\alpha_{t,j} \geq \alpha_{t+1,j}$. A similar argument gives $\beta_{t+1,j} \geq \beta_{t,j}$. Moreover, since $\ell_{\leq t,j} - \ell_{\leq t-1,j} \geq 0$ (due to $\boldsymbol{\ell}_{\leq t} = \boldsymbol{\ell}_{\leq t-1} + \boldsymbol{e}_1$), the RHS of (2.33) is nonnegative, and thus $\gamma_{t+1} \geq \gamma_t$.

By the stationary condition of (2.32), we have:

$$\begin{aligned} w_{t+1,1} - w_{t,1} &= (-\eta\ell_{\le t,1} + \alpha_{t+1,1} - \beta_{t+1,1} + \gamma_{t+1}) \ - \ (-\eta\ell_{\le t-1,1} + \alpha_{t,1} - \beta_{t,1} + \gamma_t) \\ &= -\eta \ + \ (\gamma_{t+1} - \gamma_t) \ + \ (\alpha_{t+1,1} - \alpha_{t,1}) \ + \ (\beta_{t,1} - \beta_{t+1,1}), \qquad (2.34) \end{aligned}$$

where we used $\ell_{\le t,1} = \ell_{\le t-1,1} + 1$. If $\alpha_{t,1} \neq 0$, then $w_{t,1} = 0$ due to complementary slackness, and the lemma trivially holds. Similarly if $\beta_{t+1,1} \neq 0$, then $w_{t+1,1} = 1$, and again the lemma holds trivially. Thus, we may assume that $\alpha_{t,1} = \beta_{t+1,1} = 0$. However then (2.34) becomes

$$w_{t+1,1} - w_{t,1} \quad = \quad -\eta \ + \ (\gamma_{t+1} - \gamma_t) \ + \ \alpha_{t+1,1} \ + \ \beta_{t,1} \quad \ge \quad -\eta.$$

We now show the second statement of the lemma, that $w_{t+1,i} \ge w_{t,i}$ for all $i > 1$. First note that if $\alpha_{t,i} > 0$, then by the complementary slackness condition of (2.32), $w_{t,i} = 0$, and the statement trivially holds. Similarly, if $\beta_{t+1,i} > 0$, then by the complementary slackness condition, $w_{t+1,i} = 1$, and, again the statement trivially holds. Therefore we prove the stamement assuming that $\alpha_{t,i} = 0$ and $\beta_{t+1,i} = 0$. Since $\ell_{\le t,i} = \ell_{\le t-1,i}$, the complementary slackness condition of (2.32) implies:

$$\begin{aligned} w_{t+1,i} - w_{t,i} &= (\cancel{-\eta\ell_{\le t,i}} + \alpha_{t+1,i} - \beta_{t+1,i} + \gamma_{t+1}) \ - \ (\cancel{-\eta\ell_{\le t-1,i}} + \alpha_{t,i} - \beta_{t,i} + \gamma_t) \\ &= (\alpha_{t+1,i} - \underbrace{\alpha_{t,i}}_{=0}) \ + \ (\beta_{t,i} - \underbrace{\beta_{t+1,i}}_{=0}) \ + \ \underbrace{(\gamma_{t+1} - \gamma_t)}_{\ge 0} \\ &\ge \ \alpha_{t+1,i} + \beta_{t,i} \ \ge \ 0, \end{aligned}$$

where the last inequality is by the dual feasibility condition of (2.32). This finishes the proof. □

Lemma 2.6. *Consider the m-set problem with n experts, and $m = 1$. Assume at the end of trial t, the cumulative losses of all experts are the same. Assume further that the loss sequence in trials $t+1, \ldots, n$ is $\boldsymbol{\ell}_{t+1} = \boldsymbol{e}_1, \boldsymbol{\ell}_{t+2} = \boldsymbol{e}_2, \ldots, \boldsymbol{\ell}_{t+n} = \boldsymbol{e}_n$, i.e. each expert subsequently incurs a unit of loss. Then the cumulative loss incurred by the FRL-GD algorithm in iterations $t+1, \ldots, n$ is at least $1 + \frac{1}{4}\min\{n\eta, 1\}$.*

Proof. The proof goes by providing primal and dual variables satisfying the KKT conditions (2.32). Since the solution $\boldsymbol{w}_{t+1}$ to (2.32) does not change if we shift all cumulative losses $\ell_{\leq t,i}$ by a constant we can assume w.l.o.g. that the cumulative loss of all experts at the end of trial t is 0.

Take trial $t+j+1$ ($j \geq 0$), at the beginning of which each of the first j experts have already incurred a unit of loss and the remaining $n-j$ experts are loss free. If $\eta \leq \frac{1}{n-j}$, then the KKT conditions (2.32) are satisfied by taking $\alpha_i = \beta_i = 0$ for all $i = 1, \ldots, n$, $\gamma = \frac{j}{n}\eta + \frac{1}{n}$, and

$$w_{t+j+1,i} = \begin{cases} \frac{1}{n} - \frac{n-j}{n}\eta & \text{for } i \leq j \\ \\ \frac{1}{n} + \frac{j}{n}\eta & \text{for } i > j \end{cases} .$$

In this trial, expert $j+1$ incurs a unit of loss, and hence the algorithm's loss is $\frac{1}{n} + \frac{j}{n}\eta$.

If $\eta > \frac{1}{n-j}$, then the KKT conditions (2.32) are satisfied by taking $\gamma = \frac{1}{n-j}$ and for $1 \leq i \leq n$, $\beta_i = 0$,

$$w_{t+j+1,i} = \begin{cases} 0 & \text{for } i \leq j \\ \\ \frac{1}{n-j} & \text{for } i > j \end{cases}, \qquad \alpha_i = \begin{cases} \eta - \frac{1}{n-j} & \text{for } i \leq j \\ \\ 0 & \text{for } i > j \end{cases}.$$

The loss of the algorithm in such a case is $\frac{1}{n-j}$.

Thus depending on η, the algorithm's loss at trial $t+j+1$ is equal to

$$\begin{cases} \frac{1}{n}+\frac{j}{n}\eta & \text{if } \eta \le \frac{1}{n-j} \\ \frac{1}{n-j} = \frac{1}{n}+\frac{j}{n}\frac{1}{n-k} & \text{if } \eta > \frac{1}{n-j} \end{cases},$$

which can be concisely written as: $\frac{1}{n}+\frac{j}{n}\min\left\{\eta, \frac{1}{n-j}\right\}$. Summing the above over $j = 0,\ldots,n$ gives the cumulative loss of the algorithm incurred at trials $t+1,\ldots,t+n$:

$$\begin{aligned} \sum_{j=0}^{n-1} \frac{1}{n}+\frac{j}{n}\min\left\{\eta, \frac{1}{n-j}\right\} \quad &\ge \quad \sum_{j=0}^{n-1} \frac{1}{n}+\frac{j}{n}\min\left\{\eta, \frac{1}{n}\right\} \\ &= \quad 1+\frac{n-1}{2}\min\left\{\eta, \frac{1}{n}\right\} \\ &\ge \quad 1+\frac{1}{4}\min\{\eta n, 1\}, \end{aligned}$$

where the last inequality is due to $n-1 > \frac{n}{2}$ for $n \ge 2$. □

We are now ready to give the proof of Theorem 2.3:

Theorem 2.3 *Consider the m-set problem with $k \le n/2$ and unit loss vectors. Then for any fixed learning rate η, the FRL-GD algorithm* (2.31) *can be forced to have regret at least* $\Omega(\max\{\min\{B_L, k\sqrt{B_L}\}, k\})$.

Proof. Theorem 2.7 gives a lower bound of $\Omega(\sqrt{B_L m \log\frac{n}{m}}+m\log\frac{n}{m})$ that holds for any algorithm. This lower bound is at least $\Omega(k)$ since $m\log\frac{n}{m} = m\log(\frac{k}{m}+1) \ge k$. Hence to prove this theorem, we only need to show a lower bound of $\Omega(\{\min\{B_L, k\sqrt{B_L}\})$. Similarly as in the proof of Theorem 2.2, we show this in two steps: First, we give two loss sequences that result in the regret of FRL-GD at least $\Omega(k/\eta)$ and $\Omega(\min\{B_L, kB_L\eta\})$, respectively. Then, the lower bound follows by taking the maximum between the two lower bounds.

The first loss sequence is exactly the same as in the proof of Theorem 2.2, i.e. the sequence consists of $\left\lfloor \frac{km}{n\eta} \right\rfloor + 1$ trials and in each trial, the expert with the largest weight (within the first k experts) incurs one unit of loss. With Lemma 2.5 in place of Lemma 2.2, one can easily show an $\Omega(k/\eta)$ regret lower bound for FRL-GD by repeating the argument from the proof of Theorem 2.2.

Now we describe the second loss sequence which forces the FRL-GD algorithm to suffer regret $\Omega(\min\{(B_L), kB_L\eta\})$. The sequence consists of B_L "rounds", and each round consist of $k+1$ trials (so that there are $(k+1)B_L$ trials in total). In each round, one unit of loss is given alternately to each of the first $k+1$ experts, one at a time. In other words, in trial t, the loss vector $\boldsymbol{\ell}_t$ equals to $\boldsymbol{e}_r$ where $r = t \mod (k+1)$. The best comparator of this sequence consists of the last $m-1$ loss free experts and any of the first $k+1$ experts, which incurs cumulative loss B_L.

To lower bound the loss of the algorithm, first notice that in each round, each of the first $k+1$ experts incurs exactly one unit of loss. The sum of weights of these experts at the beginning of a round lower bounds the algorithm's loss in this round. This is because the weight of a given expert cannot decrease if the expert does not incur any loss (Lemma 2.5); hence, the weight of a given expert at a trial, in which that expert receives a unit of loss, will be at least as large as the weight of that expert at the beginning of a round. Since the weights are initialized uniformly, this sum is $m(k+1)/n$ before round 1, and by Lemma 2.4, each of the following rounds decreases it by $(m-1)(k+1)\eta/n$ until it is lower capped at 1 (Since the total sum of the weights is m, and none of the remaining $m-1$ weights can exceed 1, the sum of weights of the first $k+1$ experts must be at least 1).

We first assume that after $B_L/2$ rounds, this sum is strictly larger than 1 which means the sum decreases as an arithmetic series for all the first $B_L/2$ rounds and the algorithm's loss can be lower bounded by

$$\frac{1}{2}(m(k+1)/n+1)\frac{B_L}{2} \overset{\substack{\text{Use the same}\\ \text{argument as in (2.30)}}}{=} B_L/2+\Omega(B_L).$$

Since the sum of the first $k+1$ weights at the beginning of any trial is at least 1, the algorithm incurs loss at least $B_L/2$ in the remaining $B_L/2$ rounds. Summing up the algorithm's loss on both halves of the sequence, we get a regret lower bound of $\Omega(B_L)$.

Now consider the case, when after the first $B_L/2$ rounds, the sum of the first $k+1$ weights is 1. This implies that the weights of $m-1$ remaining experts are all equal to 1, and will stay at this value, since only the first $k+1$ experts incur any loss (and, by Lemma 2.5, the weight of an expert cannot decrease if that expert does not incur any loss). Thus, we can disregard the loss free $m-1$ experts, and in the remaining $B_L/2$ rounds, the first $k+1$ expert weights are updated as in the m-set problem with $m=1$. Notice that the algorithm suffers loss at least $B_L/2$ in the first $B_L/2$ rounds and by Lemma 2.6, suffers loss at least $B_L/2 + B_L \min\{(k+1)\eta, 1\}/8$ in the second $B_L/2$ rounds. Summing up the algorithm's loss on both halves of the sequence, we get a regret lower bound of $\Omega(\min\{B_L, kB_l\eta\})$. $\square$

2.E A Discussion on the Final Parameter of FRL-GD

In this appendix, we show that the final parameter matrix of the FRL-GD algorithm essentially contains the solution to the batch PCA problem. First recall that given n dimensional data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$, the batch version of the k-PCA problem is solved by the eigenvectors of the k largest eigenvalues of the covariance matrix $\boldsymbol{C} = \sum_{t=1}^{T} \boldsymbol{x}_t \boldsymbol{x}_t^\intercal$. Let $\boldsymbol{W}_{T+1}$ be the final parameter matrix of the FRL-GD algorithm when the instance matrices are $\boldsymbol{X}_1 = \boldsymbol{x}_1\boldsymbol{x}_1^\intercal, \dots, \boldsymbol{X}_T = \boldsymbol{x}_T\boldsymbol{x}_T^\intercal$. We will show that the eigenvectors of the $m = n - k$ largest eigenvalues of $\boldsymbol{W}_{T+1}$ are the same as the eigenvectors of the m largest eigenvalues of the negated covariance matrix $-\boldsymbol{C}$. Thus, by computing the complementary subspace of rank k, one finds the solution of the batch PCA problem with respect to data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$.

Recall that the final parameter $\boldsymbol{W}_{T+1}$ of FRL-GD is the projection of the $-\boldsymbol{C}$ into the parameter set $\boldsymbol{\mathcal{W}}_m$:

$$\boldsymbol{W}_{T+1} = \operatorname*{argmin}_{\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_m} \| - \boldsymbol{C} - \boldsymbol{W}\|_F^2.$$

Let $-\boldsymbol{C}$ have eigendecomposition $-\boldsymbol{C} = \boldsymbol{U} diag(\boldsymbol{\lambda}) \boldsymbol{U}^\intercal$, where $\boldsymbol{\lambda}$ is the vector of the eigenvalues of $-\boldsymbol{C}$. **(author?)** [ACS13, Lemma 3.2] shows that the projection of $-\boldsymbol{C}$ is solved by projecting the eigenvalues $\boldsymbol{\lambda}$ into $\boldsymbol{\mathcal{S}}_m$ while keeping its eigensystem unchanged:

$$\boldsymbol{W}_{T+1} = \boldsymbol{U} diag(\boldsymbol{\lambda}') \boldsymbol{U}^\intercal \qquad \text{and} \qquad \boldsymbol{\lambda}' = \operatorname*{argmin}_{\boldsymbol{v} \in \boldsymbol{\mathcal{S}}_m} \|\boldsymbol{\lambda} - \boldsymbol{v}\|_2^2.$$

W.l.o.g., assume the elements of $\boldsymbol{\lambda}$ are in descend order, i.e. $\lambda_1 \geq \lambda_2 \geq \dots \lambda_n$. To prove that the eigenvectors of the m largest eigenvalues are the same in $\boldsymbol{W}_{T+1}$ and $-\boldsymbol{C}$, we only need to show the following: for any integers pair i and j such that $1 \leq i \leq m < j \leq n$, if $\lambda_i > \lambda_j$, then $\lambda'_i > \lambda'_j$. First note that by the KKT analysis of the problem of projecting into $\boldsymbol{\mathcal{S}}_m$ (see (2.32)), it is easy to see that if $\lambda_i > \lambda_j$, then exactly one of the following three cases holds.

$$\lambda'_i > \lambda'_j \qquad \text{or} \qquad \lambda'_i = \lambda'_j = 0 \qquad \text{or} \qquad \lambda'_i = \lambda'_j = 1.$$

Now we show that when i and j further satisfy $i \leq m < j$, the latter two cases can never happen. Suppose $\lambda'_i = \lambda'_j = 1$ for some $i \leq m < j$. In this case for any $i' \leq m$, $\lambda'_{i'} = \lambda'_j = 1$ also holds. Therefore, the sum of all the coordinates of $\boldsymbol{\lambda}'$ will be at least $m+1$ which contradicts $\boldsymbol{\lambda}' \in \boldsymbol{\mathcal{S}}_m$. Now assume $\lambda'_i = \lambda'_j = 0$ for some $i \leq m < j$. In this case for any $m < j'$, $\lambda'_i = \lambda'_{j'} = 0$ also holds. This implies that the sum of all the coordinates of $\boldsymbol{\lambda}'$s will be at most $m-1$ which again contradicts $\boldsymbol{\lambda}' \in \boldsymbol{\mathcal{S}}_m$.

2.F Regret Lower Bounds When the Number of Trials Is Large

This appendix proves lower bounds on the regret of any online algorithm for the m-set problem: Theorem 2.4 and Theorem 2.5 prove lower bounds for unit loss vectors and Theorem 2.6 proves lower bounds for dense loss vectors. In all of these lower bounds, we assume that the number of the trials T is larger than either the number of experts n or some function of n, m and k (see details of the assumptions in individual theorems). The regret lower bounds for small number of the trials are given in the next Appendix 2.G.

All the lower bounds given in this appendix are proved with the probabilistic bounding technique described in Section 2.5, i.e. in each case, we first choose a probability distribution $\mathcal{P}$ and then show a lower bound on the expected regret of any algorithm when the loss vectors are generated i.i.d. from $\mathcal{P}$. Our lower bounds on the expected regret make use of the following lemma which gives an upper bound on the expected loss of the best comparator in a two expert game.

Lemma 2.7. *Consider a two expert game in which the random loss pairs of both experts are i.i.d. between trials, and at each trial the random pair follows the distribution:*

value of the loss pair	$(0,1)$	$(1,0)$	$(1,1)$	$(0,0)$
probability	p	p	q	$1-2p-q$

(2.35)

where non-negative parameters p and q satisfy $2p+q \leq 1$. Let M be the minimum total loss of the two experts in T such trials. If T and p satisfy $Tp \geq 1/2$, then

$$\mathrm{E}\,[\,M\,] \leq \; T(p+q) - c\sqrt{Tp}$$

for some constant $c > 0$ independent of T, p and q.

Later we will use the case $q = 0$ of the two expert distribution (2.35) as a submodule for building sparse distributions over n experts and $p = q = 1/4$ for building dense distributions over n experts. To prove Lemma 2.7, we need the following lemma from [Koo11, Theorem 2.5.3]:

Lemma 2.8. *Let a_t and b_t be two binary random variables following the distribution*

value of (a_t, b_t)	$(0,1)$	$(1,0)$
probability	0.5	0.5

.

For T independent such pairs, we have

$$\frac{T}{2} - \sqrt{\frac{T-1}{2\pi}} \quad \leq \quad \mathrm{E}\left[\min\left\{\sum_{t=1}^{T} a_t, \sum_{t=1}^{T} b_t\right\}\right] \quad \leq \quad \frac{T}{2} - \sqrt{\frac{T+1}{2\pi}}.$$

Proof. **of Lemma 2.7** Denote the experts' losses at trials $1 \leq t \leq T$ by $\tilde{a}_t$ and $\tilde{b}_t$. In this notation, the statement of Lemma 2.7 is equivalent to:

$$\mathrm{E}\left[\min\left\{\sum_t \tilde{a}_t, \sum_t \tilde{b}_t\right\}\right] \leq T(p+q) - c\sqrt{Tp}.$$

At each trial, the random variable pair $(\tilde{a}_t, \tilde{b}_t)$ has four possible values: $(1,0)$, $(0,1)$, $(1,1)$ or $(0,0)$. If $\tilde{a}_t \neq \tilde{b}_t$, then this trial is "covered by" Lemma 2.8. If $\tilde{a}_t = \tilde{b}_t$, then this trial affects $\sum_t \tilde{a}_t$ and $\sum_t \tilde{b}_t$ the same way and therefore can be excluded from the minimization. We formalize this observation as follows:

$$\begin{aligned}
\mathrm{E}\left[\min\left\{\sum_t \tilde{a}_t, \sum_t \tilde{b}_t\right\}\right] \quad &= \quad \mathrm{E}\left[\min\left\{\sum_{t:\tilde{a}_t\neq\tilde{b}_t} \tilde{a}_t, \sum_{t:\tilde{a}_t\neq\tilde{b}_t} \tilde{b}_t\right\}\right] + \mathrm{E}\left[\sum_{t:\tilde{a}=\tilde{b}} \tilde{a}_t\right] \\
&\overset{\text{Lemma 2.8}}{\leq} \quad \mathrm{E}\left[\frac{R}{2} - \sqrt{\frac{R-1}{2\pi}}\right] + Tq
\end{aligned}$$

where R is a binomial random variable with T draws and success probability $2p$. Clearly $\mathrm{E}[R] = 2Tp$ and therefore $\mathrm{E}[\frac{R}{2}] = Tp$. Moreover under the assumption that $Tp \geq 1/2$, we will show in Lemma 2.12 of Appendix 2.I (using an application of the Chernoff bound) that $\mathrm{E}\left[\sqrt{\frac{R-1}{2\pi}}\right] \geq c\sqrt{Tp}$ for some constant c that does not depend on T, p and q. $\square$

We now use Lemma 2.35 to prove the following theorem which addresses the m-set problem with unit loss vectors for the case $k \leq \frac{n}{2}$.

