Optimization Methods for Semi-Supervised Learning

by

Edward Cheung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Javier Peña Professor, Operations Research, Carnegie Mellon University
Supervisor(s):	Yuying Li Professor, Computer Science, University of Waterloo
Internal Member:	Justin Wan Professor, Computer Science, University of Waterloo
Internal Member:	Pascal Poupart Professor, Computer Science, University of Waterloo
Internal-External Member:	Stephen Vavasis Professor, Combinatorics & Optimization , University of Waterloo

Abstract

The goal of this thesis is to provide efficient optimization algorithms for some semi-supervised learning (SSL) tasks in machine learning. For many machine learning tasks, training a classifier requires a large amount of labeled data; however, providing labels typically requires costly manual annotation. Fortunately, there is typically an abundance of unlabeled data that can be easily collected for many domains. In this thesis, we focus on problems where an underlying structure allows us to leverage the large amounts of unlabeled data, while only requiring small amounts of labeled data. In particular, we consider low-rank matrix completion problems with applications to recommender systems, and semi-supervised support vector machines (S^3VM) to solve binary classification problems, such as digit recognition or disease classification.

For the first class of problems, we study convex approximations to the low-rank matrix completion problem. Instead of restricting the solution space to low-rank matrices, we use the trace norm as a convex surrogate. Unfortunately, many trace norm minimization algorithms scale very poorly in practice since they require a full singular value decomposition (SVD) at each iteration. Recently, there has been renewed interest in the trace norm constrained problem utilizing the Frank-Wolfe algorithm, which only requires calculating the leading singular vector pair, providing an order of magnitude improvement on the iteration complexity. However, the Frank-Wolfe algorithm empirically has very slow convergence and in practice yields high-rank solutions, which greatly increases computational costs. To address this issue, we investigate a *rank-drop step* for Frank-Wolfe, which solves a subproblem specifically designed to decrease the rank of the iterate, ensuring that the Frank-Wolfe algorithm converges along a low-rank path. We show that this rank-drop subproblem can be decomposed into two cases, where each subproblem can be solved efficiently and we guarantee that the iterates remain feasible, preserving the projection-free property of Frank-Wolfe.

Next we show that these ideas can be used to provide scalable algorithms for simultaneously sparse and low-rank matrix completion problems. We extend the Frank-Wolfe analysis to accommodate nonsmooth objectives, which can be used to solve the simultaneously sparse and low-rank problem. We replace the traditional linear approximation used in Frank-Wolfe by a uniform affine approximation to better address poor local approximations given by the first-order Taylor approximation. We show that this naturally leads to a sequence of smooth functions that uniformly converges to the original nonsmooth objective, allowing for a careful balance between approximation quality and convergence that is closely related to the step sizes of the Frank-Wolfe algorithm. We apply this algorithm to solve sparse covariance estimation problems, graph link prediction, and robust matrix completion problems.

Finally, we propose a variant of self-training for the semi-supervised binary classification problem by leveraging ideas from S^{3}VM. To address common issues associated with self-training, such as error propagation and label imbalances, we proposed an adaptive scheme using the functional margin of S^{3}VM to construct a confidence measure. The confidence score is used to create rules to adapt the optimization problems to incorporate label uncertainty and class imbalances. Moreover, we show that the incremental training approach leverages warm-starts very well, leading to much faster training than standard S^{3}VM methods alone, with much stronger empirical performance on imbalanced datasets.

Acknowledgements

I would like to thank:

My supervisor, Yuying Li, who has helped support and guide the research ideas.

The Scientific Computing Lab for providing several meaningful relationships and fond memories. A special mention to Haofan Zhang for helping me transition into the lab, as well as Parsiad Azimzadeh for endless discussions about mathematics and philosophy.

The internal committee members - Justin Wan, Pascal Poupart, Stephen Vavasis, and Javier Peña - for volunteering their time to participate in my committee.

My family for supporting me and sacrificing so much to ensure that I could pursue my dreams.

Luna Wei, for providing endless support and patience, fully believing in me.

Dedication

To my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In many machine learning applications, unlabeled data is typically abundant. On the internet alone, companies are passively collecting massive amounts of data each day. Images uploaded, articles written, and online purchases all can be easily recorded and utilized for machine learning tasks. However, it is often very difficult to obtain labels for the data, usually requiring manual annotation. Thus, it seems natural to create algorithms that can leverage the massive amounts of unlabeled data without also requiring large amounts of labeled data.

1.1 Semi-supervised Learning

Semi-supervised learning (SSL) is a paradigm that incorporates both unlabeled and labeled data into the training process, allowing algorithms to leverage the abundant unlabeled data. The main idea is to utilize any structure given by the unlabeled data to infer the missing labels. Figure 1.1 illustrates that, incorporating the unlabeled data into the training set reveals structure to better guide a classifier.

A computational difficulty associated with SSL problems is that the optimization problems formulated to incorporate the structure typically lead to NP-hard problems [13, 85, 30]. Below, we give an overview of the problems we consider as well as the computational difficulties associated with the structure.

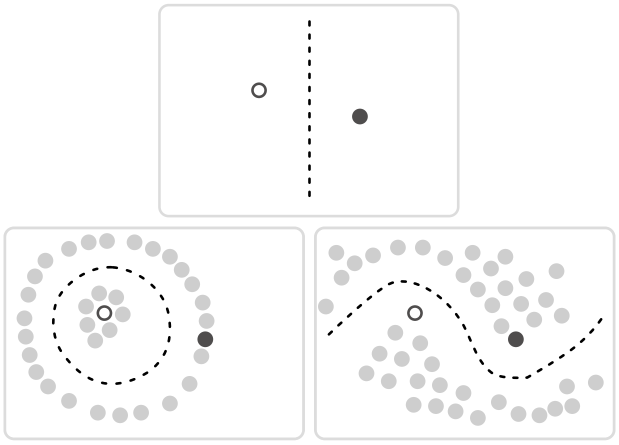


Figure 1.1: The structure given from the unlabeled data greatly helps with creating a classifier that should generalize better to unseen data. The top row is a classifier trained on only two labeled points. In the bottom row, we show how different sets of unlabeled data can greatly impact the choice of classifier.

1.1.1 Low-Rank Approximations

The first class of problems we consider is *low-rank approximation*. Finding a low-rank approximation is a natural solution for many recommender systems. Consider a problem such as recommending movies on Netflix. The ratings can be represented as a matrix where entry (i, j) yields the rating user i gave for movie j. Since most users do not rate most movies, we are only given access to a sparse set of observations from a true underlying rating matrix. To recover the missing entries, we seek a rating matrix that is consistent with the set of observations which is as low-rank as possible. The low-rank assumption is natural for recommender systems since we assume that there should be high level of correlation between ratings of similar users. Similarly, decomposing the matrix into rank-one matrices can be interpreted as factors that led to the predicted ratings, for example, one rank one factor can represent the preference of action movies. The low-rank assumption can also be interpreted as assuming that there should not be too many factors that contribute to movie preferences, and simple predictive models are preferred.

However, adding a rank-constraint or regularizer makes the optimization problem NP-hard. In this thesis, we will investigate efficient algorithms to solve convex relaxations to the low-rank problem where we utilize the *trace norm*. Despite the convex relaxation making the problem theoretically tractable, in practice, the computational complexity for many trace norm minimization are still too expensive for large machine learning applications. A very popular approach for trace norm constrained problems is *proximal gradient descent*, which requires a full singular value decomposition (SVD) at each iteration [62]. This leads to an iteration cost of $O(mn^2)$ for the proximal gradient algorithm, which is far too expensive when we consider that both m and n will be large. Interest in Frank-Wolfe methods have been recently renewed for trace norm constrained problems, largely due to the fact that the Frank-Wolfe subproblem for trace norm constrained problems only require computing the leading singular vector pair of the gradient, reducing the iteration complexity to $O(mn)$, an order of magnitude better. However, an obstacle with Frank-Wolfe methods is the poor observed convergence rates, leading to high-rank intermediate iterates, which in practice leads to much more expensive iterates and less interpretable solutions. We investigate methods for maintaining low-rank intermediate iterates while maintaining the projection-free nature of Frank-Wolfe in an effort to greatly reduce the computational time and space required for large-scale Frank-Wolfe on trace norm constrained problems.

We will also explore applications that exhibit a simultaneously sparse and low-rank structure, such as graph link prediction or sparse covariance estimation. Requiring regularization from both the ℓ_1 and trace norm terms make the optimization problem much more challenging since the proximal map and Frank-Wolfe subproblem are no longer simple to solve [69]. A simple approach to address this issue is to solve the problem in an alternating fashion via proximal gradient descent [69], but again, this leads to a full SVD at each iteration. Ideas using Frank-Wolfe have also been tried by using a smooth surrogate function to replace the ℓ_1 term and treating this problem as a trace norm constrained problem [65, 3], but we show these methods either quickly converge to suboptimal solutions, or take many iterations to make progress to a reasonable solution [17].

1.1.2 Semi-supervised Classification

The second class of problems we consider are *semi-supervised binary classification problems*. This paradigm is useful when we only have access to very small amounts of labeled data. This is particularly useful in the *imbalanced* setting, where having access to one class is particularly difficult, for example, diagnosing medical conditions or assessing fraud. Since getting a labeled point of data would require an expert such as a doctor to assess a patient, it is unlikely there will be an abundance of labeled data, and in particular, it may be very difficult finding patients that are positive for a rare disease. By leveraging the vast amount of unlabeled data, e.g. all patients that are not specifically tested for a particular disease, it may be possible to uncover structure to earlier identify warning signs of disease or fraud.

The structure we assume is that there exists a labeling of the data such that a decision boundary can be found that separates the classes sufficiently well and passes through a *low-density region*. In this context, passing through a low-density region means that very few points are close to the decision boundary, as in Figure 1.1. Optimizing over all labelings formulates a mixed-integer programming problem and the problems are NP-hard [16]. While approximations exist to the S^{3}VM problem [26, 54] which use semi-definite programming approaches, empirically, the solutions obtained are very sensitive to the initial labeling and can often be misled when the true labeling distribution is imbalanced [19]. In this thesis, we will investigate adaptive methods to find approximate solutions to the mixed-integer programming problem. We show that this nonconvex approach very naturally incorporates class distribution information and label confidence, improving the performance on imbalanced and multiclass classification tasks.

1.2 Contributions

The main contributions of this thesis are:

- We improve the computational performance of Frank-Wolfe on trace norm constrained problems by proposing *rank-drop steps*. We increase the set of candidate directions to ensure that the solution is always low-rank. Empirically, we have found that this dramatically decreases both the computational time and space required, leading to much more scalable algorithms. (Appeared in IJCAI 2017 [18]).

- We motivate a nonsmooth variant of Frank-Wolfe to address nondifferentiable objectives for matrix estimation problems. We show that the variant we propose outperforms many existing approaches utilizing subgradients or smooth surrogate functions. This extension is particularly useful for applications such as completing social graphs and sparse covariance estimation. (To appear in IJCAI 2018 [17]).

- We propose a variant of self-training for the semi-supervised binary classification problem. We use ideas from Semi-Supervised Support Vector Machines (S^{3}VM) to establish a confidence measure and create rules to adapt the optimization problems to incorporate label uncertainty and class imbalances. We show that the modified optimization problems motivated are extremely efficient in leveraging warm-starts and scale much faster than many existing SSL methods. The classification performance is also much stronger for datasets with imbalanced classes. (Appeared in IJCNN 2017 [19]).

1.3 Outline

The thesis is organized as follows:

- In Chapter 2, we give an overview of optimization algorithms for learning sparse structure.
- In Chapter 3, we describe the rank-drop steps. We highlight several computational issues associated with trace norm constrained problems for Frank-Wolfe and demonstrate how the rank-drop steps can overcome these issues [18].
- In Chapter 4, we motivate a natural way to incorporate nonsmoothness into the Frank-Wolfe algorithm. We show that this method improves upon existing nonsmooth solvers in the sparse and low-rank setting. Combining these ideas with Chapter 3, we demonstrate how to efficiently solve very large ℓ_1-loss matrix completion problems [17].
- In Chapter 5, we propose a self-training framework with adaptive regularization to solve semi-supervised binary classification problems. We show that our method greatly improves on existing SSL methods on imbalanced class settings [19].
- In Chapter 6, we summarize our results and provide possible directions for future work.

Chapter 2

Optimization Methods for Sparsity

In this chapter, we will give a brief overview of various convex optimization algorithms that are designed to recover solutions that exhibit a *sparse structure*. We will highlight the benefits and limitations of the various methods with an emphasis on the Frank-Wolfe method to motivate the problems to be studied.

2.1 Sparse Models

In many machine learning applications, the models desired exhibit a sparse structure to improve generalization performance and interpretability. The canonical example to consider is linear regression, where we observe m outcomes $\{y_i\}_{i=1}^m$, where each y_i is associated with a vector of covariates $x_i \in \mathbb{R}^n$. For example, we can imagine modeling housing prices where the price of house i is observed to be y_i, and we describe the house's features in the vector x_i, e.g. size, location, number of washrooms, etc.

A standard approach to model this behavior is to fit a linear function that minimizes the least squares distance,

$$\beta^* := \operatorname*{arg\,min}_{\beta \in \mathbb{R}^{n+1}} \|y - X\beta\|_2^2.$$

The coefficients β_j can be viewed as the importance associated with the j^{th} feature. Concretely, if $|\beta_j|$ is small, we can consider the j^{th} feature relatively unimportant for the modeling $\{y_i\}$.

A key idea proposed in [73] is that the model's performance can be improved if we prefer simpler models. The intuition is that only a few features should be important to the predictive performance of the model and the remaining features may only be fitting to the noise. To accomplish this, LASSO (least absolute shrinkage and selection operator) added a regularization term $\lambda\|\beta\|_1$ to the objective [73]. It is shown in [73] that this indeed produces sparser solutions and can improve generalization performance in practice.

The intuition that we wish to carry forward is that simplicity of the model can often be viewed as a parsimonious structure. Even considering the linear regression example further, the choice of linear regression versus a higher order polynomial can be viewed as a sparse choice of polynomial basis functions. The primary focus of this thesis will be on matrix estimation problems, where the sparsity will come from a *low-rank assumption*, which corresponds to the ability to describe the matrix with a small number of rank-one matrices.

2.2 Definitions and Notation

We will begin with various terminology and notation that will be used for the remainder of the thesis.

Matrix Notations

For square matrices $X \in \mathbb{R}^{n\times n}$, we define the trace as $\mathbf{tr}(X) = \sum_i X_{ii}$. We will also adopt the convention that λ_i is the i^{th} largest eigenvalue in magnitude of a matrix. We will use $\lambda_{\max}$ and $\lambda_{\min}$ to refer to the largest and smallest eigenvalues in magnitude. We also denote I_n to be the identity matrix in $\mathbb{R}^{n\times n}$, where we drop the subscript if the dimension is clear in the context.

For a general real matrix $X \in \mathbb{R}^{m\times n}$, we will use $\sigma_i(X)$ to represent the i^{th} largest singular value. If $\mathbf{rank}(X) = r$, we will say that $X = U\Sigma V^\top$ is a *thin singular value decomposition* (thin SVD) if $U \in \mathbb{R}^{m\times r}, \Sigma \in \mathbb{R}^{r\times r}$ and $V \in \mathbb{R}^{n\times r}$, where $U^\top U = I_r$, $V^\top V = I_r$, and Σ is a diagonal matrix with positive entries.

We let $\langle x, y\rangle$ denote the usual inner product $x^\top y$ for vectors and the trace inner product for matrices $\langle X, Y\rangle = \mathbf{tr}(X^\top Y)$.

Norms

For any norm $\|\cdot\|$, we will let $\|\cdot\|_*$ be the corresponding *dual norm* defined by,

$$\|x\|_* := \max_{y:\|y\|\le 1} \langle y, x\rangle. \tag{2.1}$$

For vectors $x \in \mathbb{R}^n$, we will consider the ℓ_p norms, $\|x\|_p = \sum_{i=1}^n \sqrt[p]{|x_i|^p}$ with $\ell_\infty = \max_i |x_i|$. Lastly, we will denote $\|x\|_0$ to be the number of nonzeros in x, which is not a norm.

For matrices $X \in \mathbb{R}^{m \times n}$, we will consider the following norms,

$$\begin{aligned}
\|X\|_F &\coloneqq \sqrt{\sum_{i=1}^m \sum_{j=1}^n |X_{ij}|^2}, \quad \text{(Frobenius Norm)} \\
\|X\|_1 &\coloneqq \sum_{i=1}^m \sum_{j=1}^n |X_{ij}|, \text{(Matrix } \ell_1 \text{ norm)} \\
\|X\|_{\text{tr}} &\coloneqq \sum_{j=1}^{\mathbf{rank}(X)} \sigma_i(X), \text{(Trace/Nuclear norm)} \\
\|X\|_{\text{sp}} &\coloneqq \sigma_1(X), \quad \text{(Spectral norm)}.
\end{aligned}$$

Common sets

For any norm, the associated norm ball $\mathcal{B}_a(x, \delta)$ is defined to be the open ball around x of radius δ, where the subscript a will correspond to the subscript of the norm of interest. Formally,

$$\mathcal{B}_a(x, \delta) \coloneqq \{y : \|x - y\|_a < \delta\}.$$

Define the *unit simplex of order* n to be

$$\Delta_n = \{x \in \mathbb{R}^{n+1} : \sum_{i=1}^{n+1} x_i = 1, x_i \geq 0\}. \tag{2.2}$$

Let $S \subseteq \mathbb{R}^n$. We denote the *convex hull* of S, $\mathbf{conv}(S)$, to be the minimal convex set containing S. The *closure* of a set X, written as $\mathbf{cl}(X)$ or $\bar{X}$ is denoted as the smallest closed set containing S, i.e. S and all its limit points. Given some set $C \subseteq \mathbb{R}^n$, the *affine hull* of C, denoted as $\mathbf{aff}(C)$, is the set

$$\mathbf{aff}(C) = \{\theta_1 x_1 + \cdots + \theta_k x_k : x_1, ..., x_k \in C, \ \sum_{i=1}^k \theta_i = 1\}.$$

For a nonempty convex set C, its *relative interior*, denoted as $\mathbf{relint}(C)$, is defined as

$$\mathbf{relint}(C) = \{x \in \mathcal{C} : \exists \epsilon > 0, \mathcal{B}(x, \epsilon) \cap \mathbf{aff}(C) \subseteq C\}.$$

If $S \subseteq \mathbb{R}^n$ is a convex set, we say that x is an *extreme point* of S if x cannot be written as a convex combination of distinct points in S. That is, x is an extreme point of S if $x = (1-\theta)a + \theta b$ for $\theta \in [0, 1]$ and $a, b \in S$, then $x = a$ and/or $x = b$. We say that F is a *face* of a convex set S if F is convex, and for any line segment $L \subseteq S$, if $\mathbf{relint}(L) \cap F \neq \emptyset$, then $L \subseteq F$.

Finally, for any function $f : \mathbb{R}^n \to \mathbb{R}$, the *subdifferential* of f at x is defined as,

$$\partial f(x) \coloneqq \{g : f(y) \geq f(x) + \langle y - x, g\rangle, \ \forall y \in \mathbb{R}^n\}.$$

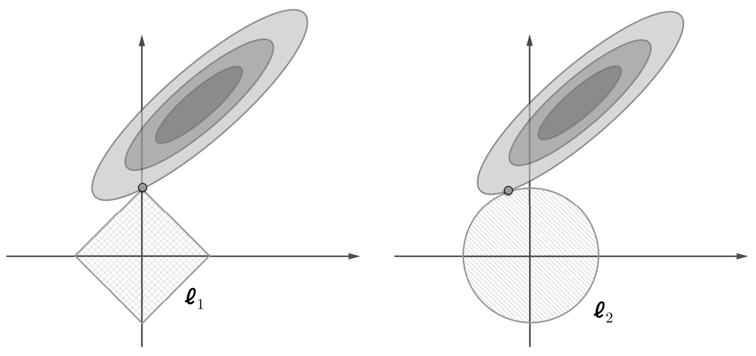
We will call any $g \in \partial f(x)$ a *subgradient* of f at x.

2.3 Sparsity Inducing Norms

When discussing LASSO, we have briefly mentioned that ℓ_1 regularization is enough to recover sparse solutions. We will provide some intuition on why this is the case.

Consider the level sets of the function we wish to minimize as in Figure 2.1. If we also draw $\mathcal{B}_1(0, \delta)$, if δ is large enough, the global minimum of the function may be contained in the norm ball. In this case, minimizing the function constrained to the norm ball of radius δ does not change the solution, and the additional ℓ_1 constraint does nothing. Now consider shrinking the radius of the ball. The optimal solution for the constrained problem now will be at the intersection of a level set that is tangent to $\mathcal{B}_1(0, \delta)$. The idea is that the level sets are more likely to intersect $\mathcal{B}_1(0, \delta)$ at a sparse solution. To see why, consider some dense solution in $\mathcal{B}_1(0, 1)$, e.g. $x = (1/n, 1/n, ..., 1/n)^\top$. We see that $\|x\|_2 = 1/\sqrt{n} < 1$. As the dimension grows larger, the Euclidean distance from the origin of this uniformly dense solution diminishes. The points which have maximum Euclidean distance from the origin on the unit ℓ_1 ball are the elementary basis vectors, which has distance 1, independent of the dimension. We see that when the mass of the support of the vector is more concentrated, then it can be further from the origin and more likely to be the point of tangency to the level set.

We see this claim of sparse support is true for all ℓ_p norms with $1 \leq p < 2$. A general trend we see is that as p grows larger, denser solutions become more likely to be chosen. In the case of $\mathcal{B}_\infty(0, 1)$, the point furthest from the origin on the unit ball would be the all



(a) ℓ_1 ball (b) ℓ_2 ball

Figure 2.1: Comparison of level sets intersecting the norm balls. Note that the ℓ_1 ball intersects at a sparser solutions than the ℓ_2 ball.

ones vector with Euclidean distance $\sqrt{n}$ from the origin, which is much further from the origin than the sparse elementary basis vectors.

When $0 \leq p < 1$, the sparse points become more exaggerated, but these are not norms and do not lead to convex optimization problems. A further justification to formalize the correctness of the ℓ_1 norm is to consider the convex relaxation to the ℓ_0 "norm" constrained problem. If we consider the unit ℓ_0 ball, then we recognize $\mathcal{B}_0(0,1) = \{ce_i : c \in \mathbb{R}\}$ where e_i are the elementary basis vectors. If we restrict our attention to a bounded set, say $\mathcal{B}_0(0,1) \cap \mathcal{B}_\infty(0,\delta)$ for some $\delta > 0$, then the convex hull of this set is precisely $\mathcal{B}_1(0,\delta)$. Thus, we can view the ℓ_1 constraint as the tightest convex relaxation to the ℓ_0 norm problem. Moreover, the fact that the extreme points of the $\mathcal{B}_1(0,1)$ are the elementary basis vectors will provide a key role for Frank-Wolfe methods.

The trace norm ball will be the primary focus in this thesis and often it appears as a convex surrogate for a low-rank matrix estimation, for example matrix completion for recommender systems [30, 8]. The intuition for using the trace norm as a surrogate for the rank constraint is similar to using the ℓ_1 for the sparsity constraint. We note that the rank of the matrix is the number of nonzero singular values. Since the trace norm is simply the ℓ_1 norm of the singular values, the same intuition will carry over.

2.4 Nonsmooth Convex Algorithms

We consider problems of the form,

$$\min_x F(x) := f(x) + g(x) \tag{2.3}$$

where f and g are convex and Lipschitz continuous. Relating back to machine learning problems, we can consider f to be a loss function and g to be a regularization term, such as the sparsity inducing norms discussed earlier. We will give a brief overview of methods to solve (2.3).

Subgradient Methods

If we do not assume that either function is differentiable, then we can consider *subgradient methods*.

The iterates for the subgradient method are of the form,

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \xi_k$$

where $\xi_k \in \partial F(x^{(k)})$ and $\alpha^{(k)} > 0$ is a step size. While the subgradient method is very flexible and assumes little about the objective function, it is shown in [59] that the algorithm converges at a rate of $O(1/\sqrt{k})$ and this bound is tight. Thus, for flexibility, we may have to accept very slow convergence rates.

Proximal Methods

While the convergence rate for the subgradient method is tight for general nonsmooth functions, if we assume additional properties on f, then it can be shown that the algorithms can be improved.

Definition 2.4.1. *We say that $f \in \mathbb{R}^n$ is L-*smooth *if f is differentiable and ∇f is Lipschitz continuous with Lipschitz constant L. That is for all $x, y \in \mathbb{R}^n$,*

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

Specifically, suppose f is L-smooth, then algorithms such as *proximal gradient descent* can be shown to improve the global convergence rate.

Recall the following concepts from convex optimization.

Lemma 2.4.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-smooth function. Then for any $x, y \in \mathbb{R}^n$,*

$$|f(y) - f(x) - \langle y - x, \nabla f(x) \rangle| \leq \frac{L}{2}\|y - x\|_2^2.$$

Note that when f is convex, L-smoothness implies that at each point x, there exists a quadratic centred around x that upper bounds the function globally, i.e.,

$$f(y) \leq f(x) + \langle y - x, \nabla f(x) \rangle + \frac{L}{2}\|y - x\|_2^2, \ \forall y \in \mathbb{R}^n. \tag{2.4}$$

The idea behind the algorithm is to replace the smooth term in the objective by the quadratic approximation given by (2.4). This gives an upper bound on the objective while potentially simplifying the problem. At each iteration k, we consider minimizing the following problem instead,

$$\begin{aligned} x^+ &:= \arg\min_x f(x^{(k)}) + \langle \nabla f(x^{(k)}), x - x^{(k)} \rangle + \frac{L}{2}\|x - x^{(k)}\|_2^2 + g(x) \\ &= \arg\min_x \frac{L}{2}\|x - (x^{(k)} - \frac{1}{L}\nabla f(x^{(k)}))\|_2^2 + g(x) \end{aligned}$$

These updates can be interpreted as finding a point x^+ that is close to the gradient step for f (with step size $1/L$) that is also good for g.

For a nonsmooth function g, we define its *proximal map* to be the operation,

$$\mathop{\mathbf{prox}}_{\alpha}(x) := \arg\min_z \frac{1}{2\alpha}\|x - z\|_2^2 + g(z).$$

The iterates of proximal gradient descent can then be written as,

$$x^{(k+1)} = \mathop{\mathbf{prox}}_{\alpha^{(k)}}\left(x^{(k)} - \alpha^{(k)}\nabla f(x^{(k)})\right)$$

where $\alpha^{(k)} \leq 1/L$.

In [6, 62], it is shown that proximal gradient descent converges at a rate of $O(1/k)$ and can be further accelerated to $O(1/k^2)$ using the momentum idea proposed in [57]. We see that as long as the proximal map is easy to compute, then we can achieve much faster convergence than subgradient descent.

The optimality conditions for the proximal map are given by

$$\nabla f(x^{(k)}) + \frac{1}{\alpha^{(k)}}(x - x^{(k)}) \in -\partial g(x).$$

Thus, when $g(x) = \lambda\|x\|_1$, the proximal map is given by,

$$\underset{\alpha^{(k)}}{\mathbf{prox}}\left(x^{(k)} - \alpha^{(k)}\nabla f(x^{(k)})\right) = \mathcal{S}_{\lambda\alpha^{(k)}}\left(x^{(k)} - \alpha^{(k)}\nabla f(x^{(k)})\right)$$

where $\mathcal{S}_\lambda(x)$ is the *soft-thresholding operator*

$$[\mathcal{S}_\lambda(x)]_i := \mathbf{sgn}(x_i)\max\{|x_i| - \lambda, 0\}. \tag{2.5}$$

When $g(X) = \lambda\|X\|_{\text{tr}}$, the proximal map can be computed as,

$$\underset{\alpha^{(k)}}{\mathbf{prox}}\left(X^{(k)} - \alpha^{(k)}\nabla f(X^{(k)})\right) = U\mathbf{diag}(\mathcal{S}_{\lambda\alpha^{(k)}}(\sigma))V^\top$$

where $U\mathbf{diag}(\sigma)V^\top$ is an SVD of $X^{(k)} - \alpha^{(k)}\nabla f(X^{(k)})$, and σ is the vector of singular values of $X^{(k)} - \alpha^{(k)}\nabla f(X^{(k)})$ [62].

For the trace regularized problem, the proximal map requires a *full SVD* at each iteration. Note that even if $X^{(k)}$ were low-rank, the matrix $X^{(k)} - \frac{1}{L}\nabla f(X^{(k)})$ could be full-rank, which incurs an $O(mn \cdot \min\{m,n\})$ cost for the full SVD, preventing scalability to very large datasets.

