Diss. ETH No. 20013

Sparse Convex Optimization Methods for Machine Learning

A dissertation submitted to
ETH Zürich

for the degree of
Doctor of Sciences

presented by
Martin Jaggi
Dipl. Math. ETH
born Mai 23, 1982
citizen of Lenk BE, Switzerland

accepted on the recommendation of
Prof. Dr. Emo Welzl, examiner
Dr. Bernd Gärtner, co-examiner
Dr. Elad Hazan, co-examiner
Prof. Dr. Joachim Giesen, co-examiner
Prof. Dr. Joachim M. Buhmann, co-examiner

2011

Abstract

Convex optimization is at the core of many of today's analysis tools for large datasets, and in particular machine learning methods. In this thesis we will study the general setting of optimizing (minimizing) a convex function over a compact convex domain.

In the first part of this thesis, we study a simple iterative approximation algorithm for that class of optimization problems, based on the classical method by Frank & Wolfe. The algorithm only relies on supporting hyperplanes to the function that we need to optimize. In each iteration, we move slightly towards a point which (approximately) minimizes the linear function given by the supporting hyperplane at the current point, where the minimum is taken over the original optimization domain. In contrast to gradient-descent-type methods, this algorithm does not need any projection steps in order to stay inside the optimization domain.

Our framework generalizes the sparse greedy algorithm of Frank & Wolfe and its recent primal-dual analysis by Clarkson (and the low-rank SDP approach by Hazan) to arbitrary compact convex domains. Analogously, we give a convergence proof guaranteeing ε-small error — which in our context is the duality gap — after $O(\frac{1}{\varepsilon})$ iterations.

This method allows us to understand the *sparsity* of approximate solutions for any ℓ_1-regularized convex optimization problem (and for optimization over the simplex), expressed as a function of the approximation quality. Here we obtain matching upper and lower bounds of $\Theta\big(\frac{1}{\varepsilon}\big)$ for the sparsity. The same bounds apply to low-rank semidefinite optimization with bounded trace, showing that rank $O\big(\frac{1}{\varepsilon}\big)$ is best possible here as well.

For some classes of geometric optimization problems, our algorithm has a simple geometric interpretation, which is also known as the coreset concept. Here we will study linear classifiers such as support vector machines (SVM) and perceptrons, as well as general distance computations between convex hulls (or polytopes). Here the framework will allow us to understand the sparsity of SVM solutions, here being the number of support vectors, in terms of the required approximation quality.

For matrix optimization problems, we show that our proposed first-order method also applies to convex optimization over bounded nuclear norm or max-norm. This class of optimization problems has prominent applications in several areas, such as low-rank recovery, matrix completion and recommender systems. We demonstrate the practical efficiency and scalability of our algorithm for large matrix problems, as e.g. the Netflix dataset. For general convex optimization over bounded matrix max-norm, our algorithm is the first with a convergence guarantee, to the best of our knowledge.

In the last part of this thesis, we will consider convex optimization problems which are parameterized by a single additional parameter, as for example a time or regularization parameter. In several applications, one is interested in the entire path of the solution, as the parameter changes.

Here, a continuity argument together with our simple concept of optimization duality for compact domains will allow us to obtain solution paths (of some guaranteed approximation quality) for such problems. We show that piecewise constant solutions can be obtained, and that $O\left(\frac{1}{\varepsilon}\right)$ many such solutions are enough in order to guarantee approximation quality ε along the entire path, independent of the dimension of the problem. Our method allows us to compute solution paths for e.g. SVMs, ℓ_1- or ℓ_∞-regularized problems, nuclear norm regularized problems such as matrix completion, and robust principal component analysis.

Zusammenfassung

Konvexe Optimierung steht im Mittelpunkt von vielen der heute verfügbaren Analyse-Methoden für grosse Datenmengen, insbesondere von Methoden des maschinellen Lernens. In dieser Arbeit untersuchen wir das allgemeine Problem der Optimierung (bzw Minimierung) einer konvexen Funktion über einem kompakten konvexen Gebiet.

Im ersten Teil dieser Arbeit untersuchen wir einen einfachen iterativen Algorithmus für diese Klasse von Optimierungsproblemen, basierend auf der klassischen Methode von Frank & Wolfe. Der Algorithmus benützt stützende Hyperebenen der zu optimierenden Funktion. In jeder Iteration bewegen wir uns etwas in die Richtung eines Punktes, welcher die lineare Funktion gegeben durch eine momentane stützende Ebene (annähernd) minimiert, wobei das Minimum über das ursprüngliche Optimierungs-Gebiet genommen wird. Im Gegensatz zu Gradienten-Abstiegs-Methoden benötigt dieser Algorithmus keine Projektions-Schritte, um innerhalb des zulässigen Gebiets zu bleiben.

Unser Ansatz verallgemeinert den Sparse-Greedy-Algorithmus von Frank & Wolfe und seine neuere primal-duale Analysis durch Clarkson (sowie die semi-definite Optimierungsmethode mittels kleinem Rang von Hazan) auf beliebige kompakte konvexe Gebiete. Unser Konvergenz-Beweis garantiert ε-kleinen Fehler (Dualitäts-Lücke) nach $O\left(\frac{1}{\varepsilon}\right)$ Iterationen.

Die Methode ermöglicht es uns die Dünnbesetztheit (Sparsity) von approximativen Lösungen für ℓ_1-regularisierte konvexe Optimierungs-Probleme (und für die Optimierung über dem Simplex), als Funktion der Approximations-Güte zu verstehen. Hier erhalten wir passende obere und untere Schranken von $\Theta\left(\frac{1}{\varepsilon}\right)$ für die Dünnbesetztheit. Die gleichen Grenzen gelten für semi-definite Optimierung unter beschränkter Spur, in Bezug auf den Matrix-Rang. Hier zeigen wir dass Rang $O\left(\frac{1}{\varepsilon}\right)$ ebenfalls optimal ist.

Für einige Klassen von geometrischen Optimierungsproblemen hat unser Algorithmus eine einfache geometrische Interpretation, die auch als das Konzept der *Coresets* bekannt ist. Hier untersuchen wir lineare Klassifikatoren wie zum Beispiel Support Vector Machines (SVM), sowie allgemeine Entfernungsberechnungen zwischen konvexen Hüllen (oder Poly-

topen). Hier ermöglicht unser Framework die Dünnbesetztheit der SVM-Lösungen zu verstehen, was in diesem Zusammenhang die Anzahl der Support-Vektoren bedeutet, in Abhängigkeit der erforderlichen Approximations-Güte.

Unsere Methode ist ebenfalls direkt anwendbar für Matrix-Optimierungs-Probleme, insbesondere für konvexe Optimierung unter beschränkter Spur-Norm oder Max-Norm. Diese Klasse von Optimierungsproblemen hat prominente Anwendungen in verschiedenen Bereichen, wie z.B. die Rekonstruktion von Matrizen von kleinem Rang, oder die Komplettierung von Matrizen zum Beispiel bei Empfehlungs-Systemen. Wir demonstrieren die praktische Effizienz und Skalierbarkeit unseres Algorithmus für grosse Matrix-Probleme, wie z.B. dem Netflix-Datenset. Für allgemeine konvexe Optimierung unter beschränkter Matrix Max-Norm ist unser Algorithmus nach unserem Wissen die erste Methode mit einer Konvergenz-Garantie.

Im letzten Teil dieser Arbeit betrachten wir konvexe Optimierungsprobleme die von einem zusätzlichen Parameter abhängen, wie z.B. einem Zeit- oder Regularisierungs-Parameter. In einigen Anwendungen ist man am gesamten Pfad der Lösung interessiert, in Abhängigkeit des zusätzlichen Parameters.

Mittels eines Stetigkeits-Arguments, zusammen mit unserem einfachen alternativen Konzept der Dualität für Optimierung über kompaktem Gebiet, erhalten wir die vollständigen Lösungspfade (für eine garantierte Approximations-Güte) für solche Probleme. Wir zeigen, dass stückweise konstante Lösungen existieren, und dass $O\left(\frac{1}{\varepsilon}\right)$ viele solcher Lösungen genügen, um eine Approximations-Güte von ε entlang dem gesamten Pfad zu garantieren, unabhängig von der Dimension des Problems. Unsere Methode erlaubt das Berechnen von Lösungspfaden für z.B. SVMs, ℓ_1- oder ℓ_∞-regularisierte Probleme, Spur-Norm-regularisierte Probleme wie Matrix-Komplettierung, sowie robuste Hauptkomponentenanalyse (PCA).

Acknowledgments

I would like to express my gratitude to my advisor Bernd Gärtner. Without his continuous support, openness and patience, this thesis would never have been written. I am also very grateful to Emo Welzl for letting me be part of his research group, and for providing a working environment that could not possibly be any better.

Furthermore I would like to thank Joachim Giesen for sparking my interest in support vector machines, for the collaboration on path algorithms, for inviting me to Saarbrücken and Jena, and for co-refereeing this thesis.

Many thanks to Elad Hazan and Joachim Buhmann for agreeing to be co-examiners and for providing many helpful comments. For the research collaboration, I would like to thank Soeren Laue and Marek Sulovský.

All members of the Gremo team made the time at ETH unforgettable: Andrea Francke, Andrea Salow, Anna Gundert, Heidi Gebauer, Michael Hoffmann, Robin Moser, Sebastian Stich, Timon Hertli, Uli Wagner, Vincent Kusters, Yves Brise, as well as all former members whom I had the pleasure to meet: Tobias Christ, Gabriel Nivasch, Marek Sulovský, Dominik Scheder, Floris Tschurr, Patrick Traxler, Andreas Razen, Philipp Zumstein, Tibor Szabó, Robert Berke, Eva Schuberth, and Leo Rüst.

I am indebted to Heidi, Tobias and Sebastian for proof-reading parts of this thesis, and to Robert Carnecky for the 3d visualization of convex minimization. For helpful and inspiring discussions and support, I would like to thank Andreas Krause, Arkadi Nemirovski, Christian Lorenz Müller, Christoph Krautz, Clément Maria, Florian Jug, Gabriel Katz, Mark Cieliebak, Michael Bürgisser, Michel Baes, Michel Verlinden, Pankaj Agarwal, Simon Meier and Yves Ineichen.

For gently introducing me to some real-world machine learning applications, I am very grateful to Daniel Mahler, Hartmut Maennel and Lars Engebretsen from Google, and to Francois Rüf from Netbreeze.

I want to thank all friends, WG-gspänli and climbing partners, to ETH mensa for roughly a decade of calories, and to Minimum and Milandia for all the nicely arranged colorful plastic holds.

Last but not least I want to thank my family for always being there, my parents Theres and Walter, my brother Thomas; and Nadja for her love and understanding.

Contents

1 Introduction

With the immense growth of available digital data, algorithmic analysis techniques for large datasets have become increasingly important. The efficiency and scalability of many such techniques, in particular from the area of machine learning, is often limited by the currently available methods to solve the underlying convex optimization problems.

Machine Learning. In an informal sense, *machine learning* is the task of building a model for some quantity (or function) that we would like to predict, or in other words, learn. The model is usually built from a set of "training" data for which the corresponding quantity of interest is known. Later, the obtained model is used to predict on new or unknown data, where we will then evaluate the performance of the obtained model. So far, this task description strikingly resembles classical *regression*, which is not a coincidence.

Concrete practical examples of such machine learning questions include classifying handwritten characters, reconstructing radio signals from very noisy sources, detecting a disease from MRI brain images, recommending movies or other products depending on personal ratings given to other items, ranking websites in a search engine based on their text content, modeling the terrain from the data from the sensor of an autonomous car, and predicting climate parameters or stock prices based on historical data, as well as many other applications.

1.1. Convex Optimization

The first part of this thesis addresses the general topic of convex optimization over bounded domains. Such convex optimization problems have applications in a very large variety of different areas, such as signal processing, computational biology, control theory, combinatorial optimization, communications and networking, statistics, finance, data mining, and — last but not least — in machine learning.

In Chapter 2, we consider a simple first-order[1] method for minimizing a convex objective function over a bounded convex domain. The algorithm is a generalization of an existing method originally proposed by Frank & Wolfe [FW56]. In contrast to gradient descent, our method does not need any projection steps in order to stay inside the optimization domain. Instead, in each iteration, we solve a linear optimization problem over the same domain. More precisely, the algorithm in each iteration moves towards an approximate minimizer of a linear supporting hyperplane at the current point, where the minimum is taken over the original optimization domain. Our analysis generalizes the convergence analysis of [Cla10] for optimization over the unit simplex to the more general setting of arbitrary convex optimization domains.

Furthermore, we suggest to use a very simple alternative duality concept for optimizing over bounded domain, which will allow us to efficiently compute certificates for the approximation quality of any candidate solution. The concept will later also allow us to track approximate solution paths to convex optimization problems that change over time, i.e. are parameterized by a single additional parameter. This solution path idea will be explained in the last two Chapters 6 and 7.

Consequences and Applications. One of the main interesting consequences of the described optimization approach is that the obtained approximate solutions during the run of the algorithm always have a "sparse" representation. This property will be studied for several applications in this thesis, and is also a very desirable property in many applications, in particular for regularized optimization methods.

[1] Here the term "first-order" refers to optimization methods that only use knowledge obtained from the first derivatives of the objective function (e.g. the gradient), or in other words do not use any information from second or higher derivatives.

1.2. Sparsity and Generalizations Thereof

Suppose we are given a convex optimization problem, namely that we need to minimize a convex function over $\mathbb{R}^n$. In many practical situations, we would not only like to have any (approximate) solution, but we want a solution which has the additional property that it is also sparse, or in other words contains just few non-zero coordinates, say just $k < n$ many of these.

Unfortunately, the additional requirement of sparsity immediately turns the original convex problem (for which efficient algorithms are known) into a very hard combinatorial problem. Solving it would require us to try all possible patterns of $\binom{n}{k}$ non-zeros in a brute-force way, requiring time exponential in k. Also note that the set of sparse vectors (at most k many non-zero coordinates) is not a convex set anymore.

Convex Relaxations by Using the ℓ_1-Norm. To allow for more efficient convex optimization approaches, we need the domain to be a convex set. In the literature, a widely successful approach has been to replace the requirement of sparsity by optimizing over the ℓ_1-ball instead. The ℓ_1-ball is the set of all vectors in $\mathbb{R}^n$, for which the absolute values of the coordinates sum up to at most one (this quantity is known as the ℓ_1-norm). Also, it is not hard to see that the ℓ_1-ball is exactly the convex hull of the "sparsest possible" vectors, namely the standard basis vectors (and their negatives), i.e. the vectors with only one non-zero coordinate.

A Geometric Intuition. As one attempt to explain the usefulness of the ℓ_1-ball as a domain for obtaining sparse solutions, a simple geometric fact comes into play. It has been known for a very long time that for optimizing any *linear* function $c^T x$ over the ℓ_1-ball, there will always be a vertex of the ball where the optimum value is obtained. In other words there will always be an optimal solution of the best possible sparsity.

This fact has been widely cited as the motivation to apply the very same ℓ_1-relaxation trick as described above, also for the case of *non-linear* convex optimization problems over $\mathbb{R}^n$, when sparse solutions are desired. Intuitively, the hope is that also for these more general functions, optimizing over the ℓ_1-ball would "often" return sparse solutions. More formally, one can show that the ℓ_1-norm is the convex function that best approximates the sparsity.

Sparsity as a Function of the Approximation Quality. In this thesis, we can quantify this sparsity more precisely, by studying its dependency on the desired approximation quality. We say that a point is an ε-approximate solution to the optimization problem if its value is at least ε-close to the unknown true optimal value.

We show that if an approximation quality of $\varepsilon > 0$ is required, then the very simple greedy algorithm that we described above will always provide sparse solutions of only $O\left(\frac{1}{\varepsilon}\right)$ non-zero entries, if the optimization domain is the ℓ_1-ball. The analysis will follow analogously to the result of [Cla10] for the unit simplex. Also, we provide an asymptotically matching lower bound, that for some natural convex function, at least $\Omega\left(\frac{1}{\varepsilon}\right)$ non-zero entries are strictly necessary for any ε-approximate solution.

This means we can characterize the sparsity as a function of the approximation quality. In other words, we have translated the fact that every linear optimization problem has a 1-sparse solution, to arbitrary non-linear convex optimization problems, where we obtain $O\left(\frac{1}{\varepsilon}\right)$-sparse approximate solutions. This is particularly remarkable because we are not making any assumptions about the sparsity of the true optimal solution, which might be completely dense for example. Our approach is not limited to the domain being the ℓ_1-ball or the unit simplex, but in fact works on every bounded convex set, which is useful as follows:

Generalizing Sparsity to Description Complexity. Our generalized sparse greedy optimization approach for arbitrary bounded convex domains will also allow us to slightly generalize the concept of sparse solutions. For the convex hull of arbitrary elements of some vector space, it is easy to see that every linear function obtains its minimum at a vertex, or in other words one of the original "atomic" elements.

This implies that the algorithm will only ever use elements from the initial "atoms" as the step direction in each iteration. In other words it will obtain an ε-approximate solution which is a convex combination (therefore a weighted sum) of just $O\left(\frac{1}{\varepsilon}\right)$ many of the original atomic elements. Alternatively we say that each obtained approximate solution has a low *description complexity*, in terms of the elements defining the convex hull, which is our optimization domain. This idea is also closely related to the recent concept of *structured sparsity*, which refers to the same idea of measuring the complexity of an element in terms of e.g. how many atomic elements of some defined structure are needed to represent the element of consideration.

There are many interesting applications building upon such convex com-

binations of various objects. One prominent example is given by the symmetric rank-1 matrices of unit trace, whose convex hull is known to be the set of all positive semidefinite matrices of bounded trace. In this example, we will recover the approach of Hazan [Haz08] for semidefinite optimization, and obtain solutions of rank $O\left(\frac{1}{\varepsilon}\right)$. This particular class of optimization problems has many prominent examples e.g. in dimensionality reduction, low-rank recovery, or recommender systems, as we will discuss in more details in Chapter 4, where we will show how the algorithm can be used to solve arbitrary nuclear norm or max-norm constrained optimization problems over matrices.

Coresets. The *coreset* approach originally proposed in computational geometry is another name for the same concept of sparse solutions for the case of point set problems. Clarkson [Cla10] has already translated this concept to convex optimization over the unit simplex. Here we extend this concept also to convex optimization over bounded domains in arbitrary vector spaces.

1.3. Regularization Methods

In many real world applications, the available data is not perfect, but contains small errors, additional noise, or other kinds of corruptions. This issue has only increased with the advancement of technology and the growing amount of available data of all kinds, and proposes a significant challenge for all methods to analyze and/or further process such data. Regularization is widely used as a way to make existing techniques robust to noise or corruptions in the data, and has seen many successful applications. Stephen Boyd and Emmanuel Candes, two of the leading researchers in this area, are referring to ℓ_1-regularized optimization methods as the "least-squares approach" of the 21st century, because of the simplicity and wide applicability of such techniques.

One is probably tempted to think that in order to successfully attack machine learning tasks (such as regression over noisy data), much more complex models would have to be used than for classical regression, say. However, the idea of regularization is in a sense exactly the opposite, namely that a model of *small complexity* should be used instead.

An Example. Here we will briefly introduce the regularization idea for a concrete example, for two different methods of data analysis and machine

learning. Suppose that we are given a set of m holiday pictures. We can represent each such digital image as a point $x_i \in \mathbb{R}^n$, being the large vector consisting of the numerical values (or colors) of each pixel of the image, stored one after each other. In other words $x_1, \ldots, x_m \in \mathbb{R}^n$ are the points representing the m pictures, and n is the number of pixels of our digital camera.

1.3.1. Least Squares Regression

We assume that for each holiday image x_i, $1 \leq i \leq m$, we are also given a numerical value y_i describing how much we like that particular picture (say on a scale from 1 to 5). We would like to "learn" or "fit" a linear function to this data, so that we can later apply this function to a set of arbitrary images (e.g. from Google image search), with the hope to automatically identify the best suitable future holiday destination for our taste.

Formally, we now search for a linear function (described by a vector $\omega \in \mathbb{R}^n$) that minimizes the squared error

$$L(\omega) := \sum_{i=1}^{m} (x_i^T \omega - y_i)^2 = \|A\omega - y\|_2^2 \ .$$

Here $A \in \mathbb{R}^{m \times n}$ is the matrix that contains all our m datapoints x_i^T as its rows, and $y \in \mathbb{R}^m$ is the vector consisting of the values y_i that we would like to approximate.

Now the convex optimization problem $\min_{\omega \in \mathbb{R}^n} L(\omega)$ can definitely be solved efficiently. However, most solutions will very likely be totally meaningless, because the number of variables n is much higher than the number of data points (m) that we have. In other words, since the system is extremely under-determined, we will definitely suffer from the "curse of too many parameters". Changing only a single pixel in one of our holiday pictures will likely result in a totally different solution vector ω.

Here, regularization comes into play. The idea is that if there is any meaningful solution (or model) ω to our problem at all, then that solution should be of small complexity. The same paradigm is known as *Occam's razor*, that if there are several explanations for some phenomenon, then the simplest one should be preferred. In our case, we will simply measure the complexity of each model ω by its ℓ_1-norm. Instead of solving the original unconstrained optimization version $\min_{\omega \in \mathbb{R}^n} L(\omega)$, we will now restrict to ω being of small complexity, by constraining on a suitably scaled version

of the ℓ_1-ball, or formally

$$\min_{\substack{\omega \in \mathbb{R}^n \\ \|\omega\|_1 \leq t}} L(\omega) \ . \tag{1.1}$$

This formulation is called the ℓ_1-regularized version of the original (unconstrained) least squares problem. Solutions to this problem tend to be much more meaningful for almost all applications where the available data is subject to noise, or systems that have a very large number of free parameters. The question of what "suitable" means for the so called regularization parameter t will be studied in Chapters 6 and 7 in more details.

Applying our simple greedy convex optimization procedure to ℓ_1-regularized least squares, we will obtain ε-approximate solutions ω which are additionally sparse, i.e. have $O\left(\frac{1}{\varepsilon}\right)$ non-zero coordinates, which will be explained in Chapter 2.

1.3.2. Two equivalent Variants of Regularization

As a small technical remark, we note that there are two equivalent variants of adding regularization to a given original optimization problem $\min_{\omega \in \mathbb{R}^n} L(\omega)$. Here we assume $L(\omega)$ is an arbitrary convex objective function that we would like to optimize. If the model complexity of any ω is measured by a convex function $R(\omega)$, then the following two formulations are equivalent. In machine learning terms, the function $R(.)$ is usually called the *regularization* term, while $L(.)$ is called the *loss* function. The constrained variant of the regularized problem is given by

$$\min_{\substack{\omega \in \mathbb{R}^n \\ R(\omega) \leq t}} L(\omega) \ . \tag{1.2}$$

The analogous trade-off version, for some trade-off parameter $\lambda > 0$ is given by

$$\min_{\omega \in \mathbb{R}^n} L(\omega) + \lambda R(\omega) \ . \tag{1.3}$$

The two parameters t and λ are in a direct correspondence. For choice of $\lambda > 0$ together with some optimal solution $\omega_{(\lambda)}$ to the trade-off variant (1.3), setting $t_{(\lambda)} := R(\omega_{(\lambda)})$ in the constrained variant (1.2) will result in the same objective value $L(\omega_{(\lambda)})$ for the optimum of (1.2). On the other hand, formulation (1.3) is also known as the *Lagrangian* version of (1.2). For any fixed t, there is always some choice of $\lambda = \lambda_{(t)}$ such that the optimum of (1.3) has the same loss value $L(\omega_{(t)})$ as an optimal solution $\omega_{(t)}$ to (1.2).

This direct correspondence between the solutions to the two problem variants is the reason that both variants are used interchangeably in most applications of regularization methods. The following short section on linear classifiers is an example where it is more common to work with the trade-off variant (1.3).

1.3.3. Linear Classifiers and Support Vector Machines

As a second example, we would like to present linear classifiers. In the same setting as for the holiday pictures $x_1, \dots, x_m \in \mathbb{R}^n$ as given in the above setting, we now consider classification instead of regression. This means that each of the pictures $x_i \in \mathbb{R}^n$ comes together with a label $y_i \in \pm 1$. We think of this label y_i being $+1$ if the picture x_i contains the face of some person, and $y_i = -1$ otherwise. We would like to find a linear classifier for such images, or in other words we again search for a linear function $\omega^T x$ determined by $\omega \in \mathbb{R}^n$, such that hopefully $\omega^T x_i > 0$ for pictures x_i containing faces, and $\omega^T x_i < 0$ otherwise. This approach is widely known as the linear *Perceptron*.

Regularization and Support Vector Machines. It turns out that instead of just solving the above constraints for some optimal ω, a much better and more robust classifier $\omega^T x$ is obtained by adding a regularization on the complexity of ω. Here, by a geometric interpretation that we will study in more details in Chapter 5, the effect of the regularization will be that one obtains a linear function that separates the datapoints not just in some way, but with the best possible margin of separation. This type of large margin linear classifier is called the support vector machine (SVM).

The standard optimization problem for the linear SVM is given by

$$\min_{\omega \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \left(1 - y_i \omega^T x_i\right)^+ + \lambda \left\|\omega\right\|_2^2$$

Here the notation $(.)^+$ returns the positive part of its argument, meaning that $(s)^+$ is equal to s for $s \geq 0$, and zero otherwise. The interpretation of the loss-term $L(\omega) := \frac{1}{m} \sum_{i=1}^{m} \left(1 - y_i \omega^T x_i\right)^+$ is that all points which are on their correct side of the hyperplane (given by $\omega^T x = 0$) and are at least some distance away from the plane, i.e. $y_i \omega^T x_i > 1$, will not contribute to the loss $L(\omega)$. All other points that are too close to the hyperplane — or do lie on the wrong side corresponding to the opposite label — will be punished by the term $L(\omega)$.

For linear classifiers, the effect of the above regularization term $\|\omega\|_2$ on the resulting solution ω is well studied, and is equivalent to choosing the separating hyperplane ω of the largest possible separation margin. It can be shown this particular hyperplane indeed results in the best expected prediction accuracy on unknown "test" data, where we only have to assume that the test data is drawn from the same distribution as the "training" points (from which we have obtained the hyperplane). Results of this type are often proven by using the concept of Rademacher complexity, see e.g. [STC04].

Also in this completely different optimization approach as compared to our first example of ℓ_1-regularized regression, it turns out that the regularization idea (here by constraining $\|\omega\|_2$ as compared to $\|\omega\|_1$), is an extremely powerful tool in practice. For classification tasks, SVMs have become the basic tool of choice in most applications.

The idea of adding regularization to linear classifiers — and thereby also enabling its application to "noisy" point sets that are not perfectly linearly separable — was originally proposed by [CV95], and was awarded the ACM Paris Kanellakis Theory and Practice Award in 2008.

Sparsity and the Number of Support Vectors. Using optimization duality, it is not hard to see that every meaningful hyperplane for an SVM can be expressed as a convex combination of the original datapoints. Applying our simple greedy convex optimization procedure to SVMs, we will obtain coresets (small sets of support vectors) of size $O\left(\frac{1}{\varepsilon}\right)$, see Chapter 5. We also provide examples showing that this is best possible up to a constant factor. This means we can understand the number of support vectors as a function of the approximation quality ε.

1.4. Geometric Problems

In Chapter 5, we will study the geometric interpretation of the general convex optimization technique described above, for the case of computing distances between convex hulls (or polytopes). This problem also appears naturally in collision detection for geometric objects in physical simulations or computer games.

However, here we will focus mostly on applications to linear classifiers for machine learning applications, and in particular support vector machines (SVM) and perceptrons. As mentioned above, our method translates the coreset concept to these problems, and allows us to understand the sparsity

of SVM solutions, here being the number of support vectors.

1.5. Solution Path Methods

In the two last Chapters 6 and 7, we will study convex optimization problems which are parameterized by a single additional parameter, as for example a time or regularization parameter.

A continuity argument together with our simple proposed concept of optimization duality for compact domains will allow us to study solution paths (of some guaranteed approximation quality) for such problems. We show that piecewise constant solutions can be obtained, and that $O\left(\frac{1}{\varepsilon}\right)$ many such solutions are enough in order to guarantee approximation quality ε along the entire path, independent of the dimension of the problem.

The entire path of solutions to a parameterized problem is interesting in many application areas, in particular for regularization methods as described above. Here the parameter of interest is the regularization parameter t in (1.2), or λ in (1.3), and one would like to study the performance of the solutions (such as e.g. the prediction quality on a separate set of test data) for the continuous path of the solution, as the regularization parameter changes. Other applications of such continuation or homotopy methods can be found e.g. in control theory.

1.6. Notation and Terminology

Convexity. A subset D of some vector space is called *convex*, if for any choice of two points $a, b \in D$, the line segment

$$[a,b] := \big\{\lambda a + (1-\lambda)b \;\big|\; 0 \le \lambda \le 1\big\} \subseteq D$$

is also contained in D. A real-valued function $D \to \mathbb{R}$ is called *convex*, if

$$f\big(\lambda a + (1-\lambda)b\big) \le \lambda f(a) + (1-\lambda) f(b) \qquad \forall a, b \in D,\ 0 \le \lambda \le 1$$

or in other words the line segment connecting the function values at a and b must lie above the graph of the function f.

Spaces and Norms. Consider a vector space $\mathcal{X}$ equipped with an inner product $\langle ., . \rangle$. As the most prominent example of such a space, the reader might always think of the standard n-dimensional Euclidean space $\mathcal{X} = \mathbb{R}^n$ with $\langle x, y \rangle = x^T y$ being the standard scalar product.

For any norm $\|.\|$ on $\mathcal{X}$, the corresponding *dual norm* is defined by

$$\|x\|_* := \sup_{y \in \mathcal{X}, \|y\| \le 1} \langle y, x \rangle \ .$$

Vectors. For a vector $x \in \mathbb{R}^n$, we write $x \ge 0$ if and only if $x_i \ge 0\ \forall i$ holds coordinate-wise. The *sparsity*, or cardinality, of $x \in \mathbb{R}^n$, also written as $\mathrm{card}(x)$, is the number of non-zero coordinates x_i. The ℓ_1*-norm* of a vector is defined as $\|x\|_1 = \sum_{i=1}^n |x_i|$. The standard *Euclidean norm* is given by $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$, and the *max-norm* is defined as $\|x\|_\infty = \max_{i=1}^n |x_i|$. Observe that $\|x\|_\infty \le \|x\|_2 \le \|x\|_1$.

Matrices. For real matrices in $\mathbb{R}^{m \times n}$, the standard *inner product* is defined as $A \bullet B := Tr(A^T B)$, and the (squared) *Frobenius matrix norm* is given by $\|A\|_{Fro}^2 := A \bullet A$, the sum of all squared entries in the matrix. We will use of the fact that the inner product is symmetric, i.e. $A \bullet B = B \bullet A$, which equivalently means that $\mathrm{Tr}(A^T B) = \mathrm{Tr}(B^T A)$.

We will sometimes write $A_{i:}$ for the i-th row of a matrix $A \in \mathbb{R}^{m \times n}$, and $A_{:j}$ for the j-th column.

$\mathbb{S}^{n \times n}$ is the set of *symmetric* $n \times n$ matrices. We write $\lambda_{\max}(A)$ for the largest eigenvalue of a matrix $A \in \mathbb{S}^{n \times n}$, and $\lambda_{\min}(A)$ for the smallest eigenvalue respectively. A is called *positive semidefinite* (PSD), written as $A \succeq 0$, iff $v^T A v \ge 0\ \forall v \in \mathbb{R}^n$. We observe that $v^T A v = A \bullet vv^T$.

2

Convex Optimization without Projection Steps

In this chapter, we study the general problem of minimizing a convex function over a compact convex domain. We will investigate a simple iterative approximation algorithm that does not need projection steps in order to stay inside the optimization domain.

2.1. Introduction

Motivation. For the performance of large scale approximation algorithms for convex optimization, the trade-off between the number of iterations on one hand, and the computational cost per iteration on the other hand, is of crucial importance. The lower complexity per iteration is among the main reasons why first-order methods (i.e., methods using only information from the first derivative of the objective function), as for example stochastic gradient descent, are currently used much more widely and successfully in many machine learning applications — despite the fact that they often need a larger number of iterations than for example second-order methods.

Classical gradient descent optimization techniques usually require a projection step in each iteration, in order to get back to the feasible region. For a variety of applications, this is a non-trivial and costly step. One

prominent example is semidefinite optimization, where the projection of an arbitrary symmetric matrix back to the PSD matrices requires the computation of a complete eigenvalue-decomposition.

Here we study a simple first-order approach that does not need any projection steps, and is applicable to any convex optimization problem over a compact convex domain. The algorithm is a generalization of an existing method originally proposed by Frank & Wolfe [FW56], which was recently extended and analyzed in the seminal paper of Clarkson [Cla10] for optimization over the unit simplex.

Instead of a projection, the primitive operation of the optimizer here is to minimize a *linear* approximation to the function over the same (compact) optimization domain. Any (approximate) minimizer of this simpler linearized problem is then chosen as the next step-direction. Because all such candidates are always feasible for the original problem, the algorithm will automatically stay in our convex feasible region. The analysis will show that the number of steps needed is roughly identical to classical gradient descent schemes, meaning that $O\left(\frac{1}{\varepsilon}\right)$ steps suffice in order to obtain an approximation quality of $\varepsilon > 0$.

The main question about the efficiency per iteration of our algorithm, compared to a classical gradient descent step, can not be answered generally in favor of one or the other. Whether a projection or a linearized problem is computationally cheaper will crucially depend on the shape and the representation of the feasible region. Interestingly, if we consider convex optimization over the Euclidean $\|.\|_2$-ball, the two approaches fully coincide, i.e., we exactly recover classical gradient descent. However there are several classes of optimization problems where the linearization approach we present here is definitely very attractive, and leads to faster and simpler algorithms. This includes for example ℓ_1-regularized problems, which we discuss in Sections 3.1 and 3.2, as well as semidefinite optimization under bounded trace, as studied by [Haz08], see Section 3.4.

Sparsity and Low-Rank. For these mentioned specific classes of convex optimization problems, we will in Chapter 3 additionally demonstrate that our algorithm leads to (optimally) sparse or low-rank solutions. This property is a crucial side-effect that can usually not be achieved by classical optimization techniques, and corresponds to the *coreset* concept known from computational geometry, see also Chapter 5. More precisely, we show matching upper and lower bounds of $\Theta\left(\frac{1}{\varepsilon}\right)$ for the sparsity of solutions to general ℓ_1-regularized problems, and also for optimizing over the simplex, if the required approximation quality is ε. For matrix optimiza-

tion, an analogous statement will hold for the rank in case of nuclear norm regularized problems.

Applications. Applications of the first mentioned class of ℓ_1-regularized problems do include many machine learning algorithms ranging from support vector machines (SVMs) to boosting and multiple kernel learning, as well as ℓ_2-support vector regression (SVR), mean-variance analysis in portfolio selection [Mar52], the smallest enclosing ball problem [BC07], ℓ_1-regularized least squares (also known as basis pursuit de-noising in compressed sensing), the *Lasso* [Tib96], and ℓ_1-regularized logistic regression [KKB07] as well as walking of artificial dogs over rough terrain [KBP+10].

The second mentioned class of matrix problems, that is, optimizing over semidefinite matrices with bounded trace, has applications in low-rank recovery [FHB01, CR09, CT10], dimensionality reduction, matrix factorization and completion problems, as well as general semidefinite programs (SDPs). Further applications to nuclear norm and max-norm optimization, such as sparse/robust PCA will also be discussed in Chapter 4.

History and Related Work. The class of first-order optimization methods in the spirit of Frank and Wolfe [FW56] has a rich history in the literature. Although the focus of the original paper was on quadratic programming, its last section [FW56, Section 6] already introduces the general algorithm for minimizing convex functions using the above mentioned linearization idea, when the optimization domain is given by linear inequality constraints. In this case, each intermediate step consists of solving a linear program. The given convergence guarantee bounds the primal error, and assumes that all internal problems are solved exactly.

Later [DH78] has generalized the same method to arbitrary convex domains, and improved the analysis to also work when the internal subproblems are only solved approximately, see also [Dun80]. Patriksson in [Pat93, Pat98] then revisited the general optimization paradigm, investigated several interesting classes of convex domains, and coined the term "cost approximation" for this type of algorithms. More recently, [Zha03] considered optimization over convex hulls, and studies the crucial concept of sparsity of the resulting approximate solutions. However, this proposed algorithm does not use linear subproblems.

The most recent work of Clarkson [Cla10] provides a good overview of the existing lines of research, and investigates the sparsity solutions when the optimization domain is the unit simplex, and establishing the con-

nection to coreset methods from computational geometry. Furthermore, [Cla10] was the first to introduce the stronger notion of convergence in primal-dual error for this class of problems, and relating this notion of duality gap to Wolfe duality.

Our Contributions. The character of this chapter mostly lies in reviewing, re-interpreting and generalizing the existing approach given by [Cla10], [Haz08] and the earlier papers by [Zha03, DH78, FW56], who do deserve credit for the analysis techniques. Our contribution here is to transfer these methods to the more general case of convex optimization over arbitrary bounded convex subsets of a vector space, while providing stronger primal-dual convergence guarantees. To do so, we propose a very simple alternative concept of optimization duality, which will allow us to generalize the stronger primal-dual convergence analysis which [Cla10] has provided for the the simplex case, to optimization over arbitrary convex domains. So far, no such guarantees on the duality gap were known in the literature for the Frank-Wolfe-type algorithms [FW56], except when optimizing over the simplex. Furthermore, we generalize Clarkson's analysis [Cla10] to work when only approximate linear internal optimizers are used, and to arbitrary starting points. Also, we study the sparsity of solutions in more detail, obtaining upper and lower bounds for the sparsity of approximate solutions for a wider class of domains.

Our proposed notion of duality gives simple certificates for the current approximation quality, which can be used for any optimization algorithm for convex optimization problems over bounded domain, even in the case of non-differentiable objective functions.

We demonstrate the broad applicability of our general technique to several important classes of optimization problems, such as ℓ_1- and ℓ_∞-regularized problems, as well as semidefinite optimization with uniformly bounded diagonal, and sparse semidefinite optimization.

Later in Chapter 4 we will give a simple transformation in order to apply the first-order optimization techniques we review here also to nuclear norm and max-norm matrix optimization problems.

2.2. The Poor Man's Approach to Convex Optimization and Duality

The Idea of a Duality given by Supporting Hyperplanes. Suppose we are given the task of minimizing a convex function f over a bounded convex set $D \subset \mathbb{R}^n$, and let us assume for the moment that f is continuously differentiable.

Then for any point $x \in D$, it seems natural to consider the tangential "supporting" hyperplane to the graph of the function f at the point $(x, f(x))$. Since the function f is convex, any such linear approximation must lie below the graph of the function.

Using this linear approximation for each point $x \in D$, we define a *dual* function value $\omega(x)$ as the minimum of the linear approximation to f at point x, where the minimum is taken over the domain D. We note that the point attaining this linear minimum also seems to be good direction of improvement for our original minimization problem given by f, as seen from the current point x. This idea will lead to the optimization algorithm that we will discuss below.

As the entire graph of f lies above any such linear approximation, it is easy to see that $\omega(x) \leq f(y)$ holds for each pair $x, y \in D$. This fact is called *weak duality* in the optimization literature.

This rather simple definition already completes the duality concept that we will need in this thesis. We will provide a slightly more formal and concise definition in the next subsection, which is useful also for the case of non-differentiable convex functions. The reason we call this concept a *poor man's* duality is that we think it is considerably more direct and intuitive for the setting here, when compared to classical Lagrange duality or Wolfe duality, see e.g. [BV04].

2.2.1. Subgradients of a Convex Function

In the following, we will work over a general vector space $\mathcal{X}$ equipped with an inner product $\langle ., . \rangle$. As the most prominent example in our investigations, the reader might always think of the case $\mathcal{X} = \mathbb{R}^n$ with $\langle x, y \rangle = x^T y$ being the standard Euclidean scalar product.

We consider general convex optimization problems given by a convex

function $f : \mathcal{X} \to \mathbb{R}$ over a compact[1] convex domain $D \subseteq \mathcal{X}$, or formally

$$\underset{x \in D}{\text{minimize}} \; f(x) \; . \tag{2.1}$$

In order to develop both our algorithm and the notion of duality for such convex optimization problems in the following, we need to formally define the supporting hyperplanes at a given point $x \in D$. These planes coincide exactly with the well-studied concept of *subgradients* of a convex function.

For each point $x \in D$, the *subdifferential* at x is defined as the set of normal vectors of the affine linear functions through $(x, f(x))$ that lie below the function f. Formally

$$\partial f(x) := \{d_x \in \mathcal{X} \mid f(y) \geq f(x) + \langle y - x, d_x \rangle \quad \forall y \in D\} \; . \tag{2.2}$$

Any element $d_x \in \partial f(x)$ is called a *subgradient* to f at x. Note that for each x, $\partial f(x)$ is a closed convex set. Furthermore, if f is differentiable, then the subdifferential consists of exactly one element for each $x \in D$, namely $\partial f(x) = \{\nabla f(x)\}$, as explained e.g. in [Nem05, KZ05].

If we assume that f is convex and lower semicontinuous[2] on D, then it is known that $\partial f(x)$ is non-empty, meaning that there exists at least one subgradient d_x for every point $x \in D$. For a more detailed investigation of subgradients, we refer the reader to one of the works of e.g. [Roc97, BV04, Nem05, KZ05, BL06].

2.2.2. A Duality for Convex Optimization over Compact Domain

For a given point $x \in D$, and any choice of a subgradient $d_x \in \partial f(x)$, we define a *dual* function value

$$\omega(x, d_x) := \min_{y \in D} \; f(x) + \langle y - x, d_x \rangle \; . \tag{2.3}$$

In other words $\omega(x, d_x) \in \mathbb{R}$ is the minimum of the linear approximation to f defined by the subgradient d_x at the supporting point x, where the

[1] Here we call a set $D \subseteq X$ *compact* if it is closed and bounded. See [KZ05] for more details.

[2] The assumption that our objective function f is lower semicontinuous on D, is equivalent to the fact that its epigraph — i.e. the set $\{(x, t) \in D \times \mathbb{R} \mid t \geq f(x)\}$ of all points lying on or above the graph of the function — is closed, see also [KZ05, Theorem 7.1.2].

minimum is taken over the domain D. This minimum is always attained, since D is compact, and the linear function is continuous in y.

By the definition of the subgradient — as lying below the graph of the function f — we readily attain the property of weak-duality, which is at the core of the optimization approach we will study below.

Lemma 2.1 (Weak duality). *For all pairs $x, y \in D$, it holds that*

$$\omega(x, d_x) \leq f(y)$$

Proof. Immediately from the definition of the dual $\omega(.,.)$:

$$\begin{aligned} \omega(x, d_x) = & \quad \min_{z \in D} \; f(x) + \langle z - x, d_x \rangle \\ \leq & \quad f(x) + \langle y - x, d_x \rangle \\ \leq & \quad f(y) \; . \end{aligned}$$

Here the last inequality is by the definition (2.2) of a subgradient. □

Geometrically, this fact can be understood as that any function value $f(y)$, which is "part of" the graph of f, always lies higher than the minimum over any linear approximation (given by d_x) to f.

In the case that f is differentiable, there is only one possible choice for a subgradient, namely $d_x = \nabla f(x)$, and so we will then denote the (unique) dual value for each x by

$$\omega(x) := \omega(x, \nabla f(x)) = \min_{y \in D} \; f(x) + \langle y - x, \nabla f(x) \rangle \; . \tag{2.4}$$

The Duality Gap as a Measure of Approximation Quality. The above duality concept allows us to compute a very simple measure of approximation quality, for any candidate solution $x \in D$ to problem (2.1). This measure will be easy to compute even if the true optimum value $f(x^*)$ is unknown, which will very often be the case in practical applications. The duality gap $g(x, d_x)$ at any point $x \in D$ and any subgradient subgradient $d_x \in \partial f(x)$ is defined as

$$g(x, d_x) := f(x) - \omega(x, d_x) = \max_{y \in D} \langle x - y, d_x \rangle \; , \tag{2.5}$$

or in other words as the difference of the current function value $f(x)$ to the minimum value of the corresponding linearization at x, taken over the domain D. The quantity $g(x, d_x) = f(x) - \omega(x, d_x)$ will be called the *duality gap* at x, for the chosen d_x.

By the weak duality Lemma 2.1, we obtain that for any optimal solution x^* to problem (2.1), it holds that

$$g(x, d_x) \geq f(x) - f(x^*) \geq 0 \qquad \forall x \in D,\ \forall d_x \in \partial f(x)\ . \tag{2.6}$$

Here the quantity $f(x) - f(x^*)$ is what we call the *primal error* at point x, which is usually impossible to compute due to x^* being unknown. The above inequality (2.6) now gives us that the duality gap — which is easy to compute, given d_x — is always an upper bound on the primal error. This property makes the duality gap an extremely useful measure for example as a stopping criterion in practical optimizers or heuristics.

We call a point $x \in \mathcal{X}$ an *ε-approximation* if $g(x, d_x) \leq \varepsilon$ for some choice of subgradient $d_x \in \partial f(x)$.

For the special case that f is differentiable, we will again use the simpler notation $g(x)$ for the (unique) duality gap for each x, i.e.

$$g(x) := g(x, \nabla f(x)) = \max_{y \in D} \langle x - y, \nabla f(x) \rangle\ .$$

Relation to Duality of Norms. In the special case when the optimization domain D is given by the unit ball of some norm on the space $\mathcal{X}$, we observe the following:

Observation 2.2. *For optimization over any domain $D = \{x \in \mathcal{X} \mid \|x\| \leq 1\}$ being the unit ball of some norm $\|.\|$, the duality gap for the optimization problem $\min_{x \in D} f(x)$ is given by*

$$g(x, d_x) = \|d_x\|_* + \langle x, d_x \rangle\ ,$$

where $\|.\|_*$ is the dual norm of $\|.\|$.

Proof. Directly by the definitions of the dual norm $\|x\|_* = \sup_{\|y\| \leq 1} \langle y, x \rangle$, and the duality gap $g(x, d_x) = \max_{y \in D} \langle y, -d_x \rangle + \langle x, d_x \rangle$ as in (2.5). □

2.3. A Projection-Free First-Order Method for Convex Optimization

2.3.1. The Algorithm

In the following we will generalize the sparse greedy algorithm of [FW56] and its analysis by [Cla10] to convex optimization over arbitrary compact

convex sets $D \subseteq \mathcal{X}$ of a vector space. More formally, we assume that the space $\mathcal{X}$ is a Hilbert space, and consider problems of the form (2.1), i.e.,

$$\underset{x \in D}{\text{minimize}} \; f(x) \; .$$

Here we suppose that the objective function f is differentiable over the domain D, and that for any $x \in D$, we are given the gradient $\nabla f(x)$ via an oracle.

The existing algorithms so far did only apply to convex optimization over the simplex (or convex hulls in some cases) [Cla10], or over the spectahedron of PSD matrices [Haz08], or then did not provide guarantees on the duality gap. Inspired by the work of Hazan [Haz08], we can also relax the requirement of *exactly* solving the linearized problem in each step, to just computing *approximations* thereof, while keeping the same convergence results. This allows for more efficient steps in many applications.

Also, our algorithm variant works for arbitrary starting points, without needing to compute the best initial "starting vertex" of D as in [Cla10].

The Primal-Dual Idea. We are motivated by the geometric interpretation of the "poor man's" duality gap, as explained in the previous Section 2.2. This duality gap is the maximum difference of the current value $f(x)$, to the linear approximation to f at the currently fixed point x, where the linear maximum is taken over the domain D. This observation leads to the algorithmic idea of directly using the current linear approximation (over the convex domain D) to obtain a good direction for the next step, automatically staying in the feasible region.

The general optimization method with its two precision variants is given in Algorithm 1. For the approximate variant, the constant $C_f > 0$ is an upper bound on the curvature of the objective function f, which we will explain below in more details.

The Linearized Optimization Primitive. The internal "step direction" procedure ExactLinear(c, D) used in Algorithm 1 is a method that minimizes the linear function $\langle x, c \rangle$ over the compact convex domain D. Formally it must return a point $s \in D$ such that $\langle s, c \rangle = \min_{y \in D} \langle y, c \rangle$. In terms of the smooth convex optimization literature, the vectors y that have negative scalar product with the gradient, i.e. $\langle y, \nabla f(x) \rangle < 0$, are called *descent directions*, see e.g. [BV04, Chapter 9]. The main difference to classical convex optimization is that we always choose descent steps staying in the domain D, where traditional gradient descend techniques usually use

Algorithm 1 Greedy on a Convex Set

Input: Convex function f, convex set D, target accuracy ε
Output: ε-approximate solution for problem (2.1)
Pick an arbitrary starting point $x^{(0)} \in D$
for $k = 0 \dots \infty$ **do**
  Let $\alpha := \frac{2}{k+2}$
    Compute $s := \textsc{ExactLinear}\left(\nabla f(x^{(k)}), D\right)$
      {Solve the linearized primitive problem exactly}
  —or—
    Compute $s := \textsc{ApproxLinear}\left(\nabla f(x^{(k)}), D, \alpha C_f\right)$
      {Approximate the linearized primitive problem}
  Update $x^{(k+1)} := x^{(k)} + \alpha(s - x^{(k)})$
end for

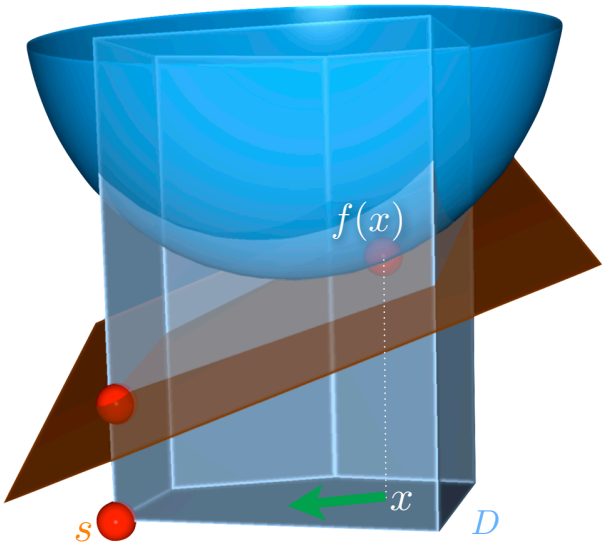


Figure 2.1.: Visualization of a step of Algorithm 1, moving from the current point $x = x^{(k)}$ towards a linear minimizer $s \in D$. Here the two-dimensional domain D is part of the ground plane, and we plot the function values vertically. Visualization by Robert Carnecky.

arbitrary directions and need to project back onto D after each step. We will comment more on this analogy in Section 2.3.7.

In the alternative interpretation of our duality concept from Section 2.2, the linearized sub-task means that we search for a point s that "realizes" the current duality gap $g(x)$, that is the distance to the linear approximation, as given in (2.5).

In the special case that the set D is given by an intersection of linear constraints, this sub-task is exactly equivalent to *linear programming*, as already observed by [FW56, Section 6]. However, for many other representations of important specific feasible domains D, this internal primitive operation is significantly easier to solve, as we will see in the later sections.

The alternative approximate variant of Algorithm 1 uses the procedure ApproxLinear (c, D, ε') as the internal "step-direction generator". Analogously to the exact primitive, this procedure *approximates* the minimum of the linear function $\langle x, c\rangle$ over the convex domain D, with additive error $\varepsilon' > 0$. Formally ApproxLinear (c, D, ε') must return a point $s \in D$ such that $\langle s, c\rangle \leq \min_{y \in D} \langle y, c\rangle + \varepsilon'$. For several applications, this can be done significantly more efficiently than the exact variant, see e.g. the applications for semidefinite programming in Section 3.4.

The Curvature. Everything we need for the analysis of Algorithm 1 is that the linear approximation to our (convex) function f at any point x does not deviate from f by too much, when taken over the whole optimization domain D.

The *curvature constant* C_f of a convex and differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, with respect to the compact domain D is defined as.

$$C_f := \sup_{\substack{x,s \in D,\\ \alpha \in [0,1],\\ y = x + \alpha(s-x)}} \frac{1}{\alpha^2} \left(f(y) - f(x) - \langle y - x, \nabla f(x)\rangle \right) . \tag{2.7}$$

A motivation to consider this quantity follows if we imagine our optimization procedure at the current point $x = x^{(k)}$, and choosing the next iterate as $y = x^{(k+1)} := x + \alpha(s - x)$. Bounded C_f then means that the deviation of f at y from the "best" linear prediction given by $\nabla f(x)$ is bounded, where the acceptable deviation is weighted by the inverse of the squared step-size α. For linear functions f for example, it holds that $C_f = 0$.

The defining term $f(y) - f(x) - \langle y - x, d_x\rangle$ is also widely known as the *Bregman divergence* defined by f. The quantity C_f turns out to be small for many relevant applications, some of which we will discuss later, or see also [Cla10].

The assumption of bounded curvature C_f is indeed very similar to a Lipschitz assumption on the gradient of f, see also the discussion in Sections 2.3.4 and 2.3.7. In the optimization literature, this property is sometimes also called *C_f-strong smoothness*.

It will not always be possible to compute the constant C_f exactly. However, for all algorithms in the following, and also their analysis, it is sufficient to just use some upper bound on C_f. We will comment in some more details on the curvature measure in Section 2.3.4.

Convergence. The following theorem shows that after $O\left(\frac{1}{\varepsilon}\right)$ many iterations, Algorithm 1 obtains an ε-approximate solution. The analysis essentially follows the approach of [Cla10], inspired by the earlier work of [FW56, DH78, Pat93] and [Zha03]. Later, in Section 3.1.2, we will show that this convergence rate is indeed best possible for this type of algorithm, when considering optimization over the unit-simplex. More precisely, we will show that the dependence of the sparsity on the approximation quality, as given by the algorithm here, is best possible up to a constant factor. Analogously, for the case of semidefinite optimization with bounded trace, we will prove in Section 3.4.4 that the obtained (low) rank of the approximations given by this algorithm is optimal, for the given approximation quality.

Theorem 2.3 (Primal Convergence)**.** *For each $k \geq 1$, the iterate $x^{(k)}$ of the exact variant of Algorithm 1 satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{4C_f}{k+2} \ ,$$

where $x^* \in D$ is an optimal solution to problem (2.1). For the approximate variant of Algorithm 1, it holds that

$$f(x^{(k)}) - f(x^*) \leq \frac{8C_f}{k+2} \ .$$

(In other words both algorithm variants deliver a solution of primal error at most ε after $O(\frac{1}{\varepsilon})$ many iterations.)

The proof of the above theorem on the convergence-rate of the primal error crucially depends on the following Lemma 2.4 on the improvement in each iteration. We recall from Section 2.2 that the duality gap for the general convex problem (2.1) over the domain D is given by $g(x) = \max_{s \in D} \langle x - s, \nabla f(x) \rangle$.