Theorem 2.4 *Consider the m-set problem with unit loss vectors, where $m = n - k$. Then for $k \leq \frac{n}{2}$ and $T \geq k$, any online algorithm suffers worst case regret at least $\Omega(\sqrt{Tk})$.*

Proof. In this proof, each loss vector is uniformly sampled from the first $2k$ unit vectors, i.e. at each trial, one of the first $2k$ experts is uniformly chosen to incur a unit of loss. To show an upper bound on the loss of the comparator, we group these $2k$ experts into k pairs and note that the loss of each expert pair follows the joint distribution described Lemma 2.7 with $p = \frac{1}{2k}$ and $q = 0$. Furthermore, the condition $Tp \geq 1/2$ of Lemma 2.7 is also satisfied because of the assumption $T \geq k$. Hence, by applying Lemma 2.7 we know that the expected loss of the winner in each pair is at most $T/2k - c\sqrt{T/2k}$, and the total expected loss for k winners from all k pairs is upper bounded by $T/2 - c\sqrt{kT/2}$. Now recalling that the comparator consists of the $m = n - k$ best experts, its total expected loss is upper bounded by the expected loss of the k winners, which is at most $T/2 - c\sqrt{kT/2}$, plus the expected loss of the remaining $n - 2k$ experts, which is zero. Therefore, we have an upper bound of $T/2 - c\sqrt{kT/2}$ on the expected loss of the comparator. On the other hand, since losses are generated independently between trials, any online algorithm suffers loss at least $T/2$. The difference between the lower bound on the expected loss of the algorithm and the upper bound on the expected loss of the best m-set gives the regret lower bound of the theorem. $\square$

The case $k \geq \frac{n}{2}$ is more complicated. Recall that $k = n - 1$ reproduces the vanilla single expert case. Therefore additional $\log n$ factor must appear in the square root of the lower bound. We need the following lemma, which is a generalization of Lemma 2.7 to n experts. In the proof, we upper bound the minimum loss of the experts by the loss of the winner of a tournament among the n experts. The tournament winner does not necessarily have the lowest loss. However as we shall see, its expected loss is close enough to the expected loss of the best expert so that this bounding technique is still useful for obtaining lower bounds on the regret.

Lemma 2.9. *Choose any n, S and T, such that $n = 2^S$ and S divides T. If the loss sequence of length T is generated from a distribution $\mathcal{P}$, such that:*

- *at each trial t, the distribution of losses on n experts is exchangeable, i.e. for any permutation π on a set $\{1, \ldots, n\}$, and for any t, $\boldsymbol{\ell}_t = (\ell_{t,1}, \ell_{t,2}, \ldots, \ell_{t,n})$ and $\boldsymbol{\ell}_t^{\pi} = (\ell_{t,\pi(1)}, \ell_{t,\pi(2)}, \ldots, \ell_{t,\pi(n)})$ have the same distribution,*
- *and the distribution of losses is i.i.d. between trials,*

then,

$$\begin{aligned}&\mathrm{E}\,[\,\textit{minimum loss among } n \textit{ experts in } T \textit{ trials}\,]\\ &\qquad\qquad \leq S\,\mathrm{E}\,[\,\textit{minimum loss among experts 1 and 2 in } T/S \textit{ trials}\,]\,.\end{aligned}$$

Proof. The key idea is to upper bound the minimum loss of any expert by the loss of the expert that wins an S round tournament. In the first round, we start with n experts and pair each expert with a random partner. The round lasts for T/S trials. For each pair, the expert with the smaller loss wins in this round (tie always broken randomly). The $n/2$ winners continue to the second round. At round s, the remaining $n/2^{s-1}$ experts

	first round			second round		
	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
expert 1	1	0	1	0	0	0
expert 4	**0**	**1**	**0**	1	1	1
expert 2	**0**	**0**	**1**	**1**	**1**	**0**
expert 3	1	1	0	1	0	1

Table 2.3: A tournament with $T = 6$ trials, $S = 2$ rounds, and $n = 4$ experts. The bits in the table are the binary losses of the experts in each trial. The brackets show the pairing in each round. The losses of the winners are in bold.

are randomly paired and the winners are determined based on the losses in another set of T/S trials. After S rounds and $T = ST/S$ trials we are left with 1 overall winner.

For example for $S = 2$ rounds, $n = 2^2 = 4$ experts and $T = 6$ trials, consider the sequence of losses shown in Table 2.3. Each of the two tournament consists of $6/2 = 3$ trials. In the first round, expert 1 is paired with expert 4 and expert 2 with expert 3. In the first round, the cumulative losses of experts $1, 2, 3, 4$ are $2, 1, 2, 1$, respective. So expert 4 is the winner of the first pair and expert 2 is the winner of the second pair. In the second round, the two winners (experts 2 and 4) are paired, and they incur cumulative loss 2 and 3, respectively. Hence, expert 2 wins the tournament. The total loss of the tournament winner in all 6 trials is 3. Note that this is larger than the minimum total loss of the 4 experts since expert 1 incurred total loss 2. Nevertheless we shall see that for our probabilistic lower bound proof, the total loss of the tournament winner is close enough to the total loss of the best expert.

To complete the proof it suffices to show that

$$\begin{aligned}&\mathrm{E}\,[\text{ total loss of tournament winner in } T \text{ trials }]\\ &\qquad = S\,\mathrm{E}\,[\text{ minimum loss among experts 1 and 2 in } T/S \text{ trials }]\,.\end{aligned}$$

Due to linearity of expectation:

$$\begin{aligned}&\mathrm{E}\,[\text{ total loss of tournament winner in } T \text{ trials }]\\ &\qquad = \sum_{i=1}^{S} \mathrm{E}\,[\text{ total loss of tournament winner in } i\text{-th round }]\,.\end{aligned}$$

The exchangeability of the losses and the symmetry of the tournament guarantees that each expert is equally likely to be the overall winner. Therefore w.l.o.g., expert 1 is the overall winner. Consider i-th round of the tournament ($1 \le i \le S$), and let (w.l.o.g.) expert 2 be the partner of expert 1 in this round. We have:

$$\begin{aligned}&\mathrm{E}\,[\text{ total loss of tournament winner in } i\text{-th round }]\\ &\quad = \mathrm{E}\left[\text{ total loss of exp. 1 in } i\text{-th round} \;\middle|\; \begin{array}{l}\text{exp. 1 is the tournament winner,}\\ \text{exp. 2 won all past competitions}\\ \text{at rounds } 1,\ldots,i-1.\end{array}\right]\\ &\quad = \mathrm{E}\,[\text{ total loss of exp. 1 in } i\text{-th round} \mid \text{exp. 1 wins over exp. 2 in } i\text{-th round }]\,.\end{aligned}$$

The second equality is due to the fact that the distribution of losses is i.i.d. between trials, and therefore the future and past rounds are independent of the current round. Since the last expression is the same for each of the S rounds we have:

$$\begin{aligned}&\mathrm{E}\,[\text{ total loss of tournament winner in } T \text{ trials }]\\ &\qquad = S\,\mathrm{E}\,[\text{ expected loss of expert 1 in } T/S \text{ trials} \mid \text{expert 1 wins over expert 2 }]\,.\end{aligned}$$

Remains to be shown that the latter expectation is simple the expectation of the minimum of the two experts losses in a single round. Let L_1 and L_2 be the total losses of both experts in the T/S trials and let "$L_1 > L_2$" denote the event that 1 wins over 2 (ties broken uniformly, so that, e.g., $\Pr(L_1 > L_2) + \Pr(L_2 > L_1) = 1$). Then

$$\begin{aligned}
\mathrm{E}\,[\,L_1|L_2 > L_1\,] &= \Big(\Pr(L_2 > L_1) + \Pr(L_1 > L_2)\Big)\,\mathrm{E}\,[\,L_1|L_2 > L_1\,]\,, \\
&= \Pr(L_2 > L_1)\,\mathrm{E}\,[\,L_1|L_2 > L_1\,] \;+\; \Pr(L_1 > L_2)\,\mathrm{E}\,[\,\mathbf{L_1}|\mathbf{L_2} > \mathbf{L_1}\,] \\
\textbf{exchangeability} \quad &= \Pr(L_2 > L_1)\,\mathrm{E}\,[\,L_1|L_2 > L_1\,] \;+\; \Pr(L_1 > L_2)\,\mathrm{E}\,[\,\mathbf{L_2}|\mathbf{L_1} > \mathbf{L_2}\,] \\
&= \Pr(L_2 > L_1)\,\mathrm{E}\,[\,\min\{L_1, L_2\}|L_2 > L_1\,] \\
&\quad + \;\Pr(L_1 > L_2)\,\mathrm{E}\,[\,\min\{L_1, L_2\}|L_1 > L_2\,] \\
&= \mathrm{E}\,[\,\min\{L_1, L_2\}\,]\,.
\end{aligned}$$

□

Now, we use this lemma to prove a lower bound for the m-set problem with $k \geq \frac{n}{2}$:

Theorem 2.5 *Consider the m-set problem with unit loss vectors, where $m = n - k$. Then for $k \geq \frac{n}{2}$ and $T \geq n \log_2(n/m)$, any online algorithm suffers worst case regret at least $\Omega(m\sqrt{T \ln(n/m)/n})$.*

Proof. Let us first assume that $n = 2^j m$ for some integer $j > 0$, i.e. $\log_2(n/m)$ is a positive integer, and that $\frac{T}{\log_2(n/m)}$ is an integer value as well.

At each trial, a randomly chosen expert out of n experts incurs a unit of loss. To show an upper bound on the loss of the comparator, we partition the n experts into

m groups (n divides m from the assumption), and notice that the losses of the n/m experts in each group are exchangeable. Applying Lemma 2.9 to each group of n/m experts with $S = \log_2(n/m)$ rounds and T/S trials per round, we obtain:

$$\begin{aligned}&\mathrm{E}\left[\text{ Loss of the winner in a given group in } T \text{ trials }\right]\\ &\quad\le \log_2\left(\frac{n}{m}\right)\ \mathrm{E}\left[\text{Loss of the winner of two experts in } \frac{T}{\log_2(n/m)} \text{ trials}\right]. \end{aligned} \tag{2.36}$$

The expectation on the RHS is the two expert game considered in Lemma 2.7 with parameters $p = 1/n$ and $q = 0$. Note that $q = 0$ because only one expert suffers loss in each trial. Applying this lemma bounds the expectation on the RHS as

$$\frac{T}{\log_2(n/m)n} - c\sqrt{\frac{T}{\log_2(n/m)n}},$$

Plugging this into (2.36) gives an $T/n - c\sqrt{T\log_2(n/m)/n}$ upper bound on the expected loss of the winner in a given group. We upper bound the expected loss of the comparator by the total loss of m winners from the m groups, which in expectation is at most $Tm/n - cm\sqrt{T\log_2(n/m)/n}$.

Finally the loss of the algorithm is lower bounded as follows: Every expert incurs loss $1/n$ in expectation at each trail and losses are i.i.d. between trials. Therefore any online algorithm suffers loss at least mT/n. and the expected regret is lower bounded by $cm\sqrt{T\log_2(n/m)/n}$. This concludes the proof when $n = 2^j m$ and $\log_2(n/m)$ divides T.

If n is not of this form, we take the largest $n_0 < n$, such that $n_0 = 2^j m$ for some integer j, i.e. $n_0 = \max_{j\in\mathrm{N}}\{2^j m \colon 2^j m \le n\}$. We then apply the reasoning above to n_0 experts, while the remaining $n - n_0$ will incurs loss 1 all the time, which can only increase the loss of the algorithm, but this will not affect the loss of the comparator (comparator will never pick these experts). Since $n_0 \geq n/2$ (otherwise n_0 would not be the largest integer of the form $2^j n$, smaller than n), this does not change the rate under $\Omega(\cdot)$ of the lower bound of the theorem. Finally, if $\frac{T}{\log_2(n/m)}$ is not an integer value, we can choose the largest $T_0 < T$, such $\frac{T_0}{\log_2(n/m)}$ is integer, and use the proof with T_0 rounds, while in the remaining $T - T_0$ rounds all losses are zero. Since $T_0 \geq T/2$, this, again, does not change the rate under $\Omega(\cdot)$. □

Finally, we consider the m-set problems with dense loss vectors. The following theorem proves lower bounds for such problems when $k \leq \frac{n}{2}$ and when $k \geq \frac{n}{2}$.

Theorem 2.6 *Consider the m-set problem with dense loss vectors , where $m = n - k$. Then for $T \geq \log_2 \frac{n}{\min\{k,m\}}$, any online algorithm suffers worst case regret of at least*

$$\Omega(k\sqrt{T \ln \frac{n}{k}}) \text{ when } k \leq \frac{n}{2} \quad \text{or} \quad \Omega(m\sqrt{T \ln \frac{n}{m}}) \text{ when } k \geq \frac{n}{2}.$$

Proof. The proof is similar to the proof of Theorem 2.5, except that at each trial, the losses of all n experts are i.i.d. Bernoulli random variable with probability $p = 1/2$. For such a distribution over losses, any algorithm suffers expected cumulative loss at least $m\,T/2$ for the m-set problem.

For the sake of simplicity, we make some assumptions about n, k and T that avoid rounding issues. When $k \leq n/2$, we assume that $n = 2^j k$ for some integer $j \geq 1$ and that $\frac{T}{\log_2(n/k)}$ is an integer. When $k \geq n/2$, i.e. $m = n - k \leq n/2$, we assume that $n = 2^j m$ for some integer $j \geq 1$ and that $\frac{T}{\log_2(n/m)}$ is an integer. As in the proof of Theorem 2.5, it is easy to generalize the theorem to arbitrary n, k and T satisfying $T \geq \log_2 \frac{n}{\min\{k,m\}}$.

Now, we prove our regret lower bounds for each of the two cases: When $m \leq n/2$, we group the experts into m groups of size n/m and upper bound the loss of the comparator using the m group winners. As before, the loss of each winner can be upper bounded by the lemmas 2.7 (with $p = q = 1$) and 2.9:

$$\begin{aligned}
&\mathrm{E}\,[\text{ Loss of the winner in a given group in } T \text{ trials }]\\
\overset{\text{Lemma 2.9}}{\leq}\quad & \log_2 \frac{n}{m}\ \mathrm{E}\left[\text{Loss of the winner of two experts in } \frac{T}{\log_2(n/m)} \text{ trials}\right]\\
\overset{\text{Lemma 2.7}}{\leq}\quad & \frac{T}{2} - c\sqrt{\frac{T}{4}\log_2\frac{n}{m}}.
\end{aligned}$$

Note that since the experts here incur i.i.d. *Bernoulli*$(\frac{1}{2})$ losses, the above application of Lemma 2.7 requires $p = q = 1/4$. Next, summing up m winners, we have the expected loss of the comparator upper bounded by $Tm/2 - cm\sqrt{T\log_2(n/m)/4}$. Taking the difference between this upper bound and the $Tm/2$ lower bound on loss of any algorithm results in the claimed $\Omega(m\sqrt{T\ln(n/m)})$ lower bound on the regret.

When $k \leq n/2$, we group the experts into k groups and consider a *loser* out of each group which is the expert which incurs the *largest* loss in each group. One can flip around the content of Lemma 2.7 and 2.9 to show that the loser in a group of n/k experts incurs loss in expectation at least $T/2 + c\sqrt{T\log_2(n/k)/4}$. Therefore, the expected loss of all k losers is lower bounded by $Tk/2 + ck\sqrt{T\log_2(n/k)/4}$. Now note that the expected loss of the comparator is upper bounded by the expected total loss of all the experts, which is $Tn/2$, minus the expected loss of the k losers, and hence upper bounded by

$$\frac{Tn}{2} - \left(\frac{Tk}{2} + ck\sqrt{\frac{T}{4}\log_2\frac{n}{k}}\right) = \frac{Tm}{2} - ck\sqrt{\frac{T}{4}\log_2\frac{n}{k}}.$$

Finally, the claimed regret bounds follows from taking the difference between this upper bound and the $Tm/2$ lower bound on the loss of any online algorithm. $\square$

2.G Regret Lower Bounds When the Number of Trials Is Small

This appendix gives general regret lower bounds for the m-set problem when the number of trials T is small: Theorem 2.8 and Theorem 2.9 show lower bounds when the loss vectors are unit vectors; Theorem 2.10 and Theorem 2.11 show lower bounds when the loss vectors are dense vectors. Unlike the lower bounds for large T that are proved with probabilistic arguments (see previous Appendix 2.F) all of the lower bounds in this appendix are proved by showing explicit adversary strategies that force large regret to any online algorithm. The matching upper bounds for small T are trivial and can be found in Section 2.5.

Theorem 2.8. *Consider the m-set problem with unit loss vectors, where $m = n - k$. Then for $k \leq \frac{n}{2}$ and $T \leq k$, any online algorithm suffers worst case regret at least $\Omega(T)$.*

Proof. Consider an adversary that at each trial gives a unit of loss to the expert with the largest weight assigned by the algorithm. Recall that $m = n - k$ and $k \leq \frac{n}{2}$. Therefore all the weights assigned by the algorithm sum to $m \geq \frac{n}{2}$ and the largest weight out of

n experts is at least $\frac{1}{2}$. Hence, after T trials, any algorithm suffers total loss at least $\frac{T}{2}$. On the other hand, since there are at least $n - T \geq m$ (becaue $T \leq k$) experts that are loss free, the loss of the best m-set of experts is zero. Therefore, the regret of any algorithm is at least $\frac{T}{2}$. $\square$

Now we consider the case when $k \geq \frac{n}{2}$. We start with a lemma which is parameterized by an integer $1 \leq i \leq k$ instead of the number of the trials T .

Lemma 2.10. *Consider the m-set problem with unit loss vectors, where $m = n-k$. For any integer $1 \leq i \leq k$, an adversary can force any algorithm to suffer loss $\Omega(m \log_2 \frac{n}{n-i})$ in $O(n \log_2 \frac{n}{n-i})$ trials, and at the same time, keep a set of m experts with loss zero.*

Proof. The adversary's strategy has i rounds, where the j-th round ($1 \leq j \leq i$) has at most $\left\lceil \frac{n}{n-j+1} \right\rceil$ trials and after it finishes, there will be at least $n - j$ experts that still have loss zero. The first round has only one trial, in which a unit of loss is given to the expert with the largest weight. Since all the weights assigned by the algorithm sum to m, the algorithm suffers loss at least $\frac{m}{n}$ in the first round.

Each of the following rounds may contain multiple trials and at the end of round $j-1$ ($2 \leq j \leq i$), there are still at least $n-j+1$ loss free experts. In round j, the adversary uses a strategy with two sub-cases as follows: The adversary first considers the experts that are still loss free. If any of the first $n-j+1$ of them has weight at least $\frac{m}{2(n-j+1)}$, then we are in case 1, where a unit of loss is given to this expert. Otherwise, we are in case 2, in which the adversary considers the remaining $j-1$ experts (which may or may not be loss free) and gives a unit of loss to the one with the largest weight among them. The j-th round ends when the algorithm has suffered total loss at least $\frac{m}{2(n-j+1)}$ in that round. Note that whenever case 1 is reached, a round ends immediately. Our strategy guarantees that after round j, there are at least $n-j$ experts that are loss free. Next we upper bound the number of case 2 trials in a round by showing a lower bound on the loss of the algorithm in case 2 trials. Recall that in case 2, $n-j+1$ experts have weight no more than $\frac{m}{2(n-j+1)}$ each, and the expert that has the largest weight in the remaining $j-1$ experts incurs a unit of loss. Using these facts as well as the fact that all the weights sum to m, we can lower bound the weight of the expert that incurs loss (which is also the loss of the algorithm) as follows:

$$\frac{\left(m - \frac{m}{2(n-j+1)}(n-j+1)\right)}{j-1} \geq \frac{m}{2(j-1)} \geq \frac{m}{2n}.$$

Recalling that the j-th round ends when the algorithm suffers total loss $\frac{m}{2(n-j+1)}$ in that round, we conclude that the j-th round can have at most $\left\lceil \frac{n}{n-j+1} \right\rceil$ trials.

Summing up over i rounds, the algorithm suffers total loss at least $\sum_{j=1}^{i} \frac{m}{2(n-j+1)} = \Omega(m \log \frac{n}{n+i})$ in at most $\sum_{j=1}^{i} \left\lceil \frac{n}{2(n-j-1)} \right\rceil = O(n \log \frac{n}{n-i})$ trials. On the other hand, the loss of the best m-set of experts is zero due to assumption $i \leq k$ and the fact that after $j = i$ rounds, there are at least $n-i$ loss free experts. Hence the lemma follows. □

Theorem 2.9. *Consider the m-set problem with unit loss vectors, where $m = n-k$. Then for $k \geq \frac{n}{2}$ and $T \leq n \log_2 \frac{n}{m}$, any algorithm suffers worst case regret at least $\Omega(\frac{m}{n}T)$.*

Proof. Lemma 2.10 states that there exist two positive constants c_1 and c_2, such that for any integer $1 \leq i \leq k$, the adversary can force any algorithm to suffer regret at least

$c_1 m \log_2 \frac{n}{n-i}$ in at most $c_2 n \log_2 \frac{n}{n-i}$ trials. The proof splits into two cases, depending on the number of the trials T:

- When $T < c_2 n \log_2 \frac{n}{n-1}$, T is upper bounded by a constant as follows:

$$T < c_2 n \log_2 \frac{n}{n-1} = \frac{c_2 n}{\log 2} \log \left(1 + \frac{1}{n-1}\right) \leq \frac{c_2 n}{(n-1)\log 2} \overset{n \geq 2}{\leq} \frac{2c_2}{\log 2}.$$

  Since the adversary can always force any algorithm to suffer constant regret, the theorem holds trivially.