A main application we will focus on, for trace regularized problems, is the matrix completion problem, e.g.,

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\|P_\Omega(X - Y)\|_F^2 + \lambda\|X\|_{\text{tr}}. \tag{2.6}$$

Here Y is the true underlying matrix that we wish to estimate and $\Omega \subseteq \{1,...,m\}\times\{1,...,n\}$ is a collection of indices (i,j) where the value of Y_{ij} is known. The operation $P_\Omega(\cdot)$ restricts the loss only to the set Ω, i.e.,

$$P_\Omega(X) = \begin{cases} X_{ij}, \text{ if } (i,j) \in \Omega \\ 0, \text{ otherwise.} \end{cases}$$

For matrix completion problems, a key insight observed in [56] is that the proximal step is taken on a matrix that exhibits a "sparse + low-rank structure". To see why, note that $\nabla f(X) = P_\Omega(X - Y)$ is sparse, and $X^{(k)}$ is assumed to be low-rank and efficient solvers can be used.

Specifically when the step size $\alpha^{(k)} = 1$ for the proximal gradient step, this algorithm is known as the SoftImpute algorithm. Due to this structure, it is observed that SoftImpute typically scales much better than the accelerated version of the proximal gradient descent since the acceleration destroys the sparse + low-rank structure [82]. This highlights the fact that the SVD cost can become prohibitive in a large scale setting, that a simple algorithm can be preferable even when the convergence rate is much worse.

Projected Gradient Descent

If we consider the constrained form of (2.3) instead, i.e. $\min_x f(x)\ s.t.\ g(x) \leq \delta$ for some $\delta > 0$, we can see that this becomes a special case of the proximal gradient descent on the function $f(x) + I_{g(x)\leq\delta}(x)$, where $I_{g(x)\leq\delta}(x) = 0$ when $g(x) \leq \delta$, and ∞ otherwise. Thus, strong convergence for the projected gradient descent is immediately implied by the proximal algorithms. However, when $g(x) \leq \delta$ is a trace norm constraint, the same computational issues arise since the proximal operator becomes the projection operator, and projection onto the trace norm ball also requires a full SVD.

2.4.1 Summary of related methods

We have seen that for nonsmooth optimization, if the proximal map is cheap to compute, then proximal gradient descent can provide a large improvement over a simple method such as subgradient descent. Moreover, we see that the soft-thresholding operator used in the proximal map directly promotes sparsity which gives structured iterates. The obvious drawback that we emphasize is that the SVD required for the proximal map in the trace norm case can be too expensive for very large datasets. It is natural to wonder whether we can utilize the sparse structure of the problem more efficiently without sacrificing the improved convergence rate over subgradient methods. One of our main objectives in this thesis is to solve very large trace-norm constrained problems.

2.5 Frank-Wolfe

The Frank-Wolfe algorithm, also known as the Conditional Gradient algorithm, is a first order method proposed by Marguerite Frank and Philip Wolfe in 1956 to maximize concave quadratics over polytopes [31]. At a high level, the main idea of the algorithm is to generate iterates by minimizing the first order Taylor approximation over the domain. One of the original motivations for the polytope constraint is that each iteration reduces to an easy to solve linear program. The historical context is that, at the time, many efficient linear programming algorithms were just emerging and the Frank-Wolfe method provided a way to leverage this research to solve more complicated problems. This motivation alone is not very strong and unsurprisingly, the algorithm was not popular for many years. However, in recent years, there has been renewed interest where insights, as seen in [41, 46], in which the Frank-Wolfe algorithm is shown to have very nice properties for optimization problems with sparse structures. For the remainder of this chapter, we will highlight why the Frank-Wolfe algorithm succeeds in this problem setting, as well as potential drawbacks, in order to motivate our work.

2.5.1 Overview

The Frank-Wolfe method is a first order method for solving problems of the form

$$\min_{x \in \mathcal{D}} f(x)$$

where f is convex and differentiable and $\mathcal{D}$ is convex and compact.

The algorithm proposed by Frank and Wolfe considers a simple iterative scheme, where the first-order Taylor approximation is minimized on the domain at each iteration. Thus, at the current iterate $x^{(k)}$, a linear optimization problem is solved

$$s^{(k)} := \arg\min_{s \in \mathcal{D}} f(x^{(k)}) + \langle \nabla f(x^{(k)}), s - x^{(k)} \rangle = \arg\min_{s \in \mathcal{D}} \langle \nabla f(x^{(k)}), s \rangle. \tag{2.7}$$

We assume that we have access to a linear minimization oracle (LMO) that solves (2.7). For Frank-Wolfe to be useful, the LMO has to be able to solve (2.7) efficiently. This is similar to the assumption required in proximal methods that the proximal map must be evaluated efficiently. In fact, when we view $g(x) = I_{\mathcal{D}}$ as the indicator function, we see that the LMO and the proximal map only differ by the term $\frac{1}{2}L\|y - x\|_2^2$.

When $\mathcal{D}$ is a polytope, as in the Frank-Wolfe paper [31], this is a standard linear program and efficient oracles exist. To motivate why Frank-Wolfe updates are useful for sparse structure, we recall that for linear optimization, it is sufficient to consider the extreme points of the set. As alluded to earlier, the extreme points of the ℓ_1 ball are the elementary basis vectors are the *atoms*. Thus, the intuition for why Frank-Wolfe will be useful is that the solutions will be gradually built from these atoms, constructing a sparse solution incrementally. We will formalize this later in the chapter and show that the LMO for norm balls returns solutions efficiently.

The algorithm continues by taking a convex combination of the current iterate $x^{(k)}$ and the solution found in (2.7). That is,

$$x^{(k+1)} := (1 - \alpha^{(k)})x^{(k)} + \alpha^{(k)} s^{(k)}.$$

It is important to note that since $s^{(k)}$ is constrained to be feasible in (2.7) and $\mathcal{D}$ is a convex set, then $x^{(k+1)}$ is immediately feasible as long as x_0 is feasible. Thus, each iterate will be guaranteed to be feasible and we will call this property *projection-free*. Recall that an issue with projected gradient descent is that the projection operator requires a full SVD, preventing scalability despite the strong convergence rates. By avoiding this costly projection step, as long as the LMO can be computed quickly, there is potential for large computational savings.

The Frank-Wolfe algorithm is now summarized in Algorithm 1. There are many vari-

Algorithm 1 Frank-Wolfe (**FW**)

```
Input: Convex and differentiable $f$, convex and compact $\mathcal{D}$, max iteration count $T$, initial
    point $x_0 \in \mathcal{D}$.
Output: Solution $x^{(T+1)} \in \mathcal{D}$.
    for $k = 0...T$ do
        $s^{(k)} \leftarrow \arg\min_{s \in \mathcal{D}} \langle s, \nabla f(x_k) \rangle$
        $\alpha^{(k)} \leftarrow \frac{2}{k+2}$
        $x^{(k+1)} \leftarrow (1 - \alpha^{(k)}) x^{(k)} + \alpha^{(k)} s^{(k)}$
    end for
```

ations of choosing the appropriate step size $\alpha^{(k)}$ (see e.g. [32]) as well as various update rules for the next iterate $x^{(k+1)}$, (e.g. away steps [80] or the fully-corrective steps [46]). To guarantee convergence, a simple rule such as $\alpha^{(k)} = 2/(k+2), k = 0, 1, ...$ is sufficient [46].

2.6 Optimization over Norm Balls

It is not always obvious that there exists an efficient LMO to yield solutions to (2.7). An important class of problems that have efficient oracles are optimization problems over norm balls. This is of particular interest in machine learning when the norm is a *sparsity* inducing ball such as the ℓ_1 norm or the trace norm.

Theorem 2.6.1. *[45] Let $z \in \mathbb{R}^n$ and let $\|\cdot\|$ be an arbitrary norm in $\mathbb{R}^n$. Then,*

$$\underset{s:\|s\| \leq \delta}{\arg\min} \langle z, s \rangle = -\delta \cdot \partial \|z\|_*$$

where $\|\cdot\|_*$ denotes the dual norm defined as,

$$\|z\|_* = \max_{\|s\| \leq 1} \langle s, z \rangle.$$

Theorem 2.6.1 can be used to show that the LMO for Frank-Wolfe over norm balls only needs to return some $s^{(k)} \in -\delta \cdot \partial \|\nabla f(x^{(k)})\|_*$. This is an important result since characterizing dual norms and their subdifferentials is typically easier than characterizing the solution space of arbitrary linear optimization problems. Unless one is working with a very esoteric norm, these quantities are typically well known, e.g. [77] demonstrates how to compute several dual norms and their subdifferentials for common matrix norms. We contrast this with the proximal map, which may be difficult to evaluate, even for norm balls. An example is the latent overlapping group norm ball as proposed in [60], where the dual norm and its subdifferential are also simply stated, but the corresponding proximal map is very difficult to solve and requires specialized algorithms to approximate the solution [74].

Optimization over the ℓ_1-ball

For ℓ_p norms, the dual norms are well known to be ℓ_q norms such that $1/p + 1/q = 1$ and the dual norm of the ℓ_1 norm is the ℓ_∞ norm. It is easy to verify this from the definition,

$$\left(\max_{z:\|z\|_1 \le 1} \langle z, x \rangle \right) = \max_i |x_i| = \|x\|_\infty$$

where the solution is attained at $z = \mathbf{sgn}(x_i) e_i$, where e_i is the i^{th} elementary basis vector.

Thus,

$$\partial \|x\|_\infty = \mathbf{conv}\left\{ \mathbf{sgn}(x_i) e_i : x_i = \|x\|_\infty \right\}$$

and

$$s^{(k)} = \arg\min_{s:\|s\| \le \delta} \langle \nabla f(x), s \rangle = -\delta e_{i^*}$$

for any i^* such that $x_{i^*} = \|x\|_\infty$ and the Frank-Wolfe step can be evaluated in $O(n)$ time.

2.6.1 Optimization over the trace norm ball

The trace norm ball will be the primary focus in this thesis and often appears as a convex surrogate for the low-rank matrix estimation problem, e.g.,

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & f(X) \\ s.t. \quad & \|X\|_{\text{tr}} \le \delta. \end{aligned}$$

The canonical example is the Netflix Prize challenge where the matrix of movie ratings are best predicted using the low-rank matrix assumption [8, 30].

The dual norm of the trace norm is the spectral norm, i.e., the largest singular value. Since this fact seems less obvious than the ℓ_1 norm case, we will provide a proof of this claim for completeness.

Lemma 2.6.2. *The dual norm to the trace norm is the spectral norm. Explicitly, if $X \in \mathbb{R}^{m\times n}$ and*

$$\|X\|_{tr} = \sum_{i=1}^{\min\{m,n\}} \sigma_i(X)$$

where $\sigma_i(X)$ is the i^{th} largest singular value of X, then the dual norm of the trace norm, denoted as $\|X\|_*$, is

$$\|X\|_* = \|X\|_{sp} = \sigma_1(X)$$

Proof. Let $X = U\Sigma V^\top$ be an SVD of X. Let $Z = UV^\top$. We have,

$$\langle X, Z\rangle = \mathbf{tr}(X^\top Z) = \mathbf{tr}(V\Sigma U^\top UV^\top) = \|X\|_{\mathrm{tr}}.$$

Since $\|Z\|_{\mathrm{sp}} \le 1$, we have $\max_{\|Z'\|_{\mathrm{sp}}\le 1}\langle Z', X\rangle \ge \|X\|_{\mathrm{tr}}$.

In the other direction, we have,

$$\begin{aligned}
\max_{\|Z\|_{\mathrm{sp}}\le 1}\langle Z, X\rangle &= \max_{\|Z\|_{\mathrm{sp}}\le 1}\langle UZV^\top, \Sigma\rangle \\
&= \max_{\|Z\|_{\mathrm{sp}}\le 1} \sum_{i=1}^{\min\{m,n\}} u_i^\top Z v_i \sigma_i(X) \\
&\le \max_{\|Z\|_{\mathrm{sp}}\le 1} \sum_{i=1}^{\min\{m,n\}} \sigma_1(Z)\sigma_i(X) \\
&\le \|X\|_{\mathrm{tr}}
\end{aligned}$$

Thus, we have shown that $\max_{\|Z\|_{\mathrm{sp}}\le 1}\langle Z, X\rangle = \|X\|_{\mathrm{tr}}$, so the spectral norm is indeed the dual norm to the trace norm. □

We have established that the Frank-Wolfe steps can be determined by identifying an element in the subdifferential of the spectral norm. It only remains to characterize elements of the subdifferential.

Theorem 2.6.3. *[77] Let $X \in \mathbb{R}^{m\times n}$ and $\|\cdot\|_{sp}$ be the spectral norm. Then,*

$$\partial\|X\|_{sp} = \mathbf{conv}\{uv^\top : \|u\|_2 = \|v\|_2 = 1, Xv = \sigma_1 u\}.$$

Proof. (sketch) The full proof can be found in [77], but we remark that for the Frank-Wolfe algorithm, it is sufficient to verify that $\partial\|X\|_{\text{sp}} \supseteq \textbf{conv}\{uv^\top : \|u\|_2 = \|v\|_2 = 1, Xv = \sigma_1 u\}$ since we only require finding a single element of the subdifferential for the Frank-Wolfe algorithm.

Clearly, for all leading singular vector pairs (u_i, v_i),

$$u_i^\top X v_i = \sigma_1(X) = \|X\|_{\text{sp}} = \underset{Z:\|Z\|_{\text{tr}}\leq z}{\arg\max}\langle Z, X\rangle.$$

Thus, any $Z = u_i v_i^\top \in \partial\|X\|_{sp}$ for any leading singular vector pair. Moreover, if (u_1, v_1) and (u_2, v_2) are distinct singular vector pairs of X, then it is straightforward to verify that,

$$(1-\alpha)u_1v_1^\top + \alpha u_2 v_2^\top \in \partial\|X\|_{\text{sp}}$$

and

$$\partial\|X\|_{\text{sp}} \supseteq \textbf{conv}\{uv^\top : \|u\|_2 = \|v\|_2 = 1, Xv = \sigma_1 u\}.$$

□

Theorem 2.6.3 shows that the LMO for trace norm balls requires computing the leading singular vector pair of the matrix $X^{(k)}$.

We say that v is an ϵ-approximate eigenvector to A, if $\|v\| = 1$ and $v^\top A v \leq \lambda_{\max}(A) + \epsilon$. It is shown in [49, 52] that for a sparse matrix X, an ϵ-approximate eigenvector can be computed in $O(\texttt{nnz}(X)\log(n)/\sqrt{\epsilon})$ operations using an approximate Lanczos method, thus the cost of the LMO is almost linear in terms of the number of nonzeros of X.

Theorem 2.6.4. *[49] For any matrix $A \in \mathbb{R}^{m\times n}$, $\epsilon > 0$, and $\hat{\sigma} > \sigma_1(A)$ is an upper bound on the largest value of the singular values of A, then the Lanczos bidiagonlization algorithm returns a pair of unit vectors (u, v) such that,*

$$u^\top A v \geq \sigma_1(A) - \epsilon$$

with high probability, using at most, $\frac{\texttt{nnz}(A)\log(m+n)\sqrt{\hat{\sigma}}}{\sqrt{\epsilon}}$ flops.

2.7 Implications for Sparsity

We emphasize that the iterates formed from the solutions of the LMO for a sparsity inducing norm maintains a sparse structure. Specifically, for the ℓ_1 norm for example, at each iteration, the iterate adds at most one nonzero entry to the solution, and in the trace norm case, the rank increases by at most one.

This result is not surprising since the extreme points of the ℓ_1 ball are the signed elementary basis vectors. Similarly, the extreme points of the unit trace norm ball are the rank-one matrices with $\|X\|_{\text{sp}} \leq 1$. In both cases, the linear minimization oracle returns an extreme point as a solution allowing the Frank-Wolfe method to control the level of sparsity explicitly. Thus, Frank-Wolfe gives an interesting bridge between combinatorial constraints, such as rank or sparsity, by directly controlling the level of sparsity with the iteration count.

2.8 Convergence Analysis

Much of the analysis and intuition for the convergence properties of FW can be attributed to [45, 46, 50, 41]. In this section, we will reproduce the results and intuitions that will be useful for the remainder of the thesis.

2.8.1 Weak Duality

In this section, we will assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable and $\mathcal{D} \subseteq \mathbb{R}^n$ is a convex and compact set. We begin with a concept of *weak duality* proposed in [45].

Definition 2.8.1. *Let $x \in \mathcal{D}$ and $d_x \in \partial f(x)$. For a convex function f, the* **dual function** *is defined as,*

$$\omega(x, d_x) := \min_{y \in \mathcal{D}} f(x) + \langle y - x, d_x \rangle. \tag{2.8}$$

Immediately from the definition of subgradients, we get that $\omega(w, d_x) \leq f(y)$, for all $x, y \in \mathcal{D}$. However, as seen in [45], this simple notion of duality gap is very useful for analyzing Frank-Wolfe since the algorithm only requires the linear approximations. From this definition, we can compute the *duality gap*

$$g(x, d_x) := f(x) - \omega(w, d_x) = \max_{y \in \mathcal{D}} \langle x - y, d_x \rangle. \tag{2.9}$$

When f is differentiable at x, we will drop the argument d_x from the defintion and simply write $g(x)$. We note that $x^* = \arg\min_{x \in \mathcal{D}} f(x)$, $g(x, d_x) \geq f(x) - f(x^*) \geq 0$, by definition of ω and g.

The motivation for using this notion of duality over Lagrange duality is that the minimization problem in (2.8) is exactly the linear subproblem solved in each Frank-Wolfe iteration (2.7). Thus, the algorithm implicitly calculates the duality gap for free.

2.8.2 Curvature

Intuitively, for the Frank-Wolfe algorithm to be successful, the linear approximation to the function f at some $x \in \mathcal{D}$ cannot deviate too far from f for *any* point $y \in \mathcal{D}$ to ensure that the solution given by (2.7) is meaningful. This concept is formalized by the *curvature constant* described in [22, 45]. Let f be a convex and differentiable function on some convex and compact domain $\mathcal{D}$, then the curvature constant is defined as follows,

$$C_f \coloneqq \sup_{\substack{x,s\in\mathcal{D}\\ \alpha\in[0,1]\\ y=x+\alpha(s-x)}} \frac{1}{\alpha^2}\left(f(y) - f(x) - \langle y - x, \nabla f(x)\rangle\right). \tag{2.10}$$

We note that $C_f \geq 0$ when $f(y)$ is convex. The term $f(y) - f(x) - \langle y - x, \nabla f(x)\rangle$, known as the *Bregman divergence* associated with f measures the deviations between f and its linear approximation at x globally. For C_f to be bounded, we see that the term $1/\alpha^2$ ensures that the Bregman divergence is $O(\alpha^2)$ as $\alpha \to 0$. To see why this is desirable, we will show that the curvature constant is closely related to the Lipschitz constant of the gradient.

Theorem 2.8.2. *Let $f \in \mathbb{R}^n$ be a convex and twice differentiable function and suppose that ∇f is Lipschitz continuous with Lipschitz constant L. Then,*

$$C_f \leq \frac{1}{2}\mathbf{diam}(\mathcal{D})^2 L,$$

where $\mathbf{diam}(\mathcal{D}) \coloneqq \max_{x,y\in\mathcal{D}}\|x-y\|_2$ is the diameter of the set.

Proof. Let $x, s \in \mathcal{D}$ and $y = x + \alpha(s-x)$ for some $\alpha \in [0,1]$. Using the second order Taylor expansion of f at x, there is some z along the line-segment from x to y such that,

$$f(x+\alpha(s-x)) = f(x) + \alpha\langle\nabla f(x), s-x\rangle + \frac{\alpha^2}{2}\langle\nabla^2 f(z)(s-x), (s-x)\rangle. \tag{2.11}$$

Thus, we have that,

$$C_f \leq \sup_{\substack{x,s\in\mathcal{D}\\ \alpha\in[0,1]\\ z\in\mathcal{D}}} \frac{1}{2}\langle\nabla^2 f(z)(s-x), (s-x)\rangle \leq \frac{1}{2}\mathbf{diam}(\mathcal{D})^2 \max_{z\in\mathcal{D}}\|\nabla^2 f(z)\|_{\mathrm{sp}}$$

where the last inequality comes from applying Cauchy-Schwarz and using the definition of the diameter.

Finally, from Lemma 2.4.2 and (2.11), we have

$$\langle (L \cdot I - \nabla^2 f(z))(s - x), s - x \rangle \geq 0 \quad \forall s, x \in \mathcal{D}. \tag{2.12}$$

Hence, $L \cdot I \succeq \nabla^2 f(z)$, and $\|\nabla^2 f(z)\|_{\text{sp}} \leq L$, for all $z \in \mathcal{D}$, completing the claim. □

Next, we will show that when C_f is bounded, the Frank-Wolfe algorithm always converges. Since the bounded curvature requirement is closely related to the Lipschitz requirement on the gradient, this loosely translates to Frank-Wolfe succeeding if we can upper bound the function with a quadratic. We now have the tools to formerly prove the convergence rates of Frank-Wolfe.

2.8.3 Convergence Rate for Frank-Wolfe

We begin with a key lemma that gives a bound on the improvement of each Frank-Wolfe iteration.

Lemma 2.8.3. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a convex and differentiable function and let $g(x)$ and C_f be defined as in (2.9) and (2.10) respectively. For any $x \in \mathcal{D}$ and $\alpha \in [0, 1]$ it holds that,*

$$f(x + \alpha(s - x)) \leq f(x) - \alpha g(x) + \alpha^2 C_f$$

where $s \coloneqq \arg\min_{s' \in \mathcal{D}} \langle \nabla f(x), s' - x \rangle$.

Proof. From the definition of C_f, we have,

$$f(x + \alpha(s - x)) \leq f(x) + \alpha \langle \nabla f(x), s - x \rangle + \alpha^2 C_f.$$

By the choice of s it follows that,

$$\begin{aligned} \langle \nabla f(x), s - x \rangle &= \min_{s' \in \mathcal{D}} \langle \nabla f(x), s' - x \rangle \\ &= -\max_{s' \in \mathcal{D}} \langle f(x), x - s' \rangle \\ &= -g(x). \end{aligned}$$

This completes the proof. □

The next lemma encapsulates the main induction steps required to show the convergence of Frank-Wolfe.

Lemma 2.8.4. *Let $C \in \mathbb{R}$ with $C \geq 0$, $\alpha^{(k)} = 2/(k+2)$, and let $\{x^{(k)}\}$ be a sequence of real numbers. If the sequence $\{x^{(k)}\}$ satisfies*

$$x^{(k+1)} \leq (1-\alpha^{(k)})x^{(k)} + (\alpha^{(k)})^2 C \tag{2.13}$$

where $\alpha^{(k)} = 2/(k+2)$, then,

$$x^{(k)} \leq \frac{4C}{k+2}.$$

Proof. We will prove the desired result by induction. We see that when $k = 0$, $\alpha^{(k)} = 1$ and the bound is trivially satisfied.

Now suppose the result holds for some $k \geq 1$. We have that,

$$\begin{aligned}
x^{(k+1)} &\leq (1-\alpha^{(k)})x^{(k)} + (\alpha^{(k)})^2 C \\
&= \left(1 - \frac{2}{k+2}\right) x^{(k)} + \left(\frac{2}{k+2}\right)^2 C \\
&\leq \left(1 - \frac{2}{k+2}\right) \frac{4C}{k+2} + \left(\frac{2}{k+2}\right)^2 C \\
&= \left(\frac{1}{k+2} - \frac{1}{(k+2)^2}\right) 4C \\
&= \frac{4C}{k+2}\left(\frac{k+2-1}{k+2}\right) \\
&\leq \frac{4C}{k+2}\frac{k+2}{k+2+1} \\
&= \frac{4C}{(k+2)+1}
\end{aligned}$$

and we have proven the claim for $k \geq 1$. □

Now we can prove the main convergence result for Frank-Wolfe.

Theorem 2.8.5 ([45]). *Let $x^{(k)}$ be the k^{th} iterate given by Algorithm 1 and let $x^* \in \arg\min_{z\in\mathcal{D}} f(z)$. If the step-size at iteration k is given by $\alpha^{(k)} = 2/(k+2)$, then,*

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

Proof. Let $x^{(k+1)} := x^{(k)} + \alpha^{(k)}(s^{(k)} - x^{(k)})$ where $s^{(k)} := \arg\min_{s' \in \mathcal{D}} \langle \nabla f(x), s' - x \rangle$. From Lemma 2.8.3 we have that,

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

Since $g(x^{(k)}) \geq f(x^{(k)}) - f(x^*)$ we have,

$$\begin{aligned} f(x^{(k+1)}) - f(x^*) &\leq f(x^{(k)}) - f(x^{(k)}) - \alpha^{(k)}(f(x^{(k)}) - f(x^*)) + (\alpha^{(k)})^2 C_f \\ &= (1 - \alpha^{(k)})(f(x^{(k)}) - f(x^*)) + (\alpha^{(k)})^2 C_f \end{aligned}$$

The proof now follows from Lemma 2.8.4 since the sequence $\{f(x^{(k)}) - f(x^*)\}$ satisfies (2.13). $\square$

We observe that the error is bounded by $O(1/k)$ where the constant in the convergence rate is the curvature constant C_f. Thus, as expected, when the function is less curved and closer to linear, we expect the algorithm to converge quicker.

2.9 Accelerated First-Order Methods

A key idea in many first-order methods is the concept of *acceleration*, first proposed in [57]. Accelerated gradient descent is a very simple modification of gradient descent where a specially constructed *momentum term* is added. The momentum term has also been later adopted into many modern ideas such as FISTA [6], in which the momentum step can be incorporated into the proximal gradient step. We will give a quick overview of the accelerated gradient method as proposed in [57], which can be used to accelerate convergence for smooth unconstrained convex optimization problems.

Let f be L-smooth. Then, at each iteration, an intermediate term is calculated,

$$y^{(k+1)} = x^{(k)} - \frac{1}{L} \nabla f(x^{(k)}).$$

We recognize this as the gradient step with a fixed step size relating to the Lipschitz constant of the gradient. Next, define the following sequences,

$$\lambda^{(0)} = 0, \lambda^{(k)} = \frac{1 + \sqrt{1 + 4(\lambda^{(k-1)})^2}}{2}, \text{ and } \gamma^{(k)} = \frac{1 - \lambda_k}{\lambda_{k+1}}.$$

The accelerated gradient update is then given by

$$x^{(k+1)} = (1-\gamma^{(k)})y^{(k+1)} + \gamma^{(k)}y^{(k)}.$$

It was shown in [57] that for convex L-smooth functions, accelerated gradient descent converges at a rate of $O(1/k^2)$, which is optimal for smooth functions. The intuition behind why this momentum term helps with convergence is that as the algorithm approaches the minimizer, the gradient values also diminish, and the rate of progress also slows down. We might view the additional momentum term as pushing the algorithm further along the path it is currently going to overcome the diminishing gradient. An intuitive idea is that if we assume additional properties about the function, such as assuming the function is never "too flat", the rate of convergence may be further improved since the gradients diminishes more slowly. This concept is formalized with a *strong convexity* assumption.

Definition 2.9.1. *Given $\mu > 0$, a differentiable function f is μ-***strongly convex** *over some convex set $\mathcal{D}$, if for all $x, y \in \mathcal{D}$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y-x\|_2^2. \tag{2.14}$$

If we additionally assume that the function is *μ-strongly convex*, then it can be shown the iterates achieve a *global linear convergence rate*, i.e.,

$$f(x^{(k)}) - f(x^*) \in O\left(\rho^k\right)$$

for some $0 < \rho < 1$.

2.9.1 Challenges with Accelerating Frank-Wolfe

It is natural to wonder whether a simple modification of Frank-Wolfe can also achieve a faster rate similar to [57]. To figure out whether this is possible, we will investigate where potential issues occur in the convergence analysis.

We will first show that even under the strong convexity assumption, there will be hurdles that prevent fast convergence using the standard Frank-Wolfe algorithm.

Consider some function $f : \mathbb{R}^n \to \mathbb{R}$ that is L-smooth and μ-strongly convex. As seen in §2.4, the convexity of f and the L-smoothness allows us to write

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y-x\|_2^2$$

for any $x, y \in \mathbb{R}^n$.