Lemma 2.4. *For any step* $x^{(k+1)} := x^{(k)} + \alpha(s - x^{(k)})$ *with arbitrary step-size* $\alpha \in [0, 1]$*, it holds that*

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \alpha g(x^{(k)}) + \alpha^2 C_f$$

if s *is given by* $s := \textsc{ExactLinear}\,(\nabla f(x), D)$*.*

If the approximate primitive $s := \textsc{ApproxLinear}\,(\nabla f(x), D, \alpha C_f)$ *is used instead, then it holds that*

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \alpha g(x^{(k)}) + 2\alpha^2 C_f \ .$$

Proof. We write $x := x^{(k)}$, $y := x^{(k+1)} = x + \alpha(s - x)$, and $d_x := \nabla f(x)$ to simplify the notation, and first prove the second part of the lemma. We use the definition of the curvature constant C_f of our convex function f, to obtain

$$\begin{aligned} f(y) = \ & f(x + \alpha(s - x)) \\ \leq \ & f(x) + \alpha\langle s - x, d_x\rangle + \alpha^2 C_f \ . \end{aligned}$$

Now we use that the choice of $s := \textsc{ApproxLinear}\,(d_x, D, \varepsilon')$ is a good "descent direction" on the linear approximation to f at x. Formally, we are given a point s that satisfies $\langle s, d_x\rangle \leq \min_{y \in D}\langle y, d_x\rangle + \varepsilon'$, or in other words we have

$$\begin{aligned} \langle s - x, d_x\rangle \leq \ & \min_{y \in D}\langle y, d_x\rangle - \langle x, d_x\rangle + \varepsilon' \\ = \ & -g(x, d_x) + \varepsilon' \ , \end{aligned}$$

from the definition (2.5) of the duality gap $g(x) = g(x, d_x)$. Altogether, we obtain

$$\begin{aligned} f(y) \leq \ & f(x) + \alpha(-g(x) + \varepsilon') + \alpha^2 C_f \\ = \ & f(x) - \alpha g(x) + 2\alpha^2 C_f \ , \end{aligned}$$

the last equality following by our choice of $\varepsilon' = \alpha C_f$. This proves the lemma for the approximate case. The first claim for the exact linear primitive $\textsc{ExactLinear}()$ follows by the same proof for $\varepsilon' = 0$. □

Having Lemma 2.4 at hand, the proof of our above primal convergence Theorem 2.3 now follows along the same idea as in [Cla10, Theorem 2.3] or [Haz08, Theorem 1]. Note that a weaker variant of Lemma 2.4 was already proven by [FW56].

Proof of Theorem 2.3. From Lemma 2.4 we know that for every step of the exact variant of Algorithm 1, it holds that $f(x^{(k+1)}) \leq f(x^{(k)}) - \alpha g(x^{(k)}) + \alpha^2 C_f$.

Writing $h(x) := f(x) - f(x^*)$ for the (unknown) primal error at any point x, this implies that

$$\begin{aligned} h(x^{(k+1)}) &\leq h(x^{(k)}) - \alpha g(x^{(k)}) + \alpha^2 C_f \\ &\leq h(x^{(k)}) - \alpha h(x^{(k)}) + \alpha^2 C_f \\ &= (1-\alpha) h(x^{(k)}) + \alpha^2 C_f \ , \end{aligned} \tag{2.8}$$

where we have used weak duality $h(x) \leq g(x)$ as given by in (2.6). We will now use induction over k in order to prove our claimed bound, i.e.,

$$h(x^{(k+1)}) \leq \frac{4C_f}{k+1+2} \qquad k = 0 \ldots \infty \ .$$

The base-case $k = 0$ follows from (2.8) applied for the first step of the algorithm, using $\alpha = \alpha^{(0)} = \frac{2}{0+2} = 1$.

Now considering $k \geq 1$, the bound (2.8) gives us

$$\begin{aligned} h(x^{(k+1)}) &\leq (1-\alpha^{(k)}) h(x^{(k)}) + {\alpha^{(k)}}^2 C_f \\ &= (1 - \tfrac{2}{k+2}) h(x^{(k)}) + (\tfrac{2}{k+2})^2 C_f \\ &\leq (1 - \tfrac{2}{k+2}) \tfrac{4C_f}{k+2} + (\tfrac{2}{k+2})^2 C_f \ , \end{aligned}$$

where in the last inequality we have used the induction hypothesis for $x^{(k)}$. Simply rearranging the terms gives

$$\begin{aligned} h(x^{(k+1)}) &\leq \tfrac{4C_f}{k+2} - \tfrac{8C_f}{(k+2)^2} + \tfrac{4C_f}{(k+2)^2} \\ &= 4C_f \left(\tfrac{1}{k+2} - \tfrac{1}{(k+2)^2} \right) \\ &= \tfrac{4C_f}{k+2} \tfrac{k+2-1}{k+2} \\ &\leq \tfrac{4C_f}{k+2} \tfrac{k+2}{k+3} \\ &= \tfrac{4C_f}{k+3} \ , \end{aligned}$$

which is our claimed bound for $k \geq 1$.

The analogous claim for Algorithm 1 using the approximate linear primitive ApproxLinear() follows from the exactly same argument, by replacing every occurrence of C_f in the proof here by $2C_f$ instead (compare to Lemma 2.4 also). □

2.3.2. Obtaining a Guaranteed Small Duality Gap

From the above Theorem 2.3 on the convergence of Algorithm 1, we have obtained small primal error. However, the optimum value $f(x^*)$ is unknown in most practical applications, where we would prefer to have an

easily computable measure of the current approximation quality, for example as a stopping criterion of our optimizer in the case that C_f is unknown. The duality gap $g(x)$ that we defined in Section 2.2 satisfies these requirements, and always upper bounds the primal error $f(x) - f(x^*)$.

By a nice argument of Clarkson [Cla10, Theorem 2.3], the convergence on the simplex optimization domain can be extended to obtain the stronger property of guaranteed small duality gap $g(x^{(k)}) \leq \varepsilon$, after at most $O(\frac{1}{\varepsilon})$ many iterations. This stronger convergence result was not yet known in earlier papers of [FW56, DH78, Jon92, Pat93] and [Zha03]. Here we will generalize the primal-dual convergence to arbitrary compact convex domains. The proof of our theorem below again relies on Lemma 2.4.

Theorem 2.5 (Primal-Dual Convergence). *Let* $K := \left\lceil \frac{4C_f}{\varepsilon} \right\rceil$. *We run the exact variant of Algorithm 1 for* K *iterations (recall that the step-sizes are given by* $\alpha^{(k)} := \frac{2}{k+2}$, $0 \leq k \leq K$*), and then continue for another* $K+1$ *iterations, now with the fixed step-size* $\alpha^{(k)} := \frac{2}{K+2}$ *for* $K \leq k \leq 2K+1$.

Then the algorithm has an iterate $x^{(\hat{k})}$, $K \leq \hat{k} \leq 2K+1$, *with duality gap bounded by*

$$g(x^{(\hat{k})}) \leq \varepsilon \ .$$

The same statement holds for the approximate variant of Algorithm 1, when setting $K := \left\lceil \frac{8C_f}{\varepsilon} \right\rceil$ *instead.*

Proof. The proof follows the idea of [Cla10, Section 7].

By our previous Theorem 2.3 we already know that the primal error satisfies $h(x^{(K)}) = f(x^{(K)}) - f(x^*) \leq \frac{4C_f}{K+2}$ after K iterations. In the subsequent $K+1$ iterations, we will now suppose that $g(x^{(k)})$ always stays larger than $\frac{4C_f}{K+2}$. We will try to derive a contradiction to this assumption.

Putting the assumption $g(x^{(k)}) > \frac{4C_f}{K+2}$ into the step improvement bound given by Lemma 2.4, we get that

$$\begin{aligned} f(x^{(k+1)}) - f(x^{(k)}) \leq & \ -\alpha^{(k)} g(x^{(k)}) + {\alpha^{(k)}}^2 C_f \\ < & \ -\alpha^{(k)} \tfrac{4C_f}{K+2} + {\alpha^{(k)}}^2 C_f \end{aligned}$$

holds for any step size $\alpha^{(k)} \in (0, 1]$. Now using the fixed step-size $\alpha^{(k)} = \frac{2}{K+2}$ in the iterations $k \geq K$ of Algorithm 1, this reads as

$$\begin{aligned} f(x^{(k+1)}) - f(x^{(k)}) < & \ -\tfrac{2}{K+2}\tfrac{4C_f}{K+2} + \tfrac{4}{(K+2)^2} C_f \\ = & \ -\tfrac{4C_f}{(K+2)^2} \end{aligned}$$

Summing up over the additional steps, we obtain

$$\begin{aligned} f(x^{(2K+2)}) - f(x^{(K)}) = & \sum_{k=K}^{2K+1} f(x^{(k+1)}) - f(x^{(k)}) \\ < & -(K+2)\tfrac{4C_f}{(K+2)^2} = -\tfrac{4C_f}{K+2} \ , \end{aligned}$$

which together with our known primal approximation error $f(x^{(K)}) - f(x^*) \leq \frac{4C_f}{K+2}$ would result in $f(x^{(2K+2)}) - f(x^*) < 0$, a contradiction. Therefore there must exist $\hat{k}$, $K \leq \hat{k} \leq 2K+1$, with $g(x^{(\hat{k})}) \leq \frac{4C_f}{K+2} \leq \varepsilon$.

The analysis for the approximate variant of Algorithm 1 follows using the analogous second bound from Lemma 2.4. □

Following [Cla10, Theorem 2.3], one can also prove a similar primal-dual convergence theorem for the line-search algorithm variant that uses the optimal step-size in each iteration, as we will discuss in the next Section 2.3.3. This is somewhat expected as the line-search algorithm in each step is at least as good as the fixed step-size variant we consider here.

2.3.3. Choosing the Optimal Step-Size by Line-Search

Alternatively, instead of the fixed step-size $\alpha = \frac{2}{k+2}$ in Algorithm 1, one can also find the optimal $\alpha \in [0,1]$ by line-search. This will not improve the theoretical convergence guarantee, but might still be considered in practical applications if the best α is easy to compute. Experimentally, we observed that line-search can improve the numerical stability in particular if approximate step directions are used, which we will discuss e.g. for semidefinite matrix completion problems in Section 4.5.

Formally, given the chosen direction s, we then search for the α of best improvement in the objective function f, that is

$$\alpha := \operatorname*{arg\,min}_{\alpha \in [0,1]} f\left(x^{(k)} + \alpha(s - x^{(k)})\right) \ . \tag{2.9}$$

The resulting modified version of Algorithm 1 is depicted again in Algorithm 2, and was precisely analyzed in [Cla10] for the case of optimizing over the simplex.

In many cases, we can solve this line-search analytically in a straightforward manner, by differentiating the above expression with respect to α: Consider $f_\alpha := f\left(x_{(\alpha)}^{(k+1)}\right) = f\left(x^{(k)} + \alpha\left(s - x^{(k)}\right)\right)$ and compute

$$0 \stackrel{!}{=} \frac{\partial}{\partial \alpha} f_\alpha = \left\langle s - x^{(k)}, \nabla f\big(x_{(\alpha)}^{(k+1)}\big) \right\rangle \ . \tag{2.10}$$

Algorithm 2 Greedy on a Convex Set, using Line-Search

Input: Convex function f, convex set D, target accuracy ε
Output: ε-approximate solution for problem (3.1)
Pick an arbitrary starting point $x^{(0)} \in D$
for $k = 0 \dots \infty$ **do**
  Compute $s := \textsc{ExactLinear}\left(\nabla f(x^{(k)}), D\right)$
  —or—
  Compute $s := \textsc{ApproxLinear}\left(\nabla f(x^{(k)}), D, \frac{2C_f}{k+2}\right)$
  Find the optimal step-size $\alpha := \arg\min_{\alpha \in [0,1]} f\left(x^{(k)} + \alpha(s - x^{(k)})\right)$
  Update $x^{(k+1)} := x^{(k)} + \alpha(s - x^{(k)})$
end for

If this equation can be solved for α, then the optimal such α can directly be used as the step-size in Algorithm 1, and the convergence guarantee of Theorem 2.3 still holds. This is because the improvement in each step will be at least as large as if we were using the older (potentially sub-optimal) fixed choice of $\alpha = \frac{2}{k+2}$. Here we have assumed that $\alpha^{(k)} \in [0, 1]$ always holds, which can be done when using some caution, see also [Cla10].

Note that the line-search can also be used for the approximate variant of Algorithm 1, where we keep the accuracy for the internal primitive method $\textsc{ApproxLinear}\left(\nabla f(x^{(k)}), D, \varepsilon'\right)$ fixed to $\varepsilon' = \alpha_{\text{fixed}} C_f = \frac{2C_f}{k+2}$. Theorem 2.3 then holds as as in the original case.

2.3.4. The Curvature Measure of a Convex Function

For the case of differentiable f over the space $\mathcal{X} = \mathbb{R}^n$, we recall the definition of the curvature constant C_f with respect to the domain $D \subset \mathbb{R}^n$, as stated in (2.7),

$$C_f := \sup_{\substack{x,s \in D,\\ \alpha \in [0,1],\\ y = x + \alpha(s-x)}} \frac{1}{\alpha^2} \left(f(y) - f(x) - (y - x)^T \nabla f(x) \right) .$$

An overview of values of C_f for several classes of functions f over domains that are related to the unit simplex can be found in [Cla10].

Asymptotic Curvature. As Algorithm 1 converges towards some optimal solution x^*, it also makes sense to consider the asymptotic curvature C_f^*,

defined as

$$C_f^* := \sup_{\substack{s\in D,\\ \alpha\in[0,1],\\ y=x^*+\alpha(s-x^*)}} \tfrac{1}{\alpha^2}\left(f(y) - f(x^*) - (y-x^*)^T\nabla f(x^*)\right) \ . \tag{2.11}$$

Clearly $C_f^* \leq C_f$. As described in [Cla10, Section 4.4], we expect that as the algorithm converges towards x^*, also the improvement bound as given by Lemma 2.4 should be determined by $C_f^* + o(1)$ instead of C_f, resulting in a better convergence speed constant than given Theorem 2.3, at least for large k. The class of *strongly convex* functions is an example for which the convergence of the relevant constant towards C_f^* is easy to see, since for these functions, convergence in the function value also imlies convergence in the domain, towards a unique point x^*, see e.g. [BV04, Section 9.1.2].

Relating the Curvature to the Hessian Matrix. Before we can compare the assumption of bounded curvature C_f to a Lipschitz assumption on the gradient of f, we will need to relate C_f to the Hessian matrix (matrix of all second derivatives) of f.

Here the idea described in [Cla10, Section 4.1] is to make use of the degree-2 Taylor-expansion of our function f at the fixed point x, as a function of α, which is

$$f(x+\alpha(s-x)) = f(x) + \alpha(s-x)^T\nabla f(x) + \frac{\alpha^2}{2}(s-x)^T\nabla^2 f(z)(s-x) \ ,$$

where z is a point on the line-segment $[x,y] \subseteq D \subset \mathbb{R}^d$ between the two points $x \in \mathbb{R}^n$ and $y = x + \alpha(s-x) \in \mathbb{R}^n$. To upper bound the curvature measure, we can now directly plug in this expression for $f(y)$ into the above definition of C_f, obtaining

$$C_f \leq \sup_{\substack{x,y\in D,\\ z\in[x,y]\subseteq D}} \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x) \ . \tag{2.12}$$

From this bound, it follows that C_f is upper bounded by the largest eigenvalue of the Hessian matrix of f, scaled with the domain's Euclidean diameter, or formally

Lemma 2.6. *For any twice differentiable convex function f over a compact convex domain D, it holds that*

$$C_f \ \leq \ \frac{1}{2}\operatorname{diam}(D)^2 \cdot \sup_{z\in D} \lambda_{\max}\left(\nabla^2 f(z)\right) \ .$$

Note that as f is convex, the Hessian matrix $\nabla^2 f(z)$ is positive semidefinite for all z, see e.g. [KZ05, Theorem 7.3.6].

Proof. Applying the Cauchy-Schwarz inequality to (2.12) for any $x, y \in D$ (as in the definition of C_f), we get

$$\begin{aligned}(y-x)^T \nabla^2 f(z)(y-x) \leq & \quad \|y-x\|_2 \left\|\nabla^2 f(z)(y-x)\right\|_2 \\ \leq & \quad \|y-x\|_2^2 \left\|\nabla^2 f(z)\right\|_{spec} \\ \leq & \quad \operatorname{diam}(D)^2 \cdot \sup_{z \in D} \lambda_{\max}\left(\nabla^2 f(z)\right) \ .\end{aligned}$$

The middle inequality follows from the variational characterization of the matrix spectral norm, i.e. $\|A\|_{spec} := \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$. Finally, in the last inequality we have used that by convexity of f, the Hessian matrix $\nabla^2 f(z)$ is PSD, so that its spectral norm is its largest eigenvalue. □

Note that in the case of D being the unit simplex (see also the following Section 3.1), we have that $\operatorname{diam}(\Delta_n) = \sqrt{2}$, meaning the scaling factor disappears, i.e. $C_f \leq \sup_{z \in \Delta_n} \lambda_{\max}\left(\nabla^2 f(z)\right)$.

Bounded Curvature vs. Lipschitz-Continuous Gradient. Our core assumption on the given optimization problems is that that the curvature C_f of the function is bounded over the domain. Equivalently, this means that the function does not deviate from its linear approximation by too much. Here we will explain that this assumption is in fact very close to the natural assumption that the gradient ∇f is Lipschitz-continuous, which is often assumed in classical convex optimization literature, where it is sometimes called C_f*-strong smoothness*, see e.g. [Nem05, KSST09] (or [d'A08] if the gradient information is only approximate).

Lemma 2.7. *Let f be a convex and twice differentiable function, and assume that the gradient ∇f is Lipschitz-continuous over the domain D with Lipschitz-constant $L > 0$. Then*

$$C_f \leq \frac{1}{2} \operatorname{diam}(D)^2 L \ .$$

Proof. Having $\|\nabla f(y) - \nabla f(x)\|_2 \leq L \|y-x\|_2 \ \forall x, y \in D$ by the Cauchy-Schwarz inequality implies that $(y-x)^T(\nabla f(y) - \nabla f(x)) \leq L \|y-x\|_2^2$, so that

$$f(y) \leq f(x) + (y-x)^T \nabla f(x) + \frac{L}{2} \|y-x\|_2^2 \ . \tag{2.13}$$

If f is twice differentiable, it can directly be seen that the above condition implies that $L \cdot \mathbf{I} \succeq \nabla^2 f(z)$ holds for the Hessian of f, that is $\lambda_{\max}\left(\nabla^2 f(z)\right) \leq L$.

Together with our result from Lemma 2.6, the claim follows. □

The above bound (2.13) which is implied by Lipschitz-continuous gradient means that the function is not "curved" by more than L in some sense, which is an interesting property. In fact this is exactly the opposite inequality compared to the property of *strong convexity*, which is an assumption on the function f that we do not impose here. Strong convexity on a compact domain means that the function is always curved *at least* by some constant (as our L). We just note that for strongly convex functions, "accelerated" algorithms with an even faster convergence of $\frac{1}{k^2}$ (meaning $O(\frac{1}{\sqrt{\varepsilon}})$ steps) do exist [Nes04, Nes07a].

2.3.5. Optimizing over Convex Hulls

In the case that the optimization domain D is given as the convex hull of a (finite or infinite) subset $V \subset \mathcal{X}$, i.e.

$$D = \operatorname{conv}(V) \ ,$$

then it is particularly easy to solve the linear optimization subproblems as needed in our Algorithm 1. Recall that $\operatorname{conv}(V)$ is defined as the set of all finite convex combinations $\sum_i \alpha_i v_i$ for a finite subset $\{v_1, \ldots, v_k\} \subseteq V$, while V can also be an infinite set.

Lemma 2.8 (Linear Optimization over Convex Hulls). *Let $D = \operatorname{conv}(V)$ for any subset $V \subset \mathcal{X}$, and D compact. Then any linear function $y \mapsto \langle y, c\rangle$ will attain its minimum and maximum over D at some "vertex" $v \in V$.*

Proof. W.l.g. we will only show the case for the maximum. Let $s \in D$ be a point attaining the linear optimum $\langle s, c\rangle = \max_{y \in D} \langle y, c\rangle$. Then by definition of D, we have that $s = \sum_{i=1}^{k} \alpha_i v_i$, meaning that s is the weighted average of some finite set of "vertices" $v_1, \ldots, v_k \in V$, with $\alpha_i \geq 0, \sum_i \alpha_i = 1$. By linearity of the inner product,

$$\langle s, c\rangle = \left\langle \sum_{i=1}^{k} \alpha_i v_i, c \right\rangle = \sum_{i=1}^{k} \alpha_i \langle v_i, c\rangle \ ,$$

and therefore we must have that $\langle v_i, c\rangle \geq \langle s, c\rangle$ for at least one of the indices i, meaning that $v_i \in V$ is also attaining the linear maximum. □

In the following we will discuss several application where this simple fact will be useful to solve the linearized subproblems ExactLinear() more efficiently, as the set V is often much easier to describe than the full compact domain D.

The setting of convex optimization over a convex hull in a vector space was already studied by [Zha03]. There, each iteration of the optimizer consists of greedily selecting the point (or "vertex") of V which promises best improvement. [Zha03] then gave a similar primal convergence guarantee as in our Theorem 2.3 (but no primal-dual convergence result on general convex hulls was known so far, to the best of our knowledge). The above Lemma 2.8 in a sense explains the relation to our linearized internal problem. The main difference is that the algorithm of [Zha03] always evaluates the original non-linear function f at all vertices V, while our slightly more general framework only relies on the linear subproblem, and allows for arbitrary means to approximately solve the subproblem.

2.3.6. Randomized Variants, and Stochastic Optimization

For a variety of classes of our convex optimization problems, randomization can help to solve the linearized subproblem more efficiently. This idea is strongly related to online and stochastic optimization, see e.g. [Nes11], and also the popular stochastic gradient descent (SGD) techniques [Bot10].

We can also apply such cheaper randomized steps in our described framework, instead of deterministically solving the internal linear problem in each iteration. Assumed that the user of our method is able to decompose the linearized problem in some arbitrary way using randomization, and if the randomization is such that the linearized problem will be solved "accurately enough" with some probability $p > 0$ in each iteration, then our convergence analysis still holds also in this probabilistic setting as follows:

Formally, we assume that we are given access to a randomized procedure RandomLinear (c, D, ε'), which returns a point $s \in D$ such that $\langle s, c\rangle \leq \min_{y \in D} \langle y, c\rangle + \varepsilon'$ with probability at least $p > 0$. In other words, with a positive probability, RandomLinear() will behave like ApproxLinear(). In each iteration of the line-search variant of our algorithm (see Algorithm 2), we will now use that randomized internal procedure instead. The expected improvement given by a step towards $s =$ RandomLinear() is at least p times the amount given in Lemma 2.4. (Here we have used that in the events of "failure" of RandomLinear(), the objective function value will at least not become worse, due to the use of line-search).

In other words if we perform $\frac{1}{p}$ times more iterations than required

for the deterministic Algorithm 1, then we have that the convergence by Theorem 2.3 also holds for the randomized variant described here.

Stochastic Gradient Descent (SGD). A classical example is when the linearized problem is given by simply finding the maximum over say n coordinates, as we will e.g. see in the following Sections 3.1 and 3.2 for optimizing over the simplex, or over bounded ℓ_1-norm. In this case, by sampling uniformly at random, with probability $\frac{1}{n}$ we will pick the correct coordinate, for which the step improvement is as in the deterministic Algorithm 1. Therefore we have obtained the same convergence guarantee as for the deterministic algorithm, but the necessary number of steps is multiplied by n.

For unconstrained convex optimization, the convergence of SGD and other related methods was analyzed e.g. in [Nes11] and also the earlier paper [Nes07b, Section 6]. Also here, a comparable slow-down was observed when using the cheaper randomized steps.

2.3.7. Relation to Classical Convex Optimization

Relation to Gradient Descent and Steepest Descent. The internal linear optimizer in our Algorithm 1 can also be interpreted in terms of descent directions. Recall that all vectors y that have negative scalar product with the current gradient, i.e. $\langle y, \nabla f(x)\rangle < 0$, are called *descent directions*, see e.g. [BV04, Chapter 9.4]. Also observe that $\langle y, \nabla f(x)\rangle$ is the directional derivative of f in direction of y if y is of unit length. Our method therefore chooses the best descent direction over the entire domain D, where the quality is measured as the best possible absolute improvement as suggested by the linearization at the current point x. In any iteration, this will crucially depend on the *global* shape of the domain D, even if the actual step-size $\alpha^{(k)}$ might be very small.

This crucially contrasts classical gradient descent techniques, which only use *local* information to determine the step-directions, facing the risk of walking out of the domain D and therefore requiring projection steps after each iteration.

Relation to Inaccurate and Missing Gradient Information. The ability of our Algorithm 1 to deal with only approximate internal linear optimizers as in ApproxLinear() is also related to existing methods that assume that gradient information is only available with noise, or in a stochastic or sampling setting.

For the case of optimizing smooth convex functions, [d'A08] has used a similar measure of error, namely that the linearization given by the "noisy" version $\tilde{d}_x$ of the gradient $\nabla f(x)$ does not differ by more than say ε' when measured over the entire domain D, or formally

$$\left| \langle y - z, \tilde{d}_x \rangle - \langle y - z, \nabla f(x) \rangle \right| \leq \varepsilon' \ , \quad \forall x, y, z \in D \ .$$

This assumption is similar, but stronger than the additive approximation quality that we require in our above setting (we only need that the linearized *optimal values* are within ε'). Also, the algorithm as well as the analysis in [d'A08] are more complicated than the method proposed here, due to the need of projections and proximal operators.

We have discussed the case where gradient information is available only in a stochastic oracle (e.g. such that the gradient is obtained in expectation) in the above Subsection 2.3.6. For an overview of related randomized methods in unconstrained convex optimization, we refer the reader to the recent work by [Nes11], which also applies when the gradient itself is not available and has to be estimated by oracle calls to the function alone.

If gradient information can be constructed in any way such that the linearized problem ApproxLinear() can be solved to the desired additive error, then our above analysis of Algorithm 1 will still hold.

Relation to Mirror Descent, Proximal Methods and Conjugate Functions. Our proposed method is related to *mirror descent* as well as *proximal methods* in convex optimization, but our approach is usually simpler. The mirror descent technique originates from e.g. [BT03, BTN05]. For a brief overview of proximal methods with applications to some of the classes of sparse convex optimization problems as studied here, we refer to [BJMO11, Section 3].

To investigate the connection, we write $f_{\mathrm{lin}|x}(y) := f(x) + \langle y - x, d_x \rangle$ for the linearization given by the (sub)gradient $d_x = \nabla f(x)$ at a fixed point $x \in D$. A variant of mirror descent, see e.g. [BTN05, Haz11] is to find the next iterate y as the point maximizing the Bregman divergence

$$f(y) - f(x) - \langle y - x, d_x \rangle = f(y) - f_{\mathrm{lin}|x}(y) \tag{2.14}$$

relative to the currently fixed old iterate x. This is the same task as maximizing the gap between the function $f(y)$ and its linear approximation at x, or equivalently we evaluate the *conjugate function* $f^*(z) := \sup_{y \in D} \langle y, z \rangle - f(y)$ at $z = d_x$. The definition of the conjugate dual is also known as *Fenchel*

duality, see e.g. [BL06]. In [Nem05], the conjugate function is also called the Legendre transformation.

However in our approach, the inner task $\textsc{ExactLinear}(d_x, D)$ as well as $\textsc{ApproxLinear}(d_x, D, \varepsilon')$ is a simpler linear problem. Namely, we directly minimize the linearization at the current point x, i.e. we maximize

$$-f(x) - \langle y - x, d_x \rangle = -f_{\text{lin}|x}(y) \tag{2.15}$$

and then move towards an approximate maximizer y. In terms of Fenchel duality, this simpler linear problem is the evaluation of the conjugate dual of the *characteristic function* of our domain D, i.e.

$$\mathbf{1}_D^*(z) := \sup_{y \in \mathcal{X}} \langle y, z \rangle - \mathbf{1}_D(y) \ ,$$

where this function is evaluated at the current subgradient $z = d_x$. The *characteristic function* $\mathbf{1}_D : \mathcal{X} \to \overline{\mathbb{R}}$ of a set $D \subseteq \mathcal{X}$ is defined as $\mathbf{1}_D(y) = 0$ for $y \in D$ and $\mathbf{1}_D(y) = \infty$ otherwise.

Compared to our algorithm, mirror descent schemes require a "projection" step in each iteration, sometimes also called the *proximal* or *mirror operator*. This refers to minimizing the linearization plus a strongly convex prox-function that punishes the distance to the starting point. If the squared Euclidean norm is used, the mirror operator corresponds to the standard projection back onto the domain D. Our method uses no such prox-function, and neither is the zero-function a strongly convex one, as would be required for mirror descent to work. It is expected that the computational cost per iteration of our method will in most application cases be lower compared to mirror descent schemes.

For convex optimization over the simplex, which we will study in more details in the following Section 3.1, [BT03] have proposed a mirror descent algorithm, obtaining a convergence of $f(x^{(k)}) - f(x^*) \leq \sqrt{2 \ln n} \frac{L}{\sqrt{k}}$. This however is worse than the convergence of our methods as given by Theorem 2.3. Our convergence is independent of the dimension n, and goes with $\frac{1}{k}$ instead of $\frac{1}{\sqrt{k}}$. Also the earlier paper by [BTMN01] only obtained a convergence of $O\big(\frac{1}{\sqrt{k}}\big)$ for the case of Lipschitz-continuous convex functions over the simplex.

The NERML optimizer by [BTN05] is a variant of mirror descent that memorizes several past linearizations of the objective function, to improve the available knowledge about the current function landscape. It is an open research question if this idea could also help in our setting here, or for stochastic gradient descent schemes [Bot10].

3

Applications to Sparse and Low Rank Approximation

3.1. Sparse Approximation over the Simplex

As a first application of the general optimization scheme from the previous Chapter 2, we will now consider optimization problems defined over the unit simplex, or in other words the non-negative vectors of ℓ_1-norm equal to one. This will serve as a warm-up case before considering ℓ_1-norm regularized problems in the next Section 3.2.

Our approach here will allow us to understand the best achievable sparsity of approximate solutions, as a function of the approximation quality, as already shown by [Cla10].

In particular, we will show that our main Algorithm 1 on page 22 and its analysis do lead to Clarkson's approach [Cla10] for optimizing over the simplex. In this case, it was already known that sparsity $O\left(\frac{1}{\varepsilon}\right)$ can always be achieved by applying Algorithm 1 to the simplex domain, see [Cla10]. We will also show that this is indeed optimal, by providing an asymptotically matching lower bound in Section 3.1.2. Also, our analysis holds even if the linear subproblems are only solved approximately, and allows arbitrary starting points, in contrast to [Cla10].

Having this efficient algorithm giving sparsity $O\left(\frac{1}{\varepsilon}\right)$ is in particularly at-

tractive in view of the computational complexity of *vector cardinality minimization*, which is known to be NP-hard, by a reduction to Exact-Cover, see [Nat95]. Vector cardinality minimization here refers to finding the sparsest vector that is an ε-approximation to some given convex minimization problem. Formally, finding the sparsest x that satisfies $\|Ax - b\|_2 \leq \varepsilon$ for given A, b and ε.

Set-Up. We suppose that a basis has been chosen in the space $\mathcal{X}$, so that we can assume $\mathcal{X} = \mathbb{R}^n$ with the standard inner product $\langle x, y \rangle = x^T y$. Here we consider one special class of the general optimization problems (2.1), namely we optimize over non-negative vectors that sum up to one, that is

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ s.t. & \|x\|_1 = 1 \ , \\ & x \geq 0 \ . \end{array} \tag{3.1}$$

In the following we write $\Delta_n := \{x \in \mathbb{R}^n \mid x \geq 0, \ \|x\|_1 = 1\}$ for the unit simplex in $\mathbb{R}^n$. As the objective function f is now defined over $\mathbb{R}^n$, all subgradients or gradients of f will also be represented by vectors in $\mathbb{R}^n$ in the following.

Note that the alternative case of optimizing under an inequality constraint $\|x\|_1 \leq 1$ instead of $\|x\|_1 = 1$ can easily be reduced to the above form (3.1) by introducing a new "slack" variable. Formally, one uses vectors $\hat{x} = (x_1, \ldots, x_n, x_{n+1}) \in \mathbb{R}^{n+1}$ instead and optimizes the function $\hat{f}(\hat{x}) := f(x_1, \ldots, x_n)$ over the simplex domain $\|\hat{x}\|_1 = 1$, $\hat{x} \geq 0$.

Coresets. The coreset concept was originally introduced in computational geometry by [BHPI02] and [APV02]. For point-set problems, the coreset idea refers to identifying a very small subset (coreset) of the points, such that the solution just on the coreset is guaranteed to be a good approximation to original problem, as we e.g. describe in Chapter 5. Here for general convex optimization, the role of the coreset points is taken by the non-zero coordinates of our sought vector x instead. The coreset size then corresponds to the sparsity of x.

Formally if there exists an ε-approximate solution $x \in D \subseteq \mathbb{R}^n$ to the convex optimization problem (2.1), using only k many non-zero coordinates, then we say that the corresponding coordinates of x form an *ε-coreset* of size k for problem (2.1).

In other words, the following upper and lower bounds of $O\left(\frac{1}{\varepsilon}\right)$ on the sparsity of approximations for problem (3.1) are indeed matching upper

and lower bounds on the coreset size for convex optimization over the simplex, analogous to what we have found in the geometric problem setting of Chapter 5.

3.1.1. Upper Bound: Sparse Greedy on the Simplex

Here we will show how the general algorithm and its analysis from the previous Section 2.3 do in particular lead to Clarkson's approach [Cla10] for minimizing any convex function over the unit simplex. The algorithm follows directly from Algorithm 1, and will have a running time of $O\left(\frac{1}{\varepsilon}\right)$ many gradient evaluations. We will crucially make use of the fact that every linear function attains its minimum at a vertex of the simplex Δ_n. Formally, for any vector $c \in \mathbb{R}^n$, it holds that $\min_{s \in \Delta_n} s^T c = \min_i c_i$. This property is easy to verify in the special case here, but is also a direct consequence of the small Lemma 2.8 which we have proven for general convex hulls, if we accept that the unit simplex is the convex hull of the unit basis vectors. We have obtained that the internal linearized primitive can be solved exactly by choosing

$$\textsc{ExactLinear}\,(c, \Delta_n) := \mathbf{e}_i \quad \text{with } i = \arg\min_i c_i \ .$$

Algorithm 3 Sparse Greedy on the Simplex

Input: Convex function f, target accuracy ε
Output: ε-approximate solution for problem (3.1)
Set $x^{(0)} := \mathbf{e}_1$
for $k = 0 \ldots \infty$ **do**
  Compute $i := \arg\min_i \left(\nabla f(x^{(k)})\right)_i$
  Let $\alpha := \frac{2}{k+2}$
  Update $x^{(k+1)} := x^{(k)} + \alpha(\mathbf{e}_i - x^{(k)})$
end for

Observe that in each iteration, this algorithm only introduces at most one new non-zero coordinate, so that the sparsity of $x^{(k)}$ is always upper bounded by the number of steps k, plus one, given that we start at a vertex. Since Algorithm 3 only moves in coordinate directions, it can be seen as a variant of *coordinate descent*. The convergence result directly follows from the general analysis we gave in the previous Section 2.3.

Theorem 3.1 ([Cla10, Theorem 2.3], Convergence of Sparse Greedy on the Simplex). *For each $k \geq 1$, the iterate $x^{(k)}$ of Algorithm 3 satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{4C_f}{k+2} \ .$$

where $x^* \in \Delta_n$ is an optimal solution to problem (3.1).

Furthermore, for any $\varepsilon > 0$, after at most $2\left\lceil \frac{4C_f}{\varepsilon} \right\rceil + 1 = O\left(\frac{1}{\varepsilon}\right)$ many steps[1], it has an iterate $x^{(k)}$ of sparsity $O\left(\frac{1}{\varepsilon}\right)$, satisfying $g(x^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. □

Duality Gap. We recall from Section 2.2 that the duality gap (2.5) at any point $x \in \Delta_n$ is easily computable from any subgradient, and in our case becomes

$$\begin{aligned} g(x, d_x) &= x^T d_x - \min_i (d_x)_i \ , \qquad \text{and} \\ g(x) &= x^T \nabla f(x) - \min_i (\nabla f(x))_i \ . \end{aligned} \tag{3.2}$$

Here we have again used the observation that linear functions attain their minimum at a vertex of the domain, i.e, $\min_{s \in \Delta_n} s^T c = \min_i c_i$.

Applications. Many practically relevant optimization problems do fit into our setting (3.1) here, allowing the application of Algorithm 3. This includes linear classifiers such as support vector machines (SVMs), see also Chapter 5, as well as kernel learning (finding the best convex combination among a set of base kernels) [BLJ04]. Some other applications that directly fit into our framework are ℓ_2-support vector regression (SVR), AdaBoost [Zha03], mean-variance analysis in portfolio selection [Mar52], the smallest enclosing ball problem [BC07], and estimating mixtures of probability densities [Cla10]. For more applications we refer to [Cla10].

Line-Search for the Best Step-Size. In most applications it will be a straight-forward task to find the optimal step-size $\alpha \in [0, 1]$ in each step instead, as described in Section 2.3.3.

[1] Note that in order for our Theorem 2.5 on the bounded duality gap to apply, the step-size in the second half of the iterations needs to be fixed to $\alpha^{(k)} := \frac{2}{K+2}$, see Section 2.3.2. This remark also applies to the later applications of our general Algorithm 1 in this chapter. We already mentioned above that if line-search is used instead, then no such technicality is necessary, see also [Cla10].

For the special case of polytope distance and SVM problems, the resulting method then exactly corresponds to Gilbert's geometric algorithm [Gil66], which we will study in Chapter 5. Here the wording of "line-search" makes geometric sense in that we need to find the point s on a given line, such that s is closest to the origin.

Away Steps. By performing more work with the currently non-zero coordinates, one can get the sparsity even smaller. More precisely the number of non-zeros can be improved close to $\frac{2C_f}{\varepsilon}$ instead of $2\left\lceil\frac{4C_f}{\varepsilon}\right\rceil$ as given by the above Theorem 3.1. The idea of *away-steps* introduced by [TY07] is to keep the total number of non-zero coordinates (i.e. the coreset size) fixed over all iterations, by removing the smallest non-zero coordinate from x after each adding step. We will discuss this idea for the geometric formulation of coresets in Chapter 5. For more background we refer to [Cla10, Algorithm 9.1].

3.1.2. $\Omega(\frac{1}{\varepsilon})$ Lower Bound on the Sparsity

We will now show that sparsity $O\left(\frac{1}{\varepsilon}\right)$, as obtained by the greedy algorithm we analyzed in the previous section is indeed best possible, by providing a lower bound of $\Omega\left(\frac{1}{\varepsilon}\right)$. In the language of coresets, this means we will provide a matching lower bound on the size of coresets for convex optimization over the simplex. Together with the upper bound, this therefore completely characterizes the trade-off between sparsity and approximation quality for the family of optimization problems of the form (3.1). The same matching sparsity upper and lower bounds will also hold for optimizing over the ℓ_1-ball instead, see Section 3.2.

For the following lower bound construction we consider the differentiable function $f(x) := \|x\|_2^2 = x^T x$. This function has gradient $\nabla f(x) = 2x$. Its curvature constant is $C_f = 2$, which follows directly from the definition (2.7), and the fact that here $f(y) - f(x) - (y-x)^T \nabla f(x) = y^T y - x^T x - (y-x)^T 2x = \|x-y\|_2^2$, so that $C_f = \sup_{x,s\in\Delta_n} \|x-s\|_2^2 = \operatorname{diam}(\Delta_n)^2 = 2$.

The following lemmata show that the sparse greedy algorithm of [Cla10] from Section 3.1.1 is indeed optimal for the approximation quality (primal as well as dual error respectively), giving best possible sparsity, up to a small multiplicative constant.

Lemma 3.2. *For $f(x) := \|x\|_2^2$, and $1 \le k \le n$, it holds that*

$$\min_{\substack{x \in \Delta_n \\ \mathrm{card}(x) \le k}} f(x) = \frac{1}{k}.$$

Proof. We prove the inequality $\min_{x..} f(x) \ge \frac{1}{k}$ by induction on k.

Case $k = 1$ For any unit length vector $x \in \Delta_n$ having just a single non-zero entry, $f(x) = \|x\|_2 = \|x\|_1 = 1$.

Case $k > 1$ For every $x \in \Delta_n$ of sparsity $\mathrm{card}(x) \le k$, we can pick a coordinate i with $x_i \neq 0$, and write $x = (1-\alpha)v + \alpha \mathbf{e}_i$ as the sum of two orthogonal vectors v and a unit basis vector $\mathbf{e}_i$, where $v \in \Delta_n$ of sparsity $\le k-1$, $v_i = 0$, and $\alpha = x_i$. So for every $x \in \Delta_n$ of sparsity $\le k$, we therefore get that

$$\begin{aligned} f(x) = \|x\|_2^2 &= x^T x \\ &= ((1-\alpha)v + \alpha \mathbf{e}_i)^T ((1-\alpha)v + \alpha \mathbf{e}_i) \\ &= (1-\alpha)^2 v^T v + \alpha^2 \\ &\ge (1-\alpha)^2 \tfrac{1}{k-1} + \alpha^2 \\ &\ge \min_{0 \le \beta \le 1} (1-\beta)^2 \tfrac{1}{k-1} + \beta^2 \\ &= \tfrac{1}{k}. \end{aligned}$$

In the first inequality we have applied the induction hypothesis for $v \in \Delta_n$ of sparsity $\le k-1$.

Equality: The function value $f(x) = \frac{1}{k}$ is obtained by setting k of the coordinates of x to $\frac{1}{k}$ each. □

In other words for any vector x of sparsity $\mathrm{card}(x) = k$, the primal error $f(x) - f(x^*)$ is always lower bounded by $\frac{1}{k} - \frac{1}{n}$. For the duality gap $g(x)$, the lower bound is even slightly higher:

Lemma 3.3. *For $f(x) := \|x\|_2^2$, and any $k \in \mathbb{N}$, $k < n$, it holds that*

$$g(x) \ge \frac{2}{k} \qquad \forall x \in \Delta_n \text{ s.t. } \mathrm{card}(x) \le k.$$

Proof. $g(x) = x^T \nabla f(x) - \min_i (\nabla f(x))_i = 2(x^T x - \min_i x_i)$. We now use $\min_i x_i = 0$ because $\mathrm{card}(x) < n$, and that by Lemma 3.2 we have $x^T x = f(x) \ge \frac{1}{k}$. □

Note: We could also consider the function $f(x) := \gamma \|x\|_2^2$ instead, for some $\gamma > 0$. This f has *curvature* constant $C_f = 2\gamma$, and for this scaling, our above lower bound on the duality gap will also scale linearly, giving $\frac{C_f}{k}$.

3.2. Sparse Approximation with Bounded ℓ_1-Norm

In this second application case, will apply the general greedy approach from Section 2.3 in order to understand the best achievable sparsity for convex optimization under bounded ℓ_1-norm, as a function of the approximation quality. Here the situation is indeed extremely similar to the above Section 3.1 of optimizing over the simplex, and the resulting algorithm will again have a running time of $O\left(\frac{1}{\varepsilon}\right)$ many gradient evaluations.

It is known that the vector $\|.\|_1$-norm is the best convex approximation to the sparsity (cardinality) of a vector, that is card(.). More precisely, the function $\|.\|_1$ is the convex envelope of the sparsity, meaning that it is the "largest" convex function that is upper bounded by the sparsity on the convex domain of vectors $\{x \mid \|x\|_\infty \leq 1\}$. This can be seen by observing that $\text{card}(x) \geq \frac{\|x\|_1}{\|x\|_\infty}$, see e.g. [RFP10]. We will discuss the analogous generalization to matrices in Chapter 4, namely using the matrix nuclear norm as the "best" convex approximation of the matrix rank.

Set-Up. Here we consider one special class of the general optimization problem (2.1), namely problems over vectors in $\mathbb{R}^n$ with bounded $\|.\|_1$-norm, that is

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ s.t. \quad & \|x\|_1 \leq 1 \ . \end{aligned} \tag{3.3}$$

We write $\diamondsuit_n := \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 1\}$ for the ℓ_1-ball in $\mathbb{R}^n$. Note that one can simply rescale the function argument to allow for more general constraints $\|x\|_1 \leq t$ for $t > 0$. Again we have $\mathcal{X} = \mathbb{R}^n$ with the standard inner product $\langle x, y \rangle = x^T y$, so that also the subgradients or gradients of f are represented as vectors in $\mathbb{R}^n$.

The Linearized Problem. As already in the simplex case, the subproblem of optimizing a linear function over the ℓ_1-ball is particularly easy to solve, allowing us to provide a fast implementation of the internal primitive procedure ExactLinear $(c, \diamondsuit_n)$.

Namely, it is again easy to see that every linear function attains its minimum/maximum at a vertex of the ball $\diamondsuit_n$, as we have already seen for general convex hulls in our earlier Lemma 2.8, and $\diamondsuit_n = \text{conv}(\{\pm \mathbf{e}_i \mid i \in [n]\})$. Here this crucial observation can also alternatively be interpreted as the known fact that the dual norm to the ℓ_1-norm is in fact the ℓ_∞-norm, see also our earlier Observation 2.2.

Observation 3.4. *For any vector $c \in \mathbb{R}^n$, it holds that*

$$\mathbf{e}_i \cdot \operatorname{sign}(c_i) \in \arg\max_{y \in \diamondsuit_n} y^T c$$

where $i \in [n]$ is an index of a maximal coordinate of c measured in absolute value, or formally $i \in \arg\max_j |c_j|$.

Using this observation for $c = -\nabla f(x)$ in our general Algorithm 1, we therefore directly obtain the following simple method for ℓ_1-regularized convex optimization, as depicted in the Algorithm 4.

Algorithm 4 Sparse Greedy on the ℓ_1-Ball

Input: Convex function f, target accuracy ε
Output: ε-approximate solution for problem (3.3)
Set $x^{(0)} := \mathbf{0}$
for $k = 0 \dots \infty$ **do**
  Compute $i := \arg\max_i \left|\left(\nabla f(x^{(k)})\right)_i\right|$,
  and let $s := \mathbf{e}_i \cdot \operatorname{sign}\left(\left(-\nabla f(x^{(k)})\right)_i\right)$
  Let $\alpha := \frac{2}{k+2}$
  Update $x^{(k+1)} := x^{(k)} + \alpha(s - x^{(k)})$
end for

Observe that in each iteration, this algorithm only introduces at most one new non-zero coordinate, so that the sparsity of $x^{(k)}$ is always upper bounded by the number of steps k. This means that the method is again of coordinate-descent-type, as in the simplex case of the previous Section 3.1.1. Its convergence analysis again directly follows from the general analysis from Section 2.3.

Theorem 3.5 (Convergence of Sparse Greedy on the ℓ_1-Ball). *For each $k \geq 1$, the iterate $x^{(k)}$ of Algorithm 4 satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{4C_f}{k+2} \ .$$

where $x^* \in \diamondsuit_n$ is an optimal solution to problem (3.3).

Furthermore, for any $\varepsilon > 0$, after at most $2\left\lceil \frac{4C_f}{\varepsilon} \right\rceil + 1 = O\left(\frac{1}{\varepsilon}\right)$ many steps, it has an iterate $x^{(k)}$ of sparsity $O\left(\frac{1}{\varepsilon}\right)$, satisfying $g(x^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. □

The Duality Gap, and Duality of the Norms. We recall the definition of the duality gap (2.5) given by the linearization at any point $x \in \diamondsuit_n$, see Section 2.2. Thanks to our Observation 2.2, the computation of the duality gap in the case of the ℓ_1-ball here becomes extremely simple, and is given by the norm that is dual to the ℓ_1-norm, namely the ℓ_∞-norm of the used subgradient, i.e.,

$$g(x, d_x) = \|d_x\|_\infty + x^T d_x, \qquad \text{and}$$
$$g(x) = \|\nabla f(x)\|_\infty + x^T \nabla f(x) \ .$$

Alternatively, the same expression can also be derived directly (without explicitly using duality of norms) by applying the Observation 3.4.

A Lower Bound on the Sparsity. The lower bound of $\Omega\left(\frac{1}{\varepsilon}\right)$ on the sparsity as proved in Section 3.1.2 for the simplex case in fact directly translates to the ℓ_1-ball as well. Instead of choosing the objective function f as the distance to the origin (which is part of the ℓ_1-ball), we consider the optimization problem $\min_{\|x\|_1 \leq 1} f(x) := \|x - r\|_2^2$ with respect to the fixed point $r := (\frac{2}{n}, \ldots, \frac{2}{n}) \in \mathbb{R}^n$. This problem is of the form (3.3), and corresponds to optimizing the Euclidean distance to the point r given by mirroring the origin at the positive facet of the ℓ_1-ball. Here by the "positive facet", we mean the hyperplane defined by the intersection of the boundary of the ℓ_1-ball with the positive orthant, which is exactly the unit simplex. Therefore, the proof for the simplex case from Section 3.1.2 holds analogously for our setting here.

We have thus obtained that sparsity $O\left(\frac{1}{\varepsilon}\right)$ as obtained by the greedy Algorithm 4 is indeed best possible for ℓ_1-regularized optimization problems of the form (3.3).

Using Barycentric Coordinates Instead. Clarkson [Cla10, Theorem 4.2] already observed that Algorithm 3 over the simplex Δ_n can be used to optimize a convex function $f(y)$ over arbitrary convex hulls, by just using barycentric coordinates $y = Ax$, $x \in \Delta_n$, for $A \in \mathbb{R}^{n \times m}$ being the matrix containing all m vertices of the convex domain as its columns. Here however we saw that for the ℓ_1-ball, the steps of the algorithm are even slightly simpler, as well as that the duality gap can be computed instantly from the ℓ_∞-norm of the gradient.

Applications. Our Algorithm 4 applies to arbitrary convex vector optimization problems with an $\|.\|_1$-norm regularization term, giving a guar-

anteed sparsity of $O\left(\frac{1}{\varepsilon}\right)$ for all these applications.

A classical example for problems of this class is given by the important $\|.\|_1$-*regularized least squares* regression approach, which we already discussed in our introductory Section 1.3.1. Formally,

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \mu \|x\|_1$$

for a fixed matrix $A \in \mathbb{R}^{m \times n}$, a vector $b \in \mathbb{R}^m$ and a fixed regularization parameter $\mu > 0$. The same problem is also known as *basis pursuit denoising* in the compressed sensing literature, which we will discuss more precisely in Section 3.2.1. The above formulation is in fact the Lagrangian formulation of the corresponding constrained problem for $\|x\|_1 \leq t$ for some fixed parameter t corresponding to μ, see also Section 1.3.2. This equivalent formulation is also known as the *Lasso* problem [Tib96] which is

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|Ax - b\|_2^2 \\ \text{s.t.} \quad & \|x\|_1 \leq t \ . \end{aligned}$$

The above formulation is exactly a problem of our above form (3.3), namely

$$\min_{\hat{x} \in \diamondsuit_n} \|tA\hat{x} - b\|_2^2 \ ,$$

if we rescale the argument $x =: t\hat{x}$ so that $\|\hat{x}\|_1 \leq 1$.

Another important application for our result is *logistic regression* with $\|.\|_1$-norm regularization, see e.g. [KKB07], which is also a convex optimization problem [Ren05]. The reduction to an ℓ_1-problem of our form (3.3) works exactly the same way as described here.

Related Work. As we mentioned above, the optimization problem (3.3) — if f is the squared error of a linear function — is very well studied as the *Lasso* approach, see e.g. [Tib96] and the references therein. For general objective functions f of bounded curvature, the above interesting trade-off between sparsity and the approximation quality was already investigated by [SSSZ10], and also by our earlier paper [GJ09] (see also Chapter 5) for the analogous case of optimizing over the simplex. [SSSZ10, Theorem 2.4] shows a sparse convergence analogous to our above Theorem 3.5, for the "forward greedy selection" algorithm on problem (3.3), but only for the case that f is differentiable.

3.2.1. Relation to Matching Pursuit and Basis Pursuit in Compressed Sensing

Both our sparse greedy Algorithm 3 for optimizing over the simplex and also Algorithm 4 for general ℓ_1-problems are very similar to the technique of *matching pursuit*, which is one of the most popular techniques in sparse recovery in the vector case [Tro04].

Suppose we want to recover a sparse signal vector $x \in \mathbb{R}^n$ from a noisy measurement vector $Ax = y \in \mathbb{R}^m$. For a given dictionary matrix $A \in \mathbb{R}^{m \times n}$, matching pursuit iteratively chooses the dictionary element $A_i \in \mathbb{R}^m$ that has the highest inner product with the current residual, and therefore reduces the representation error $f(x) = \|Ax - y\|_2^2$ by the largest amount. This choice of coordinate $i = \arg\max_j A_j^T(Ax - y)$ exactly corresponds[2] to the choice of $i := \arg\min_j \left(\nabla f(x^{(k)})\right)_j$ in Algorithm 3.

Another variant of matching pursuit, called orthogonal matching pursuit (OMP) [Tro04, TG07], includes an extra orthogonalization step, and is closer related to the coreset algorithms that optimize over the all existing set of non-zero coordinates before adding a new one, see e.g. [Cla10, Algorithm 8.2], or the analogous "fully corrective" variant of [SSSZ10]. If $y = Ax$, with x sparse and the columns of A sufficiently incoherent, then OMP recovers the sparsest representation for the given y [Tro04].

The paper [Zha11] recently proposed another algorithm that generalizes OMP, comes with a guarantee on correct sparse recovery, and also corresponds to "completely optimize within each coreset". The method uses the same choice of the new coordinate $i := \arg\max_j \left|\left(\nabla f(x^{(k)})\right)_j\right|$ as in our Algorithm 4. However the analysis of [Zha11] requires the not only bounded curvature as in our case, but also needs strong convexity of the objective function (which then also appears as a multiplicative factor in the number of iterations needed). Our Algorithm 4 as well as the earlier method by [Zha03] are simpler to implement, and have a lower complexity per iteration, as we do not need to optimize over several currently non-zero coordinates, but only change one coordinate by a fixed amount in each iteration.