- When $T \geq c_2 n \log_2 \frac{n}{n-1}$, we set $i = \min\{\lfloor i' \rfloor, k\}$, where $i' = n(1 - 2^{-T/c_2 n})$ is the solution of $c_2 n \log_2 \frac{n}{n-i'} = T$. We note that the function $c_2 n \log_2 \frac{n}{n-i}$ is monotonically increasing in i, which results in two facts: first, $i \geq 1$, since we assumed $T \geq c_2 n \log_2 \frac{n}{n-1}$; second, $c_2 n \log_2 \frac{n}{n-i} \leq T$, since $i \leq \lfloor i' \rfloor$. We further show that $c_2 n \log_2 \frac{n}{n-i} \geq \min\{c_2, \frac{1}{3}\}T$ as follows:

  - When $i = \lfloor i' \rfloor$, first note that $\left(\frac{n}{n-i}\right)^3 \geq \frac{n}{n-i'}$, since:

$$(n-i')n^2 - (n-i)^3 \geq (n-i-1)n^2 - (n-i)^3 = 2n^2 i + 3ni^2 - i^3 - n^2 \overset{1 \leq i < n}{\geq} 0.$$

    Plugging $c_2 n \log_2 \frac{n}{n-i'} = T$, we have $c_2 n \log_2 \frac{n}{n-i} \geq \frac{1}{3}T$.

  - When $i = k$, $c_2 n \log_2 \frac{n}{n-i} = c_2 n \log_2 \frac{n}{m} \geq c_2 T$, since $T \leq n \log_2 \frac{n}{m}$ is assumed in the theorem.

  Now, using Lemma 2.10 with i set as $i = \min\{\lfloor i' \rfloor, k\}$, results in an adversary that forces the algorithm to suffer regret at least $c_1 m \log \frac{n}{n-i} \geq \frac{mc_1}{nc_2} \min\{c_2, \frac{1}{3}\}T = \Omega\left(\frac{m}{n}T\right)$ in at most T trials. When the adversary uses less than T trials, then the game can be extended to last exactly T trials by playing zero loss vectors for the remaining trials.

$\square$

Theorem 2.10. *Consider the m-set problem with dense loss vectors, where $m = n - k$. Then for $k \geq \frac{n}{2}$ and $T \leq \log_2 \frac{n}{m}$, the worst case regret of any algorithm is at least $\Omega(Tm)$.*

Proof. The proof uses an adversary which forces any algorithm to suffer loss $\Omega(Tm)$, and still keeps the best m-set of experts to be loss free. Note that at each trial, the adversary decides on the loss vector after the algorithm makes its prediction $\boldsymbol{w}_t$, where $\boldsymbol{w}_t \in [0,1]^n$ with $\sum_i w_{t,i} = m$.

At trial one, the adversary first sorts the n experts by their weights assigned by the algorithm, and then gives a unit of loss to each of the experts in the first half, i.e. the experts with larger weights. Since the weights sum to m, the total weight assigned to the experts in the first half is at least $\frac{m}{2}$. Hence in the first trial, the algorithm suffers loss at least $\frac{m}{2}$.

At each of the following trials, the adversary only sorts those experts that have not incur any loss so far and gives unit losses to the first half (the half with larger weights) of these experts, as well as all the experts that have already incurred losses before this trial. It is easy to see that in this way the algorithm suffers loss at least $\frac{m}{2}$ at each trial.

Since the number of the experts that are loss free halves at each trial, after

$T \le \log_2 \frac{n}{m}$ trials, there will still be at least m loss free experts. Now since the algorithm suffers loss at least $\frac{mT}{2}$ in T trials, the theorem follows. $\square$

Theorem 2.11. *Consider the m-set problem with dense loss vectors, where $m = n - k$. Then for $k \le \frac{n}{2}$ and $T \le \log_2 \frac{n}{k}$, any algorithm suffers worst case regret at least $\Omega(Tk)$.*

Proof. The proof becomes conceptually simpler if we use the notion of *gain* defined as the follows: if $\boldsymbol{w}_t$ is the parameter of the algorithm, we define its complement $\bar{\boldsymbol{w}}_t$ as $\bar{w}_{t,i} = 1 - w_{t,i}$. The gain of the algorithm at trial t is the inner product between the "gain" vector $\boldsymbol{\ell}_t$ and the complement $\bar{\boldsymbol{w}}_t$, i.e. $\bar{\boldsymbol{w}}_t \cdot \boldsymbol{\ell}_t$. Similarly, for any comparator $\boldsymbol{w} \in \boldsymbol{\mathcal{S}}_m$, we define its gain as $\bar{\boldsymbol{w}} \cdot \boldsymbol{\ell}_t = \sum_{i=1}^n (1 - w_i) l_{t,i}$. It is easy to verify that the regret of the algorithm can be written as the difference between the largest gain of any subset of k experts and the gain of the algorithm:

$$\text{REG} = \max_{\bar{\boldsymbol{w}} \in \boldsymbol{\mathcal{S}}_k} \sum_{t=1}^T \bar{\boldsymbol{w}} \cdot \boldsymbol{\ell}_t - \sum_{t=1}^T \bar{\boldsymbol{w}}_t \cdot \boldsymbol{\ell}_t,$$

where $\boldsymbol{\mathcal{S}}_k = \{\boldsymbol{w} \in [0,1]^n \colon \sum_i w_i = k\}$. At trial one, the adversary first sorts the n experts by their complementary weights and then gives a unit of gain to each of the experts in the second half, i.e. the experts with smaller complementary weights. Since the complementary weights sum to k, the gain of the algorithm is at most $\frac{k}{2}$ in the first trial.

At each of the following trials, the adversary only sorts the experts that received gains in all of the previous trials by their complementary weights. It then gives unit gains to the second half (the half with smaller complementary weights) of these experts. It is easy to see that in this way the gain of the algorithm is at most $\frac{k}{2}$ at each trial.

Note that half of the experts that always receive gain prior to a trial t will receive gain again in trial t. Hence, after $T \leq \log_2 \frac{n}{k}$ trials, there will be at least k experts that received gains in all of the T trials, which means that the total gain of the best k experts is Tk. Now, since the algorithm receives total gain at most $\frac{kT}{2}$ in T trials, the theorem follows. $\square$

2.H Proof of Theorem 2.7

The following theorem gives a regret lower bound that is expressed as a function of the loss budget B_L. This lower bound holds for any online algorithm that solves the m-set problem with either unit or dense loss vectors. The proof is based on the time dependent regret lower bounds proven in the previous appendices.

Theorem 2.7 *For the m set problem with either unit or dense loss vectors, any online algorithm suffers worst case regret of at least* $\Omega(\max\{\sqrt{B_L m \ln(n/m)}, m\ln(n/m)\})$.

Proof. Since the unit loss vectors are a subset of the dense loss vectors, we only need to prove the theorem with the unit loss vectors. First consider the case that $B_L \leq m \log_2 \frac{n}{m}$ and the lower bound we need to prove becomes $\Omega(m\ln(n/m))$. In this case, the lower bound follows directly from Lemma 2.10 by setting the variable i of the lemma to k.

Next, consider the case when $B_L \geq m \log_2 \frac{n}{m}$ and the lower bound we need to prove becomes $\Omega(\sqrt{B_L m \ln(n/m)})$. We need to construct a instance sequence of loss budget B_L incurring regret at least $\Omega(\sqrt{B_L m \ln(n/m)})$ to any algorithm. This instance sequence is constructed via Theorem 2.4 and Theorem 2.5: For any algorithm, these theorems provide a sequence of T unit loss vectors that incurs regret at least $\Omega(m\sqrt{\frac{T\ln(n/m)}{n}})$. We apply these theorems with $T = \frac{n}{m}B_L \geq n\log_2\frac{n}{m}$. Since the produced sequence consists of unit loss vectors and has length $\frac{n}{m}B_L$, the total loss of the m best experts is at most B_L. Finally plugging $T = \frac{n}{m}B_L$ into the regret bounds guaranteed by the theorems results in the regret $\Omega(\sqrt{B_L m \ln(n/m)})$. $\square$

2.I Auxiliary Lemmas

Lemma 2.11. *Inequality* $\max\{\min\{a,b\},c\} \geq \min\{\max\{a,c\},b\}$ *holds for any real number* a, b *and* c.

Proof. If $c \geq \max\{a,b\}$, LHS is c and RHS is b. Hence, the inequality holds. If $a \geq c \geq b$ or $b \geq c \geq a$, LHS is c while RHS is at most c. If $c \leq a$ and $c \leq b$, both sides are $\min\{a,b\}$. $\square$

Lemma 2.12. *Let* $X \sim$ *Binomial*(T,p). *If* $Tp \geq 8c$ *for any positive constant* c, *then* $\mathrm{E}[\sqrt{X}] \geq \frac{c}{\sqrt{2}(1+c)}\sqrt{Tp}$.

Proof. We use the following form of the Chernoff bound [DS02]:

$$\Pr(X \leq Tp - \delta) \leq e^{-\frac{\delta^2}{2Tp}}.$$

Setting $\delta = \frac{1}{2}Tp$, we have $\Pr(X \leq \frac{1}{2}Tp) \leq e^{-Tp/8} \leq e^{-c}$. Since for $c > 0$, $\log(c) \leq c-1$, this implies $e^{-c} \leq \frac{1}{1+c}$, so that we further have $\Pr(X \leq \frac{1}{2}Tp) \leq \frac{1}{1+c} = 1 - \frac{c}{1+c}$. Now we calculate $\mathrm{E}[\sqrt{X}]$ from its definition,

$$\begin{aligned}
\mathrm{E}[\sqrt{X}] = \sum_{x=0}^{T} \Pr(X = x)\sqrt{x} &\geq \sum_{x=\left\lfloor \frac{Tp}{2} \right\rfloor + 1}^{T} \Pr(X = x)\sqrt{x} \\
&\geq \sum_{x=\left\lfloor \frac{Tp}{2} \right\rfloor + 1}^{T} \Pr(X = x)\sqrt{\left\lfloor \frac{Tp}{2} \right\rfloor + 1} \\
&= \Pr(X > \tfrac{1}{2}Tp)\sqrt{\left\lfloor \frac{Tp}{2} \right\rfloor + 1} \\
&\geq \frac{c}{\sqrt{2}(1+c)}\sqrt{TP}.
\end{aligned}$$

$\square$

Chapter 3

Learning a Set of Directions

3.1 Introduction

In this chapter we consider learning directions. Let us fix the dimensionality n throughout. Then a direction is simply a vector $\boldsymbol{u} \in \mathrm{R}^n$ of unit length. We model the learning problem as a sequential game where each round the learner predicts by playing a direction $\boldsymbol{u}$ and nature responds with an instance direction $\boldsymbol{x}$. We define the resulting *directional gain* as

$$\big(\boldsymbol{u}^\intercal \boldsymbol{x} + c\big)^2 \tag{3.1}$$

where the constant c is a fixed design parameter known to the learner. We choose to study this gain because it is a simple and smooth trade-off (governed by c) between two intuitively reasonable criteria of closeness: the angle cosine and the subspace similarity. To see this, we expand our gain as:

$$\big(\boldsymbol{u}^\intercal \boldsymbol{x} + c\big)^2 \;=\; (\boldsymbol{u}^\intercal \boldsymbol{x})^2 + 2c\,\boldsymbol{u}^\intercal \boldsymbol{x} + c^2. \tag{3.2}$$

- As $c \to \infty$, then our gain becomes the angle cosine $\boldsymbol{u}^\intercal \boldsymbol{x} = \cos(\boldsymbol{u}, \boldsymbol{x})$. There is a simple minimax algorithm for this angle gain by [KW11].

- When $c = 0$, then our gain becomes the subspace similarity $(\boldsymbol{u}^\intercal \boldsymbol{x})^2$. This gain is optimized in rank one (uncentered) PCA. [WK08]. The main disadvantage of the PCA gain $(\boldsymbol{u}^\intercal \boldsymbol{x})^2$ is that it is fundamentally bidirectional, i.e. reversing either $\boldsymbol{u}$ or $\boldsymbol{x}$ does not affect this gain.

- For general c, the directional gain (3.2) is a trade-off between the above two gains. Unfortunately the algorithms for the linear and quadratic gains cannot just be merged somehow. As we shall see the tools needed for the trade-off gain are much more involved. In Figure 3.1, we compare the directional gain for $c = 1$ with the original PCA gain, i.e. the directional gain for $c = 0$. As one can see from the figure, for $c = 1$ the directional gain is highly sensitive[1] to the direction of the prediction $\boldsymbol{u}$ as well as the target instance $\boldsymbol{x}$: it attains maximum value 4 when $\boldsymbol{x}$ is the same direction as $\boldsymbol{u}$ (i.e. $\boldsymbol{x} = \boldsymbol{u}$) and minimum value 0 at the opposite $\boldsymbol{x} = -\boldsymbol{u}$. Figure 3.2 further plots this gain for several different values of c between 0 and 5. Note that since $(\boldsymbol{u}^\intercal \boldsymbol{x} - c)^2 = ((-\boldsymbol{u})^\intercal x + c)^2$, the corresponding plots for negative c values can be obtained by rotating the plots in Figure 3.1 by 180 degrees. A discussion on the range of this gain for general values of c is given in

[1]The bidirectional PCA gain is essentially the average of the directional gain for $\boldsymbol{x}$ and $-\boldsymbol{x}$:

$$\underbrace{(\boldsymbol{u}^\intercal \boldsymbol{x})^2}_{\text{PCA gain}} = \frac{1}{2}\Big(\underbrace{(\boldsymbol{u}^\intercal \boldsymbol{x} + c)^2}_{\text{directional gain of } \boldsymbol{x}} + \underbrace{(\boldsymbol{u}^\intercal (-\boldsymbol{x}) + c)^2}_{\text{directional gain of } -\boldsymbol{x}} \Big) - c^2.$$

Thus the algorithms of this paper retain PCA as a special case when the instance directions are doubled.

Appendix 3.A.

- Note that the quadratic Taylor approximation of any gain $g(\boldsymbol{u}^\intercal \boldsymbol{x})$ at $\boldsymbol{u} = \boldsymbol{0}$ has the form $g(0) + g'(0)\, \boldsymbol{u}^\intercal \boldsymbol{x} + \frac{1}{2} g''(0)\, (\boldsymbol{u}^\intercal \boldsymbol{x})^2$. Dividing by $\frac{1}{2}g''(0)$ results in our gain (3.2) except for an immaterial constant shift.

So how can we get away with maximizing a quadratic gain? Note that the gain is linear in $\boldsymbol{u}$ and $\boldsymbol{u}\boldsymbol{u}^\intercal$, and therefore the underlying optimization problems become linear semi-definite.

We think of a sequence of instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ as "easy" if there is a single direction $\boldsymbol{u}$ with high cumulative gain. The goal of the learner is to predict well if the data are easy. To this end, we evaluate the performance of the learner after T rounds by measuring its *regret*:

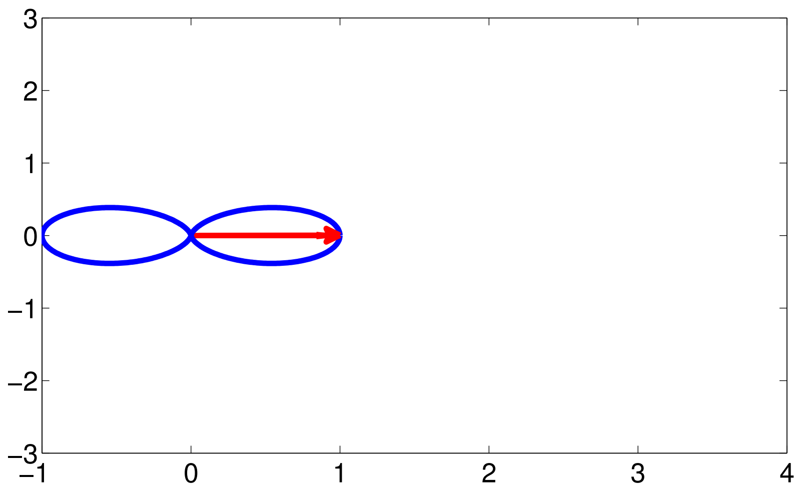
$$\underbrace{\max_{\text{unit } \boldsymbol{u}} \sum_{t=1}^{T} \big(\boldsymbol{u}^\intercal \boldsymbol{x}_t + c\big)^2}_{\text{offline gain}} - \underbrace{\sum_{t=1}^{T} \big(\boldsymbol{u}_t^\intercal \boldsymbol{x}_t + c\big)^2}_{\text{online gain}}.$$

Here $\boldsymbol{u}_t$ denotes the direction of the online algorithm chosen at trial t. To be able to guarantee low regret in an adversarial environment, it is sometimes advantageous to choose the direction $\boldsymbol{u}_t$ probabilistically and define the regret as the offline gain minus the *expected* online gain. A probability distribution $\mathcal{P}$ on predictions $\boldsymbol{u}$ has expected gain given by

$$\mathrm{E}\left[\big(\boldsymbol{x}^\intercal \boldsymbol{u} + c\big)^2\right] = \mathrm{E}\left[\boldsymbol{x}^\intercal \boldsymbol{u}\boldsymbol{u}^\intercal \boldsymbol{x} + 2c\boldsymbol{x}^\intercal \boldsymbol{u} + c^2\right] = \boldsymbol{x}^\intercal\, \mathrm{E}\big[\boldsymbol{u}\boldsymbol{u}^\intercal\big] \boldsymbol{x} + 2c\boldsymbol{x}^\intercal\, \mathrm{E}\left[\boldsymbol{u}\right] + c^2.$$

This shows that most of $\mathcal{P}$ is irrelevant. The expected gain is determined by just the first moment (mean) $\mathrm{E}[\boldsymbol{u}]$ and second moment $\mathrm{E}[\boldsymbol{u}\boldsymbol{u}^\intercal]$. In this chapter we never

(a) PCA gain $(\boldsymbol{u}^\intercal \boldsymbol{x})^2$



(b) Directional gain $\left(\boldsymbol{u}^\intercal \boldsymbol{x} + 1\right)^2$

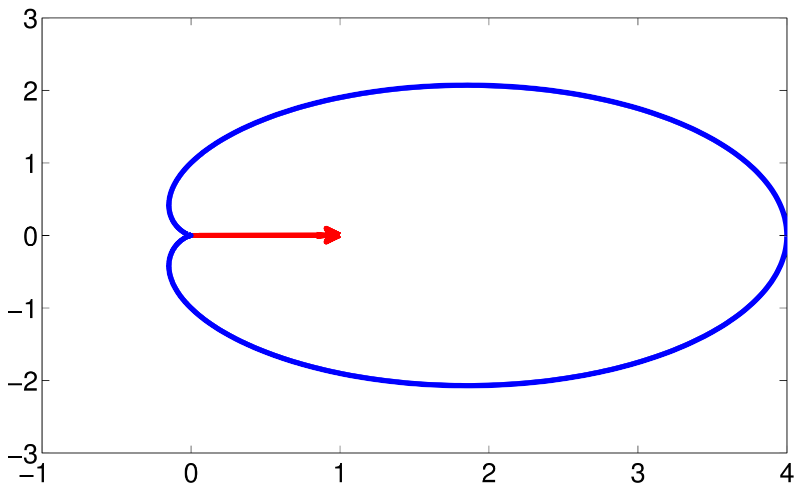
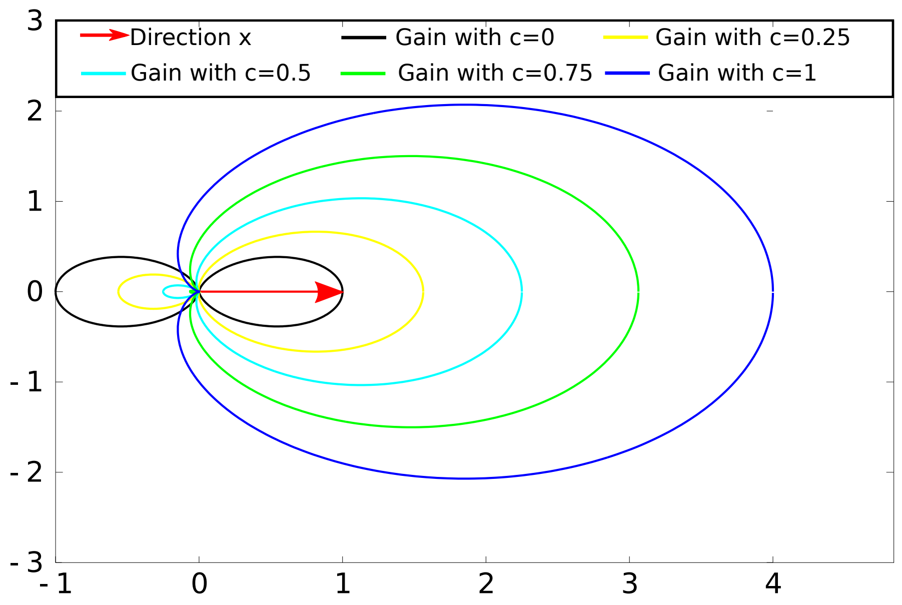


Figure 3.1: Comparison of PCA gain (directional gain for $c = 0$) and directional gain for $c = 1$: The target direction $\boldsymbol{x}$ is depicted by a red arrow. In each case the blue curve is $\boldsymbol{u}$ scaled by the gain of $\boldsymbol{u}$, as the prediction $\boldsymbol{u}$ goes around the unit circle.