Let $h_k = f(x^{(k)}) - f(x^*)$. As seen in the proof of Theorem 2.8.5, if $x^{(k+1)} = x^{(k)} + \alpha^{(k)}(s^{(k)} - x^{(k)})$ with $s^{(k)}$ being the solution given by the LMO, we can rewrite this as,

$$h_{k+1} \leq (1 - \alpha^{(k)})h_k + (\alpha^{(k)})^2 \frac{L}{2} \|s^{(k)} - x^{(k)}\|_2^2.$$

We see that the error can be written as a geometric decrease of the previous iteration plus some error term that is decreasing quadratically with the step size. A hurdle that prevents Frank-Wolfe from converging quickly is that the term $\|s^{(k)} - x^{(k)}\|_2^2$ has no reason to decrease and can be as large as the diameter of the set at any iteration. For many applications, we also expect this quantity to be large. For example, for trace-norm constrained problems, $s^{(k)}$ is a rank-one matrix and we do not expect $x^{(k)}$ to also be a rank-one matrix. Thus, we can expect $\|s^{(k)} - x^{(k)}\|_F^2 \geq \sigma_2(x^{(k)})$ due to the Eckart-Young-Mirsky theorem, which characterizes the best rank-k approximations to a matrix as the rank-k truncated SVD.

One may wonder why not choose the step size appropriately to ensure that $\alpha^{(k)} \frac{L}{2} \|s^{(k)} - x^{(k)}\|_2^2 \leq h_k/2$ since this would lead to the following expression,

$$h_{k+1} \leq \left(1 - \frac{\alpha^{(k)}}{2}\right) h_k$$

which appears to give geometric decrease. But again, we note that since $\|s^{(k)} - x^{(k)}\|_2^2$ does not decrease to 0, the step size sequence $\alpha^{(k)} \to 0$ and $(1 - \frac{\alpha^{(k)}}{2}) \to 1$, and the rate of decrease is not geometrically decreasing. We see that for the accelerated gradient descent, the fixed step size of $1/L$ plays an important role to guarantee that the error decreases geometrically.

Another idea one might consider is to generate the Frank-Wolfe iterates by directly minimizing the quadratic upperbound on the iteration error instead of the linear minimization problem. Specifically, one can consider solving

$$s^{(k)} \in \arg\min_{s \in \mathcal{D}} \langle f(x^{(k)}), s - x^{(k)} \rangle + \frac{L}{2} \|s - x^{(k)}\|_2^2. \tag{2.15}$$

Then from the L-smoothness and choice of $s^{(k)}$, we can write

$$\begin{aligned} h_{k+1} &\leq h_k + \alpha^{(k)} \langle \nabla f(x^{(k)}), s^{(k)} - x^{(k)} \rangle + (\alpha^{(k)})^2 \frac{L}{2} \|s^{(k)} - x^{(k)}\|_2^2 \\ &\leq h_k + \alpha^{(k)} \langle \nabla f(x^{(k)}), x^* - x^{(k)} \rangle + (\alpha^{(k)})^2 \frac{L}{2} \|x^* - x^{(k)}\|_2^2 \\ &\leq (1 - \alpha^{(k)})h_k + (\alpha^{(k)})^2 \frac{L}{2} \|x^* - x^{(k)}\|_2^2 \end{aligned}$$

Note that in the definition of μ-strongly convex functions, setting $x = x^*$ in (2.14) yields,

$$f(y) - f(x^*) \geq \frac{\mu}{2}\|y - x^*\|_2^2$$

for all $y \in \mathcal{D}$ since $\langle \nabla f(x^*), y - x\rangle \geq 0$ at optimality (no feasible descent directions).

Thus, the error bound can be further simplified to,

$$h_{k+1} \leq \left(1 - \alpha^{(k)} + (\alpha^{(k)})^2 L\mu\right) h_k.$$

Then setting $\alpha^{(k)} = 1/(\sqrt{2L\mu})$ is sufficient to guarantee global linear convergence. The issue with this approach is that solving (2.15) is exactly the projected gradient descent step which is unsuitable for large problems.

The last idea we will mention is to solve the Frank-Wolfe LMO with an additional constraint that the point s cannot be too far from x. By considering balls around x^* of radius $\sqrt{h_k}$, then we again see that it may be possible to achieve linear convergence as seen in [37]. However, this makes solving the subproblem much more expensive, particularly for trace norm constrained problems and the simple structure of the Frank-Wolfe iterates may be lost. Recently, there has been renewed interest in pursuing improved convergence rates for Frank-Wolfe specifically over trace norm balls by considering hybrid algorithms for Frank-Wolfe and proximal gradient descent for trace norm constrained problems [36, 2].

While global linear rates have been established under the L-smooth and μ-strongly convex assumptions, it was shown in [2] that outside the very low-rank case, there is insufficient improvements over standard Frank-Wolfe algorithms in terms of number of SVD solves required. However, it is worth noting that both [36] and [2] consider the matrix completion problem which is not strongly convex.

The main point we wish to highlight is that the simplicity of the Frank-Wolfe updates leads to slower convergence both theoretically and in practice. Since the solution to the LMO is an atom, as the algorithm proceeds, the information from each atom becomes less and less relevant, adding a large error term $\|s^{(k)} - x^{(k)}\|_2^2$ at each iteration.

This idea is formalized in [46], where it is shown that there exist problems such that the convergence is guaranteed to be slow in practice due to the simple "one atom at a time" nature of Frank-Wolfe.

Lemma 2.9.2. *[46] Let $f(x) = \|x\|_2^2$ and Δ_{n-1} be the unit simplex of order $n-1$. It holds that for any $1 \leq k \leq n$,*

$$\left(\min_{\substack{x \in \Delta_{n-1} \\ \|x\|_0 \leq k}} f(x)\right) = \frac{1}{k}.$$

We note that the extreme points of Δ_{n-1} are the elementary basis vectors of $\mathbb{R}^n$. Thus at each iteration of any Frank-Wolfe like algorithm, at most one nonzero entry can be updated in $x^{(k)}$. We can then bound the duality gap from below by $f(x^{(k)}) - f(x^*) \geq \frac{1}{k} - \frac{1}{n}$.

2.10 Improved Variants of Frank-Wolfe

In the previous section, we have highlighted several issues that can potentially prevent Frank-Wolfe from converging quickly. In this section, we will now present several variants that can improve the practical performance of Frank-Wolfe as well as certain problem settings where these variants theoretically guarantee global linear convergence.

We will begin by formally defining what we consider to be a "Frank-Wolfe variant".

Definition 2.10.1. *Suppose f is a convex function and $\mathcal{D}$ is a convex and compact domain. We call an algorithm which solves $x^* := \arg\min_{x\in\mathcal{D}} f(x)$ a* **Frank-Wolfe variant** *if the iterates can be written in the form,*

$$x^{(k+1)} = \mathbf{conv}(\{x^{(0)}, s^{(0)}, ..., s^{(k)}\})$$

where $s^{(j)} \in \arg\min_{s'\in\mathcal{D}}\langle d_x, s' - x^{(j)}\rangle$ where $d_x \in \partial f(x^{(k)})$.

Exact Line Search

Instead of fixing $\alpha^{(k)} = 2/(k+2)$, one might consider choosing the step size that satisfies,

$$\alpha^{(k)} := \underset{\alpha\in[0,1]}{\arg\min} f(x^{(k)} + \alpha(s^{(k)} - x^{(k)})).$$

A nice property of this line search is to note that if f is differentiable, $s^{(k)} - x^{(k)}$ is a descent direction since it minimizes $\langle \nabla f(x^{(k)}), s - x^{(k)}\rangle$. Thus, the exact line search variant will be a monotonically decreasing variant of Frank-Wolfe.

Away Steps

A common issue observed with the Frank-Wolfe algorithm is that if the optimal solution x^* is on the boundary and is not an extreme point, then the iterates tend to "zig-zag" between the vertices of the minimal face containing x^* [51]. As the current iterate gets

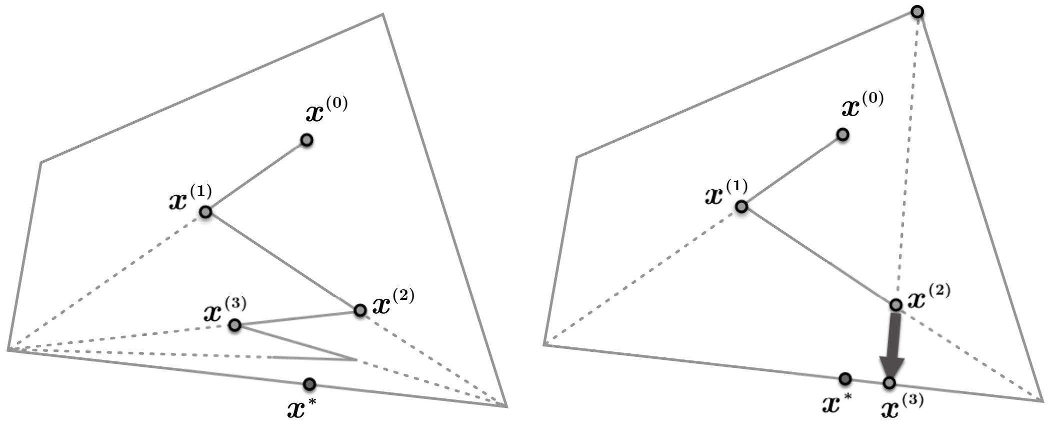


Figure 2.2: On the left, the zig-zagging phenomenon. On the right, the away-step.

closer to the boundary, the Frank-Wolfe directions become closer to being orthogonal to the gradient and progress is diminished [51].

If we define $\mathcal{A}_k := \{x^{(0)}, s^{(0)}, ..., s^{(k-1)}\}$, assuming $x^{(k)} \in \mathbf{conv}(\mathcal{A}_k)$, we can rewrite $x^{(k)} = \beta_k x^{(0)} + \sum_{i=0}^{k-1} \beta_i s^{(i)}$ for some $\beta \in \Delta_{n-1}$ where Δ_{n-1} is the unit simplex defined in (2.2). We call this an *atomic decomposition.* The set $\mathcal{S}_k = \{a \in \mathcal{A}_k : \beta_a > 0\}$ is referred to as the active set. Essentially, the current iterate is expressed as a convex combination of the atoms we have observed through the Frank-Wolfe algorithm.

The idea is to expand the set of directions considered by Frank-Wolfe to more directions than just the extreme points of the set. The away step then also considers the direction given by,

$$a^* \in \arg\min_{a' \in \mathcal{A}_k} \langle \nabla f(x^{(k)}), x^{(k)} - a' \rangle \tag{2.16}$$

which can be interpreted as the best direction found by moving away from any active vertex of $\mathbf{conv}(\mathcal{S}_k)$. We note that since $\mathcal{A}_k$ is a finite set of at most k elements, this minimization is typically much easier than solving the LMO problem.

If the away step yields a more promising descent direction, i.e.,

$$\langle \nabla f(x^{(k)}), x - a^* \rangle \leq \min_{s \in \mathcal{D}} \langle \nabla f(x^{(k)}), s - x^{(k)} \rangle$$

then the next iterate takes the form $x^{(k+1)} = x^{(k)} + \alpha^{(k)}(x^{(k)} - a)$, where $\alpha^{(k)} \in [0, \beta_a/(1 - \beta_a)]$. The values that $\alpha^{(k)}$ can take will ensure that $x^{(k+1)} \in \mathbf{conv}(\{x^{(0)}, s^{(0)}, ..., s^{(k)}\})$, hence $x^{(k+1)}$ is feasible. Note that the maximum value of $\alpha^{(k)}$ corresponds to setting the weight of the bad atom to zero.

A reason for using directions of this form can be appreicated by considering the weights of the atomic decomposition described. Suppose at iteration k that only $c < k$ atoms are active. But suppose further that x^* can be expressed by a strict subset of those c atoms, call this set A^*. Then what might be observed is that each successive Frank-Wolfe step returns an atom in A^*. Since the iterates are updated by a convex combination, we see that the weights of the atoms will gradually move to the atoms of A^*, and the remaining atoms will diminish at a rate of $(1 - \alpha^{(k)})$ at each iteration, where we expect $\alpha^{(k)} \to 0$. Thus, many iterations can be required to finally purge the "bad atoms" from the active set, leading to slow convergence. If instead the away-steps are considered, each bad atom may be removed in a single iteration moving the iterate toward the minimal face much quicker.

Fully-Corrective Steps

The last variant of Frank-Wolfe we will consider fully optimizes over the convex hull of the solutions provided by the LMO [46]. Formally, the iterates will be generated by,

$$x^{(k+1)} = \underset{x \in \mathbf{conv}(\{x^{(0)}, s^{(0)}, \ldots, s^{(k)}\})}{\arg\min} f(x).$$

We see that at iteration k, taking the fully-corrective step will lead to the best possible objective over all possible Frank-Wolfe variants. But this also comes at the obvious cost of solving a more difficult subproblem to determine the next iterate, where solving this subproblem can be just as hard as the original problem (e.g., if $\mathcal{D}$ is a polytope and $\mathbf{conv}(\{x^{(0)}, s^{(0)}, ..., s^{(k)}\}) = \mathcal{D}$).

2.11 Optimization over Polytopes

In [51, 64], it was shown that if the function is both L-smooth and μ-strongly convex and $\mathcal{D}$ is a polytope, then Frank-Wolfe can achieve global linear convergence with the away-step variant. In [63], it was shown that the ideas in [51, 64] are related by a geometric notion called *facial distances* which we will briefly summarize.

Let $\mathcal{A}$ be a set of m atoms in $\mathbb{R}^n$ and let $A \in \mathbb{R}^{m \times n}$ be a matrix in which the columns of A represent the atoms in $\mathcal{A}$. For any $x \in \Delta_{n-1}$, define $S(x) \coloneqq \{a_i \in \mathcal{A} : x_i \neq 0\}$. This is analogous to the set of active atoms in the atomic decomposition for the away step.

If $x, z \in \Delta_{n-1}$ such that $A(x-z) \neq 0$ and $d = A(x-z)/\|A(x-z)\|_2$, then we define the following quantity,

$$\Phi(A, x, z) := \min_{p \in \mathbb{R}^m : \langle p, d \rangle = 1} \max_{s \in \mathcal{S}(x), a \in \mathcal{A}} \langle p, s - a \rangle. \tag{2.17}$$

The *facial distance* is then minimizing the above quantity across all valid x and z, i.e.,

$$\Phi(A) := \min_{x, z \in \Delta_{n-1} : A(x-z) \neq 0} \Phi(A, x, z). \tag{2.18}$$

One can view taking an away step as moving the weight from a bad atom and uniformly redistributing its weight amongst the other active atoms. Instead of redistributing the weight uniformly, we can move all the weight from the bad atom to the best active atom. This concept is similar to the inner maximization optimization problem in (2.17) if we set $p = \nabla f(x^{(k)})$. Recall that an issue with the regular Frank-Wolfe steps is that the quantity $\langle \nabla f(x^{(k)}), s^{(k)} - x^{(k)} \rangle$ can become arbitrarily small as the direction becomes orthogonal to the gradient, slowing down convergence. In [63], the facial distance can be used to bound the quantity,

$$-\frac{\langle \nabla f(x^{(k)}), v \rangle}{\langle \nabla f(x^{(k)}), \frac{x^{(k)} - x^*}{\|x^{(k)} - x^*\|_2} \rangle} \geq \frac{\Phi(A)}{2}. \tag{2.19}$$

where v is either an away-step or a regular Frank-Wolfe step (whichever achieves the smaller inner product) and A is a finite set of atoms such that $\mathcal{D} = \mathbf{conv}\{A\}$. This shows that the directions generated from the away-step variant will not become close to orthogonal with the gradient, overcoming the issues from the Frank-Wolfe step.

2.12 Summary

We are motivated to study the Frank-Wolfe method due to the projection-free nature as well as the sparse structure given by the algorithm. In the next chapter, we will see that for trace norm constrained problems, this projection-free property will lead to significant computational advantages where the projection operator will require an expensive SVD.

However, there are two issues that we will investigate more closely over the next two chapters. Firstly, we will see that for the standard trace norm constrained matrix completion problem, the domain is not a polytope and the objective is not strongly convex. Therefore, we cannot take advantage of the linear convergence guaranteed by some of the Frank-Wolfe variants. We show that due to the slow convergence and large iteration count required, the algorithm no longer remains competitive with many state-of-the-art solvers. We will propose a *Rank-Drop step* variation to ensure that each iterate is indeed a low-rank solution, greatly decreasing the time and space requirements of the Frank-Wolfe subproblem.

Secondly, we will address the issue of bounded curvature and its role in Frank-Wolfe. Often, it is difficult to ensure that the curvature C_f is bounded restricting Frank-Wolfe to problems where the objective is L-smooth. For nonsmooth problems, a standard approach is to use a smooth surrogate to which Frank-Wolfe is applied on. This leads to extensive parameter tuning as well as tradeoffs between convergence rate and accuracy. We will propose a variant using *Uniform Affine Approximations* to motivate a parameter-free smoothing schedule without these tradeoffs.

Chapter 3

Rank-Drop Steps

3.1 Introduction

In this chapter, we will focus on trace norm constrained problems. As discussed earlier, there are some fundamental issues with accelerating Frank-Wolfe for trace norm constrained problems since the feasible region is not a polytope. Instead, we will focus on computational issues to improve convergence speed with respect to CPU time rather than iteration count. We will highlight computational issues with existing convex approaches, including Frank-Wolfe, which prevent these algorithms from scaling to very large problems. We will propose a *rank-drop step* which we will demonstrate greatly reduces the computational costs and space requirements, allowing Frank-Wolfe to scale to much larger problems.

If f is a convex and Lipschitz continuous function, then the trace norm constrained problem takes the form,

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) \ s.t. \ \|X\|_{\text{tr}} \leq \delta. \tag{3.1}$$

This problem is a typical convex relaxation for rank constrained optimization problems [30]. Common applications of trace norm constrained problems are matrix completion, multivariate regression, multi-task learning, and clustering with missing information [30, 46]. As discussed in the previous chapter, projection and proximal methods struggle to scale for solving (3.1) due to the expensive SVD operations required at each iteration. A key idea that motivates the study of Frank-Wolfe methods for solving (3.1) is the *projection-free* nature of the algorithm. As seen in Theorem 2.6.3, minimizing the Frank-Wolfe linear subproblem over trace norm balls only requires computing the largest singular vector pair of $\nabla f(X^{(k)})$. This is significantly cheaper than computing the full SVD when the dimension of the matrix is large.

In this chapter, we will focus on matrix completion tasks for clarity, although the methods we describe can immediately be used for any convex trace norm constrained problem.

3.1.1 Matrix Completion

Due to the Netflix Prize competition, the matrix completion problem has become one of the most popular and well-studied trace norm constrained problems [8, 30]. Suppose we are given a matrix $Y \in \mathbb{R}^{m \times n}$, where Y_{ij} can be viewed as the rating user i gives to movie j in the movie recommendation context. Usually, users only rate a very small subset of the movies and most of the entries in Y are unknown, i.e., Y is a sparse matrix.

Let $\Omega \subseteq \{1, ..., m\} \times \{1, ..., n\}$ be an index set representing the entries of Y that are observed, i.e., Y_{ij} is known if and only if $(i, j) \in \Omega$ and let $P_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ be the projection operator onto the observed set, where the entries are defined as,

$$[P_\Omega(X)]_{ij} := \begin{cases} X_{ij}, & \text{if } (i, j) \in \Omega \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The matrix completion optimization problem is then,

$$\min_{X \in \mathbb{R}^{m\times n}} \frac{1}{2} \|P_\Omega(X - Y)\|_F^2 \quad s.t. \quad \|X\|_{tr} \le \delta.$$

3.1.2 Computational Issues

When the $X^{(k)}$ iterates become too large to store in memory, a factorization of $X^{(k)}$ is used instead. Since Frank-Wolfe only adds a rank-one matrix at each iteration, an atomic decomposition is naturally obtained, reducing the storage cost of $X^{(k)}$ from $m \times n$ to $(m+n+1)\times k$, where we mention that typically $k \ll \min\{m, n\}$. We call this decomposition the *natural atomic decomposition*. However, this decrease in storage comes with an increase in computational cost since the factorized form requires additional operations to compute the gradient. For the matrix completion problem, the gradient is,

$$\nabla f(X^{(k)}) = P_\Omega(X^{(k)} - Y). \tag{3.3}$$

To compute $\nabla f(X^{(k)})$ from the atomic decomposition, $|\Omega| \cdot k$ operations are required. Despite Y being a sparse matrix, $|\Omega|$ is usually still very large for most applications of interest. For example in many recommendation services, $|\Omega|$ is in the order of ten to one hundred million entries. We see that the number of operations introduced by computing the gradient can quickly become too burdensome as the iteration count grows larger.

A computational challenge that arises for Frank-Wolfe when solving (3.1) is that the trace norm ball is not a polytope and hence, we cannot upper bound the number of atoms required to describe any current iterate $X^{(k)}$ using the atomic decomposition from Frank-Wolfe.

Since we expect the optimal solution X^* to have low-rank, a natural alternative decomposition would be to consider using the *thin SVD* of X^k, and the number of atoms required is upper bounded by $\min\{k, m, n\}$. We will briefly mention that due to the rank-one update nature of the algorithm, performing an SVD update can be carried out efficiently as seen in [10] and will not require a full SVD at each iteration. We will give further details on the implementation later in the chapter.

Clearly $\mathbf{rank}(X^{(k)}) \leq k$, but for any given iteration k, it is unclear if there is any difference between $\mathbf{rank}(X^{(k)})$ and k at all. In fact in practice, we often see that $\mathbf{rank}(X^{(k)}) = k$ for many iterations leading to *high-rank intermediate iterates*. As discussed during the away-steps in the previous chapter, the weights of any "bad atoms" degrade very slowly at a rate of $(1 - \alpha^{(k)})$ per iteration. Thus, even if the Frank-Wolfe algorithm would converge to a low-rank solution, it would take many iterations for the current iterate to sufficiently "deflate" and remove the unnecessary atoms to achieve a low-rank solution, and usually, a high rank solution is returned upon convergence.

3.2 Addressing the High-Rank Phenomenon

In this section, we will discuss potential variants to the Frank-Wolfe algorithm that could address the high-rank phenomenon seen for the standard Frank-Wolfe algorithm. The idea of using away steps seems very natural idea to remove bad atoms from the set of active atoms and hopefully reduce the rank of the current iterate.

3.2.1 Away Steps

The away step method maintains a set of active atoms, $\mathcal{A}_k$, such that the current iterate $X^{(k)}$ is decomposed into an *atomic decomposition*, i.e., $X^{(k)} = \sum_{A_i \in \mathcal{A}_k} \beta_{A_i} A_i$, where the weights vector satisfy $\beta_{A_i} > 0$ and $\sum_{A_i \in \mathcal{A}_k} \beta_{A_i} \leq 1$. The pseudo-code for the away step algorithm is provided in Algorithm 2.

Algorithm 2 Away Steps

```
1: Let $X^{(0)} \in \mathcal{D}$ and $\mathcal{A}^{(0)} \leftarrow \{X^{(0)}\}$.
2: Initialize $\beta_Z \leftarrow 0, \forall Z \in \mathcal{D}$.
3: for $k = 0...T$ do
4:     $S^{(k)} \leftarrow \arg\min_{S \in \mathcal{D}} \langle S, \nabla f(X^{(k)}) \rangle$
5:     $d_{fw}^{(k)} \leftarrow S^{(k)} - X^{(k)}$
6:     $V^{(k)} \leftarrow \arg\max_{V \in \mathcal{A}^{(k)}} \langle V, \nabla f(X^{(k)}) \rangle$
7:     $d_{away}^{(k)} \leftarrow X^{(k)} - V^{(k)}$
8:     if $\langle \nabla f(X^{(k)}), d_{fw} \rangle \leq \langle \nabla f(X^{(k)}), d_{away} \rangle$ then
9:         (Standard Frank-Wolfe step)
10:        $d^{(k)} \leftarrow d_{fw}$
11:        $\alpha^* \leftarrow \arg\min_{\alpha \in [0,1]} f(X^{(k)} + \alpha d^{(k)})$
12:        $\beta_{A_i} \leftarrow (1 - \alpha^*)\beta_{A_i},\ \forall A_i \in \mathcal{A}^{(k)}$
13:        $\beta_{S^{(k)}} \leftarrow \beta_{S^{(k)}} + \alpha^*$
14:        $\mathcal{A}^{(k)} \leftarrow \mathcal{A}^{(k)} \cup \{S^{(k)}\}$
15:    else
16:        (Away Step)
17:        $d^{(k)} \leftarrow d_{away}$
18:        $\alpha^* \leftarrow \arg\min_{\alpha \in \left[0, \frac{\beta_{V^{(k)}}}{1 - \beta_{V^{(k)}}}\right]} f(X^{(k)} + \alpha d_k)$
19:        $\beta_{A_i} \leftarrow (1 + \alpha^*)\beta_{A_i},\ \forall A_i \in \mathcal{A}^{(k)}$
20:        $\beta_{V^{(k)}} \leftarrow \beta_{V^{(k)}} - \alpha^*$
21:    end if
22:    $X^{(k+1)} \leftarrow X^{(k)} + \alpha^* d_k$
23:    $\mathcal{A}^{(k+1)} \leftarrow \{A \in \mathcal{A}^{(k)} : \beta_A > 0\}$
24: end for
```

While the pseudocode for the away-steps seems much longer than the standard Frank-Wolfe algorithm, the bulk of the code is spent updating the weights of the active set. When a standard Frank-Wolfe step is taken (the first block of the if statement), the next iterate of Frank-Wolfe can be written as,

$$X^{(k+1)} = (1 - \alpha^*)X^{(k)} + \alpha^* S^{(k)}$$

where α^* is the step size found from the exact line search and $S^{(k)}$ is the solution given by the LMO. Then the line,

$$\beta_{A_i} \leftarrow (1-\alpha^*), \forall A_i \in \mathcal{A}^{(k)} \tag{3.4}$$

uniformly decreases the weights for all the active atoms, which corresponds to the $(1-\alpha^*)X^{(k)}$ term, and adds a weight of α^* to the new atom, $S^{(k)}$.

When an away step is taken, the update to the atoms is slightly more complicated. The next iterate can be written as,

$$X^{(k+1)} = (1+\alpha^*)X^{(k)} - \alpha^* V^{(k)}$$

which explains why the atoms are updated as

$$\begin{aligned} \beta_{A_i} &\leftarrow (1+\alpha^*)\beta_{A_i} \\ \beta_{V^{(k)}} &\leftarrow \beta_{V^{(k)}} - \alpha^*. \end{aligned}$$

However, what is less clear is that $X^{(k+1)}$ is feasible since this is not written as a convex combination of points in $\mathcal{D}$. We observe that summing across the updated weights and using the fact that $V^{(k)}$ corresponds to some A_i,

$$\sum_{A_i \in \mathcal{A}^{(k)}} (1+\alpha^*)\beta_{A_i} - \alpha^* = (1+\alpha^*)\left(\sum_{A_i \in \mathcal{A}^{(k)}} \beta_{A_i}\right) - \alpha^* \leq 1.$$

Thus, as long as all the updated weights β_{A_i} are nonnegative, this will still be a valid convex combination. Let $\beta'_{V^{(k)}}$ denote the updated weight of $\beta_{V^{(k)}}$. Since only the weight of $V^{(k)}$ is decreasing, it is sufficient to verify that $\beta'_{V^{(k)}} \geq 0$. It is simple to verify that,

$$0 \leq \beta'_{V^{(k)}} = (1+\alpha^*)\beta_{V^{(k)}} - \alpha^* \Leftrightarrow \alpha^* \leq \frac{\beta_{V^{(k)}}}{1-\beta_{V^{(k)}}}$$

leading to the upper bound on α^* in the step size search. This is only a conservative step size search and it is possible that there exists a larger step size that would remain feasible. However, verifying feasibility for trace norm constraints, as discussed previously, is expensive and this upper bound has a nice interpretation of dropping an atom from the active set.

An issue with using the natural atomic decomposition (where the atoms correspond to the Frank-Wolfe steps) is that removing an active atom does not necessarily decrease the rank of the current iterate. Moreover, the step size returned by the exact line search may be much smaller than what is necessary to remove the atom from the active set and setting the step size to the maximum step size does not usually lead to decreases in objective value. We will see that away steps using the natural atomic decomposition, which we will call *natural away steps*, still usually lead to high-rank intermediate iterates.