Our Algorithm 4 for general ℓ_1-regularized problems also applies to solving the so called *basis pursuit* problem [CDS98, FNW07] and [BV04, Section 6.5.4], which is $\min_{x \in \mathbb{R}^n} \|x\|_1$ s.t. $Ax = y$. Note that this is in fact just the constrained variant of the corresponding "robust" ℓ_1-regularized

[2] The objective function $f(x) := \|Ax - y\|_2^2$ can be written as $f(x) = (Ax - y)^T(Ax - y) = x^T A^T Ax - 2y^T Ax - y^T y$, so its gradient is $\nabla f(x) = 2A^T Ax - 2A^T y = 2A^T(Ax - y) \in \mathbb{R}^n$.

least squares regression problem

$$\min_{x \in \mathbb{R}^n} \|Ax - y\|_2^2 + \mu \|x\|_1 \ ,$$

which is the equivalent trade-off variant of our problem of the form (3.3). [FNW07] propose a traditional gradient descent technique for solving the above least squares problem, but do not give a convergence analysis.

We will discuss solution path algorithms with approximation guarantees for this problem (obtaining solutions for all values of the tradeoff parameter μ) in Chapter 6.

3.3. Optimization with Bounded ℓ_∞-Norm

Applying our above general optimization framework for the special case of the domain being the $\|.\|_\infty$-norm unit ball, we again obtain a very simple greedy algorithm. The running time will again correspond to $O\left(\frac{1}{\varepsilon}\right)$ many gradient evaluations. Formally, we consider problems of the form

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ s.t. \quad & \|x\|_\infty \leq 1 \ . \end{aligned} \tag{3.4}$$

We denote the feasible set, i.e. the $\|.\|_\infty$-norm unit ball, by $\square_n := \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$. For this set, it will again be very simple to implement the internal primitive operation of optimizing a linear function over the same domain. The following crucial observation allows us to implement ExactLinear $(c, \square_n)$ in a very simple way. This can also alternatively be interpreted as the known fact that the dual-norm to the ℓ_∞-norm is the ℓ_1-norm, which also explains why the greedy algorithm we will obtain here is very similar to the ℓ_1-version from the previous Section 3.2.

Observation 3.6. *For any vector $c \in \mathbb{R}^n$, it holds that*

$$\mathbf{s}^c \in \arg\max_{y \in \square_n} y^T c$$

where $\mathbf{s}^c \in \mathbb{R}^n$ is the sign-vector of c, defined by the sign of each individual coordinate, i.e. $(\mathbf{s}^c)_i = \operatorname{sign}(c_i) \in \{-1, 1\}$.

Using this observation for $c = -d_x$ in our general Algorithm 1, we directly obtain the following simple method for optimization over a box-domain $\square_n$, as depicted in Algorithm 5.

The convergence analysis again directly follows from the general analysis from Section 2.3.

Algorithm 5 Sparse Greedy on the Cube

Input: Convex function f, target accuracy ε
Output: ε-approximate solution for problem (3.4)
Set $x^{(0)} := \mathbf{0}$
for $k = 0 \dots \infty$ **do**
  Compute the sign-vector $\mathbf{s}$ of $\nabla f(x^{(k)})$, such that
    $\mathbf{s}_i = \operatorname{sign}\left(\left(-\nabla f(x^{(k)})\right)_i\right), \quad i = 1..n$
  Let $\alpha := \frac{2}{k+2}$
  Update $x^{(k+1)} := x^{(k)} + \alpha(\mathbf{s} - x^{(k)})$
end for

Theorem 3.7. *For each $k \geq 1$, the iterate $x^{(k)}$ of Algorithm 5 satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{4C_f}{k+2} \ .$$

where $x^* \in \square_n$ is an optimal solution to problem (3.4).

Furthermore, for any $\varepsilon > 0$, after at most $2\left\lceil \frac{4C_f}{\varepsilon} \right\rceil + 1 = O\left(\frac{1}{\varepsilon}\right)$ many steps, it has an iterate $x^{(k)}$ with $g(x^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. □

The Duality Gap, and Duality of the Norms. We recall the definition of the duality gap (2.5) given by the linearization at any point $x \in \square_n$, see Section 2.2. Thanks to our Observation 2.2, the computation of the duality gap in the case of the ℓ_∞-ball here becomes extremely simple, and is given by the norm that is dual to the ℓ_∞-norm, namely the ℓ_1-norm of the used subgradient, i.e.,

$$g(x, d_x) = \|d_x\|_1 + x^T d_x, \qquad \text{and}$$
$$g(x) = \|\nabla f(x)\|_1 + x^T \nabla f(x) \ .$$

Alternatively, the same expression can also be derived directly (without explicitly using duality of norms) by applying the Observation 3.6.

Sparsity and Compact Representations. The analogue of "sparsity" as in Sections 3.1 and 3.2 in the context of our Algorithm 5 means that we can describe the obtained approximate solution x as a convex combination of few (i.e. $O(\frac{1}{\varepsilon})$ many) cube vertices. This does not imply that x has few

non-zero coordinates, but that we have a compact representation given by only $O(\frac{1}{\varepsilon})$ many binary n-vectors indicating the corresponding cube vertices, of which x is a convex combination.

Applications. Any convex problem under coordinate-wise upper and lower constraints can be transformed to the form (3.4) by re-scaling the optimization argument. A specific interesting application was given by [MR11], who have demonstrated that integer linear programs can be relaxed to convex problems of the above form, such that the solutions coincide with high probability under some mild additional assumptions.

Using Barycentric Coordinates Instead. Clarkson [Cla10, Theorem 4.2] already observed that Algorithm 3 over the simplex Δ_n can be used to optimize a convex function $f(y)$ over arbitrary convex hulls, by just using barycentric coordinates $y = Ax,\ x \in \Delta_n$, for $A \in \mathbb{R}^{n \times m}$ being the matrix containing all m vertices of the convex domain as its columns. Here however we saw that for the unit box, the steps of the algorithm are much simpler, as well as that the duality gap can be computed instantly, without having to explicitly deal with the exponentially many vertices (here $m = 2^n$) of the cube.

3.4. Semidefinite Optimization with Bounded Trace

We will now apply the greedy approach from the previous Section 2.3 to semidefinite optimization problems, for the case of bounded trace. The main paradigm in this section will be to understand the best achievable low-rank property of approximate solutions as a function of the approximation quality.

In particular, we will show that our general Algorithm 1 and its analysis do lead to Hazan's method for convex semidefinite optimization with bounded trace, as given by [Haz08]. Hazan's algorithm can also be used as a simple solver for general SDPs. [Haz08] has already shown that guaranteed ε-approximations of rank $O\left(\frac{1}{\varepsilon}\right)$ can always be found. Here we will also show that this is indeed optimal, by providing an asymptotically matching lower bound in Section 3.4.4. Furthermore, we fix some problems in the original analysis of [Haz08], and require only a weaker approximation quality for the internal linearized primitive problems. We also propose two improvement variants for the method in Section 3.4.3.

Later in Chapter 4, we will discuss the application of these algorithms for nuclear norm and max-norm optimization problems, which have many important applications in practice, such as dimensionality reduction, low-rank recovery as well as matrix completion and factorizations.

We now consider convex optimization problems of the form (2.1) over the space $\mathcal{X} = \mathbb{S}^{n\times n}$ of symmetric matrices, equipped with the standard Frobenius inner product $\langle X, Y\rangle = X \bullet Y$. It is left to the choice of the reader to identify the symmetric matrices either with $\mathbb{R}^{n^2}$ and consider functions with $f(X) = f(X^T)$, or only "using" the variables in the upper right (or lower left) triangle, corresponding to $\mathbb{R}^{n(n+1)/2}$. In any case, the subgradients or gradients of our objective function f need to be available in the same representation (same choice of basis for the vector space $\mathcal{X}$).

Formally, we consider the following special case of the general optimization problems (2.1), i.e.,

$$\begin{array}{ll} \underset{X\in\mathbb{S}^{n\times n}}{\text{minimize}} & f(X) \\ s.t. & \operatorname{Tr}(X) = 1\ , \\ & X \succeq 0 \end{array} \tag{3.5}$$

We will write $\mathcal{S} := \{X \in \mathbb{S}^{n\times n} \mid X \succeq 0,\ \operatorname{Tr}(X) = 1\}$ for the feasible set, that is the PSD matrices of unit trace. The set $\mathcal{S}$ is sometimes called the *Spectahedron*, and can be seen as a natural generalization of the unit simplex to symmetric matrices. By the Cholesky factorization, it can be seen that the Spectahedron is the convex hull of all rank-1 matrices of unit trace (i.e. the matrices of the form vv^T for a unit vector $v \in \mathbb{R}^n$, $\|v\|_2 = 1$).

3.4.1. Low-Rank Semidefinite Optimization with Bounded Trace: The $O(\frac{1}{\varepsilon})$ Algorithm by Hazan

Applying our general greedy Algorithm 1 that we studied in Section 2.3 to the above semidefinite optimization problem, we directly obtain the following Algorithm 6, which is Hazan's method [Haz08, GM11].

Note that this is now a first application of Algorithm 1 where the internal linearized problem ApproxLinear() is not trivial to solve, contrasting the applications for vector optimization problems we studied above. The algorithm here obtains low-rank solutions (sum of rank-1 matrices) to any convex optimization problem of the form (3.5). More precisely, it guarantees ε-small duality gap after at most $O\left(\frac{1}{\varepsilon}\right)$ iterations, where each iteration only involves the calculation of a single approximate eigenvector of a ma-

trix $M \in \mathbb{S}^{n\times n}$. We will see that in practice for example Lanczos' or the power method can be used as the internal optimizer ApproxLinear().

Algorithm 6 Hazan's Algorithm / Sparse Greedy for Bounded Trace

Input: Convex function f with curvature C_f, target accuracy ε
Output: ε-approximate solution for problem (3.5)
Set $X^{(0)} := vv^T$ for an arbitrary unit length vector $v \in \mathbb{R}^n$.
for $k = 0 \dots \infty$ **do**
  Let $\alpha := \frac{2}{k+2}$
  Compute $v := v^{(k)} = \text{ApproxEV}\left(\nabla f(X^{(k)}), \alpha C_f\right)$
  Update $X^{(k+1)} := X^{(k)} + \alpha(vv^T - X^{(k)})$
end for

Here ApproxEV(A, ε') is a subroutine that delivers an approximate smallest eigenvector (the eigenvector corresponding to the smallest eigenvalue) to a matrix A with the desired accuracy $\varepsilon' > 0$. More precisely, it must return a unit length vector v such that $v^T Av \leq \lambda_{\min}(A) + \varepsilon'$. Note that as our convex function f takes a symmetric matrix X as an argument, its gradients $\nabla f(X)$ are given as symmetric matrices as well.

If we want to understand this proposed Algorithm 6 as an instance of the general convex optimization Algorithm 1, we just need to explain why the largest eigenvector should indeed be a solution to the internal linearized problem ApproxLinear(), as required in Algorithm 1. Formally, we have to show that $v := \text{ApproxEV}(A, \varepsilon')$ does approximate the linearized problem, that is

$$vv^T \bullet A \leq \min_{Y \in \mathcal{S}} Y \bullet A + \varepsilon'$$

for the choice of $v := \text{ApproxEV}(A, \varepsilon')$, and any matrix $A \in \mathbb{S}^{n\times n}$.

This fact is formalized in Lemma 3.8 below, and will be the crucial property enabling the fast implementation of Algorithm 6.

Alternatively, if exact eigenvector computations are available, we can also implement the exact variant of Algorithm 1 using ExactLinear(), thereby halving the total number of iterations.

Observe that an approximate eigenvector here is significantly easier to compute than a projection onto the feasible set $\mathcal{S}$. If we were to find the $\|.\|_{Fro}$-closest PSD matrix to a given symmetric matrix A, we would have to compute a complete eigenvector decomposition of A, and only keeping those corresponding to positive eigenvalues, which is computationally

expensive. By contrast, a single approximate smallest eigenvector computation as in ApproxEV(A, ε') can be done in near linear time in the number of non-zero entries of A. We will discuss the implementation of ApproxEV(A, ε') in more detail further below.

Sparsity becomes Low Rank. As the rank-1 matrices are indeed the "vertices" of the domain $\mathcal{S}$ as shown in Lemma 3.8 below, our Algorithm 6 can be therefore seen as a matrix generalization of the sparse greedy approximation algorithm of [Cla10] for vectors in the unit simplex, see Section 3.1, which has seen many successful applications. Here *sparsity* just gets replaced by *low rank*. By the analysis of the general algorithm in Theorem 2.3, we already know that we obtain ε-approximate solutions for any convex optimization problem (3.5) over the spectahedron $\mathcal{S}$. Because each iterate $X^{(k)}$ is represented as a sum (convex combination) of k many rank-1 matrices vv^T, it follows that $X^{(k)}$ is of rank at most k. Therefore, the resulting ε-approximations are of low rank, i.e. rank $O\left(\frac{1}{\varepsilon}\right)$.

For large-scale applications where $\frac{1}{\varepsilon} \ll n$, the representation of $X^{(k)}$ as a sum of rank-1 matrices is much more efficient than storing an entire matrix $X^{(k)} \in \mathbb{S}^{n \times n}$. Later in Section 4.5 (or see also [JS10]) we will demonstrate that Algorithm 6 can readily be applied to practical problems for $n \geq 10^6$ on an ordinary computer, well exceeding the possibilities of interior point methods.

[Haz08] already observed that the same Algorithm 6 with a well-crafted function f can also be used to approximately solve arbitrary SDPs with bounded trace, which we will briefly explain in Section 3.4.2.

Linearization, the Duality Gap, and Duality of the Norms. Here we will prove that the general duality gap (2.5) can be calculated very efficiently for the domain being the spectahedron $\mathcal{S}$. From the following Lemma 3.8, we obtain that

$$\begin{aligned} g(X) =& X \bullet \nabla f(X) + \lambda_{\max}(-\nabla f(X)) \\ =& X \bullet \nabla f(X) - \lambda_{\min}(\nabla f(X)) \ . \end{aligned} \tag{3.6}$$

As predicted by our Observation 2.2 on formulating the duality gap, we have again obtained the *dual norm* to the norm that determines the domain D. It can be seen that over the space of symmetric matrices, the dual norm of the matrix trace-norm (also known as the nuclear norm) is given by the spectral norm, i.e. the largest eigenvalue. To see this, we refer the reader to the later Section 4.2 on the properties of the nuclear norm and its dual characterization.

The following Lemma 3.8 shows that any linear function attains its minimum and maximum at a "vertex" of the Spectahedron $\mathcal{S}$, as we have already proved for the case of general convex hulls in Lemma 2.8.

Lemma 3.8. *The spectahedron is the convex hull of the rank-1 matrices,*

$$\mathcal{S} = \operatorname{conv}(\{vv^T \mid v \in \mathbb{R}^n, \|v\|_2 = 1\}) .$$

Furthermore, for any symmetric matrix $A \in \mathbb{S}^{n\times n}$, it holds that

$$\max_{X\in\mathcal{S}} A \bullet X = \lambda_{\max}(A) .$$

Proof. Clearly, it holds that $vv^T \in \mathcal{S}$ for any unit length vector $v \in \mathbb{R}^n$, as $\operatorname{Tr}(vv^T) = \|v\|_2^2$. To prove the other inclusion, we consider an arbitrary matrix $X \in \mathcal{S}$, and let $X = U^TU$ be its Cholesky factorization. We let α_i be the squared norms of the rows of U, and let u_i be the row vectors of U, scaled to unit length. From the observation $1 = \operatorname{Tr}(X) = \operatorname{Tr}(U^TU) = \operatorname{Tr}(UU^T) = \sum_i \alpha_i$ it follows that any $X \in \mathcal{S}$ can be written as a convex combination of at most n many rank-1 matrices $X = \sum_{i=1}^n \alpha_i u_i u_i^T$ with unit vectors $u_i \in \mathbb{R}^n$, proving the first part of the claim. Furthermore, this implies that we can write

$$\max_{X\in\mathcal{S}} A \bullet X = \max_{u_i,\alpha_i} A \bullet \sum_{i=1}^n \alpha_i u_i u_i^T = \max_{u_i,\alpha_i} \sum_{i=1}^n \alpha_i (A \bullet u_i u_i^T),$$

where the maximization $\max_{u_i,\alpha_i}$ is taken over unit vectors $u_i \in \mathbb{R}^n$, $\|u_i\| = 1$, for $1 \le i \le n$, and real coefficients $\alpha_i \ge 0$, with $\sum_{i=1}^n \alpha_i = 1$. Therefore

$$\begin{aligned}\max_{X\in\mathcal{S}} A \bullet X &= \max_{u_i,\alpha_i} \sum_{i=1}^n \alpha_i (A \bullet u_i u_i^T) \\ &= \max_{v\in\mathbb{R}^n, \|v\|=1} A \bullet vv^T \\ &= \max_{v\in\mathbb{R}^n, \|v\|=1} v^T A v \\ &= \lambda_{\max}(A) ,\end{aligned}$$

where the last equality is the variational characterization of the largest eigenvalue. □

Curvature. We know that the constant in the actual running time for a given convex function $f : \mathbb{S}^{d\times d} \to \mathbb{R}$ is given by the *curvature constant* C_f as given in (2.7), which for the domain $\mathcal{S}$ becomes

$$C_f := \sup_{\substack{X,V\in\mathcal{S},\, \alpha\in[0,1],\\ Y=X+\alpha(V-X)}} \tfrac{1}{\alpha^2}\big(f(Y) - f(X) + (Y-X) \bullet \nabla f(X)\big) . \tag{3.7}$$

Convergence. We can now see the convergence analysis for Algorithm 6 following directly as a corollary of our simple analysis of the general framework in Section 2.3. The following theorem proves that $O\left(\frac{1}{\varepsilon}\right)$ many iterations are sufficient to obtain primal error $\leq \varepsilon$. This result was already known in [Haz08, Theorem 1], or [GM11, Chapter 5] where some corrections to the original paper were made.

Theorem 3.9. *For each $k \geq 1$, the iterate $X^{(k)}$ of Algorithm 6 satisfies*

$$f(X^{(k)}) - f(X^*) \leq \frac{8C_f}{k+2} \ .$$

where $X^* \in \mathcal{S}$ is an optimal solution to problem (3.5).

Furthermore, for any $\varepsilon > 0$, after at most $2\left\lceil \frac{8C_f}{\varepsilon} \right\rceil + 1 = O\left(\frac{1}{\varepsilon}\right)$ many steps, it has an iterate $X^{(k)}$ of rank $O\left(\frac{1}{\varepsilon}\right)$, satisfying $g(X^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. □

Approximating the Largest Eigenvector. Approximating the smallest eigenvector of a symmetric matrix $\nabla f(X)$ (which is the largest eigenvector of $-\nabla f(X)$) is a well-studied problem in the literature. We will see in the following that the internal procedure ApproxEV(M, ε'), can be performed in *near-linear* time, when measured in the number of non-zero entries of the gradient matrix $\nabla f(X)$. This will follow from the analysis of [KW92] for the power method or Lanczos' algorithm, both with a random start vector. A similar statement has been used in [AHK05, Lemma 2].

Theorem 3.10. *Let $M \in \mathbb{S}^{n \times n}$ be a positive semidefinite matrix. Then with high probability, both*

i) $O\left(\frac{\log(n)}{\gamma}\right)$ iterations of the power method, or

ii) $O\left(\frac{\log(n)}{\sqrt{\gamma}}\right)$ iterations of Lanczos' algorithm

will produce a unit vector x such that $\frac{x^T M x}{\lambda_1(M)} \geq 1 - \gamma$.

Proof. The statement for the power method follows from [KW92, Theorem 3.1(a)], and for Lanczos' algorithm by [KW92, Theorem 3.2(a)]. □

The only remaining obstacle to use this result for our internal procedure ApproxEV(M, ε') is that our gradient matrix $M = -\nabla f(X)$ is usually not PSD. However, this can easily be fixed by adding a large enough constant t

to the diagonal, i.e. $\hat{M} := M + t\mathbf{I}$, or in other words shifting the spectrum of M so that the eigenvalues satisfy $\lambda_i(\hat{M}) = \lambda_i(M) + t \geq 0 \; \forall i$. The choice of $t = -\lambda_{\min}(M)$ is good enough for this to hold.

Now by setting $\gamma := \frac{\varepsilon'}{L} \leq \frac{\varepsilon'}{\lambda_{\max}(\hat{M})}$ for some upper bound $L \geq \lambda_{\max}(\hat{M}) = \lambda_{\max}(M) - \lambda_{\min}(M)$, this implies that our internal procedure $\text{ApproxEV}(M, \varepsilon')$ can be implemented by performing $O\left(\frac{\log(n)\sqrt{L}}{\sqrt{\varepsilon'}}\right)$ many Lanczos steps (that is matrix-vector multiplications). Note that a simple choice for L is given by the spectral norm of M, since $2\|M\|_{spec} = 2\max_i \lambda_i(M) \geq \lambda_{\max}(M) - \lambda_{\min}(M)$. We state the implication for our algorithm in the following corollary.

Theorem 3.11. *For $M \in \mathbb{S}^{n\times n}$, and $\varepsilon' > 0$, the procedure ApproxEV(M, ε') requires a total of $O\left(N_f \frac{\log(n)\sqrt{L}}{\sqrt{\varepsilon'}}\right)$ many arithmetic operations, with high probability, by using Lanczos' algorithm.*

Here N_f is the number of non-zero entries in M, which in the setting of Algorithm 6 is the gradient matrix $-\nabla f(X)$. We have also assumed that the spectral norm of M is bounded by L.

Since we already know the number of necessary "outer" iterations of Algorithm 6, by Theorem 3.9, we conclude with the following analysis of the total running time. Here we again use that the required internal accuracy is given by $\varepsilon' = \alpha C_f \leq \varepsilon C_f$.

Corollary 3.12. *When using Lanczos' algorithm for the approximate eigenvector procedure ApproxEV$(.,.)$, then Algorithm 6 provides an ε-approximate solution in $O\left(\frac{1}{\varepsilon}\right)$ iterations, requiring a total of $\tilde{O}\left(\frac{N_f}{\varepsilon^{1.5}}\right)$ arithmetic operations (with high probability).*

Here the notation $\tilde{O}(.)$ suppresses the logarithmic factor in n. This corollary improves the original analysis of [Haz08] by a factor of $\frac{1}{\sqrt{\varepsilon}}$, since [Haz08, Algorithm 1] as well as the proof of [Haz08, Theorem 1] used an internal accuracy bound of $\varepsilon' = O\left(\frac{1}{k^2}\right)$ instead of the sufficient choice of $\varepsilon' = O\left(\frac{1}{k}\right)$ as in our general analysis here.

Representation of the Estimate X in the Algorithm. The above result on the total running time assumes the following: After having obtained an approximate eigenvector v, the rank-1 update $X^{(k+1)} := (1-\alpha)X^{(k)} + \alpha vv^T$ can be performed efficiently, or more precisely in time N_f. In the worst case, when a fully dense matrix X is needed, this update cost is $N_f = n^2$. However, there are many interesting applications where the function f

depends only on a small fraction of the entries of X, so that $N_f \ll n^2$. Here, a prominent example is matrix completion for recommender systems. In this case, only those N_f many entries of X will be stored and affected by the rank-1 update, see also our Section 4.5.

An alternative representation of X consists of the low-rank factorization, given by the v-vectors of each of the $O\left(\frac{1}{\varepsilon}\right)$ many update steps, using a smaller memory of size $O\left(\frac{n}{\varepsilon}\right)$. However, computing the gradient $\nabla f(X)$ from this representation of X might require more time then.

3.4.2. Solving Arbitrary SDPs

In [Haz08] it was established that Algorithm 6 can also be used to approximately solve arbitrary semidefinite programs (SDPs) in feasibility form, i.e.,

$$\begin{aligned} \text{find } X \text{ s.t.} \quad & A_i \bullet X \leq b_i \quad i = 1..m \\ & X \succeq 0 \ . \end{aligned} \tag{3.8}$$

Also every classical SDP with a linear objective function

$$\begin{aligned} \underset{X}{\text{maximize}} \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X \leq b_i \quad i = 1..m' \\ & X \succeq 0 \ . \end{aligned} \tag{3.9}$$

can be turned into a feasibility SDP (3.8) by "guessing" the optimal value $C \bullet X$ by binary search [AHK05, Haz08].

Here we will therefore assume that we are given a feasibility SDP of the form (3.8) by its constraints $A_i \bullet X \leq b_i$, which we want to solve for X. We can represent the constraints of (3.8) in a smooth optimization objective instead, using the *soft-max* function

$$f(X) := \frac{1}{\sigma} \log \left(\sum_{i=1}^{m} e^{\sigma(A_i \bullet X - b_i)} \right) \ . \tag{3.10}$$

Suppose that the original SDP was feasible, then after $O\left(\frac{1}{\varepsilon}\right)$ many iterations of Algorithm 6, for a suitable choice of σ, we have obtained X such that $f(X) \leq \varepsilon$, which implies that all constraints are violated by at most ε. This means that $A_i \bullet X \leq b_i + \varepsilon$, or in other words we say that X is ε-feasible [Haz08, GM11]. It turns out the best choice for the parameter σ is $\frac{\log m}{\varepsilon}$, and the curvature constant $C_f(\sigma)$ for this function is bounded by $\sigma \cdot \max_i \lambda_{\max}(A_i)^2$. The total number of necessary approximate eigenvector computations is therefore in $O\left(\frac{\log m}{\varepsilon^2}\right)$. In fact, Algorithm 6 when

applied to the function (3.10) is very similar to the multiplicative weights method [AHK05]. Note that the soft-max function (3.10) is convex in X, see also [Ren05]. For a slightly more detailed exhibition of this approach of using Algorithm 6 to approximately solving SDPs, we refer the reader to the book of [GM11].

Note that this technique of introducing the soft-max function is closely related to smoothing techniques in the optimization literature [Nem04, BB09], where the soft-max function is introduced to get a smooth approximation to the largest eigenvalue function. The transformation to a smooth saddle-point problem suggested by [BB09] is more complicated than the simple notion of ε-feasibility suggested here, and will lead to a comparable computational complexity in total.

3.4.3. Two Improved Variants of Algorithm 6

Choosing the Optimal α by Line-Search. As we mentioned already for the general algorithm for convex optimization in Section 2.3, the optimal α in Algorithm 6, i.e. the $\alpha \in [0,1]$ of best improvement in the objective function f can be found by line-search.

In particular for matrix completion problems, which we will discuss in more details in Section 4.5, the widely used squared error is easy to analyze in this respect: If the optimization function is given by $f(X) = \frac{1}{2}\sum_{ij\in P}(X_{ij} - Y_{ij})^2$, where P is the set of observed positions of the matrix Y, then the optimality condition (2.10) from Section 2.3.3 is equivalent to

$$\alpha = \frac{\sum_{ij\in P}(X_{ij} - y_{ij})(X_{ij} - v_i v_j)}{\sum_{ij\in P}(X_{ij} - v_i v_j)^2} \ . \tag{3.11}$$

Here $X = X^{(k)}$, and v is the approximate eigenvector $v^{(k)}$ used in step k of Algorithm 6. The above expression is computable very efficiently compared to the eigenvector approximation task.

Immediate Feedback in the Power Method. As a second improvement, we propose a heuristic to speed up the eigenvector computation, i.e. the internal procedure ApproxEV $(\nabla f(X), \varepsilon')$. Instead of multiplying the current candidate vector v_k with the gradient matrix $\nabla f(X)$ in each power iteration, we multiply with $\frac{1}{2}\left(\nabla f(X) + \nabla f(\overline{X})\right)$, or in other words the average between the current gradient and the gradient at the new candidate location $\overline{X} = \left(1 - \frac{1}{k}\right)X^{(k)} + \frac{1}{k}v^{(k)}v^{(k)^T}$. Therefore, we immediately take into account the effect of the new feature vector $v^{(k)}$. This heuristic (which

unfortunately does not fall into our current theoretical guarantee) is inspired by stochastic gradient descent as in Simon Funk's method, which we will describe in Section 4.5.4. In practical experiments, this proposed slight modification will result in a significant speed-up of Algorithm 6, as we will observe e.g. for matrix completion problems in Section 4.5.

3.4.4. $\Omega(\frac{1}{\varepsilon})$ Lower Bound on the Rank

Analogous to the vector case discussed in Section 3.1.2, we can also show that the rank of $O(\frac{1}{\varepsilon})$, as obtained by the greedy Algorithm 6 is indeed optimal, by providing a lower bound of $\Omega(\frac{1}{\varepsilon})$. In other words we can now exactly characterize the trade-off between rank and approximation quality, for convex optimization over the spectahedron.

For the lower bound construction, we consider the convex function $f(X) := \|X\|_{Fro}^2 = X \bullet X$ over the symmetric matrices $\mathbb{S}^{n\times n}$. This function has gradient $\nabla f(X) = 2X$. We will later see that its curvature constant is $C_f = 2$.

The following lemmata show that the above sparse SDP Algorithm 6 is optimal for the approximation quality (primal as well as dual error respectively), giving lowest possible rank, up to a small multiplicative constant.

Lemma 3.13. *For $f(X) := \|X\|_{Fro}^2$, and $1 \leq k \leq n$, it holds that*

$$\min_{\substack{X\in\mathcal{S} \\ \mathrm{Rk}(X)\leq k}} f(X) = \frac{1}{k}.$$

We will see that this claim can be reduced to the analogous Lemma 3.2 for the vector case, by the standard technique of diagonalizing a symmetric matrix. (This idea was suggested by Elad Hazan). Alternatively, an explicit (but slightly longer) proof without requiring the spectral theorem can be obtained by using the Cholesky-decomposition together with induction on k.

Proof. We observe that the objective function $\|.\|_{Fro}^2$, the trace, as well as the property of being positive semidefinite, are all invariant under orthogonal transformations (or in other words under the choice of basis).

By the standard spectral theorem, for any symmetric matrix X of $\mathrm{Rk}(X) \leq k$, there exists an orthogonal transformation mapping X to a *diagonal* matrix X' with at most k non-zero entries on the diagonal (being eigenvalues of X by the way). For diagonal matrices, the $\|.\|_{Fro}$ matrix norm coincides with the $\|.\|_2$ vector norm of the diagonal of the matrix. Finally by

applying the vector case Lemma 3.2 for the diagonal of X', we obtain that $f(X) = f(X') \geq \frac{1}{k}$.

To see that the minimum can indeed be attained, one again chooses the "uniform" example $X := \frac{1}{k}\mathbf{I}_k \in \mathcal{S}$, being the matrix consisting of k non-zero entries (of $\frac{1}{k}$ each) on the diagonal. This gives $f(X) = \frac{1}{k}$. □

Recall from Section 3.4.1 that for convex problems of the form (3.5) over the Spectahedron, the *duality gap* is the non-negative value $g(X) := f(X) - \omega(X) = X \bullet \nabla f(X) - \lambda_{\min}(\nabla f(X))$. Also, by weak duality as given in Lemma 2.1, this value is always an upper bound for the primal error, that is $f(X) - f(X^*) \leq g(X)\ \forall X$.

Lemma 3.14. *For $f(X) := \|X\|_{Fro}^2$, and any $k \in \mathbb{N}$, $k < n$, it holds that*

$$g(X) \geq \frac{1}{k} \qquad \forall X \in \mathcal{S} \text{ s.t. } \mathrm{Rk}(X) \leq k.$$

Proof. $g(X) = \lambda_{\max}(-\nabla f(X)) + X \bullet \nabla f(X) = -\lambda_{\min}(X) + X \bullet 2X$. We now use that $\lambda_{\min}(X) = 0$ for all symmetric PSD matrices X that are not of full rank n, and that by Lemma 3.13, we have $X \bullet X = \mathrm{Tr}(X^T X) = f(X) \geq \frac{1}{k}$. □

The Curvature. We will compute the curvature C_f of our function $f(X) := X \bullet X$, showing that $C_f = 2$ in this case. Using the definition (2.7), and the fact that here

$$\begin{aligned} & f(Y) - f(X) - (Y - X) \bullet \nabla f(X) \\ = \ & Y \bullet Y - X \bullet X - (Y - X) \bullet 2X \\ = \ & \|X - Y\|_{Fro}^2 \ , \end{aligned}$$

one obtains that $C_f = \sup_{X,Y \in \mathcal{S}} \|X - Y\|_{Fro}^2 = \mathrm{diam}_{Fro}(\mathcal{S})^2 = 2$. Finally the following Lemma 3.15 shows that the diameter is indeed 2.

Lemma 3.15 (Diameter of the Spectahedron)**.**

$$\mathrm{diam}_{Fro}(\mathcal{S})^2 = 2 \ .$$

Proof. Using the fact that the spectahedron $\mathcal{S}$ is the convex hull of the rank-1 matrices of unit trace, see Lemma 3.8, we know that the diameter must be attained at two vertices of $\mathcal{S}$, i.e. $u, v \in \mathbb{R}^n$ with $\|u\|_2 = \|v\|_2 = 1$, and

$$\begin{aligned} & \left\| vv^T - uu^T \right\|_{Fro}^2 \\ = \; & vv^T \bullet vv^T + uu^T \bullet uu^T - 2vv^T \bullet uu^T \\ = \; & v^T vv^T v + u^T uu^T u - 2u^T vv^T u \\ = \; & \|v\|^4 + \|u\|^4 - 2(u^T v)^2 \; . \end{aligned}$$

Clearly, this quantity is maximized if u and v are orthogonal. □

Note: We could also study $f(X) := \gamma \|X\|_{Fro}^2$ instead, for some $\gamma > 0$. This function has *curvature* constant $C_f = 2\gamma$, and for this scaling our above lower bounds will also just scale linearly, giving $\frac{C_f}{k}$ instead of $\frac{1}{k}$.

3.5. Semidefinite Optimization with ℓ_∞-Bounded Diagonal

Here we specialize our general Algorithm 1 to semidefinite optimization problems where all diagonal entries are individually constrained. This will result in a new optimization method that can also be applied to max-norm optimization problems, which we will discuss in more detail in Chapter 4. As in the previous Section 3.4, here we also consider matrix optimization problems over the space $\mathcal{X} = \mathbb{S}^{n \times n}$ of symmetric matrices, equipped with the standard Frobenius inner product $\langle X, Y \rangle = X \bullet Y$.

Formally, we consider the following special case of the general optimization problems (2.1), i.e.

$$\begin{array}{ll} \underset{X \in \mathbb{S}^{n \times n}}{\text{minimize}} & f(X) \\ s.t. & X_{ii} \leq 1 \quad \forall i, \\ & X \succeq 0 \; . \end{array} \tag{3.12}$$

We will write $\boxplus := \{X \in \mathbb{S}^{n \times n} \mid X \succeq 0, \; X_{ii} \leq 1 \; \forall i\}$ for the feasible set in this case, that is the PSD matrices whose diagonal entries are all upper bounded by one. This class of optimization problems has become widely known for the linear objective case when $f(X) = A \bullet X$, if A being the Laplacian matrix of a graph. In this case, one obtains the standard SDP relaxation of the Max-Cut problem [GW95], which we will briefly discuss below. Also, this optimization domain is strongly related to the matrix *max-norm*, which we study in more detail in Section 4.3.

Our general optimization Algorithm 1 directly applies to this specialized class of optimization problems as well, in which case it becomes the method depicted in the following Algorithm 7.

Algorithm 7 Sparse Greedy for Max-Norm Bounded Semidefinite Opt.

Input: Convex function f with curvature C_f, target accuracy ε
Output: ε-approximate solution for problem (3.12)
Set $X^{(0)} := vv^T$ for an arbitrary unit length vector $v \in \mathbb{R}^n$.
for $k = 0 \dots \infty$ **do**
  Let $\alpha := \frac{2}{k+2}$
  Compute $S := \text{ApproxLinear}\left(\nabla f(X^{(k)}), \boxplus, \alpha C_f\right)$
  Update $X^{(k+1)} := X^{(k)} + \alpha(S - X^{(k)})$
end for

The Linearized Problem. Here, the internal subproblem ApproxLinear() of approximately minimizing a linear function over the domain $\boxplus$ of PSD matrices is a non-trivial task. Every call of $\text{ApproxLinear}(A, \boxplus, \varepsilon')$ in fact means that we have to solve a semidefinite program $\min_{Y \in \boxplus} Y \bullet A$ for a given matrix A, or in other words

$$\begin{aligned} \underset{Y}{\text{minimize}} \quad & Y \bullet A \\ s.t. \quad & Y_{ii} \leq 1 \quad \forall i, \\ & Y \succeq 0 \end{aligned} \tag{3.13}$$

up to an additive approximation error of $\varepsilon' = \alpha C_f$.

Relation to Max-Cut. In [AHK05, Kal07], the same linear problem is denoted by (MaxQP). In the special case that A is chosen as the Laplacian matrix of a graph, then the above SDP is widely known as the standard SDP relaxation of the Max-Cut problem [GW95] (not to be confused with the combinatorial Max-Cut problem itself, which is known to be NP-hard). In fact the original relaxation uses equality constraints $Y_{ii} = 1$ on the diagonal instead, but for any matrix A of positive diagonal entries (such as e.g. a graph Laplacian), this condition follows automatically in the maximization variant of (3.13), see [KL96], or also [GM11, Kal07] for more background.

Duality and Duality of Norms. In Section 4.3 we will see that the above quantity (3.13) that determines both the step in our greedy Algorithm 7, but also the duality gap, is in fact the norm of A that is dual to the matrix max-norm.

Additionally, in the appendix Section A.7 we will explain that also for optimization problems of the form (3.12), the poor-man's duality given by

the linearization here (see also Section 2.2) indeed coincides with classical *Wolfe-duality* from the optimization literature.

Fortunately, it was shown by [AHK05] that also this linearized convex optimization problem (3.13) — and therefore also our internal procedure ApproxLinear(.) — can be solved relatively efficiently, if the matrix A (i.e. $\nabla f(X)$ in our case) is sparse.[3]

Theorem 3.16. *The algorithm of [AHK05] delivers an additive ε'-approximation to the linearized problem (3.13) in time*

$$\tilde{O}\left(\frac{n^{1.5}L^{2.5}}{\varepsilon'^{2.5}}N_A\right)$$

where the constant $L > 0$ is an upper bound on the maximum value of $Y \bullet A$ over $Y \in \boxplus$, and N_A is the number of non-zeros in A.

Proof. The results of [AHK05, Theorem 3] and [Kal07, Theorem 33] give a running time of order $\tilde{O}\left(\frac{n^{1.5}}{\varepsilon^{2.5}} \cdot \min\left\{N, \frac{n^{1.5}}{\varepsilon\alpha^*}\right\}\right)$ to obtain a multiplicative $(1-\varepsilon)$-approximation, where α^* is the value of an optimal solution. Formally we obtain $S \in \boxplus$ with $S \bullet A \geq (1-\varepsilon)\alpha^*$. In other words by using an accuracy of $\varepsilon := \frac{\varepsilon'}{\alpha^*}$, we obtain an additive ε'-approximation to (3.13). □

Here the notation $\tilde{O}(.)$ again suppresses poly-logarithmic factors in n, and N is the number of non-zero entries of the matrix A. Note that analogous to the approximate eigenvector computation for Hazan's Algorithm 6, we need the assumption that the linear function given by $Y \bullet \nabla f(X)$ is bounded over the domain $Y \in \boxplus$. However this is a reasonable assumption, as our function has bounded curvature C_f (corresponding to $\nabla f(X)$ being Lipschitz-continuous over the domain $\boxplus$), and the diameter of $\boxplus$ is bounded.

The reason we need an absolute approximation quality lies in the analysis of Algorithm 1, even if it would feel much more natural to work with relative approximation quality in many cases.

[3] Also, Kale in [Kal07, Theorem 14] has shown that this problem can be solved very efficiently if the matrix $A = -\nabla f(X)$ is sparse. Namely if A is the Laplacian matrix of a weighted graph, then a multiplicative ε-approximation to (3.13) can be computed in time $\tilde{O}(\frac{\Delta^2}{d^2}N_A)$ time, where N_A is the number of non-zero entries of the matrix A. Here Δ is the maximum entry on the diagonal of A, and d is the average value on the diagonal.

Convergence. The convergence result for the general Algorithm 1 directly gives us the analysis for the specialized algorithm here. Note that the curvature over the domain $\boxplus$ here is given by

$$C_f := \sup_{\substack{X,V\in\boxplus,\,\alpha\in[0,1],\\ Y=X+\alpha(V-X)}} \tfrac{1}{\alpha^2}\big(f(Y) - f(X) + (Y-X)\bullet\nabla f(X)\big)\ . \tag{3.14}$$

Theorem 3.17. *For each $k \geq 1$, the iterate $X^{(k)}$ of Algorithm 7 satisfies*

$$f(X^{(k)}) - f(X^*) \leq \frac{8C_f}{k+2}\ .$$

where $X^* \in \mathcal{S}$ is an optimal solution to problem (3.12).

Furthermore, after at most $2\left\lceil\frac{8C_f}{\varepsilon}\right\rceil + 1 = O(\frac{1}{\varepsilon})$ many steps, it has an iterate $X^{(k)}$ with $g(X^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. □

Applications. The new algorithm can be used to solve arbitrary max-norm constrained convex optimization problems, such as max-norm regularized matrix completion problems, which we will study in the next Chapter 4.

3.6. Sparse Semidefinite Optimization

Another interesting optimization domain among the semidefinite matrices is given by the matrices with only one non-zero off-diagonal entry. Here we specialize our general Algorithm 1 to convex optimization over the convex hull given by such matrices. Our algorithm will therefore obtain ε-approximate solutions given by only $O\left(\frac{1}{\varepsilon}\right)$ such sparse matrices, or in other words solutions of sparsity $O\left(\frac{1}{\varepsilon}\right)$.

Why bother? The same sparse matrices are also used in the graph sparsification approach by [BSS09][4]. Furthermore, sparse solutions to convex matrix optimization problems have gained interest in dimensionality reduction, as in sparse PCA, see [ZdG10] for an overview.

[4] The theoretical result of [BSS09] guarantees that all eigenvalues of the resulting sparse matrix (corresponding to the Laplacian of a sparse graph) do not differ too much from their counterparts in the original graph.

Setup. Formally, here we again use the standard Frobenius inner product $\langle X, Y \rangle = X \bullet Y$ over the symmetric matrices $\mathbb{S}^{n \times n}$, and consider the sparse PSD matrices given by $P^{(ij)} := (\mathbf{e}_i + \mathbf{e}_j)(\mathbf{e}_i + \mathbf{e}_j)^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$, for any fixed pair of indices $i, j \in [n]$, $i \neq j$. In other words $P^{(ij)}_{uv} = 1$ for $u \in \{i, j\}, v \in \{i, j\}$, and zero everywhere else. We also consider the analogous "negative" counterparts of such matrices, namely $N^{(kl)} := (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & -1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -1 & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$, i.e. $N^{(ij)}_{uv} = -1$ for the two off-diagonal entries $(u, v) \in \{(i, j), (j, i)\}$, and $N^{(ij)}_{uv} = 1$ for the two diagonal entries $(u, v) \in \{(i, i), (j, j)\}$, and zero everywhere else.

Analogously to the two previous applications of our method to semidefinite optimization, we now optimize a convex function, i.e.

$$\underset{X \in \mathcal{S}^4_{\text{sparse}}}{\text{minimize}} \, f(X) \tag{3.15}$$

over the domain

$$D = \mathcal{S}^4_{\text{sparse}} := \text{conv}\left(\bigcup_{ij} P^{(ij)} \cup \bigcup_{ij} N^{(ij)} \right) .$$

Optimizing over Sparse Matrices, and Solving the Linearization. Applying our general Algorithm 1 to this class of problems (3.15) becomes very simple, as the linear primitive problem ExactLinear $\left(D_X, \mathcal{S}^4_{\text{sparse}}\right)$ for any fixed matrix $D_X \in \mathbb{S}^{n \times n}$ is easily solved over $\mathcal{S}^4_{\text{sparse}}$. From our simple Lemma 2.8 on linear functions over convex hulls, we know that this linear minimum is attained by the single sparse matrix $P^{(ij)}$ or $N^{(ij)}$ that maximizes the inner product with $-D_X$. The optimal pair of indices (k, l) can be found by a linear pass through the gradient $D_X = \nabla f(X)$. This means that the linearized problem is much easier to solve than in the above two Sections 3.4 and 3.5. Altogether, Algorithm 1 will build approximate solutions $X^{(k)}$, each of which is a convex combination of k of the atomic matrices $P^{(ij)}$ or $N^{(ij)}$, as formalized in the following theorem:

Theorem 3.18. *Let $n \geq 2$ and let $X^{(0)} := P^{(12)}$ be the starting point. Then for each $k \geq 1$, the iterate $X^{(k)}$ of Algorithm 1 has at most $4(k+1)$ non-zero entries, and satisfies*

$$f(X^{(k)}) - f(X^*) \leq \frac{8 C_f}{k+2} .$$

where $X^* \in \mathcal{S}^4_{sparse}$ is an optimal solution to problem (3.15).

Furthermore, for any $\varepsilon > 0$, after at most $2\left\lceil \frac{8C_f}{\varepsilon} \right\rceil + 1 = O\left(\frac{1}{\varepsilon}\right)$ many steps, it has an iterate $X^{(k)}$ of only $O\left(\frac{1}{\varepsilon}\right)$ many non-zero entries, satisfying $g(X^{(k)}) \leq \varepsilon$.

Proof. This is a corollary of Theorem 2.3 and Theorem 2.5. The sparsity claim follows from our observation that the step directions given by $\textsc{ExactLinear}\left(\nabla f(X), \mathcal{S}^4_{\text{sparse}}\right)$ are always given by one of the sparse matrices $P^{(ij)}$ or $N^{(ij)}$. □

Optimizing over Non-Negative Matrix Factorizations. We also consider the slight variant of (3.15), namely optimizing only over one of the two types of matrices as the domain D, i.e. only combinations of positive $P^{(ij)}$ or only of negative $N^{(ij)}$. This means that the domain is given by $D = \mathcal{S}^{4+}_{\text{sparse}} := \text{conv}\left(\bigcup_{ij} P^{(ij)}\right)$ or $D = \mathcal{S}^{4-}_{\text{sparse}} := \text{conv}\left(\bigcup_{ij} N^{(ij)}\right)$. The above analysis for Algorithm 1 holds in exactly the same way. Now for $\mathcal{S}^{4+}_{\text{sparse}}$, each step direction $s = s^{(k)}$ used by the algorithm is given by $s = P^{(ij)} = (\mathbf{e}_i + \mathbf{e}_j)(\mathbf{e}_i + \mathbf{e}_j)^T$ for some i, j, and so we have that each of the approximations $X^{(k)}$ is a sum of k many positive rank-1 factors of this form. In other words in each step k, $X^{(k)} = LR^T$ is a product of two (entry-wise) non-negative matrices of at most k columns each, i.e. $L \in \mathbb{R}^{n \times k}$ and $R^{n \times k}$. Consequently, our algorithm provides solutions that are *non-negative matrix factorizations*, which is a successful technique in matrix completion problems from recommender systems, see e.g. [Wu07].

Relation to Bounded Trace and Diagonally Dominant Matrices. Observe the matrices in $\mathcal{S}^4_{\text{sparse}}$ form a subset of the bounded trace PSD matrices $\mathcal{S}$ that we studied in the previous Section 3.4, since every matrix $P^{(ij)}$ or $N^{(ij)}$ is PSD and has trace equal two. Furthermore, we observe that all matrices $X \in \mathcal{S}^4_{\text{sparse}}$ are *diagonally dominant*, meaning that

$$|X_{ii}| \geq \sum_{j \neq i} |X_{ij}| \qquad \forall i \in [n]$$

In the case that we restrict to using only one of the two types of matrices $\mathcal{S}^{4+}_{\text{sparse}}$ or $\mathcal{S}^{4-}_{\text{sparse}}$ as the domain, then we have that *equality* $|X_{ii}| = \sum_{j \neq i} |X_{ij}|$ always holds, since this equality is preserved under taking convex combinations, and holds for the atomic matrices $P^{(ij)}$ and $N^{(ij)}$.

The Curvature. The above reasoning also implies that the curvature C_f for problems of the form (3.15) is upper bounded by the curvature in the spectahedron-case as given in (3.7), since $\mathcal{S}^{4+}_{\text{sparse}} \subseteq \mathcal{S}^{4}_{\text{sparse}} \subseteq 2 \cdot \mathcal{S}$.

Applications and Future Research. Computationally, the approach here looks very attractive, as the cost of a "sparse" step here is much cheaper than an approximate eigenvector computation which is needed in the bounded trace case as explained in Section 3.4.

Also, it will be interesting to see how a regularization by constraining to a scaled domain $\mathcal{S}^{4}_{\text{sparse}}$ or $\mathcal{S}^{4+}_{\text{sparse}}$ will perform in practical machine learning applications as for example dimensionality reduction, compared to nuclear norm regularization that we will discuss in the following Chapter 4.

It also remains to investigate further on whether we can approximate general bounded trace semidefinite problems of the form (3.5) by using only sparse matrices.

3.7. Submodular Optimization

For a finite ground set S, a real valued function defined on all subsets of S, is called *submodular*, if

$$g(X \cap Y) + g(X \cup Y) \leq g(X) + g(Y) \qquad \forall X, Y \subseteq S$$

For any given submodular function g with $g(\emptyset) = 0$, the paper [Lov83, Section 3] introduces a corresponding convex set in $\mathbb{R}^{|S|}$, called the *submodular polyhedron* (or also Lovasz polyhedron),

$$\mathcal{P}_g := \left\{ x \in \mathbb{R}^{|S|} \;\middle|\; \sum_{i \in T} x_i \leq g(T) \quad \forall T \subseteq S \right\} .$$

We would now like to study convex optimization problems over such domains, which become compact convex sets if we intersect with the positive orthant, i.e. $D := \mathcal{P}_g \cap \mathbb{R}^{|S|}_{\geq 0}$.

Nicely for our optimization framework, [Lov83, Section 3] already showed that there is a simple greedy algorithm which optimizes any linear function over the domain $\mathcal{P}_g$, i.e. it solves $\max_{x \in \mathcal{P}_g} c^T x$, or in other words it exactly solves our internal problem $\textsc{ExactLinear}\,(c, \mathcal{P}_g)$.

Lovasz [Lov83] already demonstrated how to use this kind of linear optimization over $\mathcal{P}_g$ to solve submodular minimization problems. It remains

to investigate if there are interesting applications for the wider class of more general convex (non-linear) functions f over such domains, as addressed by our Algorithm 1.

4

Optimization with the Nuclear and Max-Norm

Matrix optimization problems with a nuclear norm or max-norm regularization, such as e.g. low norm matrix factorizations, have seen many applications recently, ranging from low-rank recovery, dimensionality reduction, to recommender systems. We propose two new first-order approximation methods building upon two of the simple semidefinite optimizers from the previous Chapter, that is the approximate SDP solver of [Haz08] from Section 3.4 on one hand, and our bounded diagonal optimizer from Section 3.5 on the other hand. The algorithms come with strong convergence guarantees.

In contrast to existing methods, our nuclear norm optimizer does not need any Cholesky or singular value decompositions internally, and provides guaranteed approximations that are simultaneously of low rank. The method is free of tuning parameters, and easy to parallelize.

4.1. Introduction

In this chapter we consider convex optimization problems over matrices, which come with a regularization on either the nuclear norm or the max-norm of the optimization variable.

Convex optimization with the nuclear norm has become a very successful technique in various machine learning, computer vision and compressed sensing areas such as low-rank recovery [FHB01, CR09, CT10], dimensionality reduction (such as robust principal component analysis [CLMW11]), and also recommender systems and matrix completion. Here matrix factorizations [SRJ04, KBV09] — regularized by the nuclear or max-norm — have gained a lot of attention with the recently ended *Netflix Prize* competition. Many more applications of similar optimization problems can be found among dimensionality reduction, matrix classification, multi-task learning, spectral clustering and others. The success of these methods is fueled by the property of the nuclear norm being a natural convex relaxation of the rank, allowing the use of scalable convex optimization techniques.

Based on the semidefinite optimization methods that we have presented in the above Sections 3.4 and 3.5, we propose two new, yet simple, first-order algorithms for nuclear norm as well as max-norm regularized convex optimization.

For the nuclear norm case, our proposed method builds upon the first-order scheme for semidefinite optimization by [Haz08], which we have investigated in Section 3.4.1. This approach allows us to significantly reduce the computational complexity per iteration, and therefore scale to much larger datasets: While existing methods need an entire and exact singular value decomposition (SVD) in each step, our method only uses a single approximate eigenvector computation per iteration, which can be done by e.g. the power method. A conference version of our work for nuclear norm regularized problems has appeared in [JS10].

In the same spirit, we will also give a new algorithm with a convergence guarantee for optimizing with a max-norm regularization. For matrix completion problems, experiments show that the max-norm can result in an improved generalization performance compared to the nuclear norm in some cases [SRJ04, LRS+10].

Nuclear Norm Regularized Convex Optimization. We consider the following convex optimization problems over matrices:

$$\min_{Z \in \mathbb{R}^{m \times n}} f(Z) + \mu \|Z\|_* \tag{4.1}$$

and the corresponding constrained variant

$$\min_{Z \in \mathbb{R}^{m \times n},\, \|Z\|_* \leq \frac{t}{2}} f(Z) \tag{4.2}$$

where $f(Z)$ is any differentiable convex function (usually called the loss function), $\|.\|_*$ is the *nuclear norm* of a matrix, also known as the *trace norm* (sum of the singular values, or ℓ_1-norm of the spectrum). Here $\mu > 0$ and $t > 0$ respectively are given parameters, usually called the *regularization parameter*.

The nuclear norm is know as the natural generalization of the (sparsity inducing) ℓ_1-norm for vectors, to the case of semidefinite matrices. When choosing $f(X) := \|\mathcal{A}(X) - b\|_2^2$ for some linear map $\mathcal{A} : \mathbb{R}^{n\times m} \to \mathbb{R}^p$, the above formulation (4.1) is the matrix generalization of the problem $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 + \mu \|x\|_1$, for a fixed matrix A, which is the important ℓ_1-*regularized least squares problem*, also known as the basis pursuit denoising problem in the compressed sensing literature, see also Section 3.2.1. The analoguous vector variant of (4.2) is the *Lasso* problem [Tib96] which is $\min_{x\in\mathbb{R}^n} \left\{ \|Ax - b\|_2^2 \;\middle|\; \|x\|_1 \le t \right\}$.

Max-Norm Regularized Convex Optimization. Intuitively, one can think of the matrix max-norm as the generalization of the vector ℓ_∞-norm to PSD matrices. Here we consider optimization problems with a max-norm regularization, which are given by

$$\min_{Z\in\mathbb{R}^{m\times n}} f(Z) + \mu \|Z\|_{\max} \tag{4.3}$$

and the corresponding constrained variant being

$$\min_{Z\in\mathbb{R}^{m\times n},\, \|Z\|_{\max}\le t} f(Z) \ . \tag{4.4}$$

Our Contribution. Applying our general optimization method from the previous Chapters 2 and 3, we present a much simpler algorithm to solve problems of the form (4.2), which does not need any internal SVD computations. The same approach will also solve the max-norm regularized problems (4.4). We achieve this by transforming the problems to the convex optimization setting over positive semidefinite matrices which we have studied in the above Sections 3.4.1 and 3.5.