(a) Directional gain $\left(\boldsymbol{u}^\intercal\boldsymbol{x}+c\right)^2$ with $c=0,0.25,0.5,0.75,1$



(b) Directional gain $\left(\boldsymbol{u}^\intercal\boldsymbol{x}+c\right)^2$ with $c=1,2,3,4,5$

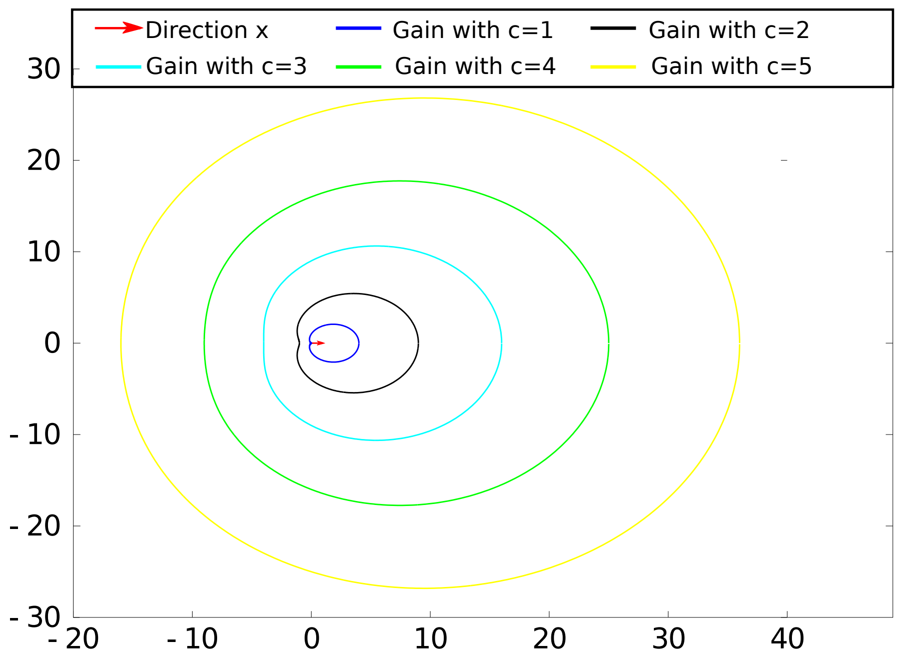


Figure 3.2: Plots of directional gain for different values of constant c. In both figures, the red arrows are the target direction $\boldsymbol{x}$ and the curves are the prediction $\boldsymbol{u}$ scaled by direction gain $(\boldsymbol{u}^\intercal\boldsymbol{x}+c)^2$ as $\boldsymbol{u}$ goes around the unit circle. Each of these curves is plotted for a different value of the constant c (see the individual values in the legends).

work with full distributions, but always with these simple two statistics. That is, the parameter of the algorithm has the form $(\boldsymbol{\mu}, \boldsymbol{D})$, s.t. $(\boldsymbol{\mu}, \boldsymbol{D}) = \mathrm{E}\left[(\boldsymbol{u}, \boldsymbol{u}\boldsymbol{u}^\intercal)\right]$ for some $\mathcal{P}$. It is hence important to characterize which pairs of first and second moments can arise from distributions: We will show that a vector $\boldsymbol{\mu}$ and symmetric matrix $\boldsymbol{D}$ are the first and second moment of some distribution on directions iff $\mathrm{tr}(\boldsymbol{D}) = 1$ and $\boldsymbol{D} \succeq \boldsymbol{\mu}\boldsymbol{\mu}^\intercal$. Note that these conditions imply that $\boldsymbol{D}$ is a density matrix, i.e. a positive semi-definite matrix of unit trace.

Our algorithm has the following outline. At the beginning of each trial we decompose the current parameter $(\boldsymbol{\mu}_t, \boldsymbol{D}_t)$ into a sparse mixture of pure events $(\boldsymbol{u}, \boldsymbol{u}\boldsymbol{u}^\intercal)$ and choose a direction $\boldsymbol{u}_t$ at random from this mixture. We then update the parameter based on the observed instance $\boldsymbol{x}_t$ and project the updated parameter back into the parameter space.

We also consider the direction learning problem where each round the learner plays a *set* of k orthogonal directions $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$. The set size k is a fixed design parameter known to the learner. After nature reveals its instance $\boldsymbol{x}$, the algorithm now achieves the total gain over the set:

$$\sum_{i=1}^{k} \left(\boldsymbol{u}_i^\intercal \boldsymbol{x} + c\right)^2 . \tag{3.3}$$

The online algorithm chooses such a set probabilistically in each trial. If $\mathcal{P}$ is a probability distribution on such sets $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$, then the expectation of the gain (3.3) expands to

$$\mathrm{E}\left[\sum_{i=1}^{k} \left(\boldsymbol{x}^\intercal \boldsymbol{u}_i + c\right)^2\right] = \boldsymbol{x}^\intercal \, \mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\intercal\right] \boldsymbol{x} + 2c\boldsymbol{x}^\intercal \, \mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i\right] + kc^2.$$

We see that the expected gain is again determined by the first moment $\mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i\right]$ and second moment $\mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\top\right]$. We will show that a vector $\boldsymbol{\mu}$ and matrix $\boldsymbol{D}$ are the first two moments of a distribution on sets of k orthogonal directions iff $\mathrm{tr}(\boldsymbol{D}) = k$ and $\boldsymbol{\mu}\boldsymbol{\mu}^\top/k \preceq \boldsymbol{D} \preceq \boldsymbol{I}$. The parameter space of our algorithm hence consists of all $(\boldsymbol{\mu}, \boldsymbol{D})$ with these properties. Again we present an algorithm for decomposing an arbitrary parameter $(\boldsymbol{\mu}, \boldsymbol{D})$ into a sparse mixture of pure events $(\sum_{i=1}^{k} \boldsymbol{u}_i, \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\top)$ with orthonormal $\boldsymbol{u}_i$ and sample from this mixture at the beginning of each trial. We also generalize our projection algorithm to the $k > 1$ case.

The gain (3.1) (and set generalization (3.3)) are quadratic in their natural parameterization by the direction $\boldsymbol{u}$. However by expanding the square, we find that they are *linear* in the two parts $\boldsymbol{u}$ and $\boldsymbol{u}\boldsymbol{u}^\top$. Our setup exploits this linear reformulation of the gain.

We still need to discuss which type of algorithms should be used for updating the parameter matrix after processing the current direction. Recall that there are two algorithms to consider: the Matrix Exponentiated Gradient (MEG) algorithm that is based on regularizing with the Quantum Relative Entropy [TRW05] and the Gradient Descent (GD) algorithm which uses the squared Frobenius norm as a regularizer. For our problem both of these algorithms essentially have the same regret bounds when the bounds are expressed as functions of the number of trials (examples) T, i.e. when they are time dependent bounds. Similar to the loss dependent regret bound we discussed in Chapter 2, we could also develop *gain* dependent regret bound for our problem, i.e. the number of the trials T in the regret can be replaced by the total gain of the best comparator. However, such a bound is not interesting in practice since the total gain of the best comparator is usually as large as the number of examples T. Therefore in this chapter, we only develop for the problem of learning directions a time dependent regret bound. We prove this bound with GD algorithm, which is the simpler of the two aforementioned online algorithms.

Related Work

The outline of our algorithm is similar to Component Hedge [KWK10] which deals with distributions on exponentially many combinatorial concepts by maintaining the expectation of their constituent components. The key two pieces are the convex decomposition and the projection algorithm. This method was lifted to the matrix domain in the work on online PCA [WK08]. However each piece is significantly more complicated in our setting because our gain trades off first and second order parts.

Our work is related to centered PCA [WK08] which also uses a mean and a density matrix as the parameter. However in that case the mean is unconstrained and can be optimized independently from the density, leading to a much simpler problem.

Our gain $(\boldsymbol{u}^\intercal \boldsymbol{x} + c)^2$ is a simple polynomial kernel with the feature map $\phi(\boldsymbol{u})$ being comprised of the n components of $\boldsymbol{u}$, the n^2 components of $\boldsymbol{u}\boldsymbol{u}^\intercal$ and a constant feature. However our methods are decidedly different from kernel methods (including Kernel PCA [KW07]). Our algorithms don't just rely on dot products $\phi(\boldsymbol{x}_t)^\intercal \phi(\boldsymbol{x}_q)$ in feature space (the kernel paradigm). Instead, our parameter is always a convex combination of $\phi(\boldsymbol{u})$ and we project back into this parameter space. This projection step clearly violates the kernel paradigm.

Outline

We warm up by optimizing the gain offline in Section 3.2. We then present the online algorithm in Section 3.3 and analyze its regret. The essential building block in both these sections is the characterization of the parameter space. We prove the difficult direction of the characterization theorem in Section 3.4 by presenting our new decomposition algorithm. We conclude by discussing the big picture in Section 3.5.

3.2 The Offline Problem

Given a sequence of directions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the offline problem is to optimize the total gain:

$$\begin{aligned}
&\max_{\text{orthonormal } \boldsymbol{u}_1 \ldots \boldsymbol{u}_k} \sum_{t=1}^{T} \sum_{i=1}^{k} (\boldsymbol{u}_i^\intercal \boldsymbol{x}_t + c)^2 \\
= &\max_{\text{orthonormal } \boldsymbol{u}_1 \ldots \boldsymbol{u}_k} \operatorname{tr}\Bigg(\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\intercal \underbrace{\sum_{t=1}^{T} \boldsymbol{x}_t \boldsymbol{x}_t^\intercal}_{=:\boldsymbol{R}} \Bigg) + 2c \Bigg(\sum_{i=1}^{k} \boldsymbol{u}_i \Bigg)^\intercal \underbrace{\sum_{t=1}^{T} \boldsymbol{x}_t}_{=:\boldsymbol{r}} + Tc^2.
\end{aligned}$$

We will reformulate the above as a semi-definite optimization problem. Instead of maximizing over a single orthonormal set, we maximize the expected value of the objective over distributions on such sets. This does not change the value of the optimization problem. For any probability distribution on sets of k orthogonal directions, we can characterize the first moment $\mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i\right]$ and second moment $\mathrm{E}\left[\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\intercal\right]$ as follows:

Theorem 3.1. *A vector $\boldsymbol{\mu} \in \mathrm{R}^n$ and symmetric matrix $\boldsymbol{D} \in \mathrm{R}^{n\times n}$ are the first and second moment of a probability distribution on sets of k orthogonal directions if and only if*

$$\operatorname{tr}(\boldsymbol{D}) \;=\; k \qquad \textit{and} \qquad \boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D} \preceq \boldsymbol{I}. \tag{3.4}$$

Proof. For the $\Longrightarrow$ direction, it suffices to show that (3.4) is satisfied for "pure" distributions, i.e. when $\boldsymbol{D} = \sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{u}_i^\intercal$ and $\boldsymbol{\mu} = \sum_{i=1}^k \boldsymbol{u}_i$, for some set of orthogonal directions. The result then extends to all distributions by convexity. Since the condition is invariant under basis transformations, we may as well verify it for the set of standard basis vectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k$. Its first and second moment are

$$\boldsymbol{\mu} \;=\; \begin{bmatrix} \mathbf{1}_k \\ \mathbf{0}_{n-k} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{D} \;=\; \begin{bmatrix} \boldsymbol{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Clearly, $\operatorname{tr}(\boldsymbol{D}) = k$ and $\boldsymbol{D} \preceq \boldsymbol{I}$. To show that $\boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D}$, note that $\boldsymbol{\mu}$ is the only eigenvector of $\boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k$, and its associated eigenvalue is 1. However, $\boldsymbol{\mu}$ is also an eigenvector of $\boldsymbol{D}$, again with eigenvalue 1. The $\Longleftarrow$ direction is much harder. It follows from the decomposition procedure presented in Section 3.4. $\square$

This means that our offline problem becomes the following semi-definite program:

$$\max_{(\boldsymbol{\mu},\boldsymbol{D}) \text{ s.t. } \operatorname{tr}(\boldsymbol{D})=k \text{ and } \boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D} \preceq \boldsymbol{I}} \qquad \operatorname{tr}(\boldsymbol{D}\boldsymbol{R}) + 2c\,\boldsymbol{\mu}^\intercal \boldsymbol{r} + Tc^2.$$

In Appendix 3.B we discuss a special condition on $(\boldsymbol{r}, \boldsymbol{R})$ when the solution of the k directions problem can be constructed from the solution to the k-PCA problem.

Note that the solution $(\boldsymbol{\mu}^*, \boldsymbol{D}^*)$ returned for the above optimization problem might not be a pure set of k directions but the first and second moment of a distribution on sets of k orthogonal directions, all of which have the same gain. In that case we can employ the decomposition algorithm of Section 3.4 which decomposes the moments $(\boldsymbol{\mu}^*, \boldsymbol{D}^*)$ into a mixture of pure solutions. To obtain one set, simply run this greedy algorithm for one step.

3.3 Online Algorithm

The algorithm maintains the two moments $(\boldsymbol{\mu}_t, \boldsymbol{D}_t)$ as its parameter. It follows the protocol:

```
At trial $t = 1 \dots T$,
1. Learner *decomposes* parameter $(\boldsymbol{\mu}_t, \boldsymbol{D}_t)$ into a mixture of $2(n+1)$ sets
   of $k$ orthonormal directions and chooses a set $\boldsymbol{u}_1, \dots, \boldsymbol{u}_k$ at random from it
2. Nature reveals direction $\boldsymbol{x}_t \in \mathrm{R}^n$
3. Learner receives expected gain $\mathrm{E}\left[\sum_{i=1}^k (\boldsymbol{u}_i^\intercal \boldsymbol{x}_t + c)^2\right]$
4. Learner *updates* $(\boldsymbol{\mu}_t, \boldsymbol{D}_t)$ to $(\widehat{\boldsymbol{\mu}}_{t+1}, \widehat{\boldsymbol{D}}_{t+1})$ based on the gradient of the gain on $\boldsymbol{x}_t$
5. Learner produces new parameter $(\boldsymbol{\mu}_{t+1}, \boldsymbol{D}_{t+1})$ by
   *projecting* $(\widehat{\boldsymbol{\mu}}_{t+1}, \widehat{\boldsymbol{D}}_{t+1})$ back into the parameter space.
```

The goal of the learner is to minimize the regret which is the gain of the offline algorithm minus the expected gain of the online algorithm. We first show how to update and project (steps 4 and 5) and defer the decomposition step 1 to the end, since it is the hardest.

3.3.1 The Update and Projection

We update using the Gradient Descent algorithm (see e.g. [Zin03, CBLW96])

$$\widehat{\boldsymbol{\mu}}_{t+1} \;\coloneqq\; \boldsymbol{\mu}_t + 2\eta c\, \boldsymbol{x}_t \qquad \text{and} \qquad \widehat{\boldsymbol{D}}_{t+1} \;\coloneqq\; \boldsymbol{D}_t + \eta\, \boldsymbol{x}_t \boldsymbol{x}_t^\intercal,$$

and project back into the parameter space as follows:

$$(\boldsymbol{\mu}_{t+1}, \boldsymbol{D}_{t+1}) \;\coloneqq\; \underset{(\boldsymbol{\mu},\boldsymbol{D}) \text{ s.t. } \operatorname{tr}(\boldsymbol{D})=k \text{ and } \boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D} \preceq \boldsymbol{I}}{\operatorname{argmin}} \quad \|\boldsymbol{D} - \widehat{\boldsymbol{D}}_{t+1}\|_F^2 + \|\boldsymbol{\mu} - \widehat{\boldsymbol{\mu}}_{t+1}\|^2.$$

Since both the objective and the constraint set are convex, this projection can be efficiently computed using a convex optimization package.[2]

The above GD update and the projection are based on regularizing with the square Frobenius norm. An alternate would be the Matrix Exponentiated Gradient update which uses the Quantum Relative Entropy as a regularizer. Since the MEG update has the same regret bound (not shown) for our specific problem based on unit instance vectors, we chose to only present the simpler GD update.

The following theorem develops a regret bound for the GD algorithm. Note that the drop of the squared Frobenius norm is used as a measure of progress. We don't need to be concerned with the projection step since the Pythagorean Theorem implies that the projection step does not hurt [HW01].

Theorem 3.2. *Fix dimension n, set size k and gain constant c. The regret after T trials of the GD algorithm with learning rate $\eta = \sqrt{\frac{k+\frac{k(n-k)}{n}}{(4c^2+1)T}}$ and initial parameters $\boldsymbol{\mu}_1 = \boldsymbol{0}$ and $\boldsymbol{D}_1 = \frac{k}{n}\boldsymbol{I}$ is upper bounded by $\sqrt{2(4c^2+1)\left(\frac{n-k}{n}+1\right)kT}$.*

[2]There are several SDP packages (such as CVX) that are guaranteed to output the value of the SDP up to an additive error of ϵ in time polynomial in the size of the program description and $\log \frac{1}{\epsilon}$.

Proof. Let $\boldsymbol{W} = \begin{bmatrix} \boldsymbol{\mu} & \boldsymbol{D} \end{bmatrix}$ denote the matrix formed by concatenating column vector $\boldsymbol{\mu}$ and matrix $\boldsymbol{D}$. Similarly, let $\boldsymbol{X} = \begin{bmatrix} 2c\boldsymbol{x} & \boldsymbol{x}\boldsymbol{x}^\intercal \end{bmatrix}$. With this notation, the expected gain $\mathrm{tr}(\boldsymbol{D}\boldsymbol{x}\boldsymbol{x}^\intercal) + 2c\boldsymbol{\mu}^\intercal\boldsymbol{x} + kc^2$ of parameter $(\boldsymbol{\mu}, \boldsymbol{D})$ on instance $\boldsymbol{x}$ becomes $\mathrm{tr}(\boldsymbol{W}\boldsymbol{X}^\intercal) + k\boldsymbol{c}^2$.

For any offline comparator $\boldsymbol{W}^* = \begin{bmatrix} \sum_{i=1}^k \boldsymbol{u}_i & \sum_{i=1}^k \boldsymbol{u}_i\boldsymbol{u}_i^\intercal \end{bmatrix}$, we have

$$\|\boldsymbol{W}_{t+1} - \boldsymbol{W}^*\|_F^2 \leq \|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}^*\|_F^2 = \|\boldsymbol{W}_t - \boldsymbol{W}^*\|_F^2 - 2\eta\,\mathrm{tr}((\boldsymbol{W}^* - \boldsymbol{W}_t)\boldsymbol{X}_t^\intercal) + \eta^2\|\boldsymbol{X}_t\|_F^2,$$

where the inequality follows from the Pythagorean Theorem [HW01]. Since $\boldsymbol{x}_t$ has unit length, $\|\boldsymbol{X}_t\|_F^2 = \left\|\begin{bmatrix} 2c\boldsymbol{x}_t & \boldsymbol{x}_t\boldsymbol{x}_t^\intercal \end{bmatrix}\right\|_F^2 = 4c^2\|\boldsymbol{x}_t\|^2 + \|\boldsymbol{x}_t\boldsymbol{x}_t^\intercal\|_F^2 = 4c^2 + 1$. By rearranging terms, we have

$$\mathrm{tr}(\boldsymbol{W}^*\boldsymbol{X}_t^\intercal) - \mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t^\intercal) \leq \frac{\|\boldsymbol{W}_t - \boldsymbol{W}^*\|_F^2 - \|\boldsymbol{W}_{t+1} - \boldsymbol{W}^*\|_F^2}{2\eta} + \frac{(4c^2+1)\eta}{2}. \quad (3.5)$$

Note that the LHS of (3.5) is the regret in trial t. Summing the inequality over all T trials, we have that the total regret is upper bounded by

$$\frac{\|\boldsymbol{W}_1 - \boldsymbol{W}^*\|_F^2 - \cancel{\|\boldsymbol{W}_{T+1} - \boldsymbol{W}^*\|_F^2}}{2\eta} + \frac{(4c^2+1)\eta T}{2} \leq \frac{k + \frac{k(n-k)}{n}}{2\eta} + \frac{(4c^2+1)\eta T}{2},$$

since $\|\boldsymbol{W}_1 - \boldsymbol{W}^*\|_F^2 = \|\begin{bmatrix} \boldsymbol{0} & \frac{k}{n}\boldsymbol{I} \end{bmatrix} - \begin{bmatrix} \sum_i \boldsymbol{u}_i & \sum_i \boldsymbol{u}_i\boldsymbol{u}_i^\intercal \end{bmatrix}\|_F^2$ is by the rotation invariance of $\|.\|_F^2$ equal to $\|\begin{bmatrix} \boldsymbol{0} & \frac{k}{n}\boldsymbol{I} \end{bmatrix} - \begin{bmatrix} \boldsymbol{1}_k & \boldsymbol{I}_k \end{bmatrix}\|_F^2 = \frac{k(n-k)}{n} + k$. Choosing $\eta = \sqrt{\frac{k + \frac{k(n-k)}{n}}{(4c^2+1)T}}$ proves the theorem. □

We now reason that the above regret bound for GD (expressed as a function of the number of the trials T) cannot be improved by more than a constant factor. We first consider the original online PCA problem, where $c = 0$. In this case our regret bound for GD becomes $\sqrt{2\left(\frac{n-k}{n} + 1\right)kT}$ and a matching lower bound (up to a constant factor) was shown in [NKW13].