The atomic decompositions are of course not unique, and instead of using the natural atomic decomposition given by the Frank-Wolfe steps to form the atoms, one might consider using the SVD to provide the atomic decomposition. The corresponding weights can then be calculated as, $\beta_i = \sigma_i(X^{(k)})/\delta$. This guarantees that any atom that is removed will guarantee a decrease in rank. Moreover, the weight of the atom naturally corresponds to the importance to the solution $X^{(k)}$. We will call this variant of away-steps the *SVD away step*. We contrast the SVD away steps with the weights given by the natural away steps where earlier iterations typically receive larger weights. A drawback of the SVD away step is that the set of directions are restricted to the rank-one matrices corresponding to the singular vector pairs, and the maximum step size is the corresponding singular values. Unsurprisingly, it is usually the singular vector pair corresponding to the smallest singular value which is chosen as the away step, and the corresponding step sizes are usually very small. Additionally, the directions given by the singular vector pairs do not necessarily give good descent directions. Because of these issues, the SVD away step variant typically converges much slower than the other Frank-Wolfe variants we will consider.

3.2.2 CoGEnT

The *Conditional Gradient with Enhancement and Truncation* (CoGEnT) method proposed in [67] also can be used to remove active atoms in a set. Instead of using a first-order approximation as in away-steps, CoGEnT uses a quadratic approximation to estimate the effect of removing atoms from the active set. Since the SVD is used for the atomic basis, the performance can still be limited by this choice of atomic basis.

3.2.3 In-Face Steps

In [33], *In-Face Steps* are proposed to generalize away-steps. Instead of generating descent directions by moving away from bad atoms, the in-face step selects the best descent direction along the *minimal face* containing the current iterate. In the polytope setting, the atoms in the atomic decomposition of $X^{(k)}$ must be a subset of the vertices of the minimal face containing $X^{(k)}$, denoted $\mathcal{F}_{\mathcal{D}(X^{(k)})}$. The main idea motivating the in-face step is that lower dimensional faces can usually be related to the sparse structure we desire, and in particular for trace norm constraints, moving to a lower dimensional face will lead to a lower rank solution. Thus, by prioritizing solutions on the minimal face containing the current iterate, the algorithm can maintain sparse structure throughout the algorithm.

For (3.1), as in [70, 33], the minimal face $\mathcal{F}(X^{(k)})$ of $\mathcal{B}_{\text{tr}}(0,\delta)$ containing a point $X^{(k)}$ is given by the set,

$$\mathcal{F}(X^{(k)}) = \begin{cases} \mathcal{B}_{\text{tr}}(0,\delta), & \text{when } \|X^{(k)}\| < \delta \\ UMV^\top, & \text{otherwise,} \end{cases} \tag{3.5}$$

where $X^{(k)}$ has a thin r rank SVD, $U\Sigma V^\top$, M is a real positive semidefinite matrix with $\text{tr}(M) = \delta$.

The pseudocode for the in-face algorithm can now be summarized below in Algorithm 3, where it is assumed that f is an L-smooth function.

The algorithm begins by finding any descent direction that stays in the current face. Then $\alpha_{\max}$ is calculated to determine the maximum step size that we can take in the direction $d^{(k)}$ and remain in the minimal face of $X^{(k)}$. $X_B^{(k)}$ is then a point on the boundary of the minimal face, and hence a lower dimension face, and $X_A^{(k)}$ is a point in the interior of the minimal face. Then, these candidates are checked for sufficient decrease in objective with a priority of moving to a lower dimension face over staying in the current face, where the term $B^{(k)}$ is the largest lower bound found for f^*. Finally, if the decrease condition is not satisfied or a descent direction $d^{(k)}$ is not found, then a regular Frank-Wolfe step is taken.

The main benefits of the in-face variant are twofold. First as mentioned earlier, the priority of moving to a lower dimensional face encourages a sparse structure for the iterates. Secondly, the sufficient decrease condition does not require knowledge about the Frank-Wolfe step, unlike the standard away step calculation. Thus, this linear subproblem may be skipped for each iteration an in-face step is taken.

There is some flexibility for how to choose $d^{(k)}$ and the variants are explored thoroughly in [33]. A natural choice for $d^{(k)}$, called the *away step strategy*, is to choose the direction of steepest descent that moves away from a point on the minimal face. That is,

$$d^{(k)} := X^{(k)} - \operatorname*{arg\,max}_{X \in \mathcal{F}(X^{(k)})} \langle \nabla f(X^{(k)}), X \rangle. \tag{3.6}$$

The authors of [33] also suggest choosing $\gamma_1 = 0$ and $\gamma_2 = \infty$, which corresponds to always taking any in-face step that moves to a boundary of the minimal face as long as it does not worsen the objective and never takes partial steps in the interior of the face. We will call this the InFace$(0,\infty)$ variant.

Algorithm 3 In-Face Steps

```
1: Let $X^{(0)} \in \mathcal{D}$ and $B^{(-1)}$ be an initial lower bound on $f^*$.
2: Choose constants $\bar{L} \geq L$, $\bar{D} \geq \mathbf{diam}(\mathcal{D})$, and constants $\gamma_1, \gamma_2$ such that $0 \leq \gamma_1 \leq \gamma_2 \leq 1$.
3: for $k = 0...T$ do
4:     $B^{(k)} \leftarrow B^{(k-1)}$
5:     Find $d^{(k)}$ such that $X^{(k)} + d^{(k)} \in \mathcal{F}_{\mathcal{D}}(X^{(k)})$ and $\langle \nabla f(X^{(k)}), d^{(k)} \rangle < 0$.
6:     $\alpha_{\max} \leftarrow \arg\max_\alpha \{\alpha : x^{(k)} + \alpha d^{(k)} \in \mathcal{F}_{\mathcal{D}}(X^{(k)})\}$
7:     $X_B^{(k)} \leftarrow X^{(k)} + \alpha_{\max} d^{(k)}$
8:     $X_A^{(k)} \leftarrow X^{(k)} + \alpha^{(k)} d^{(k)}$ for some $\alpha^{(k)} \in [0, \alpha_{\max})$.
9:     if $\frac{1}{f(X_B^{(k)}) - B^{(k)}} \geq \frac{1}{f(X^{(k)}) - B^{(k)}} + \frac{\gamma_1}{2\bar{L}\bar{D}^2}$ then
10:        (Check if there is sufficient improvement by taking the in-face step to the boundary. If true, move to lower-dimensional face.)
11:        $X^{(k+1)} \leftarrow X_B^{(k)}$
12:    else if $\frac{1}{f(X_A^{(k)}) - B^{(k)}} \geq \frac{1}{f(X^{(k)}) - B^{(k)}} + \frac{\gamma_2}{2\bar{L}\bar{D}^2}$ then
13:        (Check if there is a step in the current minimal face that gives sufficient decrease. If true, stay in current face.)
14:        $X^{(k+1)} \leftarrow X_A^{(k)}$
15:    else
16:        (No in-face step is sufficiently good. Do a regular Frank-Wolfe step)
17:        $S^{(k)} \leftarrow \arg\min_{S \in \mathcal{D}} \langle \nabla f(X^{(k)}), S \rangle$
18:        $X^{(k+1)} \leftarrow X^{(k)} + \alpha^{(k)}(S^{(k)} - X^{(k)})$ for some $\alpha^{(k)} \in [0, 1]$.
19:        $\tilde{B}^{(k)} \leftarrow f(X^{(k)}) + \langle \nabla f(X^{(k)}), S^{(k)} - X^{(k)} \rangle$
20:        $B^{(k)} \leftarrow \max\{B^{(k-1)}, \tilde{B}^{(k)}\}$
21:    end if
22: end for
```

A minor drawback of the in-face step we observe for the trace norm case is that the minimal face for any point in the interior of the ball is the entire trace norm ball. Thus any iterate in the interior of the ball will likely not decrease in rank even if an in-face step is taken. Moreover, the parameters of $\gamma_1 = 0$ and $\gamma_2 = \infty$ suggested by the authors of [33] indicate that the preference for the in-face steps are any steps that can decrease the rank of the current iterate without worsening the objective. However, the objective in (3.6) only accounts for local decrease in the objective value along the minimal face. Thus, it is likely possible to find better directions if we assume that a rank decrease is much more important than a decrease in the objective in any given iteration.

3.3 Rank-Drop Steps

If we analyze the InFace$(0, \infty)$ variant more carefully, we see the choice of $\gamma_2 = \infty$ seems unintuitive since this automatically rejects all directions that stay in the interior of the minimal face. From [33], it appears that, when $\gamma_2 = 1$ instead, the performance is strictly worse. A reason for this behavior is that the algorithm will spend several iterations optimizing over the current minimal face before exploring higher dimensional faces. When $\gamma_2 = \infty$ with $\gamma_1 = 0$, we typically observe the algorithm converging to a specific rank r by alternating between increasing the rank with a Frank-Wolfe step and decreasing the rank with an in-face step. It appears that as long as the in-face step can effectively reduce the rank of the current solution and maintain the rank r solution, the improvement from searching a higher dimensional face outweighs the drawback of temporarily increasing the rank of the solution compared to the InFace(0, 1) variant.

Since it appears that the we should only consider in-face directions that decrease the rank of the current iterate, we motivate a step that incorporates this idea specifically. We note that the optimization problem in (3.6) only considers the descent direction along the minimal face, which can lead to a promising descent direction in the interior of the minimal face with no guarantee of the objective quality at the boundary of the face. The goal for the proposed rank-drop steps will be to formulate an objective that better accounts for the solution after the rank is decreased. Additionally, to be useful for the trace norm constrained problems, the iterates should also maintain the projection-free nature of the Frank-Wolfe algorithm. We show that this leads to a challenging nonconvex optimization problem and propose efficient solutions to approximate better *rank-drop steps.*

In [28], for a given matrix A, a detailed characterization of all rank-one matrices that lead to a decrease in the rank of A is given.

Theorem 3.3.1 ([28]). *Let $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $B = A - \sigma^{-1}uv^\top$. Then* $\mathbf{rank}(B) = \mathbf{rank}(A) - 1$ *if and only if there are vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ such that $u = Ax$, $v = A^\top y$, and $\sigma = y^\top Ax \neq 0$.*

Proof. Suppose that $\mathbf{rank}(B) = \mathbf{rank}(A) - 1$. Then there exists some $z \in \mathbb{R}^n$ such that $Bz = 0$ and $Az \neq 0$. Since $Bz = (A - \sigma^{-1}uv^\top)z$, this implies that $Az = \sigma^{-1}uv^\top z$. By setting $x = (\sigma/(v^\top z))z$, we conclude that $u = Ax$. Note that $v^\top z \neq 0$ since it is assumed that $Az = \sigma^{-1}u(v^\top z) \neq 0$.

When $\mathbf{rank}(B) = \mathbf{rank}(A) - 1$, there exists some $w \in \mathbb{R}^m$ such that $B^\top w = 0$ and $A^\top w \neq 0$. Similarly, we set $y = (\sigma/(u^\top w))w$ and conclude that $v = A^\top y$ for some $y \in \mathbb{R}^m$.

In addition,

$$\begin{aligned} y^\top Ax &= \frac{\sigma}{w^\top u} w^\top Ax \\ &= \frac{\sigma}{w^\top Ax} w^\top Ax \\ &= \sigma. \end{aligned}$$

This also shows that for every choice of u and v, σ is uniquely defined.

Conversely, suppose that there are vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ such that $u = Ax$, $v = A^\top y$, and $\sigma = y^\top Ax \neq 0$. Then,

$$B = A - \sigma^{-1}uv^\top = A - \frac{Axy^\top A}{y^\top Ax}.$$

Thus,

$$Bx = \left(A - \frac{Axy^\top A}{y^\top Ax}\right)x = Ax - Ax = 0.$$

Since $Ax \neq 0$ by assumption, we have shown that $x \in \mathbf{null}(B)$ and $x \notin \mathbf{null}(A)$. Now, consider any $z \in \mathbf{null}(A)$. We have that

$$Bz = \left(A - \frac{Axy^\top A}{y^\top Ax}\right)z = Az - \frac{Axy^\top (Az)}{y^\top Ax} = 0$$

Thus, the dimension of $\mathbf{null}(B)$ is strictly larger than the dimension of $\mathbf{null}(A)$, which implies that $\mathbf{rank}(B) < \mathbf{rank}(A)$. Since B only differs A by a rank-one matrix, it must be the case that $\mathbf{rank}(B) = \mathbf{rank}(A) - 1$. □

3.3.1 The Rank-Drop Optimization Problem

For the remainder of this chapter, we consider the case that $\mathbf{rank}(X^{(k)}) > 1$. Our goal is to determine a rank-drop step which reduces the rank of $X^{(k)}$ which also leads to the best objective value. We restrict our attention to the rank-one matrices that can decrease the rank of a matrix. For a matrix $A \in \mathbb{R}^{m\times n}$, let $\mathcal{R}(A)$ denote the set of *rank-drop steps*, which is defined below

$$\begin{aligned}\mathcal{R}(A) \coloneqq &\{\sigma^{-1}uv^\top : \exists x \in \mathbb{R}^n, y \in \mathbb{R}^m \\ &s.t.\ u = Ax, v = A^\top y, \sigma = y^\top Ax \neq 0\}.\end{aligned} \tag{3.7}$$

Since u and v must be in the column and row spans of A respectively, Lemma 3.3.2 shows that the set of rank-drop steps can be expressed in a more concise form.

Lemma 3.3.2. *Let $A \in \mathbb{R}^{m\times n}$ with a thin rank-r SVD, $A = U\Sigma V^\top$. Then we can rewrite the set of rank-drop steps as,*

$$\mathcal{R}(A) = \left\{ \frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t}, s^\top\Sigma^{-1}t > 0 \right\}, \tag{3.8}$$

where $s, t \in \mathbb{R}^r$.

Proof. Assume that $\sigma^{-1}uv^\top \in \mathcal{R}(A)$. Then, following Theorem 3.3.1, $u = Ax$ and $v = A^\top y$ for some x and y, and $u, v \neq 0$. Let $A = U\Sigma V^\top$ be a thin SVD and write $u = Us$ and $v = Vt$ for some $s, t \in \mathbb{R}^r$.

We have,

$$Vt = A^\top y = V\Sigma U^\top y \Rightarrow \Sigma^{-1}t = U^\top y.$$

Consequently,

$$\sigma = y^\top Ax = y^\top(Us) = (U^\top y)^\top s = t^\top\Sigma^{-1}s. \tag{3.9}$$

Finally, we note that $t^\top\Sigma^{-1}s$ can always be made positive by replacing s with $-s$. ☐

Let $Z^{(k)} \in \mathcal{R}(X^{(k)})$. The motivation for rank-drop steps is similar to that of away steps: we wish to find atoms to move away from that also lead to rank decreases. Formally, we consider iterates in the following form,

$$X^{(k+1)} \coloneqq X^{(k)} + \alpha D_{rd}, \qquad \text{where } D_{rd} \coloneqq X^{(k)} - \delta\frac{Z^{(k)}}{\|Z^{(k)}\|_{\mathrm{tr}}}. \tag{3.10}$$

It is important to consider atoms on the boundary, i.e. $\delta Z^{(k)}/\|Z^{(k)}\|_{\text{tr}}$, since this guarantees that there exists some step size, $0 < \alpha \leq 1$ such that the rank of $X^{(k)}$ decreases. Currently, the update rule for (3.10) does not guarantee a convex combination of points in $\mathcal{B}_{\text{tr}}(0, \delta)$. However, by guaranteeing the step size is in the interval $(0, 1]$, we can show that there are easily verifiable conditions for feasibility, preserving the projection free nature of the algorithm.

Recall for away-steps, the feasibility is guaranteed by expressing the the next iterate as a convex combination of a subset of the active atoms. The rank-drop steps will use a similar idea, but instead of maintaining an active set of atoms, we will consider any rank-drop step where $\delta Z^{(k)}/\|Z^{(k)}\|_{\text{tr}}$ is an active atom in some feasible atomic decomposition for $X^{(k)}$.

Let $\mathbf{rank}(X^{(k)}) = r$ and suppose that for a given rank-drop step $Z^{(k)}$, there exists some atomic decomposition of $X^{(k)}$ that uses the atom $\delta Z^{(k)}/\|Z^{(k)}\|_{\text{tr}}$, i.e.,

$$X^{(k)} = \beta_0 \delta \frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}} + \sum_{i=1}^{r-1} \beta_i A_i,$$

where $\beta \in \Delta_{r-1}$. Then we can perform an away-step using this atomic decomposition, where we know there is some value of α such that the weight of the atom $\delta \hat{Z}^{(k)}$ becomes zero. Specifically, we know there is an α such that

$$X^{(k+1)} = (1+\alpha) \sum_{i=1}^{\mathbf{rank}(X^{(k)})-1} \beta_i A_i = (1+\alpha)(X^{(k)} - Z^{(k)}).$$

This decomposition specifically corresponds to when the rank of $X^{(k)}$ decreases after following the direction $(X^{(k)} - \delta Z^{(k)}/\|Z^{(k)}\|_{\text{tr}})$. The first-order Taylor approximation for f after any rank-drop step is,

$$f(X^{(k+1)}) \approx f(X^{(k)}) + (1+\alpha)\langle \nabla f(X^{(k)}), X^{(k)} - Z^{(k)} \rangle.$$

We would like to solve the following optimization problem to find the best rank-drop step,

$$\begin{aligned}
\min_{\substack{Z \in \mathbb{R}^{m \times n} \\ \alpha \in \mathbb{R}}} \quad & (1+\alpha)\langle \nabla f(X^{(k)}), X^{(k)} - Z \rangle \\
s.t. \quad & Z \in \mathcal{R}(X^{(k)}) \\
& \left\| X^{(k)} + \alpha \left((X^{(k)} - \delta \frac{Z}{\|Z\|_{\text{tr}}} \right) \right\|_{\text{tr}} \leq \delta. \\
& \mathbf{rank} \left(X^{(k)} + \alpha \left((X^{(k)} - \delta \frac{Z}{\|Z\|_{\text{tr}}} \right) \right) = \mathbf{rank}(X^{(k)}) - 1.
\end{aligned}$$

This problem is too difficult to solve for a general f, since the trace norm constraint is much more challenging to enforce and verify, and the set $\mathcal{R}(X^{(k)})$ is not convex. In addition, α has a nontrivial dependency on Z since it is the exact step size required to achieve a rank decrease. Thus, we will make a few concessions and solve a more tractable problem.

First, we will simplify the objective by removing the α term.

$$\begin{aligned} \min_{\substack{Z\in\mathbb{R}^{m\times n}\\ \alpha\in\mathbb{R}}} \quad & \langle \nabla f(X^k), X^{(k)} - Z\rangle \\ s.t. \quad & Z \in \mathcal{R}(X^{(k)}) \\ & \left\| X^{(k)} + \alpha\left((X^{(k)} - \delta \frac{Z}{\|Z\|_{\mathrm{tr}}}\right) \right\|_{\mathrm{tr}} \leq \delta. \\ & \mathbf{rank}\left(X^{(k)} + \alpha\left(X^{(k)} - \delta \frac{Z}{\|Z\|_{\mathrm{tr}}}\right)\right) = \mathbf{rank}(X^{(k)}) - 1. \end{aligned} \tag{3.11}$$

Note that by considering the matrix $X^{(k)} - Z$ in the objective of (3.11), we are explicitly only considering matrices after the full rank-drop step is taken. Thus, the objective can be interpreted as finding the best $r-1$ atoms across all atomic decompositions with the highest descent potential. The rank-drop step Z can be viewed as the least important subspace of $X^{(k)}$ with respect to the objective, since reassigning its weight to the remaining $r-1$ atoms gives the highest potential for descent.

The distinction that we consider *all* possible atomic decompositions is important since the away-steps only consider specific atomic decompositions either given from the LMO or from the SVD of the current iterate. Moreover, the objective we consider explicitly considers the alignment with the gradient after the rank-drop rather than before, which we contend is more important if the suggestions of [33] are taken to exclusively choose away steps that decrease the rank of the current solution as long as the objective is not worsened. This also allows for finding rank-drop steps inside the trace norm ball whereas the in-face steps can only decrease the rank when the solutions is on the boundary of the trace norm ball (or sufficiently close numerically).

Handling the constraints of (3.11) is still a large challenge and testing for feasibility at each iteration is computationally expensive. We want to establish a verifiable sufficient condition for rank-drop steps that ensure feasibility.

A key quantity we will use to ensure feasibility of the iterate after a rank-drop step is the distance from the current iterate to the boundary of the trace norm ball. We consider two distinct cases. Let $\kappa(X^{(k)})$ be half of the distance between $X^{(k)}$ and the boundary of the trace norm ball,

$$\kappa(X^{(k)}) := \frac{\delta - \|X^{(k)}\|_{\mathrm{tr}}}{2} \tag{3.12}$$

We will assume that the iterate $X^{(k)}$ is rank r, and $\kappa(X^{(k)}) \geq \sigma_r(X^{(k)})$, where $\kappa(X^{(k)})$ is defined in (3.12). Recall that an issue with guaranteeing feasibility of these rank-drop steps is that we cannot guarantee that there exists an atomic decomposition of $X^{(k)}$ using only feasible atoms (points in $\mathcal{B}_{\mathrm{tr}}(0,\delta)$, in which the atom $\delta\hat{Z}^{(k)}$ is active. We will show that if the rank-drop step is sufficiently small, it is possible to guarantee that such an atomic decomposition exists. Thus, we can guarantee feasibility of the next iterate with an easy to enforce constraint on the rank-drop step.

Theorem 3.3.3. *Let $X^{(k)} \in \mathcal{B}_{tr}(0,\delta)$, $Z^{(k)} \in \mathcal{R}(A)$, and $\hat{Z}^{(k)} = Z^{(k)}/\|Z^{(k)}\|_{tr}$. If $\|Z^{(k)}\|_F \leq \kappa(X^{(k)})$, then*

$$X^{(k+1)} = X^{(k)} + \alpha(X^{(k)} - \delta\hat{Z}^{(k)}) \in \mathcal{B}_{tr}(0,\delta), \ \forall \alpha \in [0, \alpha_{\max}]$$

where

$$\alpha_{\max} := \frac{\|Z^{(k)}\|_F}{\delta - \|Z^{(k)}\|_F}.$$

Moreover, if $\alpha = \alpha_{\max}$, then $\mathbf{rank}(X^{(k+1)}) = \mathbf{rank}(X^{(k)}) - 1$.

This theorem states that if we find some sufficiently small rank-drop step, specifically some rank drop step $Z^{(k)} \in \mathcal{R}(X^{(k)}) \cap \mathcal{B}_F(0, \kappa(X^{(k)})$, then decreasing the rank using $Z^{(k)}$ will lead to a feasible iterate. The idea for the proof is simply to verify that under this condition, a valid atomic decomposition exists, which includes the atom $\delta Z^{(k)}/\|Z^{(k)}\|_{\mathrm{tr}}$.

Proof. Define $R^{(k)} = X^{(k)} - Z^{(k)}$. Note that,

$$\begin{aligned} \|R^{(k)}\|_{\mathrm{tr}} + \|Z^{(k)}\|_{\mathrm{tr}} &= \|X^{(k)} - Z^{(k)}\|_{\mathrm{tr}} + \|Z^{(k)}\|_{\mathrm{tr}} \\ &\leq \|X^{(k)}\|_{\mathrm{tr}} + 2\|Z^{(k)}\|_{\mathrm{tr}} \\ &\leq \delta \end{aligned}$$

where the last inequality uses the fact that $Z^{(k)}$ is rank one and $\|Z^{(k)}\|_{\mathrm{tr}} = \|Z^{(k)}\|_F \leq (\delta - \|X^{(k)}\|_{\mathrm{tr}})/2$.

Let $R^{(k)}$ have the thin SVD, $R^{(k)} = \sum_{i=1}^{r-1} \sigma_i u_i v_i^\top$. We can express $X^{(k)}$ as,

$$X^{(k)} = \beta_0 \delta \frac{Z^{(k)}}{\|Z^{(k)}\|_{\mathrm{tr}}} + \sum_{i=1}^{r-1} \beta_i \delta u_i v_i^\top \tag{3.13}$$

where $\beta_0 = \|Z^{(k)}\|_{\text{tr}}/\delta$ and $\beta_i = \sigma_i/\delta$ for $i \in \{1, ..., r-1\}$. Note that $\beta_i \geq 0$ for all $i \in \{0, ..., r-1\}$, and

$$\begin{aligned}
\|R^{(k)}\|_{\text{tr}} + \|Z^{(k)}\|_{\text{tr}} &\leq \delta \\
\Rightarrow \delta \left(\sum_{i=1}^{r-1} \beta_i \right) + \delta\beta_0 &\leq \delta \\
\Rightarrow \sum_{i=0}^{r-1} \beta_i &\leq 1
\end{aligned}$$

Thus, there exists an atomic decomposition for $X^{(k)}$ that only uses atoms from $\mathcal{B}_{\text{tr}}(0, \delta)$ where the atom, $\delta Z^{(k)}/\|Z^{(k)}\|_{\text{tr}}$, is active. Thus, the remaining arguments for showing feasibility will be identical to the feasibility for away steps given a feasible atomic decomposition.

To show that $X^{(k+1)} \in \mathcal{B}_{\text{tr}}(0, \delta)$. Rewrite $X^{(k+1)}$ as,

$$\begin{aligned}
X^{(k+1)} &= (1+\alpha)X^{(k)} - \alpha\delta \frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}} \\
&= (1+\alpha)R^{(k)} + (1+\alpha)\beta_0\delta\hat{Z}^{(k)} - \alpha\delta\frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}} \quad \text{(using (3.13))} \\
&= (1+\alpha)R^{(k)} + (\beta_0 + \beta_0\alpha - \alpha)\delta\frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}} \\
&= \sum_{i=1}^{r-1}(1+\alpha)\beta_i\delta u_i v_i^\top + (\beta_0 + \beta_0\alpha - \alpha)\delta\frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}} \\
&= \sum_{i=1}^{r-1}\gamma_i\delta u_i v_i^\top + \gamma_0\delta\frac{Z^{(k)}}{\|Z^{(k)}\|_{\text{tr}}}
\end{aligned}$$

where $\gamma_0 = (\beta_0 + \beta_0\alpha - \alpha)$ and $\gamma_i = (1+\alpha)\beta_i, \forall i \in \{1, ..., r-1\}$. If we can show that all $\gamma_i \geq 0$ and $\sum_{i=0}^{r-1}\gamma_i \leq 1$, then we have shown that $X^{(k+1)}$ can be expressed as a convex combination of points from $\mathcal{B}_{\text{tr}}(0, \delta)$, and is feasible.

Recall that,

$$\alpha_{\max} = \frac{\|Z^{(k)}\|_F}{\delta - \|Z^{(k)}\|_F} \text{ and } \beta_0 = \frac{\|Z^{(k)}\|_F}{\delta}.$$

If $\alpha \in [0, \alpha_{\max}]$, then it is easy to verify that $\beta_0 + \beta_0\alpha - \alpha \geq 0$, with equality at $\alpha = \alpha_{\max}$.

Thus, $\gamma_i \geq 0, \forall i \in \{0, 1, ..., r-1\}$. Next, we have that,

$$\begin{aligned}\sum_{i=0}^{r-1} \gamma_i &= (1+\alpha)\sum_{i=1}^{r-1} \beta_i + (1+\alpha)\beta_0 - \alpha \\ &= (1+\alpha)\sum_{i=0}^{r-1} \beta_i - \alpha \\ &\leq (1+\alpha) - \alpha \\ &= 1.\end{aligned}$$

Thus, $\gamma_i \geq 0,\ i = 0, \ldots, r-1$, and $\sum_{i=0}^{r-1} \gamma_i \leq 1$. This implies that $X^{(k+1)} \in \mathcal{B}_{\text{tr}}(0, \delta)$ if $\alpha \in [0, \alpha_{\max}]$.

Finally, when $\alpha = \alpha_{\max}$, we have that $(\beta_0 + \beta_0\alpha_{\max} - \alpha_{\max}) = 0$. Hence, $X^{(k+1)}$ is a sum of $r-1$ rank one matrices and $\mathbf{rank}(X^{(k+1)}) \leq r-1$. Since $X^{(k+1)}$ is a rank one perturbation of $X^{(k)}$, the most the rank can decrease by is 1 and $\mathbf{rank}(X^{(k+1)}) = r-1$. $\square$

Theorem 3.3.3 motivates the following formulation to find a rank-drop step,

$$\begin{aligned} \min_{Z} \quad & \langle -Z, \nabla f(X^{(k)}) \rangle \\ s.t. \quad & Z \in \mathcal{B}_F(0, \kappa(X^{(k)})) \cap \mathcal{R}(X^{(k)}). \end{aligned} \tag{3.14}$$

While the additional restrictions make this problem easier to solve, it is unclear if this feasible region is nonempty. In Lemma 3.3.4, we first establish a lower bound on the trace norm of the rank-drop step.