Our new approach has several advantages for nuclear norm optimization when compared to the existing algorithms such as "proximal gradient" methods (APG) and "singular value thresholding" (SVT), see e.g. [GLW+09, CCS10, TY10, JY09], and also in comparison to the alternating-gradient-descent-type methods (as e.g. [RS05, Lin07]).

i) By employing the approximate SDP solver by [Haz08], see Algorithm 6, we obtain a guaranteed ε-approximate solution Z after $O\left(\frac{1}{\varepsilon}\right)$ iterations. Crucially, the resulting solution Z is simultaneously of low rank, namely rank $O\left(\frac{1}{\varepsilon}\right)$. Also the algorithm maintains a compact representation of Z in terms of a low-rank matrix factorization $Z = LR^T$ (with the desired bounded nuclear norm), and can therefore even be applied if the full matrix Z would be far too large to even be stored.

ii) Compared to the alternating-gradient-descent-type methods from machine learning, we overcome the problem of working with non-convex formulations of the form $f(LR^T)$, which is NP-hard, and instead solve the original convex problem in $f(Z)$.

iii) The total running time of our algorithm for nuclear norm problems grows linear in the problem size, allows to take full advantage of sparse problems such as e.g. for matrix completion. More precisely, the algorithm runs in time $O\left(\frac{N_f}{\varepsilon^{1.5}}\right)$, where N_f is the number of matrix entries on which the objective function f depends. Per iteration, our method consists of only a single approximate (largest) eigenvector computation, allowing it to scale to any problem size where the power method (or Lanczos' algorithm) can still be applied. This also makes the method easy to implement and to parallelize. Existing APG/SVT methods by contrast need an entire SVD in each step, which is significantly more expensive.

iv) On the theory side, our simple convergence guarantee of $O\left(\frac{1}{\varepsilon}\right)$ steps holds even if the used eigenvectors are only approximate. In comparison, those existing methods that come with a convergence guarantee do require an exact SVD in each iteration, which might not always be a realistic assumption in practice.

We demonstrate that our new algorithm on standard datasets improves over the state of the art methods, and scales to large problems such as matrix factorizations on the Netflix dataset.

Hazan's Algorithm 6 can be interpreted as the generalization of the *coreset* approach to problems on symmetric matrices, which we have explained in the previous Section 3.4.1. Compared to the $O(1/\sqrt{\varepsilon})$ convergence methods in the spirit of [Nes83, Nes07a], our number of steps is larger, which is however more than compensated by the improved step complexity, being lower by a factor of roughly $(n + m)$.

Our new method for the nuclear norm case can also be interpreted as a modified, theoretically justified variant of Simon Funk's popular SVD

heuristic [Web06] for regularized matrix factorization. To our knowledge this is the first guaranteed convergence result for this class of alternating-gradient-descent-type algorithms.

Related Work. For nuclear norm optimization, there are two lines of existing methods. On the one hand, in the optimization community, [TY10, LST09], [GLW+09] and [JY09] independently proposed algorithms that obtain an ε-accurate solution to (4.1) in $O(1/\sqrt{\varepsilon})$ steps, by improving the algorithm of [CCS10]. These methods are known under the names "accelerated proximal gradient" (APG) and "singular value thresholding" (SVT). More recently also [MHT10] and [MGC09] proposed algorithms along the same idea. Each step of all those algorithms requires the computation of the singular value decomposition (SVD) of a matrix of the same size as the solution matrix, which is expensive even with the currently available fast methods such as PROPACK. [TY10] and [JY09] and also [GLW+09] show that the primal error of their algorithm is smaller than ε after $O(1/\sqrt{\varepsilon})$ steps, using an analysis inspired by [Nes83] and [BT09]. For an overview of related algorithms, we also refer the reader to [CLMW11]. As mentioned above, the method presented here has a significantly lower computational cost per iteration (one approximate eigenvector compared to a full exact SVD), and is also faster in practice on large matrix completion problems.

On the other hand, in the machine learning community, research originated from matrix completion and factorization [SRJ04], later motivated by the Netflix prize challenge, getting significant momentum from the famous blog post by [Web06]. Only very recently an understanding has formed that many of these methods can indeed by seen as optimizing with regularization term closely related to the nuclear norm, see Section 4.5.4 and [SS10]. The majority of the currently existing machine learning methods such as for example [RS05, Lin07] and later also [Pat07, ZWSP08, KBV09, TPNT09, IR10, GNHS11] are of the type of "alternating" gradient descent applied to $f(LR^T)$, where at each step one of the factors L and R is kept fixed, and the other factor is updated by a gradient or stochastic gradient step. Therefore, despite working well in many practical applications, all these mentioned methods can get stuck in local minima — and so are theoretically not well justified, see also the discussion in [DeC06] and our Section 4.4.

The same issue also comes up for max-norm optimization, where for example [LRS+10] optimize over the non-convex factorization (4.8) for bounded max-norm. To our knowledge, no algorithm with a convergence guarantee was known so far.

Furthermore, optimizing with a rank constraint was recently shown to be NP-hard [GG10]. In practical applications, nearly all approaches for large scale problems are working over a factorization $Z = LR^T$ of bounded rank, therefore ruling out their ability to obtain a solution in polynomial time in the worst-case, unless P = NP.

Our new method for both nuclear and max-norm avoids all the above described problems by solving an equivalent convex optimization problem, and provably runs in near linear time in the nuclear norm case.

4.2. The Nuclear Norm for Matrices

The *nuclear norm* $\|Z\|_*$ of a rectangular matrix $Z \in \mathbb{R}^{m \times n}$, also known as the *trace norm* or *Ky Fan norm*, is given by the sum of the singular values of Z, which is equal to the ℓ_1-norm of the singular values of Z (because singular values are always non-negative). Therefore, the nuclear norm is often called the Schatten ℓ_1-norm. In this sense, it is a natural generalization of the ℓ_1-norm for vectors which we have studied earlier.

The nuclear norm has a nice equivalent characterization in terms of matrix factorizations of Z, i.e.

$$\|Z\|_* := \min_{LR^T=Z} \frac{1}{2}\big(\|L\|_{Fro}^2 + \|R\|_{Fro}^2 \big) \ , \tag{4.5}$$

where the number of columns of the factors $L \in \mathbb{R}^{m \times k}$ and $R^{n \times k}$ is not constrained [FHB01, SRJ04]. In other words, the nuclear norm constrains the average Euclidean row or column norms of any factorization of the original matrix Z.

Furthermore, the nuclear norm is *dual* to the standard spectral matrix norm (i.e. the matrix operator norm), meaning that

$$\|Z\|_* = \max_{B, \|B\|_{spec} \leq 1} B \bullet Z \ ,$$

see also [RFP10]. Recall that $\|B\|_{spec}$ is defined as the first singular value $\sigma_1(B)$ of the matrix B.

Similarly to the property of the vector $\|.\|_1$-norm being the best convex approximation to the sparsity of a vector, as we discussed in Section 3.2 the nuclear norm is the best convex approximation of the matrix rank. More precisely, $\|.\|_*$ is the convex envelope of the rank [FHB01], meaning that it is the largest convex function that is upper bounded by the rank on the convex domain of matrices $\Big\{ Z \ \Big| \ \|Z\|_{spec} \leq 1 \Big\}$. This motivates why

the nuclear norm is widely used as a proxy function (or convex relaxation) for *rank minimization*, which otherwise is a hard combinatorial problem.

Its relation to semidefinite optimization — which explains why the nuclear norm is often called the trace norm — is that

$$\begin{aligned} \|Z\|_* = \underset{V,W}{\text{minimize}} \quad & t \\ s.t. \quad & \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix} \succeq 0 \ \text{ and} \\ & \operatorname{Tr}(V) + \operatorname{Tr}(W) \leq 2t \ . \end{aligned} \tag{4.6}$$

Here the two optimization variables range over the symmetric matrices $V \in \mathbb{S}^{m\times m}$ and $W \in \mathbb{S}^{n\times n}$. This semidefinite characterization will in fact be the central tool for our algorithmic approach for nuclear norm regularized problems in the following. The equivalence of the above characterization to the earlier "factorization" formulation (4.5) is a consequence of the following simple Lemma 4.1. The Lemma gives a correspondence between the (rectangular) matrices $Z \in \mathbb{R}^{m\times n}$ of bounded nuclear norm on one hand, and the (symmetric) PSD matrices $X \in \mathbb{S}^{(m+n)\times(m+n)}$ of bounded trace on the other hand.

Lemma 4.1 ([FHB01, Lemma 1]). *For any non-zero matrix $Z \in \mathbb{R}^{m\times n}$ and $t \in \mathbb{R}$, it holds that*

$$\|Z\|_* \leq \frac{t}{2}$$

if and only if

$$\begin{aligned} & \exists \text{ symmetric matrices } V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n} \\ & s.t. \ \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix} \succeq 0 \quad \text{and} \quad \operatorname{Tr}(V) + \operatorname{Tr}(W) \leq t \ . \end{aligned}$$

Proof. $\boxed{\Rightarrow}$ Using the characterization (4.5) of the nuclear norm $\|Z\|_* = \min_{LR^T=Z} \frac{1}{2}(\|L\|_{Fro}^2 + \|R\|_{Fro}^2)$ we get that $\exists\, L, R,\ LR^T = Z$ s.t. $\|L\|_{Fro}^2 + \|R\|_{Fro}^2 = \operatorname{Tr}(LL^T) + \operatorname{Tr}(RR^T) \leq t$, or in other words we have found a matrix $\left(\begin{smallmatrix} LL^T & Z \\ Z^T & RR^T \end{smallmatrix}\right) = \left(\begin{smallmatrix} L \\ R \end{smallmatrix}\right)\left(\begin{smallmatrix} L \\ R \end{smallmatrix}\right)^T \succeq 0$ of trace $\leq t$.
$\boxed{\Leftarrow}$ As the matrix $\left(\begin{smallmatrix} V & Z \\ Z^T & W \end{smallmatrix}\right)$ is symmetric and PSD, it can be (Cholesky) factorized to $(L; R)(L; R)^T$ s.t. $LR^T = Z$ and $t \geq \operatorname{Tr}(LL^T) + \operatorname{Tr}(RR^T) = \|L\|_{Fro}^2 + \|R\|_{Fro}^2$, therefore $\|Z\|_* \leq \frac{t}{2}$. □

Interestingly, for characterizing bounded nuclear norm matrices, it does not make any difference whether we enforce an equality or inequality constraint on the trace. This fact will turn out to be useful in order to apply our Algorithm 6 later on.

Corollary 4.2. *For any non-zero matrix $Z \in \mathbb{R}^{m\times n}$ and $t \in \mathbb{R}$, it holds that*

$$\|Z\|_* \leq \frac{t}{2}$$

if and only if

$$\exists \text{ symmetric matrices } V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n}$$
$$\text{s.t. } \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix} \succeq 0 \quad \text{and} \quad \operatorname{Tr}(V) + \operatorname{Tr}(W) = t \ .$$

Proof. $\boxed{\Rightarrow}$ From Lemma 4.1 we obtain a matrix $\left(\begin{smallmatrix} V & Z \\ Z^T & W \end{smallmatrix}\right) =: X \succeq 0$ of trace say $s \leq t$. If $s < t$, we add $(t-s)$ to the top-left entry of V, i.e. we add to X the PSD rank-1 matrix $(t-s)\mathbf{e}_1\mathbf{e}_1^T$ (which again gives a PSD matrix). $\boxed{\Leftarrow}$ follows directly from Lemma 4.1. □

4.2.1. Weighted Nuclear Norm

A promising weighted nuclear norm regularization for matrix completion was recently proposed by [SS10]. For fixed weight vectors $p \in \mathbb{R}^m, q \in \mathbb{R}^n$, the weighted nuclear norm $\|Z\|_{nuc(p,q)}$ of $Z \in \mathbb{R}^{m\times n}$ is defined as

$$\|Z\|_{nuc(p,q)} := \|PZQ\|_* \,,$$

where $P = \operatorname{diag}(\sqrt{p}) \in \mathbb{R}^{m\times m}$ denotes the diagonal matrix whose i-th diagonal entry is $\sqrt{p_i}$, and analogously for $Q = \operatorname{diag}(\sqrt{q}) \in \mathbb{R}^{n\times n}$. Here $p \in \mathbb{R}^m$ is the vector whose entries are the probabilities $p(i) > 0$ that the i-th row is observed in the sampling Ω. Analogously, $q \in \mathbb{R}^n$ contains the probability $q(j) > 0$ for each column j. The opposite weighting (using $\frac{1}{p(i)}$ and $\frac{1}{q(j)}$ instead of $p(i)$,$q(j)$) has also been suggested by [WKS08].

Any optimization problem with a weighted nuclear norm regularization

$$\min_{Z\in\mathbb{R}^{m\times n},\, \|Z\|_{nuc(p,q)} \leq t/2} f(Z) \tag{4.7}$$

and arbitrary loss function f can therefore be formulated equivalently over the domain $\|PZQ\|_* \leq t/2$, such that it reads as (if we substitute $\bar{Z} := PZQ$),

$$\min_{\bar{Z}\in\mathbb{R}^{m\times n},\, \|\bar{Z}\|_* \leq t/2} f(P^{-1}\bar{Z}Q^{-1}).$$

Hence, we have reduced the task to our standard convex problem (4.2) for $\hat{f}$ that here is defined as

$$\hat{f}(X) := f(P^{-1}\bar{Z}Q^{-1}),$$

where $X =: \left(\begin{smallmatrix} V & \bar{Z} \\ Z^T & W \end{smallmatrix}\right)$. This equivalence implies that any algorithm solving (4.2) also serves as an algorithm for weighted nuclear norm regularization. In particular, Hazan's Algorithm 6 does imply a guaranteed approximation quality of ε for problem (4.7) after $O\left(\frac{1}{\varepsilon}\right)$ many rank-1 updates, as we discussed in Section 3.4. So far, to the best of our knowledge, no approximation guarantees were known for the weighted nuclear norm.

We will discuss solution path algorithms (maintaining approximation guarantees when the regularization parameter t changes) also for the weighted nuclear norm case in Chapter 7.

4.3. The Max-Norm for Matrices

We think of the matrix max-norm as a generalization of the vector ℓ_∞-norm to the case of positive semidefinite matrices, which we have studied before.

In some matrix completion applications, the max-norm has been observed to provide solutions of better generalization performance than the nuclear norm [SRJ04]. Both matrix norms can be seen as a convex surrogate of the rank [SS05].

The *max-norm* $\|Z\|_{\max}$ of a rectangular matrix $Z \in \mathbb{R}^{m\times n}$ has a nice characterization in terms of matrix factorizations of Z, i.e.

$$\|Z\|_{\max} := \min_{LR^T=Z} \max\{\|L\|_{2,\infty}^2, \|R\|_{2,\infty}^2\} , \tag{4.8}$$

where the number of columns of the factors $L \in \mathbb{R}^{m\times k}$ and $R^{n\times k}$ is not constrained [LRS+10]. Here $\|L\|_{2,\infty}$ is the maximum ℓ_2-norm of any row $L_{i:}$ of L, that is $\|L\|_{2,\infty} := \max_i \|L_{i:}\|_2 = \max_i \sqrt{\sum_k L_{ik}^2}$. Compared to the nuclear norm, we therefore observe that the max-norm constrains the maximal Euclidean row-norms of any factorization of the original matrix Z, see also [SS05]. [1]

An alternative formulation of the max-norm was given by [LMSS07] and [SS05], stating that

$$\|Z\|_{\max} = \min_{LR^T=Z} (\max_i ||L_{i:}||_2)(\max_i ||R_{i:}||_2) .$$

[1] Note that the max-norm does *not* coincide with the matrix norm induced by the vector $\|.\|_\infty$-norm, that is $\|Z\|_\infty := \sup_{x\neq 0} \frac{\|Zx\|_\infty}{\|x\|_\infty}$. The latter matrix norm by contrast is known to be the maximum of the row sums of Z (i.e. the ℓ_1-norms of the rows).

The dual norm to the max-norm, as given in [SS05], is

$$\begin{aligned} \|Z\|_{\max}^* = & \max_{\|Y\|_{\max}\leq 1} Z \bullet Y \\ = & \max_{\substack{k,\\ l_i\in\mathbb{R}^k, \|l_i\|_2\leq 1\\ r_j\in\mathbb{R}^k, \|r_j\|_2\leq 1}} \sum_{i,j} Z_{ij}\, l_i^T r_j \ , \end{aligned}$$

where the last equality follows from the characterization (4.8).

The relation of the max-norm to semidefinite optimization — which also explains the naming of the max-norm — is that

$$\begin{aligned} \|Z\|_{\max} = \underset{V,W}{\text{minimize}} \quad & t \\ s.t. \quad & \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix} \succeq 0 \ \text{ and } \ \begin{matrix} V_{ii} \leq t & \forall i \in [m], \\ W_{ii} \leq t & \forall i \in [n] \end{matrix} \end{aligned} \tag{4.9}$$

Here the two optimization variables range over the symmetric matrices $V \in \mathbb{S}^{m\times m}$ and $W \in \mathbb{S}^{n\times n}$, see for example [LRS+10]. As already in the nuclear norm case, this semidefinite characterization will again be the central tool for our algorithmic approach for max-norm regularized problems in the following. The equivalence of the above characterization to the earlier "factorization" formulation (4.8) is a consequence of the following simple Lemma 4.3. The Lemma gives a correspondence between the (rectangular) matrices $Z \in \mathbb{R}^{m\times n}$ of bounded max-norm on one hand, and the (symmetric) PSD matrices $X \in \mathbb{S}^{(m+n)\times(m+n)}$ of uniformly bounded diagonal on the other hand.

Lemma 4.3. *For any non-zero matrix $Z \in \mathbb{R}^{n\times m}$ and $t \in \mathbb{R}$:*

$$\|Z\|_{\max} \leq t$$

if and only if

$$\begin{aligned} & \exists \text{ symmetric matrices } V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n} \\ & s.t. \ \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix} \succeq 0 \ \text{ and } \ \begin{matrix} V_{ii} \leq t & \forall i \in [m], \\ W_{ii} \leq t & \forall i \in [n] \end{matrix} \end{aligned}$$

Proof. $\boxed{\Rightarrow}$ Using the characterization (4.8) of the max-norm, i.e. $\|Z\|_{\max} = \min_{LR^T=Z} \max\{\|L\|_{2,\infty}^2, \|R\|_{2,\infty}^2\}$, we get that there exist L, R with $LR^T = Z$, s.t. $\max\{\|L\|_{2,\infty}^2, \|R\|_{2,\infty}^2\} = \max\{\max_i \|L_{i:}\|_2^2, \max_i \|R_{i:}\|_2^2\} \leq t$, or in other words we have found a matrix $\left(\begin{smallmatrix} LL^T & Z \\ Z^T & RR^T \end{smallmatrix}\right) = (L;R)(L;R)^T \succeq 0$ where every diagonal element is at most t, that is $\|L_{i:}\|_2^2 = (LL^T)_{ii} \leq$

$t\ \forall i \in [m]$, and $\|R_{i:}\|_2^2 = (RR^T)_{ii} \leq t\ \forall i \in [n]$.
$\boxed{\Leftarrow}$ As the matrix $\left(\begin{smallmatrix} V & Z \\ Z^T & W \end{smallmatrix}\right)$ is symmetric and PSD, it can be (Cholesky) factorized to $(L; R)(L; R)^T$ s.t. $LR^T = Z$ and $\|L_{i:}\|_2^2 = (LL^T)_{ii} \leq t\ \forall i \in [m]$ and $\|R_{i:}\|_2^2 = (RR^T)_{ii} \leq t\ \forall i \in [n]$, which implies $\|Z\|_{\max} \leq t$. □

4.4. Optimizing with Bounded Nuclear Norm and Max-Norm

Most of the currently known algorithms for matrix factorizations as well as nuclear norm or max-norm regularized optimization problems, such as (4.1), (4.2), (4.3) or (4.4), do suffer from the following problem:

In order to optimize the convex objective function $f(Z)$ while controlling the norm $\|Z\|_*$ or $\|Z\|_{\max}$, the methods instead try to optimize $f(LR^T)$, with respect to both factors $L \in \mathbb{R}^{m\times k}$ and $R \in \mathbb{R}^{n\times k}$, with the corresponding regularization constraint imposed on L and R. This approach is of course very tempting, as the constraints on the factors — which originate from the matrix factorization characterizations (4.5) and (4.8) — are simple and in some sense easier to enforce.

Unhealthy Side-Effects of Factorizing. However, there is a significant price to pay: Even if the objective function $f(Z)$ is *convex* in Z, the very same function expressed as a function $f(LR^T)$ of both the factor variables L and R becomes a severely *non-convex* problem, naturally consisting of a large number of saddle-points (consider for example just the smallest case $L, R \in \mathbb{R}^{1\times 1}$ together with the identity function $f(Z) = Z \in \mathbb{R}$).

The majority of the currently existing methods such as for example [RS05, Lin07] and later also [Pat07, ZWSP08, KBV09, TPNT09, IR10, GNHS11] is of this "alternating" gradient descent type, where at each step one of the factors L and R is kept fixed, and the other factor is updated by e.g. a gradient or stochastic gradient step. Therefore, despite working well in many practical applications, all these mentioned methods can get stuck in local minima — and so are theoretically not well justified, see also the discussion in [DeC06].

The same issue also comes up for max-norm optimization, where for example [LRS+10] optimize over the non-convex factorization (4.8) for bounded max-norm.

Concerning the fixed rank of the factorization, [GG10] have shown that finding the optimum under a rank constraint (even if the rank is one) is

NP-hard (here the used function f was the standard squared error on an incomplete matrix). On the positive side, [BM03] have shown that if the rank k of the factors L and R exceeds the rank of the optimum solution X^*, then — in some cases — it can be guaranteed that the local minima (or saddle points) are also global minima. However, in nearly all practical applications it is computationally infeasible for the above mentioned methods to optimize with the rank k being in the same order of magnitude as the original matrix size m and n (as e.g. in the Netflix problem, such factors L, R could possibly not even be stored on a single machine[2]).

Relief: Optimizing Over an Equivalent Convex Problem. Here we simply overcome this problem by using the transformation to semidefinite matrices, which we have outlined in the above Corollary 4.2 and Lemma 4.3. These bijections of bounded nuclear and max-norm matrices to the PSD matrices over the corresponding natural convex domains do allow us to directly optimize a convex problem, avoiding the factorization problems explained above. We describe this simple trick formally in the next two Subsections 4.4.1 and 4.4.2.

But what if you really need a Matrix Factorization? In some applications (such as for example embeddings or certain collaborative filtering problems) of the above mentioned regularized optimization problems over $f(Z)$, one would still want to obtain the solution (or approximation) Z in a factorized representation, that is $Z = LR^T$. We note that this is also straight-forward to achieve when using our transformation: An explicit factorization of any feasible solution to the transformed problem (3.5) or (3.12) — if needed — can always be directly obtained since $X \succeq 0$.

Alternatively, algorithms for solving the transformed problem (3.5) can directly maintain the approximate solution X in a factorized representation (as a sum of rank-1 matrices), as achieved for example by Algorithms 6 and 7.

4.4.1. Optimization with a Nuclear Norm Regularization

Having Lemma 4.1 at hand, we immediately get to the crucial observation of this section, allowing us to apply Algorithm 6:

[2] Algorithm 6 in contrast does never need to store a full estimate matrix X, but instead just keeps the rank-1 factors v obtained in each step, maintaining a factorized representation of X.

Any optimization problem over bounded nuclear norm matrices (4.2) is in fact equivalent to a standard bounded trace semidefinite problem (3.5). The same transformation also holds for problems with a bound on the *weighted* nuclear norm, as given in (4.7).

Corollary 4.4. *Any nuclear norm regularized problem of the form (4.2) is equivalent to a bounded trace convex problem of the form (3.5), namely*

$$\begin{aligned} \underset{X \in \mathbb{S}^{(m+n)\times(m+n)}}{\text{minimize}} \quad & \hat{f}(X) \\ s.t. \quad & \operatorname{Tr}(X) = t \ , \\ & X \succeq 0 \end{aligned} \tag{4.10}$$

where $\hat{f}$ is defined by $\hat{f}(X) := f(Z)$ for $Z \in \mathbb{R}^{m\times n}$ being the upper right part of the symmetric matrix X. Formally we again think of $X \in \mathbb{S}^{(n+m)\times(n+m)}$ as consisting of the four parts $X =: \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix}$ with $V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n}$ and $Z \in \mathbb{R}^{m\times n}$.

Here "equivalent" means that for any feasible point of one problem, we have a feasible point of the other problem, attaining the same objective value. The only difference to the original formulation (3.5) is that the function argument X needs to be rescaled by $\frac{1}{t}$ in order to have unit trace, which however is a very simple operation in practical applications. Therefore, we can directly apply Hazan's Algorithm 6 for any max-norm regularized problem as follows:

Algorithm 8 Nuclear Norm Regularized Solver

Input: A convex nuclear norm regularized problem (4.2), target accuracy ε

Output: ε-approximate solution for problem (4.2)

1. Consider the transformed symmetric problem for $\hat{f}$, as given by Corollary 4.4

2. Adjust the function $\hat{f}$ so that it first rescales its argument by t

3. Run Hazan's Algorithm 6 for $\hat{f}(X)$ over the domain $X \in \mathcal{S}$.

Using our analysis of Algorithm 6 from Section 3.4.1, we see that Algorithm 8 runs in time *near linear* in the number N_f of non-zero entries of the gradient ∇f. This makes it very attractive in particular for recommender systems applications and matrix completion, where ∇f is a sparse matrix (same sparsity pattern as the observed entries), which we will discuss in more detail in Section 4.5.

Corollary 4.5. *After at most $O\left(\frac{1}{\varepsilon}\right)$ many iterations (i.e. approximate eigenvalue computations), Algorithm 8 obtains a solution that is ε close to the optimum of (4.2). The algorithm requires a total of $\tilde{O}\left(\frac{N_f}{\varepsilon^{1.5}}\right)$ arithmetic operations (with high probability).*

Proof. We use the transformation from Corollary 4.4 and then rescale all matrix entries by $\frac{1}{t}$. Then result then follows from Corollary 3.12 on page 56 on the running time of Hazan's algorithm. □

The fact that each iteration of our algorithm is computationally very cheap — consisting only of the computation of an approximate eigenvector — strongly contrasts the existing "proximal gradient" and "singular value thresholding" methods [GLW+09, JY09, MGC09, LST09, CCS10, TY10], which *in each step* need to compute an entire SVD. Such a single incomplete SVD computation (first k singular vectors) amounts to the same computational cost as an entire run of our algorithm (for k steps). Furthermore, those existing methods which come with a theoretical guarantee, in their analysis assume that all SVDs used during the algorithm are exact, which is not feasible in practice. By contrast, our analysis is rigorous even if the used eigenvectors are only ε'-approximate.

Another nice property of Hazan's method is that the returned solution is guaranteed to be simultaneously of low rank (k after k steps), and that by incrementally adding the rank-1 matrices $v_k v_k^T$, the algorithm automatically maintains a matrix factorization of the approximate solution.

Also, Hazan's algorithm, as being an instance of our general framework from Chapter 2, is designed to automatically stay within the feasible region $\mathcal{S}$, where most of the existing methods do need a projection step to get back to the feasible region (as e.g. [Lin07, LST09]), making both the theoretical analysis and implementation more complicated.

4.4.2. Optimization with a Max-Norm Regularization

The same approach works analogously for the max-norm, by using Lemma 4.3 in order to apply Algorithm 7:

Any optimization problem over bounded max-norm matrices (4.4) is in fact equivalent to a semidefinite problem (3.12) over the "box" of matrices where each element on the diagonal is bounded above by t. We think of this domain as generalizing the positive cube of vectors, to the PSD matrices.

Corollary 4.6. *Any max-norm regularized problem of the form (4.4) is equivalent to a bounded diagonal convex problem of the form (3.12), i.e.,*

$$\begin{aligned} \underset{X \in \mathbb{S}^{(m+n)\times(m+n)}}{\text{minimize}} \quad & \hat{f}(X) \\ s.t. \quad & X_{ii} \leq 1 \quad \forall i, \\ & X \succeq 0 \end{aligned} \tag{4.11}$$

where $\hat{f}$ is defined by $\hat{f}(X) := f(Z)$ for $Z \in \mathbb{R}^{m\times n}$ being the upper right part of the symmetric matrix X. Formally we again think of any $X \in \mathbb{S}^{(n+m)\times(n+m)}$ as consisting of the four parts $X =: \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix}$ with $V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n}$ and $Z \in \mathbb{R}^{m\times n}$.

Again the only difference to the original formulation (3.12) is that the function argument X needs to be rescaled by $\frac{1}{t}$ in order to have the diagonal bounded by one, which however is a very simple operation in practical applications. This means we can directly apply Algorithm 7 for any max-norm regularized problem as follows:

Algorithm 9 Max-Norm Regularized Solver

Input: A convex max-norm regularized problem (4.4), target accuracy ε

Output: ε-approximate solution for problem (4.4)

1. Consider the transformed symmetric problem for $\hat{f}$, as given by Corollary 4.6

2. Adjust the function $\hat{f}$ so that it first rescales its argument by t

3. Run Algorithm 7 for $\hat{f}(X)$ over the domain $X \in \boxplus$.

Using the analysis of our new Algorithm 7 from Section 3.4.1, we obtain the following guarantee:

Corollary 4.7. *After $\left\lceil \frac{8C_f}{\varepsilon} \right\rceil$ many iterations, Algorithm 9 obtains a solution that is ε close to the optimum of (4.4).*

Proof. We use the transformation from Corollary 4.6 and then rescale all matrix entries by $\frac{1}{t}$. Then the running time of the algorithm follows from Theorem 3.17. □

Maximum Margin Matrix Factorizations. In the case of matrix completion, the "loss" function f is defined as measuring the error from X to some

fixed observed matrix, but just at a small fixed set of "observed" positions of the matrices. As we already mentioned, semidefinite optimization over X as above can always be interpreted as finding a *matrix factorization*, as a symmetric PSD matrix X always has a (unique) Cholesky factorization.

Now for the setting of matrix completion, it is known that the above described optimization task under bounded max-norm, can be geometrically interpreted as learning a maximum margin separating hyperplane for each user/movie. In other words the factorization problem decomposes into a collection of SVMs, one for each user or movie, if we think of the corresponding other factor to be fixed for a moment [SRJ04]. We will discuss matrix completion in more detail in Section 4.5.

Other Applications of Max-Norm Optimization. Apart from matrix completion, optimization problems employing the max-norm have other prominent applications in spectral methods, spectral graph properties, low-rank recovery, and combinatorial problems such as Max-Cut.

4.5. Applications

Our Algorithm 8 directly applies to arbitrary nuclear norm regularized problems of the form (4.2). Since the nuclear norm is in a sense the most natural generalization of the sparsity-inducing ℓ_1-norm to the case of low rank matrices (see also the discussion in the previous chapters) there are many applications of this class of optimization problems.

4.5.1. Robust Principal Component Analysis

One prominent example of a nuclear norm regularized problem in the area of dimensionality reduction is given by the technique of *robust PCA* as introduced by [CLMW11], also called principal component pursuit, which is the optimization task

$$\min_{Z \in \mathbb{R}^{m \times n}} \|Z\|_* + \mu \|M - Z\|_1 \ . \tag{4.12}$$

Here $M \in \mathbb{R}^{m \times n}$ is the given data matrix, and $\|.\|_1$ denotes the entry-wise ℓ_1-norm. By considering the equivalent constrained variant $\|Z\|_* \leq \frac{t}{2}$ instead, we obtain a problem the form (4.2), suitable for our Algorithm 8. However, since the original objective function $f(Z) = \|M - Z\|_1$ is not differentiable, a smoothed version of the ℓ_1-norm has to be used instead.

This situation is analogous to the hinge-loss objective in maximum margin matrix factorization [SRJ04].

Existing algorithms for robust PCA do usually require a complete (and exact) SVD in each iteration, as e.g. [TY10, AGI11], and are often harder to analyze compared to our approach. The first algorithm with a convergence guarantee of $O\left(\frac{1}{\varepsilon}\right)$ was given by [AGI11], requiring a SVD computation per step. Our Algorithm 8 obtains the same guarantee in the same order of steps, but only requires a single approximate eigenvector computation per step, which is significantly cheaper.

Last but not least, the fact that our algorithm delivers approximate solutions to (4.12) of rank $O\left(\frac{1}{\varepsilon}\right)$ will be interesting for practical dimensionality reduction applications, as it re-introduces the important concept of low-rank factorizations as in classical PCA. In other words our algorithm produces an embedding into at most $O\left(\frac{1}{\varepsilon}\right)$ many new dimensions, which is much easier to deal with in practice compared to the full rank n solutions resulting from the existing solvers for robust PCA, see e.g. [CLMW11] and the references therein.

We did not yet perform practical experiments for robust PCA, but chose to demonstrate the practical performance of Algorithm 6 for matrix completion problems first.

4.5.2. Matrix Completion and Low Norm Matrix Factorizations

For matrix completion problems as for example in collaborative filtering and recommender systems [KBV09], our algorithm is particularly suitable as it retains the sparsity of the observations, and constructs the solution in a factorized way. In the setting of a partially observed matrix such as in the Netflix case, the loss function $f(X)$ only depends on the observed positions, which are very sparse, so $\nabla f(X)$ — which is all we need for our algorithm — is also sparse.

We want to approximate a partially given matrix Y (let Ω be the set of known training entries of the matrix) by a product $Z = LR^T$ such that some convex loss function $f(Z)$ is minimized. By Ω_{test} we denote the unknown test entries of the matrix we want to predict.

Complexity. Just recently it has been shown that the standard low-rank matrix completion problem — that is finding the best approximation to an incomplete matrix by the standard ℓ_2-norm — is an NP-hard problem,

if the rank of the approximation is constrained. The hardness is claimed to hold even for the rank 1 case [GG10].

In the light of this hardness result, the advantage of relaxing the rank by replacing it by the nuclear norm (or max-norm) is even more evident.

Our near linear time Algorithm 8 relies on a convex optimization formulation and does indeed deliver an guaranteed ε-accurate solution for the nuclear norm regularization, for arbitrary $\varepsilon > 0$. Such a guarantee is lacking for the "alternating" descent heuristics such as [RS05, Lin07, Pat07, ZWSP08, KBV09, TPNT09, IR10, GNHS11, SS10, LRS$^+$10, RR11] (which build upon the non-convex factorized versions (4.5) and (4.8) while constraining the rank of the used factors L and R).

Different Regularizations. Regularization by the weighted nuclear norm is observed by [SS10] to provide better generalization performance than the classical nuclear norm. As it can be simply reduced to the nuclear norm, see Section 4.2.1, our Algorithm 8 can directly be applied in the weighted case as well.

On the other hand, experimental evidence also shows that the max-norm sometimes provides better generalization performance than the nuclear norm [SRJ04, LRS$^+$10]. For any convex loss function, our Algorithm 9 solves the corresponding max-norm regularized matrix completion task.

Different Loss Functions. Our method applies to any convex loss function on a low norm matrix factorization problem, and we will only mention two loss functions in particular:

Maximum Margin Matrix Factorization (MMMF) [SRJ04] can directly be solved by our Algorithm 8. Here the original (soft margin) formulation is the trade-off formulation (4.1) with $f(Z) := \sum_{ij\in\Omega} |Z_{ij} - y_{ij}|$ being the hinge or ℓ_1-loss. Because this function is not differentiable, the authors recommend using the differentiable smoothed hinge loss instead.

When using the standard squared loss function $f(Z) := \sum_{ij\in\Omega}(Z_{ij} - y_{ij})^2$, the problem is known as *Regularized Matrix Factorization* [Wu07], and both our algorithms directly apply. This loss function is widely used in practice, has a very simple gradient, and is the natural matrix generalization of the ℓ_2-loss (notice the analogous Lasso and regularized least squares formulation). The same function is known as the rooted mean squared error, which was the quality measure used in the Netflix competition. We write $RMSE_{train}$ and $RMSE_{test}$ for the rooted error on the training ratings Ω and test ratings Ω_{test} respectively.

Running time and memory. From Corollary 4.5 we have that the running time of our nuclear norm optimization Algorithm 8 is linear in the size of the input: Each matrix-vector multiplication in Lanczos' or the power method exactly costs $|\Omega|$ (the number of observed positions of the matrix) operations, and we know that in total we need at most $O\left(1/\varepsilon^{1.5}\right)$ many such matrix-vector multiplications.

Also the memory requirements are very small: Either we store the entire factorization of $X^{(k)}$ (meaning the $O\left(\frac{1}{\varepsilon}\right)$ many vectors $v^{(k)}$) — which is still much smaller than the full matrix X — or then instead we can only update and store the prediction values $X_{ij}^{(k)}$ for $ij \in \Omega \cup \Omega_{test}$ in each step. This, together with the known ratings y_{ij} determines the sparse gradient matrix $\nabla f(X^{(k)})$ during the algorithm. Therefore, the total memory requirement is only $|\Omega \cup \Omega_{test}|$ (the size of the output) plus the size $(n+m)$ of a single feature vector v.

The constant C_f in the running time of Algorithm 6. One might ask if the constant hidden in the $O(\frac{1}{\varepsilon})$ number of iterations is indeed controllable. Here we show that for the standard squared error on any fixed set of observed entries Ω, this is indeed the case. For more details on the constant C_f, we refer the reader to Sections 2.3.4 and 3.4.1.

Lemma 4.8. *For the squared error $f(Z) = \frac{1}{2}\sum_{ij\in\Omega}(Z_{ij} - y_{ij})^2$ over the spectahedron $\mathcal{S}$, it holds that $C_{\hat{f}} \leq 1$.*

Proof. In Lemma 2.6, we have seen that the constant $C_{\hat{f}}$ is upper bounded by half the diameter of the domain, times the largest eigenvalue of the Hessian $\nabla^2 \hat{f}(\vec{X})$. Here we consider $\hat{f}$ as a function on vectors $\vec{X} \in \mathbb{R}^{n^2}$ corresponding to the matrices $X \in \mathbb{S}^{n\times n}$. However for the squared error as in our case here, the Hessian will be a diagonal matrix. One can directly compute that the diagonal entries of $\nabla^2 \hat{f}(\vec{X})$ are 1 at the entries corresponding to Ω, and zero everywhere else. Furthermore, the squared diameter of the spectahedron is upper bounded by 2, as we have shown in Lemma 3.15. Therefore $C_{\hat{f}} \leq 1$ for the domain $\mathcal{S}$. □

If the domain is the scaled spectahedron $t \cdot \mathcal{S}$ as used in our Algorithm 8, then the squared diameter of the domain is $2t^2$, compare to Lemma 3.15. This means that the curvature is upper bounded by $C_{\hat{f}} \leq t^2$ in this case. Alternatively, the same bound for the curvature of $\tilde{f}(X) := \hat{f}(tX)$ can be obtained along the same lines as for the spectahedron domain in the previous lemma, and the same factor of t^2 will be the scaling factor of the Hessian, resulting from the chain-rule for taking derivatives.

4.5.3. The Structure of the Resulting Eigenvalue Problems

For the actual computation of the approximate largest eigenvector in Algorithm 6, i.e. the internal procedure ApproxEV $\left(-\nabla\hat{f}(X^{(k)}), \frac{2C_{\hat{f}}}{k+2}\right)$, either Lanczos' method or the power method (as in PageRank, see e.g. [Ber05]) can be used. In our Theorem 3.10 of Section 3.4.1, we stated that both the power method as well as Lanczos' algorithm do provably obtain the required approximation quality in a bounded number of steps if the matrix is PSD, with high probability, see also [KW92, AHK05].

Both methods are known to scale well to very large problems and can be parallelized easily, as each iteration consists of just one matrix-vector multiplication. However, we have to be careful that we obtain the eigenvector for the *largest* eigenvalue which is not necessarily the principal one (largest in absolute value). In that case the spectrum can be shifted by adding an appropriate constant to the diagonal of the matrix.

For arbitrary loss function f, the gradient $-\nabla\hat{f}(X)$, which is the matrix whose largest eigenvector we have to compute in the algorithm, is always a symmetric matrix of the block form $\nabla\hat{f}(X) = \begin{pmatrix} 0 & G \\ G^T & 0 \end{pmatrix}$ for $G = \nabla f(Z)$, when $X =: \begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix}$. In other words $\nabla\hat{f}(X)$ is the adjacency matrix of a weighted *bipartite graph*. One vertex class corresponds to the n rows of the original matrix X_2 (*users* in recommender systems), the other class corresponds to the m columns (*items* or *movies*). It is easy to see that the spectrum of $\nabla\hat{f}$ is always *symmetric*: Whenever $\binom{v}{w}$ is an eigenvector for some eigenvalue λ, then $\binom{v}{-w}$ is an eigenvector for $-\lambda$.

Hence, we have exactly the same setting as in the established Hubs and Authorities (HITS) model [Kle99]. The first part of any eigenvector is always an eigenvector of the hub matrix G^TG, and the second part is an eigenvector of the authority matrix GG^T.

Repeated squaring. In the special case that the matrix G is very rectangular ($n \ll m$ or $n \gg m$), one of the two square matrices G^TG or GG^T is very small. Then it is known that one can obtain an exponential speed-up in the power method by repeatedly squaring the smaller one of the matrices, analogously to the "square and multiply"-approach for computing large integer powers of real numbers. In other words we can perform $O(\log\frac{1}{\varepsilon})$ many *matrix-matrix* multiplications instead of $O(\frac{1}{\varepsilon})$ *matrix-vector* multiplications.

4.5.4. Relation to Simon Funk's SVD Method

Interestingly, our proposed framework can also be seen as a theoretically justified variant of Simon Funk's [Web06] and related approximate SVD methods, which were used as a building block by most of the teams participating in the Netflix competition (including the winner team). Those methods have been further investigated by [Pat07, TPNT09] and also [KBC07], which already proposed a heuristic using the HITS formulation. These approaches are algorithmically extremely similar to our method, although they are aimed at a slightly different optimization problem, and do *not* directly guarantee bounded nuclear norm. Recently, [SS10] observed that Funk's algorithm can be seen as stochastic gradient descent to optimize (4.1) when the regularization term is replaced by a *weighted* variant of the nuclear norm.

Simon Funk's method considers the standard squared loss function $\hat{f}(X) = \frac{1}{2}\sum_{ij\in S}(X_{ij} - y_{ij})^2$, and finds the new rank-1 estimate (or feature) v by iterating $v := v + \lambda(-\nabla\hat{f}(X)v - Kv)$, or equivalently

$$v := \lambda\left(-\nabla\hat{f}(X) + \left(\frac{1}{\lambda} - K\right)\mathbf{I}\right)v\ , \tag{4.13}$$

a fixed number of times. Here λ is a small fixed constant called the learning rate. Additionally a decay rate $K > 0$ is used for regularization, i.e. to penalize the magnitude of the resulting feature v. This matrix-vector multiplication formulation (4.13) is equivalent to a step of the power method applied within our framework[3], and for small enough learning rates λ the resulting feature vector will converge to the largest eigenvector of $-\nabla\hat{f}(Z)$.

However in Funk's method, the magnitude of each new feature strongly depends on the starting vector v_0, the number of iterations, the learning rate λ as well as the decay K, making the convergence very sensitive to these parameters. This might be one of the reasons that so far no results on the convergence speed could be obtained. Our method is free of these parameters, the k-th new feature vector is always a unit vector scaled by $\frac{1}{\sqrt{k}}$. Also, we keep the Frobenius norm $\|U\|_{Fro}^2 + \|V\|_{Fro}^2$ of the obtained factorization exactly fixed during the algorithm, whereas in Funk's method — which has a different optimization objective — this norm strictly increases with every newly added feature.

[3] Another difference of our method to Simon Funk's lies in the stochastic gradient descent type of the latter, i.e. "immediate feedback": During each matrix multiplication, it already takes the modified current feature v into account when calculating the loss $\hat{f}(Z)$, whereas our Algorithm 6 alters Z only after the eigenvector computation is finished.

Our described framework therefore gives theoretically justified variant of the experimentally successful method [Web06] and its related variants such as [KBC07, Pat07, TPNT09].

4.6. Experimental Results

We run our algorithm for the following standard datasets[4] for matrix completion problems, using the squared error function.

dataset	*#ratings*	n	m
MovieLens 100k	10^5	943	1682
MovieLens 1M	10^6	6040	3706
MovieLens 10M	10^7	69878	10677
Netflix	10^8	480189	17770

Any eigenvector method can be used as a black-box in our algorithm. To keep the experiments simple, we used the power method[5], and performed $0.2 \cdot k$ power iterations in step k. If not stated otherwise, the only optimization we used is the improvement by averaging the old and new gradient as explained in Section 3.4.3. All results were obtained by our (single-thread) implementation in Java 6 on a 2.4 GHz Intel C2D laptop.

Sensitivity. The generalization performance of our method is relatively stable under changes of the regularization parameter, see Figure 4.1:

Movielens. Table 4.1 reports the running times of our algorithm on the three MovieLens datasets. Our algorithm gives an about 5.6 fold speed increase over the reported timings by [TY10], which is a very similar method to [JY09]. [TY10] already improves the "singular value thresholding" methods [CCS10] and [MGC09]. For MMMF, [RS05] report an optimization time of about 5 hours on the 1M dataset, but use the different smoothed hinge loss function so that the results cannot be directly compared. [MGC09], [SJ03] and [JY09] only obtained results on much smaller datasets.

In the following experiments on the MovieLens and Netflix datasets we have pre-normalized all training ratings to the simple average $\frac{\mu_i+\mu_j}{2}$ of

[4] See www.grouplens.org and archive.ics.uci.edu/ml.

[5] We used the power method starting with the uniform unit vector. $\frac{1}{2}$ of the approximate eigenvalue corresponding to the previously obtained feature v_{k-1} was added to the matrix diagonal to ensure good convergence.

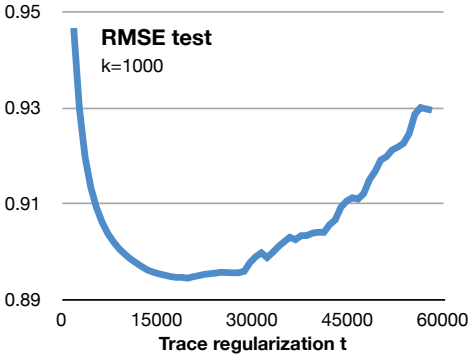


Figure 4.1.: Sensitivity of the method on the choice of the regularization parameter t in (4.2), on MovieLens 1M.

the user and movie mean values, for the sake of being consistent with comparable literature.

For MovieLens 10M, we used partition r_b provided with the dataset (10 test ratings per user). The regularization parameter t was set to 48333. We obtained a $RMSE_{test}$ of 0.8617 after $k = 400$ steps, in a total running time of 52 minutes (16291 matrix multiplications). Our best $RMSE_{test}$ value was 0.8573, compared to 0.8543 obtained by [LU09] using their non-linear improvement of MMMF.

Algorithm Variants. Comparing the proposed algorithm variants from Section 3.4.3, Figure 4.2 demonstrates moderate improvements compared to our original Algorithm 8.

Netflix. Table 4.2 compares our method to the two "hard impute" and "soft impute" singular value thresholding methods of [MHT10] on the Netflix dataset, where they used Matlab/PROPACK on an Intel Xeon 3 GHz processor. The "soft impute" variant uses a constrained rank heuristic in each update step, and an "un-shrinking" or fitting heuristic as post-processing. Both are advantages for their method, and were not used for our implementation. Nevertheless, our algorithm seems to perform competitive compared to the reported timings of [MHT10].

Note that the primary goal of this experimental section is *not* to compete

Table 4.1.: Running times t_{our} (in seconds) of our algorithm on the three MovieLens datasets compared to the reported timings t_{TY} of [TY10]. The ratings $\{1, \dots, 5\}$ were used as-is and *not* normalized to any user and/or movie means. In accordance with [TY10], 50% of the ratings were used for training, the others were used as the test set. Here *NMAE* is the mean absolute error, times $\frac{1}{5-1}$, over the total set of ratings. k is the number of iterations of our algorithm, $\#mm$ is the total number of sparse matrix-vector multiplications performed, and tr is the used trace parameter t in (4.2). They used Matlab/PROPACK on an Intel Xeon 3.20 GHz processor.

	NMAE	t_{TY}	t_{our}	k	$\#mm$	tr
100k	0.205	7.39	0.156	15	33	9975
1M	0.176	24.5	1.376	35	147	36060
10M	0.164	202	36.10	65	468	281942

Table 4.2.: Running times t_{our} (in hours) of our algorithm on the Netflix dataset compared to the reported timings $t_{\text{M,hard}}$ for "hard impute" by [MHT09] and $t_{\text{M,soft}}$ for "soft impute" by [MHT10].

$RMSE_{test}$	$t_{\text{M,hard}}$	$t_{\text{M,soft}}$	t_{our}	k	$\#mm$	tr
0.986	3.3	n.a.	0.144	20	50	99592
0.977	5.8	n.a.	0.306	30	109	99592
0.965	6.6	n.a.	0.504	40	185	99592
0.962	n.a.	1.36	1.08	45	243	174285
0.957	n.a.	2.21	1.69	60	416	174285
0.954	n.a.	2.83	2.68	80	715	174285
0.9497	n.a.	3.27	6.73	135	1942	174285
0.9478	n.a.	n.a.	13.6	200	4165	174285

with the prediction quality of the best engineered recommender systems (which are usually ensemble methods, i.e. combinations of many different individual methods). We just demonstrate that our method solves nuclear norm regularized problems of the form (4.2) on large sample datasets, obtaining strong performance improvements.

4.7. Conclusion

We have introduced a new method to solve arbitrary convex problems with a nuclear norm regularization, which is simple to implement and to

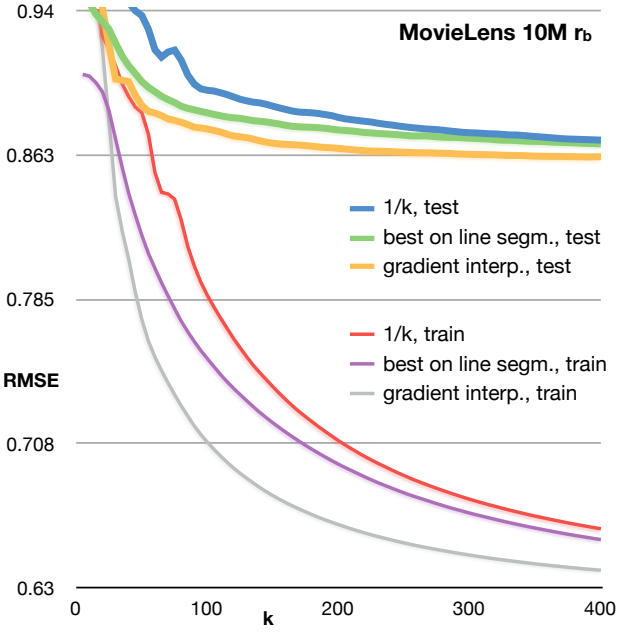


Figure 4.2.: Improvements for the two algorithm variants described in Section 3.4.3, when running on MovieLens 10M. The thick lines above indicate the error on the test set, while the thinner lines indicate the training error.

parallelize. The method is parameter-free and comes with a convergence guarantee. This guarantee is, to our knowledge, the first guaranteed convergence result for the class of Simon-Funk-type algorithms, as well as the first algorithm with a guarantee for max-norm regularized problems.

It remains to investigate if our algorithm can be applied to other matrix factorization problems such as (potentially only partially observed) low rank approximations to kernel matrices as used e.g. by the PSVM technique [CZW+07], regularized versions of latent semantic analysis (LSA), or non-negative matrix factorization [Wu07].

5

A Geometric Optimization Method, and Coresets for Polytope Distance and SVMs

In this chapter, we translate the coreset framework to the problems of finding the point closest to the origin inside a polytope, finding the shortest distance between two polytopes, Perceptrons, and soft- as well as hard-margin Support Vector Machines (SVM). To do so, we will study a geometric equivalent of Clarkson's optimization approach over the simplex [Cla10], which we have described in Chapters 2 and 3.

We prove asymptotically matching upper and lower bounds on the size of coresets, stating that ε-coresets of size $\lceil (1+o(1))E^*/\varepsilon \rceil$ do always exist as $\varepsilon \to 0$, and that this is optimal. The crucial quantity E^* is what we call the excentricity of a polytope, or a pair of polytopes, corresponding to the curvature constant in the general convex optimization setting in Chapter 2.

Additionally, we prove linear convergence speed of Gilbert's algorithm, one of the earliest known approximation algorithms for polytope distance, and generalize both the algorithm and the proof to the two polytope case.

Interestingly, our coreset bounds also imply that we can for the first time prove asymptotically matching upper and lower bounds for the sparsity of Perceptron and SVM solutions (which in this case is the number of support vectors), as a function of the approximation quality.

This chapter is based on the paper [GJ09] with Bernd Gärtner.

5.1. Introduction

Coresets. The concept of coresets has proven to be a very successful one for approximation algorithms for many discrete geometric problems. On one hand coreset algorithms are much faster than exact algorithms, and on the other hand they simultaneously ensure that the obtained approximate solutions still have very compact (sparse) representations, making them very appealing for many practical applications e.g. in machine learning.

Originally introduced for smallest enclosing ball problem and clustering by [BHPI02], and for extent measures by [APV02], the idea of a coreset is the following: instead of solving the original problem, one tries to identify a very small subset (coreset) of the points, such that the solution just on the coreset is guaranteed to be a good approximation of the true solution to the original problem. For the problem of finding the smallest enclosing ball of n points $P \in \mathbb{R}^d$, and $\varepsilon > 0$, an ε-coreset S is a small subset of the points P such that the smallest enclosing ball of just S, blown up by a factor of $1+\varepsilon$, contains all the original points P. It was shown that here ε-coresets of size $\lceil 1/\varepsilon \rceil$ do always exist [BC03, BC07] and that this is best possible [BC07]. This is very remarkable because the size of the coreset is independent of the dimension d of the space, and also independent of the number of points n, making it very attractive for the use in large scale problems (high n) and *kernel methods* (high d). This nice property is in contrast to many other geometric problems for which coresets usually have size exponential in the dimension, e.g. $\Theta(1/\varepsilon^{(d-1)/2})$ for the extent problem [AHPV04]. For a nice review on existing coreset algorithms we refer to [AHPV05].

Clarkson in [Cla10] significantly widened the class of problems where the coreset idea can be applied, and showed that the nice property of constant $O(1/\varepsilon)$ sized coresets indeed holds for the general problem of minimizing a convex function over the unit simplex. In the previous Chapter 2, we have generalized this coreset technique to optimizing over arbitrary compact domains, and have obtained matching lower bounds on the coreset size for

the simplex as well as for general ℓ_1-regularized convex problems.