Theorem 3.3. *The minimax regret of the T-round directional gain game with constant $c \neq 0$ and orthonormal sets of size k is $\Omega(\sqrt{c^2 kT})$.*

3.4 The Decomposition

In this section we decompose any parameter $(\boldsymbol{\mu}, \boldsymbol{D})$ satisfying (3.4), that is, we write it as a convex combination of (first and second moments of) sets of k orthogonal directions. Our algorithm is a greedy iterative removal scheme, like the decomposition algorithms for sets and subspaces [WK08], permutations [HW09], paths and trees [KWK10].

Note that the condition $\boldsymbol{\mu}\boldsymbol{\mu}^\intercal/k \preceq \boldsymbol{D}$ of Theorem 3.1 is equivalent to the following, where $\boldsymbol{D}^\dagger$ denotes the pseudo-inverse: $\boldsymbol{D} \succeq \boldsymbol{0}$, $\boldsymbol{\mu}\boldsymbol{D}^\dagger\boldsymbol{\mu} \leq k$ and $\boldsymbol{\mu} \in \text{range}(\boldsymbol{D})$(see e.g. [Ber11, Proposition 8.2.4]). It will be convenient to assume that the mean is extreme, i.e. $\boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} = k$. If instead $\boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} < k$ we may decompose by mixing the two decompositions[3] of the extreme opposites $\left(\pm\boldsymbol{\mu}\sqrt{\frac{k}{\boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu}}}, \boldsymbol{D}\right)$ with probabilities $\frac{k\pm\boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu}}{2k}$. (If the mean $\boldsymbol{\mu}$ is zero we may choose any pair of opposites in the range of $\boldsymbol{D}$.) So we henceforth assume that

$$\text{tr}(\boldsymbol{D}) \;=\; k, \qquad \boldsymbol{0} \;\preceq\; \boldsymbol{D} \;\preceq\; \boldsymbol{I}, \qquad \boldsymbol{\mu} \in \text{range}(\boldsymbol{D}) \qquad \text{and} \qquad \boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} \;=\; k. \quad (3.6)$$

This equation implies that the eigenvalues of $\boldsymbol{D}$ lie in $[0,1]$. We proceed by recursion on

$$\chi(\boldsymbol{D}) \;\coloneqq\; \text{the number of eigenvalues of } \boldsymbol{D} \text{ in } (0,1).$$

[3]Each decomposition will be of size $n+1$, for a total of $2(n+1)$.

In the base case $\chi(\boldsymbol{D}) = 0$ all eigenvalues of $\boldsymbol{D}$ are either 0 or 1, and since $\operatorname{tr}(\boldsymbol{D}) = k$ there must be k ones and $n - k$ zeroes. In particular this means that $\boldsymbol{\mu}^\intercal \boldsymbol{\mu} = k$. To obtain an orthonormal set with mean $\boldsymbol{\mu}$ and second moment $\boldsymbol{D}$, we may choose $\boldsymbol{U}$ to be any orthonormal basis spanning the range of $\boldsymbol{D}$ with sum equal to $\boldsymbol{\mu}$.

If $\chi(\boldsymbol{D}) > 0$ we find an orthonormal set $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ (with moments $(\sum_{i=1}^k \boldsymbol{u}_i, \sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{u}_i^\intercal)$ that are abbreviated as $(\boldsymbol{s}, \boldsymbol{S})$ throughout), a probability $\rho \in (0,1)$, and decompose

$$(\boldsymbol{\mu}, \boldsymbol{D}) \;=\; \rho\, (\boldsymbol{s}, \boldsymbol{S}) + (1-\rho)\, (\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{D}}),$$

where the normalized remainder $(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{D}}) := \left(\frac{\boldsymbol{\mu} - \rho \boldsymbol{s}}{1-\rho}, \frac{\boldsymbol{D} - \rho \boldsymbol{S}}{1-\rho} \right)$ again satisfies (3.6) so that it can be decomposed recursively and moreover $\chi(\widetilde{\boldsymbol{D}}) < \chi(\boldsymbol{D})$. This recursive process must therefore terminate in at most $n+1$ steps.

A similar but simpler recursive process is used in the original online PCA problem (where $c = 0$) [WK08]. In this case, the learner only needs to decompose the parameter matrix $\boldsymbol{D}$ into a small mixture of orthonormal sets of size k. These orthonormal sets can always be chosen as subsets of the eigenvectors of $\boldsymbol{D}$. In the general case (when $c \neq 0$), the sets need to simultaneously decompose the mean parameter $\boldsymbol{\mu}$, and the additional constraints this imposes are not generally satisfied by the eigenvectors of $\boldsymbol{D}$.

The rest of this section will be concerned with finding the set $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$ and the probability ρ and proving that $\chi(\widetilde{\boldsymbol{D}}) < \chi(\boldsymbol{D})$. First in Theorem 3.4 we prove that Algorithm 3.1 will find an orthonormal set of k so-called *tangent* directions. We call a direction $\boldsymbol{u}$ *tangent* to $(\boldsymbol{\mu}, \boldsymbol{D})$ if $\boldsymbol{u}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = 1$. Then in Lemma 3.2 we show that splitting off a tangent set $\boldsymbol{U}$ preserves (3.6). Finally in Theorem 3.5 we show that the probability $\rho \in (0,1)$ can be found, and that $\chi(\widetilde{\boldsymbol{D}}) < \chi(\boldsymbol{D})$.

3.4.1 Finding a Tangent Set

In this section we present Algorithm 3.1 for finding a tangent set. The algorithm will make use of the following simple lemma.

Lemma 3.1. *A linear equation $\boldsymbol{v}^\intercal \boldsymbol{x} = a$ of dimension at least 2 has a solution for $\boldsymbol{x}$ of unit length if $\|\boldsymbol{v}\| \geq |a|$.*

Proof. Let $\boldsymbol{v}^\perp$ be a unit vector perpendicular to $\boldsymbol{v}$. If $\|\boldsymbol{v}\| = a = 0$, return $\boldsymbol{v}^\perp$. Otherwise $\frac{a}{\|\boldsymbol{v}\|^2}\boldsymbol{v} + \sqrt{1 - \frac{a^2}{\|\boldsymbol{v}\|^2}}\boldsymbol{v}^\perp$ is a unit length solution. $\square$

We are now ready to show that the algorithm indeed produces a tangent set.

Theorem 3.4. *Let $\boldsymbol{\mu}$ and $\boldsymbol{D}$ satisfy* (3.6)*. Let $[\boldsymbol{A}\ \boldsymbol{B}\ \boldsymbol{C}]$ be an orthonormal eigenbasis for $\boldsymbol{D}$, with $\boldsymbol{A}$ associated to the eigenvalue 1, $\boldsymbol{C}$ to eigenvalue* 0 *and $\boldsymbol{B}$ to the remaining intermediate eigenvalues. (Any of them can be empty). Then Algorithm 3.1 applied to $(\boldsymbol{\mu}, \boldsymbol{D})$ produces a set $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$ of k orthonormal vectors with moments $(\boldsymbol{s}, \boldsymbol{S})$ such that*

$$\boldsymbol{U}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \boldsymbol{1}_k \qquad \textit{$\boldsymbol{U}$ is a tangent set} \tag{3.7a}$$

$$\boldsymbol{S} = \boldsymbol{D}\boldsymbol{D}^\dagger \boldsymbol{S} \qquad \textit{$\boldsymbol{U}$ avoids the 0 eigenspace of $\boldsymbol{D}$} \tag{3.7b}$$

$$\boldsymbol{I} - \boldsymbol{S} = (\boldsymbol{I} - \boldsymbol{D})(\boldsymbol{I} - \boldsymbol{D})^\dagger(\boldsymbol{I} - \boldsymbol{S}) \qquad \textit{$\boldsymbol{U}$ contains the 1 eigenspace of $\boldsymbol{D}$} \tag{3.7c}$$

The algorithm can be implemented in time $O(kn^2)$ when $\boldsymbol{C}\boldsymbol{C}^\intercal$ is precomputed.

Proof. We first show that $\widehat{\boldsymbol{A}}$ consists of k orthonormal vectors and that $\|\widehat{\boldsymbol{A}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}\|^2 = k$. When $\operatorname{rank}(\boldsymbol{A}) = k$, since $\boldsymbol{I} \succeq \boldsymbol{D}$ and $\operatorname{tr}(\boldsymbol{D}) = k$, $\boldsymbol{B}$ is empty and $\boldsymbol{D} = \boldsymbol{D}^\dagger = \boldsymbol{A}\boldsymbol{A}^\intercal$.

$$\|\boldsymbol{A}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}\|^2 = \boldsymbol{\mu}^\intercal \boldsymbol{D}^\dagger \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \boldsymbol{\mu}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = k.$$

When $\operatorname{rank}(\boldsymbol{A}) < k$, $\boldsymbol{D}$ can be eigendecomposed as $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{A}^\intercal + \boldsymbol{B}\widehat{\boldsymbol{D}}\boldsymbol{B}^\intercal$ where $\widehat{\boldsymbol{D}}$ is a diagonal matrix and $\boldsymbol{0} \prec \widehat{\boldsymbol{D}} \prec \boldsymbol{I}$. We rewrite $\boldsymbol{D}^\dagger$, $\boldsymbol{v}_{\boldsymbol{A}}$ and $\boldsymbol{v}_{\boldsymbol{B}}$ with the decomposition as:

$$\boldsymbol{D}^\dagger = \boldsymbol{A}\boldsymbol{A}^\intercal + \boldsymbol{B}\widehat{\boldsymbol{D}}^\dagger \boldsymbol{B}^\intercal, \qquad \boldsymbol{v}_A = \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{\mu}, \qquad \boldsymbol{v}_B = \boldsymbol{B}\boldsymbol{B}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \boldsymbol{B}\widehat{\boldsymbol{D}}^\dagger \boldsymbol{B}^\intercal \boldsymbol{\mu}.$$

Now we show that the conditions for using Lemma 3.1 to compute $\hat{\boldsymbol{v}}$ are met.

- $\operatorname{rank}(\boldsymbol{B}) = \underbrace{\operatorname{rank}(\boldsymbol{A}) + \operatorname{rank}(\boldsymbol{B})}_{>k} - \underbrace{\operatorname{rank}(\boldsymbol{A})}_{<k} \geq 2$. The lower bound on $\operatorname{rank}(\boldsymbol{A}) + \operatorname{rank}(\boldsymbol{B})$ follows from

$$\underbrace{\operatorname{rank}(\boldsymbol{A}) + \operatorname{rank}(\boldsymbol{B}) = \operatorname{tr}(\boldsymbol{A}\boldsymbol{A}^\intercal + \boldsymbol{B}\boldsymbol{B}^\intercal)}_{\boldsymbol{A} \text{ and } \boldsymbol{B} \text{ consist of orthonormal vectors}} \overbrace{\vphantom{)} > \operatorname{tr}(A\boldsymbol{A}^\intercal + \boldsymbol{B}\widehat{\boldsymbol{D}}\boldsymbol{B}^\intercal)}^{\boldsymbol{I} \succ \widehat{\boldsymbol{D}}} = k.$$

- To show $k \geq \|\boldsymbol{v}_A\|^2$, notice that

$$k = \boldsymbol{\mu}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \boldsymbol{\mu}^\intercal (\boldsymbol{A}\boldsymbol{A}^\intercal + \boldsymbol{B}\widehat{\boldsymbol{D}}^\dagger \boldsymbol{B}^\intercal)\boldsymbol{\mu}^\intercal = \underbrace{\boldsymbol{\mu}^\intercal \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{\mu}}_{\|\boldsymbol{v}_A\|^2} + \underbrace{\boldsymbol{\mu}^\intercal \boldsymbol{B}\widehat{\boldsymbol{D}}^\dagger \boldsymbol{B}^\intercal \boldsymbol{\mu}}_{\geq 0}$$

- Finally $\|\boldsymbol{v}_B\| \geq \sqrt{k - \|\boldsymbol{v}_A\|^2}$ follows from $(\widehat{\boldsymbol{D}}^\dagger)^2 \succeq \widehat{\boldsymbol{D}}^\dagger$ and

$$\|\boldsymbol{v}_B\|^2 = \boldsymbol{\mu}^\intercal \boldsymbol{B}(\widehat{\boldsymbol{D}}^\dagger)^2 \boldsymbol{B}^\intercal \boldsymbol{\mu} \geq \boldsymbol{\mu}^\intercal \boldsymbol{B}\widehat{\boldsymbol{D}}^\dagger \boldsymbol{B}^\intercal \boldsymbol{\mu} = k - \|v_A\|^2.$$

The next step is to show that finding $\widehat{\boldsymbol{B}}$ is always possible. This follows simply from $k - \mathrm{rank}(\boldsymbol{A}) - 1 \leq (\mathrm{rank}(\boldsymbol{A}) + \mathrm{rank}(\boldsymbol{B}) - 1) - \mathrm{rank}(\boldsymbol{A}) - 1 = \mathrm{rank}(\boldsymbol{B}) - 2$. Since $\boldsymbol{A}$, $\hat{\boldsymbol{v}}$ and $\widehat{\boldsymbol{B}}$ are orthogonal to each other and $\widehat{\boldsymbol{B}}$ is also orthogonal to $\boldsymbol{D}^\dagger \boldsymbol{\mu}$, $\|\widehat{\boldsymbol{A}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}\|^2 = \|\boldsymbol{v}_A\|^2 + (\hat{\boldsymbol{v}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu})^2 = k$.

So in both cases, $\widehat{\boldsymbol{A}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}$ is a vector in R^k with length $\sqrt{k}$. By a rotation matrix $\widehat{\boldsymbol{U}}$ in $\mathrm{R}^{k \times k}$, we can rotate $\widehat{\boldsymbol{A}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}$ to a vector of the same length, $\mathbf{1}_k$(see e.g. [HKW15]). As a result, $\boldsymbol{U}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} = \widehat{\boldsymbol{U}} \widehat{\boldsymbol{A}}^\intercal \boldsymbol{D} \boldsymbol{\mu} = \mathbf{1}_k$ and $\boldsymbol{U}^\intercal \boldsymbol{U} = \widehat{\boldsymbol{U}} \widehat{\boldsymbol{A}}^\intercal \widehat{\boldsymbol{A}} \widehat{\boldsymbol{U}} = \boldsymbol{I}_k$.

Finally, noticing that by construction of $\widehat{\boldsymbol{A}}$, $\mathrm{range}(\boldsymbol{U}) \in \mathrm{range}(\boldsymbol{D})$, $\boldsymbol{U} = \boldsymbol{D}\boldsymbol{D}^\dagger \boldsymbol{U}$. Also,

$$\boldsymbol{I} - \boldsymbol{S} = \boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}^\intercal - \hat{\boldsymbol{v}}\hat{\boldsymbol{v}}^\intercal - \widehat{\boldsymbol{B}}\widehat{\boldsymbol{B}}^\intercal = \boldsymbol{B}\boldsymbol{B} + \boldsymbol{C}\boldsymbol{C} - \hat{\boldsymbol{v}}\hat{\boldsymbol{v}}^\intercal - \widehat{\boldsymbol{B}}\widehat{\boldsymbol{B}}^\intercal$$

means $\mathrm{range}(\boldsymbol{I} - \boldsymbol{S}) \in \mathrm{range}(\boldsymbol{B}\boldsymbol{B}^\intercal + \boldsymbol{C}\boldsymbol{C}^\intercal) = \mathrm{range}((\boldsymbol{I} - \boldsymbol{D})(\boldsymbol{I} - \boldsymbol{D})^\dagger)$ as required.

Now we show how to implement the algorithm in $O(kn^2)$ with precomputed $\boldsymbol{C}\boldsymbol{C}^\intercal$. First computing $\boldsymbol{A}$ can be done in $O(kn^2)$ time. Noticing that $\boldsymbol{A}\boldsymbol{A}^\intercal + \boldsymbol{B}\boldsymbol{B}^\intercal + \boldsymbol{C}\boldsymbol{C}^\intercal = \boldsymbol{I}$, we obtain $\boldsymbol{B}\boldsymbol{B}^\intercal$ in $O(n^2)$. Using columns of $\boldsymbol{B}\boldsymbol{B}^\intercal$ as a basis of $\boldsymbol{B}$, $\widehat{\boldsymbol{B}}$ can be computed in $(k^2 n)$ with a Gram-Schmidt process. Finally, computing a rotation matrix in $\mathrm{SO}(k)$ needs time $O(k^2)$ and computing $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{U}}$ needs time $O(kn^2)$. $\square$

3.4.2 Removing a Tangent Set Preserves the Mean Constraints

At this point we have a tangent set $\boldsymbol{U}$ to split off. We now show that the remainder $(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{D}})$ satisfies (3.6). We start with the rightmost two conditions, which will be satisfied for any weight $\rho > 0$. Lemma 3.2 covers a single tangent vector, whereas

Algorithm 3.1 Find a removable set $\boldsymbol{U}$

```
**Input**: parameter $(\boldsymbol{\mu}, \boldsymbol{D})$ satisfying (3.6)
**Output**:orthonormal $k$-set $\boldsymbol{U}$ satisfying (3.7)
Compute orthonormal eigenbasis $\boldsymbol{A}$ and $\boldsymbol{B}$ of $\boldsymbol{D}$ as described in Theorem 3.4
**if** rank$(\boldsymbol{A}) = k$ **then**
    $\widehat{\boldsymbol{A}} = \boldsymbol{A}$
**else**
    $\boldsymbol{v}_A = \boldsymbol{A}\boldsymbol{A}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}$            // Project $\boldsymbol{D}^\dagger \boldsymbol{\mu}$ on $\boldsymbol{A}$
    $\boldsymbol{v}_B = \boldsymbol{B}\boldsymbol{B}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}$            // Project $\boldsymbol{D}^\dagger \boldsymbol{\mu}$ on $\boldsymbol{B}$
    Compute a unit vector $\hat{\boldsymbol{v}}$ in $\boldsymbol{B}$ satisfying $\hat{\boldsymbol{v}}^\intercal \boldsymbol{v}_B = \sqrt{k - \|\boldsymbol{v}_A\|^2}$ via Lemma 3.1
    Pick $k - \text{rank}(\boldsymbol{A}) - 1$ orthonormal basis $\widehat{\boldsymbol{B}}$ from the complementary of $[\boldsymbol{v}_B, \hat{\boldsymbol{v}}]$ in $\boldsymbol{B}$
    $\widehat{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \hat{\boldsymbol{v}} & \widehat{\boldsymbol{B}} \end{bmatrix}$
**end if**
Compute a rotation matrix $\widehat{\boldsymbol{U}} \in \text{SO}(k)$ which rotates $\widehat{\boldsymbol{A}}^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu}$ to $\mathbf{1}_k$
**return** $\boldsymbol{U} = \widehat{\boldsymbol{A}}\widehat{\boldsymbol{U}}^\intercal$
```

Lemma 3.3 covers sets.