Lemma 3.3.4. *Let $A \in \mathbb{R}^{m \times n}$ have rank r and let $Z \in \mathcal{R}(A)$ be an arbitrary rank-drop step. Then $\|Z\|_{tr} \geq \sigma_r(A)$.*

Proof. From Lemma 3.3.2, $Z = \frac{Ust^\top V^\top}{s^\top \Sigma^{-1} t}$ for some $s, t \in \mathbb{R}^r$, with $\sigma = s^\top \Sigma^{-1} t > 0$.

We have,

$$\begin{aligned}
\|Z\|_{\mathrm{tr}} &= \sigma^{-1}\|Us\|_2\|Vt\|_2 \\
&= \frac{\|Us\|_2\|Vt\|_2}{|s^\top\Sigma^{-1}t|} \\
&\geq \frac{\|s\|_2\|t\|_2}{\|s\|_2\|\Sigma^{-1}t\|_2} \\
&\geq \frac{1}{\max_{z:\|z\|_2=1}\|\Sigma^{-1}z\|_2} \\
&= \sigma_r(A)
\end{aligned}$$

Thus, $\|Z\|_{\mathrm{tr}} \geq \sigma_r(A)$. □

Now, we can characterize conditions under which the feasible region of (3.14) is nonempty.

Theorem 3.3.5. *If* $\mathbf{rank}(X^{(k)}) \geq 1$ *with* $\sigma_r(X^{(k)}) \leq \kappa(X^{(k)})$*, then the feasible region for* (3.14) *is non-empty.*

Proof. Assume that $X^{(k)}$ has the thin SVD $X^{(k)} = U\Sigma V^\top$. We show that the singular vector pair corresponding to the smallest singular value can be made into a rank-drop step. Let $s = e_r$, where e_r is the r^{th} elementary basis vector, $t = (\sigma_r(X^{(k)})/\kappa(X^{(k)}))e_r$, and $Z = Ust^\top V^\top / s^\top\Sigma^{-1}t$. Since $\sigma_r(X^{(k)}) \leq \kappa(X^{(k)})$, $\|t\|_2 \leq 1$. We conclude that,

$$\begin{aligned}
\|Z\|_F &= \left\|\frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t}\right\|_F \\
&\leq \|Us\|_2\|Vt\|_2(\|\Sigma^{-1}s\|_2\|t\|_2)^{-1} \qquad (3.15)\\
&\leq \sigma_r(X^{(k)}) \\
&\leq \kappa(X^{(k)})
\end{aligned}$$

Thus $Z \in \mathcal{B}_F(0, \kappa(X^{(k)})) \cap \mathcal{R}(X^{(k)})$, and the feasible region to (3.14) is non-empty. More generally, we can see that any singular vector pair with singular value $\sigma_i(X^{(k)}) \leq \kappa(X^{(k)})$ will be feasible to (3.14) as well. □

Subsequently, we refer to $\kappa(X^{(k)}) \geq \sigma_r(X^{(k)})$, as the *interior case*, and $\kappa(X^{(k)}) < \sigma_r(X^{(k)})$ as the *exterior case.*

3.3.2 The Interior Rank-Drop Problem

Assume that a thin SVD for $X^{(k)}$, $X^{(k)} = U\Sigma V^\top$, is given. Using Lemma 3.3.2, the constraint in (3.14) can be made explicit,

$$\begin{aligned} \min_{s,t\in\mathbb{R}^r} & \left\langle X^{(k)} - \frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t}, \nabla f(X^{(k)})\right\rangle \\ s.t.\ & \frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t} \in \mathcal{B}_{\text{tr}}(0, \kappa(X^{(k)})) \\ & s^\top\Sigma^{-1}t > 0. \end{aligned} \tag{3.16}$$

To make (3.16) more amenable to computation, we remove the fraction using the normalization constraint $s^\top\Sigma^{-1}t = \kappa(X^{(k)})^{-1}$ and formulate (3.16) equivalently as follows,

$$\begin{aligned} \min_{s,t\in\mathbb{R}^r} \ & q(s,t) := \langle \nabla f(X^{(k)}), -\kappa(X^{(k)})Ust^\top V^\top\rangle \\ s.t.\ & s^\top\Sigma^{-1}t = \kappa(X^{(k)})^{-1} \\ & \|s\|_2 = 1,\ \|t\|_2 \le 1 \end{aligned} \tag{3.17}$$

Note that the constraints in (3.17) also ensure that s and t cannot be rescaled to obtain a different solution yielding an identical rank-drop step. The equivalence of (3.14) and (3.17) is formally established in Theorem 3.3.6.

Theorem 3.3.6. *If $X^{(k)} \in \mathbb{R}^{m\times n}$ with $\|X^{(k)}\|_{tr} \le \delta$ and $\kappa(X^{(k)}) \ge \sigma_r(X^{(k)})$, then an optimal solution to (3.17) is an optimal solution to (3.16). Moreover, an optimal to (3.16) can always be rescaled into an optimal solution to (3.17).*

Proof. First we will show that for any feasible solution (s,t) to (3.16), there exists a corresponding feasible solution to (3.17) with the same objective value.

Let (s,t) be a feasible solution to (3.16). Define $\hat{s} := \frac{s}{\|s\|_2}$ and $\hat{t} := \frac{\kappa(X^{(k)})^{-1}}{\hat{s}^\top\Sigma^{-1}t}t$. We will show $(\hat{s},\hat{t})$ is feasible for (3.17) and the objective values of (3.16) and (3.17) are equal at (s,t) and $(\hat{s},\hat{t})$. To see this, note that,

$$\begin{aligned} \frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t} &= \frac{\frac{\kappa(X^{(k)})^{-1}}{\|s\|_2\hat{s}^\top\Sigma_k^{-1}t}}{\frac{\kappa(X^{(k)})^{-1}}{\|s\|_2\hat{s}^\top\Sigma_k^{-1}t}} \cdot \frac{Ust^\top V^\top}{s^\top\Sigma^{-1}t} \\ &= \frac{U\hat{s}\hat{t}^\top V^\top}{\hat{s}^\top\Sigma^{-1}\hat{t}} \end{aligned} \tag{3.18}$$

But,

$$\hat{s}^\top \Sigma^{-1} \hat{t} = \frac{\kappa(X^{(k)})^{-1}}{\hat{s}^\top \Sigma^{-1} t} \hat{s}^\top \Sigma^{-1} t = \kappa(X^{(k)})^{-1}, \tag{3.19}$$

satisfying the first constraint of (3.17). We can also conclude,

$$\frac{U s t^\top V^\top}{s^\top \Sigma^{-1} t} = \kappa(X^{(k)}) U \hat{s} \hat{t}^\top V^\top \tag{3.20}$$

showing the objective values are equal (up to constants that do not depend on s or t).

For the norm constraint, $\|\hat{s}\|_2 = 1$ by construction. Next, to see that the solution satisfies the last constraint, the fact that (s, t) is feasible gives us,

$$\begin{aligned} \left\| \frac{U s t^\top V^\top}{s^\top \Sigma^{-1} t} \right\|_{\mathrm{tr}} &\leq \kappa(X^{(k)}) \\ \Rightarrow \kappa(X^{(k)}) \|U \hat{s} \hat{t}^\top V^\top\|_{\mathrm{tr}} &\leq \kappa(X^{(k)}) \\ \Rightarrow \|U \hat{s} \hat{t}^\top V^\top\|_{\mathrm{tr}} &\leq 1 \end{aligned} \tag{3.21}$$

where the first implication uses the result from (3.20). Next, using the fact that the trace norm of a rank-one matrix is equivalent to its Frobenius norm, we have,

$$\begin{aligned} \|U \hat{s} \hat{t}^\top V^\top\|_{\mathrm{tr}} &= \|U \hat{s} \hat{t}^\top V^\top\|_F \\ &= \sqrt{\mathbf{tr}((U \hat{s} \hat{t}^\top V^\top)^\top (U \hat{s} \hat{t}^\top V^\top))} \\ &= \sqrt{\mathbf{tr}(V \hat{t} \hat{s}^\top U^\top U \hat{s} \hat{t} V^\top)} \\ &= \sqrt{\mathbf{tr}(V \hat{t} \hat{t}^\top V^\top)} \\ &= \sqrt{\mathbf{tr}(\hat{t}^\top V^\top V \hat{t})} \\ &= \|\hat{t}\|_2 \end{aligned} \tag{3.22}$$

Thus, the results from (3.21) and (3.22) jointly imply that $\|\hat{t}\|_2 \leq 1$. It is readily seen that any feasible solution to (3.17) is a feasible solution to (3.16). Since there exists a mapping from feasible points in (3.16) to (3.17) and vice versa preserving objective values, the optimal values must be equal. Thus, an optimal solution to (3.17) is an optimal solution to (3.16) and the converse result holds as stated. $\square$

Solving the Optimization Problem in the Interior Case

Now we discuss how to solve the Rank-Drop optimization problem (3.17) in the interior case, i.e., when $\kappa(X^{(k)}) \geq \sigma_r(X^{(k)})$, which is illustrated in Figure 3.2. Let $W \coloneqq U^\top \nabla f(X^{(k)}) V$.

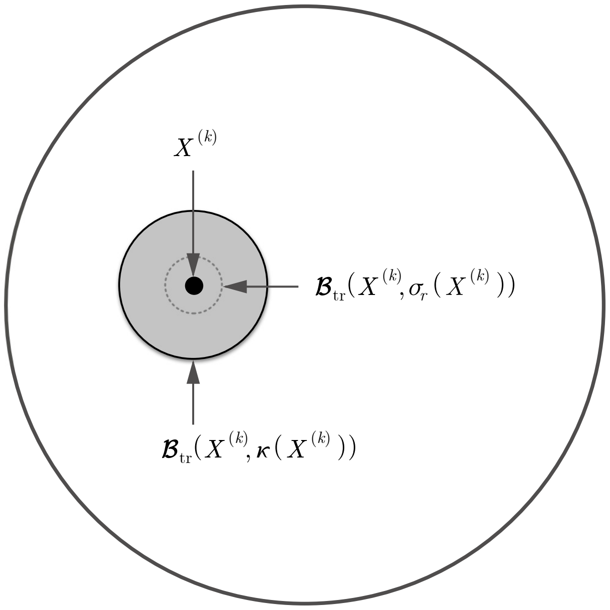

$X^{(k)}$
$\mathcal{B}_{\mathrm{tr}}(X^{(k)}, \sigma_r(X^{(k)}))$
$\mathcal{B}_{\mathrm{tr}}(X^{(k)}, \kappa(X^{(k)}))$

Figure 3.1: A diagram of the interior case. Any rank-drop step found in the $\kappa(X^{(k)})$ ball around $X^{(k)}$ can move the solution at most to the boundary of the trace norm ball and is guaranteed to be feasible.

The Lagrangian for (3.17) is,

$$\begin{aligned}\mathcal{L}(s,t,\lambda,\rho,\nu) \coloneqq & s^\top W t + \lambda(s^\top \Sigma^{-1} t - \kappa(X^{(k)})^{-1}) \\ & + \rho(s^\top s - 1) + \nu(t^\top t - 1).\end{aligned}$$

Now suppose that (s,t,λ,ρ,ν) satisfies the KKT conditions. Then the stationarity conditions give

$$\begin{aligned}\nabla_t \mathcal{L} = 0 &\Leftrightarrow W^\top s + \lambda \Sigma^{-1} s = -2\nu t \\ \nabla_s \mathcal{L} = 0 &\Leftrightarrow W t + \lambda \Sigma^{-1} t = -2\rho s.\end{aligned} \tag{3.23}$$

Let $M_\lambda = -\frac{1}{2}(W + \lambda\Sigma^{-1})$. Then we can rewrite (3.23) as,

$$\begin{aligned} M_\lambda^\top s &= \nu t \\ M_\lambda t &= \rho s. \end{aligned} \tag{3.24}$$

Following these equations, we conclude that $\rho = s^\top M_\lambda t = \nu\|t\|_2^2$ for any feasible solution.

Suppose that $\|t\|_2 < 1$. Then by complementary slackness, we must have that $\nu = 0 \Rightarrow \rho = 0$. Equation (3.24) implies that M_λ is rank deficient and there exists $\lambda \in \mathbb{R}$ and a vector $x \in \mathbb{R}^r$ with $x \neq 0$, such that,

$$(W + \lambda\Sigma^{-1})x = 0 \Leftrightarrow -\Sigma W x = \lambda x.$$

Hence λ is an eigenvalue of $-\Sigma W$.

Moreover, (3.24) also implies that $s \in \mathbf{null}(M_\lambda^\top)$ and $t \in \mathbf{null}(M_\lambda)$. For simplicity, we assume that the eigenvalues of $-\Sigma W$ are distinct. In this case, (3.23) is satisfied only when (s, t) are scalar multiples of a singular vector pair of M_λ with the associated singular value 0.

Let $(\hat{s}, \hat{t})$ be the smallest singular vector pair of M_λ for some real $\lambda \in \mathbf{eig}(-\Sigma W)$. Then if we let $s = \hat{s}$ and $t = \hat{t}/(\kappa(X^{(k)})\hat{s}^\top\Sigma^{-1}\hat{t})$, then we can show that $(s, t, \lambda, 0, 0)$ satisfy the KKT conditions as long as $\hat{s}^\top\Sigma^{-1}\hat{t} < \kappa(X^{(k)})^{-1}$. Note that (s, t, λ) still satisfy (3.24) since we have only multiplied t by a scalar multiple. Moreover, if $\hat{s}^\top\Sigma^{-1}\hat{t} < \kappa(X^{(k)})^{-1}$, then $\|t\|_2 < 1$ and,

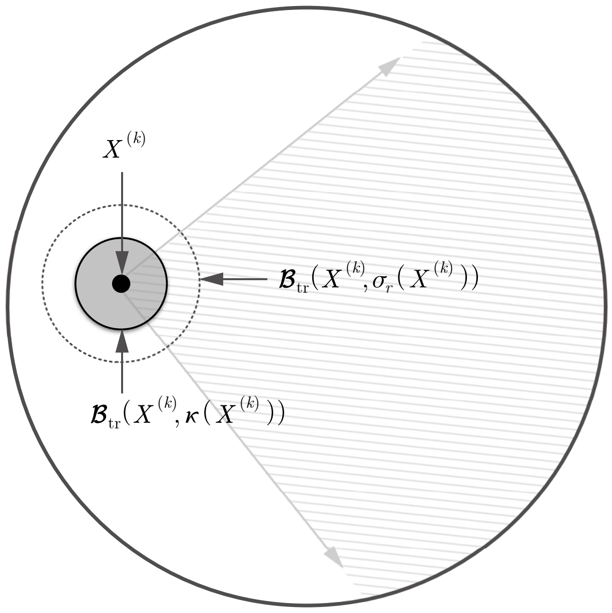
$$s^\top\Sigma^{-1}t = \frac{\hat{s}^\top\Sigma^{-1}\hat{t}}{\kappa(X^{(k)})\hat{s}^\top\Sigma^{-1}\hat{t}} = \kappa(X^{(k)})^{-1}. \tag{3.25}$$

Thus, $(s, t, \lambda, 0, 0)$ are primal and dual feasible and satisfy the KKT conditions.

Since problem (3.17) is nonconvex and generally difficult to solve, we only consider candidate KKT points characterized in this fashion, specifically with $\|t\|_2 < 1$. We note that since $-\Sigma W$ is not symmetric, the eigenvalues are not always real-valued, and we only consider candidates generated from the real eigenvalues. In the event that no feasible candidate is found, we will show that the rank-drop solution for the exterior case always yields a feasible rank-drop step and can then be used to generate candidate solutions instead.

3.3.3 The Exterior Rank-Drop Problem

When $\kappa(X^{(k)}) < \sigma_r(X^{(k)})$, we can no longer find rank-drop steps in the ball $\mathcal{B}_F(0, \kappa(X^{(k)}))$, see Figure 3.2. In this case, Lemma 3.3.7 shows that $X^{(k)}$ is either close to the boundary or has low-rank.



$X^{(k)}$
$\mathcal{B}_{\text{tr}}(X^{(k)}, \sigma_r(X^{(k)}))$
$\mathcal{B}_{\text{tr}}(X^{(k)}, \kappa(X^{(k)}))$

Figure 3.2: A diagram of the exterior case. The directions considered are restricted to the cone of directions that are nonascent directions for the trace norm.

Lemma 3.3.7. *Let* $r = \textbf{rank}(X^{(k)})$*. If* $\kappa(X^{(k)}) < \sigma_r(X^{(k)})$*, then* $\|X^{(k)}\|_{tr} > \frac{r}{r+2}\delta$*.*

Proof. Note that $\sigma_r(X^{(k)}) \leq \|X^{(k)}\|_{\text{tr}}/r$. Then, the inequality can be rearranged as follows.

$$\begin{aligned} \kappa(X^{(k)}) &< \sigma_r(X^{(k)}) \\ \delta - \|X^{(k)}\|_{\text{tr}} &< 2\frac{\|X^{(k)}\|_{\text{tr}}}{r} \\ \|X^{(k)}\|_{\text{tr}} &> \frac{r}{r+2}\delta \end{aligned}$$

□

If we are in the exterior case and $X^{(k)}$ is not close to the boundary, i.e. $\kappa(X^{(k)})$ is large, then from Lemma 3.3.7, the rank of $X^{(k)}$ should be small. Moreover, since $\kappa(X^{(k)}) < \sigma_r(X^{(k)})$, it implies that $\sigma_r(X^{(k)})$ must also be large. Thus, if we are in the exterior case far from the boundary, we do not expect to find a good rank-drop step since the current solution is low-rank, where each rank-one matrix provides a significant contribution to the current solution.

Hence, we will only consider the case when the current iterate is close to the boundary. To ensure feasibility, we will restrict our attention to directions that move into the trace norm ball. That is, directions that do not increase the trace norm of the current iterate. We establish the following theorems to facilitate formulating appropriate optimization problems for this case.

Theorem 3.3.8. *Let $X^{(k)} \in \mathcal{B}_{tr}(0,\delta)$ have the thin SVD $X^{(k)} = U\Sigma V^\top$. Define $D^{(k)} = X^{(k)} - \delta \hat{Z}^{(k)}$ with $Z^{(k)} \in \mathcal{R}(X^{(k)})$ and $\hat{Z}^{(k)} = Z^{(k)}/\|Z^{(k)}\|_{tr} = Ust^\top V^\top$, for some s and t with $\|s\| = \|t\| = 1$. Then,*

$$\left(\max_{G \in \partial \|X^{(k)}\|_{tr}} \langle D_k, G \rangle \right) \leq 0, \tag{3.26}$$

if and only if $\delta s^\top t \geq \|X^{(k)}\|_{tr}$.

Proof. From [77], the subdifferential of the trace norm is,

$$\partial \|X^{(k)}\|_{\mathrm{tr}} := \{UV^\top + H : U^\top H = HV = 0, \|H\|_{sp} \leq 1\}.$$

Let $G \in \partial \|X^{(k)}\|_{\mathrm{tr}}$ be an arbitrary subgradient. Then,

$$\begin{aligned} & \langle G, X^{(k)} - \delta U s t^\top V^\top \rangle \\ &= \langle UV^\top + H, U\Sigma V^\top - \delta U s t^\top V^\top \rangle \\ &= \mathbf{tr}((UV^\top + H)^\top (U\Sigma V^\top - \delta U s t^\top V^\top) \\ &= \mathbf{tr}(VU^\top U\Sigma V^\top + H^\top U \Sigma V^\top) - \\ & \quad \delta\, \mathbf{tr}(VU^\top U s t^\top V^\top + H^\top U s t^\top V^\top) \\ &= \mathbf{tr}(\Sigma) - \delta s^\top t \\ &= \|X^{(k)}\|_{\mathrm{tr}} - \delta s^\top t \end{aligned}$$

This implies that $\max_{G \in \partial \|X\|_{\mathrm{tr}}} \langle D^{(k)}, G \rangle \leq 0$ if and only if $\delta s^\top t \geq \|X^{(k)}\|_{\mathrm{tr}}$. Since $D^{(k)}$ has a nonpositive inner product with all elements in the subdifferential, it must be a nonascent direction for the trace norm at $X^{(k)}$. □

Following Theorem 3.3.8, $D^{(k)}$ will be a nonascent direction for the trace norm at $X^{(k)}$ if and only if $\delta s^\top t \geq \|X^{(k)}\|_{\mathrm{tr}}$. When $X^{(k)}$ is on the boundary of the trace norm ball, Corollary 3.3.8.1 below states a simpler and useful characterization.

Corollary 3.3.8.1. *Let $X^{(k)}$ have the thin SVD $X^{(k)} = U\Sigma V^\top$ with $\|X^{(k)}\|_{tr} = \delta$. Define $D_k = X^{(k)} - \delta \hat{Z}_k$ with $Z^{(k)} \in \mathcal{R}(X^{(k)})$ and $\hat{Z}_k = Z^{(k)}/\|Z^{(k)}\|_{tr} = Ust^\top V^\top$, for some s and t with $\|s\|_2 = \|t\|_2 = 1$. Then,*

$$\left(\max_{G \in \partial \|X^{(k)}\|_{tr}} \langle D_k, G \rangle \right) \leq 0 \tag{3.27}$$

if and only if $s = t$.

Proof. From Theorem 3.3.8, we must have that $\delta s^\top t \geq \|X^{(k)}\|_{\text{tr}} = \delta$. This implies that $s^\top t \geq 1$. Since $\|s\|_2 = \|t\|_2 = 1$, we have that $s^\top t \leq 1$, where equality is attained only when $s = t$. Thus, $\delta s^\top t \geq \|X^{(k)}\|_{\text{tr}}$ if and only if $s = t$, completing the proof. □

In the exterior case, since we are mostly interested in the situation when $X^{(k)}$ is close to the boundary of the trace norm ball. Assume that $\|X^{(k)}\|_{\text{tr}} \approx \delta$ and use Corollary 3.3.8.1 to symmetrize the problem, where we know the rank-drop step takes the form $Z = Uss^\top V^\top$ for some unit vector s. The optimization problem for the exterior case will then be (3.11) except we simplify the feasible region to directions that do not increase the trace norm.

$$\begin{aligned} \min_{s \in \mathbb{R}^r} \quad & - \langle \nabla f(X^{(k)}), \frac{Uss^\top V^\top}{s^\top \Sigma^{-1} s} \rangle \\ s.t. \quad & \|s\|_2 = 1 \end{aligned} \tag{3.28}$$

Let $W = U^\top \nabla f(X^{(k)}) V$. Note that $\frac{1}{2} s^\top (W^\top + W) s = \langle \nabla f(X^{(k)}), Uss^\top V^\top \rangle$. Hence, in the exterior case, we solve the following optimization problem,

$$\begin{aligned} \max_{s \in \mathbb{R}^r} \quad & \frac{1}{2} \frac{s^\top (W^\top + W) s}{s^\top \Sigma^{-1} s} \\ s.t. \quad & \|s\|_2 = 1. \end{aligned} \tag{3.29}$$

Here, (3.29) is a generalized eigenvalue problem [21], which solves for,

$$\max \lambda \text{ s.t. } (W + W^\top) s = \lambda \Sigma^{-1} s,$$

which can be further reduced to a standard eigenvalue problem since Σ^{-1} is a nonsingular diagonal matrix.

We will now establish that, using (3.29), the required step size to decrease the rank of the current iterate also guarantees that the next iterate is feasible, and the step size calculation is straightforward. Before we prove Theorem 3.3.10, we will require the following Theorem from [20].

Theorem 3.3.9 ([20])**.** *Suppose that D is symmetric positive semidefinite, S is symmetric, and $\mathbf{rank}(D - S) = \mathbf{rank}(D) - \mathbf{rank}(S)$. Then $D - S$ is positive semidefinite.*

Theorem 3.3.10. *Let s be an optimal solution to (3.29) and let $D^{(k)} = X^{(k)} - \delta U s s^\top V^\top$, where $X^{(k)} = U\Sigma V^\top$ is a thin SVD with $\|X^{(k)}\|_{tr} \leq \delta$. If $\alpha_{\max} = (\delta s^\top \Sigma^{-1} s - 1)^{-1}$ and $X_{k+1} = X^{(k)} + \alpha_{\max} D_k$, then $rank(X_{k+1}) = rank(X^{(k)}) - 1$ and $\|X_{k+1}\|_{tr} \leq \delta$.*

Proof. From the definition of X_{k+1},

$$\begin{aligned} X_{k+1} &= (1 + \alpha_{\max}) U\Sigma V^\top - \alpha_{\max}\delta U s s^\top V^\top \\ &= U((1 + \alpha_{\max})\Sigma - \alpha_{\max}\delta s s^\top)V^\top. \end{aligned}$$

Define

$$M := (1 + \alpha_{\max})\Sigma - \alpha_{\max}\delta s s^\top$$

From Theorem 3.3.1, it is straightforward to verify that $\alpha_{\max}\delta s s^\top \in \mathcal{R}((1 + \alpha_{\max})\Sigma)$ and $\mathbf{rank}(M) = \mathbf{rank}(X^{(k)}) - 1$.

Let $\lambda_i(M)$ be the i^{th} eigenvalue of M, then,

$$\begin{aligned} \sum_i^r \lambda_i(M) &= \mathbf{tr}(M) \\ &= (1 + \alpha_{\max})\,\mathbf{tr}(\Sigma) - \alpha_{\max}\,\mathbf{tr}(\delta s s^\top) \\ &= (1 + \alpha_{\max})\delta - \delta\alpha_{\max}\,\mathbf{tr}(s^\top s) \\ &\leq \delta \end{aligned}$$

where the second last line uses the fact that $\text{tr}(\Sigma) = \|X\|_{\text{tr}} \leq \delta$ and the cyclic property of the trace.

From Theorem 3.3.9, M is symmetric positive semidefinite. From $\sum_i \lambda_i(M) = \sum_i \sigma_i(M)$, we have $\|M\|_{\text{tr}} \leq \delta$. Since $X_{k+1} = UMV^\top$, it follows that $\|X_{k+1}\|_{\text{tr}} \leq \delta$. □

We remark that the proof of Theorem 3.3.10 only requires that $\|X^{(k)}\|_{\text{tr}} \leq \delta$ and does not require strict equality as the assumption for the exterior case. For the interior case, if a valid candidate is not found by exploring all KKT points with $\|t\|_2 < 1$, then we generate candidate solutions instead by solving (3.29). Note that even though it is always possible to determine rank-drop steps in this fashion, the additional constraint that $s = t$ greatly reduces the search space and it is preferable to solve the interior case properly if possible.

3.3.4 Summary of Algorithm

We will give a brief summary of the *Rank-Drop Frank-Wolfe* (RDFW) algorithm. At each iteration, the algorithm generates a candidate rank-drop step, $Z^{(k)} = U s_k t_k^\top V^\top$, depending on whether the iterate is in the interior case or the exterior case. To determine whether to accept the rank-drop candidate step, we use the same criterion proposed in [33], i.e., we accept any rank-drop step that does not increase the objective.

The criteria in [40] for away-steps require solving the regular Frank-Wolfe linear subproblem at each iteration to determine whether an away-step should be accepted. This can lead to many unnecessary SVD calculations since the Frank-Wolfe step is computed even when the step is not taken. The idea used in [33] addresses this issue by constructing a sufficient decrease condition that is independent of the Frank-Wolfe step, in particular, for the InFace$(0, \infty)$ variant, any in-face step that moves to the boundary and does not worsen the objective is accepted. This allows the algorithm to maintain a lower rank SVD as well as allows the algorithm to skip computing a Frank-Wolfe step unnecessarily.

Similarly for RDFW, if it is possible to decrease the rank of the current iterate without worsening the objective, i.e. if $f(X^{(k)} + \alpha^{(k)}(X^{(k)} - \delta Z^{(k)}) \leq f(X^{(k)})$, the candidate rank-drop step is accepted and iterate is updated as $X^{(k+1)} \leftarrow X^{(k)} + \alpha^{(k)}(X^{(k)} - \delta Z^{(k)})$. If the candidate step is rejected, i.e. $f(X^{(k)} + \alpha^{(k)}(X^{(k)} - \delta Z^{(k)}) > f(X^{(k)})$, then a regular Frank-Wolfe step is performed instead, where $Z^{(k)} \leftarrow \arg\min_{Z \in \mathcal{B}_{\mathrm{tr}}(0,\delta)} \langle Z, \nabla f(X^{(k)}) \rangle$ and $X^{(k)} \leftarrow X^{(k)} + \alpha^{(k)}(Z^{(k)} - X^{(k)})$.