Our Contributions and Related Work. Following the approach of [Cla10], we translate the coreset framework to the polytope distance problem of one polytope (w.r.t. the origin), distance between two polytopes, and hard- as well as soft-margin support vector machines, and introduce the geometric meaning of coresets and strong primal-dual approximation in this context.

We prove a new lower bound of $\left\lceil \frac{E^*}{\varepsilon} \right\rceil + 1$ for the size of ε-coresets for polytope distance, where E^* is what we call the *excentricity* of the polytope. Together with the upper bound of $\left\lceil \frac{E^*(1+o(1))}{\varepsilon} \right\rceil$ as $\varepsilon \to 0$, this shows that the size of the obtained ε-coresets is asymptotically optimal. We also show tight bounds (up to a factor of two) for the distance problem between two polytopes.

For Gilbert's algorithm [Gil66], one of the earliest known approximation algorithms for polytope distance, we give the first two proofs of convergence speed: First by observing that is in fact just an instance of the Frank Wolfe approximation algorithm for quadratic programs [FW56], which is now often called *sparse greedy approximation*, and secondly by giving a slightly easier geometric interpretation of the analytic proof of [Cla10]. Also, we generalize Gilbert's algorithm to the distance problem between two polytopes, where we are able to prove the same convergence speed. Furthermore, we can get rid of the expensive search for a starting point in this case which in previous approaches needed time quadratic in the number of points [Roo00, VSM03].

Applications to Machine Learning. On the application side, it is our goal to apply concepts and algorithms from computational geometry to machine learning. *Support Vector Machines* (SVM) [BGV92, CV95, Bur98] are among the most established and successful classification tools in machine learning, where from the name it is not immediately clear that the concept refers to nothing else than the separation of two classes of points by a hyperplane, with the largest possible margin. From the formulation as a quadratic program it follows that the problem is equivalent to the polytope distance problem, either for one or for two polytopes, depending on which SVM variant is considered (See Section 5.5). The *Perceptron* [Ros58] refers to the case where we search for any hyperplane that separates two point classes, not necessarily one of maximum margin. The term *kernel methods* summarizes SVMs and Perceptrons where the points are assumed to live in an implicit high-dimensional feature space where we just know their

pairwise scalar-products which is then called the *kernel*.

Sparsity of solutions. Our main contribution is to relate the coreset concept to *sparsity* of solutions of kernel methods: Using our bounds for the size of coresets, we derive a new fundamental property of SVMs and Perceptrons, giving nearly matching upper and lower bounds on the sparsity of their solutions, a parameter which is absolutely crucial for the practical performance of these methods on large scale problems. More precisely we show that any solution for a SVM or Perceptron, attaining at least a fraction μ of the optimal margin, must have at least $\left\lceil \frac{E^*}{1-\mu} \right\rceil + 1$ many (or $\left\lceil \frac{\frac{1}{2}E^*}{1-\mu} \right\rceil + 2$ in the two class case) non-zero coefficients in the worst case, and that a solution with $\left\lceil \frac{E^*(1+o(1))}{1-\mu} \right\rceil$ many non-zero coefficients can always be obtained for all instances. We are not aware of any existing lower bounds on the sparsity in the literature.

Training SVM in linear time. For any fixed fraction $0 \leq \mu < 1$, we show that Gilbert's algorithm in time $O(n)$ finds a solution attaining at least a μ-fraction of the optimal margin to the SVM and Perceptron (no matter if a kernel is used or not). This guarantee contrasts most of the existing SVM training algorithms which run in time usually cubic in n, or then often have no theoretical approximation guarantees except from converging in a finite number of steps [Pla99], or have guarantees only on the primal or dual objective value, but not both. Tsang et al. have already applied the smallest enclosing ball coreset approach to train SVMs under the name *Core Vector Machine* (CVM) [TKC05, TKK07], for one particular SVM variant (ℓ_2-loss with regularized offset), in the case that all points have the same norm. In this case the smallest enclosing ball problem is equivalent to finding the distance of one polytope from the origin. In another work [HPRZ07] directly used coresets to solve the problem of separating two polytopes by a hyperplane passing through the origin, but this is again equivalent to a one polytope distance problem. Both approaches are therefore generalized by [Cla10] and this work, proving faster algorithm convergence and smaller coresets. Here we generalize the coreset methods further to the two polytope case, encompassing all the currently most used hard- and soft-margin SVM variants with arbitrary kernels, with both ℓ_1 and ℓ_2-loss, in particular the special case of the CVM [TKC05, TKZ06, TKK07] and [HPRZ07], while obtaining faster convergence and smaller coresets. Our generalization shows that all of the mostly used SVM variants can be trained in time

linear in the number of sample points n, i.e. using linearly many kernel evaluations, for arbitrary kernels. Until now this was only known for the CVM case for radial base function kernels and for linear SVMs without using a kernel [Joa06].

5.2. Concepts and Definitions

5.2.1. Polytope Distance

Let $P \subset \mathbb{R}^d$ be a finite set of points. We want to compute the shortest distance $\rho = \rho(P)$ of any point inside the polytope $\text{conv}(P)$ to the origin. In the following we will assume $\rho > 0$.

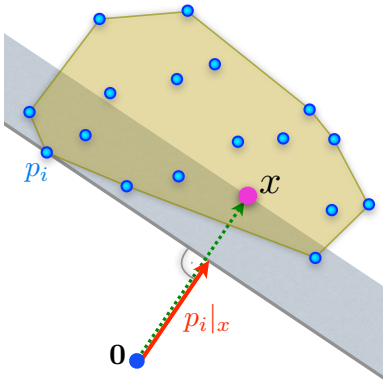
For $v, x \in \mathbb{R}^d$, Let $v|_x := \frac{\langle v,x \rangle}{\|x\|}$ denote the signed length of the *projection* of v onto the direction of the vector x. In this chapter, $\|.\|$ will always denote the Euclidean norm $\|.\|_2$.

Definition 5.1. *For any $\varepsilon > 0$,*

*i) A point $x \in \text{conv}(P)$ is called an ε-*approximation[1] *to the optimal polytope distance, iff*

$$\|x\| - p|_x \leq \varepsilon \|x\| \quad \forall p \in P \ .$$

*ii) A set of points $S \subseteq P$ with the property that the (optimal) closest point of $\text{conv}(S)$ to the origin is an ε-approximation to the distance of $\text{conv}(P)$ is called an ε-*coreset *of P.*



Definition 5.2.
The sparsity *of a convex combination $\sum_{i=1}^n \alpha_i p_i \in \text{conv}(P)$, $\sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0$ is the number of α_i that are non-zero.*

By definition, any ε-coreset of size s implies that we have an ε-approximation of sparsity at most s.

[1] Note that this is a *multiplicative* or relative approximation measure, where sometimes in the literature also *additive* ε-approximations are used. The corresponding coresets are sometimes called multiplicative ε-coresets to distinguish them from additive coresets [Cla10].

The Duality Gap as a Certificate of Approximation Quality. Being an ε-approximation can be interpreted as the multiplicative gap between the "primal" distance $f(x) = \|x\|$, and the corresponding "dual" value $\omega(x) = \min_{p \in P} p|_x$, being small, analogously to our concept of the "poor-man's" duality as described in Section 2.2. We will explain this connection in more detail in Section 5.2.3 below.

Also, the *weak-duality* observation (2.6), as we have seen for our general convex optimization setting from Section 2.2 has a clear geometric meaning here: Our geometric definition of an approximation automatically implies that the current distance $\|x\|$ must already be close to the (unknown) optimal value ρ, i.e.

Lemma 5.3. *If x is an ε-approximation, then* $(1-\varepsilon)\|x\| \leq \rho \leq \|x\|$.

Proof. RHS: Clear by definition of the distance ρ. *LHS:* By definition of an ε-approximation, we have a closed halfspace (normal to x), with distance $(1-\varepsilon)\|x\|$ from the origin, which contains $\text{conv}(P)$, which itself contains the optimal point x^* with $\|x^*\| = \rho$. □

Excentricity and Curvature.

Definition 5.4. *We define the* excentricity *of a point set P as $E := \frac{D^2}{\rho^2}$, where $D := \max_{p,q \in P} \|p-q\|$ is the diameter of the polytope and ρ is the true polytope distance to the origin.*

Also, we define the asymptotic excentricity *$E^* := \frac{R^2}{\rho^2}$, where we call $R := \max_{p \in P} \|p-c\|$ the radius of the polytope. Here c is the unique[2] point attaining the minimum distance ρ to the origin.*

It immediately follows that $E^* \leq E \leq 4E^*$ by triangle inequality ($R \leq D \leq 2R$).

Also, the quantities E and E^* do indeed correspond to the *curvature* or "non-linearity" introduced by [Cla10], that we studied in Section 2.3.4. We will explain this correspondence in the following Section 5.2.3.

5.2.2. Distance Between Two Polytopes

It is easy to see that the problem of finding the shortest distance between two polytopes is equivalent to finding the shortest vector in a single poly-

[2] We note that the point c attaining the optimal distance $\min_{p \in P} \|p\|$ of the polytope to the origin is always *unique*: Assume there would be two distinct points c_1, c_2 attaining the minimum, then their mid-point $\frac{1}{2}(c_1 + c_2)$ would have even shorter distance to the origin, contradicting the assumption of optimality of c_1.

tope, their Minkowski difference:

Definition 5.5. *The* Minkowski difference

$$\mathrm{MD}(P_1, P_2) := \{u - v \mid u \in \mathrm{conv}(P_1), v \in \mathrm{conv}(P_2)\}$$

of two polytopes $\mathrm{conv}(P_1)$ *and* $\mathrm{conv}(P_2)$ *is the set (in fact it is also a polytope [Zie95]) consisting of all difference vectors.*

Observe that $\mathrm{conv}(P_1)$ and $\mathrm{conv}(P_2)$ are separable by a hyperplane *iff* $\mathbf{0} \notin \mathrm{MD}(P_1, P_2)$. We call a vector $x = x_1 - x_2$ an *ε-approximation* for the distance problem between the two polytopes $\mathrm{conv}(P_1)$ and $\mathrm{conv}(P_2)$ iff x is an ε-approximation for $\mathrm{MD}(P_1, P_2)$. By the *sparsity* of a convex combination in $\mathrm{MD}(P_1, P_2)$ we always mean the minimum number of non-zero coefficients of a representation as a difference of two convex combinations in the original polytopes. An *ε-coreset* is a subset $P_1' \cup P_2'$ of the two original point sets, $P_1' \subseteq P_1, P_2' \subseteq P_2$, such that the shortest vector in the restricted Minkowski difference $\mathrm{MD}(P_1', P_2')$ is an ε-approximation.

Definition 5.6. *We define the* excentricity *of a pair of two polytopes as* $E_{P_1,P_2} := \frac{(D_1+D_2)^2}{\rho^2}$ *and the* asymptotic excentricity *as* $E^*_{P_1,P_2} := \frac{(R_1+R_2)^2}{\rho^2}$*, with* D_k*,* R_k *denoting diameter and radius*[3]*,* ρ *being the true distance between the two polytopes.*

Interpreting this excentricity in comparison to the equivalent single polytope problem given by the Minkowski difference, it is easy to see that $E_{\mathrm{MD}(P_1,P_2)} \leq E_{P_1,P_2}$, since $\mathrm{diam}(\mathrm{MD}(P_1, P_2)) \leq \mathrm{diam}(P_1) + \mathrm{diam}(P_2)$.

Alternatively, for comparison, we can also write $E_{P_1,P_2} = \frac{(D_1+D_2)^2}{\rho^2} = \left(\sqrt{E_1} + \sqrt{E_2}\right)^2$ and $E^*_{P_1,P_2} = \left(\sqrt{E_1^*} + \sqrt{E_2^*}\right)^2$ if we think of the E_k, E_k^* are being the "individual" excentricities of each polytope $\mathrm{conv}(P_k)$, $k = 1, 2$, here with respect to the corresponding closest point in the other polytope (if the closest pair is unique).

5.2.3. Relation to our General Setting of Convex Optimization Over Bounded Domain

Duality. Comparing the above geometric concept of approximation to our general optimization framework from Chapter 2, we observe that there is indeed a direct correspondence:

[3] Here the radius R_k is with respect to the corresponding endpoint of the optimal pair attaining the closest distance between the two polytopes. When using $E^*_{P_1,P_2}$, we will assume that the optimal pair is unique.

For the choice of convex objective function $f(x) := \|x\|$, the gradient is given by $\nabla f(x) = \frac{x}{\|x\|}$ for any $x \neq 0$. Therefore, the corresponding "poor-man's" dual value for each point x as given in equation (2.4) is in fact exactly the same geometric quantity as used in Definition 5.1,

$$\begin{aligned}\omega(x) = & \min_{p \in P} \|x\| + (p-x)^T \frac{x}{\|x\|} \\ = & \min_{p \in P} p^T \frac{x}{\|x\|} \\ = & \min_{p \in P} p|_x \ .\end{aligned}$$

Alternatively, one can also reduce the polytope distance problem to convex optimization over the simplex, as in Clarkson's framework [Cla10], by using barycentric coordinates, see also Section 3.1. In this case, the corresponding convex objective function is $f(x) := \|Ax\|$, where $A \in \mathbb{R}^{d \times n}$ is the matrix containing all points of P as columns.

Curvature. The above defined excentricity and the more general curvature constant C_f of a convex function are in fact closely related.

Lemma 5.7. *For the objective function $f(x) := \|x\|$ over the domain $D = \operatorname{conv}(P)$, the curvature of f is bounded by the excentricity as follows.*

$$C_f \leq \frac{\rho}{2} E \quad \text{and} \quad C_f^* \leq \frac{\rho}{2} E^* \ .$$

Proof. Using that $\nabla f(x) = \frac{x}{\|x\|}$ for any $x \neq 0$, we can consider the Hessian matrix $(\nabla^2 f(x))_{ij} = \frac{\partial f(x)}{\partial x_i \partial x_j}$, and calculate

$$\nabla^2 f(x) = \frac{\mathbf{I}}{\|x\|} - \frac{x x^T}{\|x\|^3} \ .$$

Here $\mathbf{I}$ is the $n \times n$ identity matrix. Plugging this expression into the upper bound for the curvature C_f as given by inequality (2.12) from Section 2.3.4, we obtain

$$\begin{aligned} C_f \leq & \sup_{\substack{x,y \in D, \\ z \in [x,y] \subseteq D}} \tfrac{1}{2}(y-x)^T \left(\tfrac{\mathbf{I}}{\|z\|} - \tfrac{zz^T}{\|z\|^3} \right)(y-x) \\ = & \sup_{\substack{x,y \in D, \\ z \in [x,y] \subseteq D}} \tfrac{1}{2} \left(\tfrac{\|y-x\|^2}{\|z\|} - \tfrac{(y-x)^T zz^T (y-x)}{\|z\|^3} \right) \\ = & \sup_{\substack{x,y \in D, \\ z \in [x,y] \subseteq D}} \tfrac{1}{2} \left(\tfrac{\|y-x\|^2 - ((y-x)|_z)^2}{\|z\|} \right) \end{aligned}$$

To obtain the middle equality, we have just rewritten the second term $\frac{(y-x)^T z z^T (y-x)}{\|z\|^3} = \frac{\left\|(y-x)^T \frac{z}{\|z\|}\right\|^2}{\|z\|} = \frac{((y-x)|_z)^2}{\|z\|}$ using the definition of the projection $v|_z$.

The desired upper bound $C_f \leq \frac{\rho}{2} E = \frac{D^2}{2\rho}$ now directly follows from just ignoring the negative term in the above expression of the supremum.

Analogously, when using the definition (2.11) of the asymptotic curvature C_f^*, together with the fact that the closest point x^* is the unique optimum, we have that

$$C_f^* \leq \sup_{\substack{y \in D, \\ z \in [x^*, y] \subseteq D}} \frac{1}{2} \left(\frac{\|y-x^*\|^2 - ((y-x^*)|_z)^2}{\|z\|} \right) ,$$

for which again ignoring the projection term, the optimum choice of $z := x^*$ will result in the bound $C_f^* \leq \frac{\rho}{2} E^* = \frac{R^2}{2\rho}$. □

5.3. Lower Bounds on the Sparsity of ε-Approximations

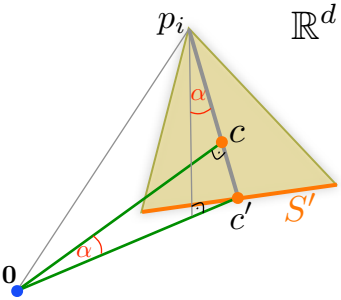
In this Section we will give two constructions of point sets, such that no small ε-coresets can possibly exist for the polytope distance problem. The geometric interpretation of these constructions is in fact very simple:

5.3.1. Distance of One Polytope from the Origin

Warm-Up. Consider the $(d-1)$-simplex spanned by P being the set of the d standard unit vectors in $\mathbb{R}^d$, but imagine the simplex being just slightly blown up (scaled linearly) from its barycenter c. Then the projection of any point $p_i \in P$ onto any convex combination x of some of the other points becomes negative, implying that no ε-coresets of size $< d$ (and also no ε-approximations of sparsity $< d$) can exist for any $\varepsilon \leq 1$. We will make this more formal in the following lemma. The unit simplex has $D = \sqrt{2}$ and $\rho = \frac{1}{\sqrt{d}}$ so the excentricity of our polytope $\operatorname{conv}(P)$ is roughly $E = 2d$, or $E^* = d - 1$ respectively. In the following we will see that the quantities $\frac{1}{2}E$ and $E^* + 1$ can indeed be turned into lower bounds on the size of ε-coresets and the sparsity of ε-approximations, in terms of the approximation quality ε. The following lemma explains the construction:

Lemma 5.8. *For any given $0 < \varepsilon < 1$, and for any $d \geq 2$, there exists a set of d points $P \subset \mathbb{R}^d$, such that*

i) any ε-coreset of P has size d (i.e. no strict subset can possibly be an ε-coreset).
ii) any ε-approximation of P has sparsity exactly d.
iii) any vector $x \in \mathrm{conv}(P)$, satisfying $\frac{p|_x}{\rho} \geq 1-\varepsilon \ \ \forall p \in P$, has sparsity exactly d.
iv) the excentricity of $\mathrm{conv}(P)$ is $E = 2\varepsilon d$ and the asymptotic excentricity is $E^* = \varepsilon(d-1)$.



Proof. By definition we already know that i) $\Leftarrow$ ii) $\Leftarrow$ iii), but we will prove the former statement first:

i) Let $\lambda > 0$ be a real parameter to be fixed later, and let our points be $p_j := \lambda \mathbf{e}_j + (1-\lambda)c \in \mathbb{R}^d$ for $j \in [d]$, with the barycenter $c = (\frac{1}{d}, \ldots, \frac{1}{d})^T$ being the point closest to the origin of the standard $(d-1)$-simplex in $\mathbb{R}^d$. I.e. we just linearly scale the unit simplex from its barycenter.

We will now show that for this particular set of points P, for some suitable choice of λ, no strict subset can possibly be an ε-coreset. To do so, let p_i be an arbitrary point of P and denote by c' the barycenter $c' := \frac{1}{d-1}\sum_{j \neq i} p_j$ which is the point closest to the origin of the sub-simplex $S' := \mathrm{conv}(P \setminus \{p_i\})$.

By definition we have $\|c'-c\|^2 = \frac{\lambda^2}{d(d-1)}$, $\|c'-p_i\|^2 = \frac{\lambda^2 d}{d-1}$ and $\|c'\|^2 = \frac{d-1+\lambda^2}{d(d-1)}$. From planar geometry in the triangle c', p_i and the origin we have that $\sin(\alpha) := \frac{\|c'\| - p_i|_{c'}}{\|c'-p_i\|} = \frac{\|c'-c\|}{\|c'\|}$, which implies $1 - \frac{p_i|_{c'}}{\|c'\|} = \frac{\|c'-c\|\|c'-p_i\|}{\|c'\|^2}$ $= \frac{\sqrt{\frac{\lambda^2}{d(d-1)}}\sqrt{\frac{d\lambda^2}{d-1}}}{\frac{d-1+\lambda^2}{d(d-1)}} = \frac{d\lambda^2}{d-1+\lambda^2}$.

Now if we choose our parameter $\lambda := \sqrt{\varepsilon}$, the above term on the right hand side is $\frac{d\varepsilon}{d-1+\varepsilon} > \varepsilon$. In other words we now have that $\|c'\| - p_i|_{c'} > \varepsilon \|c'\|$, so we have shown that no strict subset of the points can possibly be an ε-coreset.

ii) Using the construction above, we have shown that the barycenter $c' \in S'$ is not an ε-approximation, because it results in an insufficient approximation ratio $\frac{p_i|_{c'}}{\|c'\|} < 1-\varepsilon$. Now for every other point $x \in S'$, we can

show that the ratio becomes even worse, if we argue as follows: In the denominator we know that c' is the point in S' of minimum norm, and in the numerator it holds that $p_i|_{c'} \geq p_i|_x \quad \forall x \in S'$. The last inequality follows from the fact that the distance of a point x to a linear space is always at most as large as the distance to a subspace of it — or more formally if $p^{(x)}$, $p^{(c')}$ are the two projections of p_i onto x and c' respectively, we get that $\left\|p_i - p^{(x)}\right\| \geq \left\|p_i - p^{(c')}\right\|$ since $x \in \text{lin}(S')$ and $p_i|_{c'} = \left\|p^{(c')}\right\| = p_i|_{\text{lin}(S')}$. But by the Pythagorean theorem this implies $p_i|_x \leq p_i|_{c'}$. We have shown that no ε-approximation of sparsity $< d$ can possibly exist for our given point set.

iii) Let again $\lambda^2 := \varepsilon$, and suppose $x \in S' = \text{conv}(P \setminus \{p_i\})$ for some $p_i \in P$ is such a convex combination of sparsity $\leq d-1$. We use the above result and calculate $\frac{p_i|_x}{\rho} \leq \frac{p_i|_{c'}}{\rho} = \|c'\|\,(1 - \frac{d\varepsilon}{d-1+\varepsilon})\sqrt{d} = \sqrt{\frac{d-1+\varepsilon}{d(d-1)}\,\frac{(d-1)(1-\varepsilon)}{d-1+\varepsilon}}\sqrt{d} = \sqrt{\frac{d-1}{d-1+\varepsilon}}(1-\varepsilon) < 1 - \varepsilon$.

iv) It is straightforward to calculate that our point set has diameter $D^2 = \|p_1 - p_2\|^2 = 2\lambda^2$, radius $R^2 = \lambda^2\frac{d-1}{d}$, and true distance $\rho^2 = \|c\|^2 = \frac{1}{d}$, so for the excentricity we obtain $E = 2\varepsilon d$ and $E^* = \varepsilon(d-1)$. □

Theorem 5.9. *For any given $0 < \varepsilon < 1$, for any $d \geq 2$, there exists a set of d points $P \subset \mathbb{R}^d$, such that the sparsity of any ε-approximation, and the size of any ε-coreset of P is at least*

$$\left\lceil \frac{\frac{1}{2}E}{\varepsilon} \right\rceil \quad \text{and} \quad \left\lceil \frac{E^*}{\varepsilon} \right\rceil + 1$$

Proof. The point set P from Lemma 5.8 satisfies $\frac{\frac{1}{2}E}{\varepsilon} = d$, and $\frac{E^*}{\varepsilon} = d - 1$. □

Note that the bound using E is by a factor of 2 better than if we would just have used the trivial bound $E \leq 4E^*$ together with the result for E^*.

5.3.2. Distance Between Two Polytopes

Observation 5.10. *If we have two point sets, one consisting of just one point and the other consisting of d points, and we consider the polytope distance problem between the corresponding two polytopes, the lower bound of Lemma 5.8 directly applies to the Minkowski difference, resulting in the lower bounds $\left\lceil \frac{\frac{1}{2}E}{\varepsilon} \right\rceil + 1$ and $\left\lceil \frac{E^*}{\varepsilon} \right\rceil + 2$ because we always need the single point of the second class in any linear combination. In this case the pair*

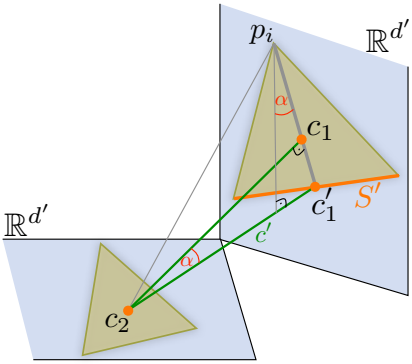
excentricities $E_{P_1,P_2}, E^*_{P_1,P_2}$ coincide with the single polytope excentricities E, E^* of the Minkowski difference.

However, we can generalize the lower bound construction of Lemma 5.8 to the distance problem between two polytopes spanned by equally sized point classes:

Lemma 5.11. *For any given $0 < \varepsilon < 1$, for any $d \geq 2$, there exist two equally sized point sets $P_1, P_2 \subset \mathbb{R}^{d=2d'}$, each consisting of d' points, such that*

- *i) any ε-coreset of $\mathrm{MD}(P_1, P_2)$ has size d (i.e. no strict subset can possibly be an ε-coreset).*
- *ii) any ε-approximation of $\mathrm{MD}(P_1, P_2)$ has sparsity exactly d.*
- *iii) any vector $x \in \mathrm{MD}(P_1, P_2)$ satisfying $\frac{p|_x}{\rho} \geq 1-\varepsilon \ \ \forall p \in \mathrm{MD}(P_1, P_2)$ has sparsity d.*
- *iv) the excentricity of the polytope pair is $E_{P_1,P_2} = 8\varepsilon d'$ or $E^*_{P_1,P_2} = 4\varepsilon(d'-1)$ respectively.*

Proof. By definition we already know that i) $\Leftarrow$ ii) $\Leftarrow$ iii), but we will prove the former statement first:



Consider the two point classes $P_1 = \{p_1 \dots p_{d'}\}$ and $P_2 = \{p_{d'+1} \dots p_{2d'}\}$ living in $\mathbb{R}^d$, $d = 2d'$,

with

$$p_i := \begin{cases} \lambda \mathbf{e}_i + (1-\lambda)c_1 & \text{for } 1 \leq i \leq d' \\ \lambda \mathbf{e}_i + (1-\lambda)c_2 & \text{for } d'+1 \leq i \leq d \end{cases}$$

where $c_1 := (\frac{1}{d'}, \dots, \frac{1}{d'}, 0, \dots, 0)$ and $c_2 := (0, \dots, 0, \frac{1}{d'}, \dots, \frac{1}{d'})$. I.e. we have two 'copies' of scaled unit simplices. It is not hard to see that the shortest vector in the Minkowski difference is $c_1 - c_2$.

i) Analogously to the single-polytope case of Lemma 5.8, we will now show that for suitable λ, no strict subset of $P_1 \cup P_2$ is an ε-coreset of $\mathrm{MD}(P_1, P_2)$. To do so, let p_i be an arbitrary point of P_1 and define $c' :=$

$c_1' - c_2$ where the barycenter $c_1' := \frac{1}{d'-1}\sum_{j\neq i} p_j$ is the point of the subsimplex $S' := \mathrm{conv}(P_1 \setminus \{p_i\})$ closest to c_2. It is easy to check that c' is indeed the new shortest distance after removal of p_i.

By definition we have $\|c_1' - c_1\|^2 = \frac{\lambda^2}{d'(d'-1)}$, $\|c_1' - p_i\|^2 = \frac{\lambda^2 d'}{d'-1}$ and $\|c'\|^2 = \frac{2(d'-1)+\lambda^2}{d'(d'-1)}$. From planar geometry in the triangle c_1', p_i and c_2 we again have that
$\sin(\alpha) := \frac{\|c'\| - (p_i-c_2)|_{c'}}{\|c_1'-p_i\|} = \frac{\|c_1'-c_1\|}{\|c'\|}$, which implies $1 - \frac{(p_i-c_2)|_{c'}}{\|c'\|}$
$= \frac{\|c_1'-c_1\|\|c_1'-p_i\|}{\|c'\|^2} = \frac{\sqrt{\frac{\lambda^2}{d'(d'-1)}}\sqrt{\frac{d'\lambda^2}{d'-1}}}{\frac{2(d'-1)+\lambda^2}{d'(d'-1)}} = \frac{d'\lambda^2}{2(d'-1)+\lambda^2}$.

Now if we choose our parameter $\lambda := \sqrt{2\varepsilon}$, the above term on the right hand side is $\frac{d'2\varepsilon}{2(d'-1)+2\varepsilon} > \varepsilon$. In other words we now have that $\|c'\| - (p_i - c_2)|_{c'} > \varepsilon\|c'\|$, so we have shown that no strict subset of the $2d'$ points can possibly be an ε-coreset.

ii) The argument that the approximation ratio $\frac{p|_x}{\|x\|}$ is indeed best for the distance vector $x := c'$ — and thus there really is no ε-approximation of sparsity $< d$ — is the same as in the proof of Lemma 5.8, and additionally using that $(p_i - y_2)|_{c'} = (p_i - c_2)|_{c'} \quad \forall y_2 \in \mathrm{conv}(P_2)$, as c' is orthogonal on $\mathrm{conv}(P_2)$.

iii) Let again $\lambda^2 := 2\varepsilon$, and suppose $x \in S' = \mathrm{conv}(P_1 \setminus \{p_i\})$ for some $p_i \in P_1$ is such a convex combination of sparsity $\leq d-1$. We use the above result and calculate $\frac{(p_i-c_2)|_x}{\rho} \leq \frac{(p_i-c_2)|_{c'}}{\rho} = \|c'\|\,(1 - \frac{d'2\varepsilon}{2(d'-1)+2\varepsilon})\sqrt{\frac{d'}{2}}$
$= \sqrt{\frac{2(d'-1)+2\varepsilon}{d'(d'-1)}}\,\frac{(d'-1)(1-\varepsilon)}{d'-1+\varepsilon}\sqrt{\frac{d'}{2}} = \sqrt{\frac{d'-1}{d'-1+\varepsilon}}(1-\varepsilon) < 1-\varepsilon$.

iv) Check that each of our two polytopes has diameter $D^2 = 2\lambda^2$ and radius $R^2 = \lambda^2\frac{d'-1}{d'}$. Since the optimal distance $\rho^2 = \frac{2}{d'}$, it follows that the pair excentricity is $E_{P_1,P_2} = \frac{(\sqrt{2}\lambda+\sqrt{2}\lambda)^2}{2/d'} = \frac{8\lambda^2}{2/d'} = 8\varepsilon d'$ and the asymptotic pair excentricity is $E^*_{P_1,P_2} = \frac{\left(\sqrt{\frac{d'-1}{d'}}\lambda+\sqrt{\frac{d'-1}{d'}}\lambda\right)^2}{2/d'} = \frac{4\frac{d'-1}{d'}\lambda^2}{2/d'} = 4\varepsilon(d'-1)$. □

Theorem 5.12. *For any given $0 < \varepsilon < 1$, for any $d \geq 2$, there exist two equally sized point sets $P_1, P_2 \subset \mathbb{R}^{d=2d'}$, each consisting of d' points, such that the sparsity of any ε-approximation of $\mathrm{MD}(P_1, P_2)$, and the size of any ε-coreset is at least*

$$\left\lceil \frac{\frac{1}{4}E_{P_1,P_2}}{\varepsilon} \right\rceil \quad \text{and} \quad \left\lceil \frac{\frac{1}{2}E^*_{P_1,P_2}}{\varepsilon} \right\rceil + 2\ .$$

Proof. P_1 and P_2 from Lemma 5.11 satisfy $\frac{\frac{1}{4}E_{P_1,P_2}}{\varepsilon} = 2d' = d$, and $\frac{\frac{1}{2}E^*_{P_1,P_2}}{\varepsilon} + 2 = 2(d'-1)+2 = d$. □

5.4. Upper Bounds: Algorithms to Construct Coresets

5.4.1. Gilbert's Algorithm

In Section 3.1, we already studied the general sparse greedy Algorithm 1 for bounded convex domains. Here we will see that following geometric algorithm is a special case of our Algorithm 1. The algorithm below was originally introduced by Frank and Wolfe [FW56] as an approximation algorithm for quadratic programs. Since then, it has independently been proposed again several times under different names. For polytope distance, it is known as Gilbert's algorithm [Gil66]. As we saw in Section 3.1, this algorithm provides ε-approximations of sparsity $O(\frac{1}{\varepsilon})$ for any convex minimization problem on the standard simplex [Cla10].

Algorithm 10 Gilbert's approximation algorithm for polytope distance [Gil66]

Input: Polytope P
Output: ε-approximate solution to the polytope distance problem
Start with $x_1 := p_0$, $p_0 \in P$ being the closest point to the origin.
for $i = 1 \dots \infty$ **do**
  Find the point $p_i \in P$ with smallest projection $p_i|_{x_i}$, and move to x_{i+1} being the point on the line segment $[x_i, p_i]$ which is closest to the origin.
  We stop as soon as x_{i+1} is an ε-approximation.
end for

Note that in order to run the algorithm, we only need to compute the projections of all points onto a given direction, and find the point closest to the origin on a line. Both can easily be achieved by evaluating scalar products, thus the algorithm works fine for *kernel methods*. Also, it can directly run on the Minkowski difference for the two polytope case.

Correspondence to the General Optimization Framework. As we have mentioned in Section 5.2.3, the point p_i minimizing the projection $p_i|_{x_i}$

(which is then used as the step-direction in the above Algorithm 10) is exactly a solution to the linearized problem $\textsc{ExactLinear}(\nabla f(x_i), D)$, for the convex function $f(x) := \|x\|$ over the domain $D = \text{conv}(P)$. This means that Algorithm 10 in fact coincides with the general line-search Algorithm 2 for this choice of objective function and domain.

Variants and applications. Gilbert's geometric algorithm has been applied to SVM training in the case of hard-margin as well as soft-margin with both ℓ_2-loss [KSBM00] and ℓ_1-loss [MST06, MT06][4]. A variant of Gilbert's algorithm, the GJK algorithm, is one of the most popular algorithm for collision detection in 3 dimensional space [GJK88]. For SVM training, Gilbert's algorithm was recently also implemented on FPGA hardware [PB08].

Another important variant of this, called the MDM algorithm [MDM74], is in fact equivalent to one of the most used SVM training algorithms, SMO [Pla99, LBD08]. For SVM training, [KSBM00] obtained good experimental results with a combination of Gilbert's and the MDM algorithm.

Convergence speed and running time. All mentioned algorithms in the above paragraph have in common that they converge, were successfully applied in practice, but no convergence speed or bound on the running time has ever been proved so far. Here we prove the convergence speed for Gilbert's algorithm, on one hand by observing for the first time that it is nothing else than the Frank-Wolfe algorithm [FW56] applied to the standard quadratic programming formulation of polytope distance[5], and on the other hand by giving a new slightly easier geometric variant of recent proofs by [AST08, Cla10] and our Chapter 2 on the convergence speed of sparse greedy approximation.

For the following analysis, let $f_i := \|x_i\|$ be the current distance, $h_i := f_i - \rho$ be the primal error, and let $g_i := f_i - \omega_i$ denote the 'primal-dual'

[4]Note that [MST06, Proposition 3] is not true the way stated therein. The Minkowski sum of two reduced convex hulls is in general not the reduced hull of the Minkowski sum. However this does not effect the applicability of Gilbert's algorithm to soft margin ℓ_1-SVMs.

[5]The quadratic programming formulation is $\min_x f(x) = (Ax)^2, x_i \geq 0, \sum_i x_i = 1$ when A is the $d \times n$-matrix containing all points as columns. Then the gradient $\nabla f(x)^T = A^T(Ax)$ consists of the scaled projections of all points onto the current vector Ax, so the Frank-Wolfe algorithm's choice (see Algorithm 3 or [Cla10, Algorithm 1.1]) of the smallest coordinate of the gradient is equivalent to moving towards the point with minimum projection, as in Gilbert's Algorithm [Gil66].

gap of our current estimate, with $\omega_i := \min_{p \in P} p|_{x_i}$. The key fact enabling linear convergence is the following bound, originally due to [AST08]:

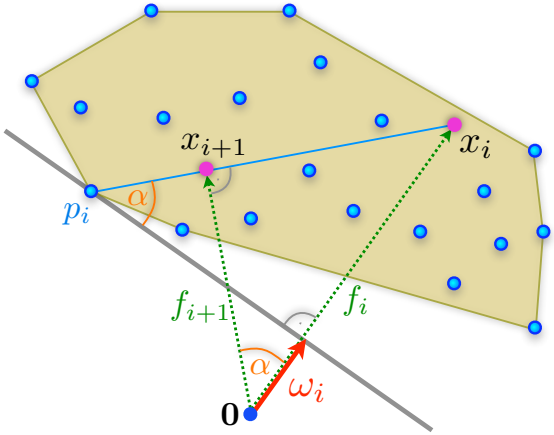


Figure 5.1.: A step of Gilbert's algorithm.

Lemma 5.13 (Geometric variant of Lemma 2.4 or [Cla10, Theorem 2.1])**.** *In each step of Gilbert's algorithm, the improvement in the primal error h_i is at least*

$$h_i - h_{i+1} \geq \frac{1}{2E\rho} g_i^2$$

Proof. Suppose x_{i+1} is perpendicular to the line segment $[x_i, p_i]$: Then we have $f_i - f_{i+1} = (1 - \cos\alpha) f_i$. Using the inequality[6] $1 - \cos\alpha \geq \frac{1}{2}\sin^2\alpha$, we get

$$(1 - \cos\alpha) f_i \geq \frac{1}{2}\sin^2\alpha f_i = \frac{g_i^2}{2\|p_i - x_i\|^2} f_i \tag{5.1}$$

but now we use the fact that since both x_i and p_i are inside $\mathrm{conv}(P)$, $\|p_i - x_i\|$ is at most D. Now we already have proven the claim, since by definition $f_i \geq \rho$:

$$h_i - h_{i+1} = f_i - f_{i+1} \geq \frac{\rho}{2D^2} g_i^2 = \frac{1}{2E\rho} g_i^2 \tag{5.2}$$

The case that x_{i+1} is at the endpoint p_i will in fact never occur: this would contradict the fact that the starting point for the algorithm was the point of shortest norm (since f_i decreases in each step). □

[6] This inequality is equivalent to $(1 - \cos\alpha)^2 \geq 0$.

The following theorem can be seen as a purely geometric analysis of the analogous primal-dual convergence Theorem 2.5 for the more general sparse greedy algorithm for convex optimization.

Theorem 5.14. *Gilbert's algorithm succeeds after at most* $2\left\lceil \frac{2E}{\varepsilon} \right\rceil$ *many steps.*

Proof. Using Lemma 5.13, we can now follow along the same lines as in our Theorem 2.5, or [Cla10, Theorem 2.3]: If we switch to a re-scaled version of the error-parameters, $h'_i := \frac{1}{2E\rho} h_i$, $g'_i := \frac{1}{2E\rho} g_i$, then the inequality becomes

$$h'_i - h'_{i+1} \geq {g'_i}^2 \geq {h'_i}^2 \tag{5.3}$$

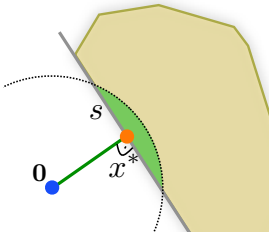
($g_i \geq h_i$ does always hold by definition) or equivalently $h'_{i+1} \leq h'_i(1 - h'_i)$: Plugging in $1 - \gamma \leq \frac{1}{1+\gamma}$ for $\gamma \geq -1$ gives $h'_{i+1} \leq \frac{h'_i}{1+h'_i} = \frac{1}{1+\frac{1}{h'_i}}$. Then by induction it is easy to obtain $h'_k \leq \frac{1}{k+1}$ and therefore $h'_k < \varepsilon'$ for $k \geq K := \left\lceil \frac{1}{\varepsilon'} \right\rceil$, if we can just show the induction hypothesis that $h'_1 \leq \frac{1}{2}$.

But this follows since our starting point is the point of shortest norm, which implies x_2 will always see x_1 and the origin in a right angle, therefore $\|x_1 - x_2\|^2 = f_1^2 - f_2^2 \leq D^2$, which implies $f_1 - f_2 \leq \frac{D^2}{2\rho}$ and therefore ${h'_1}^2 \leq {g'_1}^2 \leq h'_1 - h'_2 \leq \frac{1}{4}$ by using (5.3).

Now we have obtained small h'_i, but this does not necessarily imply yet that g'_i is also sufficiently small. For this we continue Gilbert's algorithm for another K steps, and suppose that in these subsequent steps, g' always remains $\geq \varepsilon'$, then we always have that $h'_i - h'_{i+1} \geq {\varepsilon'}^2$, and so $h'_K - h'_{2K} \geq K{\varepsilon'}^2 \geq \varepsilon'$, but this implies that $h'_{2K} < 0$, a contradiction. Thus for some $K \leq k \leq 2K$, we must have that $g'_k < \varepsilon'$.

If we choose our $\varepsilon' := \frac{1}{2E}\varepsilon$, we know that after at most $2K = 2\left\lceil \frac{2E}{\varepsilon} \right\rceil$ steps of the algorithm, the obtained primal-dual error is $g_k < \varepsilon\rho \leq \varepsilon f_k$, thus x_k is an ε-approximation. □

Observation 5.15. (Asymptotic convergence of Gilbert's Algorithm). *Note that if we are are already very close to the true solution* x^**, i.e. assume* $f_i - \rho$ *is small, say* $h_i = f_i - \rho < \gamma$ *for some* $\gamma > 0$*, then the inequality* $\|p_i - x_i\| \leq D$ *can be improved as follows: Observe that* x_i *is inside the optimal halfspace of distance* ρ *from the origin (as the entire polytope is), intersected with the ball of radius*



$\rho+\gamma$ around the origin. Let s be the furthest distance from x^* of any point in this intersection area. It is easy to see that $s = O(\sqrt{\gamma})$ is also small.

By triangle inequality we have $\|p_i - x_i\| \leq \|p_i - x^*\| + s$, so we get the stronger inequality $\|p_i - x_i\|^2 \leq R^2 + O(\sqrt{\gamma})$ in the proof of Lemma 5.13. Therefore, Gilbert's algorithm always succeeds in at most $2\left\lceil \frac{2E\rho}{\gamma} \right\rceil + 2\left\lceil \frac{2(E^*+O(\sqrt{\gamma}))}{\varepsilon} \right\rceil = 2\left\lceil \frac{2E^*(1+o(1))}{\varepsilon} \right\rceil$ many steps, as $\varepsilon \to 0$.

Note that the above analysis also proves the existence of ε-coresets of size $2\left\lceil \frac{2E}{\varepsilon} \right\rceil$ (and of size $2\left\lceil \frac{2E^*(1+o(1))}{\varepsilon} \right\rceil$ in the asymptotic notation), because the same improvement bound applies to the "exact" combinatorial algorithm [Cla10, Algorithm 4.1] that, when adding a new point to the set, computes the exact polytope distance of the new set.

5.4.2. An Improved Version of Gilbert's Algorithm for Two Polytopes

Coresets for the distance between two polytopes. All coreset methods for the single polytope case can directly be applied to the distance problem between two polytopes $\text{conv}(P^{(1)})$ and $\text{conv}(P^{(1)})$ by just running on the Minkowski difference $\text{MD}(P_1, P_2)$. This already makes the coreset approach available for all machine learning methods corresponding to a two polytope problem (as for example the standard SVM), see Section 5.5.

However, using the Minkowski difference has two major disadvantages: On one hand every vertex of $\text{MD}(P_1, P_2)$ always corresponds to two original vertices, one from each point class. Apart from potentially doubling the coreset size, this is a very unfortunate restriction if the shapes of the two point classes are very unbalanced, as e.g. in the *one-against-all* approach for multi-class classification, as it will create unnecessarily many non-zero coefficients in the smaller class. On the other hand, to run Gilbert's algorithm (or also the abstract version [Cla10, Algorithm 1.1] or the reduced hull variant [MST06]) on $\text{MD}(P_1, P_2)$, we have to determine the starting point of shortest norm and therefore have to consider all pairs of original points. Although this starting configuration was used in practice (see e.g. the DirectSVM [Roo00] and SimpleSVM [VSM03] implementations), this should definitely be avoided for large sets of points. We overcome both difficulties as follows:

An Improved Algorithm for Two Polytopes. The following modified algorithm maintains a difference vector $x_i^{(1)} - x_i^{(2)}$ between the two polytopes,

with $x_i^{(1)} \in \operatorname{conv}(P^{(1)})$ and $x_i^{(2)} \in \operatorname{conv}(P^{(2)})$. We again fix some notation first: Let $f_i := \left\| x_i^{(1)} - x_i^{(2)} \right\|$ be the current distance, $h_i := f_i - \rho$ be the primal error, and let $\omega_i := \min_{p \in P^{(1)}, q \in P^{(2)}} (p-q)|_{(x_i^{(1)} - x_i^{(2)})}$ be the 'dual' value. Then we can interpret $g_i^{(1)} := \max_{p \in P^{(1)}} (p - x_i^{(1)})|_{(x_i^{(2)} - x_i^{(1)})}$ and $g_i^{(2)} := \max_{p \in P^{(2)}} (p - x_i^{(2)})|_{(x_i^{(1)} - x_i^{(2)})}$ as being the two contributions to the 'primal-dual'-gap, so that $g_i := f_i - \omega_i = g_i^{(1)} + g_i^{(2)}$, see Figure 5.2.

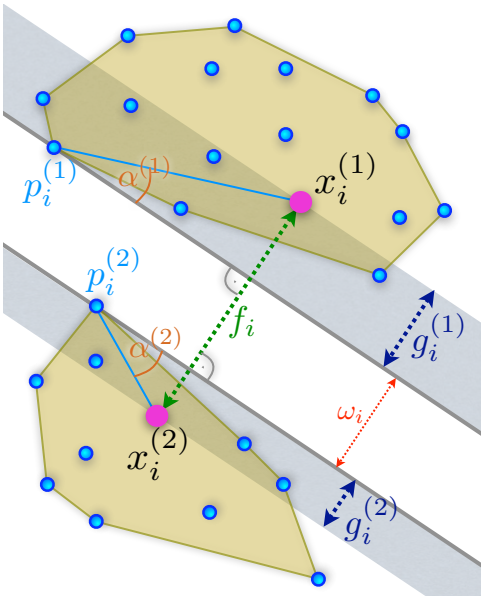


Figure 5.2.: A step of Algorithm 11.

Geometrically, the choice of $k \in \{1, 2\}$ in the algorithm corresponds to choosing the polytope for which the angle $\alpha^{(k)}$ is largest (see Figure 5.2), and in the following we will show how this is beneficial for the improvement in each step.

Rectangular steps and hit steps. If the new point $x_{i+1}^{(k)}$ lies in the interior of the line segment $[x_i^{(k)}, p_i^{(k)}]$, we say that this step is a *rectangular step*, as indicated e.g. in the upper polytope in Figure 5.2. However if the new point ends up at the endpoint $p_i^{(k)}$ of the line segment, then we say

Algorithm 11 Improved Gilbert's algorithm for distance between two polytopes

Input: Two polytopes $P^{(1)}$, $P^{(2)}$
Output: ε-approximate solution to the polytope distance problem
Start with an arbitrary point pair $(\, x_1^{(1)} \in P^{(1)},\ x_1^{(2)} \in P^{(2)}\,)$.
for $i = 1 \ldots \infty$ **do**
Find the points $p_i^{(1)} \in P^{(1)}$ and $p_i^{(2)} \in P^{(2)}$ with smallest projection $(p_i^{(1)} - p_i^{(2)})|_{(x_i^{(1)} - x_i^{(2)})}$, and now decide in which of the two polytopes to do a Gilbert step:
Choose the polytope $k \in \{1, 2\}$ for which the ratio $\frac{g_i^{(k)}}{\max\left\{\left\|x_i^{(k)} - p_i^{(k)}\right\|, \sqrt{g_i^{(k)} f_i}\right\}}$ is maximal, and move to $x_{i+1}^{(k)}$ being the point on the line segment $[x_i^{(k)}, p_i^{(k)}]$ which is closest to the "opposite" point $x_{i+1}^{(\bar{k})} := x_i^{(\bar{k})}$, which we keep unchanged.
We stop as soon as $x_i^{(1)} - x_i^{(2)}$ is an ε-approximation.
end for

that this is a *hit step* in polytope k, as e.g. in the lower polytope in Figure 5.2. It is not hard to see that a hit step in polytope k occurs if and only if $\left\|x_i^{(k)} - p_i^{(k)}\right\| \leq \sqrt{g_i^{(k)} f_i}$, and otherwise the step is rectangular. Note that in the single polytope case as in Lemma 5.13, hit steps are impossible by choice of the starting point, but here in the two polytope case this might indeed happen. From a computational perspective, hit steps are advantageous, as in each such step the number of points involved to describe the current approximation point inside one of the polytopes (called the number of *support vectors* in the SVM setting) decreases from a possibly large number down to one. However in the analysis of the algorithm, these hit steps pose some technical difficulties:

Lemma 5.16. *The improvement in the primal error h_i in each step of Algorithm 11 is either*

$$h_i - h_{i+1} \geq \frac{1}{2\rho E_{P_1,P_2}} g_i^2 \tag{5.4}$$

or

$$h_i - h_{i+1} \geq C_{P_1,P_2}\, g_i \tag{5.5}$$

for $C_{P_1,P_2} := \frac{\rho}{4(\rho + D^{(1)} + D^{(2)})} \left(\min \frac{D^{(1)}}{D^{(2)}}, \frac{D^{(2)}}{D^{(1)}}\right)^2$ *otherwise.*

Proof. Case R: In the case that the steps in both polytopes are rectangular we can follow the proof of Lemma 5.13: Assuming that the polytope chosen by the algorithm is k, we can follow (5.1) and (5.2) to get

$$h_i - h_{i+1} \geq \frac{\rho}{2}\left(\frac{g_i^{(k)}}{\left\|x_i^{(k)}-p_i^{(k)}\right\|}\right)^2 \geq \frac{\rho}{2}\left(\max_{m=1,2}\frac{g_i^{(m)}}{D^{(m)}}\right)^2$$
$$\geq \frac{\rho}{2}\left(\frac{g_i^{(1)}+g_i^{(2)}}{D^{(1)}+D^{(2)}}\right)^2 = \frac{1}{2\rho E_{P_1,P_2}} g_i^2 \,, \tag{5.6}$$

where we used $\max\left(\frac{a}{c},\frac{b}{d}\right) \geq \frac{a+b}{c+d} \quad \forall a,b,c,d \geq 0$, Definition 5.6 of the Excentricity, and that $g_i = g_i^{(1)} + g_i^{(2)}$.

Case M: In the "mixed" case that the step in one polytope (r) is rectangular, but in the other polytope (h) is a hit step (due to $\left\|x_i^{(h)} - p_i^{(h)}\right\|^2 \leq g_i^{(h)} f_i$), we can argue as follows:

Case M(r): If the algorithm chooses polytope (r), we can proceed analogously to (5.6) to obtain

$$h_i - h_{i+1} \geq \frac{\rho}{2}\frac{{g_i^{(r)}}^2}{\left\|x_i^{(r)}-p_i^{(r)}\right\|^2} = \frac{\rho}{2}\max\left\{\frac{{g_i^{(r)}}^2}{\left\|x_i^{(r)}-p_i^{(r)}\right\|^2}, \frac{{g_i^{(h)}}^2}{g_i^{(h)} f_i}\right\}, \tag{5.7}$$

but now there are two cases: If $g_i^{(h)} f_i \leq {D^{(h)}}^2$, then we use the same arithmetic trick as in (5.6),

$$h_i - h_{i+1} \geq \frac{\rho}{2}\left(\max_{m=1,2}\frac{g_i^{(m)}}{D^{(m)}}\right)^2 \geq \frac{1}{2\rho E_{P_1,P_2}} g_i^2. \tag{5.8}$$

However if $g_i^{(h)} f_i \geq {D^{(h)}}^2$, we have to argue differently: By the choice of the algorithm we know that $g_i^{(r)} \geq g_i^{(h)}$ because $\frac{g_i^{(r)}}{f_i} = \frac{{g_i^{(r)}}^2}{g_i^{(r)} f_i} \geq \frac{{g_i^{(r)}}^2}{\left\|x_i^{(r)}-p_i^{(r)}\right\|^2} \geq \frac{g_i^{(h)}}{f_i}$. As in (5.7), the improvement in the step can then be bounded by $h_i - h_{i+1} \geq \frac{\rho}{2}\frac{{g_i^{(r)}}^2}{\left\|x_i^{(r)}-p_i^{(r)}\right\|^2} \geq \frac{\rho}{2}\frac{{g_i^{(r)}}^2}{{D^{(r)}}^2} \geq \frac{\rho}{2}\frac{g_i^{(h)} g_i^{(r)}}{{D^{(r)}}^2}$ which by our assumption is $\geq \frac{\rho}{2}\frac{{D^{(h)}}^2}{{D^{(r)}}^2 f_i} g_i^{(r)} \geq \frac{\rho}{2}\frac{{D^{(h)}}^2}{{D^{(r)}}^2 f_i}\frac{1}{2}\left(g_i^{(r)} + g_i^{(h)}\right) \geq C_{P_1,P_2} g_i$. The last inequality follows because f_i is always smaller than $D^{(1)} + D^{(2)} + \rho$ by triangle inequality.

Case M(h): If the choice criterium is $\frac{{g_i^{(r)}}^2}{\left\|x_i^{(r)}-p_i^{(r)}\right\|^2} \leq \frac{{g_i^{(h)}}^2}{g_i^{(h)} f_i}$, then algorithm will choose the polytope (h) where a hit step will occur. The

improvement in a hit step is worst if $x_{i+1}^{(h)}$ ends up exactly at distance $\sqrt{f_i g_i^{(h)}}$ from $x_i^{(h)}$ — i.e. on the Thales ball over f_i — therefore $f_i^2 - f_{i+1}^2 \geq f_i g_i^{(h)} \Rightarrow h_i - h_{i+1} = f_i - f_{i+1} \geq \frac{1}{2} g_i^{(h)} \geq \frac{\rho}{2} \frac{g_i^{(h)}}{f_i}$. Now if we again assume that $g_i^{(h)} f_i \leq {D^{(h)}}^2$, then analogously to (5.7), (5.8) we have $h_i - h_{i+1} \geq \frac{\rho}{2} \frac{g_i^{(h)}}{f_i} = \frac{\rho}{2} \max\left\{ \frac{{g_i^{(r)}}^2}{\left\| x_i^{(r)} - p_i^{(r)} \right\|^2}, \frac{{g_i^{(h)}}^2}{g_i^{(h)} f_i} \right\} \geq \frac{1}{2\rho E_{P_1,P_2}} g_i^2$. On the other hand if $g_i^{(h)} f_i \geq {D^{(h)}}^2$, we distinguish two cases: First if $g_i^{(h)} \geq g_i^{(r)}$, then directly $h_i - h_{i+1} \geq \frac{1}{2} g_i^{(h)} \geq \frac{1}{4}(g_i^{(h)} + g_i^{(r)}) = \frac{1}{4} g_i \geq C_{P_1,P_2} g_i$. Secondly if $g_i^{(r)} \geq g_i^{(h)}$, we use that by the choice of the algorithm $\frac{g_i^{(h)}}{f_i} \geq \frac{{g_i^{(r)}}^2}{\left\| x_i^{(r)} - p_i^{(r)} \right\|^2} \geq \frac{{g_i^{(r)}}^2}{{D^{(r)}}^2}$, so we have $h_i - h_{i+1} \geq \frac{\rho}{2} \frac{{g_i^{(r)}}^2}{{D^{(r)}}^2} \geq C_{P_1,P_2} g_i$ analogous to the last part of the previous case M(r).