Lemma 3.2. *Fix a matrix* $\boldsymbol{D} \in \mathrm{R}^{n\times n}$, *vectors* $\boldsymbol{\mu}, \boldsymbol{u} \in \operatorname{range}(\boldsymbol{D})$ *with* $\boldsymbol{u}\boldsymbol{D}^\dagger\boldsymbol{\mu} = 1$ *and a weight* $\rho \in \mathrm{R}$. *Define* $\widetilde{\boldsymbol{D}} \coloneqq \boldsymbol{D} - \rho\boldsymbol{u}\boldsymbol{u}^\intercal$ *and* $\widetilde{\boldsymbol{\mu}} \coloneqq \boldsymbol{\mu} - \rho\boldsymbol{u}$. *Then*

$$\widetilde{\boldsymbol{\mu}}^\intercal\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = \boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} - \rho \quad \textit{and} \quad \widetilde{\boldsymbol{\mu}} \in \operatorname{range}(\widetilde{\boldsymbol{D}}).$$

If $\operatorname{rank}(\widetilde{\boldsymbol{D}}) = \operatorname{rank}(\boldsymbol{D})$ *then* $\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = \boldsymbol{D}^\dagger\boldsymbol{\mu}$. *Otherwise there is a real number* α *such that*

$$\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = \boldsymbol{D}^\dagger\boldsymbol{\mu} + \alpha\boldsymbol{D}^\dagger\boldsymbol{u} \quad \textit{and} \quad \widetilde{\boldsymbol{D}}\boldsymbol{D}^\dagger\boldsymbol{u} = 0.$$

Proof. First notice that $\operatorname{rank}(\boldsymbol{D}) - 1 \le \operatorname{rank}(\widetilde{\boldsymbol{D}}) \le \operatorname{rank}(\boldsymbol{D})$, since $\boldsymbol{u} \in \mathrm{R}^n$ and $\rho\boldsymbol{u}^\intercal\boldsymbol{u}$ is a rank one modification. So $\operatorname{rank}(\widetilde{\boldsymbol{D}})$ equals either $\operatorname{rank}(\boldsymbol{D})$ or $\operatorname{rank}(\boldsymbol{D}) - 1$. We cover these two cases separately. In the first case when $\operatorname{rank}(\widetilde{\boldsymbol{D}}) = \operatorname{rank}(\boldsymbol{D})$ we have

$$\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} \;=\; (\boldsymbol{D} - \rho\boldsymbol{u}\boldsymbol{u}^\intercal)^\dagger(\boldsymbol{\mu} - \rho\boldsymbol{u}) \;=\; \left(\boldsymbol{D}^\dagger + \rho\frac{\boldsymbol{D}^\dagger\boldsymbol{u}\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger}{1 - \rho\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u}}\right)(\boldsymbol{\mu} - \rho\boldsymbol{u}) \;=\; \boldsymbol{D}^\dagger\boldsymbol{\mu}$$

by [Ber11, Fact 6.4.2]. And so $\widetilde{\boldsymbol{\mu}}\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = (\boldsymbol{\mu} - \rho\boldsymbol{u})^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} = \boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} - \rho$. Also in this case $\widetilde{\boldsymbol{\mu}} \in \operatorname{range}(\boldsymbol{D}) = \operatorname{range}(\widetilde{\boldsymbol{D}})$.

In the second case $\operatorname{rank}(\widetilde{\boldsymbol{D}}) = \operatorname{rank}(\boldsymbol{D}) - 1$ or equivalently $\rho\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} = 1$. We first show $\boldsymbol{D}^\dagger\boldsymbol{u}$ is a null vector of $\widetilde{\boldsymbol{D}}$.

$$\widetilde{\boldsymbol{D}}\boldsymbol{D}^\dagger\boldsymbol{u} \;=\; (\boldsymbol{D} - \rho\boldsymbol{u}\boldsymbol{u}^\intercal)\boldsymbol{D}^\dagger\boldsymbol{u} \;=\; \boldsymbol{D}\boldsymbol{D}^\dagger\boldsymbol{u} - \boldsymbol{u}\rho\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} \;=\; \boldsymbol{u} - \boldsymbol{u} = 0$$

Notice that $\boldsymbol{D}\boldsymbol{D}^\dagger\boldsymbol{u} = \boldsymbol{u} \ne 0$, so $\operatorname{range}(\widetilde{\boldsymbol{D}})$ is exactly the complementary space of $\boldsymbol{D}^\dagger\boldsymbol{u}$ in $\operatorname{range}(\boldsymbol{D})$. This implies $\widetilde{\boldsymbol{\mu}} \in \operatorname{range}(\widetilde{\boldsymbol{D}})$ since $\boldsymbol{D}^\dagger\boldsymbol{u}$ is also null to $\widetilde{\boldsymbol{\mu}}$:

$$\widetilde{\boldsymbol{\mu}}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} = (\boldsymbol{\mu} - \rho\boldsymbol{u})^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} = \boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} - \rho\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger\boldsymbol{u} = 1 - 1 = 0$$

We now use [Ber11, Fact 6.4.2] to rewrite $\widetilde{\boldsymbol{D}}^\dagger$ (α_i are unimportant scalars)

$$\begin{aligned}\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} =&[\boldsymbol{D}^\dagger + \alpha_1\boldsymbol{D}^\dagger\boldsymbol{u}\boldsymbol{u}^\intercal(\boldsymbol{D}^\dagger)^2 + \underbrace{\alpha_2(\boldsymbol{D}^\dagger)^2\boldsymbol{u}\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger + \alpha_3\boldsymbol{D}^\dagger\boldsymbol{u}\boldsymbol{u}^\intercal\boldsymbol{D}^\dagger}_{\text{(become 0 after distribution)}}]\widetilde{\boldsymbol{\mu}}\\ =&[\boldsymbol{D}^\dagger + \alpha_1\boldsymbol{D}^\dagger\boldsymbol{u}\boldsymbol{u}^\intercal(\boldsymbol{D}^\dagger)^2](\boldsymbol{\mu} - \rho\boldsymbol{u})\\ =&\boldsymbol{D}^\dagger\boldsymbol{\mu} - \rho\boldsymbol{D}^\dagger\boldsymbol{u} + \alpha_1\boldsymbol{D}^\dagger\boldsymbol{u}\underbrace{\boldsymbol{u}^\intercal(\boldsymbol{D}^\dagger)^2(\boldsymbol{\mu} - \rho\boldsymbol{u})}_{\text{a number}} = \boldsymbol{D}^\dagger\boldsymbol{\mu} + \alpha\boldsymbol{D}^\dagger\boldsymbol{u}.\end{aligned}$$

The last thing to show is $\widetilde{\boldsymbol{\mu}}\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = \boldsymbol{\mu}\boldsymbol{D}^\dagger\boldsymbol{\mu} - \rho$ which follows by

□

$$\widetilde{\boldsymbol{\mu}}\widetilde{\boldsymbol{D}}^\dagger\widetilde{\boldsymbol{\mu}} = \widetilde{\boldsymbol{\mu}}^\intercal(\boldsymbol{D}^\dagger\boldsymbol{\mu} + \alpha\boldsymbol{D}^\dagger\boldsymbol{u}) = (\boldsymbol{\mu} - \rho\boldsymbol{u})^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} = \boldsymbol{\mu}^\intercal\boldsymbol{D}^\dagger\boldsymbol{\mu} - \rho.$$

The previous lemma covered single tangent vectors. Next we take out a full tangent set.

Lemma 3.3. *Let $\boldsymbol{\mu}, \boldsymbol{D}$ satisfy* (3.6)*, and let the orthonormal set $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ (with moments $(\boldsymbol{s}, \boldsymbol{S})$) be tangent. Then for any $\rho > 0$ if $\boldsymbol{D} \succeq \rho\boldsymbol{S}$, we have*

$$(\boldsymbol{\mu} - \rho\boldsymbol{s})^\intercal(\boldsymbol{D} - \rho\boldsymbol{S})(\boldsymbol{\mu} - \rho\boldsymbol{s}) \ = \ \boldsymbol{\mu}^\intercal\boldsymbol{D}\boldsymbol{\mu} - k\rho \qquad \textit{and} \qquad \boldsymbol{\mu} - \rho\boldsymbol{s} \in \operatorname{range}(\boldsymbol{D} - \rho\boldsymbol{S}).$$

Proof. For $1 \le d \le k$, define the intermediate remainder as $\widetilde{\boldsymbol{\mu}}_d \coloneqq \boldsymbol{\mu} - \rho\sum_{i=1}^d \boldsymbol{u}_i$ and $\widetilde{\boldsymbol{D}}_d \coloneqq \boldsymbol{D} - \rho\sum_{i=1}^d \boldsymbol{u}_i\boldsymbol{u}_i^\intercal$. Also $\widetilde{\boldsymbol{D}}_0 = \boldsymbol{D}$ and $\widetilde{\boldsymbol{\mu}}_0 = \boldsymbol{\mu}$. We show by induction that $\boldsymbol{u}_i$ remains tangent to $(\widetilde{\boldsymbol{\mu}}_d, \widetilde{\boldsymbol{D}}_d)$ for $d < i \le k$ and

$$\widetilde{\boldsymbol{\mu}}_d^\intercal\widetilde{\boldsymbol{D}}_d\widetilde{\boldsymbol{\mu}}_d = \boldsymbol{\mu}^\intercal\boldsymbol{D}\boldsymbol{\mu} - d\rho \qquad \text{and} \qquad \widetilde{\boldsymbol{\mu}}_d \in \operatorname{range}(\widetilde{\boldsymbol{D}}_d).$$

The base case $d = 0$ is trivial. Let us, to simplify notation, show the induction step for $d = 1$. The last two claims follow directly from Lemma 3.2. We now show that for $2 \le i \le k$, $\boldsymbol{u}_i$ is also tangent to $\widetilde{\boldsymbol{\mu}}_1$ and $\widetilde{\boldsymbol{D}}_1$. When $\operatorname{rank}(\widetilde{\boldsymbol{D}}_1) = \operatorname{rank}(\boldsymbol{D})$ we have

$\boldsymbol{u}_i^\intercal \widetilde{\boldsymbol{D}}_1^\dagger \widetilde{\boldsymbol{\mu}}_1 = \boldsymbol{u}_i^\intercal \boldsymbol{D}\boldsymbol{\mu} = 1$ as required. When $\text{rank}(\widetilde{\boldsymbol{D}}_1) = \text{rank}(\boldsymbol{D}) - 1$,

$$\boldsymbol{u}_i^\intercal \widetilde{\boldsymbol{D}}_1^\dagger \widetilde{\boldsymbol{\mu}}_1 = \boldsymbol{u}_i^\intercal \boldsymbol{D}^\dagger \boldsymbol{\mu} - \alpha \boldsymbol{u}_i^\intercal \boldsymbol{D}^\dagger \boldsymbol{u}_1 = 1 - \alpha \boldsymbol{u}_i^\intercal \boldsymbol{D}^\dagger \boldsymbol{u}_1.$$

Note that $\boldsymbol{u}_i^\intercal \boldsymbol{D}^\dagger \boldsymbol{u}_1 = 0$, for otherwise, $(\boldsymbol{D}^\dagger \boldsymbol{u}_1)^\intercal (\widetilde{\boldsymbol{D}}_1 - \rho \boldsymbol{u}_i \boldsymbol{u}_i^\intercal)(\boldsymbol{D}^\dagger \boldsymbol{u}_1) = -\rho(\boldsymbol{u}_i^\intercal \boldsymbol{D}^\dagger \boldsymbol{u}_1)^2 < 0$, which contradicts $\widetilde{\boldsymbol{D}}_1 - \rho \boldsymbol{u}_i \boldsymbol{u}_i^\intercal \succeq \widetilde{\boldsymbol{D}}_k \succeq \boldsymbol{0}$. This also implies $\boldsymbol{u}_i \in \text{range}(\widetilde{\boldsymbol{D}}_1)$ which means $\boldsymbol{u}_i$ is tangent to $\widetilde{\boldsymbol{\mu}}_1$ and $\widetilde{\boldsymbol{D}}_1$. □

3.4.3 Choosing the Weight ρ

We know that taking out a tangent set $\boldsymbol{U}$ preserves the rightmost two constraints of (3.6) on the remainder for any weight ρ. To satisfy the leftmost two, we investigate how semi-definiteness and rank of $\widetilde{\boldsymbol{D}}$ are related to ρ.

Lemma 3.4. *Let $\boldsymbol{D}, \boldsymbol{S} \in \mathrm{R}^{n\times n}$ be non-zero positive semi-definite matrices with $\boldsymbol{S} = \boldsymbol{D}\boldsymbol{D}^\dagger \boldsymbol{S}$. Define $\rho_s := \frac{1}{\lambda_{\max}(\boldsymbol{D}^\dagger \boldsymbol{S})}$ where $\lambda_{\max}(\boldsymbol{M})$ is the largest eigenvalue of $\boldsymbol{M}$. Then the following hold for $\widetilde{\boldsymbol{D}} = \boldsymbol{D} - \rho \boldsymbol{S}$:*

- $0 < \rho_s < \infty$
- $\widetilde{\boldsymbol{D}} \succeq \boldsymbol{0}$ *for any* $\rho \leq \rho_s$
- $\text{rank}(\widetilde{\boldsymbol{D}}) \leq \text{rank}(\boldsymbol{D})$, *and* $\text{rank}(\widetilde{\boldsymbol{D}}) < \text{rank}(\boldsymbol{D})$ *when* $\rho = \rho_s$.

Proof. First notice that $\boldsymbol{S} = \boldsymbol{D}\boldsymbol{D}^\dagger \boldsymbol{S}$ implies both $\boldsymbol{S} \in \text{range}(\boldsymbol{D})$ and $\widetilde{\boldsymbol{D}} = \boldsymbol{D} - \rho \boldsymbol{S} \in \text{range}(\boldsymbol{D})$. So $\text{rank}(\widetilde{\boldsymbol{D}}) \leq \text{rank}(\boldsymbol{D})$. Next, $0 < \rho_s < \infty$ follows from that $\boldsymbol{D}^\dagger \boldsymbol{S}$ is non-zero and positive semi-definite. To show $\widetilde{\boldsymbol{D}} \succeq \boldsymbol{0}$, consider an eigenpair $(\boldsymbol{v}, p)$ of $\widetilde{\boldsymbol{D}}$ where

$\boldsymbol{v}$ is a unit vector.

$$\boldsymbol{v}^\intercal \boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{v} = \frac{\boldsymbol{v}^\intercal \boldsymbol{D}^\dagger}{\rho}(\boldsymbol{D}\boldsymbol{v} - (\boldsymbol{D} - \rho\boldsymbol{S})\boldsymbol{v}) = \frac{\boldsymbol{v}^\intercal \boldsymbol{D}^\dagger}{\rho}(\boldsymbol{D}\boldsymbol{v} - p\boldsymbol{v}) = \frac{1}{\rho} - \frac{p}{\rho}\boldsymbol{v}^\intercal \boldsymbol{D}^\dagger \boldsymbol{v}.$$

When $\rho \leq \rho_s$, $\frac{1}{\rho} \geq \lambda_{\max}(\boldsymbol{D}^\dagger \boldsymbol{S}) \geq \boldsymbol{v}^\intercal \boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{v}$ which implies $p \geq 0$. So $\widetilde{\boldsymbol{D}} \succeq \boldsymbol{0}$.

When $\rho = \rho_s$, let $\boldsymbol{x}$ be a eigenvector of eigenvalue $\frac{1}{\rho_s}$: $\boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{x} = \frac{1}{\rho_s}\boldsymbol{x} \neq 0$ and notice that

$$\boldsymbol{D}(\boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{x}) = \boldsymbol{S}\boldsymbol{x} \neq 0 \qquad \widetilde{\boldsymbol{D}}(\boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{x}) = (\boldsymbol{D} - \rho\boldsymbol{S})\boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{x} = \boldsymbol{D}\boldsymbol{D}^\dagger \boldsymbol{S}\boldsymbol{x} - \rho\boldsymbol{S}\frac{1}{\rho}\boldsymbol{x} = 0.$$

So $\widetilde{\boldsymbol{D}}$ has at least one more null dimension than $\boldsymbol{D}$. Together with $\widetilde{\boldsymbol{D}} \in \operatorname{range}(\boldsymbol{D})$, this implies $\operatorname{rank}(\widetilde{\boldsymbol{D}}) < \operatorname{rank}(\boldsymbol{D})$. □

Finally, we are able to choose ρ to in addition satisfy the leftmost two conditions of (3.6), and reduce the complexity $\chi(\widetilde{\boldsymbol{D}})$.

Theorem 3.5. *Let* $(\boldsymbol{\mu}, \boldsymbol{D})$ *satisfy* (3.6). *Let* $\boldsymbol{U}$ *be the output of Algorithm 3.1, and let*

$$\rho \;=\; \min\left\{\frac{1}{\lambda_{\max}(\boldsymbol{D}^\dagger \boldsymbol{S})}, \frac{1}{\lambda_{\max}\big((\boldsymbol{I} - \boldsymbol{D})^\dagger(\boldsymbol{I} - \boldsymbol{S})\big)}\right\}.$$

Then the normalized remainder $(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{D}}) = \left(\frac{\boldsymbol{\mu} - \rho\boldsymbol{s}}{1-\rho}, \frac{\boldsymbol{D} - \rho\boldsymbol{S}}{1-\rho}\right)$ *satisfies* (3.6) *and* $\chi(\widetilde{\boldsymbol{D}}) < \chi(\boldsymbol{D})$.

Proof. If $\boldsymbol{I} \succeq \boldsymbol{D} \succeq \boldsymbol{0}$, since $\rho \leq \frac{1}{\lambda_{\max}(\boldsymbol{D}^\dagger \boldsymbol{S})}$, by Lemma 3.4 $\boldsymbol{D} - \rho\boldsymbol{S} \succeq \boldsymbol{0}$ and so $\widetilde{\boldsymbol{D}} \succeq \boldsymbol{0}$. Also since $(\boldsymbol{I} - \boldsymbol{D}) \succeq \boldsymbol{0}$ and $\rho \leq \frac{1}{\lambda_{\max}((\boldsymbol{I} - \boldsymbol{D})^\dagger(\boldsymbol{I} - \boldsymbol{S}))}$, $(\boldsymbol{I} - \boldsymbol{D}) - \rho(\boldsymbol{I} - \boldsymbol{S}) \succeq \boldsymbol{0}$ which is equivalent to $\boldsymbol{I} \succeq \widetilde{\boldsymbol{D}}$. Also

$$\operatorname{tr}(\widetilde{\boldsymbol{D}}) = \frac{\operatorname{tr}(\boldsymbol{D}) - \rho\operatorname{tr}(\boldsymbol{S})}{1-\rho} = \frac{k - \rho k}{1 - \rho} = k.$$

Since all $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ are tangent, we may apply Lemma 3.3 to show that $\widetilde{\boldsymbol{\mu}}\widetilde{\boldsymbol{D}}^{\dagger}\widetilde{\boldsymbol{\mu}} = k$ and $\widetilde{\boldsymbol{\mu}} \in \text{range}(\widetilde{\boldsymbol{D}})$. By Lemma 3.4, $\text{rank}(\widetilde{\boldsymbol{D}}) \leq \text{rank}(\boldsymbol{D})$ and $\text{rank}(\boldsymbol{I} - \widetilde{\boldsymbol{D}}) \leq \text{rank}(\boldsymbol{I} - \boldsymbol{D})$, where at least one inequality is strict since ρ equals the minimum of $\frac{1}{\lambda_{\max}(\boldsymbol{D}^{\dagger}\boldsymbol{S})}$ and $\frac{1}{\lambda_{\max}((\boldsymbol{I}-\boldsymbol{D})^{\dagger}(\boldsymbol{I}-\boldsymbol{S}))}$. Finally observe that $\chi(\boldsymbol{D}) = \text{rank}(\boldsymbol{D}) + \text{rank}(\boldsymbol{I} - \boldsymbol{D}) - n$ so that $\chi(\widetilde{\boldsymbol{D}}) < \chi(\boldsymbol{D})$. □

To implement the decomposition efficiently, one may want to compute $\boldsymbol{D}^{\dagger}$ incrementally by doing k rank one pseudo-inverse updates for each set peeled off. Since each of these updates needs $O(n^2)$, peeling one set off can be completed in $O(kn^2)$. Notice that Algorithm 3.1 can also be implemented in $O(kn^2)$ (see Theorem 3.4) with a projector of the null space of $\boldsymbol{D}$ incrementally maintained using Lemma 3.2. Combining the two parts gives a $O(kn^3)$ implementation for the entire decomposition process.

3.5 Conclusion

A new use of kernels is emerging from this line of research: The gain/loss is a kernel $k(\boldsymbol{u}, \boldsymbol{x}) = \phi(\boldsymbol{u})^{\intercal}\phi(\boldsymbol{x})$, the parameter space consists of all possible expectations $\mathrm{E}[\phi(\boldsymbol{u})]$, and after the update, the algorithm projects back into this parameter space. Finally any parameter is decomposed into a small mixture of $\phi(\boldsymbol{u})$, and thus the parameter is expressed in terms of the original domain of the feature map ϕ. We showed here how to do this for a simple quadratic kernel, and the work on Component Hedge can be reinterpreted as following this outline. However, what are the ingredients needed for the method to succeed in general? For example can this be done for higher order polynomial kernels?

In our treatment all instances $\boldsymbol{x}$ were assumed to be unit length. Ideally we want to learn vectors of varying length. To do this, more work first needs to be done on developing expert updates that can handle unbounded losses (see e.g. [McM13] for a start). This work should be transferable to the matrix domain.

We believe that the richer modeling capability developed in this chapter will make the use of matrix parameters imperative. However, one of the main criticism of this line of research is that it relies on eigendecompositions that require $O(n^3)$ time. The key open problem is to develop $O(n^2)$ algorithms without degrading the regret bounds too much (See e.g. discussions in [HKW10c]).