The full RDFW algorithm is presented in Algorithm 5, where an overview of solving the interior and exterior cases is presented in Algorithm 4.

Updating the Thin SVD

The RDFW algorithm requires access to a thin SVD representation of the current iterate $X^{(k)}$. To avoid computing the factorization at each iteration, we recognize that each iteration only updates the solution by a rank-one matrix. We can then use the ideas in [10] to efficiently update the SVD. We will summarize the procedure as follows.

Let $A \in \mathbb{R}^{m \times n}$ be a rank r matrix with thin SVD $A = U \Sigma V^\top$. We wish to find the thin SVD of the matrix $B = A + uv^\top$ for any $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$. Note that B can be written as follows,

$$B = \begin{pmatrix} U & y \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} V & v \end{pmatrix}^\top .$$

Let $Q_U R_U$ be a QR decomposition of $\begin{pmatrix} U & y \end{pmatrix}$, and $Q_V R_V$ be a QR decomposition of $\begin{pmatrix} V & v \end{pmatrix}$. Then,

$$B = Q_U \underbrace{\left(R_U \begin{pmatrix} \Sigma & 0 \\ 0 & 1 \end{pmatrix} R_V^\top \right)}_{=:K} Q_V^\top.$$

The last step requires taking the SVD of the middle matrix K. Note that $K \in \mathbb{R}^{(r+1)\times(r+1)}$, so the cost of computing the SVD of K is $O((r+1)^3)$. Since we assume that $r \ll \min\{m, n\}$, this is much cheaper than computing the leading r singular values of the A. If $K = U_K \Sigma_K V_K^\top$ is an SVD of K, then the SVD of B is given by,

$$B = \begin{pmatrix} Q_U & U_K \end{pmatrix} \Sigma_K \begin{pmatrix} Q_V & V_K \end{pmatrix}^\top.$$

The QR decompositions can be constructed efficiently by viewing the QR decompositions as QR updates to the matrices U and V, where $U = UI$ and $V = VI$ are valid QR factorizations since U and V already have orthogonal columns. A simple method to compute a rank-one update to a QR factorization involves Gram-Schmidt to reorthogonalize the Q matrix, and Givens rotations to ensure that the R matrix is upper triangular [38]. Thus, constructing the matrices Q_U and Q_V only require Gram-Schmidt. The final step of applying Givens rotations to ensure that R_U and R_V are upper triangular is not necessary, since these matrices are only used in a product to compute K. Thus, the dominant cost is usually the SVD of the $(r+1)\times(r+1)$ system K.

3.4 Convergence Analysis

Following the proof for Theorem 4 in [40], we show that the iterates, from Rank Drop FW in Algorithm 5, converge to the global optimum of (3.1).

Theorem 3.4.1. *Let $\{X^{(k)}\}$ be a sequence generated by Algorithm 5 and let f^* be the optimal value for problem (3.1). Assume $\nabla f(X)$ is Lipschitz continuous in the feasible region. Then $f(X^{(k)}) - f^* \leq \frac{8\delta^2 L}{4+N_{fw}^k}$, where N_{fw}^k be the number of FW steps taken up to the iteration k.*

Proof. We will use a similar proof as [40]. Since rank-drop steps always decrease the rank of the solution, the number of rank-drop steps is bounded by the number of Frank-Wolfe steps. Thus, any sequence $\{X^{(k)}\}$ contains an infinite number of Frank-Wolfe steps. Since rank-drop steps can only decrease the objective, the convergence is guaranteed by the same arguments as the regular Frank-Wolfe algorithm. □

3.5 Complexity Per Iteration

When computing the rank-drop steps, we note that the dimension of subproblems (3.17) and (3.29) is r, the rank of the current iterate. However, forming the matrix $W \coloneqq U^\top \nabla f(X^{(k)})V$ requires $O(r^2 h \min\{m, n\})$ operations, where h is the maximum number of nonzero elements in any row or column. In the interior case, we must compute an eigendecomposition of an $r \times r$ matrix which takes $O(r^3)$ time. Then, each eigenvalue λ is used to form the matrix $-0.5(W + \lambda\Sigma^{-1})$ where the singular vector pair corresponding to the zero singular value is computed. The total time required for the interior case is $O(r^3 + r^2 h \min\{m, n\})$. In the exterior case, $O(r^2)$ flops are required to compute the largest eigenvalue. Thus, the total complexity per iteration is $O(r^3 + r^2 h \min\{m, n\})$.

Algorithm 4 Compute Rank-Drop Direction (`rankDrop`)

```
1: Input: thin SVD $X^{(k)} \coloneqq U\Sigma V^\top$ and $\nabla f(X^{(k)})$.
2: $\kappa(X^{(k)}) \leftarrow \frac{\delta - \|X^{(k)}\|_{\mathrm{tr}}}{2}$
3: $W \leftarrow U^\top \nabla f(X^{(k)}) V$
4: if $\kappa(X^{(k)}) \geq \sigma_r(X^{(k)})$ then
5:     (Interior Case)
6:     $\Lambda \leftarrow \texttt{eigs}(-\Sigma W)$
7:     $b \leftarrow -\infty$
8:     for $\lambda_i \in \Lambda$ do
9:         $M_\lambda \coloneqq -\frac{1}{2}(W + \lambda_i \Sigma^{-1})$
10:        $(s, t) \leftarrow \texttt{SVD}(M_\lambda)$ [1]
11:        if $\kappa(X^{(k)}) s^\top \Sigma^{-1} t \geq 1$ and $q\left(s, \frac{t}{\kappa(X^{(k)}) s^\top \Sigma^{-1} t}\right)$[2] $> b$ then
12:            $(s^*, t^*) \leftarrow (s, t)$
13:            $b \leftarrow q\left(s, \frac{t}{\kappa(X^{(k)}) s^\top \Sigma^{-1} t}\right)$
14:        end if
15:    end for
16:    $\alpha \leftarrow (s^\top \Sigma^{-1} t)^{-1}$
17:    $\alpha_{\max} \leftarrow \frac{\alpha}{\delta - \alpha}$
18: end if
19: if $\kappa(X^{(k)}) < \sigma_r(X^{(k)})$ or $b = -\infty$ then
20:    (Exterior case or no candidates from the Interior Case)
21:    $s^* \leftarrow \texttt{genEig}(0.5(W + W^\top), \Sigma^{-1})$ [3]
22:    $t^* \leftarrow s^*$
23:    $\alpha_{\max} \leftarrow (\delta s^\top \Sigma^{-1} s - 1)^{-1}$
24: end if
25: return $(s^*, t^*, \alpha_{\max})$
```

[1]Return the singular vector pair corresponding to the singular value 0

[2]q is the objective of the interior rank-drop problem (3.17).

[3]Solve the generalized eigenvalue problem for the system $0.5(W + W^\top)x = \lambda \Sigma^{-1} x$.

Algorithm 5 Rank-Drop Frank-Wolfe (**RDFW**)

```
1: Let $X_0 \in \mathcal{S}$, with initial SVD $X_0 = U\Sigma V^\top$, and maximum iteration $T$
2: for $k = 0...T$ do
3:     Compute $\nabla f(X^{(k)})$
4:     Compute Rank-Drop direction (see Algorithm 4):
5:     $(s_k, t_k, \alpha^{(k)}) \leftarrow \texttt{rankDrop}(U, \Sigma, V, \nabla f(X^{(k)}))$
6:     $\tilde{X} \leftarrow X^{(k)} + \alpha_{\max}(X^{(k)} - \delta U s_k t_k^\top V^\top)$
7:     if $f(\tilde{X}) \le f(X^{(k)})$ then
8:         $X_{k+1} \leftarrow \tilde{X}$
9:         $Z^{(k)} \leftarrow -\delta U s_k t_k^\top V^\top$
10:    else
11:        (Frank-Wolfe)
12:        $Z^{(k)} \leftarrow \arg\min_{Z \in \mathcal{B}_{\mathrm{tr}}(0,\delta)} \langle Z, \nabla f(X^{(k)}) \rangle$
13:        $\alpha^{(k)} \leftarrow \arg\min_{\tau \in [0,1]} f(X^{(k)} + \tau(Z^{(k)} - X^{(k)}))$
14:        $X_{k+1} \leftarrow X^{(k)} + \alpha^{(k)}(Z^{(k)} - X^{(k)})$
15:    end if
16:    $(U, \Sigma, V) \leftarrow \texttt{updateSVD}(U, \Sigma, V, \alpha_{\max}, Z^{(k)})$
17: end for
```

3.6 Experimental Results

We validate RDFW on a matrix completion task using various datasets from MovieLens[1]. We first center and scale each data set to have mean 0 and standard deviation 1. We

Dataset	**# Users**	**# Movies**	**# Ratings**
MovieLens 100k	943	1,682	100,000
MovieLens 1M	6,040	3,900	1,000,209
MovieLens 10M	82,248	10,681	10,000,054
MovieLens 20M	138,493	27,278	20,000,263

Table 3.1: MovieLens Data

compare the proposed RDFW algorithm against the aforementioned Frank-Wolfe variants:

1. The original Frank-Wolfe algorithm with no modifications (Vanilla) [31]

[1] http://grouplens.org/datasets/movielens/

2. The away-step variant of Frank-Wolfe using the atoms found observed from the iterations, (Away-Atom) [51]

3. The away-step variant using the SVD as the atomic set, (Away-SVD) [33]

4. the In-Face steps using the InFace$(0, \infty)$ variant [33], (In-Face)

We will also compare the performance for matrix completion problems against a state-of-the-art trace norm regularized solver, ActiveALT [44], which we will briefly summarize. The algorithm solves the trace norm regularized problem $\min_X F(X) \coloneqq f(X) + \lambda \|X\|_{\mathrm{tr}}$. The ActiveALT algorithm begins by expressing any iterate as $X^{(k)} = \sum_{ij} \sigma_{ij} u_i v_j^\top$, where $U = [u_i]$ and $V = [v_i]$ are orthonormal bases for $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. The idea is to prune subspaces that will likely be inactive at the optimal solution, i.e., find the indices (i, j) where $\sigma_{ij} = 0$ at optimality. The pruning rule is based on the subdifferential $\partial_{\sigma_{ij}} F$, and the active subspace at a given iteration $X^{(k)}$, are identified as

$$\mathcal{A}(X^{(k)}) = \{uv^\top : u \in U, v \in V, u^\top X^{(k)} v \neq 0 \text{ or } |u^\top \nabla f(X^{(k)} v| \leq \lambda\}$$

where we use the notation $u \in U$ to denote that u is a column of U.

Once the active subspaces are obtained, the matrices U_A and V_A are formed where

$$\begin{aligned} U_A &\coloneqq \{u : \exists v \in V : uv^\top \in \mathcal{A}(X^{(k)})\}, \text{ and} \\ V_A &\coloneqq \{v : \exists u \in U : uv^\top \in \mathcal{A}(X^{(k)})\}. \end{aligned}$$

The optimization problem reduces to,

$$\min_{S \in \mathbb{R}^{r \times r}} f(U_A S V_A^\top) + \|U_A S V_A^\top\|_{\mathrm{tr}} = f(U_A S V_A^\top) + \|S\|_{\mathrm{tr}}, \tag{3.30}$$

where r is the number of active columns in U and V. Empirically, it is observed that $r \ll \min\{m, n\}$ and the problem is effectively solved using an alternating minimization approach using second order information for the smooth loss function f.

3.6.1 Experimental Setup

Following [81], we randomly partition each dataset into 50% training, 25% validation, and 25% testing. The δ value in (3.1) is tuned with $\delta = \mu_j \cdot \|Y\|_F$, where $\|Y\|_F$ is the Frobenius norm of the training data matrix, and $\mu_j = 2 + 0.2j, j \in \mathbb{N}$. We increase j until the mean RMSE on the validation set does not improve by more than 10^{-3}. We terminate the algorithm when an upper bound on the relative optimality gap ensures $(f(X^{(k)}) - f^*)/f^* < 10^{-2}$ or a maximum iteration count of 2000 is reached. Since the optimal solution f^* is unknown to us, we use

$$B^{(k)} = \max\{B^{(k-1)}, f(X^{(k)}) + \min_{S \in \mathcal{D}} \langle \nabla f(X^{(k)}), S - X^{(k)} \rangle$$

as described in Algorithm 3 as a lower bound on f^*.

For ActiveALT, a regularized trace norm problem is solved where the regularization parameter λ is chosen by approximately solving for the Lagrange multiplier from the solution to the constrained problem. From the optimality conditions, we have $U^\top \nabla f(X^*) V + \lambda I = 0$. Thus, λ is approximated by the mean of the diagonal values of $U^\top f(X^{(k)}) V$, where $X^{(k)}$ is the converged solution of RDFW. ActiveALT terminates when $f(X^{(k-1)}) - f(X^{(k)}) < 10^{-4}$, to match with the criterion suggested in [81] with a maximum iteration count of 150.

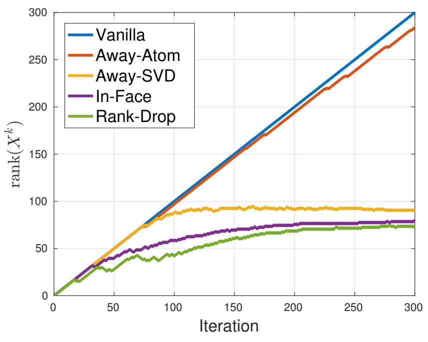
Dataset	Frank-Wolfe δ	ActiveALT λ
MovieLens 100k	3.0	10.94
MovieLens 1M	3.4	22.7
MovieLens 10M	5.2	49.4
MovieLens 20M	6.6	59.04

Table 3.2: Parameters used for each dataset.

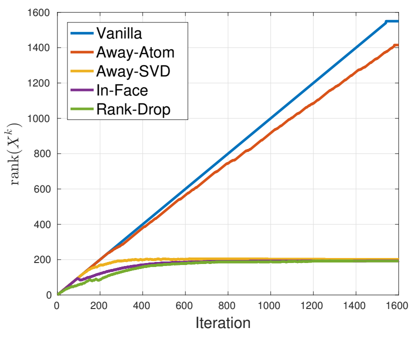
3.6.2 Computational Details

All simulations have been run in MATLAB. For all FW variants, we maintain a thin SVD for the current iterate, where the SVD is updated at each iteration using a rank-one update as described in [10]. The rank is calculated using the same default rank criterion in MATLAB, by counting all singular values larger than $\sigma_1(X^{(k)}) \cdot \epsilon$, where $\epsilon \approx 2.2204e-16$[2].

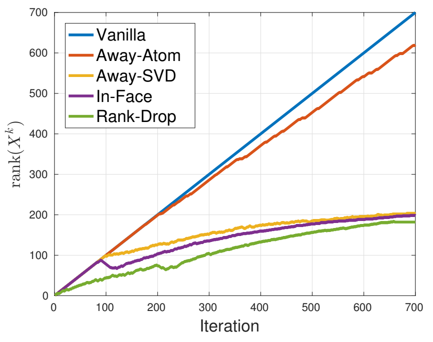
[2]A tolerance of 10^{-6} was also tested and the results were almost identical. This can be explained by the fact that anytime any of the away-step SVD variant, in-face steps, or RDFW achieved a step size that should remove an active atom, we deliberately truncate the SVD to avoid numerical issues. The tolerance is only meant to impact the vanilla and natural away-step ranks, but experimentally, we did not notice any differences on the final rank.



(a) MovieLens-100k

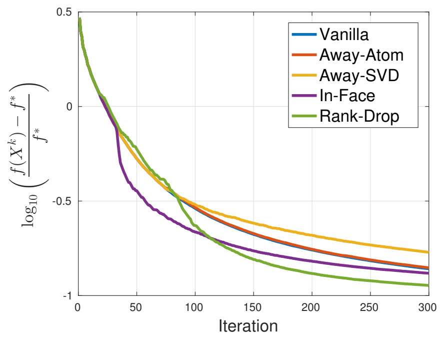


(b) MovieLens-1M

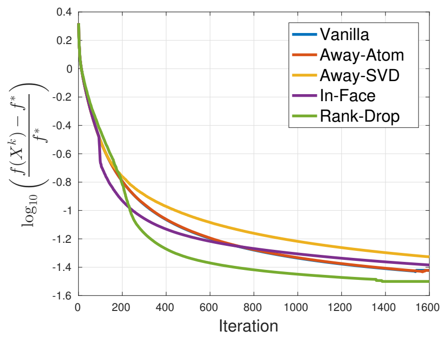


(c) MovieLens-10M

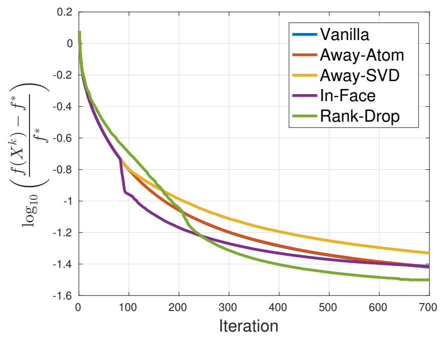
Figure 3.3: The ranks of the iterates $X^{(k)}$ compared with the iteration for each algorithm.



(a) MovieLens-100k

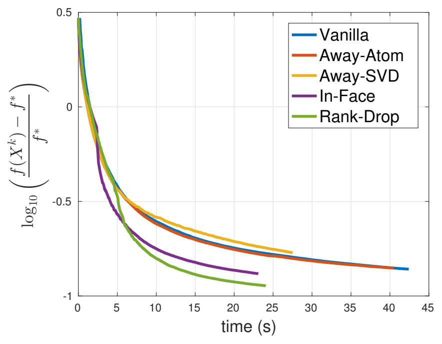


(b) MovieLens-1M

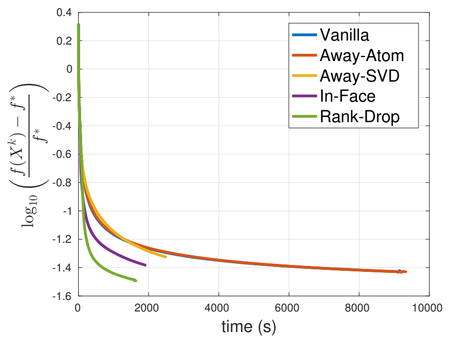


(c) MovieLens-10M

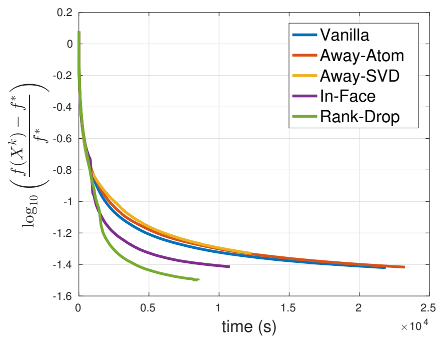
Figure 3.4: The objective values compared with the iterations.



(a) MovieLens-100k



(b) MovieLens-1M



(c) MovieLens-10M

Figure 3.5: The objective values compared with the time (s).

Dataset		Frank-Wolfe	Away-Step (Atom)	Away-Step (SVD)	In-Face$(0, \infty)$	ActiveALT	Rank-Drop FW
	RMSE	0.874	0.874	0.874	0.874	0.876	0.872
ML-100k	Rank (Max)	943 (943)	943 (943)	82 (97)	81.6 (138)	85 (108)	80 (81)
	Time (s)	852.34	807.86	259.19	166.17	92.13	121.90
	RMSE	0.805	0.805	0.806	0.807	0.805	0.807
ML-1M	Rank (Max)	1,539 (1,550)	1,405.4 (1,415)	199.2 (208)	192.8 (195)	192 (206)	192 (194)
	Time (s)	9,182.07	9,225.69	3,209.33	2,507.60	1,677.23	1,593.35
	RMSE	0.799	0.799	0.800	0.800	0.799	0.800
ML-10M	Rank (Max)	724.2 (730)	661.8 (671)	212.8 (215)	201.2 (206)	196.2 (206)	180.8 (184)
	Time (s)	23,810.79	25,130.63	19,961.73	12,804.82	14,529.8	8,242.30
	RMSE	-	-	-	0.800	0.800	0.801
ML-20M	Rank (Max)	-	-	-	274.6 (471)	206.2 (214)	202 (203)
	Time (s)	-	-	-	117,535.82	38,497.16	29,102.62

Table 3.3: Computational results on matrix completion problems averaged over 5 random initializations. The max rank is the maximum rank observed over all 5 trials. For ML-20M, the FW and AFW algorithms took too long to successfully terminate.

3.7 Discussion

When we observe the rank plots in Figure 3.3, we see that the regular Frank-Wolfe algorithm and the atomic away-step variant both maintain very high-rank iterates. In addition, we observe in Figure 3.4 that the atomic away steps do not appear to improve the convergence rate of the Frank-Wolfe method. This can be attributed to the fact that the trace norm ball is not a polytope and the away steps can still become orthogonal to the gradient direction.

While both the in-face step and the SVD variant of the away-step can maintain low-rank iterates, there are tradeoffs compared to the RDFW method we propose. The SVD variant of the away steps converges very slowly both in terms of the iteration counts and CPU time as seen in Figures 3.4 and 3.5. Moreover, we see that the ranks of the SVD variant of the away step is slightly higher than that of the in-face step or the RDFW method we suggest.

The in-face steps appears to make more improvement per iteration when the iterate is inside the trace norm ball than RDFW. This makes sense since the away-steps in the interior of the trace norm are optimized for the best descent direction amongst all away steps in the trace norm ball. However, these interior in-face steps do not typically lead to rank-drop steps and calculating the step size requires a costly binary search. We observe that the rank of the in-face steps is not as low as RDFW on the earlier iterations. Moreover, we notice that when the in-face step reaches the boundary of the trace norm ball, the convergence becomes much slower than RDFW. We conjecture that this is because the set of rank-drop steps we consider becomes much larger than the set of in-face steps available, allowing for more promising descent directions. From the CPU time perspective, we see that this difference magnifies as the size of the dataset increases.

When comparing to ActiveALT, the leading trace norm based matrix completion solver, is very competitive. RDFW has the additional attractive property that it does not require knowledge of the structure of the Hessian, or even require $f(\cdot)$ to be twice differentiable, allowing for greater generality. Moreover, since the matrix completion objective is quadratic, this is the ideal situation for ActiveALT since the underlying solver in ActiveALT only requires one Newton step to converge, whereas for a nonquadratic objective, the computational challenge can increase.

3.8 Conclusions

We have proposed a rank-drop optimization formulation to determine optimally descent rank-drop steps for the nuclear-norm constrained minimization. By considering the interior and exterior cases separately to ensure feasibility, we also devise subproblems that can be efficiently solved. The proposed formulation can be deployed in a projection free minimization method, e.g., Frank-Wolfe method, to efficiently compute a low rank solution by maintaining low rank intermediate iterates, without compromising the strong convergence guarantees. While classic Frank-Wolfe methods tend to have very high rank solutions for nuclear-norm constrained problems, we have shown that the addition of rank-drop steps can drastically reduce the rank of the iterates, allowing for much faster algorithms to reach low rank solutions with a significantly smaller space requirement.

Chapter 4

Nonsmooth Extensions to Frank-Wolfe

4.1 Introduction

Frank-Wolfe methods (FW) have gained significant interest in the machine learning community due to its ability to efficiently solve large problems that admit a sparse structure (e.g. sparse vectors and low-rank matrices). However the performance of the existing Frank-Wolfe method hinges on the quality of the linear approximation. This typically restricts Frank-Wolfe to smooth functions for which the approximation quality, indicated by a global curvature measure, is reasonably good.

We are primarily interested in solving problems of the form,

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) \; s.t. \; \|X\|_{\mathbf{tr}} \leq \delta$$

which we recognize as a convex surrogate to the rank constrained optimization problem. As discussed in the previous chapter, this problem is well studied in the case where f is a smooth convex function. For example, in matrix completion, many efficient algorithms have been proposed, including the Rank-Drop Frank-Wolfe [18], active set methods [44], and proximal methods [62].

Recently, there has also been interest in solving the trace norm constrained problem where the objective function is not differentiable, e.g.,

$$\min_{X : \|X\|_{\mathbf{tr}} \leq \delta} f(X) := L(X) + \lambda_1 \|X\|_1. \tag{4.1}$$

where $L(X)$ is an empirical loss function. For example, problem (4.1) has been found useful [69] for sparse covariance estimation and graph link prediction, for which solutions are expected to exhibit simultaneously sparse and low-rank structure. An inherent challenge with handling simultaneously sparse and low-rank structure is that these goals are often competing objectives.

Consider the task of predicting missing edges of a social graph. As an input, we are given a partially observed symmetric graph $Y \in \{0,1\}^{m\times m}$, where only the indices in $\Omega \subseteq \{1,...,m\} \times \{1,...,m\}$ are known. The entry $Y_{ij} = 1$ indicates that users i and j are friends, and $Y_{ij} = 0$[1] indicates that these users are not friends. The social graph link prediction problem can be written as,

$$\min_{X\in\mathbb{S}^m} \|P_\Omega(X-Y)\|_F^2 + R(X)$$

where $\mathbb{S}^m$ is the set of symmetric $m \times m$ matrices, $P_\Omega(\cdot)$ projects the loss onto the set Ω, and $R(X)$ is a regularization term. Suppose that we only observed concrete friendships, that is, $Y_{ij} = 1, \forall (i,j) \in \Omega$, for example,

$$Y = \begin{pmatrix} 1 & ? & ? & 1 & 1 \\ ? & 1 & ? & ? & ? \\ ? & ? & 1 & ? & 1 \\ 1 & ? & ? & 1 & ? \\ 1 & ? & 1 & ? & 1 \end{pmatrix}.$$

If $R(X) = \|X\|_1$, the optimal solution is achieved by setting all the unboserved entries to 0, i.e.,

$$X^* = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

However, if instead $R(X) = \|X\|_{\mathrm{tr}}$, the optimal solution is achieved by setting all the

[1]Observing that two users are not friends is typically difficult to measure directly. Certain actions such as removing friends and rejecting requests would fall into this category, but measuring inactivity to label two users as not friends (e.g. not trying to connect with a suggested user) will require some rule based labeling.

observed entries to 1, i.e.,

$$X^* = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We see that the solutions given by the regularizations are on the opposing extremes where the predictors do not agree on any prediction for the edges on the unobserved set. When $R(X) = \|X\|_1$, the solution is a full-rank matrix with a very high degree of sparsity. On the other hand, when $R(X) = \|X\|_{\text{tr}}$, the solution is a rank-one matrix that is completely dense. In both cases, the prediction is not very informative and the truth is likely somewhere in between. The low-rank assumption encodes the group structure, where we assume that if user i has many friends in common with user j, then user j's friends are good suggestions for the missing labels for i. The sparse regularizer encodes the assumption that most people are not friends with each other and we do not expect too many relationships in the graph. By simultaneously requiring both low-rank and sparsity in our iterates, intuitively the returned graph should have a few dense clusters of friends with few connections between groups which better agrees with reality compared to either extreme from low-rank or sparsity alone.

A natural approach to (4.1) is to treat the sum of the trace and ℓ_1 norms as one nonsmooth regularization function, e.g.

$$\min_X L(X) + \lambda(\|X\|_1 + \|X\|_{\text{tr}}) \tag{4.2}$$

which is a sum of a smooth loss and nonsmooth regularization term and the standard framework for proximal methods or Frank-Wolfe can be applied. However, the appeal for using the proximal method or Frank-Wolfe exist only when the oracle can be evaluted efficiently. Unfortunately, the proximal map for the combined regularizer in (4.2) is no longer a simple thresholding operation. In [69], alternating proximal steps are taken, but this method still fails to scale due to the full SVD computation required at each iteration. In addition, the alternating heuristic loses convergence guarantees.

When there are both trace norm and ℓ_1 norm constraints, the LMO for Frank-Wolfe is much more expensive to compute than the ℓ_1 or trace norm case alone. We also lose the structural guarantees of the iterates, i.e., we cannot guarantee each iterate has at most k nonzeros or has rank bounded by k at the k^{th} iteration.

We consider the nonsmooth objective with ℓ_1 regularization and trace norm constraints, explicitly

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) + \lambda \|X\|_1 \text{ s.t. } \|X\|_{\mathrm{tr}} \leq \delta.$$

We note that when (4.1) corresponds to matrix completion, solvers such as the active set method in [44] explicitly require second order information, making it unsuitable when the function is not differentiable, let alone twice differentiable, and are not suitable for (4.1).