Case H: If there is a hit step in both polytopes, then we use that the algorithm has chosen the one with the larger value of $g_i^{(k)}$, therefore again by the "angle constraint" reasoning that for hit steps $f_i^2 - f_{i+1}^2 \geq f_i g_i^{(k)}$ we obtain $f_i - f_{i+1} \geq \frac{f_i}{2 f_i} g_i^{(k)} = \frac{1}{2} g_i^{(k)} \geq \frac{1}{4} g_i \geq C_{P_1,P_2} g_i$. □

Theorem 5.17. *Algorithm 11 succeeds after at most* $2 \left\lceil \frac{2E_{P_1,P_2}}{\varepsilon} \right\rceil + 3 + \frac{1}{C_{P_1,P_2}} \log \frac{D^{(1)} + D^{(2)}}{\rho\, C_{P_1,P_2}\, \varepsilon} = O(1/\varepsilon)$ *many steps.*

Proof. We count the steps of quadratic improvement (5.4) and those of linear improvement (5.5) separately, using that each kind of step results in strict improvement in the primal error h_i:

i) For quadratic improvement (5.4) we follow exactly along the proof of Theorem 5.14, for E_{P_1,P_2} being the pair excentricity: If we rescale by $h_i' := \frac{1}{2E_{P_1,P_2}\rho} h_i$, (5.4) gives $\frac{1}{h_{i+1}'} \geq 1 + \frac{1}{h_i'}$. Now we just use that initially h_1' is finite, therefore $\frac{1}{h_1'} \geq 0$ and by induction we get $\frac{1}{h_k'} \geq k - 1$ for all $k \geq 2$. It follows that $h_k' \leq \varepsilon'$ for $k \geq K := \left\lceil \frac{1}{\varepsilon'} \right\rceil + 1$. By the same argument as in the proof of Theorem 5.14, we have that after at most $2K = 2 \left\lceil \frac{2E_{P_1,P_2}}{\varepsilon} \right\rceil + 2$ many rectangular steps, $g_k < \varepsilon\rho \leq \varepsilon f_k$, thus $x_k^{(1)} - x_k^{(2)}$ is an ε-approximation.

ii) On the other hand we can bound the number of steps of linear improvement (5.5) by an easier argument: Let $C := C_{P_1,P_2}$. Now $h_i - h_{i+1} \geq C g_i \geq C h_i$ is equivalent to $h_{i+1} \leq (1 - C) h_i$ (recall that $0 < C < 1$). Using that initially $h_1 \leq D^{(1)} + D^{(2)}$, we get $h_k \leq (1 - C)^{k-1} h_1 \leq$

$(1-C)^{k-1}(D^{(1)}+D^{(2)})$ for all $k \geq 2$, which is $\leq \varepsilon'$ as soon as $k-1 \geq \frac{\log \frac{D^{(1)}+D^{(2)}}{\varepsilon'}}{-\log(1-C)}$, which in particular holds if $k-1 \geq \frac{1}{C}\log \frac{D^{(1)}+D^{(2)}}{\varepsilon'}$ by using the inequality $\lambda < -\log(1-\lambda)$ for $0 < \lambda < 1$. For $\varepsilon' := \rho\, C\, \varepsilon$, this is enough because $\varepsilon' \geq h_k \geq h_k - h_{k+1} \geq C g_k$ implies $g_k \leq \varepsilon\rho \leq \varepsilon f_k$, or in other words the algorithm obtains an ε-approximation after at most $\frac{1}{C}\log \frac{D^{(1)}+D^{(2)}}{\rho\, C\, \varepsilon} + 1$ many steps of linear improvement. □

Generalization of our algorithm for convex optimization over products of simplices. We can also generalize our above new variant of sparse greedy approximation in terms of the general framework by Clarkson [Cla10], when we extend it to solving any concave maximization problem over a product of finitely many simplices or convex hulls. To do so, we can prove the same step improvement (5.4) also for the case of convex functions defined on any product of simplices. We are currently investigating the details in this setting.

5.4.3. Smaller Coresets by "Away" Steps

Gilbert's algorithm and also its "exact" variant, due to their greedy nature, are not optimal in the size of the coresets they deliver. However, Clarkson showed that a modified procedure [Cla10, Algorithm 5.1] based on an idea by [TY07], called the away steps algorithm, obtains smaller coresets. The following Theorem, together with our lower bounds from Section 5.3.1, will settle the question on the size of coresets for the distance problem of one polytope to the origin, because the size of the coreset obtained by the algorithm matches our lower bound, and therefore is best possible:

Theorem 5.18. *For any $\varepsilon > 0$, the away steps algorithm [Cla10, Algorithm 5.1] returns an ε-coreset of size at most $\lceil \frac{E}{\varepsilon} \rceil$, and at most $\left\lceil \frac{E^*(1+o(1))}{\varepsilon} \right\rceil$ as $\varepsilon \to 0$.*

Proof. We consider the equivalent convex optimization description of the polytope distance problem by $f(x) = \|Ax\|$, as described in our Section 5.2.3. In this setting, [Cla10, Theorem 5.1] shows that the away steps algorithm delivers a subset of coordinate indices N of size $\lceil \frac{1}{\varepsilon'} \rceil$ such that $f(x_N) - \omega(x_N) \leq 2C_f\varepsilon'$, where x_N is the true optimal solution of the problem restricted to the coordinates in N. By the scaling $\varepsilon' := \frac{\rho\varepsilon}{2C_f}$, this means that in our setting, the away steps algorithm returns an ε-coreset of size $\left\lceil \frac{2C_f}{\rho\varepsilon} \right\rceil \leq \lceil \frac{E}{\varepsilon} \rceil$, and $\left\lceil \frac{2C_f^*(1+o(1))}{\rho\varepsilon} \right\rceil \leq \left\lceil \frac{E^*(1+o(1)}{\varepsilon} \right\rceil$ in the asymptotic case; the last two inequalities holding by Lemma 5.7. □

Away steps in the case of two polytopes. We can adjust [Cla10, Algorithm 5.1] for two polytopes as follows: Start with the *closest* point pair $(x_1^{(1)} \in P^{(1)},\ x_1^{(2)} \in P^{(2)})$, and proceed as in [Cla10, Algorithm 5.1]. In each step choose the polytope according to the choice criterium of our Algorithm 11, but with the modification that whenever a hit step is possible one either side, we choose to do this hit step.

Theorem 5.19. *For any $\varepsilon > 0$, the modified away steps algorithm on two polytopes returns an ε-coreset of size at most $\left\lceil \frac{E_{P_1,P_2}}{\varepsilon} \right\rceil$, and at most $\left\lceil \frac{E^*_{P_1,P_2}(1+o(1))}{\varepsilon} \right\rceil$ as $\varepsilon \to 0$.*

Proof. Sketch: We can follow the proof of [Cla10, Theorem 5.1] using our Lemma 5.16, and observe that a hit step never increases the coreset size. The key point is that for any step that increases the coreset size, the improvement bound (5.4) always holds. The induction hypothesis $h_1' := \frac{1}{2E_{P_1,P_2}\rho} h_1 \leq \frac{1}{2}$ follows if we start at the closest pair. □

5.5. Applications to Machine Learning

The advantage of the coreset approach is that both the running time of the algorithms and the sparsity of the obtained solutions is independent of the dimension d of the space and independent of the number of points n. This property makes the approach very attractive for kernel methods, where the points are implicitly assumed to live in a (possibly very high dimensional) feature space.

Table 5.5 briefly recapitulates the fact that nearly all well known SVM variants are equivalent to a polytope distance problem between either one or two polytopes, showing that all these variants do fit into our framework of coresets. This also holds for the soft-margin variants [CV95] if the punishment of the outliers is proportional to the squared distance to the classification hyperplane (ℓ_2-loss SVM). In the table, $x_i \in \mathbb{R}^d$, $1 \leq i \leq n$ denote the original points, $\phi(x_i)$ are their implicit images in the feature space defined by the kernel, and in the two class cases the labels of the points are given by $y_i = \pm 1$. ω and b are the normal and offset of the maximum margin hyperplane that we are searching for, and the ξ_i represent slack variables for the case of possible (punished) outliers.

	SVM Variant	Primal Problem	Equivalent Polytope Distance Formulation
1a	one-class SVM (hard-margin)	$\min_{w,\rho} \frac{1}{2}\|w\|^2 - \rho$ $w^T\phi(x_i) \geq \rho \quad \forall i$	one polytope
1b	one-class ℓ_2-SVM (soft-margin)	$\min_{w,\rho,\xi} \frac{1}{2}\|w\|^2 - \rho + \frac{C}{2}\sum_i \xi_i^2$ $w^T\phi(x_i) \geq \rho - \xi_i \quad \forall i$	one polytope [TKC05, Eqn. (8)]
1c	two-class ℓ_2-SVM (with regularized or no offset)	$\min_{w,b,\rho,\xi} \frac{1}{2}(\|w\|^2+b^2) - \rho + \frac{C}{2}\sum_i \xi_i^2$ $y_i(w^T\phi(x_i) - b) \geq \rho - \xi_i \ \forall i$	one polytope [TKC05, Eqn.(13)], [HPRZ07]
2a	two-class SVM, Perceptron (hard-margin)	$\min_{w,b} \frac{1}{2}\|w\|^2$ $y_i(w^T\phi(x_i) - b) \geq 1 \quad \forall i$	two polytopes
2b	two-class ℓ_2-SVM (standard version)	$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_i \xi_i^2$ $y_i(w^T\phi(x_i) - b) \geq 1 - \xi_i \ \forall i$	two polytopes [KSBM00, Sect. II]
2c	two-class ℓ_1-SVM (C-SVM)	$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$ $y_i(w^T\phi(x_i) - b) \geq 1 - \xi_i,$ $\xi_i \geq 0 \quad \forall i$	two (reduced) polytopes [BB00]
2d	two-class ℓ_1-SVM (ν-SVM)	$\min_{w,b,\rho,\xi} \frac{1}{2}\|w\|^2 - \rho + \frac{\nu}{2}\sum_i \xi_i$ $y_i(w^T\phi(x_i) - b) \geq \rho - \xi_i,$ $\xi_i \geq 0 \quad \forall i$	two (reduced) polytopes [CB00]

Table 5.1.: SVM variants and their equivalent polytope distance formulations.

5.5.1. Sparsity of SVM and Perceptron Solutions

The sparsity of kernel SVM and Perceptron solutions is *the* crucial ingredient for the performance of these methods on large scale problems: If we have an approximate solution ω, then still for every evaluation of the classifier (this means we are given a new "unseen" point and have to answer on which side of the hyperplane it lies), the scalar products to all points which appear with non-zero coefficient in ω (those are called the *support vectors*) have to be evaluated. The computational performance in practical use is therefore directly proportional to the sparsity of ω. Interestingly not much is known in the literature on this question, in particular no lower bounds are known to our knowledge. Using the above equivalences, we are now for the first time able to prove asymptotically matching upper and lower bounds for the sparsity of approximate SVM and Perceptron solutions:

Theorem 5.20. (Characterization of the Sparsity of Perceptron and SVM solutions using the Excentricity). *For any fraction $0 \leq \mu < 1$, the sparsity of an approximate solution attaining at least a μ-fraction of the optimal margin*[7]*, is bounded from above by $\left\lceil \frac{E}{1-\mu} \right\rceil$ and*

[7] In the single polytope case this means there is a separating hyperplane of distance $\mu\rho$

$\left\lceil \frac{E^*(1+o(1))}{1-\mu} \right\rceil$ *as* $\mu \to 1$ *for the single polytope variants 1a),1b),1c) and by* $\left\lceil \frac{E_{P_1,P_2}}{1-\mu} \right\rceil$ *and* $\left\lceil \frac{E^*_{P_1,P_2}(1+o(1))}{1-\mu} \right\rceil$ *as* $\mu \to 1$ *for the two polytope variants 2a),2b) and 2c),2d)*[8].

The sparsity is bounded from below by $\left\lceil \frac{\frac{1}{2}E}{1-\mu} \right\rceil$ *and* $\left\lceil \frac{E^*}{1-\mu} \right\rceil + 1$ *for SVM variant 1a), and by* $\left\lceil \frac{\frac{1}{4}E_{P_1,P_2}}{1-\mu} \right\rceil$ *and* $\left\lceil \frac{\frac{1}{2}E^*_{P_1,P_2}}{1-\mu} \right\rceil + 2$ *for the standard SVM or Perceptron 2a).*

Proof. Upper bound: This is a direct consequence of Theorem 5.18 in the single polytope case, and Theorem 5.19 in the two polytope case, showing that the away steps algorithm returns an $(1-\mu)$-coreset of the desired size, whose corresponding $(1-\mu)$-approximation proves our upper bound.

Lower bound: Any approximate solution that attains at least a μ-fraction of the optimal margin, is represented by a convex combination $x \in \text{conv}(P)$ (or $x \in \text{MD}(P_1, P_2)$ in the two polytope case) such that $p|_x \geq \mu\rho \ \ \forall p \in P$, or in other words $\frac{p|_x}{\rho} \geq 1-\varepsilon$ if we set $\varepsilon := 1-\mu$. By iii) of Lemma 5.8 (or Lemma 5.11 respectively), we have constructed a point set such that the sparsity of any such x has to be at least the claimed lower bound. □

Corollary 5.21. *The sparsity of any separating solution to a standard hard-margin two-class SVM or Perceptron is at least* $\lceil \frac{1}{2}E^*_{P_1,P_2} \rceil + 2$*, and at least* $\lceil \frac{1}{4}E_{P_1,P_2} \rceil$ *for some training point sets, whereas solutions of sparsity* $\lceil E_{P_1,P_2} \rceil$ *do always exist for all instances.*

Interpretation of the excentricity in the SVM and Perceptron case. For the Perceptron, [GHW00] have proven a similar upper bound on the sparsity of separating solutions, and found it remarkable that it depends on the margin between the two classes. Our lower bound now confirms that this indeed has to be the case. For SVM, already [Bur98, Section 7.5] conjectured, on the basis of empirical results, that it might be good to choose the free kernel parameters so that the quantity E is minimized. By

from the origin, whereas in the two polytope case it refers to a separating hyperplane such that all points have distance at least $\mu\frac{\rho}{2}$ from the plane.

[8] For the ℓ_1-loss SVM variants 2c),2d), our stated upper bound holds for the number of reduced hulls vertices [BB00, CB00, MST06] that are needed to represent a solution, however each vertex of a reduced hull corresponds to a fixed larger subset of non-zero coefficients when expressed in the original points. Thus the sparsity upper bound when expressed in the original points has to be multiplied by this factor, which for the ν-SVM variant 2d) is $\lceil \frac{\nu n}{2} \rceil$ [CB00].

our derived bounds we can now confirm that this choice is indeed good in the sense that it result in the best possible sparsity of the solutions. [Bur98, Theorem 6] also showed that E gives an upper bound for the VC dimension of gap tolerant classifiers, a concept closely related to the complexity of the classification problem.

5.5.2. Linear Time Training of SVMs and Perceptrons

The following is a direct consequence of the analysis of Gilbert's Algorithm and our geometric interpretation of approximation in the one and two polytope setting:

Theorem 5.22. *For all SVM and Perceptron variants 1a) up to 2b), for arbitrary kernels, and for any fixed fraction* $0 \leq \mu < 1$*, we can find a solution attaining at least a* μ*-fraction of the optimal margin in time linear in the number of training points* n*, using Gilbert's Algorithm 10 or Algorithm 11 respectively.*

Proof. Theorem 5.14 and Theorem 5.17 show that the number of Gilbert steps needed is a constant independent of n and the dimension d. By keeping the lengths of all projections onto the previous estimate in memory, one can in each step update all projections by just calculating n scalar products (of the new point p_i to all points in P) [KSBM00], therefore the number of kernel evaluations (scalar product computations) is n in each Gilbert step, and $O(n)$ in total. □

The above theorem also holds the reduced hull SVM variants 2c),2d), but there the number of kernel evaluations has to be multiplied with the previously mentioned factor[8].

Comparison to Existing SVM Training Algorithms. Our above result means that we removed the need for the detour of reducing SVM to a smallest enclosing ball problem, which was a theoretically and experimentally very successful method suggested by Tsang under the name *Core Vector Machine* (CVM) [TKC05, TKK07], for the SVM variants 1b),1c), in the case that all points have the same norm. This is because in that special case the single polytope distance problem is equivalent to a smallest enclosing ball problem. The improved version of the CVM [TKK07] uses Panigrahy's algorithm [Pan04] to obtain a coreset of size $O(1/\varepsilon^2)$ in the same number of steps. In another work [HPRZ07] also proved the

existence of coresets of size $O(1/\varepsilon^2)$ for the problem of separating two polytopes by a hyperplane that goes through the origin (without using kernels), which is a special case of SVM variant 1c) and also equivalent to a single polytope distance problem. Using cutting-plane algorithms, also [Joa06] proved that SVM variant 2d) when no kernel is used can be trained by a method called *SVM-Perf* in linear time, using $O(n/\varepsilon^2)$ scalar product evaluations.

Our contributions can be summarized as follows:

- By generalizing the coreset approach to the two-polytope case, we encompass *all* the currently most used hard- and soft-margin SVM variants with arbitrary kernels, with both ℓ_1 and ℓ_2-loss.
- Our obtained coreset sizes — as well as the algorithm running times — are one order of magnitude smaller in terms of ε, and also have a smaller constant than most existing methods such as [TKC05, TKZ06, TKK07], [HPRZ07] and [Joa06].
- Our method works for arbitrary kernels, and is easier to apply in practice because we do not require the exact solution of small subproblems, overcoming two disadvantages of [TKC05, TKZ06] and also [HPRZ07].
- The obtained coreset sizes are worst-case optimal.

Perceptrons. In the special case $\mu := 0$, our above Theorem gives a bound similar to the well known result that the traditional *Perceptron* algorithm [Ros58] achieves perfect separation after $\frac{M^2}{\rho^2}$ many steps, where $M := \max_{p \in P} \|p\|$ is the largest norm of a sample point [Nov63]. For cases of large margin, our bound of $2\lceil 2E \rceil$ steps is faster than the $\frac{M^2}{\rho^2}$ many steps guaranteed by the currently known bounds for (kernel) Perceptron algorithms [Nov63, SSS05]. Another advantage of our result is that we can not only guarantee separation but simultaneously large margin.

Practical Experiments. [KY10] provide more experimental results, showing that our proposed improved version of Gilbert's algorithm for SVMs in the case of two polytope performs quite well in practice. In the following Chapter 6, we will use the proposed coreset algorithm as the internal optimizer for the solution path tracking approach for SVMs, that is to calculate guaranteed ε-coresets that are valid along the entire SVM regularization path.

6

Solution Paths for Convex Optimization Problems over Vectors

We consider parameterized convex optimization problems over the unit simplex or over the box, that depend on a single additional parameter. We provide a simple and efficient scheme for maintaining an ε-approximate solution (and a corresponding ε-coreset) along the entire parameter path. We prove correctness and optimality of the method. Practically relevant instances of our studied parameterized optimization problems include for example regularization paths of support vector machines (SVMs), multiple kernel learning, ℓ_1-regularized least squares or logistic regression, or geometric problems such as the minimum enclosing ball of moving points.

This is joint work with Joachim Giesen and Soeren Laue [GJL10, GJL12a].

6.1. Introduction

We study convex optimization problems that are parameterized by a single parameter. We are interested in the whole solution path in that parameter,

i.e. the set of optimal solutions for all parameter values. We provide a simple algorithmic framework for tracking guaranteed approximate solutions along the parameter path. We obtain strict guarantees on the approximation quality (continuously along the path) as well as the running time. The main idea of our scheme is to compute at a parameter value an approximate solution that is slightly better than the required quality, and then to keep this solution as the parameter changes, exactly as long as the required approximation quality can still be guaranteed. Only when the approximation quality is no longer sufficient, a new solution is computed.

We prove that, if an approximation guarantee of $\varepsilon > 0$ is required for the entire path, then the number of necessary updates along the path is only $O\left(\frac{1}{\varepsilon}\right)$, independently of the number of variables used. We also show that this obtained path complexity of $O\left(\frac{1}{\varepsilon}\right)$ is indeed best possible.

The practical need to understand the entire path of (near) optimal solutions to parameterized problems comes from many different fields, such as control theory, various multi-objective optimization applications, and most notably from regularization methods in machine learning. The question of parameter selection — e.g. the choice of the best regularization parameter — is often a non-trivial task. Many approaches have been proposed in the last decade for tracking exact solution paths for optimization problems, in the case that the paths are piecewise linear. However there are many applications where this assumption does not hold, and even if it does, the path might have exponential complexity in the worst case [GJM10]. Our new framework strongly contrasts the existing approaches in the literature, and proves a path complexity independent of the input size (only depending on the desired approximation accuracy ε), for a wider class of problems.

Our framework is not tied to a specific algorithm, and very simple to implement: Any existing optimizer or heuristic of choice can be used to compute an approximate solution at fixed parameter values. Warm-starting the optimizer with the previous solution at each update step can be used to further improve speed in practice.

Parameterized Optimization. Our goal is to compute the entire solution path (with a continuously guaranteed approximation quality) for parameterized optimization problems over the $\|.\|_1$-ball, the $\|.\|_\infty$-ball, or the unit simplex., i.e. optimization problems of the form (3.3)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_t(x) \\ s.t. \quad & \|x\|_1 \leq 1 \ , \end{aligned} \tag{6.1}$$

or the analogous box-constrained variant (3.4)

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & f_t(x) \\ s.t. & \|x\|_\infty \leq 1 \ , \end{array} \tag{6.2}$$

or problems over the simplex as in (3.1), i.e.,

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & f_t(x) \\ s.t. & \|x\|_1 = 1 \ , \\ & x \geq 0 \ , \end{array} \tag{6.3}$$

where f_t is a family of convex functions, parameterized by $t \in \mathbb{R}$. Note that a variety of other problems can also be reduced to the above form (6.1), (6.2) or (6.3), by introducing "slack" variables and/or re-scaling. All three formulations have prominent applications in various areas.

Motivation and Applications. Despite our path framework being very simple, we demonstrate that it is well applicable for practical large scale problems. Our experiments demonstrate that the computation of an entire ε-approximate solution path is only marginally more expensive than the computation of a single approximate solution.

Our work is motivated by several important machine learning techniques, some of which we have mentioned in Chapters 2 and 3, which naturally come with a parameter that is non-trivial to choose (e.g. a regularization parameter). We apply the algorithm to compute the entire regularization path for ℓ_1-regularized least squares (also known as basis pursuit de-noising in compressed sensing [CDS98, FNW07]), the *Lasso* [Tib96], ℓ_1-regularized logistic regression [KKB07], support vector machines (SVMs), and to find the best combination of two kernels, which is a special case of multiple kernel learning [BLJ04].

Other applications that directly fit into our framework include ℓ_2-support vector regression (SVR), AdaBoost, mean-variance analysis in portfolio selection [Mar52], the smallest enclosing ball problem [BC07], and also any box-constrained convex optimization problems.

Related Work. Solution path algorithms (sometimes also called homotopy methods) have a long history, in particular in the optimization community [Rit62, Rit84, Osb92] and in control theory (e.g. model predictive control, [GPM89, HHB99]). More recently these methods had an independent revival in machine learning, in particular for computing exact solution paths in the context of support vector machines and related

problems [HRTZ04, WZLS08, GGJW09], and also regression techniques such as ℓ_1-regularized least squares [OPT00, EHJT04, MCW05, Tur05, RZ07, HYZ08]. Similar methods were also applied by [EHJT04, GZ05, LC06, WCYL06, BHH06, WYL06, LGC07, LS07, Wan08] to special cases of quadratic programs, in particular cases where the solution path is piecewise linear. However, three relatively severe problems remained unresolved to large parts:

i) The assumption of piecewise linearity of the path does only hold for a small class of parameterized optimization problems, mostly quadratic programs with linearly parameterized objective or right hand side [GGJW09].

ii) The complexity of exact solution paths can be very large in the worst case, e.g., it can grow exponentially in the input size as it has been shown for support vector machines with ℓ_1-loss [GJM10].

iii) Almost all of the above mentioned existing path algorithms rely on the inversion of principal sub-matrices of large matrices, leading to numerical instability as well as combinatorial problems [HRTZ04, Section 5.2], [GGJW09].

Our proposed framework here overcomes the above issues by moving away from exact solution paths, towards paths of guaranteed ε-approximation quality. Our algorithm works for arbitrary non-linear solution paths, under weak continuity assumptions.

Approximation algorithms and *warm-start* heuristics for parameter selection have gained interest in various areas, such as e.g. in machine learning [FHHT07, KKB07], compressed sensing [HYZ08], or multi-objective optimization [BGP09]. However, to the best of our knowledge, so far no approximation guarantees along the path could be given for these existing algorithms, which sample the path at discrete parameter values.

6.2. Approximation Quality Measures

In this small section we will briefly recall the practical approximation quality measures for convex optimization problems of the form (6.3) and (6.2), which we have proposed in Chapter 2 using the "poor-man's" duality approach. For this it will be crucial to consider the gradient $\nabla f_t(x)$ of the objective function f_t with respect to x. We assume the function $f_t(x)$ to be convex and continuously differentiable in x, for each $t \in \mathbb{R}$.

Lemma 6.1. *The duality gap (2.5) at any candidate $x \in \mathbb{R}^n$ is given by the difference of $f_t(x)$ to its corresponding dual value $\omega_t(x)$, and equals*

$$g_t(x) = \|\nabla f_t(x)\|_\infty + x^T \nabla f_t(x) \text{ for problem (6.1),}$$

$$g_t(x) = \|\nabla f_t(x)\|_1 + x^T \nabla f_t(x) \text{ for problem (6.2),}$$

$$g_t(x) = \max_i (-\nabla f_t(x))_i + x^T \nabla f_t(x) \text{ for problem (6.3),}$$

If x is feasible for the respective optimization problem, $g_t(x) \geq 0$. Furthermore, in each case it holds that $f_t(x) - f_t(x^*) \leq g_t(x)$ for any feasible x, for x^* being an optimal solution.

Proof. Directly from the definition of the duality gap as in (2.5). We already considered these expressions in Sections 3.1, 3.2 and 3.3. □

Notation. Recall that we write $\diamondsuit_n := \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 1\}$ for the ℓ_1-ball in $\mathbb{R}^n$ as being the optimization domain for problem (6.1), $\square_n := \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$ for the unit box of non-negative vectors as in problem (6.2), as well as $\Delta_n := \{x \in \mathbb{R}^n \mid x \geq 0,\ \|x\|_1 = 1\}$ for the unit simplex, see problem (6.3).

The Duality Gap as a Stopping Criterion. For practical optimizers, the last mentioned fact in Lemma 6.1 — which is weak-duality — makes the duality gap an extremely useful measure of approximation and stopping criterion, because the optimum x^* is usually unknown. For all variants, the quantity g_t is easily computable for any candidate solution x even for very large problems.

As explained in Section 3.1, the gap g_t can be interpreted as the difference of the function value to the minimum value of the linear approximation to f at point x, where the minimum is taken over the feasible region. Alternatively, the above Lemma 6.1 also follows from standard Wolfe duality theory [BV04], see the also our appendix Section A.4.

Definition 6.2. *A candidate vector $x \in \mathbb{R}^n$ that is feasible for one of the above optimization problems at some parameter value t is called an* additive ε-approximation *if*

$$g_t(x) \leq \varepsilon \ .$$

It is called a relative ε-approximation *if $g_t(x) \leq \varepsilon f_t(x)$.*

A subset $C \subseteq [n]$ is called an ε-coreset *at "time" t, if there exists an ε-approximation x at parameter value t with $x_i = 0$, $\forall i \in [n] \setminus C$.*

Relative Approximation. When considering relative approximation, we assume that $f_t(x) \geq 0$ for all x in the domain. Sometimes in the literature, a multiplicative ε-approximation is defined more restrictively as $g(x) \leq \varepsilon f(x^*)$, relative to the optimal value $f(x^*)$ of the primal optimization problem. Note that this can directly be obtained from our slightly weaker definition by setting ε in the definition of an ε-approximation to $\varepsilon' := \frac{\varepsilon}{1+\varepsilon}$, because $g(x) \leq \frac{\varepsilon}{1+\varepsilon} f(x) \Leftrightarrow (1+\varepsilon)(f(x)-\omega(x)) \leq \varepsilon f(x) \Leftrightarrow g(x) \leq \varepsilon\omega(x) \leq \varepsilon f(x^*)$, recalling weak duality from Section 2.2. Note that when looking at the relative error for maximization instead of minimization problems, no such distinction is necessary.

6.3. Optimizing Parameterized Functions

We are interested in ε-approximations for problems (6.1), (6.2) and (6.3) for all valid parameter values $t \in \mathbb{R}$, and will study the complexity of such solution paths in the following.

Definition 6.3. *The* ε-approximation path complexity *of a parameterized optimization problem is defined as the minimum number of intervals over all possible partitions of the parameter range* $[t_{\min}, t_{\max}] \subseteq \mathbb{R}$*, such that for each individual interval, there is a single solution that is an* ε*-approximation for that entire interval.*

The following simple lemma is at the core of our discussion and characterizes how far we can change the parameter t such that a given $\frac{\varepsilon}{\gamma}$-approximate solution x (for $\gamma > 1$) at t stays an ε-approximate solution. The idea is a simple continuity argument for the duality gap, as the parameter changes. Lemma 6.4 gives the stability result for the case of bounded ℓ_1-norm (6.1), and Lemma 6.5 considers the ℓ_∞-case (6.2), which is in a sense dual to the ℓ_1-case. The lemmata each give two alternative conditions, which both imply the desired stability of the approximation. Two important consequences will follow from these stability results: First we will obtain that the global ε-approximation path complexity is $O(\frac{1}{\varepsilon})$. Secondly, for most practical problems, we will be able to locally exactly compute the largest possible parameter-interval over which some specific ε-approximate solution stays valid, which in practice will lead to intervals much longer than the worst-case upper bound.

6.3.1. Stability of ε-Approximations

Lemma 6.4 (Stability of an Approximation under Bounded ℓ_1-Norm)**.** *Let $x \in \diamondsuit_n$ be an $\frac{\varepsilon}{\gamma}$-approximation to problem (6.1) for some fixed parameter value t, for some $\gamma > 1$. Then for all $t' \in \mathbb{R}$ that satisfy either*

$$g_{t'}(x) - g_t(x) = \begin{array}{l} \|\nabla f_{t'}(x)\|_\infty - \|\nabla f_t(x)\|_\infty \\ +x^T\big(\nabla f_{t'}(x) - \nabla f_t(x)\big) \end{array} \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.4}$$

or

$$(1 + \|x\|_1)\ \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.5}$$

it holds that the solution x is still an ε-approximation to problem (6.1) at the changed parameter value t'.

Proof. We first prove the statement assuming that (6.4) is satisfied. To do so, we add to (6.4) the inequality stating that x is an $\frac{\varepsilon}{\gamma}$-approximate solution at value t, i.e.

$$\|\nabla f_t(x)\|_\infty + x^T \nabla f_t(x) \leq \frac{\varepsilon}{\gamma} .$$

This directly results in the claimed bound $\|\nabla f_{t'}(x)\|_\infty + x^T \nabla f_{t'}(x) \leq \varepsilon$ on the duality gap at t'.

We will prove the second part of the lemma by showing that the second condition (6.5) in fact implies the condition (6.4). On one hand, by the (reversed) triangle inequality, the first term in (6.4) is upper bounded as follows.

$$\|\nabla f_{t'}(x)\|_\infty - \|\nabla f_t(x)\|_\infty \leq \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty .$$

On the other hand, from the general norm inequality[1] $x^T y \leq \|x\| \cdot \|y\|_*$ together with the fact that the ℓ_∞-norm is dual to the ℓ_1-norm, we have

$$x^T(\nabla f_{t'}(x) - \nabla f_t(x)) \leq \|x\|_1 \cdot \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty .$$

Altogether, this means that (6.5) implies (6.4), or in other words we have reduced the case to the already verified first case of the lemma. □

[1] The inequality $x^T y \leq \|x\| \cdot \|y\|_*$ can be seen as the generalization of the Cauchy-Schwarz inequality to arbitrary norms $\|.\|$ and their corresponding dual norms $\|y\|_* := \sup_{\|x\| \leq 1} x^T y$, and follows directly from the definition of the dual norm.

Lemma 6.5 (Stability of an Approximation under Bounded ℓ_∞-Norm). *Let $x \in \square_n$ be an $\frac{\varepsilon}{\gamma}$-approximation to problem (6.2) for some fixed parameter value t, for some $\gamma > 1$. Then for all $t' \in \mathbb{R}$ that satisfy either*

$$g_{t'}(x) - g_t(x) = \begin{array}{l} \|\nabla f_{t'}(x)\|_1 - \|\nabla f_t(x)\|_1 \\ +x^T\big(\nabla f_{t'}(x) - \nabla f_t(x)\big) \end{array} \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.6}$$

or

$$(1 + \|x\|_\infty)\ \|\nabla f_{t'}(x) - \nabla f_t(x)\|_1 \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.7}$$

it holds that the solution x is still an ε-approximation to problem (6.2) at the changed parameter value t'.

Proof. Completely analogous to the proof of Lemma 6.4, by swapping the roles of $\|.\|_1$ and $\|.\|_\infty$. □

Lemma 6.6 (Stability of an Approximation over the Simplex). *Let $x \in \Delta_n$ be an $\frac{\varepsilon}{\gamma}$-approximation to problem (6.3) for some fixed parameter value t, for some $\gamma > 1$. Then for all $t' \in \mathbb{R}$ that satisfy either*

$$g_{t'}(x) - g_t(x) = \begin{array}{l} \max\limits_i \big(\nabla f_t(x) - \nabla f_{t'}(x)\big)_i \\ +x^T\big(\nabla f_{t'}(x) - \nabla f_t(x)\big) \end{array} \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.8}$$

or

$$(1 + \|x\|_1)\ \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty \leq \varepsilon\left(1 - \frac{1}{\gamma}\right) , \tag{6.9}$$

it holds that the solution x is still an ε-approximation to problem (6.3) at the changed parameter value t'.

Proof. Using the inclusion $\Delta_n \subset \diamondsuit_n$, we can follow the same proof as in Lemma 6.4. Then the first claim will follow from the fact that

$$\max_i \big(\nabla f_t(x) - \nabla f_{t'}(x)\big)_i \leq \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty ,$$

see also the definition of the duality gap given in Lemma 6.1 for the simplex domain. □

6.3.2. Bounding the Path Complexity

The following main theorem on the path complexity bounds the parameter interval lengths in the worst case. Here we assume that the global parameter range of interest, $[t_{\min}, t_{\max}] \subset \mathbb{R}$, is finite.

Theorem 6.7. *Let f_t be convex and continuously differentiable in x, and let $\nabla f_t(x)$ be Lipschitz continuous in t for all feasible x. Then the ε-approximation path complexity of problems (6.1), (6.2) and (6.3) over the parameter range $t \in [t_{\min}, t_{\max}] \subset \mathbb{R}$ is $O\left(\frac{1}{\varepsilon}\right)$.*

Proof. We prove the ℓ_1-case first. In order for the condition (6.5) in Lemma 6.4 to be satisfied, we first use that for any $x \in \diamondsuit_n$,

$$\begin{aligned} & (1 + \|x\|_1) \, \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty \\ \leq \; & 2 \cdot \|\nabla f_{t'}(x) - \nabla f_t(x)\|_\infty \\ \leq \; & 2 \cdot L \cdot |t' - t| \;\; . \end{aligned}$$

Here L is the supremum of the Lipschitz constants with respect to t of the derivatives $\nabla f_t(x)$, taken over the compact feasible domain $\diamondsuit_n$ for x. It follows that condition (6.5) in the Lemma is satisfied for any $x \in \diamondsuit_n$, if we require the parameter intervals to be of length $|t' - t| \leq \frac{\varepsilon}{2L}\left(1 - \frac{1}{\gamma}\right)$.

Dividing the total parameter range $|t_{\max} - t_{\min}|$ by this interval length, the claim follows directly: The path complexity is at most

$$\left\lceil \tfrac{1}{\varepsilon} \cdot |t_{\max} - t_{\min}| \cdot 2L \tfrac{\gamma}{\gamma - 1} \right\rceil \;\; .$$

For the ℓ_∞-case, we argue the same way, applying condition (6.7). The case of optimization over the simplex also follows from the argument for the ℓ_1-case. □

Optimality. One might expect that the above path complexity result of $O\left(\frac{1}{\varepsilon}\right)$ is best possible. As $\nabla f_t(x)$ changes with t, the interval length where $g_t(x) \leq \varepsilon$ holds can not be larger than $\Theta(\varepsilon)$ in the worst case. The next small section will formally prove that the path complexity obtained above is indeed worst-case optimal.

6.3.3. Lower Bound

We provide an explicit lower bound on the path complexity for optimizing over the simplex (being a subset of the ℓ_1-ball $\diamondsuit_n$), showing that the above complexity from Theorem 6.7 is indeed best possible up to a constant factor. We consider the following instance of a parameterized optimization problem (6.3) or (6.1):

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f_t(x) := x^T h(t) \\ \text{s.t.} \quad & x \in \Delta_n \end{aligned} \tag{6.10}$$

where $h(t) = (h_1(t), \ldots, h_n(t))$ is a vector of n parameterized values, defined as

$$h_i(t) := |t - i\varepsilon'| \ ,$$

For some parameter ε'. We assume that $n \geq \frac{1}{\varepsilon'}$. Observe that each of the "coordinate" functions $h_i(t)$ attains its minimum value (zero) at $t = i\varepsilon'$, see also Figure 6.1. As $f_t(x)$ is a linear function of x for each t, f_t is convex in x. Considering the discrete time-points $t_i := i\varepsilon'$, we have that $h_i(t_i) = 0$ and $h_j(t_i) \geq \varepsilon'$ for each $j \neq i$. In other words the optimum at time t_i is attained by x being the i-th standard basis vector $\mathbf{e}_i$.

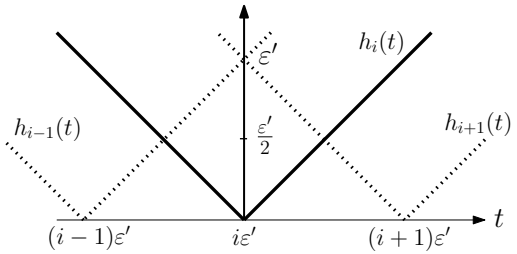


Figure 6.1.: Construction of our parameterized linear function $f_t(x) = x^T(h_1(t), \ldots, h_n(t))$.

Moving our focus towards approximations instead of optimal solutions, we make the following crucial observation: Let $\varepsilon < \frac{\varepsilon'}{2}$, and assume that x is an ε-approximation to (6.10) at time t_i, implying that $f_{t_i}(x) - 0 \leq g_{t_i}(x) < \frac{\varepsilon'}{2}$. Then since $f_{t_i}(x)$ is just the convex combination of the univariate functions h_i, weighted by $x \in \Delta_n$, it must hold that the i-th coordinate of x must be large, or formally that $x_i > \frac{1}{2}$. Therefore, no ε-approximation at t_i can be an ε-approximation at any other time-point t_j for $j \neq i$.

This means the *ε-approximation path complexity* of problem (6.10) over the parameter range $t \in [0, 1]$ is at least $n \geq \left\lfloor \frac{1}{\varepsilon'} \right\rfloor$, which is at least $\left\lfloor \frac{1}{2\varepsilon} \right\rfloor - 1$ by choosing ϵ close to $\varepsilon'/2$.

Optimality of the Path Complexity. Our above example shows that the path complexity is at least $\Omega\left(\frac{1}{\varepsilon}\right)$ in the worst case. Furthermore, we can even show that the constants in our main Theorem 6.7 are in fact best possible up to a constant factor.

To do so, we observe that $\|h(t') - h(t)\|_\infty \leq |t' - t|$. In other words the Lipschitz-constant L_x with respect to t of $\nabla_x f_t(x) = h(t)$ is constantly

equal to 1, independent of x. This means the upper bound from Theorem 6.7 for the path complexity gives precisely $\left\lceil \frac{1}{\varepsilon} \cdot |1-0| \cdot 2 \cdot 1 \frac{\gamma}{\gamma-1} \right\rceil = \left\lceil \frac{|t_{\max}-t_{\min}| \cdot 2 \frac{\gamma}{\gamma-1}}{\varepsilon} \right\rceil$ many intervals. If we choose the improvement parameter γ as being very large (corresponding to large intervals of constant solutions), we therefore have that the path complexity given by Theorem 6.7 is optimal up to a constant multiplicative factor of $4+\delta$, for $\delta > 0$ arbitrary small. For $\gamma := 2$, the corresponding factor becomes $8+\delta$. Indeed, also the dependence on the Lipschitz-constant L w.r.t the parameter t is tight: contracting the functions $h_i(t)$ along the t-direction will linearly increase the Lipschitz constant L of $(\nabla f_t(x))_i$, together with the complexity $\frac{1}{\varepsilon'}$ in our construction.

6.3.4. Relative Approximation

Our framework for the stability of approximations also applies for *relative* instead of *additive* error. The approximation path complexity result also translates analogously. Here we only state the relative approximation condition for optimization over the simplex. A similar result for ℓ_1- and ℓ_∞-problems follows analogously by using Lemma 6.4 and 6.5.

Lemma 6.8. *Let $x \in \Delta_n$ be a relative $\frac{\varepsilon}{\gamma}$-approximation to problem (6.3) for some fixed parameter value t, and for some $\gamma > 1$. Then for all $t' \in \mathbb{R}$ that satisfy*

$$g_{t'}(x) - g_t(x) = \begin{array}{c} \max\limits_i \left(\nabla f_t(x) - \nabla f_{t'}(x)\right)_i \\ + x^T \left(\nabla f_{t'}(x) - \nabla f_t(x)\right) \end{array} \leq \varepsilon \left(f_{t'}(x) - \frac{1}{\gamma} f_t(x) \right),$$

it holds that the solution x is still a relative ε-approximation to problem (6.3) at the changed parameter value t'.

Proof. Analogous to the additive approximation as in Lemma 6.6. □

6.3.5. The Weighted Sum of Two Convex Functions

We are particularly interested in a special case of problem (6.3): For any two convex, continuously differentiable functions $f^{(1)}, f^{(2)} : \mathbb{R}^n \to \mathbb{R}$ that are non-negative on the simplex Δ_n, we consider the weighted sum $f_t(x) := f^{(1)}(x) + t f^{(2)}(x)$ for a real parameter $t \geq 0$. The parameterized optimization problem (6.3) in this case becomes

$$\begin{array}{ll} \min_x & f^{(1)}(x) + t f^{(2)}(x) \\ \text{s.t.} & x \in \Delta_n \ . \end{array} \tag{6.11}$$

This class of optimization problems has many relevant applications, in particular for regularization methods, as we have mentioned in Section 1.3.2. For this formulation, the change of the gradients $\nabla f_t(x)$ with the parameter t will be very easy to control. We have the following corollary of the "approximation stability" Lemma 6.8:

Corollary 6.9. *Let $x \in \Delta_n$ be a relative $\frac{\varepsilon}{\gamma}$-approximation to problem (6.11) for some fixed parameter value $t \geq 0$, and for some $\gamma > 1$. Then for all $t' \geq 0$ that satisfy*

$$\begin{aligned}(t'-t)\left(x^T\nabla f^{(2)}(x) - \left(\nabla f^{(2)}(x)\right)_i - \varepsilon f^{(2)}(x)\right) \\ \leq \varepsilon\left(1-\tfrac{1}{\gamma}\right) f_t(x), \qquad \forall i \in [n]\end{aligned} \tag{6.12}$$

it holds that x is still a relative ε-approximation to problem (6.11) at the changed parameter value t'.

Proof. Follows directly from Lemma 6.8, and observing that $f_{t'}(x) - f_t(x) = (t'-t)f^{(2)}(x)$. □

This allows us to exactly calculate the entire interval of admissible parameter values t' such that an $\frac{\varepsilon}{\gamma}$-approximation at t is still an ε-approximation at t':

Corollary 6.10. *Let x be a relative $\frac{\varepsilon}{\gamma}$-approximation to the problem (6.11) for some fixed parameter value $t \geq 0$, for $\gamma > 1$, and let*

$$\begin{aligned} u &:= x^T\nabla f^{(2)}(x) - \min_i \left(\nabla f^{(2)}(x)\right)_i - \varepsilon f^{(2)}(x) \\ l &:= x^T\nabla f^{(2)}(x) - \max_i \left(\nabla f^{(2)}(x)\right)_i - \varepsilon f^{(2)}(x), \end{aligned}$$

then $l \leq u$ and x remains a relative ε-approximation for all $0 \leq t' = t + \delta$ for the following values of δ:

(i) If $l < 0$ and $0 < u$, then the respective admissible values for δ are

$$\varepsilon\left(1-\frac{1}{\gamma}\right)\frac{f_t(x)}{l} \leq \delta \leq \varepsilon\left(1-\frac{1}{\gamma}\right)\frac{f_t(x)}{u}$$

(ii) If $u \leq 0$, then δ (and thus t') can become arbitrarily large.

(iii) If $l \geq 0$, then δ can become as small as $-t$, and so t' can become zero.

Proof. Directly from solving condition (6.12) for $\delta := t' - t$. □

Note that the ε-approximation path complexity for problem (6.11) for a given value of $\gamma > 1$ can be upper bounded by the minimum number of points $t_j \geq 0$ such that the admissible intervals of $\frac{\varepsilon}{\gamma}$-approximate solutions x_j at t_j cover the whole parameter range $[0, t_{\max}]$.

Corollary 6.10 immediately suggests two variants of an algorithmic framework (forward- and backward version) maintaining ε-approximate solutions over the entire parameter interval, or in other words, tracking a guaranteed ε-approximate solution path. Note that as the internal optimizer, any arbitrary approximation algorithm can be used here, as long as it provides an approximation guarantee on the relative primal-dual gap. For example the standard Frank-Wolfe sparse greedy Algorithm 3 that we described in Chapter 3 is particularly suitable here, as its resulting coreset solutions are also sparse, see also [Cla10, Algorithm 1.1]. The forward variant is depicted in Algorithm 12 and the backward variant in Algorithm 13.

Algorithm 12 ApproximationPath—ForwardVersion

Input: convex function $f_t = f^{(1)}(x) + tf^{(2)}(x), t_{\min}, t_{\max}, \varepsilon, \gamma$
Output: ε-approximate solution path for problem (6.11)
Set $t := t_{\min}$.
Compute an ε-approximation x to (6.11) at parameter $t_{\min}$.
repeat
  Improve x *(which is now still ε-approximate)* to an at least $\frac{\varepsilon}{\gamma}$-approximate solution at t, by applying steps of any optimizer.
  $u := x^T \nabla f^{(2)}(x) - \min_i \left(\nabla f^{(2)}(x)\right)_i - \varepsilon f^{(2)}(x)$
  if $u > 0$ **then**
    $\delta := \varepsilon \left(1 - \frac{1}{\gamma}\right) \frac{f_t(x)}{u} > 0$
    $t := t + \delta$
  else
    $t := t_{\max}$
until $t \geq t_{\max}$

Algorithm 13 ApproximationPath—BackwardVersion

Input: convex function $f_t = f^{(1)}(x) + tf^{(2)}(x), t_{\max}, t_{\min}, \varepsilon, \gamma$
Output: ε-approximate solution path for problem (6.11)
Set $t := t_{\max}$.
Compute an ε-approximation x to (6.11) at parameter $t_{\max}$.
repeat
  Improve x *(which is now still ε-approximate)* to an at least $\frac{\varepsilon}{\gamma}$-approximate solution at t, by applying steps of any optimizer.
  $l := x^T \nabla f^{(2)}(x) - \max_i \left(\nabla f^{(2)}(x)\right)_i - \varepsilon f^{(2)}(x)$
  if $l < 0$ **then**
    $\delta := \varepsilon \left(1 - \frac{1}{\gamma}\right) \frac{f_t(x)}{l} < 0$
    $t := t + \delta$
  else
    $t := t_{\min}$
until $t \leq t_{\min}$

6.4. Applications

Special cases of the parameterized problem (6.11), and of course also the more general problems (6.1), (6.2) and (6.3), have applications in many areas, in particular computational geometry, machine learning, compressed sensing and finance. In the following we discuss three of these applications in more detail, namely, regularization paths of support vector machines (SVMs), multiple kernel learning, and smallest enclosing balls of linearly moving points. The first two applications for SVMs are special instances of the weighted sum problem (6.11) and more specifically of a parameterized polytope distance problem that we discuss at first.

6.4.1. A Parameterized Polytope Distance Problem

In the setting of Section 6.3.5, we consider the case $f^{(1)}(x) := x^T K^{(1)} x$ and $f^{(2)}(x) := x^T K^{(2)} x$, for two positive semi-definite matrices $K^{(1)}, K^{(2)} \in \mathbb{R}^{n \times n}$, or formally

$$\begin{aligned} \min_x \quad & f^{(1)}(x) + tf^{(2)}(x) = x^T \left(K^{(1)} + tK^{(2)}\right) x \\ \text{s.t.} \quad & x \in \Delta_n \ . \end{aligned} \tag{6.13}$$

The geometric interpretation of this problem is as follows, compare also to our Chapter 5: let $A(t) \in \mathbb{R}^{n \times r}$, $r \leq n$, be a matrix such $A(t)^T A(t) =$

$K^{(1)}+tK^{(2)}$ (Cholesky decomposition), for some large enough r. The solution x^* to problem (6.13) is the point in the convex hull of the column vectors of the matrix $A(t)$ that is closest to the origin. Hence, problem (6.13) is a parameterized polytope distance problem. For the geometric interpretation of an ε-approximation in this context we refer to [GJ09]. In the following we will consider two geometric parameters for any fixed polytope distance problem:

Definition 6.11. *For a positive semi-definite matrix $K \in \mathbb{R}^{n\times n}$, we define*

$$\rho_{(K)} := \min_{x\in\Delta_n} x^T K x \qquad \text{and} \qquad R_{(K)} := \max_i K_{ii}$$

or in other words when considering the polytope associated with K, $\rho_{(K)}$ is the minimum (squared) distance to the origin, and $R_{(K)}$ is the largest squared norm of a point in the polytope. We say that the polytope distance problem $\min_{x\in\Delta_n} x^T K x$ is separable *if $\rho_{(K)} > 0$.*

For the parameterized problem (6.13), the two quantities u and l that determine the admissible parameter intervals in Corollary 6.10 and the step size in both approximate path algorithms take the simpler form

$$\begin{aligned} u &= (2-\varepsilon)x^T K^{(2)} x - 2\min_i (K^{(2)}x)_i \\ \text{and} \quad l &= (2-\varepsilon)x^T K^{(2)} x - 2\max_i (K^{(2)}x)_i, \end{aligned}$$

since $\nabla f^{(2)}(x) = 2K^{(2)}x$. We can now use the following lemma to bound the path complexity for instances of problem (6.13).

Lemma 6.12. *Let $0 < \varepsilon \leq 1$ and $\gamma > 1$. Then for any parameter $t \geq \varepsilon$, the length of the interval $[t-\delta, t]$ with $\delta > 0$, on which a relative $\frac{\varepsilon}{\gamma}$-approximation x to problem (6.13) at parameter value t remains a relative ε-approximation, is at least*

$$l_f(\varepsilon,\gamma) := \frac{\varepsilon}{2}\left(1-\frac{1}{\gamma}\right)\frac{\rho_{(K^{(1)})}}{R_{(K^{(2)})}} = \Omega(\varepsilon)\ . \tag{6.14}$$

Proof. In the special case $l \geq 0$, the interval length given by Corollary 6.10 is as long as $t \geq \varepsilon$. Otherwise, for $l = (2-\varepsilon)x^T K^{(2)} x - 2\max_i (K^{(2)}x)_i < 0$, we get from Corollary 6.10 that the length of the "left" interval $[t-\delta, t]$ at point x is of length at least

$$\varepsilon\left(1-\frac{1}{\gamma}\right)\frac{f_t(x)}{-l}.$$

For any $t \geq 0$, we have the lower bound

$$f_t(x) \geq f^{(1)}(x) = x^T K^{(1)} x \geq \min_{x \in \Delta_n} x^T K^{(1)} x = \rho_{(K^{(1)})},$$

and for $\varepsilon \leq 1$ we have the upper bound

$$-l = 2 \max_i (K^{(2)} x)_i - (2 - \varepsilon) x^T K^{(2)} x \leq 2 \max_i (K^{(2)} x)_i,$$

because $f^{(2)}(x) \geq 0$. The value $\max_i (K^{(2)} x)_i = \max_i \mathbf{e}_i^T K^{(2)} x$ is the inner product between two points in the convex hull of the columns of any factorization of the positive semi-definite matrix $K^{(2)}$ (see the discussion at the beginning of this section, or the geometric Chapter 5). Let these two points be $u, v \in \mathbb{R}^n$. Using the Cauchy-Schwarz inequality we get

$$\begin{aligned} \max_i (K^{(2)} x)_i &= u^T v \leq \sqrt{\|u\|^2 \|v\|^2} \leq \frac{1}{2}(\|u\|^2 + \|v\|^2) \\ &\leq \max\{\|u\|^2, \|v\|^2\} \leq \max_{x \in \Delta_n} x^T K^{(2)} x, \end{aligned}$$

where the last expression gives the norm of the longest vector with endpoint in the convex hull of the columns of the square root of $K^{(2)}$. However, the largest such norm (in contrast to the smallest norm) is always attained at a vertex of the polytope (which can be seen by writing any such optimal point as a convex combination of some vertices). Formally, this means that $\max_{x \in \Delta_n} x^T K^{(2)} x = \max_i \mathbf{e}_i^T K^{(2)} \mathbf{e}_i = \max_i K^{(2)}_{ii} = R_{(K^{(2)})}$. Hence, $-l \leq 2R_{(K^{(2)})}$. Combining the lower bound for $f_t(x)$ and the upper bound for $-l$ gives the stated bound on the interval length. □

Now, to upper bound the approximation path complexity, we split the domain $[0, \infty]$ into two parts: the range $[0, 1]$ can be covered by at most $1/l_f(\varepsilon, \gamma)$ admissible "left" intervals $[t - \delta, t]$, i.e., by at most $1/l_f(\varepsilon, \gamma)$ many admissible intervals.