Appendix

3.A Range of the Gain

We first determine the range of the gain during a single trial. Fix any set of k orthogonal directions $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ and let $\boldsymbol{\mu} = \sum_{i=1}^{k} \boldsymbol{u}_i$ so that $\|\boldsymbol{\mu}\| = \sqrt{k}$. Let $\boldsymbol{x}$ be a direction, and let $\hat{\boldsymbol{x}} = \sum_{i=1}^{k} (\boldsymbol{u}_i^\intercal \boldsymbol{x}) \boldsymbol{u}_i$ be the projection of $\boldsymbol{x}$ on the set. This notation allows us to write

$$\sum_{i=1}^{k} (\boldsymbol{x}^\intercal \boldsymbol{u}_i + c)^2 \;=\; \|\hat{\boldsymbol{x}}\|^2 + 2c\,\hat{\boldsymbol{x}}^\intercal \boldsymbol{\mu} + kc^2.$$

Using Cauchy-Schwartz, i.e. $(\hat{\boldsymbol{x}}^\intercal \boldsymbol{\mu})^2 \leq \|\hat{\boldsymbol{x}}\| \|\boldsymbol{\mu}\|$, the gain can be sandwiched as follows:

$$(\|\hat{\boldsymbol{x}}\| - \sqrt{k}c)^2 \;=\; \|\hat{\boldsymbol{x}}\|^2 - 2\sqrt{k}c\|\hat{\boldsymbol{x}}\| + kc^2 \;\leq\; \sum_{i=1}^{k} (\boldsymbol{x}^\intercal \boldsymbol{u}_i + c)^2 \;\leq\; \|\hat{\boldsymbol{x}}\|^2 + 2\sqrt{k}c\|\hat{\boldsymbol{x}}\| + kc^2 \;=\; (\|\hat{\boldsymbol{x}}\| + \sqrt{k}c)^2.$$

Recall that Cauchy-Schwartz holds with equality when $\hat{\boldsymbol{x}}$ and $\boldsymbol{\mu}$ are parallel. For $c \geq 0$, the gain is hence maximized at $\boldsymbol{x} = \hat{\boldsymbol{x}} = \boldsymbol{\mu}/\sqrt{k}$, where it takes value $(1 + \sqrt{k}c)^2$. Minimization is slightly more complicated. If $\sqrt{k}c \geq 1$, the gain is minimized at $\boldsymbol{x} = \hat{\boldsymbol{x}} = -\boldsymbol{\mu}/\sqrt{k}$, i.e. the reverse of the maximizer, where it takes value $(1 - \sqrt{k}c)^2$. If on the other hand $\sqrt{k}c \leq 1$, the gain is minimized when $\hat{\boldsymbol{x}} = -c\boldsymbol{\mu}$. This means that we can choose any $\boldsymbol{x} = \hat{\boldsymbol{x}} + \boldsymbol{x}_\perp$, where $\boldsymbol{x}_\perp$ is any vector of length $\sqrt{1 - kc^2}$ that is perpendicular to all $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$. Now the gain takes value 0.

3.B When Do Solutions to the Problems of Learning k Directions and k-PCA Coincide?

Let $\boldsymbol{R}$ and $\boldsymbol{r}$ denote $\sum_t \boldsymbol{x}_t\boldsymbol{x}_t^\intercal$ and $\sum_t \boldsymbol{x}_t$, respectively. Let $\underbrace{\boldsymbol{U}\boldsymbol{U}^\intercal}_{(n,k)\times(k,n)}$ be the rank k projection matrix for the solution subspace of the PCA problem.

Lemma 3.5. *If $\boldsymbol{r}$ lies in the subspace of the k-PCA solution, i.e. $\boldsymbol{U}\boldsymbol{U}^\intercal\boldsymbol{r} = \boldsymbol{r}$, then there is an orthonormal basis $\widehat{\boldsymbol{U}}$ s.t. $\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\intercal = \boldsymbol{U}\boldsymbol{U}^\intercal$ which is also the solution of learning of k directions problem.*

Proof. The gains relate as follows:

$$\overbrace{\underbrace{\mathrm{tr}(\boldsymbol{U}\boldsymbol{U}^\intercal\boldsymbol{R})}_{\text{PCA gain}} + 2c\, \boldsymbol{1}^\intercal\boldsymbol{U}^\intercal\boldsymbol{r}}^{\text{directional gain}}.$$

Since $\boldsymbol{U}\boldsymbol{U}^\intercal\boldsymbol{r} = \boldsymbol{r}$, there is an orthonormal basis $\widehat{\boldsymbol{U}}$ s.t. $\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\intercal = \boldsymbol{U}\boldsymbol{U}^\intercal$, and $\widehat{\boldsymbol{U}}\boldsymbol{1}$ and $\boldsymbol{r}$ point in the same direction. So $\boldsymbol{U} = \widehat{\boldsymbol{U}}$ maximizes $\boldsymbol{1}^\intercal\boldsymbol{U}^\intercal\boldsymbol{r}$. On the other hand, since $\boldsymbol{U}\boldsymbol{U}^\intercal$ is the solution subspace of PCA, and $\boldsymbol{U}\boldsymbol{U}^\intercal = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\intercal$, $\boldsymbol{U} = \widehat{\boldsymbol{U}}$ also maximizes the PCA gain $\mathrm{tr}(\boldsymbol{U}\boldsymbol{U}^\intercal\boldsymbol{R})$. This means that $\widehat{\boldsymbol{U}}$ maximizes both terms of the directional PCA gain and is a solution to both problems. □

3.C Proof of the Lower bound (Theorem 3.3)

First notice that for any distribution $\mathcal{P}$ on instance sequences $\boldsymbol{x}_{1\ldots T}$, the minimax regret of the game is lower bounded by the difference

$$\mathrm{E}_{\boldsymbol{x}_{1\ldots T}\sim\mathcal{P}}[G_C] - \max_{\text{alg. A}} \mathrm{E}_{\boldsymbol{x}_{1\ldots T}\sim\mathcal{P}}[G_A], \tag{3.8}$$

where G_C is the gain of the comparator (i.e. the best set of k orthogonal directions) chosen in hindsight and G_A is the gain of algorithm A.

In our lower bound, we use a $\mathcal{P}$ that is i.i.d. between trails and at each trial gives probability $\frac{1}{2}$ to each of the following two opposite instances,

$$\boldsymbol{x}_+ \;\coloneqq\; (\underbrace{1/\sqrt{k},\ldots,1/\sqrt{k}}_{k},\; \underbrace{0,\ldots,0}_{n-k}) \quad \text{and} \quad \boldsymbol{x}_- \;\coloneqq\; -(\underbrace{1/\sqrt{k},\ldots,1/\sqrt{k}}_{k},\; \underbrace{0,\ldots,0}_{n-k}).$$

Now we lower bound the difference in (3.8) for this particular choice of $\mathcal{P}$. We first lower bound the gain of the comparator by the gain of the best of two orthonormal sets, either $\{\boldsymbol{e}_1,\ldots,\boldsymbol{e}_k\}$ or $\{-\boldsymbol{e}_1,\ldots,-\boldsymbol{e}_k\}$ (these sets maximize the gain on $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ respectively).

$$\mathrm{E}_{\boldsymbol{x}_{1\ldots T}\sim\mathcal{P}}[G_C] \;\geq\; \mathrm{E}_{\boldsymbol{x}_{1\ldots T}\sim\mathcal{P}}\left[\sum_{t=1}^{T}\boldsymbol{x}_t^\intercal\boldsymbol{D}\boldsymbol{x}_t + 2c\max\left\{\boldsymbol{\mu}_+^\intercal\sum_{t=1}^{T}\boldsymbol{x}_t, \boldsymbol{\mu}_-^\intercal\sum_{t=1}^{T}\boldsymbol{x}_t\right\}\right] + Tc^2,$$

where $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ are the first moments of the two sets, that is

$$\boldsymbol{\mu}_+ \;=\; \{\underbrace{1,\ldots,1}_{k},0,\ldots,0\} \qquad \text{and} \qquad \boldsymbol{\mu}_- \;=\; \{\underbrace{-1,\ldots,-1}_{k},0,\ldots 0\},$$

and $\boldsymbol{D} \;=\; \begin{bmatrix}\boldsymbol{I}_k & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$ is the common second moment of both sets. Since we only compare to two sets, the first moment part of the gain is essentially the two experts setting with loss per round equal to $\pm 2c\sqrt{k}$. With analysis in [Koo11], one can show that the first moment part is hence lower bounded by $\Omega(\sqrt{c^2kT})$. The second moment part, noticing that both instances $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ lie in the span of $\boldsymbol{D}$, always attains its maximum T. Finally, since instances are generated independently between trials with expectation zero ($\mathrm{E}[\boldsymbol{x}_t] = \mathbf{0}$), any algorithm has expected gain 0 in the first moment part, and so

$$\max_{\text{alg. A}} \mathrm{E}_{\boldsymbol{x}_{1\ldots T}\sim\mathcal{P}}[G_A] \;\leq\; T(1+c^2).$$

By combining the bounds on comparator and algorithms, we show a $\Omega(\sqrt{c^2kT})$ lower bound of the difference in (3.8) which concludes our proof.

Chapter 4

Learning Rotations with Optimal Regrets

4.1 Introduction

This chapter considers the problem of learning rotations. The goal of this problem is to find the underlying rotation from a given set of examples, i.e. vector pairs consisting of vectors before and after the rotation. Learning rotations is a fundamental problem in computer vision since it has a wide application in various fields, such as robotics, optical character recognition and motion analysis (see motivating examples in [Aro09] and [HKW15] and the references therein). The batch version of this problem was introduced by [Wah65] in which a simple and by now well known solution was presented. In contrast, the online version of this problem is more challenging: It was posed as an open problem in [SW08].

Recently, [HKW10a] tackled this problem of learning rotations online with the Gradient Descend (GD) algorithm. The paper showed that in the worst case, the GD algorithm suffers regret at most $O(\sqrt{nT})$, where n is the dimension of the examples and T is the number of the trials. A matching lower bound of $\Omega(\sqrt{nT})$ was also shown on the worst case regret of any online algorithm. Note that both of these results are time dependent bounds, i.e. they are functions of the number of the trials T. Now recall that there is another type of regret bounds which are functions of loss the best comparator $\mathcal{L}_c$, i.e. the so called loss dependent regret bounds. We argue that the loss dependent bounds are more interesting in practice since in a practical learning problem, there usually exists a good comparator with $\mathcal{L}_c$ much less than the number of the trials T. To this end, [HKW15] shows a loss dependent lower bound of $\Omega(\sqrt{n\mathcal{L}_c} + n)$ on the worst case regret of any online algorithm for this learning problem. [1] However, f This loss dependent bound follows essentially from the previous time dependent bound of $\Omega(\sqrt{nT})$. or the regret upper bounds, there is no easy way to convert time dependent ones into loss dependent ones. In fact, for this problem of learning rotations online, there has not been any result on the loss dependent regret upper bounds from any of the previous work.

This chapter proves for the first time a loss dependent upper bound of $O(\sqrt{n\mathcal{L}_c}+n)$ on the regret of the GD algorithm for learning rotations online. This upper bound follows from two improvements to the previous analysis of the algorithm, which are

[1] Besides of a loss dependent regret lower bound, the major contribution of [HKW15] is a variant of the standard GD algorithm for learning rotations online. We will discuss this variant in the conclusion of this chapter.

first, refining the previous application of Pythagorean Theorem (see (4.7)) and second, adding a new term to our algorithm's measure of progress (see (4.10)). The obtained upper bound of $O(\sqrt{n\mathcal{L}_c} + n)$ matches the previous lower bound of $\Omega(\sqrt{n\mathcal{L}_c} + n)$ and therefore is minimax optimal up to a constant factor.

The rest of this chapter is organized as follows: Section 4.2 gives the preliminaries of the problem of learning rotations online. Section 4.3 describes the GD algorithm which is introduced in [HKW10a] to solve this learning problem. The section also gives an overview of the previous analysis of the algorithm and explains why it can not prove a loss dependent regret bound. Section 4.4 improves the previous analysis with the two aforementioned techniques and proves the main result of this chapter, an $O(\sqrt{n\mathcal{L}_c} + n)$ upper bound on the worst case regret of the GD algorithm. Finally, Section 4.5 concludes the chapter with an open problem.

4.2 Preliminaries

In this chapter, integer n and T are always the dimension of the examples and the number of trials, respectively. All the vectors are in R^n and all the matrices are in $\mathrm{R}^{n\times n}$. We denote the Euclidean norm of a vector by $\|\cdot\|$, i.e. $\|\boldsymbol{x}\| = \sqrt{\sum_i x_i^2} = \sqrt{\boldsymbol{x}^\intercal \boldsymbol{x}}$. Furthermore, a vector $\boldsymbol{x}$ is called a unit vector if $\|\boldsymbol{x}\| = 1$. The Frobenius norm of a matrix $\boldsymbol{W}$ is denoted by $\|\boldsymbol{W}\|_F = \sqrt{\sum_{i,j} w_{i,j}^2} = \sqrt{\mathrm{tr}(\boldsymbol{W}\boldsymbol{W}^\intercal)}$, where $\mathrm{tr}(\boldsymbol{W})$ is the trace of matrix $\boldsymbol{W}$.

Next we give a precise definition of the problem of learning rotations online.

The problem of learning rotations online proceeds in a series of trials. In each trial, it follows the protocol:

In trial $t = 1 \dots T$:

1. The learning algorithm is first given a unit vector $\boldsymbol{x}_t$.
2. Then, it predicts (deterministically or randomly) with a unit vector $\widehat{\boldsymbol{y}}_t$.
3. Nature reveals a true rotated vector, which is also a unit vector $\boldsymbol{y}_t$.
4. The loss to the algorithm is then half of the squared norm of the difference between her predicted vector $\widehat{\boldsymbol{y}}_t$ and the true rotated vector $\boldsymbol{y}_t$:

$$\tfrac{1}{2}\|\boldsymbol{y}_t - \widehat{\boldsymbol{y}}_t\|^2 = 1 - \boldsymbol{y}_t^\intercal \widehat{\boldsymbol{y}}_t. \tag{4.1}$$

If $\widehat{\boldsymbol{y}}_t$ is chosen probabilistically, then we define the expected loss as

$$\mathrm{E}\left[\, \tfrac{1}{2}\|\boldsymbol{y}_t - \widehat{\boldsymbol{y}}_t\|^2 \,\right] = 1 - \boldsymbol{y}_t^\intercal \mathrm{E}\left[\, \widehat{\boldsymbol{y}}_t \,\right]. \tag{4.2}$$

Note that the loss function of this learning problem (see (4.1) and (4.2)) is linear in the prediction vector $\widehat{\boldsymbol{y}}_t$ or the expected prediction vector $\mathrm{E}\left[\, \widehat{\boldsymbol{y}}_t \,\right]$. The goal of the learner is to minimize its regret REG on all T examples against the best fixed rotation chosen in hindsight:

$$\mathrm{REG} = \sum_{t=1}^{T} \mathrm{E}\left[\, \tfrac{1}{2}\|\boldsymbol{y}_t - \widehat{\boldsymbol{y}}_t\|^2 \,\right] - \min_{\boldsymbol{R} \in SO(n)} \sum_{t=1}^{T} \tfrac{1}{2}\|\boldsymbol{y}_t - \boldsymbol{R}\boldsymbol{x}_t\|^2 = \sum_{t=1}^{T} \boldsymbol{y}_t^\intercal \boldsymbol{R}^* \boldsymbol{x}_t - \boldsymbol{y}_t^\intercal \mathrm{E}\left[\, \widehat{\boldsymbol{y}}_t \,\right],$$

where $\boldsymbol{R}^* = \mathrm{argmin}_{\boldsymbol{R} \in SO(n)} \sum_{t=1}^{T} \frac{1}{2}\|\boldsymbol{y}_t - \boldsymbol{R}\boldsymbol{x}_t\|^2$ is the best rotation matrix chosen in hindsight.

4.3 Previous Work

This section describes the GD algorithm for the problem of learning rotations online. The algorithm is introduced in [HKW10a], in which a standard analysis of the algorithm provides an upper bound of $O(\sqrt{nT})$ on its worst case regret. This section will outline this standard analysis and explain why it can not provide a loss dependent bound.

4.3.1 The GD Algorithm for Learning Rotations Online

The GD algorithm for learning rotations online is a probabilistic algorithm, i.e. in each trial, it predicts with a randomly chosen unit vector $\widehat{\boldsymbol{y}}_t$ The algorithm maintains a matrix parameter $\boldsymbol{W}_t$ $(1 \leq t \leq T)$ that is always a convex combination of $n \times n$ rotations matrices. We denote by $\boldsymbol{\mathcal{W}}_{rot}$ the set of all such parameter matrices. [HKW10a] shows that $\boldsymbol{\mathcal{W}}_{rot}$ consists of all the $n \times n$ matrices that have all of their singular values in $[0, 1]$. In each trial, this algorithm processes an example of unit vector pairs $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ as follows: After receiving $\boldsymbol{x}_t$, it first samples a random rotation matrix $\boldsymbol{R}_t$ whose expectation $\mathrm{E}\left[\, \boldsymbol{R}_t \,\right]$ is the algorithm's parameter matrix $\boldsymbol{W}_t$ (see an example of such sampling procedure in [HKW10a]). Given $\boldsymbol{R}_t$, the algorithm predicts with unit vector $\widehat{\boldsymbol{y}}_t = \boldsymbol{R}_t\boldsymbol{x}_t$ and suffers expected loss $1 - \boldsymbol{y}_t^\intercal \mathrm{E}\left[\, \widehat{\boldsymbol{y}}_t \,\right] = 1 - \boldsymbol{y}_t^\intercal \boldsymbol{W}_t \boldsymbol{x}_t$, where $\boldsymbol{y}_t$ is the "true" rotated vector revealed by Nature. Finally, the algorithm updates its parameter

$\boldsymbol{W}_t$ as:

$$\begin{aligned}
&\text{Descent step:} && \widehat{\boldsymbol{W}}_{t+1} = \operatorname{argmin}\Big(\eta\,\overbrace{(1-\boldsymbol{y}_t^\intercal \boldsymbol{W}\boldsymbol{x}_t)}^{\text{loss on example }(\boldsymbol{x}_t,\boldsymbol{y}_t)} + \overbrace{\frac{1}{2}\|\boldsymbol{W}-\boldsymbol{W}_t\|_F^2}^{\text{divergence}}\Big),\\
& && \phantom{\widehat{\boldsymbol{W}}_{t+1}} = \boldsymbol{W}_t + \eta \boldsymbol{y}_t \boldsymbol{x}_t^\intercal, \qquad\qquad (4.3)\\
&\text{Projection step:} && \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_{rot}} \|\boldsymbol{W}-\widehat{\boldsymbol{W}}_{t+1}\|_F^2,
\end{aligned}$$

where $\|\cdot\|_F$ is the Frobenius norm of matrices defined above. The descent step of the above GD update minimizes the trade-off between the parameter divergence $\|\boldsymbol{W}-\boldsymbol{W}_t\|_F^2$ and the loss on the current example $1-\boldsymbol{y}_t^\intercal\boldsymbol{W}\boldsymbol{x}_t$ *without* the constraint that $\boldsymbol{W}\in\boldsymbol{\mathcal{W}}_{rot}$. Then the projection step projects the intermediate parameter $\widehat{\boldsymbol{W}}_{t+1}$ back into the parameter set $\boldsymbol{\mathcal{W}}_{rot}$.

The GD algorithm is an example of the more general Mirror Descent algorithm family. A mirror descent algorithm is motivated by a Bregman divergence which measures the discrepancy between the new and the old parameters of an update. For example, the above GD algorithm is motivated by the Frobenius norm of matrices. Another important example of the Mirror Descent algorithms is the Matrix Exponential Gradient (MEG) algorithm [TRW05]. We do not consider the MEG algorithm in this chapter since as we shall see that for the problem of learning rotations online, the GD algorithm is already minimax optimal (up to a constant factor) for both the time dependent and the loss dependent bounds. Also, we do not know how to generalize the quantum relative entropy to the parameter set $\boldsymbol{\mathcal{W}}_{rot}$. Recall that for the problem of online PCA, the GD algorithm is outperformed by the MEG algorithm in loss dependent bounds. See a detailed comparison of the two algorithms for Online PCA in Chapter 2.

4.3.2 Previous Analysis of the GD Algorithm

Now we outline the previous analysis of the GD algorithm (4.3). This analysis gives an upper bound of $O(\sqrt{nT})$ on the regret of GD[HKW10a]. Let $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_{rot}$ be any comparator in the parameter set. The analysis first uses $\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2$ as a measure of progress towards the comparator $\boldsymbol{W}$:

$$\begin{aligned}
\overbrace{\boldsymbol{y}_t^\intercal \boldsymbol{W} \boldsymbol{x}_t - \boldsymbol{y}^\intercal \boldsymbol{W}_t \boldsymbol{x}_t}^{\text{Algorithm's regret in trial } t} &= \tfrac{1}{2\eta}\Big(\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_t + \eta \boldsymbol{y}_t \boldsymbol{x}_t^\intercal - \boldsymbol{W}\|_F^2\Big) + \frac{\eta}{2}\overbrace{\|\boldsymbol{y}_t \boldsymbol{x}_t^\intercal\|_F^2}^{=1} \\
&= \tfrac{1}{2\eta}\Big(\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}\|_F^2\Big) + \frac{\eta}{2} \qquad (4.4) \\
&\leq \tfrac{1}{2\eta}\Big(\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2\Big) + \frac{\eta}{2}. \qquad (4.5)
\end{aligned}$$

Here the first equality follows from the definition of Frobenius norm and the inequality follows from Pythagorean Theorem.