We propose a variant of the Frank-Wolfe algorithm to address nonsmooth objectives, focusing especially on low-rank matrix estimation problems. Nondifferentiability in the objective function often leads to an unbounded curvature constant, and standard convergence analysis can no longer be applied. Moreover, it becomes unclear how to define the linear approximation appropriately since choosing an arbitrary subgradient often leads to inadequate local approximations, leading to poor empirical results.

To address these issues, we replace the traditional linear minimization problem based on a Chebyshev uniform affine approximation. This modification allows for a well-defined linear optimization problem even when the objective is nonsmooth or has unbounded curvature. We demonstrate experimentally that this carefully selected linear minimization leads to significant improvement over a variety of matrix estimation problems, such as sparse covariance estimation, graph link prediction, and ℓ_1-loss matrix completion.

4.2 Curvature and Nonsmoothness

Recall that for smooth convex functions, the Frank-Wolfe algorithm is known to converge at a rate of $O(1/k)$. The convergence analysis relies on the concept of *curvature constant* [22, 45], which measures the quality of the linear approximation.

Let f be a convex and differentiable function $f : \mathbb{R}^{m \times n} \to \mathbb{R}$, and let $\mathcal{D}$ be a convex and compact subset of $\mathbb{R}^{m \times n}$. Recall the curvature constant

$$C_f := \sup_{\substack{X,S \in \mathcal{D} \\ \alpha \in [0,1] \\ Y = X + \alpha(S - X)}} \frac{1}{\alpha^2}\left(f(Y) - f(X) - \langle Y - X, \nabla f(X) \rangle\right).$$

When the value of C_f is large, it suggests that there are regions in $\mathcal{D}$ where the local linear approximation is poor.

Suppose we wish to extend the concept of C_f to functions $f(x)$ that are convex but nondifferentiable. We may consider redefining the curvature constant by replacing $\nabla f(X)$

with a substitute $G(x) : \mathbb{R}^n \to \mathbb{R}^n$, where a natural choice for $G(x)$ is a subgradient of f at x. However, even for simple functions, we show that the curvature constant derived from *any* linear approximation in this fashion will be unbounded.

Example 1. *Let $f(x) = \lambda\|x\|_1$ and let $\mathcal{D}$ be some convex and compact set that contains an open ball around the origin. Assume $x = 0$ and $y = x + \alpha(s - x)$. Then for any $s \in \mathcal{D} \setminus \{0\}$ and any $G(x)$, we have,*

$$\begin{aligned}
&\frac{1}{\alpha^2}(f(y) - f(x) - \langle y - x, G(x)\rangle) \\
&= \frac{1}{\alpha^2}(\alpha\lambda\|s\|_1 - \alpha\langle s, G(x)\rangle) \\
&= \frac{1}{\alpha}(\lambda\|s\|_1 - \langle s, G(x)\rangle)
\end{aligned}$$

Note that there always exists s and subgradient $G(x)$ such that $\langle s, G(x)\rangle \leq 0$. It follows that, for such s and $G(x)$,

$$\lim_{\alpha\to 0} \frac{1}{\alpha}\left(\lambda\|s\|_1 - \langle s, G(x)\rangle\right) = +\infty.$$

Hence

$$\lim_{\alpha\to 0} \frac{1}{\alpha^2}(f(y) - f(x) - \langle y - x, G(x)\rangle) = +\infty.$$

This example shows that, for $f(x) = \lambda\|x\|_1$, no matter what subgradient is chosen, the curvature constant is unbounded. This suggests that there are regions where the linear approximation is very poor. And this is easily observable in practice, not just an overly pessimistic bound given by the curvature constant. For $f(x) = \lambda|x|$. If $x = \epsilon$ for some small $\epsilon > 0$, then f is differentiable at x. However, for all $y < 0$, the linear approximation given by the Taylor series at x poorly approximates $f(y)$.

This indicates that the original Frank-Wolfe may not be suited for minimizing the objective function in (4.1). However, the Taylor approximation to ℓ_1 norm is exact in a local neighborhood except at around the points of nondifferentiability, which suggests that the problem is very well suited for Frank-Wolfe if $\lambda\|x\|$ is differentiable in a neighborhood around x. This motivates the question about how to meaningfully define an appropriate linear minimization problem around neighborhoods of nondifferentiability, specifically for the Frank-Wolfe algorithm.

4.2.1 Existing Work

To overcome the issue of poor Taylor approximations around points of nondifferentiability, an approximate subdifferential, $T(X, \epsilon)$, which considers all the subgradients for any Y in an ϵ-neighborhood of X, is employed in [79]. Specifically, the approximate subdifferential is defined as,

$$T(X, \epsilon) := \begin{cases} \nabla f(X), & \text{if } f \text{ is differentiable at } X \\ \bigcup_{Y \in \mathcal{B}_F(X,\epsilon)} \partial f(Y), & \text{otherwise.} \end{cases}$$

The linear minimization problem of Frank-Wolfe is then replaced by the following problem,

$$\min_{S \in \mathcal{D}} \max_{G \in T(X,\epsilon)} \langle -G, S - X \rangle. \tag{4.3}$$

The maximization over all approximate subgradients ensures that the direction is not misled by a particularly poor linear approximation. This leads to a natural notion of curvature as defined in [68], where the convergence arguments are extended by utilizing a proposed a generalized curvature constant,

$$C_f(\epsilon) := \sup_{\substack{X,S \in \mathcal{D} \\ \alpha \in [0,1] \\ Y = X + \alpha(S - X)}} \min_{G \in T(x,\epsilon)} \frac{1}{\alpha^2}(f(Y) - f(X) - \langle Y - X, G \rangle).$$

However, there are a few practical issues with using the above ideas. First, since G is not necessarily a subgradient of f at X, the difference $f(Y) - f(X) - \langle Y - X, G \rangle$ can become negative. Thus, taking the supremum over these values may not measure the maximum deviation of the linear approximation; in particular, it does not measure deviations from overestimating linear approximations. Furthermore, it is not obvious how to extend these ideas to solve (4.1) efficiently since S and G must be solved jointly.

In [65] and [3], the objective in (4.1) is replaced by a smoothed objective. The gradients of the smoothed objective are given by,

$$[G^{(k)}(X^{(k)})]_{ij} = \nabla L(X^{(k)}) + \begin{cases} \lambda\, \mathbf{sgn}(X_{ij}), & \text{if } |X_{ij}^{(k)}| \geq \mu \\ \frac{\lambda}{\mu} X_{ij}^{(k)}, & \text{if } |X_{ij}^{(k)}| < \mu, \end{cases}$$

for the Smoothed Composite Conditional Gradient (SCCG) algorithm in [65], and

$$G^{(k)}(X^{(k)}) = \nabla L(X^{(k)}) - \frac{1}{\beta^{(k)}} X_k + \frac{1}{\beta^{(k)}} S(X^{(k)}, \lambda\beta^{(k)})$$

for the Hybrid Conditional Gradient with Smoothing (HCGS) algorithm in [3], where

$$S(X^{(k)}, \lambda\beta^{(k)}) = \mathbf{sgn}(X) \odot \max\{|X| - \lambda\beta^{(k)}, 0\}$$

is the soft-thresholding operator. In both cases, the objective is replaced by a smooth function defined by smoothing parameters μ and $\beta^{(k)}$ respectively, where the smooth approximation given is the best approximation across all $1/\mu$-smooth (or resp. $1/\beta^{(k)}$) functions. However, determining how smooth the approximation should be is not easy to know a priori and often varies depending on the iterate. Empirically, we also observe that there is a nontrivial dependency between the smoothing parameters and convergence rate. At any given iteration, if the smoothing parameters are not set appropriately, the algorithm often makes no progress for many iterations.

In [83], a nonsmooth generalization to the rank-one matrix pursuit is proposed which utilizes subgradients in the linearized subproblem. To ensure convergence, the rank-one update at each iteration is replaced by a rank-k variant where k is computed by taking however many leading singular vectors are required to ensure the solution to the rank-k subproblem is not too far from the subgradient measured with the ℓ_2 norm. Since the subgradients are typically not low rank, the number of singular vectors required can often be very large (possibly requiring a full SVD), and this approach can still fail to scale in similar ways to proximal methods.

The work in [61] also considers Frank-Wolfe methods when the curvature constant is unbounded. However, the algorithm is specific to the phase retrieval problem for which the objective is still differentiable, simplifying the analysis.

The Generalized Forwards-Backwards (GenFB) algorithm was introduced to solve (4.1) by alternating between proximal steps using the trace and ℓ_1 norms [69]. As alluded to earlier, these algorithms tend to scale poorly due to a full SVD required at each iteration.

4.2.2 Achieving a better linear approximation

The previous work on Frank-Wolfe for nonsmooth minimization (4.1) shares a common idea, i.e., finding a meaningful way to define an appropriate linear optimization subproblem in a scalable manner. We consider directly minimizing the approximation error over all possible affine functions over a neighborhood specified carefully for the Frank-Wolfe steps. We will show that under modest assumptions, the linear subproblems we propose will be simple to solve and do not rely on specifying the desired level of smoothness as required by methods discussed in the previous section.

Definition 4.2.1. *Given some $r > 0$, the* **uniform affine approximation** *to a function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is defined as $\ell(Y) = b^* + \langle Y - X, \xi^* \rangle$ where,*

$$(\xi^*, b^*) \in \operatorname*{arg\,min}_{(\xi,b)} \max_{Y\in\bar{\mathcal{B}}_\infty(X,r)} |f(Y) - b - \langle Y - X, \xi\rangle| \tag{4.4}$$

and $\bar{\mathcal{B}}_\infty(X, r)$ is the closed element-wise infinity norm ball of radius r around X.

The above uniform affine approximation can be viewed as choosing an optimal affine approximation in a given neighborhood of X. The optimality is defined in the uniform sense, by minimizing the maximum absolute deviation from the original objective f.

This motivates a natural variant of Frank-Wolfe, where at each iteration, the linear subproblem using a subgradient is replaced with the uniform affine approximation. In particular, we can view the Frank-Wolfe iterates as,

$$X^{(k+1)} = X^{(k)} - \alpha^{(k)}(X^{(k)} - S)$$

where $S \in \mathcal{D}$ is given by the LMO. Thus, $X^{(k+1)} \in \bar{\mathcal{B}}_\infty(X^{(k)}, \alpha^{(k)}\Delta)$ and $\Delta = \mathbf{diam}(D) = \max_{S\in\mathcal{D}}\|X^{(k)} - S\|_\infty$, where $\|\cdot\|_\infty$ is taken element-wise. For Frank-Wolfe, this implies that we can restrict our attention to a neighborhood around the current iterate $X^{(k)}$ where the neighborhood has the radius $\alpha^{(k)}\Delta$.

Specifically, we observe that in Frank-Wolfe there exist step size schedules, e.g., $\alpha^{(k)} = 2/(k + 2)$, which are independent of the current iterate and guarantee convergence. Our proposed approach is to assume that such a step size schedule is specified *a priori* and to use the uniform affine approximation defined by the step size schedule for the Frank-Wolfe subproblems. This allows the linear optimization subproblems to be defined a meaningful way that is related to the Frank-Wolfe steps and does not require solving complicated subproblems as in [68] or understanding the desired curvature for the problem as in [3] or [65]. Moreover, we show that we no longer require f to have bounded curvature constant or even to be differentiable to guarantee convergence.

4.3 Frank-Wolfe with Uniform Approximations

For the proposed uniform approximation approach to be viable, it is important that the uniform affine approximation can be calculated efficiently. We begin by considering real-valued functions and Chebyshev approximations.

4.3.1 Chebyshev Approximations

Given a real-valued function f and an interval $[a, b] \subseteq \mathbf{dom}(f)$, the Chebyshev polynomial $p_d(x)$, is a polynomial of degree not exceeding d that best approximates f on the interval $[a, b]$ in the uniform sense,

$$p_d(x) = \arg\min_{p \in \Pi_d} \max_{a \leq x \leq b} |f(x) - p(x)|$$

where Π_d is the set of polynomials of degree at most d.

Theorem 4.3.1 (Chebyshev Equioscillation Theorem)**.** *Let f be a continuous function from $[a, b] \to \mathbb{R}$ and let Π_d be the set of polynomials of degree less than or equal to d. Then*

$$g^* = \arg\min_{g \in \Pi_d} \|f - g\|_\infty$$

if and only if there exists $d + 2$ points $\{x_1, ..., x_{d+2}\}$ such that $a \leq x_1 < ... < x_{d+2} \leq b$ and

$$f(x_i) - g^*(x_i) = c(-1)^i \|f - g^*\|_\infty$$

where $c = 1$ or -1.

Although the equioscillation theorem only applies to a function of one variable, we will show it can also be applied when a function is separable. Specifically, under the separability Assumption 4.3.2 below, we can construct the best uniform affine approximation by determining the best affine approximation on an interval for each component function.

Assumption 4.3.2. *Assume that $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ can be* ***separated*** *into a sum of component functions, i.e.,*

$$f(X) = \sum_{i,j} f_{ij}(X_{ij}).$$

where each $f_{ij} : \mathbb{R} \to \mathbb{R}$.

Theorem 4.3.3. *Suppose $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ is a continuous function that satisfies Assumption 4.3.2. For a given $X^{(k)} \in \mathbb{R}^{m \times n}$ and $\tau > 0$, if $\ell_{ij}(Y_{ij})$ is the Chebyshev polynomial of degree 1 for f_{ij} over the interval $[X_{ij} - \tau, X_{ij} + \tau]$, then the function $\ell(Y) = \sum_{i=1}^{m} \sum_{j=1}^{n} \ell_{ij}(Y_{ij})$ is the uniform affine approximation to f as defined in (4.4).*

Proof. Following the equioscillation Theorem 4.3.1, there exists ℓ_{ij} such that,

$$\ell_{ij} = \arg\min_{p\in\Pi_1} \max_{Y_{ij}\in[X_{ij}^{(k)}-\tau, X_{ij}^{(k)}+\tau]} |f_{ij}(Y_{ij}) - p(Y_{ij})|. \tag{4.5}$$

Let

$$b + \langle Y - X^{(k)}, \xi\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} \ell_{ij}(Y_{ij}) \tag{4.6}$$

Since $f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}(X_{ij})$, we have that,

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{n} \max_{Y_{ij}\in[X_{ij}^{(k)}-\tau, X_{ij}^{(k)}+\tau]} |f_{ij}(Y_{ij}) - \ell_{ij}(Y_{ij})| \\
&\geq \max_{Y\in\bar{\mathcal{B}}_\infty(X^{(k)},\tau)} \left|\sum_{i=1}^{m}\sum_{j=1}^{n} f(Y_{ij}) - \ell_{ij}(Y_{ij})\right| \\
&= \max_{Y\in\bar{\mathcal{B}}_\infty(X^{(k)},\tau)} \left|f(Y) - b - \langle Y - X^{(k)}, \xi\rangle\right|.
\end{aligned} \tag{4.7}$$

By continuity of f and the equioscillation Theorem 4.3.1, there exists some $\bar{Y}_{ij} \in [X_{ij}^{(k)} - \tau, X_{ij}^{(k)} + \tau]$ such that

$$f_{ij}(\bar{Y}_{ij}) - \ell_{ij}(\bar{Y}_{ij}) = \max_{Y_{ij}\in[X_{ij}^{(k)}-\tau, X_{ij}^{(k)}+\tau]} |f_{ij}(Y_{ij}) - \ell_{ij}(Y_{ij})|. \tag{4.8}$$

Let $\bar{Y} = [\bar{Y}_{ij}]$. From $f_{ij}(\bar{Y}_{ij}) - \ell_{ij}(\bar{Y}_{ij}) \geq 0$ and $f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}(X_{ij})$,

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{n} \max_{Y_{ij}\in[X_{ij}^{(k)}-\tau, X_{ij}^{(k)}+\tau]} |f_{ij}(Y_{ij}) - \ell_{ij}(Y_{ij})| \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \big(f_{ij}(\bar{Y}_{ij}) - \ell_{ij}(\bar{Y}_{ij})\big) \\
&= f(\bar{Y}) + b + \langle \bar{Y} - X^{(k)}, \xi\rangle \\
&\leq \max_{Y\in\bar{\mathcal{B}}_\infty(X^{(k)},\tau)} \left|f(Y) - b - \langle Y - X^{(k)}, \xi\rangle\right|.
\end{aligned} \tag{4.9}$$

Combining (4.7) and (4.9), we have that,

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=1}^{n} \max_{Y_{ij}\in[X_{ij}^{(k)}-\tau, X_{ij}^{(k)}+\tau]} |f_{ij}(Y_{ij}) - \ell_{ij}(Y_{ij})| \\
&= \max_{y\in\bar{\mathcal{B}}_\infty(X^{(k)},\tau)} \left|f(Y) - b - \langle Y - X^{(k)}, \xi\rangle\right|
\end{aligned} \tag{4.10}$$

□

Thus, under Assumption 4.3.2, it suffices to find the Chebyshev polynomials for the component functions. Assuming additionally that $f(X)$ is convex, we can further characterize the uniform affine approximation.

We use the following lemma.

Lemma 4.3.4 (Convex Mean Value Theorem [78]). *If $f(x)$ is a closed proper convex function from $\mathbb{R}^{m\times n} \to \mathbb{R}$ for X in a convex set $\mathcal{D} \subseteq \mathbb{R}^{m\times n}$, then X_0 and $X_1 \in \mathbf{relint}(\mathcal{D})$ implies that there exists $0 < t < 1$, and a subgradient $G \in \partial f(C)$, where $C = tX_0 + (1-t)X_1$, such that $f(X_1) - f(X_0) = \langle X_1 - X_0, G\rangle$.*

Theorem 4.3.5. *Suppose that f satisfies Assumption 4.3.2 and each f_{ij} is closed, proper, and convex. Then the best affine approximation as defined in (4.4) to each f_{ij} on the interval $[a,b]$ is given by,*

$$\ell_{ij}(x) = \frac{f_{ij}(c) + h_{ij}^{+}(c)}{2} + \frac{f_{ij}(b) - f_{ij}(a)}{b-a}(x-c)$$

where

$$h_{ij}^{+}(x) := f_{ij}(a) + \frac{f_{ij}(b) - f_{ij}(a)}{b-a}(x-a)$$

and $c \in (a,b)$ is chosen to satisfy Lemma 4.3.4, the convex mean value theorem [78] for f_{ij} on $[a,b]$.

Proof. The affine function $h_{ij}^{+}(x)$ defines the line that connects $(a, f_{ij}(a))$ to $(b, f_{ij}(b))$. Since f_{ij} is convex, $f_{ij} \leq h_{ij}^{+}$ on $[a,b]$.

From Lemma 4.3.4, there exists $c \in (a,b)$ such that

$$\frac{f_{ij}(b) - f_{ij}(a)}{b-a} \in \partial f_{ij}(c)$$

Define

$$h_{ij}^{-}(x) = f_{ij}(c) + \frac{f_{ij}(b) - f_{ij}(a)}{b-a}(x-c)$$

The function $h_{ij}^{-}(x)$ is the line tangent to $f_{ij}(x_{ij})$ at $x_{ij} = c$ and is parallel to h_{ij}^{+}. Since f_{ij} is convex, $f_{ij} \geq h_{ij}^{-}$ on $[a,b]$.

By construction, ℓ_{ij} is a line parallel and equidistant to the lines h_{ij}^{+} and h_{ij}^{-}. Thus, it is easy to verify that

$$f_{ij}(a) - \ell_{ij}(a) = -(f_{ij}(c) - \ell_{ij}(c)) = f_{ij}(b) - \ell_{ij}(b) \tag{4.11}$$

satisfying the equioscillation property. Thus, ℓ_{ij} is the minimax affine approximation to f_{ij} on $[a, b]$. $\square$

We remark that the computations in the subproblems do not require the specific value of c used in Theorem 4.3.5.

4.4 FWUA and Convergence

Using the uniform affine approximation, we propose a FW variant with Uniform Approximations (FWUA), which is described in Algorithm 6. The function `update_tau` will be described in full in Section 4.4.1, where a specific update rule for τ will be required to guarantee convergence. The idea will be to stop refining the neighborhood size τ, once the smooth approximation is sufficiently close to f. We will show that this will be sufficient to prove convergence.

To establish convergence, subsequently we make the following assumptions.

Assumption 4.4.1. *Assume that f satisfies Assumption 4.3.2. In addition, each component function f_{ij} has the following properties:*

(a) f_{ij} is a L_{ij}-Lipschitz continuous function.

(b) f_{ij} is closed, proper, and convex.

(c) f_{ij} is not differentiable on at most a finite set.

(d) If f_{ij} is differentiable at $a \in \mathbb{R}$, then it is also twice differentiable at a.

While the above set of assumptions appears restrictive, our main goal in this work is to efficiently solve (4.1) in the context of the trace-norm constrained matrix estimation problem which has a combination of ℓ_1 and ℓ_2 loss/regularization, for which these assumptions are typically satisfied.

Algorithm 6 Frank-Wolfe with Uniform Approximations (**FWUA**)

```
Input: $f$: A function satisfying Assumptions 4.3.2 and 4.4.1
       $\mathcal{D}$: A convex and compact subset of $\mathbb{R}^{m \times n}$
       $\epsilon$: Approximation threshold
       $K$: Max iteration count
1: Let $X^{(0)} \in \mathcal{D}$.
2: Let $\tau^0 \leftarrow \mathbf{diam}(\mathcal{D})$.
3: for $k = 0..K$ do
4:     $\alpha^{(k)} \leftarrow \frac{2}{k+2}$
5:     $\tau^{(k)} \leftarrow \texttt{update\_tau}(k, \tau^{(k-1)}, \epsilon, f)$
6:     $(\xi^{(k)}, b^{(k)}) \leftarrow \arg\min_{\xi, b} \max_{Y \in \bar{\mathcal{B}}_\infty(X^{(k)}, \tau^{(k)})} |f(Y) - b - \langle Y - X, \xi \rangle|$
7:     $S^{(k)} \leftarrow \arg\min_{S} b^{(k)} + \langle S - X^{(k)}, \xi_k \rangle$
8:     $X^{(k+1)} \leftarrow X^{(k)} + \alpha^{(k)}(S^{(k)} - X^{(k)})$.
9: end for
```

Definition 4.4.2. *Let f be a function that satisfies Assumption 4.3.2 and 4.4.1. For a given $\tau > 0$, we define the* **uniform slope function**, *$m_{ij}(X_{ij}, \tau) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$, for each i, j, which is the slope of the uniform affine approximation for f_{ij} on the interval $[X_{ij} - \tau, X_{ij} + \tau]$.*

The uniform slope function can be viewed as a surrogate for the gradient which does not rely on differentiability of f.

Before establishing the convergence results in Theorem 4.4.5, we require the following lemmas.

Lemma 4.4.3. *Let f be a function that satisfies Assumption 4.3.2 with convex f_{ij} and let $m_{ij}(X_{ij}, \tau)$ be the corresponding uniform slope function for f_{ij}. Then*

$$m_{ij}(X_{ij}, \tau) = g_{ij}$$

for some $c \in (X_{ij} - \tau, X_{ij} + \tau)$ where $g_{ij} \in \partial f_{ij}(c)$.

Proof. From Theorem 4.3.5, the slope function has the form,

$$m_{ij}(X_{ij}, \tau) = \frac{f_{ij}(X_{ij} + \tau) - f_{ij}(X_{ij} - \tau)}{2\tau} \tag{4.12}$$

which is simply the slope of the secant line of f_{ij} from $X_{ij} - \tau$ to $X_{ij} + \tau$. Thus, the desired result follows immediately from the convex mean value theorem. □

Lemma 4.4.4. *Let f be a function that satisfies Assumption 4.3.2 and 4.4.1, $m_{ij}(x,\tau)$ be the corresponding uniform slope function for f_{ij}, and $a \leq b$. We have that,*

$$\int_a^b |m_{ij}(x,\tau) - f'_{ij}(x)|dx \leq (b-a) \max_{\substack{y,z\in(a-\tau,b+\tau)\\ g\in\partial f_{ij}(y)\\ h\in\partial f_{ij}(z)}} |g-h|$$

where $f'_{ij}(x) \in \partial f_{ij}(x)$ can be any subgradient of f_{ij} at x.

If f_{ij} is additionally twice differentiable on $(a-\tau, b+\tau)$, then,

$$\int_a^b |m_{ij}(x,\tau) - f'_{ij}(x)|dx \leq (b-a)\tau \max_{y,z\in(a,b)} \left|\frac{f''(y) - f''(z)}{2}\right|.$$

Proof. Using intermediate value theorem for integrals, there exists $c \in (a,b)$ such that

$$\int_a^b |m_{ij}(x,\tau) - f'_{ij}(x)|dx = (b-a)|m_{ij}(c,\tau) - f'_{ij}(c)|.$$

From Theorem 4.3.5

$$m_{ij}(c,\tau) = \frac{f_{ij}(c+\tau) - f_{ij}(c-\tau)}{2\tau}. \tag{4.13}$$

From Lemma 4.4.3, we have that $m_{ij}(x,\tau) = g_{ij}$ for some $c \in (x-\tau, x+\tau)$ and some $g_{ij} \in \partial f(c)$. Since $m_{ij}(x,\tau)$ and $f'_{ij}(x)$ are just specific subgradients on the evaluated on the interval $(x-\tau, x+\tau)$, we have

$$|m_{ij}(x,\tau) - f'_{ij}(x)| \leq \max_{\substack{y,z\in(a-\tau,b+\tau)\\ g\in\partial f_{ij}(y)\\ h\in\partial f_{ij}(z)}} |g-h|.$$

Following the Lagrange Remainder Theorem, if f_{ij} is twice differentiable in $(c-\tau, c+\tau)$, we have

$$\begin{aligned} f_{ij}(c+\tau) &= f_{ij}(c) + f'_{ij}(c)\tau + \frac{1}{2}f''_{ij}(d_1)\tau^2 \\ f_{ij}(c-\tau) &= f_{ij}(c) - f'_{ij}(c)\tau + \frac{1}{2}f''_{ij}(d_2)\tau^2 \end{aligned} \tag{4.14}$$

for some $d_1 \in (c, c+\tau)$ and $d_2 \in (c-\tau, c)$. Substituting (4.14) into (4.13),

$$\begin{aligned}\frac{f_{ij}(c+\tau) - f_{ij}(c-\tau)}{2\tau} &= \frac{1}{2}\left(\frac{f_{ij}(c+\tau) - f_{ij}(c)}{\tau} - \frac{f_{ij}(c-\tau) - f_{ij}(c)}{\tau}\right) \\ &= \frac{1}{2}\left(f'_{ij}(c) + \frac{1}{2}f''_{ij}(d_1)\tau + f'_{ij}(c) - \frac{1}{2}f''_{ij}(d_2)\tau\right) \\ &= f'_{ij}(c) + \frac{f''_{ij}(d_1) - f''_{ij}(d_2)}{2}\tau\end{aligned}$$

This implies,

$$\begin{aligned}\int_a^b |m_{ij}(x,\tau) - f'_{ij}(x)|dx = (b-a)|m_{ij}(c,\tau) - f'_{ij}(c)| \\ \leq (b-a)\tau\left|\frac{f''_{ij}(d_1) - f''_{ij}(d_2)}{2}\right| \\ \leq (b-a)\tau \max_{y,z\in(a,b)}\left|\frac{f''(y) - f''(z)}{2}\right|\end{aligned}$$

and the result follows. □

Before we establish the convergence of FWUA, we begin with the following theorem.