We reduce the analysis for the range $t \in [1, \infty]$ to the analysis for $[0, 1]$ by interchanging the roles of $f^{(1)}$ and $f^{(2)}$. For any $t \geq 1$, x is an ε-approximate solution to $\min_{x \in \Delta_n} f_t(x) := f^{(1)}(x) + t f^{(2)}(x)$ if and only if x is an ε-approximate solution to $\min_{x \in \Delta_n} \tilde{f}_{t'}(x) := t' f^{(1)}(x) + f^{(2)}(x)$ for $t' = \frac{1}{t} \leq 1$, because the definition of an ε-approximation is invariant under scaling the objective function. Note that by allowing $t = \infty$ we just refer to the case $t' = 0$ in the equivalent problem for $\tilde{f}_{t'}(x)$ with $t' = \frac{1}{t} \in [0, 1]$. Using the lower bounds on the interval lengths $l_f(\varepsilon, \gamma)$ and $l_{\tilde{f}}(\varepsilon, \gamma)$ (for the problem for $\tilde{f}_{t'}(x)$ with $t' \in [0, 1]$) on both intervals we get an upper

bound of $\left\lceil \frac{1}{l_f(\varepsilon,\gamma)} \right\rceil + \left\lceil \frac{1}{l_{\tilde{f}}(\varepsilon,\gamma)} \right\rceil$ on the path complexity as is detailed in the following theorem:

Theorem 6.13. *Given any $0 < \varepsilon \leq 1$ and $\gamma > 1$, and assuming that the distance problems associated to $K^{(1)}$ and $K^{(2)}$ are both separable, we have that the ε-approximation path complexity of problem (6.13) is at most*

$$\frac{\gamma}{\gamma - 1} \left(\frac{R_{(K^{(2)})}}{\rho_{(K^{(1)})}} + \frac{R_{(K^{(1)})}}{\rho_{(K^{(2)})}} \right) \frac{2}{\varepsilon} + 2 = O\left(\frac{1}{\varepsilon}\right) .$$

This result on the path complexity immediately implies a bound on the time complexity of our approximation path Algorithm 12. In particular we have obtained a linear running time of $O\left(\frac{n}{\varepsilon^2}\right)$ many arithmetic operations for computing the global solution path, when using Algorithm 3 as the internal optimizer. This follows because the number of path intervals is $O\left(\frac{1}{\varepsilon}\right)$, and computing a single $\frac{\varepsilon}{\gamma}$-approximate solution for each interval takes $O\left(\frac{\gamma}{\varepsilon}\right)$ many iterations of Algorithm 3, each iteration having cost n to compute (update to) the new gradient.

There are interesting applications of this result, because it is known that instances of problem (6.13) include for example computing the solution path of a support vector machine – as the regularization parameter changes – and also finding the optimal combination of two kernel matrices in the setting of kernel learning. We will discuss these applications in the following sections.

6.4.2. The Regularization Path of Support Vector Machines

Support Vector Machines (SVMs) are well established machine learning techniques for classification and related problems. In Chapter 5 (see also [GJ09]) we have seen that most of the practically used SVM variants are equivalent to a polytope distance problem, i.e., finding the point in the convex hull of a set of data points that is closest to the origin. In particular the so called one class SVM with ℓ_2-loss term [TKC05, Equation (8)], and the two class ℓ_2-SVM without offset as well as with penalized offset, see [TKC05, Equation (13)] for details, are instances of the following polytope distance problem

$$\begin{array}{ll} \min_x & x^T \left(K + \frac{1}{c}\mathbf{I}\right) x \\ \text{s.t.} & x \in \Delta_n \end{array} \tag{6.15}$$

where the so called *kernel matrix* K is an arbitrary positive semi-definite matrix consisting of the inner products $K_{ij} = \langle \Psi(p_i), \Psi(p_j) \rangle$ of the data points $p_1, \ldots, p_n \in \mathbb{R}^d$ mapped into a kernel feature space $\mathcal{H}$ by $\Psi : \mathbb{R}^d \to \mathcal{H}$. The parameter c $(= \frac{1}{t})$ is called the *regularization parameter*, and controls the trade-off between the regularization and the loss term in the objective function. Selecting the right regularization parameter value and by that balancing between low model complexity and overfitting is a very important problem for SVMs and machine learning methods in general and highly influences the prediction accuracy of the method.

Problem (6.15) is a special case of (6.13) with $K^{(2)} = \mathbf{I}$, and in this case the quantities u and l (used in Corollary 6.10 and the approximate path Algorithm 12 and Algorithm 13 now have the even simpler form

$$u = (2-\varepsilon)x^T x - 2\min_i x_i \qquad \text{and} \qquad l = (2-\varepsilon)x^T x - 2\max_i x_i,$$

and from Lemma 6.12 we get the following corollary for the complexity of an approximate regularization path, i.e., the approximation path complexity for problem (6.15).

Corollary 6.14. *Let $0 < \varepsilon \leq 1$, $\gamma > 1$ and $c_{\min} \leq 1$. We assume that the distance problem associated to K is separable.*

Then the ε-approximation path complexity of the regularization parameter path for problem (6.15) for $c \in [c_{\min}, \infty)$ is at most

$$\frac{\gamma}{\gamma - 1} \frac{R_{(K)} + c_{\min}}{\rho_{(K)} \cdot c_{\min}} \cdot \frac{2}{\varepsilon} + 2 = O\left(\frac{R_{(K)}}{\rho_{(K)} c_{\min} \cdot \varepsilon}\right) = O\left(\frac{1}{\varepsilon \cdot c_{\min}}\right) .$$

Proof. As in the proof of Theorem 6.13, the number of admissible intervals that are necessary to cover the parameter range $[0, 1] \ni \frac{1}{c} = t$ can be bounded by

$$\frac{\gamma}{\gamma - 1} \frac{1}{\rho_{(K)}} \frac{2}{\varepsilon} = O\left(\frac{1}{\varepsilon}\right) ,$$

because $R_{(\mathbf{I})} = \max_i \mathbf{I}_{ii} = 1$.

The interval $t \in [1, 1/c_{\min}]$ or equivalently $c \in [c_{\min}, 1]$ (and $\tilde{f}_c(x) = x^T \mathbf{I} x + c \cdot x^T K x$) can also be analyzed following the proof of Lemma 6.12. Only, now we bound the function value as follows

$$\tilde{f}_c(x) = x^T \mathbf{I} x + c x^T K x \geq c x^T K x \geq c_{\min} \min_{x \in \Delta_n} x^T K x = c_{\min} \rho_{(K)}$$

to lower bound the length of an admissible interval. Hence, the number of admissible intervals needed to cover $[c_{\min}, 1]$ is at most

$$\frac{\gamma}{\gamma - 1} \frac{1}{c_{\min}} \frac{R_{(K)}}{\rho_{(K)}} \frac{2}{\varepsilon}.$$

Adding the complexities of both intervals gives the claimed complexity for the regularization path. □

Of course we could also have used our "path complexity" Theorem 6.13 directly, but using $\rho_{(\mathbf{I})} = \frac{1}{n}$ would only give a complexity bound that is proportional to n. However, if we choose to stay above $c_{\min}$, then we can obtain the better bound as described in the above theorem.

Globally Valid Coresets. Using the above Theorem 6.13 for the number $O\left(\frac{1}{\varepsilon \cdot c_{\min}}\right)$ of intervals of constant solutions, and combining this with the size $O\left(\frac{1}{\varepsilon}\right)$ of a coreset at a fixed parameter value, as e.g. provided by the sparse greedy algorithms from Chapter 3, we can just build the union of those individual coresets to get an ε-coreset of size $O\left(\frac{1}{\varepsilon^2 \cdot c_{\min}}\right)$ that is valid over the entire solution path. This means we have upper bounded the overall number of support vectors used in a solution valid over the entire parameter range $c \in [c_{\min}, \infty)$. This is particularly nice as this number is independent of both the number of data points and the dimension of the feature space, and can easily be constructed by our Algorithms 12 and 13.

In Section 6.5.1 we report experimental results using this algorithmic framework for choosing the best regularization parameter.

6.4.3. Multiple Kernel Learning

Another immediate application of the parameterized framework in the context of SVMs is "learning" the best combination of two kernels. This is a special case of the multiple kernel learning problem, where the optimal kernel to be used in a SVM is not known a priori, but needs to be selected out of a set of candidates. This set of candidates is often chosen to be the convex hull of a few given "base" kernels, see for example [BLJ04]. In our setting with two given kernel matrices $K^{(1)}$,$K^{(2)}$, the kernel learning problem can be written as follows:

$$\begin{array}{ll} \min_x & x^T \left(\lambda K^{(1)} + (1-\lambda) K^{(2)} + \frac{1}{c}\mathbf{I}\right) x \\ \text{s.t.} & x \in \Delta_n \end{array} \tag{6.16}$$

where $0 \leq \lambda \leq 1$, is the parameter that we want to learn. To simplify the notation, let us define the matrices $K_c^{(1)} := K^{(1)} + \frac{1}{c}$ and $K_c^{(2)} := K^{(2)} + \frac{1}{c}$. By scaling the objective function by $1/\lambda$ (where λ is assumed to be non-zero), problem (6.16) can be transformed to a special case of problem (6.13), where $t = \frac{1-\lambda}{\lambda}$ (note again that the scaling does not affect our measure of primal-dual approximation error, because the error measure is relative):

$$\begin{array}{rl} \min_x & x^T K_c^{(1)} x + t \cdot x^T K_c^{(2)} x \\ \text{s.t.} & x \in \Delta_n \end{array} \tag{6.17}$$

This again allows us to apply both approximation path Algorithms 12 and 13, and to conclude from Theorem 6.13 that the complexity of an ε-approximate path for problem (6.17) for $t \in [0, \infty]$ is in $O\left(\frac{1}{\varepsilon}\right)$. Here the assumption that the distance problems associated to $K_c^{(1)}$ and $K_c^{(2)}$ are both separable holds trivially because $1/c > 0$.

In the case that we have more than two base kernels we can still apply the above approach if we fix the weights of all kernels except one. We can then navigate along the solution paths optimizing each kernel weight separately, and therefore try to find total weights with a hopefully best possible *cross-validation* accuracy. Cross-validation refers to using splitting the available data into two disjoint sets, the training and the test set. The training set is used to obtain the model x (by solving the optimization problem), and later the separate test set is used to validate the quality of the model x.

In Section 6.5.2 we report experimental results to determine the best combination of two kernels to achieve the highest prediction accuracy.

6.4.4. Minimum Enclosing Ball of Points under Linear Motion

The framework described here can also be used to solve the smallest enclosing ball problem for points in $\mathbb{R}^d$ that exhibit a linear motion in time, see [GJL10, GJL12a]. Here our algorithm improves the size of a coreset that is valid at all time-points to $O(\frac{1}{\varepsilon^2})$, where the previous upper bound was $2^{O(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})}$ due to [AHPY07], or $O(1/\varepsilon^{2d})$ for maintaining the more informative extent measures of the moving points [AHPV04].

6.5. Experimental Results

The parameterized framework from Section 6.3 is also useful in practice. For support vector machines and multiple kernel learning, we have implemented the approximation path Algorithms 12 and 13 in Java. As the internal optimizer, we used the sparse greedy Algorithm 3 as described in Chapters 2 and 3. This algorithm can be implemented in a straightforward way to work directly with the kernel matrix, without requiring explicit representations of the datapoints in the kernel-space. We have seen in Chapter 5 that the method coincides with Gilbert's algorithm [Gil66] in the geometric setting. In our implementations, we have also used the alternative MDM [MDM74] variant, also directly applied to the kernel matrix.

We have tested our implementations on the following standard binary classification datasets (n points originally living in $\mathbb{R}^d$ each) from the UCI repository[2]: ionosphere ($n = 280, d = 34$), breast-cancer ($n = 546, d = 10$), and MNIST 4k ($n = 4000, d = 780$). The timings were obtained by our single-threaded Java 6 implementation of MDM, using kernels (and no caching of kernel evaluations), on an Intel C2D 2.4 GHz processor.

6.5.1. The Regularization Path of Support Vector Machines

Using the SVM formulation of problem (6.15) (for the ℓ_2-SVM without offset), we compute approximate regularization paths for $c \in [c_{\min} = \frac{1}{100000}, c_{\max} = 100000]$ using the polynomial kernel $(\langle p_i, p_j \rangle + 1)^2$. As experimental results we report in Table 6.1 the following quantities: (a) the time T_{init} (in seconds) needed to compute an initial $\frac{\varepsilon}{\gamma}$-approximate solution as the starting point, (b) the time T_{path} (in seconds) needed to follow the entire ε-approximate regularization path, when starting from the initial solution, (c) for comparison the time T_3 (in seconds) needed to compute a static ε-approximate solution at the three fixed parameter values $c = c_{\min}, 1$ and $c_{\max}$, and (d) the path complexity, i.e. the number $\#int$ of obtained admissible parameter intervals of constant ε-approximate solutions along the path. The lower part of the table demonstrates the dependency of the path complexity on the choice of the parameter γ.

These experimental results show that the path complexity is indeed

[2] All datasets are available from www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets. In our experiments, all features were scaled to [-1,1]. For MNIST, the first 5000 'one' or 'seven' images of the original dataset were used.

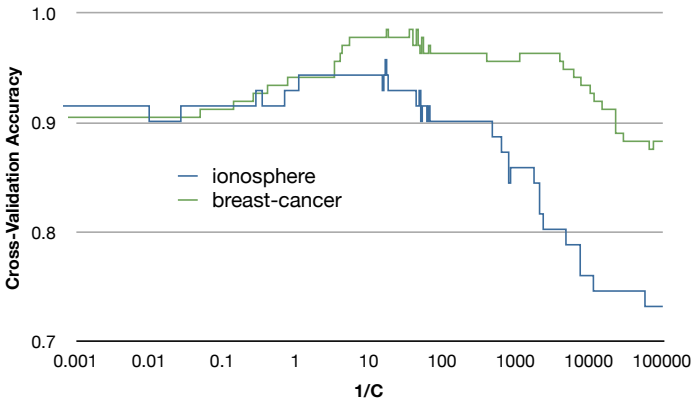


Figure 6.2.: Continuous cross-validation along the ($\varepsilon = 0.2$)-approximate regularization path.

	dataset	'Forward' Algorithm 12			'Backward' Algorithm 13			
$\gamma = 2$		$\#int$	T_{path}	T_{init}	$\#int$	T_{path}	T_{init}	T_3
$\varepsilon = 0.5$	ionosphere	53	5.2	2.8	98	4.3	0.3	3.3
	breast-cancer	65	4.7	3.3	102	5.2	0.4	3.3
	MNIST 4k	20	170.9	32.4	58	148.1	129.7	174.3
$\varepsilon = 0.1$	ionosphere	294	32.8	4.7	438	24.2	0.4	5.8
	breast-cancer	365	35.9	6.8	445	27.6	0.5	6.3
	MNIST 4k	103	837.1	52.3	274	722.7	169.7	264.0
$\varepsilon = 0.01$	ionosphere	3012	361.8	7.9	4251	251.7	0.6	9.9
	breast-cancer	3730	443.9	16.8	4307	305.9	0.7	16.5
	MNIST 4k	1030	8885.7	91.4	2692	7245.5	246.6	396.7

$\varepsilon = 0.1$	'Forward' Algorithm 12									
	$\gamma = 5$			$\gamma = 2$			$\gamma = 1.2$			
dataset	$\#int$	T_{path}	T_{init}	$\#int$	T_{path}	T_{init}	$\#int$	T_{path}	T_{init}	T_3
ionosphere	188	53.6	5.9	294	32.8	4.7	808	26.0	4.2	5.8
breast-cancer	235	67.3	12.2	365	35.9	6.8	983	26.5	5.5	6.3
MNIST 4k	66	1487.9	70.9	103	837.1	52.3	288	656.9	47.5	264.0

Table 6.1.: Path complexity and running times, depending on ε and γ.

small if ε is not too small. We note that the method can be sped up further by using a more sophisticated internal optimization procedure. In practice, already relatively large values as for example $\varepsilon = 1$ are sufficient for good generalization performance, as a primal-dual gap of ε implies that more than a $(1 - \frac{\varepsilon}{2})$-fraction of the best possible classification margin is already obtained [GJ09].

Warm Start and Running Time. The "improvement" parameter γ, as shown in the lower part of Table 6.1, is in an interesting trade-off with the computational complexity: As the required quality improvement goes down ($\gamma \to 1$), the total running time of the algorithm in our experiment decreases, despite the fact that more path intervals need to be computed. This is due to the *warm start* of the internal optimizer at the previous solution x_t being more efficient when t' is closer to t.

Warm-starting SVM related problems at slightly varied parameter values with a previous solution has already been used for a long time [DW00], but our framework is the first to provide any guarantees for a continuous solution between the fixed parameter values. It remains to investigate if the coreset framework can also be applied in the case of varying non-linear kernel hyper-parameters [WYL07].

Cross-Validation. From every dataset, a separate set of 1/5 of the original data points was kept for cross-validation. This means the computed classifier is evaluated on a small set of n_{cv} test points that have *not* been used to solve the SVM optimization problem. Since our approximate regularization path has complexity at most $O\left(\frac{1}{\varepsilon}\right)$ (and we have a constant, ε-approximate solution on each admissible interval along the path), the cost of calculating all continuous cross-validation values, i.e., the percentages of correctly classified data points among the test points, along the entire regularization path is just $O\left(\frac{n_{cv}}{\varepsilon}\right)$ kernel evaluations. Cross-validation values along the path are shown in Figure 6.2.

6.5.2. Multiple Kernel Learning

In the multiple kernel learning setting of problem (6.16), we used our implementation to compute approximate solution paths for $t \in [t_{\min} = \frac{1}{100000}, t_{\max} = 100000]$, for the problem to learn the best convex combination of the Gaussian kernel $K^{(1)}$ of width $\sigma = 8.5$, and the polynomial kernel $K^{(2)} = (\langle p_i, p_j \rangle + 1)^2$ on the same data sets as before. We chose a fixed regularization parameter value of $c = 1.5$. In Table 6.2 we report for

Forward-Algorithm 12 (a) the time T_{init} (in seconds) needed compute an initial $\frac{\varepsilon}{\gamma}$-approximate solution as the starting point $t_{\min}$, (b) the time T_{path} (in seconds) needed to follow the entire ε-approximate regularization path, when starting from the initial solution, (c) for comparison the time T_3 (in seconds) needed to compute a static ε-approximate solution at the three fixed parameter values $t = t_{\min}, 1$ and $t_{\max}$, and (d) the path complexity, i.e. the number $\#int$ of admissible parameter intervals with constant ε-approximate solutions along the path. Again a separate set of 1/5 of the original points was used to compute the resulting cross-validation values for an ε-approximate solution along the entire solution path, as shown in Figure 6.3.

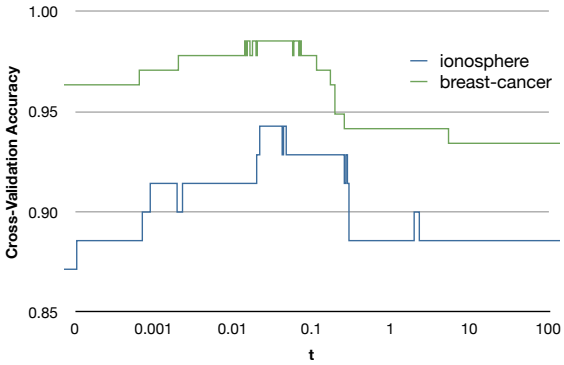


Figure 6.3.: Continuous cross-validation along the ($\varepsilon = 0.2$)-approximate solution path.

$\gamma = 2$	*dataset*	$\#int$	T_{path}	T_{init}	T_3
$\varepsilon = 0.5$	ionosphere	53	14.1	4.4	6.8
	breast-cancer	71	4.0	2.4	3.7
	MNIST 4k	30	355.5	139.1	312.5
$\varepsilon = 0.1$	ionosphere	281	87.0	8.2	12.9
	breast-cancer	382	23.5	4.6	6.8
	MNIST 4k	150	2155.5	249.4	573.5

Table 6.2.: Path complexity and running times

In practice, there are related methods that optimize a joint objective function over both the classifier weights and the combination of multiple kernels [BLJ04, RBCG08]. These methods are experimentally fast, but are not directly comparable to ours as they do not obtain a solution path and are therefore unable to provide guarantees such as an optimal cross-validation value along a parameter path.

6.6. Conclusion

We have presented a framework to optimize convex functions over the unit simplex that are parameterized by an additional parameter. This allows us to maintain a guaranteed approximation quality along the entire continuous solution path, by piecewise constant solutions. Also, for many practical cases, we can efficiently compute the exact interval length over which a solution stays valid, allowing to adapt to the local shape of the optimization function in the parameter. The framework is general, simple and has been proven to be practical on a number of machine learning problems.

7

Solution Paths for Semidefinite Optimization

We devise a framework for computing an approximate solution path for an important class of parameterized semidefinite problems that is guaranteed to be ε-close to the exact solution path. As a result, we can compute the entire regularization path for most matrix completion and factorization approaches, as well as nuclear norm or weighted nuclear norm regularized convex optimization problems. This also includes robust PCA and variants of sparse PCA and maximum variance unfolding. On the theoretical side, we show that the complexity of the approximate path is independent of the size of the input matrix. More precisely, the path complexity only grows linearly with the inverse of the desired approximation quality ε, and with the model complexity (regularization). This implies that the whole solution path can be computed in *near linear* time in the size of the input. Our experiments demonstrate the practical efficiency of the approach for large matrix completion problems.

This chapter is joint work with Joachim Giesen and Soeren Laue [GJL12b], and generalizes the path method from the previous Chapter 6 to the case of semidefinite optimization over bounded trace.

7.1. Introduction

We provide an algorithmic framework for tracking approximate solutions of parameterized semidefinite optimization problems along the parameter path. The algorithm is very simple and comes with strong guarantees on the approximation quality (continuously along the path) as well as the running time. The idea of our scheme is the same as in the previous Chapter 6: We compute at a parameter value an approximate solution that is slightly better than the required quality, and then keep this solution as the parameter changes, exactly as long as the required approximation quality can still be guaranteed. Only when the approximation quality is no longer sufficient, a new solution needs to be computed. As already in the vector case in Chapter 6, we prove that the number of necessary updates along the entire path is only $O\left(\frac{1}{\varepsilon}\right)$, if an approximation guarantee of $\varepsilon > 0$ is required along the path. The path complexity therefore again is independent of the size of the problem (number of variables).

The task of considering and computing the entire path of (near) optimal solutions to parameterized problems appears naturally in various different fields, such as control theory, various multi-objective optimization applications, and most notably from regularization methods in machine learning. The question of parameter selection — e.g. the choice of the best regularization parameter — is often a non-trivial task. Many approaches have been proposed in the last decade for tracking exact solution paths for vector optimization problems, in the case that the paths are piecewise linear. However there are many applications where this assumption does not hold, and even if it does, the path might have exponential complexity in the worst case [GJM10]. Our new framework strongly contrasts the existing approaches in the literature, and proves a path complexity of $O\left(\frac{1}{\varepsilon}\right)$ for a wider class of problems.

Again as in Chapter 6, our framework here is not tied to a specific algorithm, and very simple to implement: Any existing semidefinite optimizer or heuristic of choice can be used to compute an approximate solution at fixed parameter values. We only need to compute a bound on the *duality gap* for a given candidate at a fixed parameter value, in order to be able to apply our path framework. We show that the duality gap can efficiently be computed by a single eigenvalue computation. Despite the framework being very simple, we demonstrate that it is well applicable for practical large scale problems. Furthermore, as we have already argued for vector optimization problems, the path complexity of $O\left(\frac{1}{\varepsilon}\right)$ is indeed best possible in the worst case.

Our experiments demonstrate that often, the computation of an entire ε-approximate solution path is only marginally more expensive than the computation of a single approximate solution.

Parameterized Semidefinite Optimization. Our goal is to compute the entire solution path (with a continuously guaranteed approximation quality) for parameterized semidefinite problems over bounded trace, i.e. optimization problems of the form (3.5)

$$\begin{array}{ll} \min\limits_{X} & f_t(X) \\ s.t. & \operatorname{Tr}(X) = 1 \ , \\ & X \succeq 0 \ , \end{array} \tag{7.1}$$

or the analogous inequality-constrained variant

$$\begin{array}{ll} \min\limits_{X} & f_t(X) \\ s.t. & \operatorname{Tr}(X) \leq 1 \ , \\ & X \succeq 0 \ , \end{array} \tag{7.2}$$

where f_t is a family of convex functions, parameterized by $t \in \mathbb{R}$, defined on symmetric matrices $X \in \mathbb{S}^{n \times n}$. We are in particular interested in the special case where the parameterization is given by the trace itself, i.e.

$$\begin{array}{ll} \min\limits_{X} & f(X) \\ s.t. & \operatorname{Tr}(X) \leq t \ , \\ & X \succeq 0 \ . \end{array} \tag{7.3}$$

All three formulations have prominent applications in various areas. Clearly the last formulation can be seen as a special case of (7.2), for the function $f_t(X) := f(tX)$ that re-scales its argument. Furthermore (7.2) is a special case of (7.1), as we can always add an additional slack row/column (of which the function does not depend) to the matrix.

Motivation and Applications. Our work is motivated by nuclear norm regularized optimization problems, see also Chapter 4, which have become central to many applications in machine learning and compressed sensing, as for example low-rank recovery [FHB01, CR09, CT10], robust PCA [CLMW11], and matrix completion [SRJ04, RS05, Web06, Lin07, KBV09, TPNT09, SS10]. Formally we study problems of the form

$$\min_{Z \in \mathbb{R}^{m \times n}} f(Z) + \lambda \left\| Z \right\|_* \tag{7.4}$$

for a convex function f (the loss function), where $\|.\|_*$ is the nuclear norm. We recall that the equivalent constrained formulation (4.2) is

$$\min_{Z \in \mathbb{R}^{m \times n},\, \|Z\|_* \leq t/2} f(Z) \ . \tag{7.5}$$

Both problems are parameterized by a real regularization parameter (λ and t respectively). Note that the former is the Lagrangian formulation of the latter.

To relate the above nuclear norm regularized problems (7.4) and (7.5) to semidefinite optimization, the straightforward transformation that we explained in Section 4.4 (or see e.g. [FHB01, SRJ04, JS10]) comes to help. This along the way also explains why the nuclear norm is widely called the *trace* norm: Any problem of the form (7.5) is equivalent to optimizing the semidefinite version (7.3) for a function $\hat{f} : \mathbb{S}^{(m+n)\times(m+n)} \to \mathbb{R}$, where $\hat{f}$ is defined as

$$\hat{f}(X) = \hat{f}\left(\begin{pmatrix} V & Z \\ Z^T & W \end{pmatrix}\right) := f(Z) \tag{7.6}$$

for $Z \in \mathbb{R}^{m\times n}$ being the upper right part of X. Formally we again think of the variable $X \in \mathbb{S}^{(m+n)\times(m+n)}$ as consisting of the four parts $X =: \left(\begin{smallmatrix} V & Z \\ Z^T & W \end{smallmatrix}\right)$ with $V \in \mathbb{S}^{m\times m}, W \in \mathbb{S}^{n\times n}$ and $Z \in \mathbb{R}^{m\times n}$. Observe that $\hat{f}$ is convex whenever f is convex.

Related Work. For kernel methods and many other machine learning techniques, the resulting optimization problems often turn out to be parameterized convex quadratic programs, and in recent years a plenitude of algorithms and heuristics have been developed to "track" these solution paths, see e.g. [HRTZ04, LGC07, RZ07, LS07, PH07]. However, the exact piecewise linear solution path of parameterized quadratic programs (in particular for the SVM) is known to be of exponential complexity in the worst case [GJM10]. In the work here by contrast we show that just constantly many intervals of the parameter are sufficient for any fixed desired continuous approximation quality $\varepsilon > 0$. For vector optimization problems such as SVMs, we have described in the previous Chapter 6 on how guarantees along the entire path can be obtained, see also [GJL10, GJL12a].

To our best knowledge, no path algorithms were known so far for the more general case of semidefinite optimization. The solution path for sparse principal component analysis (PCA) was investigated by [dBG07], which however is parameterized over *discrete* integral values from 1 to n, where n is the number of variables. For a variant of low-rank matrix completion, [MHT10] suggest to perform a grid search on the regularization

parameter interval, computing a single approximate solution at each parameter t. However, they provide no approximation guarantee between the chosen grid points.

7.2. The Duality Gap

In this small section we will briefly recall the practical approximation quality measures for convex optimization problems of the form (7.1), (7.2) or (7.3), which we have discussed in more detail in Section 3.4. For this we will assume that our objective function $f_t(X)$ is continuously differentiable[1]. We consider the gradient $\nabla f_t(X)$ with respect to X, which is a symmetric matrix in $\mathbb{S}^{n\times n}$.

We write $X \bullet Y$ for the entry-wise inner product of two matrices.

Lemma 7.1. *The duality gap at any matrix $X \in \mathbb{S}^{n\times n}$ is given by the difference of $f_t(X)$ to its corresponding dual value $\omega_t(X)$, and equals*

$$g_t(X) = \lambda_{\max}\left(-\nabla f_t(X)\right) + X \bullet \nabla f_t(X) \text{ for problem (7.1),}$$

$$g_t(X) = \max\left\{0, \lambda_{\max}\left(-\nabla f_t(X)\right)\right\} + X \bullet \nabla f_t(X) \text{ for problem (7.2),}$$

$$g_t(X) = t \cdot \max\left\{0, \lambda_{\max}\left(-\nabla f(X)\right)\right\} + X \bullet \nabla f(X) \text{ for problem (7.3).}$$

Furthermore, it holds that $f_t(X) - f_t(X^*) \leq g_t(X)$ for any feasible X, for X^* being an optimal solution to the respective optimization problem.

Proof. We have calculated the duality gap for problem (7.1) in equation (3.6) in Section 3.4, using the fact that the spectahedron domain $\mathcal{S} := \{X \in \mathbb{S}^{n\times n} \mid X \succeq 0,\ \operatorname{Tr}(X) = 1\}$ is the convex hull of the rank-1 matrices of unit trace. The expressions for the later two problem variants then follow from the same result, because both problems can be reduced to the first formulation (7.1) as described above. □

The last mentioned fact makes the duality gap an extremely useful measure of approximation and stopping criterion for practical optimizers: While the optimum value $f_t(X^*)$ is usually unknown, the gap $g_t(X)$ readily guarantees a simple upper bound on the current difference to this optimum value. For all variants, the quantity g_t is easily computable for any candidate solution X even for very large problems, as we only need

[1] If $f_t(X)$ is convex but not differentiable, the concepts of standard Lagrange duality can be still generalized for subgradients analogously, so that any element of the subgradient will give an upper bound on the approximation error, see Chapter 2.

to perform a single eigenvalue computation, assumed that the gradient is available.

Having these nice mentioned properties, one would think that the achieved duality gap is the standard or routine measure when comparing different heuristics for the same optimization problems. However, in the current machine learning literature, this is not yet done as often as one would wish.

As explained in Chapter 2, the duality gap g_t can also be interpreted as the difference of the function value to the minimum value of the linear approximation to f at point X, where the minimum is taken over the feasible region. Alternatively, the above Lemma can also be obtained from standard duality theory [BV04, Section 5.9] or [Haz08].

Definition 7.2. *Let $\varepsilon > 0$. A matrix $X \in \mathbb{S}^{n\times n}$ that is feasible for one of the above optimization problems at some parameter value t is called an* ε-approximation *if the duality gap satisfies*

$$g_t(X) \leq \varepsilon \ .$$

7.3. Optimizing Parameterized Semidefinite Problems

We are interested in ε-approximations for problems (7.1), (7.2) and (7.3) for all valid parameter values $t \in \mathbb{R}$, and will study the complexity of such solution paths in the following. This definition coincides with the measure we used for vector problems in Chapter 6.

Definition 7.3. *The* ε-approximation path complexity *of a parameterized optimization problem is defined as the minimum number of intervals over all possible partitions of the parameter range $[t_{\min}, t_{\max}] \subset \mathbb{R}$, such that for each individual interval, there is a single solution that is an ε-approximation for that entire interval.*

General Parameterized Functions. The following simple lemma is at the core of our discussion of approximation paths, and characterizes how far we can change the parameter t such that a given $\frac{\varepsilon}{\gamma}$-approximate solution X (for $\gamma > 1$) at t stays an ε-approximate solution.

Lemma 7.4 (Stability of an Approximation). *Let X be an $\frac{\varepsilon}{\gamma}$-approximation to problem (7.1) for some fixed parameter value t, and for some $\gamma > 1$.*

Then for all parameters $t' \in \mathbb{R}$ and feasible X' that satisfy

$$\begin{aligned} &\lambda_{\max}\left(-\nabla f_{t'}(X')\right) - \lambda_{\max}\left(-\nabla f_t(X)\right) \\ &\quad + X' \bullet \nabla f_{t'}(X') \quad - X \bullet \nabla f_t(X) \;\leq\; \varepsilon\left(1 - \tfrac{1}{\gamma}\right), \end{aligned} \tag{7.7}$$

it holds that X' is an ε-approximation to problem (7.1) at the changed parameter value t'.

Proof. In the equality constrained case, we have to show that $g_{t'}(X') = \lambda_{\max}\left(-\nabla f_{t'}(X')\right) + X' \bullet \nabla f_{t'}(X') \leq \varepsilon$. To do so, we add to the inequality (7.7) the inequality stating that X is an $\frac{\varepsilon}{\gamma}$-approximate solution at value t, i.e.

$$\lambda_{\max}\left(-\nabla f_t(X)\right) + X \bullet \nabla f_t(X) \leq \frac{\varepsilon}{\gamma}\,,$$

to obtain the claimed bound on the gap at the new parameter value t'. □

The above Lemma 7.4 does also apply for the inequality-constrained case (7.2), if we just replace the two $\lambda_{\max}(.)$-terms in condition (7.7) by $\max\{0, \lambda_{\max}(.)\}$ instead. The proof follows analogously. Alternatively, this change can also be seen by adding a slack row/column to X, and therefore reducing to the equality constrained case (7.1).

The following alternative (more restrictive) criterion is simpler to evaluate in a concrete implementation of our path algorithms, as for example for more complicated parameterizations of (7.1) or (7.2).

Lemma 7.5. *Let X be an $\frac{\varepsilon}{\gamma}$-approximation for problem (7.1) or (7.2) for some fixed parameter value t, and for some $\gamma > 1$. Then for all $t' \in \mathbb{R}$ that satisfy*

$$\left(1 + \|X\|_{Fro}\right) \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \leq \varepsilon\left(1 - \frac{1}{\gamma}\right),$$

it holds that X is still an ε-approximation at parameter value t'.

Proof. We aim to apply Lemma 7.4, and therefore try to upper bound the terms on the left hand side of inequality (7.7) for $X' = X$. We start by upper bounding the difference in the $\lambda_{\max}$-values: Weyl's perturbation theorem on the eigenvalues of a matrix $A' = A + E$ states that

$$|\lambda_{\max}(A') - \lambda_{\max}(A)| \leq \|E\|_{spec}\,.$$

See e.g. [Nak10]. The matrix spectral norm always satisfies $\|E\|_{spec} \leq \|E\|_{Fro}$. Applying Weyl's theorem to $A' = -\nabla f_{t'}(X)$ and $A = -\nabla f_t(X)$

gives

$$\begin{aligned} &|\lambda_{\max}\left(-\nabla f_{t'}(X)\right) - \lambda_{\max}\left(-\nabla f_t(X)\right)| \\ &\leq \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \ . \end{aligned} \tag{7.8}$$

It remains to upper bound the term $X \bullet (\nabla f_{t'}(X) - \nabla f_t(X))$, which can be done by using the Cauchy-Schwarz inequality

$$|X \bullet (\nabla f_{t'}(X) - \nabla f_t(X))| \leq \|X\|_{Fro} \cdot \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \ .$$

Hence, the inequality in the assumption of this lemma implies that inequality (7.7) in Lemma 7.4 holds, with $X' = X$, from which we obtain our claimed approximation quality $g_{t'}(X) \leq \varepsilon$ for the equality constrained Problem (7.1).

For the inequality constrained case of Problem (7.2), the same result follows since the bound (7.8) that we obtained from Weyl's theorem in particular also implies that

$$\begin{aligned} &|\max\left\{0, \lambda_{\max}\left(-\nabla f_{t'}(X)\right)\right\} - \max\left\{0, \lambda_{\max}\left(-\nabla f_t(X)\right)\right\}| \\ &\leq \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \ . \end{aligned}$$

This follows by observing that $|\max\{0, a\} - \max\{0, b\}| \leq |a - b|$. This means we have the desired bound on the approximation quality at t' also for the duality gap for problem version (7.2). □

For the following main theorem on the path complexity, we again assume that the global parameter range of interest, $[t_{\min}, t_{\max}] \subset \mathbb{R}$, is finite.

Theorem 7.6. *Let f_t be convex and continuously differentiable in X, and let $\nabla f_t(X)$ be Lipschitz continuous in t, for all feasible X. Then the ε-approximation path complexity of problems (7.1) and (7.2) over the parameter range $t \in [t_{\min}, t_{\max}] \subset \mathbb{R}$ is in $O\left(\frac{1}{\varepsilon}\right)$.*

Proof. In order for the condition of Lemma 7.5 to be satisfied, we first use that for any $X \succeq 0, \mathrm{Tr}(X) \leq 1$,

$$\begin{aligned} & \left(1 + \|X\|_{Fro}\right) \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \\ \leq \ & \left(1 + \|X\|_{Fro}\right) \cdot L \cdot |t' - t| \\ \leq \ & 2 \cdot L \cdot |t' - t| \ . \end{aligned}$$

Here L is the supremum of the Lipschitz constants w.r.t. t of the derivatives $\nabla f_t(X)$, taken over the compact feasible domain for X. So if we require the intervals to be of length $|t' - t| \leq \frac{\varepsilon}{2L}\left(1 - \frac{1}{\gamma}\right)$, we have that the condition in Lemma 7.5 is satisfied for any $X \succeq 0, \mathrm{Tr}(X) \leq 1$.

Dividing the total parameter range $|t_{\max} - t_{\min}|$ by this interval length of $\frac{\varepsilon}{2L}\left(1 - \frac{1}{\gamma}\right)$, the bound on the path complexity follows directly. □

Optimality. This path complexity result is in fact best possible. When $\nabla f_t(X)$ does effectively change with t with rate L_X (here L_X is the Lipschitz constant w.r.t. t of $\nabla f_t(X)$), then the interval length where $g_t(X) \leq \varepsilon$ holds can not be longer than $\Theta(\varepsilon)$. For an explicit example of a function resulting in this worst-case complexity, we refer to our discussion for the vector case in Section 6.3.3. To adapt this to the case of trace one matrices, one can consider the analogous linear function $f_t(X) := X \bullet H(t)$, for $H(t) := \operatorname{diag}(h(t))$ being diagonal.

Optimizing with Growing Trace. In the special case of problem (7.3), the calculation of the lengths of intervals of guaranteed ε-approximation quality becomes considerably simpler:

Lemma 7.7. *Let X be an $\frac{\varepsilon}{\gamma}$-approximate solution of problem (7.3) for some fixed parameter value t, and for some $\gamma > 1$. Then for all $t' \geq t \in \mathbb{R}$ that satisfy*

$$(t' - t) \cdot \lambda_{\max}\left(-\nabla f(X)\right) \leq \varepsilon \left(1 - \tfrac{1}{\gamma}\right), \tag{7.9}$$

the solution X is still an ε-approximation to problem (7.3) at the parameter value t'.

Proof. Follows from Lemma 7.4 applied to the function $f_t(X') := f(tX')$, if we set the new approximation to $X' := \frac{t}{t'} X$. An alternative direct proof also goes along the same lines as in Lemma 7.4. □

7.3.1. Computing Approximate Solution Paths

The above Lemmata 7.4 and 7.7 on "preserving the approximation quality" do immediately suggest two simple algorithms to compute ε-approximate solution paths, which are depicted in Algorithms 14 and 15. Furthermore, they imply that we can efficiently and locally compute the exact largest possible interval length for each given pair (X, t) in practice. In those regions where f changes only slowly in t, this makes the algorithms much more efficient than if we would just work with the guaranteed $O(\varepsilon)$ worst-case upper bound on the interval lengths.

By Theorem 7.6 (or Lemma 7.7 respectively), the running time of this method is $O(T(\frac{\varepsilon}{\gamma})/\varepsilon)$, where $T(\varepsilon')$ is the time to compute a single ε'-approximate solution for problem (7.1), (7.2) or (7.3) at a fixed parameter value.

Algorithm 14 General SDP-Path

Input: convex function $f_t, t_{\min}, t_{\max}, \varepsilon, \gamma$
Output: ε-approximate solution path for problem (7.1) or (7.2)
Set $t := t_{\min}$.
repeat
  Compute an $\frac{\varepsilon}{\gamma}$-approximation X at parameter value t.
  Compute $t' > t$ such that

$$(1 + \|X\|_{Fro}) \|\nabla f_{t'}(X) - \nabla f_t(X)\|_{Fro} \leq \varepsilon \left(1 - \tfrac{1}{\gamma}\right).$$

  Update $t := t'$.
until $t \geq t_{\max}$

Algorithm 15 Growing Trace SDP-Path

Input: convex function $f, t_{\min}, t_{\max}, \varepsilon, \gamma$
Output: ε-approximate solution path for problem (7.3)
Set $t := t_{\min}$.
repeat
  Compute an $\frac{\varepsilon}{\gamma}$-approximation X
    to problem (7.3) at parameter value t.
  Update $t := t + \frac{\varepsilon(1-\frac{1}{\gamma})}{\lambda_{\max}(-\nabla f(X))}$.
until $t \geq t_{\max}$

7.3.2. Plugging-in Existing Methods for Semidefinite Optimization

We briefly review some of the existing solvers that can be used internally in our described path optimization framework. In our experiments we used Algorithm 6, originally by [Haz08], see also Section 3.4.1, because it scales well to large inputs, provides approximate solutions with guarantees, and only requires a single approximate eigenvector computation in each of its iterations. Furthermore, it returns a matrix factorization of the resulting estimates X for free, see also the discussion in Section 4.4.

There are many other popular methods to solve nuclear norm regularized problems. Alternating gradient descent or stochastic gradient descent (SGD) methods were used extensively in particular for matrix completion problems, see e.g. [RS05, Web06, Lin07, KBV09, TPNT09, RR11]. However, these methods optimize a non-convex formulation of (4.1) and can get stuck in local minima, and therefore — in contrast to Hazan's method

with our convex transformation (7.6) — come with no convergence guarantee. On the other hand, there are also several known convex optimization methods of "proximal gradient" and "singular value thresholding"-type from the optimization community, see e.g. [TY10], which however experimentally perform slower than Hazan's method [JS10].

Nevertheless, any of these other methods and heuristics can still be used as the internal optimizer in our path-following framework, as we can always compute the duality gap as a certificate for the quality of the found approximate solution.

7.4. Applications

Using our above path approximation framework, we directly obtain piecewise constant solution paths of guaranteed approximation quality for any problem of the form (7.1), (7.2) or (7.3), including all nuclear norm regularized problems (7.4) and (7.5), such as standard matrix completion problems, which we introduce next.

7.4.1. Matrix Completion

Our path framework applies to matrix completion problems with any convex differentiable loss function, such as the smoothed hinge loss or the standard squared loss, and includes the classical maximum-margin matrix factorization variants [SRJ04].

The regularized matrix completion task is exactly problem (7.5) where the function f is given by the loss over the observed entries of the matrix, $\Omega \subseteq [n] \times [m]$, i.e. $f(Z) = \sum_{(i,j)\in\Omega} L(Z_{ij}, Y_{ij})$. Here $L(.,.)$ is an arbitrary loss-function that is convex in its Z-argument. By far the most widely used variant employs the squared loss, given by

$$f(Z) = \frac{1}{2} \sum_{(i,j)\in\Omega} (Z_{ij} - Y_{ij})^2. \tag{7.10}$$

Using the notation $(A)_\Omega$ for the matrix that coincides with A on the indices Ω and is zero otherwise, $\nabla f(Z)$ can be written as

$$\nabla f(Z) = (Z - Y)_\Omega \ .$$

This implies that the symmetric gradient matrix $\nabla \hat{f}(X) \in \mathbb{S}^{(m+n)\times(m+n)}$ for our transformed problem (7.6) that we use in our algorithm is also

of this simple form (recall the notation $X = \left(\begin{smallmatrix} V & Z \\ Z^T & W \end{smallmatrix}\right)$). As this matrix is sparse — it has only $|\Omega|$ non-zero entries — storage and approximate eigenvector computations can be performed much more efficiently than for dense problems. An equivalent *matrix factorization* of any approximation X for problem (7.3) can always be obtained directly from the Cholesky decomposition of X, because $X \succeq 0$.

7.4.2. Solution Paths for the Weighted Nuclear Norm

We recall from Section 4.2.1 that any convex problem with a constrained weighted nuclear norm

$$\min_{Z \in \mathbb{R}^{m \times n},\, \|Z\|_{nuc(p,q)} \leq t/2} f(Z) \tag{7.11}$$

is equivalent to a classical nuclear norm regularized problem

$$\min_{\bar{Z} \in \mathbb{R}^{m \times n},\, \left\|\bar{Z}\right\|_* \leq t/2} f(P^{-1}\bar{Z}Q^{-1}).$$

(this corresponds to substituting $\bar{Z} := PZQ$ for some fixed diagonal matrices P, Q). Therefore the problem can be directly formulated in the form of (7.3), and our path tracking Algorithm 15 applies without modifications.

7.4.3. Solution Paths for Robust PCA

With principal component analysis (PCA) being today's most widely used tool for the analysis of high-dimensional data and dimensionality reduction, but being very sensitive to errors and noise in just a single datapoint, [CLMW11] have proposed the following robust version of PCA, also called principal component pursuit

$$\min_{Z \in \mathbb{R}^{m \times n}} \|Z\|_* + \lambda' \|M - Z\|_1 \ .$$

Here $\|.\|_1$ is the entry-wise ℓ_1-norm, and $M \in \mathbb{R}^{m \times n}$ is the given data matrix. This problem is already of the form (7.4), for $\lambda = \frac{1}{\lambda'}$. Therefore we can obtain the entire path in the nuclear norm regularization parameter λ, and the corresponding constrained variant (7.5), by using our simple Algorithm 15 for growing trace.

As the internal optimizer, any existing method for robust PCA can be used. If the $\|.\|_1$-norm is smoothened (see also Section 4.5.1), then the same stopping criterion determined by the duality gap from Lemma 7.1

applies. Alternatively, if the *original* non-smooth formulation (7.5) over the domain $\left\{ Z \in \mathbb{R}^{m \times n} \;\middle|\; \|Z\|_* \leq \frac{t}{2} \right\}$ is solved approximately by some arbitrary optimizer, then we can use our generalization of the duality gap to the case of non-smooth functions, see Section 2.2. In this case, any subgradient D_Z of our objective function $\|M - Z\|_1$ gives a certificate for some value of a duality gap $g(Z, D_Z)$ as defined in equation (2.5).

We therefore obtain piecewise constant solutions together with a continuous ε-approximation guarantee along the entire regularization path.

7.4.4. Solution Paths for Sparse PCA and Maximum Variance Unfolding

Sparse PCA. The concept of *sparse PCA* is to approximate a given data matrix $A \in \mathbb{S}^{n \times n}$ by approximate eigenvectors that are additionally sparse, see [ZdG10] for an overview. Many algorithms have been proposed for sparse PCA, see e.g. [SB08] and [dBG07], the latter also considering the discrete solution path, as the sparsity changes.

The SDP-relaxation of [dGJL07, Equation 3.2] for sparse PCA of a matrix A is given by

$$\begin{array}{ll} \min\limits_{X \succeq 0} & \rho \cdot \mathbf{1}^T |X| \, \mathbf{1} - \operatorname{Tr}(AX) \\ s.t. & \operatorname{Tr}(X) = 1 \;, \\ & X \succeq 0 \;. \end{array} \tag{7.12}$$

Here $|X|$ is element-wise for the matrix X, and $\mathbf{1} \in \mathbb{R}^n$ is the all-one vector. Using our path Algorithm 14, we can compute the approximate solution path in the penalty parameter ρ of this relaxation, which is of the form (7.1).

Maximum Variance Unfolding. Another interesting technique for non-linear dimensionality reduction is named *maximum variance unfolding* (MVU). Here we consider the Laplacian regularized MVU formulation

$$\min_{X \succeq 0} \sum_{i \sim j} \left((QXQ^T)_{ii} + (QWQ^T)_{jj} - 2(QXQ^T)_{ij} - d_{ij}^2 \right)^2 - \frac{1}{\nu} \operatorname{Tr}(X) \;, \tag{7.13}$$

see [WSZS07, Formulation (7)]. Here the notation $i \sim j$ means that the vertices i and j form an edge in the underlying graph. In this formulation, a previously fixed low-rank matrix Q is used to represent the local neighborhood structure of the graph, given by the bottom-most eigenvectors of the graph Laplacian. This directly fits into problem (7.3), therefore our Algorithm 15 applies for parameter $\frac{1}{\nu}$.

7.5. Experimental Results

Set-up. We demonstrate the practicability of our framework by applying it to nuclear norm regularized matrix completion tasks on the standard MovieLens data sets[2].

	#ratings	$m = \#users$	$n = \#movies$
MovieLens 100k	10^5	943	1682
MovieLens 1M	10^6	6040	3706
MovieLens 10M	10^7	69878	10677

The goal of these experiments is to demonstrate that the full regularization path for nuclear norm regularized problems is indeed efficiently computable for large datasets, and that our described approximation guarantees are practical. Many existing papers additionally employ low-rank heuristics for practical reasons. Since low-rank constraints do form a non-convex domain, these methods loose the merits and possible guarantees for convex optimization methods. Here we can approximate the original convex problems (7.5) and (7.11) with guaranteed approximation quality, without any restrictions on the rank, for both nuclear norm as well as weighted nuclear norm regularized problems.

In all our experiments we have used the "growing trace" Algorithm 15, for optimizing the squared loss (7.10). We calculated the full regularization paths for the nuclear norm $\|.\|_*$ regularized problem (7.5), as well as for regularization by the weighted nuclear norm $\|.\|_{nuc(p,q)}$ as in formulation (4.7).

To compute the primal-dual gap $g_t(X)$ for each candidate X, we computed 300 iterations of the power method to get an accurate bound on $\lambda_{\max}$. As the internal optimizer within Algorithm 15 we have chosen Hazan's algorithm, and used the power method as the eigenvector subroutine. We employed both slight improvements suggested in Section 3.4.1, see also [JS10]. That is, we used averaging between the new and old gradient in each power iteration, and performing a line search at each rank-1-update. All experiments were performed on a standard laptop computer.

All the provided ratings were used "as-is", without normalization to any kind of prior[3]. Each dataset was uniformly split into 50% test ratings, and

[2] See www.grouplens.org.

[3] Users or movies which do appear in the test set but *not* in the training set were kept as is, as our model is robust under this case. These ratings are accounted in the test RMSE, which is slightly worse therefore (our prediction X will always remain at the worst value, zero, at any such rating).

50% training ratings. The accuracy ε was chosen as the *relative* error, with respect to the initial function value $f_t(X = 0)$ at $t = 0$, i.e. $\varepsilon = \varepsilon' f_0(0)$. As f is the squared loss, $f_0(0)$ equals the Frobenius norm of the observed training ratings, see (7.10).

Results. Figures 7.1 and 7.2 show the rooted mean squared error (RMSE) values along a guaranteed ($\varepsilon' = 0.05$)-approximate piecewise constant solution path for the three MovieLens datasets. We used a quality improvement factor of $\gamma = 2$ in all experiments.

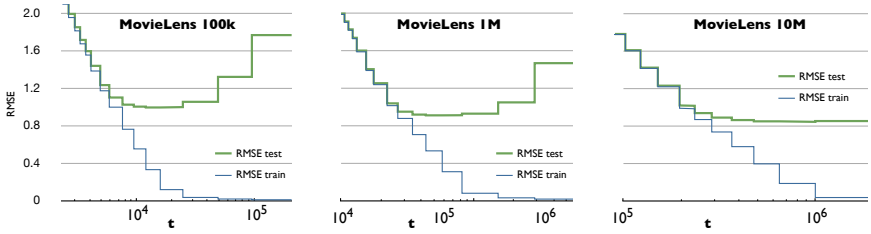


Figure 7.1.: The nuclear norm regularization path for the three MovieLens datasets.

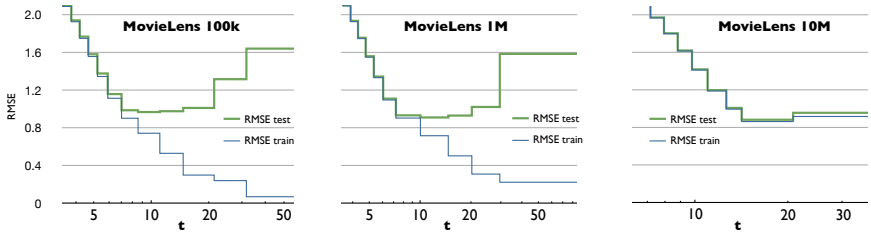


Figure 7.2.: The regularization path for the weighted nuclear norm $\|.\|_{nuc(p,q)}$.

Table 7.1.: Dependency of the path complexity ($\#int$) on the accuracy ε.

Regularization	Accuracy	MovieLens 100k, $\gamma = 2$					
	$\varepsilon/f_0(0)$	$t_{\min}$	$t_{\max}$	$\#int$	Δ_t^{avg}	$f_{t_{\max}}^{\text{train}}$	$f_{\text{opt}}^{\text{test}}$
Nuclear norm	0.05	1000	60000	21	4515	0.0150	0.9912
$\|.\|_*$	0.01	1000	60000	97	582	0.0054	0.9905
	0.002	1000	60000	386	175	0.0009	0.9981
Weighted	0.05	2	50	17	3.21	0.0619	0.9607
nuclear norm	0.01	2	50	72	1.18	0.0147	0.9559
$\|.\|_{nuc(p,q)}$	0.002	2	50	325	0.140	0.0098	0.9581

Table 7.1 shows that the dependency of the path complexity on the approximation quality is indeed favorably weak. Here $\#int$ denotes the number of intervals with constant solution of guaranteed ε-small duality gap; Δ_t^{avg} is the average length of an interval of constant solution; $f_{t_{\max}}^{\text{train}}$ is the RMSE on the training data at the largest parameter value $t_{\max}$; and finally $f_{\text{opt}}^{\text{test}}$ is the best $RMSE_{\text{test}}$ value obtained over the whole computed regularization path.