Next, we add up (4.5) over trial $t = 1 \dots T$ and the divergence term $\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2$ will telescope with each other. Therefore, the total regret of the algorithm is upper bounded by

$$\text{REG} \leq \tfrac{1}{2\eta}\Big(\|\boldsymbol{W}_1 - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_{T+1} - \boldsymbol{W}\|_F^2\Big) + \frac{\eta}{2}T. \qquad (4.6)$$

Now recalling that $\boldsymbol{W}_1, \boldsymbol{W} \in \boldsymbol{\mathcal{W}}_{rot}$, the RHS of (4.6) can be further upper bounded by $O(\frac{n}{\eta} + \eta T)$. Setting η properly gives the $O(\sqrt{nT})$ upper bound on the GD's regret.

Next we show that no matter how the learning rate η is tuned, one can never use (4.6) to prove a *loss dependent* bound for the GD algorithm. Consider a game with one dimensional examples $x_t = y_t = 1, t = 1 \dots T$ and one dimensional parameter set $\boldsymbol{\mathcal{W}}_{rot} = [-1, 1]$. Since the corresponding loss function is $1 - x_t w_t y_t = 1 - w_t$, the best comparator in $[-1, 1]$ suffers loss $\mathcal{L}_c = 0$. Hence, to achieve a loss dependent regret bound, the GD algorithm can suffer at most some constant amount of loss, where the constant does not depend on T. This implies the learning rate η of the algorithm is lower bounded by some constant that does not depend on T. However, such a learning rate makes the RHS of (4.6) grow linearly in T and therefore, it is not upper bounded by any function of the loss of the comparator (which is zero).

This observation suggests that the above application of Pythagorean Theorem (see (4.5)) is too crude. We present in Section 4.4 a refined application of Pythagorean Theorem and based on this improvement as well as adding a linear term $\operatorname{tr}(\boldsymbol{W}(\boldsymbol{W}_{t+1}^{\intercal} - \boldsymbol{W}_t^{\intercal}))$ to our algorithm's measure of progress, we derive a loss dependent bound for the GD algorithm.

Before finishing this section, we would like to discuss [SST10] which proves loss dependent regret bounds for the general Mirror Descent algorithm family. [SST10] claims that if the loss function of a learning problem satisfies a smoothness constant S, then the regret of a Mirror Descent algorithm for this problem is upper bounded by a function of the constant S and the loss of the best comparator $\mathcal{L}_c$ (see [SST10, Theorem 3]. However, their proof implicitly assumes that the loss function is non-negative on the entire vector space it lies in (see the proof of [SST10, Lemma 4.1]). Recall that in the problem of learning rotations online, the loss function is linear in the algorithm's parameter (see (4.2)), and therefore does not satisfy the “global” non-negativity assumption in [SST10]. Hence, this problem is not a special case of their results.

4.4 Loss Dependent Bound for the GD Algorithm

This section derives an upper bound of $O(\sqrt{n\mathcal{L}_c} + n)$ on the regret of the GD algorithm (4.3). We refine the previous application of Pythagorean Theorem by considering the difference between the intermediate parameter matrix $\widehat{\boldsymbol{W}}_{t+1}$ and the projected parameter matrix $\boldsymbol{W}_{t+1}$. We denote this difference matrix by $\boldsymbol{Z}_{t+1} = \boldsymbol{W}_{t+1} - \widehat{\boldsymbol{W}}_{t+1}$. Recall that $\boldsymbol{W}_{t+1}$ is the projection of $\widehat{\boldsymbol{W}}_{t+1}$ into the parameter set $\boldsymbol{\mathcal{W}}_{rot}$. Therefore, Pythagorean Theorem gives:

$$\|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}\|_F^2 \geq \|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2 + \|\boldsymbol{Z}_{t+1}\|_F^2, \tag{4.7}$$

where $\boldsymbol{W}$ is any fixed comparator in $\boldsymbol{\mathcal{W}}_{rot}$. However, as seen in (4.5), the previous application of Pythagorean Theorem ignores the $\|\boldsymbol{Z}_{t+1}\|_F^2$ term of the above inequality. Instead, it uses the relaxed version $\|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}\|_F^2 \geq \|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2$. In this section, we use the full version of (4.7) and by doing this, we prove a loss dependent regret bound for the GD algorithm. We start with the following useful observation about the difference matrix $\boldsymbol{Z}_t$.

Lemma 4.1. *Let $\widehat{\boldsymbol{W}}_{t+1}$ and $\boldsymbol{W}_{t+1}$ be the intermediate parameter matrix and the projected parameter matrix defined in* (4.3) *respectively. Then, the difference matrix $\boldsymbol{Z}_{t+1} = \boldsymbol{W}_{t+1} - \widehat{\boldsymbol{W}}_{t+1}$ has rank at most one.*

Proof. First consider $\widehat{\boldsymbol{W}}_{t+1} = \boldsymbol{W}_t + \eta \boldsymbol{y}\boldsymbol{x}^\intercal$, where $\boldsymbol{W}_t \in \boldsymbol{\mathcal{W}}_{rot}$ is the algorithm's parameter matrix in trial t and $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ is the corresponding data example. Since $\widehat{\boldsymbol{W}}_{t+1}$ is $\boldsymbol{W}_t$ plus a rank one perturbation $\eta \boldsymbol{y}\boldsymbol{x}^\intercal$, we have by [Tho76, Theorem 1] that $\sigma_2(\widehat{\boldsymbol{W}}_{t+1}) \leq$

$\sigma_1(\boldsymbol{W}_t)$ where $\sigma_2(\widehat{\boldsymbol{W}}_{t+1})$ is the second largest singular value of $\widehat{\boldsymbol{W}}_{t+1}$ and $\sigma_1(\boldsymbol{W}_t)$ is the largest singular value of $\boldsymbol{W}_t$. Now since $\boldsymbol{W}_t$ has all of its singular values in $[0,1]$, $\widehat{\boldsymbol{W}}_t$ has at most one singular value larger than 1.

Next, consider $\boldsymbol{W}_{t+1}$ which is the projection of $\widehat{\boldsymbol{W}}_{t+1}$ into $\boldsymbol{\mathcal{W}}_{rot}$. By [HKW10a, Lemma 2], $\boldsymbol{W}_{t+1}$ and $\widehat{\boldsymbol{W}}_{t+1}$ share the same singular vectors and their singular values satisfy the following: The i-th singular value of $\boldsymbol{W}_{t+1}$ is the i-th singular value of $\widehat{\boldsymbol{W}}_{t+1}$ capped at 1, i.e. for $1 \le i \le n$, $\sigma_i(\boldsymbol{W}_{t+1}) = \min\{\sigma_i(\widehat{\boldsymbol{W}}_{t+1}), 1\}$. Since we have just argued that $\widehat{\boldsymbol{W}}_{t+1}$ has at most one singular value larger than 1, $\widehat{\boldsymbol{W}}_{t+1}$ and $\boldsymbol{W}_{t+1}$ must share at least $n-1$ of their singular values. Therefore, their difference $\boldsymbol{Z}_t$ has rank at most one. □

Now we are ready to present the main result of this chapter, a loss dependent upper bound on the regret of the GD algorithm

Theorem 4.1. *Consider the problem of learning rotations online with n dimensional examples. If the loss of the best comparator is $\mathcal{L}_c$, then the GD algorithm with learning rate $\eta = \sqrt{\frac{2n}{\mathcal{L}_c}}$ suffers worst case regret at most $2\sqrt{2n\mathcal{L}_c} + 2n$.*

Proof. As in the previous analysis of the algorithm (see (4.4)), we start with the equality:

$$\overbrace{\boldsymbol{y}_t^\intercal \boldsymbol{W}\boldsymbol{x}_t - \boldsymbol{y}^\intercal \boldsymbol{W}_t\boldsymbol{x}_t}^{\text{Algorithm's regret in trial } t} = \tfrac{1}{2\eta}\Big(\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\widehat{\boldsymbol{W}}_{t+1} - \boldsymbol{W}\|_F^2 + \eta^2\Big)$$

Now plugging in the Inequality (4.7), we have the algorithm's regret in trial t upper bounded by

$$\frac{1}{2\eta}\Big(\|\boldsymbol{W}_t - \boldsymbol{W}\|_F^2 - \|\boldsymbol{W}_{t+1} - \boldsymbol{W}\|_F^2 + \eta^2 - \|\boldsymbol{Z}_{t+1}\|_F^2\Big)$$

Next we upper bound $\eta^2 - \|\boldsymbol{Z}_{t+1}\|_F^2$ as follows:

$$\begin{aligned}
\eta^2 - \|\boldsymbol{Z}_{t+1}\|_F^2 &\leq \eta^2 - \|\boldsymbol{Z}_{t+1}\|_F^2 + (\eta - \|\boldsymbol{Z}_{t+1}\|_F)^2 \\
&= 2\eta^2 - 2\eta\|\boldsymbol{Z}_{t+1}\|_F. \qquad (4.8)
\end{aligned}$$

We now argue that he term $\|\boldsymbol{Z}_{t+1}\|_F$ in the RHS of (4.8) is lower bounded by $\operatorname{tr}(\boldsymbol{W}\boldsymbol{Z}_{t+1}^\intercal)$ for any comparator $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_{rot}$:

$$\begin{aligned}
\|\boldsymbol{Z}_{t+1}\|_F &= \sigma_1(\boldsymbol{Z}_{t+1}) && (\boldsymbol{Z}_{t+1} \text{ has rank at most 1}) \\
&\geq \sigma_1(\boldsymbol{W})\sigma_1(\boldsymbol{Z}_{t+1}) && (\boldsymbol{W} \in \boldsymbol{\mathcal{W}}_{rot}) \\
&= \textstyle\sum_{i=1}^n \sigma_i(\boldsymbol{W})\sigma_i(\boldsymbol{Z}_{t+1}) && (\boldsymbol{Z}_{t+1} \text{ has rank at most 1}) \\
&\geq \operatorname{tr}(\boldsymbol{W}\boldsymbol{Z}_{t+1}^\intercal) && (\text{von Neumann's trace inequality}),
\end{aligned} \qquad (4.9)$$

where $\sigma_i(\boldsymbol{W})$ and $\sigma_i(\boldsymbol{Z}_{t+1})$ are the i-th singular values of $\boldsymbol{W}$ and $\boldsymbol{Z}_{t+1}$ respectively. Plugging (4.9) back to (4.8), we have

$$\begin{aligned}
\eta^2 - \|\boldsymbol{Z}_{t+1}\|_F^2 &\leq 2(\eta^2 - \eta\operatorname{tr}(\boldsymbol{W}\boldsymbol{Z}_{t+1}^\intercal)) \\
&= 2\eta^2 - 2\eta^2\operatorname{tr}(\boldsymbol{W}\boldsymbol{x}_t\boldsymbol{y}_t^\intercal) + 2\eta^2\operatorname{tr}(\boldsymbol{W}\boldsymbol{x}_t\boldsymbol{y}_t^\intercal) - 2\eta\operatorname{tr}(\boldsymbol{W}\boldsymbol{Z}_{t+1}^\intercal) \\
&= 2\eta^2(1 - \operatorname{tr}(\boldsymbol{y}_t^\intercal\boldsymbol{W}\boldsymbol{x}_t)) + 2\eta\operatorname{tr}(\boldsymbol{W}(\eta\boldsymbol{x}_t\boldsymbol{y}_t^\intercal - \boldsymbol{Z}_{t+1}^\intercal)) \\
&= 2\eta^2\underbrace{(1 - \operatorname{tr}(\boldsymbol{y}_t^\intercal\boldsymbol{W}\boldsymbol{x}_t))}_{\text{loss of comparator } \boldsymbol{W}} + 2\eta\underbrace{\operatorname{tr}(\boldsymbol{W}\boldsymbol{W}_{t+1}^\intercal - \boldsymbol{W}\boldsymbol{W}_t^\intercal)}_{\text{linear part in measure of progress}}. \qquad (4.10)
\end{aligned}$$

Finally, plugging (4.10) back to (4.4) and summing it over T trials gives the following upper bound on the total regret of the algorithm:

$$\mathrm{REG} \leq \frac{1}{2\eta}\|\boldsymbol{W}_1 - \boldsymbol{W}\|_F^2 + \eta\mathcal{L}_c + \operatorname{tr}(\boldsymbol{W}(\boldsymbol{W}_{T+1}^\intercal - \boldsymbol{W}_1^\intercal)).$$

Now, since all of $\boldsymbol{W}_1$, $\boldsymbol{W}_{T+1}$ and $\boldsymbol{W}$ are in $\boldsymbol{\mathcal{W}}_{rot}$, we have:

$$\|\boldsymbol{W}_1 - \boldsymbol{W}\|_F^2 \leq 4n \qquad \text{and} \qquad \operatorname{tr}(\boldsymbol{W}(\boldsymbol{W}_{T+1}^\intercal - \boldsymbol{W}_1^\intercal)) \leq 2n.$$

Plugging these inequalities back, we have the following loss dependent regret bound:

$$\text{REG} \leq \frac{2n}{\eta} + \eta\mathcal{L}_c + 2n \overset{\eta=\sqrt{\frac{2n}{\mathcal{L}_c}}}{=} 2\sqrt{2n\mathcal{L}_c} + 2n.$$

□

Note that the above loss dependent regret bound is obtained by tuning the learning rate as a function of the loss of the best comparator $\mathcal{L}_c$. In cases that $\mathcal{L}_c$ is not known beforehand, one could first estimate $\mathcal{L}_c$ with an arbitrary constant and start the algorithm with the learning rate tuned accordingly. Later on, whenever the estimate is reached by a comparator, one should double the previous estimate of $\mathcal{L}_c$ and restart the algorithm with the corresponding re-tuned learning rate. This method of tuning the learning rate is known as the "doubling trick" and was shown to achieve regret bounds in a constant factor of the bounds of the underlying algorithm [CBLW96].

4.5 Conclusion

In this chapter we prove an upper bound of $O(\sqrt{n\mathcal{L}_c} + n)$ on the regret of the GD algorithm for the problem of learning rotations online. The upper bound is obtained by introducing two new techniques: First a refined application of Pythagorean Theorem (see (4.7)) and second, adding a linear term $\text{tr}(\boldsymbol{W}(\boldsymbol{W}_{t+1}^\intercal - \boldsymbol{W}_t^\intercal))$ to our algorithm's measure of progress (see (4.10)). This upper bound is minimax optimal (up to constant factor) since a matching lower bound is previously given in [HKW15].

Now we conclude the chapter with an open problem regarding a modified version of the GD algorithm presented in this chapter. Recall that because of the projection step of Update (4.3), the parameter matrix $\boldsymbol{W}_t$ of the GD algorithm is always in $\boldsymbol{\mathcal{W}}_{rot}$. However, maintaining the parameter always in $\boldsymbol{\mathcal{W}}_{rot}$ is not necessary for the problem of learning rotations online since the algorithm is only required to predict (probabilistically or deterministically) with a unit vector $\widehat{\boldsymbol{y}}_t$ (see the protocol in Section 4.2) in each trial. Therefore, [HKW15] proposed a "Lazy Projection GD" algorithm which only projects its parameter $\boldsymbol{W}_t$ when it cannot sample a random unit vector $\widehat{\boldsymbol{y}}_t$ satisfying the condition $\mathrm{E}\,[\,\widehat{\boldsymbol{y}}_t\,] = \boldsymbol{W}_t\boldsymbol{x}_t$. This new algorithm is more efficient than the GD algorithm (4.3) since it "projects as little as necessary", and at the same time achieves the same time depend regret bound as the GD algorithm. Our open problem concerns the following question: can we use the new proof techniques presented in this chapter to prove a loss dependent upper bound on the regret of the "Lazy Projection GD" algorithm?

Bibliography

[AABR09] Jacob Abernethy, Alekh Agarwal, Peter L. Bartlett, and Alexander Rakhlin. A stochastic view of optimal regret through minimax duality. In *COLT*, pages 56–64, 2009.

[AB10] Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *Journal of Machine Learning Research*, 11:2785–2836, 2010.

[ACS13] Raman Arora, Andrew Cotter, and Nati Srebro. Stochastic optimization of PCA with capped MSG. In *NIPS*, pages 1815–1823, 2013.

[Aro09] Raman Arora. On learning rotations. In *NIPS*, pages 55–63, 2009.

[AW01] Katy S. Azoury and Manfred K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

[AWY08] Jacob Abernethy, Manfred K. Warmuth, and Joel Yellin. When random play is optimal against an adversary. In *COLT*, pages 437–446, 2008.

[Ber11] Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas (Second Edition)*. Princeton reference. Princeton University Press, 2011.

[CBFH+97] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

[CBL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[CBLW96] Nicolò Cesa-Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Trans. Neural Netw. Learning Syst.*, 7(3):604–619, 1996.

[dRvEGK14] Steven de Rooij, Tim van Erven, Peter D. Grünwald, and Wouter M. Koolen. Follow the leader if you can, hedge if you must. *Journal of Machine Learning Research*, 15:1281–1316, 2014.

[DS02] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley series in statistics. Addison-Wesley, 2002.

[FS95] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pages 23–37, 1995.

[GHM15] Dan Garber, Elad Hazan, and Tengyu Ma. Online learning of eigenvectors. In *ICML*, pages 560–568, 2015.

[HKW10a] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Corrigendum to "learning rotations with little regret". Corrigendum, 2010.

[HKW10b] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Learning rotations with little regret. In *COLT*, pages 144–154, 2010.

[HKW10c] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. On-line variance minimization in o(n2) per trial? In *COLT*, pages 314–315, 2010.

[HKW15] Elad Hazan, Satyen Kale, and Manfred K. Warmuth. Learning rotations with little regret. to appear in Machine Learning Journal, 2015.

[HW01] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[HW09] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.

[KNW13] Wouter M. Koolen, Jiazhong Nie, and Manfred K. Warmuth. Learning a set of directions. In *COLT*, pages 851–866, 2013.

[Koo11] Wouter M. Koolen. *Combining strategies efficiently: high-quality decisions from conflicting advice.* PhD thesis, Institute of Logic, Language and Computation (ILLC), University of Amsterdam, 2011.

[KW97] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.

[KW07] Dima Kuzmin and Manfred K. Warmuth. Online kernel PCA with entropic matrix updates. In *ICML*, pages 465–472, 2007.

[KW11] Wojciech Kotlowski and Manfred K. Warmuth. Minimax algorithm for learning rotations. In *COLT*, pages 821–824, 2011.

[KW15] Wojciech Kotlowski and Manfred K. Warmuth. PCA with Guassian perturbation. Private communication, 2015.

[KWK10] Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In *COLT*, pages 93–105, 2010.

[Lit87] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[McM13] Brendan McMahon. Minimax optimal algorithms for unconstrained linear optimization. Unpublished manuscript arXiv:1302.2176v1, February 2013.

[Nie14] Jiazhong Nie. Loss dependent regret bounds for online gradient descent. Unpublished manuscript, 2014.

[NKW13] Jiazhong Nie, Wojciech Kotlowski, and Manfred K. Warmuth. Online PCA with optimal regrets. In *ALT*, pages 98–112, 2013.

[NY78] Arkadi Nemirovski and D Yudin. On cesaros convergence of the gradient descent method for finding saddle points of convex-concave functions. *Doklady Akademii Nauk*, 4(249):249, 1978.

[SS07] Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

[SST10] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. In *NIPS*, pages 2199–2207, 2010.

[SST11] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. On the universality of online mirror descent. In *NIPS*, pages 2645–2653, 2011.

[ST10] Karthik Sridharan and Ambuj Tewari. Convex games in banach spaces. In *COLT*, pages 1–13, 2010.

[SW08] Adam M. Smith and Manfred K. Warmuth. Learning rotations. In *COLT*, page 517, 2008.

[Tho76] R.C. Thompson. The behavior of eigenvalues and singular values under perturbations of restricted rank. *Linear Algebra and its Applications*, 13(12):69 – 78, 1976.

[TRW05] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponential gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

[vEGKdR11] Tim van Erven, Peter Grünwald, Wouter M. Koolen, and Steven de Rooij. Adaptive hedge. In *NIPS*, pages 1656–1664, 2011.

[Vov90] Vladimir Vovk. Aggregating strategies. In *COLT*, pages 371–386, 1990.

[Wah65] Grace Wahba. Problem 651: A least squares estimate of spacecraft attitude. *SAIM Review*, 7(3):409, 1965.

[WK08] Manfred K. Warmuth and Dima Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, October 2008.

[WV05] Manfred K. Warmuth and S. V. N. Vishwanathan. Leaving the span. In *COLT*, pages 366–381, 2005.

[Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.