7.6. Conclusion

We have presented a simple but efficient framework that allows to track approximate solutions for parameterized semidefinite programs with guarantees. Many well known semidefinite optimization problems such as regularized matrix factorization/completion as well as maximum variance unfolding do fit into this framework. Our experiments show a surprisingly small path complexity when measured in the number of intervals of guaranteed ε-accurate constant solutions for the considered problems, even for large matrices. The experiments do confirm our theoretical result that the complexity is essentially independent of the input size, i.e. it only scales with $O\left(\frac{1}{\varepsilon}\right)$ in the desired accuracy, and with the largest trace parameter, which is the regularization bound imposed.

In the future we plan to explore more applications of semidefinite optimization, where our path framework could potentially also deepen the insight into these parameterized optimization problems. In particular, it will be interesting to investigate kernel learning, metric learning and other relaxations of sparse PCA in more detail.

A

Optimization Basics

In the following, we will give a brief (yet mostly self-contained) introduction to classical convex optimization and Lagrange duality.

This will allow us to show that for most classes of optimization problems that are of interest in this thesis, the concept of Wolfe duality will in fact coincide with our "poor-man's" duality as defined in Section 2.2.

A.1. Constrained Optimization Problems over Vectors

Let $f, g_1, \ldots, g_m$ be arbitrary functions from $\mathbb{R}^n \to \mathbb{R}$, and let the domain $D \subseteq \mathbb{R}^n$ be an arbitrary set.

We consider optimization problems under inequality constraints

$$\begin{aligned} \underset{x \in D}{\text{minimize}} \quad & f(x) \\ & g_i(x) \leq 0, \qquad i = 1 \ldots m \end{aligned} \tag{A.1}$$

and we assume that the functions as well as the domain D are chosen such that the minimum exists. A vector $x \in \mathbb{R}^n$ is called a *feasible point* if $x \in D$, and $g_i(x) \leq 0$ for $i = 1 \ldots m$. The *feasible region* is the set of all

feasible points. The *Lagrangian* of problem (A.1) is defined as

$$L(x, \lambda) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \ , \tag{A.2}$$

where $\lambda \in \mathbb{R}^m$ are called the *Lagrange multipliers* associated with the corresponding constraints. From this definition it follows that for any $\lambda \in \mathbb{R}^m, \lambda \geq 0$, together with any feasible point x, it holds that $L(x, \lambda) \leq f(x)$. The *Lagrange dual* problem to (A.1) is given by

$$\begin{array}{ll} \underset{\lambda \in \mathbb{R}^m}{\text{maximize}} & \varphi(\lambda) \\ & \lambda \geq 0 \ , \end{array} \tag{A.3}$$

with $\varphi(\lambda) := \inf_{x \in D} L(x, \lambda)$ being the Lagrange dual function, where we observe that $\varphi : \mathbb{R}^m \to \mathbb{R}$ is always a concave function of λ (even in the case that f is not convex!). By our previous observation we have that *weak duality*

$$\varphi(\lambda') \leq \sup_{\lambda \geq 0} \varphi(\lambda) \leq \inf_{\substack{x \in D, \\ g_i(x) \leq 0}} f(x) \leq f(x') \tag{A.4}$$

holds, for any feasible x' (meaning that $g_i(x') \leq 0$), and any $\lambda' \geq 0$.

A.2. Matrix Optimization Problems & Generalized Inequality Constraints

Optimizing over matrices instead of vectors is not in any way different than described above. We can always identify a matrix variable $X \in \mathbb{R}^{n \times n}$ by the corresponding vector in $\mathbb{R}^{n^2}$, or $\mathbb{R}^{\frac{1}{2}n(n+1)}$ for the case of symmetric matrices.

In any optimization problem (no matter if over vectors or matrices) we can also consider general inequality constraints. Here we will discuss one class of such constraints in more detail, namely semi-definite constraints, which are very important and appear naturally in many optimization tasks. For this we consider "constraint" functions $p_i : D \to \mathbb{S}^{n \times n}$ that map our optimization domain D to symmetric matrices.

$$\begin{array}{lll} \underset{X \in D}{\text{minimize}} & f(X) & \\ & g_i(X) \leq 0, & i = 1 \dots m_g \\ & h_i(X) = 0, & i = 1 \dots m_h \\ & p_i(X) \preceq 0, & i = 1 \dots m_p \end{array} \tag{A.5}$$

The *Lagrangian* of problem (A.5) is defined exactly as in the vector case

$$L(X, \lambda, \mu, \nu) := f(X) + \sum_{i=1}^{m_g} \lambda_i g_i(X) + \sum_{i=1}^{m_h} \mu_i h_i(X) + \sum_{i=1}^{m_p} \nu_i \bullet p_i(X) \ . \quad \text{(A.6)}$$

Using $A \bullet B \geq 0$ if $A \succeq 0$ and $B \succeq 0$, we see that the last term of the above sum is non-positive whenever $\nu_i \succeq 0$ and $-p_i(X) \succeq 0$, i.e. $p_i(X) \preceq 0$. The Lagrange dual function is again given by $\varphi(\lambda, \mu, \nu) := \inf_{X \in D} L(X, \lambda, \mu, \nu)$, resulting in the *Lagrange dual* problem to (A.5) being

$$\begin{aligned} \underset{\lambda,\mu,\nu}{\text{maximize}} \quad & \varphi(\lambda, \mu, \nu) \\ & \lambda \in \mathbb{R}^{m_g}, \lambda \geq 0 \ , \\ & \mu \in \mathbb{R}^{m_h} \ , \\ & \nu_i \in \mathbb{R}^{n \times n}, \nu_i \succeq 0 \quad \forall i \ . \end{aligned} \quad \text{(A.7)}$$

See also [BV04, Section 5.9] for more details. In this more general form, *weak duality*

$$\varphi(\lambda', \mu', \nu') \leq \sup_{\lambda \geq 0, \mu, \nu_i \succeq 0} \varphi(\lambda, \mu, \nu) \leq \inf_{\substack{X \in D, \\ X \text{ feasible}}} f(X) \leq f(X') \quad \text{(A.8)}$$

holds for any feasible X', and any $\lambda' \geq 0, \mu' \in \mathbb{R}^{m_h}$, and $\nu_i' \succeq 0$.

A.3. Convex Optimization and the Wolfe Dual

If the functions $f, g_1, \ldots, g_m$ are convex and continuously differentiable, and the domain $D \subseteq \mathbb{R}^n$ is a convex set, then the corresponding optimization problem (A.1) has many further desirable properties, some of which we will briefly recall here.

Now for fixed non-negative multipliers λ, a point $\bar{x}$ is a minimizer of $L(x, \lambda)$ if and only if $\nabla_x L(\bar{x}, \lambda) = 0$, assumed that $D = \mathbb{R}^n$, and that $\varphi(\lambda) = \inf_{x \in D} L(x, \lambda)$ is attained[1]. This holds because f and g_i are convex and differentiable. Other domains D can be considered as well, making the characterization of these minimizers a bit harder. This means that the Lagrange dual optimization problem (A.3) is now equivalent to maximizing the so called *Wolfe dual* function $\omega(x)$, which is defined as $\omega(x) := \sup_{\lambda \geq 0} L(x, \lambda)$ under the constraint $\nabla_x L(\bar{x}, \lambda) = 0$, or in other

[1] Do we need to say more about $\varphi(\lambda) = \inf_{x \in D} L(x, \lambda)$ being a minimum, and the case that it is not attained?

words

$$\omega(x) := \sup_{\lambda \in \mathbb{R}^m} \quad \begin{array}{l} f(x) + \sum_{i=1}^m \lambda_i g_i(x) \\ \nabla f(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x) = 0 \ , \\ \lambda \geq 0 \ . \end{array} \tag{A.9}$$

If for some x the above problem becomes infeasible (meaning there exists no λ satisfying the constraints), then we set $\omega(x) := -\infty$. The optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{maximize}} \ \omega(x) \tag{A.10}$$

is called the *Wolfe dual* problem. In other words we were able to remove the minimization over x in the Lagrange dual function $\varphi(\lambda)$, by adding the equivalent optimality constraint $\nabla_x L = 0$ instead. The definition of $\omega(x)$ directly extends to generalized inequalities as in Section A.2, meaning that the supremum is now taken over all three parameters λ, μ and ν.

In some important cases we will have that this constraint $\nabla_x L = 0$ will specify a single feasible λ for each x, thus turning the problem (A.9) into a trivial function evaluation, which will be very practical for actual approximation algorithms, and their convergence analysis.

Lemma A.1 (Weak duality). *For any pair of points $x, x' \in D$, it holds that*

$$\omega(x) \leq f(x') \tag{A.11}$$

if x' is additionally feasible (meaning that $g_i(x') \leq 0 \ \forall i$)

Proof. This follows from the inequality $\omega(x) \leq \sup_{\lambda \geq 0} \varphi(\lambda)$ (which holds for any $x \in \mathbb{R}^n$, because we just have put more restrictions on λ), together with the weak duality formulation (A.4), meaning that $\sup_{\lambda \geq 0} \varphi(\lambda) \leq \inf_{x \in D,\, g_i(x) \leq 0} f(x) \leq f(x')$ for x' feasible. □

The Duality Gap. The non-negative quantity $g(x) := f(x) - \omega(x)$ is what we call the *duality gap* at any feasible point x, and will play a very important role as the main measure of approximation quality in the following of this work.

Probably the most useful property of the duality gap is that it serves as a certificate for the current primal error, meaning that for any feasible x, we have

$$f(x) - f(x^*) \leq g(x) \ .$$

In other words that the duality gap is always an upper bound on the current error to the unknown true optimum value $f(x^*)$.

This inequality directly follows from weak duality $\omega(x) \leq f(x^*)$ as given by (A.11), since the optimum point x^* is feasible.

Furthermore, convexity together with the existence of a *Slater point* (i.e. a point $x \in \operatorname{relint} D$ such that all inequality constrains are strictly satisfied, i.e. $g_i(x) < 0$) implies that *strong duality* holds, i.e. the middle inequality in (A.4) is in fact an equality. This means that the optimum value of the Lagrange dual problem (A.7), and therefore also the Wolfe dual problem (A.10), coincides with the primal optimum. See also [BV04, Section 5] for more details.

A.4. Convex Optimization over the Simplex

We consider optimizing a convex and differentiable function over the unit simplex, scaled with a fixed parameter $t > 0$. Formally, this means that the feasible region is given by $t \cdot \Delta_n = \left\{x \in \mathbb{R}^n \;\middle|\; x \geq 0,\ x^T\mathbf{1} = t\right\}$, and we study problems of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ s.t. \quad & \|x\|_1 = t \ , \\ & x \geq 0 \ . \end{aligned} \tag{A.12}$$

Why? The class of convex optimization problems over the simplex (A.12) includes many important problems from optimization and machine learning, namely ℓ_1-regularized least squares, the *Lasso* [Tib96], but also most variants of support vector machines, support vector regression, AdaBoost, multiple kernel learning over a fixed set of base kernels, the smallest enclosing ball problem, or financial applications such as mean-variance analysis for portfolio selection. Some of these applications we have discussed in more details in Chapters 2 and 6.

The Dual Problem. For the above optimization problem (A.12), the Lagrange dual problem (A.3) is the following:

$$\underset{\lambda \geq 0, \mu}{\text{maximize}} \ \inf_x \ L(x, \lambda, \mu) = f(x) - \sum_i \lambda_i x_i + \mu(x^T\mathbf{1} - t)$$

We know by convexity and differentiability of f that the infimum over x is attained if and only if $\nabla_x L(\bar{x}, \lambda, \mu) = 0$ (see also Section A.3). This is equivalent to $\nabla f(x) - \lambda + \mu\mathbf{1} = 0$, or in other words $\lambda = \nabla f(x) + \mu\mathbf{1}$. If we rewrite the objective function by plugging in this expression for λ, we have

$L(x,\lambda,\mu) = f(x) - x^T\lambda + \mu(x^T\mathbf{1} - t) = f(x) - x^T(\nabla f(x) + \mu\mathbf{1}) + \mu(x^T\mathbf{1} - t) = f(x) - x^T\nabla f(x) - \mu \cdot t$, and therefore the Wolfe dual problem is

$$\begin{array}{rl} \underset{x,\mu}{\text{maximize}} & f(x) - x^T\nabla f(x) - \mu \cdot t \\ s.t. & \nabla f(x) + \mu\mathbf{1} \geq 0 \end{array}$$

where the inequality constraint can be written as $\min_i(\nabla f(x))_i + \mu \geq 0$, so we know that for any fixed x, the maximum is always attained for $\mu = -\min_i(\nabla f(x))_i \geq 0$, or equivalently the Wolfe dual problem is

$$\underset{x}{\text{maximize}} \quad \omega(x) := f(x) - x^T\nabla f(x) + t \cdot \min_i(\nabla f(x))_i \tag{A.13}$$

This quantity indeed coincides with our definition of the "poor man's" dual for optimizing over the simplex, as we have explained in Sections 2.2 and 3.1.

The Duality Gap. Using the above definition of the dual, we immediately get a simple expression for the *duality gap* at any point $x \in \mathbb{R}^n$:

$$\begin{array}{rcl} g(x) & := & f(x) - \omega(x) \\ & = & x^T\nabla f(x) - t \cdot \min_i(\nabla f(x))_i \end{array} \tag{A.14}$$

This quantity is easily computable. From weak duality, we know that $g(x)$ is a very useful measure of approximation quality, that can be used to track guaranteed progress of any arbitrary optimization procedure or heuristic. Also observe that $g(x)$ is non-negative for every *feasible* $x \in D$ (this is exactly the point-wise weak duality we mentioned in Section A.3).

Note that $x := \left(\frac{t}{n}, \ldots, \frac{t}{n}\right) \in \text{relint}(D)$ is a Slater point satisfying all inequalities strictly (as $x_i > 0$), so we also know that strong duality must hold.

Optimization with Bounded Instead of Fixed $\|.\|_1$-Norm. We also consider slight relaxations of problem (A.12), where we only have an inequality constraint on the norm of x, i.e.

$$\begin{array}{rl} \underset{x}{\text{minimize}} & f(x) \\ s.t. & \|x\|_1 \leq t \ , \\ & x \geq 0 \ . \end{array} \tag{A.15}$$

Using the simple trick of adding an extra slack variable, we can immediately reduce this to the equality constrained formulation (A.12) that we

discussed before. We consider an additional variable x_{n+1} where our new function does not depend on x_{n+1}, formally $\bar{f}(x_1, \ldots, x_{n+1}) := f(x_1, \ldots, x_n)$ so that $(\nabla \bar{f}(x))_{n+1} = 0$.

The *duality gap* (A.14) in this case becomes

$$g(x) := x^T \nabla f(x) - t \cdot \min\{0, \min_i (\nabla f(x))_i\} \ , \tag{A.16}$$

which is again easy to compute, non-negative for every feasible x, and always at least as large as the traditional duality gap for the equality constrained case (A.14).

A.5. Convex Optimization with ℓ_∞-Norm Regularization

We consider optimizing a convex and differentiable function over the unit box, scaled with a fixed parameter $t > 0$. Formally, this means that the feasible region is given by $t \cdot \square_n = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq t\}$, and we study problems of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ s.t. \quad & \|x\|_\infty \leq t \ . \end{aligned} \tag{A.17}$$

Why? [MR11] have demonstrated that integer linear programs can be relaxed to convex problems of the above form, such that the solutions coincide with high probability under some mild additional assumptions.

The Dual Problem. For the above optimization problem (A.17), the constraints on each coordinate are $x_i \leq t$ and $x_i \geq -t$, so the Lagrange dual problem (A.3) is the following:

$$\underset{\lambda \geq 0, \mu \geq 0}{\text{maximize}} \ \inf_x L(x, \lambda, \mu) = f(x) - \sum_i \lambda_i (x_i + t) + \sum_i \mu_i (x_i - t)$$

Again by convexity and differentiability of f, we have that the infimum over x is attained if and only if $\nabla_x L(\bar{x}, \lambda, \mu) = 0$ (see also Section A.3). This is equivalent to $\nabla f(x) - \lambda + \mu = 0$. We can therefore equivalently write the above dual optimization problem in its Wolfe dual form (A.10), if we replace λ in the objective function $L(X, \lambda, \mu) = f(x) - (\nabla f(x) +$

$\mu)^T(x+t\mathbf{1}) + \mu^T(x-t\mathbf{1}) = f(x) - (x+t\mathbf{1})^T\nabla f(x) - 2t\mu^T\mathbf{1}$. The Wolfe dual can therefore be written as

$$\begin{array}{rl} \underset{X,\mu\geq 0}{\text{maximize}} & f(x) - (x+t\mathbf{1})^T\nabla f(x) - 2t\cdot\mu^T\mathbf{1} \\ s.t. & \nabla f(x) + \mu \geq 0 \end{array}$$

from which we can see that an optimal μ will always be $-\nabla f(X)$ (or 0 in case that becomes negative), or formally $\mu_i = \max\{-(\nabla f(x))_i, 0\} = -\min\{(\nabla f(x))_i, 0\}$. This means the above objective function becomes

$$\begin{array}{rl} & f(x) - (x+t\mathbf{1})^T\nabla f(x) + 2t\sum_i \min\{0, (\nabla f(x))_i\} \\ = & f(x) - x^T\nabla f(x) + t\sum_i \min\{0, (2\nabla f(x))_i\} - (\nabla f(x))_i \\ = & f(x) - x^T\nabla f(x) + t\sum_i \min\{-(\nabla f(x))_i, (\nabla f(x))_i\} \\ = & f(x) - x^T\nabla f(x) - t\cdot\|\nabla f(x)\|_1 \end{array}$$

In other words the Wolfe dual problem is

$$\underset{x}{\text{maximize}}\ \omega(x) := \quad f(x) - x^T\nabla f(x) - t\cdot\|\nabla f(x)\|_1\ . \tag{A.18}$$

Again, this quantity indeed coincides with our definition of the "poor man's" dual for optimizing over the $\|.\|_\infty$-norm unit ball, as we have explained in Sections 2.2 and 3.3.

The Duality Gap. Using the above definition of the dual, we immediately get a simple expression for the *duality gap* at any point $x \in \mathbb{R}^n$, i.e.

$$\begin{array}{rcl} g(x) & := & f(x) - \omega(x) \\ & = & t\cdot\|\nabla f(x)\|_1 + x^T\nabla f(x) \end{array} \tag{A.19}$$

This quantity is again easily computable, and a very useful measure of approximation quality, in the light of weak duality.

A.6. Semidefinite Optimization with Bounded Trace

We consider convex optimization problems where the feasible region is the spectahedron $t\cdot\mathcal{S}^n = \{X \in \mathbb{S}^{n\times n} \mid X \succeq 0,\ \mathrm{Tr}(X) = t\}$, scaled with a fixed parameter $t > 0$, or formally

$$\begin{array}{rll} \underset{X\in\mathbb{S}^{n\times n}}{\text{minimize}} & f(X) & \\ s.t. & \mathrm{Tr}(X) = t\ , & \\ & X \succeq 0 & \end{array} \tag{A.20}$$

Why? The class of convex optimization problems (A.20) includes many important problems from machine learning and compressed sensing, such as any nuclear norm regularized convex optimization problem (A.21) as outlined in the following paragraph (and more in depth in Section 4.2). It also includes many variants of matrix completion, matrix factorizations and low rank recovery, kernel learning and also variants of metric learning as well as manifold learning. We have investigated some of these applications in more details in Section 3.4 and Chapter 4, and studied solution paths of such problems in Chapter 7.

The Nuclear Norm for Matrices. The *nuclear norm* $\|A\|_*$ of any rectangular matrix $A \in \mathbb{R}^{m \times n}$, also known as the *trace norm*, is given by the sum of the singular values, or ℓ_1-norm of the spectrum of A. Alternatively, $\|A\|_*$ is the optimal value to the following SDP:

$$\begin{aligned} \|A\|_* = \underset{V,W}{\text{minimize}} \quad & t \\ s.t. \quad & \begin{pmatrix} V & A \\ A^T & W \end{pmatrix} \succeq 0 \quad \text{and} \\ & \operatorname{Tr}(V) + \operatorname{Tr}(W) \leq 2t \ . \end{aligned}$$

Here the two variables are symmetric matrices $V \in \mathbb{S}^{m \times m}$ and $W \in \mathbb{S}^{n \times n}$. From this transformation one can obtain that optimizing over rectangular matrices A with bounded nuclear-norm, i.e.

$$\underset{A \in \mathbb{R}^{m \times n},\, \|A\|_* \leq t}{\text{minimize}} f(A) \ , \tag{A.21}$$

is indeed equivalent to our above formulation (A.20) optimizing over symmetric matrices $X \succeq 0$ of bounded trace. Both the nuclear norm and this transformation are explained in more detail in Sections 4.2 and 4.4, obtaining a simple optimization algorithm for any nuclear norm regularized convex optimization problem of the form (A.21).

The Dual Problem. For the above optimization problem (A.20), the Lagrange dual problem (A.7) is the following:

$$\underset{\mu \in \mathbb{R}, \nu \succeq 0}{\text{maximize}} \ \inf_X \ L(X, \mu, \nu) = f(X) + \mu(\operatorname{Tr}(X) - t) - \nu \bullet X \ .$$

We know by convexity and differentiability of f that the infimum over X is attained if and only if $\nabla_X L(\bar{X}, \mu, \nu) = 0$ (see also Section A.3). This is equivalent to $\nabla f(X) + \mu \mathbf{I} - \nu = 0$ (using $\operatorname{Tr}(X) = \mathbf{I} \bullet X$ so $\nabla \operatorname{Tr}(X) = \mathbf{I}$),

or in other words $\nabla f(X) + \mu \mathbf{I} = \nu$. We can therefore equivalently write the above dual optimization problem in its Wolfe dual form (A.10), if we replace ν in the objective function $L(X, \mu, \nu) = f(X) + \mu(\mathrm{Tr}(X) - t) - X \bullet (\nabla f(X) + \mu \mathbf{I}) = f(X) - X \bullet \nabla f(X) - \mu \cdot t$. The Wolfe dual can therefore be written as

$$\begin{array}{rl} \underset{X,\mu \in \mathbb{R}}{\text{maximize}} & f(X) - X \bullet \nabla f(X) - \mu \cdot t \\ s.t. & \nabla f(X) + \mu \mathbf{I} \succeq 0 \end{array}$$

from which we can see that an optimal μ will always be minus the smallest eigenvalue of $\nabla f(X)$ (or equivalently the largest eigenvalue of $-\nabla f(X)$), meaning that $\mu = \lambda_{\max}(-\nabla f(X)) = - \max_{v \in \mathbb{R}^n, v \neq 0} \frac{-v^T \nabla f(X) v}{\|v\|^2}$, or in other words the Wolfe dual function is given by

$$\omega(X) := f(X) - X \bullet \nabla f(X) - t \cdot \lambda_{\max}(-\nabla f(X)) \ , \tag{A.22}$$

which coincides with our definition of the "poor man's" dual for semidefinite optimization over bounded trace, as we have explained in Sections 2.2 and 3.4.

The Duality Gap. This means that we have a *duality gap*

$$\begin{array}{rcl} g(X) & := & f(X) - \omega(X) \\ & = & t \cdot \lambda_{\max}(-\nabla f(X)) + X \bullet \nabla f(X) \ . \end{array} \tag{A.23}$$

This quantity is easily computable in practice.

From weak duality we again know that $g(X)$ is a very useful measure of approximation quality, that can be used to track guaranteed progress of any arbitrary optimization procedure or heuristic. Also observe that $g(x)$ is non-negative for every feasible $x \in \mathcal{S}_t$, which also follows from the weak duality.

Note that $X := t \cdot \mathbf{I}$ is a Slater point satisfying all inequalities strictly (as $\mathbf{I} \succ 0$), so we know that strong duality must hold.

The following lemma shows that analogously to the vector case over the simplex, any linear function over the spectahedron $\mathcal{S}_t$ does attain its minimum at a vertex (or extreme point) of the domain $\mathcal{S}_t$, namely at the specific rank-1 matrix given by the smallest eigenvector.

Lemma A.2. *For any symmetric matrix $G \in \mathbb{S}^{n \times n}$, it holds that*

$$\max_{X \in \mathcal{S}_t} G \bullet X = t \cdot \lambda_{\max}(G)$$

Proof. Exactly follows from the proof of Lemma 3.8, with the slight modification that $\sum_i \alpha_i = t$ instead of one. □

Optimization with Bounded Instead of Fixed Nuclear Norm. We also consider slight relaxations of problem (A.20), where we only have an inequality constraint on the trace of X, i.e.

$$\begin{array}{rl} \underset{X \in \mathbb{S}^{n\times n}}{\text{minimize}} & f(X) \\ s.t. & \operatorname{Tr}(X) \leq t \ , \\ & X \succeq 0 \end{array} \tag{A.24}$$

has the Wolfe dual problem being

$$\begin{array}{rl} \underset{X,\mu\in\mathbb{R}}{\text{maximize}} & f(X) - X \bullet \nabla f(X) - \mu \cdot t \\ s.t. & \nabla f(X) + \mu \mathbf{I} \succeq 0 \ , \\ & \mu \geq 0 \end{array}$$

which means that $\mu = \max\{0, -\lambda_{\min}(\nabla f(X))\}$, or in other words

$$\omega(X) = f(X) - X \bullet \nabla f(X) + t \cdot \min\{0, \lambda_{\min}(\nabla f(X))\} \tag{A.25}$$

The *duality gap* (A.23) now becomes

$$g_{\leq t}(x) := X \bullet \nabla f(X) + t \cdot \max\{0, \lambda_{\max}(-\nabla f(X))\} \ . \tag{A.26}$$

Alternatively, to derive this duality gap we can simply reduce our problem (A.24) to the equality constrained version (A.20), by appending an additional slack row/column to the matrix X, on which the function f does not depend.

In any case, the obtained duality gap here is again easy to compute, non-negative for every feasible X, and always at least as large as the traditional duality gap $g(X)$ for the equality constrained case (A.23).

A.7. Semidefinite Optimization with ℓ_∞-Bounded Diagonal

We can generalize the ℓ_∞-constrained (or "box"-constrained) vector optimization problems (A.17) to semidefinite matrices as follows: For a fixed parameter $t > 0$, we consider convex optimization problems of the form

$$\begin{array}{rl} \underset{X \in \mathbb{S}^{n\times n}}{\text{minimize}} & f(X) \\ s.t. & X_{ii} \leq t \quad \forall i, \\ & X \succeq 0 \ . \end{array} \tag{A.27}$$

Why? The class of convex optimization problems (A.27) includes many important problems from machine learning, such as any max-norm regularized convex optimization problem (A.28) as outlined in the following paragraph (and more in depth in Section 4.3). It also includes applications from spectral methods, spectral graph properties, as well as relaxations of combinatorial problems such as Max-Cut. For matrix completion problems, [SRJ04, LRS+10] argue that the max-norm does provide better generalization performance than the nuclear norm in some cases. We have investigated some of these applications in more details in Section 3.5 and Chapter 4.

The Max-Norm for Matrices. In Section 4.3, we will study max-norm regularized optimization problems in more detail. The *max-norm* $\|A\|_{\max}$ of any rectangular matrix $A \in \mathbb{R}^{m\times n}$, is given by the optimum of the semidefinite program

$$\|A\|_{\max} = \underset{V,W}{\text{minimize}} \quad t$$
$$s.t. \quad \begin{pmatrix} V & A \\ A^T & W \end{pmatrix} \succeq 0 \quad \text{and} \quad \begin{matrix} V_{ii} \leq t & \forall i \in [m], \\ W_{ii} \leq t & \forall i \in [n] \end{matrix}$$

Here the two optimization variables are again symmetric matrices $V \in \mathbb{S}^{m\times m}$ and $W \in \mathbb{S}^{n\times n}$. From this transformation one can obtain that optimizing over matrices A with bounded max-norm, i.e.

$$\underset{A\in\mathbb{R}^{m\times n},\, \|A\|_{\max}\leq t}{\text{minimize}} f(A) \ , \tag{A.28}$$

is indeed equivalent to our above formulation (A.27) optimizing over $X \succeq 0$ of bounded diagonal. We have studied this transformation in more detail in Sections 4.3 and 4.4, obtaining simple optimization algorithms and guarantees for any max-norm regularized convex optimization problem of the form (A.28).

The Dual Problem. For the above optimization problem (A.27), the Lagrange dual problem (A.7) is the following:

$$\underset{\mu\geq 0,\nu\succeq 0}{\text{maximize}} \ \inf_X \ L(X,\mu,\nu) = f(X) + \sum_i \mu_i(X_{ii} - t) - \nu \bullet X \ .$$

We know by convexity and differentiability of f that the infimum over X is attained if and only if $\nabla_X L(\bar{X},\mu,\nu) = 0$, which is equivalent to $\nabla f(X) + \operatorname{diag}(\mu) - \nu = 0$ (using $\sum_i \mu_i X_{ii} = \operatorname{diag}(\mu) \bullet X$ so that $\nabla_X \operatorname{diag}(\mu) \bullet X =$

$\mathrm{diag}(\mu)$). We can therefore equivalently write the above dual optimization problem in its Wolfe dual form (A.10), if we replace ν in the objective function $L(X, \mu, \nu) = f(X) + \mathrm{diag}(\mu) \bullet (X - t\mathbf{I}) - (\nabla f(X) + \mathrm{diag}(\mu)) \bullet X = f(X) - X \bullet \nabla f(X) - t \cdot \mu^T \mathbf{1}$. The Wolfe dual can therefore be written as

$$\begin{array}{rl} \underset{X,\mu \geq 0}{\text{maximize}} & f(X) - X \bullet \nabla f(X) - t \cdot \mu^T \mathbf{1} \\ s.t. & \nabla f(X) + \mathrm{diag}(\mu) \succeq 0 \ . \end{array} \tag{A.29}$$

For any fixed X, the above optimum is attained at $\mu = \mu_{(X)} \in \mathbb{R}^n$ being the solution to the SDP

$$\begin{array}{rl} \underset{\mu}{\text{minimize}} & \mu^T \mathbf{1} \\ s.t. & \nabla f(X) + \mathrm{diag}(\mu) \succeq 0 \ , \\ & \mu \geq 0 \ . \end{array} \tag{A.30}$$

So the duality gap is given by

$$\begin{array}{rcl} g(X) & := & f(X) - \omega(X) \\ & = & X \bullet \nabla f(X) + t \cdot \mu_{(X)}^T \mathbf{1}, \end{array} \tag{A.31}$$

where $\mu_{(X)} \in \mathbb{R}^n$ is the optimal solution to the SDP (A.30). We observe that any sub-optimal solution μ for (A.30) immediately gives an upper bound on the true duality gap $g(X)$, which is given by the optimal $\mu_{(X)}$.

This again coincides with our definition of the "poor man's" dual for semidefinite optimization over bounded diagonal, as we have explained in Sections 2.2 and 3.5. However, the alternative derivation from Lagrange and Wolfe duality only works for differentiable objective functions, and is significantly more complicated that the linearization approach in Section 2.2.

Using standard SDP duality, see e.g. [BV04], we obtain that the dual to the "helper" SDP (A.30) is in fact given by

$$\begin{array}{rl} \underset{Y}{\text{maximize}} & Y \bullet (-\nabla f(X)) \\ s.t. & Y_{ii} \leq 1 \quad \forall i, \\ & Y \succeq 0 \ . \end{array} \tag{A.32}$$

This is in fact the linearized version of the original problem (A.27) we want to solve. We have discussed the applications of this subproblem in more detail in Section 3.5.

Max-Cut. Note that if we would drop the $\mu \geq 0$ constraint in (A.30), and if $-\nabla f(X) = A$ would be the adjacency matrix of a graph, then instead of (A.32) we would in fact obtain the famous standard relaxation of the Max-Cut problem [GW95], namely

$$\begin{aligned} \underset{Y}{\text{maximize}} \quad & A \bullet Y \\ s.t. \quad & Y_{ii} = 1 \quad \forall i, \\ & Y \succeq 0 \ . \end{aligned} \tag{A.33}$$

for a fixed matrix A, see also Section 3.5, or the references [KGST03] and [BV04, page 219] where this is called the two-way partitioning problem.

Bibliography

[AGI11] Necdet Serhat Aybat, Donald Goldfarb, and Garud Iyengar. Fast First-Order Methods for Stable Principal Component Pursuit. *arXiv math.OC*, May 2011.

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. *FOCS 2005 - 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 2005.

[AHPV04] Pankaj Agarwal, Sariel Har-Peled, and Kasturi Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, 2004.

[AHPV05] Pankaj Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and Computational Geometry, MSRI Publications*, 52:1–30, 2005.

[AHPY07] Pankaj Agarwal, Sariel Har-Peled, and Hai Yu. Embeddings of surfaces, curves, and moving points in euclidean space. *SCG '07: Proceedings of the twenty-third annual Symposium on Computational Geometry*, 2007.

[APV02] Pankaj Agarwal, Cecilia Procopiuc, and Kasturi Varadarajan. Approximation Algorithms for k-Line Center. In *Algorithms — ESA 2002*, pages 425–432. 2002.

[AST08] Selin Damla Ahipasaoglu, Peng Sun, and Michael Todd. Linear convergence of a modified Frank–Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software*, 23(1):5–19, 2008.

[BB00] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in SVM classifiers. *ICML '00: Proceedings of the 17nd International Conference on Machine Learning*, 2000.

[BB09] Michel Baes and Michael Buergisser. Smoothing techniques for solving semidefinite programs with many constraints. *Optimization Online*, 2009.

[BC03] Mihai Bădoiu and Kenneth L Clarkson. Smaller core-sets for balls. *SODA '03: Proceedings of the fourteenth annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[BC07] Mihai Bădoiu and Kenneth L Clarkson. Optimal core-sets for balls. *Computational Geometry: Theory and Applications*, 40(1):14–22, 2007.

[Ber05] Pavel Berkhin. A survey on PageRank computing. *Internet mathematics*, 2(1):73, 2005.

[BGP09] Jean-François Bérubé, Michel Gendreau, and Jean-Yves Potvin. An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, 194(1):39–50, 2009.

[BGV92] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. *COLT '92: Proceedings of the fifth annual workshop on Computational Learning Theory*, 1992.

[BHH06] Francis Bach, David Heckerman, and Eric Horvitz. Considering Cost Asymmetry in Learning Classifiers. *Journal of Machine Learning Research*, 7:1713–1741, 2006.

[BHPI02] Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. *STOC '02: Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, 2002.

[BJMO11] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with Sparsity-Inducing Penalties. Technical report, August 2011.

[BL06] Jonathan M Borwein and Adrian S Lewis. Convex analysis and nonlinear optimization: theory and examples. CMS books in mathematics. Springer, 2006.

[BLJ04] Francis Bach, Gert R.G. Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, 2004.

[BM03] Samuel Burer and Renato D C Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[Bot10] Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *COMPSTAT'2010 - Proceedings of the 19th International Conference on Computational Statistics*, pages 177–187, 2010.

[BSS09] Joshua Batson, Daniel Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *STOC '09: Proceedings of the 41st annual ACM Symposium on Theory of Computing*, 2009.

[BT03] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[BT09] Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183, 2009.

[BTMN01] Aharon Ben-Tal, Tamar Margalit, and Arkadi Nemirovski. The Ordered Subsets Mirror Descent Optimization Method with Applications to Tomography. *SIAM Journal on Optimization*, 12(1):79, 2001.

[BTN05] Aharon Ben-Tal and Arkadi Nemirovski. Non-euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456, 2005.

[Bur98] Christopher Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[BV04] Stephen P Boyd and Lieven Vandenberghe. Convex optimization. 2004.

[CB00] David J Crisp and Christopher J C Burges. A Geometric Interpretation of ν-SVM Classifiers. *NIPS '00: Advances in Neural Information Processing Systems 12*, 2000.

[CCS10] Jian-Feng Cai, Emmanuel J Candes, and Zuowei Shen. A Singular Value Thresholding Algorithm for Matrix Completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[CDS98] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33, 1998.

[Cla10] Kenneth L Clarkson. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010.

[CLMW11] Emmanuel J Candes, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3), May 2011.

[CR09] Emmanuel J Candes and Benjamin Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[CT10] Emmanuel J Candes and Terence Tao. The Power of Convex Relaxation: Near-Optimal Matrix Completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[CZW+07] Edward Chang, Kaihua Zhu, Hao Wang, Hongjie Bai, Jian Li, Zhihuan Qiu, and Hang Cui. PSVM: Parallelizing Support Vector Machines on Distributed Computers. In *NIPS '07: Advances in Neural Information Processing Systems 20*, pages 257–264, 2007.

[d'A08] Alexandre d'Aspremont. Smooth Optimization with Approximate Gradient. *SIAM Journal on Optimization*, 19(3):1171, 2008.

[dBG07] Alexandre d'Aspremont, Francis Bach, and Laurent El Ghaoui. Full regularization path for sparse principal component analysis. *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, 2007.

[DeC06] Dennis DeCoste. Collaborative prediction using ensembles of Maximum Margin Matrix Factorizations. *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[dGJL07] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert R.G. Lanckriet. A Direct Formulation for Sparse PCA Using Semidefinite Programming. *SIAM Review*, 49(3):434–448, 2007.

[DH78] Joseph C Dunn and S Harshbarger. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62(2):432–444, 1978.

[Dun80] Joseph C Dunn. Convergence Rates for Conditional Gradient Sequences Generated by Implicit Step Length Rules. *SIAM Journal on Control and Optimization*, 18(5):473, 1980.

[DW00] Dennis DeCoste and Kiri Wagstaff. Alpha seeding for support vector machines. *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 2000.

[EHJT04] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[FHB01] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A Rank Minimization Heuristic with Application to Minimum Order System Approximation. *Proceedings American Control Conference*, 6:4734–4739, 2001.

[FHHT07] Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[FNW07] Mario A T Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient Projection for Sparse Reconstruction: Application to

Compressed Sensing and Other Inverse Problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[GG10] Nicolas Gillis and François Glineur. Low-Rank Matrix Approximation with Weights or Missing Data is NP-hard. *arXiv math.OC*, 2010.

[GGJW09] Bernd Gärtner, Joachim Giesen, Martin Jaggi, and Torsten Welsch. A Combinatorial Algorithm to Compute Regularization Paths. *arXiv.org*, cs.LG, 2009.

[GHW00] Thore Graepel, Ralf Herbrich, and Robert C Williamson. From Margin To Sparsity. *NIPS '00: Advances in Neural Information Processing Systems 12*, 2000.

[Gil66] Elmer G Gilbert. An Iterative Procedure for Computing the Minimum of a Quadratic Form on a Convex Set. *SIAM Journal on Control*, 4(1):61–80, 1966.

[GJ09] Bernd Gärtner and Martin Jaggi. Coresets for polytope distance. *SCG '09: Proceedings of the 25th Annual Symposium on Computational Geometry*, 2009.

[GJK88] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[GJL10] Joachim Giesen, Martin Jaggi, and Sören Laue. Approximating Parameterized Convex Optimization Problems. In *ESA 2010 - Proceedings of the 18th annual European Conference on Algorithms: Part I*, pages 524–535. LNCS, 2010.

[GJL12a] Joachim Giesen, Martin Jaggi, and Sören Laue. Approximating Parameterized Convex Optimization Problems. To appear in *ACM Transactions on Algorithms*, 2012.

[GJL12b] Joachim Giesen, Martin Jaggi, and Sören Laue. Regularization Paths with Guarantees for Convex Semidefinite Optimization. To appear in *AISTATS - Fifteenth International Conference on Artificial Intelligence and Statistics*, 2012.

[GJM10] Bernd Gärtner, Martin Jaggi, and Clément Maria. An Exponential Lower Bound on the Complexity of Regularization Paths. *arXiv*, cs.LG, 2010.

[GLW+09] Arvind Ganesh, Zhouchen Lin, John Wright, Leqin Wu, Minming Chen, and Yi Ma. Fast Algorithms for Recovering a Corrupted Low-Rank Matrix. In *CAMSAP - 3rd IEEE International Workshop*

on Computational Advances in Multi-Sensor Adaptive Processing, pages 213–216, 2009.

[GM11] Bernd Gärtner and Jiří Matoušek. Approximation Algorithms and Semidefinite Programming. Springer, December 2011.

[GNHS11] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. In *KDD '11 - Proceedings of the 17th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 2011.

[GPM89] Carlos García, David Prett, and Manfred Morari. Model predictive control: Theory and practice - A survey. *Automatica*, 25(3):335–348, 1989.

[GW95] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6), 1995.

[GZ05] Lacey Gunter and Ji Zhu. Computing the Solution Path for the Regularized Support Vector Regression. *NIPS '05: Advances in Neural Information Processing Systems 18*, 2005.

[Haz08] Elad Hazan. Sparse Approximate Solutions to Semidefinite Programs. In *LATIN 2008*, pages 306–316. Springer, 2008.

[Haz11] Elad Hazan. The convex optimization approach to regret minimization. In *Optimization for Machine Learning*. ie.technion.ac.il, 2011.

[HHB99] Arash Hassibi, Jonathan How, and Stephen P Boyd. A path-following method for solving BMI problems in control. In *Proceedings of the 1999 American Control Conference*, pages 1385–1389 vol.2., 1999.

[HPRZ07] Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. *IJCAI*, 2007.

[HRTZ04] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The Entire Regularization Path for the Support Vector Machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[HYZ08] Elaine T Hale, Wotao Yin, and Yin Zhang. Fixed-Point Continuation for ℓ_1-Minimization: Methodology and Convergence. *SIAM Journal on Optimization*, 19(3):1107, 2008.

[IR10] Alexander Ilin and Tapani Raiko. Practical Approaches to Principal Component Analysis in the Presence of Missing Values. *Journal of Machine Learning Research*, pages 1–44, 2010.

[Joa06] Thorsten Joachims. Training linear SVMs in linear time. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 2006.

[Jon92] Lee K Jones. A Simple Lemma on Greedy Approximation in Hilbert Space and Convergence Rates for Projection Pursuit Regression and Neural Network Training. *The Annals of Statistics*, 20(1):608–613, 1992.

[JS10] Martin Jaggi and Marek Sulovský. A Simple Algorithm for Nuclear Norm Regularized Problems. *ICML 2010: Proceedings of the 27th International Conference on Machine Learning*, 2010.

[JY09] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[Kal07] Satyen Kale. Efficient Algorithms using the Multiplicative Weights Update Method. PhD thesis, cs.princeton.edu, 2007.

[KBC07] Miklós Kurucz, Andras A Benczur, and Karoly Csalogany. Methods for large scale SVD with missing values. *KDD Cup and Workshop at the 13th ACM SIGKDD Conference*, 2007.

[KBP+10] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *ICRA 2010 - IEEE International Conference on Robotics and Automation*, pages 2665–2670, 2010.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, 42(8):30–37, 2009.

[KGST03] Jaz Kandola, Thore Graepel, and John Shawe-Taylor. Reducing Kernel Matrix Diagonal Dominance Using Semi-definite Programming. In *Learning Theory And Kernel Machines, LNCS*, pages 288–302. Springer Berlin / Heidelberg, 2003.

[KKB07] Kwangmoo Koh, Seung-Jean Kim, and Stephen P Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[KL96] Philip Klein and Hsueh-I Lu. Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In *STOC '96: Proceedings of the twenty-eighth annual ACM Symposium on Theory of Computing*, 1996.

[Kle99] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.

[KSBM00] S Sathiya Keerthi, Shirish K Shevade, Chiranjib Bhattacharyya, and K R K Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.

[KSST09] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. Technical report, Toyota Technological Institute - Chicago, USA, 2009.

[KW92] Jacek Kuczyński and Henryk Woźniakowski. Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

[KY10] Piyush Kumar and E Alper Yildirim. A Linearly Convergent Linear-Time First-Order Algorithm for Support Vector Classification with a Core Set Result. *INFORMS Journal on Computing*, 2010.

[KZ05] Andrew Kurdila and Michael Zabarankin. Convex functional analysis. Birkhäuser Verlag, 2005.

[LBD08] Jorge López, Álvaro Barbero, and José Dorronsoro. On the Equivalence of the SMO and MDM Algorithms for SVM Training. In *Machine Learning and Knowledge Discovery in Databases*, pages 288–300. 2008.

[LC06] Yoonkyung Lee and Zhenhuan Cui. Characterizing the solution path of multicategory support vector machines. *Statistica Sinica*, 2006.

[LGC07] Gaëlle Loosli, Gilles Gasso, and Stéphane Canu. Regularization Paths for nu-SVM and nu-SVR. *ISNN, International Symposium on Neural Networks, LNCS*, 4493:486, 2007.

[Lin07] Chih-Jen Lin. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Comput.*, 19(10):2756–2779, 2007.

[LMSS07] Nathan Linial, Shahar Mendelson, Gideon Schechtman, and Adi Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007.

[Lov83] László Lovász. Submodular functions and convexity. *Mathematical programming: The state of the art*, 1983.

[LRS+10] Jason Lee, Benjamin Recht, Ruslan Salakhutdinov, Nathan Srebro, and Joel A Tropp. Practical Large-Scale Optimization for Max-Norm Regularization. *NIPS 2010: Advances in Neural Information Processing Systems 23*, 2010.

[LS07] Gyemin Lee and Clayton D Scott. The One Class Support Vector Machine Solution Path. *ICASSP 2007 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2:521–524, 2007.

[LST09] Yong-Jin Liu, Defeng Sun, and Kim-Chuan Toh. An Implementable Proximal Point Algorithmic Framework for Nuclear Norm Minimization. *Optimization Online*, 2009.

[LU09] Neil Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[Mar52] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952.

[MCW05] Dmitry M Malioutov, Müjdat Cetin, and Alan S Willsky. Homotopy continuation for sparse signal representation. In *ICASSP '05 - IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 733–736 Vol. 5, 2005.

[MDM74] B Mitchell, V Dem'yanov, and V Malozemov. Finding the Point of a Polyhedron Closest to the Origin. *SIAM Journal on Control*, 1974.

[MGC09] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1):321–353, 2009.

[MHT09] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Submitted to JMLR*, 2009.

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research*, 11:1–36, 2010.

[MR11] Olvi L Mangasarian and Benjamin Recht. Probability of unique integer solution to a system of linear equations. *European Journal of Operational Research*, In Press, 2011.

[MST06] Michael E Mavroforakis, Margaritis Sdralis, and Sergios Theodoridis. A novel SVM Geometric Algorithm based on Reduced Convex Hulls. *ICPR '06: 18th International Conference on Pattern Recognition*, 2:564–568, 2006.

[MT06] Michael E Mavroforakis and Sergios Theodoridis. A geometric approach to Support Vector Machine (SVM) classification. *IEEE Transactions on Neural Networks*, 17(3):671–682, 2006.

[Nak10] Yuji Nakatsukasa. Absolute and relative Weyl theorems for generalized eigenvalue problems. *Linear Algebra and Its Applications*, 432(1):242–248, 2010.

[Nat95] Balas Kausik Natarajan. Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[Nem04] Arkadi Nemirovski. Prox-Method with Rate of Convergence O(1/t) for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems. *SIAM Journal on Optimization*, 15(1):229, 2004.

[Nem05] Arkadi Nemirovski. Lectures on modern convex optimization. Georgia Institute of Technology, 2005.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

[Nes04] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2004.

[Nes07a] Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE and INMA, Université catholique de Louvain, Belgium, 2007.

[Nes07b] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2007.

[Nes11] Yurii Nesterov. Random gradient-free minimization of convex functions. *CORE Tech Report*, February 2011.

[Nov63] Albert B Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, pages 615–622, 1963.

[OPT00] Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

[Osb92] Michael R Osborne. An effective method for computing regression quantiles. *IMA Journal of Numerical Analysis*, 12(2):151–166, 1992.

[Pan04] Rina Panigrahy. Minimum Enclosing Polytope in High Dimensions. *arXiv*, cs.CG, 2004.

[Pat93] Michael Patriksson. Partial linearization methods in nonlinear programming. *Journal of Optimization Theory and Applications*, 78(2):227–246, 1993.

[Pat98] Michael Patriksson. Cost Approximation: A Unified Framework of Descent Algorithms for Nonlinear Programs. *SIAM Journal on Optimization*, 8(2):561, 1998.

[Pat07] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *KDD Cup and Workshop at the 13th ACM SIGKDD Conference*, 2007.

[PB08] Markos Papadonikolakis and Christos-Savvas Bouganis. Efficient FPGA mapping of Gilbert's algorithm for SVM training on large-scale classification problems. *FPL 2008 - International Conference on Field Programmable Logic and Applications*, pages 385–390, 2008.

[PH07] Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

[Pla99] John C Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. 1999.

[RBCG08] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[Ren05] Jason D M Rennie. Regularized Logistic Regression is Strictly Convex. *people.csail.mit.edu*, 2005.

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3):471–501, 2010.

[Rit62] Klaus Ritter. Ein Verfahren zur Lösung parameter-abhängiger, nicht-linearer Maximum-Probleme. *Unternehmensforschung*, 6:149–166, 1962.

[Rit84] Klaus Ritter. On Parametric Linear and Quadratic Programming Problems. *Mathematical Programming: Proceedings of the International Congress on Mathematical Programming. Rio de Janeiro, 6-8 April, 1981*, pages 307–335, 1984.

[Roc97] R Tyrrell Rockafellar. Convex analysis. Princeton University Press, 1997.

[Roo00] Danny Roobaert. DirectSVM: a fast and simple support vector machine perceptron. *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, 1:356–365 vol.1, 2000.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[RR11] Benjamin Recht and Christopher Ré. Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion. *submitted*, 2011.

[RS05] Jason D M Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[RZ07] Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35(3):1012–1030, 2007.

[SB08] Christian D Sigg and Joachim M Buhmann. Expectation-Maximization for Sparse and Non-Negative PCA. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 960–967, 2008.

[SJ03] Nathan Srebro and Tommi Jaakkola. Weighted Low-Rank Approximations. *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, 2003.

[SRJ04] Nathan Srebro, Jason D M Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. *NIPS '04: Advances in Neural Information Processing Systems 17*, 17:1329–1336, 2004.

[SS05] Nathan Srebro and Adi Shraibman. Rank, Trace-Norm and Max-Norm. *COLT '05: Proceedings of the 18st annual Workshop on Computational Learning Theory*, 3559:545–560, 2005.

[SS10] Ruslan Salakhutdinov and Nathan Srebro. Collaborative Filtering in a Non-Uniform World: Learning with the Weighted Trace Norm. *NIPS 2010: Advances in Neural Information Processing Systems 23*, 2010.

[SSS05] Shai Shalev-Shwartz and Yoram Singer. A New Perspective on an Old Perceptron Algorithm. In *Learning Theory, LNCS*, pages 264–278. 2005.

[SSSZ10] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading Accuracy for Sparsity in Optimization Problems with Sparsity Constraints. *SIAM Journal on Optimization*, 20:2807, 2010.

[STC04] John Shawe-Taylor and Nello Cristianini. Kernel methods for pattern analysis. Cambridge University Press, 2004.

[TG07] Joel A Tropp and Anna Gilbert. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[Tib96] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[TKC05] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[TKK07] Ivor W Tsang, Andras Kocsor, and James T Kwok. Simpler core vector machines with enclosing balls. *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, 2007.

[TKZ06] Ivor W Tsang, James T Kwok, and Jacek M Zurada. Generalized Core Vector Machines. *IEEE Transactions on Neural Networks*, 17(5):1126–1140, 2006.

[TPNT09] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable Collaborative Filtering Approaches for Large Recommender Systems. *Journal of Machine Learning Research*, 10, 2009.

[Tro04] Joel A Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[Tur05] Berwin A Turlach. On Algorithms for Solving Least Squares Problems under an L1 Penalty or an L1 Constraint. *Proc. American Statistical Association; Statistical Computing Section*, 2005.

[TY07] Michael Todd and E Alper Yildirim. On Khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

[TY10] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 2010.

[VSM03] S V N Vishwanathan, Alex J Smola, and M Narasimha Murty. SimpleSVM. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, pages 760–767, 2003.

[Wan08] Gang Wang. A New Solution Path Algorithm in Support Vector Regression. *IEEE Transactions on Neural Networks*, 2008.

[WCYL06] Gang Wang, Tao Chen, Dit-Yan Yeung, and Frederick H Lochovsky. Solution Path for Semi-Supervised Classification with Manifold Regularization. *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 1124–1129, 2006.

[Web06] Brandyn Webb. Netflix Update: Try This at Home. Blog post sifter.org/˜simon/journal/20061211.html, 2006.

[WKS08] Markus Weimer, Alexandros Karatzoglou, and Alex J Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

[WSZS07] Kilian Q Weinberger, Fei Sha, Qihui Zhu, and Lawrence K Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *NIPS '07: Advances in Neural Information Processing Systems 20*, 2007.

[Wu07] Mingrui Wu. Collaborative filtering via ensembles of matrix factorizations. *KDD Cup and Workshop at the 13th ACM SIGKDD Conference*, 2007.

[WYL06] Gang Wang, Dit-Yan Yeung, and Frederick H Lochovsky. Two-dimensional solution path for support vector regression. *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 993–1000, 2006.

[WYL07] Gang Wang, Dit-Yan Yeung, and Frederick H Lochovsky. A kernel path algorithm for support vector machines. *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, 2007.

[WZLS08] Zhi-li Wu, Aijun Zhang, Chun-hung Li, and Agus Sudjianto. Trace Solution Paths for SVMs via Parametric Quadratic Programming. *KDD '08 DMMT Workshop*, 2008.

[ZdG10] Youwei Zhang, Alexandre d'Aspremont, and Laurent El Ghaoui. Sparse PCA: Convex Relaxations, Algorithms and Applications. *arXiv math.OC*, 2010.

[Zha03] Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2003.

[Zha11] Tong Zhang. Sparse Recovery with Orthogonal Matching Pursuit under RIP. *IEEE Transactions on Information Theory*, 57(9):6215–6221, September 2011.

[Zie95] Günter M Ziegler. Lectures on Polytopes, volume 152 of *Graduate Texts in Mathematics*. Springer Verlag, 1995.

[ZWSP08] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. 2008.

Curriculum Vitae

Martin Jaggi

born Mai 23, 1982 in St.Gallen, Switzerland,
citizen of Lenk, BE, Switzerland

1997 - 2001	Kantonsschule am Burggraben, St.Gallen, Switzerland Degree: Matura, Type C
2001 - 2006	Studies of Mathematics at ETH Zürich, Switzerland Degree: Dipl. Math. ETH
since 2007	Ph.D. student at ETH Zürich, Switzerland Institute of Theoretical Computer Science