Theorem 4.4.5. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a function that satisfies Assumption 4.3.2 and 4.4.1 and $\mathcal{D}$ be a convex and compact set. Given $\tau > 0$, let $m_{ij}(X_{ij}, \tau)$ be the uniform slope function for f_{ij} at X_{ij}, and let $X_{\min} = \min\{X_{ij} | X = (X_{1,1}, ..., X_{ij}, ..., X_{mn})^\top \in \mathcal{D}\}$. Let*

$$\hat{f}(X,\tau) := \sum_{i=1}^{m}\sum_{j=1}^{n}\int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(X_{\min})$$

Then the following statements hold:

(a) $\hat{f}(X,\tau)$ is convex in X,

(b) $\nabla_X \hat{f}(X,\tau)$ is Lipschitz continuous with respect to the ℓ_∞-norm with the Lipschitz constant L/τ, where L is the maximum Lipschitz constant of all f_{ij},

(c) The difference between $\hat{f}(X)$ and $f(X)$ is uniformly bounded, i.e.,

$$\max_{X\in\mathcal{D}}\left|\hat{f}(X,\tau) - f(X)\right| \leq mn(M+1)D\Delta_f\tau, \quad \textit{where}$$

$$M := \textit{ the maximum number of points where any } f_{ij} \textit{ is not differentiable for any } i, j.$$

$$D := \mathbf{diam}(\mathcal{D}),$$

$$\Delta_f := \max\Bigg\{ \max_{\substack{i=1,\dots,m\\ j=1,\dots,n\\ X,Y\in\mathcal{D}\\ f_{ij} \text{ twice differentiable at } X_{ij} \text{ and } Y_{ij}}} \left|\frac{f''_{ij}(X_{ij}) - f''_{ij}(Y_{ij})}{2}\right|, \max_{\substack{i=1,\dots,m\\ j=1,\dots,n\\ X_{ij},Y_{ij}\in[X_{\min}-\tau, X_{ij}+\tau]\\ g_{ij}\in\partial f_{ij}(X_{ij})\\ h_{ij}\in\partial f_{ij}(Y_{ij})}} 2|g_{ij}-h_{ij}| \Bigg\} \tag{4.15}$$

Proof. (a) For notational simplicity, we drop the dependency on τ for $\tilde{f}_{ij}$ and m_{ij}.

We establish that $\tilde{f}$ is convex by showing that each $\tilde{f}_{ij}$ is convex. Since $\tilde{f}_{ij}$ is a differentiable function of one variable, $\tilde{f}_{ij}$ is convex if and only if $\tilde{f}'_{ij}$ is nondecreasing in X_{ij}. We have that for any $h > 0$,

$$\begin{aligned}\tilde{f}'_{ij}(X_{ij}+h) - \tilde{f}'_{ij}(X_{ij}) &= m_{ij}(X_{ij}+h) - m_{ij}(X_{ij})\\ &= \frac{f_{ij}(X_{ij}+h+\tau) - f_{ij}(X_{ij}+h-\tau)}{2\tau}\\ &\quad - \frac{f_{ij}(X_{ij}+\tau) - f_{ij}(X_{ij}-\tau)}{2\tau}.\end{aligned}$$

Since f_{ij} is convex, we have that the slope of any secant,

$$S(X_{ij}, Y_{ij}) = \frac{f_{ij}(X_{ij}) - f_{ij}(Y_{ij})}{X_{ij} - Y_{ij}}$$

is nondecreasing in either X_{ij} or Y_{ij} [39].

Thus, $\tilde{f}'_{ij}$ is nondecreasing follows immediately since,

$$\begin{aligned}\frac{f_{ij}(X_{ij}+\tau) - f_{ij}(X_{ij}-\tau)}{2\tau} &\le \frac{f_{ij}(X_{ij}+h+\tau) - f_{ij}(X_{ij}-\tau)}{2\tau+h}\\ &\le \frac{f_{ij}(X_{ij}+h+\tau) - f_{ij}(X_{ij}+h-\tau)}{2\tau}.\end{aligned}$$

(b) From the definition of $\tilde{f}$,

$$\frac{\partial}{\partial X_{ij}} \tilde{f}(X, \tau) = m_{ij}(X_{ij}, \tau). \tag{4.16}$$

To verify the Lipschitz condition, note that from Theorem 4.3.5,

$$m_{ij}(X_{ij}, \tau) = \frac{f_{ij}(X_{ij} + \tau) - f_{ij}(X_{ij} - \tau)}{2\tau}. \tag{4.17}$$

For any $y, z \in \mathbb{R}$ and $\tau > 0$, we have,

$$\begin{aligned} |m(z,\tau) - m(y,\tau)| &= \left| \frac{f_{ij}(z+\tau) - f_{ij}(z-\tau)}{2\tau} - \frac{f_{ij}(y+\tau) - f_{ij}(y-\tau)}{2\tau} \right| \\ &\le \frac{1}{2\tau} |(f_{ij}(z+\tau) - f_{ij}(y+\tau))| + |(f_{ij}(z-\tau) - f_{ij}(y-\tau)| \\ &= \frac{1}{\tau} L_{ij} |z - y| \text{ (since } f_{ij} \text{ is Lipschitz continuous)} \end{aligned} \tag{4.18}$$

Thus,

$$\begin{aligned} \left\| \nabla_X \tilde{f}(Z,\tau) - \nabla_X \tilde{f}(Y,\tau) \right\|_\infty &= \max_{i,j} \; |m_{ij}(Z_{ij},\tau) - m_{ij}(Y_{ij},\tau)| \\ &\le \frac{L}{\tau} \|Z - Y\|_\infty, \end{aligned}$$

where $\|\cdot\|_\infty$ is the component-wise maximum absolute value.

(c) Note we can expand the maximum as follows,

$$\max_{X \in \mathcal{D}} \left| \tilde{f}(X,\tau) - f(X) \right| = \max_{X \in \mathcal{D}} \left| \sum_{i=1}^{m} \sum_{j=1}^{n} \int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau) dx + f_{ij}(X_{\min}) - f_{ij}(X_{ij}) \right|.$$

Suppose f_{ij} is not differentiable only at the points $c_1, c_2, ..., c_{M_i}$. Partition the interval $[X_{\min}, X_{ij}]$ as follows. Let $\mathcal{A}_{ij}$ be a collection of intervals,

$$\begin{aligned} \mathcal{A}_{ij} \coloneqq \{[\alpha_t, \beta_t] : \alpha_t &= \max\{c_t - \tau, c_{t-1} + \tau, X_{\min}\}, \text{ and} \\ \beta_t &= \min\{c_t + \tau, c_{t+1} - \tau, X_{ij}\}, \end{aligned}$$

with $c_0 = X_{\min}$ and $c_{M_i+1} = X_{ij}$. Each interval $[\alpha_t, \beta_t]$ is a neighborhood around a point of nondifferentiability with length at most 2τ. Note that the intervals do not overlap except possibly at the endpoints, and do not extend past the interval $[X_{\min}, X_{ij}]$.

Let $\mathcal{B}_{ij} = \bigcup(b_t, b_{t+1})$ be the minimal set of intervals such that $\mathcal{B}_{ij} = [X_{\min}, X_{ij}] \setminus \mathcal{A}_{ij}$. Thus, $\mathcal{A}_{ij} \cup \mathcal{B}_{ij}$ covers the interval $[X_{\min}, X_{ij}]$ and we have that for every interval in $\mathcal{B}_{ij}$, f_{ij} is differentiable, and hence twice differentiable from our assumptions.

Then we can write $\int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau)dx$ as the sum of integrals over intervals from $\mathcal{A}_{ij}$ and $\mathcal{B}_{ij}$,

$$\int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau)dx = \sum_{(\alpha_t,\beta_t)\in\mathcal{A}_{ij}} \int_{\alpha_t}^{\beta_t} m_{ij}(x,\tau)dx + \sum_{[b_t,b_{t+1}]\in\mathcal{B}_{ij}} \int_{b_t}^{b_{t+1}} m_{ij}(x,\tau)dx$$

If we let $f'_{ij}(X_{ij})$ denote an arbitrary subgradient at X_{ij}, then we can write,

$$f_{ij}(X_{\min}) - f_{ij}(X_{ij}) = -\int_{X_{\min}}^{X_{ij}} f'_{ij}(x)dx$$

since f_{ij} is differentiable everywhere on $[X_{\min}, X_{ij}]$ except on at most a finite set.

Thus,

$$\begin{aligned}
&\max_{X\in\mathcal{D}} \left|\tilde{f}(X,\tau) - f(X)\right| \\
&\le \max_{X\in\mathcal{D}} \left|\sum_{i=1}^{m}\sum_{j=1}^{n} \int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(X_{\min}) - f_{ij}(X_{ij})\right| \\
&\le \max_{X\in\mathcal{D}} \left|\sum_{i=1}^{m}\sum_{j=1}^{n} \int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau)dx - \int_{X_{\min}}^{X_{ij}} f'_{ij}(x)dx\right| \\
&\le \max_{X\in\mathcal{D}} \sum_{i=1}^{m}\sum_{j=1}^{n} \left(\sum_{[\alpha_t,\beta_t]\in\mathcal{A}_{ij}} \int_{\alpha_t}^{\beta_t} |m_{ij}(x,\tau) - f'(x)|dx \right. \\
&\qquad \left. + \sum_{(b_t,b_{t+1})\in\mathcal{B}_{ij}} \int_{b_t}^{b_{t+1}} |m_{ij}(x,\tau) - f'(x)|dx \right)
\end{aligned}$$

Let $|\mathcal{A}_{ij}|$ and $|\mathcal{B}_{ij}|$ denote the number of subintervals for $\mathcal{A}_{ij}$ and $\mathcal{B}_{ij}$ respectively.

Then,

$$\begin{aligned}\max_{X\in\mathcal{D}}\left|\tilde{f}(X,\tau)-f(X)\right| &\le \max_{X\in\mathcal{D}}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(|\mathcal{A}_{ij}|\Delta_f\tau+|\mathcal{B}_{ij}|D\Delta_f\tau\right)\\ &\le mn(M+1)(1+D)\Delta_f\tau\end{aligned}$$

where in the second last line, we use Lemma 4.4.4, and the last line we have that $|\mathcal{A}_{ij}|, |\mathcal{B}_{ij}| \le M+1$ since M is the maximum number of points of nondifferentiability for all f_{ij}.

$\square$

Theorem 4.4.5 states that the sequence of uniform affine approximations generated by the FWUA algorithm corresponds to a sequence of smooth functions that uniformly converges to the original objective if the neighborhood sizes converge to zero. Since the smooth approximation functions in the sequence have a Lipschitz continuous gradient, we can leverage standard Frank-Wolfe convergence arguments even if the original function is not smooth while maintaining an upper bound on the approximation quality of the solution. Suppose the neighborhood size at iteration k is given by $\tau^{(k)}$. In particular, given a sequence of neighborhood sizes $\{\tau^{(k)}\}$, we consider the sequence of smooth approximations given by

$$\hat{f}^{(k)}(X)=\sum_{i=1}^{m}\sum_{j=1}^{n}\int_{X_{\min}}^{X_{ij}}\left(m_{ij}(x,\tau^{(k)})dx+f_{ij}(X_{\min})\right). \tag{4.19}$$

Since $\hat{f}^{(k)}$ is differentiable with a $\frac{L}{\tau^{(k)}}$-Lipschitz gradient, it can be shown that the curvature constant for $\hat{f}^{(k)}$ is bounded, i.e.,

$$C_{\hat{f}^{(k)}} \le \frac{L}{\tau^{(k)}}\mathbf{diam}(\mathcal{D})^2$$

see, e.g., [45].

To make these concepts concrete, consider $f(X)=\|X\|_1$ where we are interested in optimizing f over the trace norm ball, $\|X\|_{\text{tr}}\le\delta$. We have $f(X)=\|X\|_1=\sum_{ij}f_{ij}(X)$, with each $f_{ij}(X_{ij})=|X_{ij}|$. It it is straightforward to verify that $X_{\min}=-\delta$ and,

$$m_{ij}(X_{ij},\tau)=\begin{cases}-1, \text{ when } X_{ij}<-\tau\\ \frac{X_{ij}}{\tau}, \text{ when } -\tau\le X_{ij}\le\tau\\ 1, \text{ when } X_{ij}>\tau.\end{cases}$$

Since the lower bound of each X_{ij} is $-\delta$, we can compute the integrals using the following cases. First consider the case when $X_{ij} < -\tau$. Then,

$$\int_{-\delta}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(-\delta) = -X_{ij} - \delta + \delta = -X_{ij}.$$

When $-\tau \leq X_{ij} \leq \tau$, we have,

$$\begin{aligned}
&\int_{-\delta}^{X_{ij}} m_{ij}(t,\tau)dt + f_{ij}(-\delta)\\
&= \int_{-\delta}^{-\tau} m_{ij}(x,\tau)dx + \int_{-\tau}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(-\delta)\\
&= \frac{X_{ij}^2}{2\tau} + \frac{\tau}{2}.
\end{aligned}$$

Finally, when $X_{ij} > \tau$

$$\begin{aligned}
&\int_{-\delta}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(-\delta)\\
&= \int_{-\delta}^{\tau} m_{ij}(x,\tau)dx + \int_{\tau}^{X_{ij}} m_{ij}(x,\tau)dx + f_{ij}(-\delta)\\
&= X_{ij}.
\end{aligned}$$

Thus, the expression for $\hat{f}(x,\tau)$ is,

$$\hat{f}(x,\tau) = \sum_{ij} \left(\frac{X_{ij}^2}{2\tau} + \frac{\tau}{2} \right) \cdot \mathbb{1}_{|X_{ij}|<\tau} + |X_{ij}| \cdot \mathbb{1}_{|X_{ij}|\geq\tau}$$

where $\mathbb{1}$ is the indicator function. Figure 4.1 illustrates the component functions. Note that the constant, $X_{\min}$, related to the region feasible $\mathcal{D}$, does not appear in the definition of $\hat{f}$ and is only mentioned to show that we can relate $X_{\min}$ with the radius of the trace norm ball. This can give a better idea on the bounds which appear in the convergence result.

When $f(X) = \|X\|_1$, its uniform affine approximation has an attractive property that $\hat{f} \geq f$. In general, this property may not hold. However from the uniform error bound, there always exists some constant $N^{(k)} \in [0, n(M+1)D\Delta_f \tau^{(k)}]$ such that $\hat{f}^{(k)} + N^{(k)} \geq f$, and the function $\hat{f}^{(k)} + N^{(k)}$ has all the properties listed in Theorem 4.4.5, except the error bound in (4.15) becomes $\|\hat{f}^{(k)} - f\|_\infty \leq 2mn(M+1)D\Delta_f \tau^{(k)}$, which is doubled. Thus, we redefine the sequence of approximations as follows.

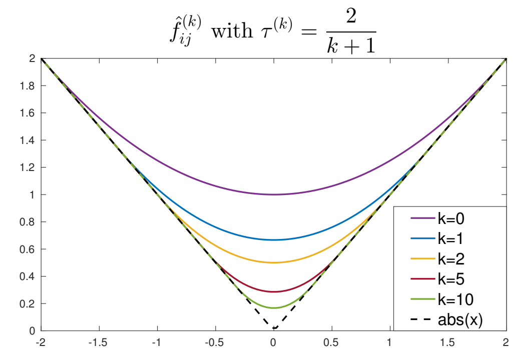


Figure 4.1: The component functions $\hat{f}_{ij}^{(k)}$ with $\tau^{(k)} = 2/(k+2)$ and varying k for $f(X) = \|X\|_1$.

Definition 4.4.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that satisfies Assumptions 4.3.2 and 4.4.1. The sequence of* **FWUA smooth approximations** *are given by*

$$\tilde{f}^{(k)}(X) = \sum_{i=1}^{m}\sum_{j=1}^{n}\int_{X_{\min}}^{X_{ij}} m_{ij}(x,\tau^{(k)})dx + f_{ij}(X_{\min}) + N^{(k)}. \tag{4.20}$$

where $N^{(k)}$ is the smallest nonnegative number such that $\tilde{f}^{(k)} \geq f$.

4.4.1 Update τ

Recall the motivation for the uniform affine approximation is that local approximations are insufficient when the step sizes are large and the neighborhood of points where the next iterate can lie should be taken into account. For the Frank-Wolfe algorithm, the next iterate can be expressed as a convex combination of the current iterate, $X^{(k)}$, and some feasible point $S^{(k)}$, and the Frank-Wolfe steps take the form,

$$X^{(k+1)} = X^{(k)} + \alpha^{(k)}(S^{(k)} - X^{(k)}).$$

Since the uniform affine approximations are taken component-wise, we are interested in bounding the maximum deviation of *any* component between successive iterations. Specifically, we are interested in estimating the quantity,

$$\|X^{(k+1)} - X^{(k)}\|_\infty = \max_{i,j}|X_{ij}^{(k+1)} - X_{ij}^{(k)}|. \tag{4.21}$$

The role of $\tau^{(k)}$ in the algorithm will be to provide an upperbound on the quantity in (4.21). A straightforward upper bound is given by,

$$\|X^{(k+1)} - X^{(k)}\|_\infty \leq \alpha^{(k)} \max_{S \in \mathcal{D}} \|S - X^{(k)}\|_\infty.$$

In particular, when $\mathcal{D} = \bar{\mathcal{B}}_{\mathrm{tr}}(0, \delta)$ is the trace norm ball, then $\|X^{(k+1)} - X^{(k)}\|_\infty \leq 2\alpha^{(k)}\delta$ and a simple updating rule then for $\tau^{(k)}$ is to set,

$$\tau^{(k)} \leftarrow 2\alpha^{(k)}\delta. \tag{4.22}$$

As seen in Theorem 4.4.5, when $\tau^{(k)}$ is large, the smooth surrogate obtained is a poor approximation for the original nonsmooth function. However, if $\tau^{(k)}$ is too small, it is possible that the next iterate $X^{(k+1)}$ lies outside the ball $\mathcal{B}_\infty(X^{(k)}, \tau^{(k)})$, and we are no longer confident in the quality of the linear approximation at $X^{(k+1)}$. We wish to find a value of $\tau^{(k)}$ that is smaller than the conservative upperbound of $2\alpha^{(k)}\delta$ such that $X^{(k+1)}$ is likely to be contained in $\mathcal{B}_\infty(X^{(k+1)}, \tau^{(k)})$.

Empirically, we found that the following update rule based upon the previous Frank-Wolfe steps,

$$\tau^{(k+1)} \leftarrow \alpha^{(k)} \max_{j \in \{0,\ldots,4\}} \|X^{(k-j)} - S^{(k-j)}\|_\infty, \tag{4.23}$$

leads to small values of $\tau^{(k)}$ such that $X^{(k+1)} \in \mathcal{B}_\infty(X^{(k)}, \tau^{(k)})$. This takes the maximum deviation over the past five iterations as an approximation for the neighborhood size.

Another concern with updating $\tau^{(k)}$ is that we cannot allow the function $\tilde{f}^{(k)}$ to become arbitrarily close to f, since the Lipschitz constant for $\tilde{f}^{(k)}$ can also grow arbitrarily large given a nonsmooth f. However, if only an ϵ-accurate solution is desired with ϵ specified a priori, one can stop refining the approximation $\tilde{f}$ at some iteration k' since there exists an explicit upper bound on the approximation error, and the FWUA algorithm can proceed as a standard Frank-Wolfe algorithm on the smoothed function when the iteration $k > k'$.

Specifically, the uniform error bound from Theorem 4.4.5 and a step size of $\alpha^{(k)} = 2/(k+2)$ guarantees that

$$\frac{4mn(M+1)D^2\Delta_f}{k+2} \leq \frac{\epsilon}{2}, \text{ when } k \geq \frac{8mn(M+1)D^2\Delta_f}{\epsilon} - 2.$$

At iteration $k' = \frac{8mn(M+1)D^2\Delta_f}{\epsilon} - 1$, we stop refining neighborhoods and $\tau^{(k)} = \tau^{(k')}$ for all $k \geq k'$. The `update_tau` function is be formalized in Algorithm 7.

Algorithm 7 `update_tau`

```
Input: $k$: Iteration number
    $\tau^{(k-1)}$: Previous neighborhood size
    $\epsilon$: accuracy tolerance
    $f$: Original function to optimize
    $\{X^{(i)}\}_{i=0}^{k-1}$: Sequence of previous Frank-Wolfe iterates
    $\{S^{(i)}\}_{i=0}^{k-1}$: Solutions to the Frank-Wolfe linear subproblems of previous iterates
    $[m, n, M, D, \Delta_f] \leftarrow$ parameters of $f$ as described in (4.15).
1: $k' \leftarrow \frac{8mn(M+1)D^2\Delta_f}{\epsilon} - 1$.
2: if $k > k'$ then
3:     return $\tau^{(k-1)}$
4: else
5:     return $\tau^{(k)} \leftarrow \frac{2\max_{j\in\{0,\dots,4\}}\|X^{(k-j)}-S^{(k-j)}\|_\infty}{k+2}$
6: end if
```

The rationale is that the global error, $f(X^{(k)}) - f(X^*)$, can be decomposed into two components. The first component is the *approximation error*, defined as,

$$\text{approximation error} := \|\tilde{f}^{(k)} - f\| = \max_{X\in\mathcal{D}} |\tilde{f}^{(k)}(X) - f(X)|$$

which is the error added by using a smoothed approximation to f. The second source of error is the *suboptimality at iteration k*, defined as,

$$\text{smoothed optimality gap} := \tilde{f}^{(k)}(X^{(k)}) - \min_{X\in\mathcal{D}} \tilde{f}^{(k)}(X)$$

which is the optimality gap of the smoothed problem,

$$\min_{X\in\mathcal{D}} \tilde{f}^{(k)}(X).$$

Once the approximation is sufficiently accurate, i.e., the approximation error is no bigger than $\epsilon/2$, the neighborhood size refinement can be stopped at the corresponding iteration k'. Since $\tilde{f}^{(k')}$ is smooth, we can minimize $\tilde{f}^{(k)}$ over $\mathcal{D}$ using Frank-Wolfe and use standard Frank-Wolfe analysis to determine the iteration be bounded by $\epsilon/2$. We formalize this statement in Theorem 4.4.8.

Before we prove Theorem 4.4.8, we require the following Lemma.

Lemma 4.4.7. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a function that satisfies Assumption 4.3.2 and 4.4.1, $X^* \in \arg\min_{X\in\mathcal{D}} f(X)$, and $\tilde{f}^{(k)}$ be the FWUA smooth approximations defined in* (4.19) *with* $\tau^{(k)} \leq 2\alpha^{(k)} D$*. Then,*

$$\tilde{f}^{(k)}(X^{(k+1)}) - \tilde{f}^{(k)}(X^{(k)}) \leq \frac{4K}{k+2}$$

where,

$$K \coloneqq \left(\max_{i \in \{0,\dots,k\}} C_{\tilde{f}^{(k)}} \right) + 8mn(M+1)D^2 \Delta_f,$$

where M, D, Δ_f *are the constants defined in* (4.15)*.*

Proof. For any two functions f and g that are defined on $\mathcal{D}$, let $\|f-g\|_\infty = \max_{X \in \mathcal{D}} |f(X) - g(X)|$. From Lemma 2.8.3, we have that,

$$\tilde{f}^{(k)}(X^{(k+1)}) \leq \tilde{f}^{(k)}(X^{(k)}) - \alpha^{(k)}(\tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_k^*)) + (\alpha^{(k)})^2 C_{\tilde{f}^{(k)}}. \tag{4.24}$$

From here, since,

$$\|f^{(k+1)} - f^{(k)}\|_\infty \leq \|f^{(k+1)} - f\|_\infty + \|f^{(k} - f\|_\infty \leq 2\|f^{(k)} - f\|_\infty,$$

we have,

$$\tilde{f}^{(k+1)}(X^{(k+1)}) - 2\|\tilde{f}^{(k)} - f\|_\infty \leq \tilde{f}^{(k)}(X^{(k+1)}).$$

Substituting this into (4.24) we get,

$$\begin{aligned}
&\tilde{f}^{(k+1)}(X^{(k+1)}) - \tilde{f}^{(k+1)}(X_{k+1}^*) \\
&\leq (1-\alpha^{(k)})(\tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_k^*)) + 4\|\tilde{f}^{(k)} - f\|_\infty + (\alpha^{(k)})^2 C_{\tilde{f}^{(k)}} \\
&\leq (1-\alpha^{(k)})(\tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_k^*)) + (\alpha^{(k)})^2 K
\end{aligned}$$

where the last line uses the fact that $\|\tilde{f}^{(k)} - f\|_\infty \leq mn(M+1)D\Delta_f \tau^{(k)}$ and $\tau^{(k)} \leq 2\alpha^{(k)} D$.

The remainder of the proof follows Lemma 2.8.4 since the sequence $\{\tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_k^*)\}$ satisfies (2.13). □

Theorem 4.4.8. *Let* $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ *be a function that satisfies Assumption 4.3.2 and 4.4.1,* $X^* \in \arg\min_{X \in \mathcal{D}} f(X)$*, and let* M, D, Δ_f *be the constants defined in* (4.15)*. Then for any* $\epsilon > 0$*, the iterates* $\{X^{(k)}\}$ *of Algorithm 6, using* $\alpha^{(k)} = 2/(k+2)$ *and* $\tau^{(k)} \leq \alpha^{(k)} D$*, satisfy*

$$f(X^{(k)}) - f(X^*) < \epsilon \text{ when } k \geq \frac{8K}{\epsilon}$$

with

$$k' = \left\lceil \frac{8mn(M+1)D^2\Delta_f}{\epsilon} - 1 \right\rceil \text{ and } K \coloneqq \left(\max_{i \in \{0,\dots,k'\}} C_{\tilde{f}^{(i)}} \right) + 8mn(M+1)D^2\Delta_f.$$

Proof. Let $X_k^* \in \arg\min_{X \in \mathcal{D}} \tilde{f}^{(k)}(X)$ and for any two functions f and g that are defined on $\mathcal{D}$, let $\|f - g\|_\infty = \max_{X \in \mathcal{D}} |f(X) - g(X)|$.

We have,

$$\begin{aligned} f(X^{(k)}) - f(X^*) &\leq \tilde{f}^{(k)}(X^{(k)}) - f(X^*) \\ &\leq \tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X^*) + \|\tilde{f}^{(k)} - f\|_\infty \\ &\leq \tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_k^*) + \|\tilde{f}^{(k)} - f\|_\infty \end{aligned}$$

For all $k \geq k'$, we have that $\tilde{f}^{(k)} = \tilde{f}^{(k')}$, and it follows that

$$f(X^{(k)}) - f(X^*) \leq \tilde{f}^{(k')}(X^{(k)}) - \tilde{f}^{(k')}(X_k^*) + \|\tilde{f}^{(k')} - f\|_\infty.$$

Theorem 4.4.5 implies that,

$$\|\tilde{f}^{(k)} - f\|_\infty \leq \frac{4mn(M+1)D^2\Delta_f}{k+2} \leq \frac{\epsilon}{2}, \text{ when } k > k'$$

Thus, when $k \geq k'$,

$$\|\tilde{f}^{(k)} - f\|_\infty = \|\tilde{f}^{(k')} - f\|_\infty \leq \frac{\epsilon}{2}. \tag{4.25}$$

When $k \leq k'$, it follows from Lemma 4.4.7 that

$$\tilde{f}^{(k)}(X^{(k+1)}) - \tilde{f}^{(k)}(X^{(k)}) \leq \frac{4K}{k+2}. \tag{4.26}$$

When $k \geq k'$, we have that $\tilde{f}^{(k)} = \tilde{f}^{(k')}$. We will show that

$$\tilde{f}^{(k')}(X^{(k+1)}) - \tilde{f}^{(k')}(X^{(k)}) \leq \frac{4K}{k+2}, \ \forall k \geq k'. \tag{4.27}$$

We proceed with induction, where (4.26) provides the base case when $k = k'$. We assume (4.27) holds for some $k > k'$. From Lemma 2.8.3, we have that,

$$\tilde{f}^{(k')}(X^{(k+1)}) \leq \tilde{f}^{(k')}(X^{(k)}) - \alpha^{(k)}(\tilde{f}^{(k')}(X^{(k)}) - \tilde{f}^{(k')}(X_k^*)) + (\alpha^{(k)})^2 C_{\tilde{f}^{(k')}}.$$

This implies that,

$$\tilde{f}^{(k')}(X^{(k+1)}) - \tilde{f}^{(k')}(X_{k'}^*) \leq (1 - \alpha^{(k)})(\tilde{f}^{(k')}(X^{(k)}) - \tilde{f}^{(k')}(X^{(k)})) + (\alpha^{(k)})^2 K \tag{4.28}$$

since $K \geq C_{\tilde{f}^{(k')}}$. The desired result then follows from Lemma 2.8.4.

Thus,

$$\frac{4K}{k+2} \leq \frac{\epsilon}{2} \text{ whenever } k \geq \frac{8K}{\epsilon} - 2. \tag{4.29}$$

Since at most $\frac{8K}{\epsilon}$ iterations are required to guarantee that $\tilde{f}^{(k)}(X^{(k)}) - \tilde{f}^{(k)}(X_{(k)}^*) \leq \frac{\epsilon}{2}$, and $8K/\epsilon \geq k'$, we get the desired bound by combining (4.25) and (4.29). $\square$

4.5 Experimental Results

4.5.1 Sparse and Low-Rank Structure

To highlight benefits of the proposed FWUA, we first compare it against other state-of-the-art solvers for the problem,

$$\min_{X:\|X\|_{\mathbf{tr}}\leq\delta}\|P_\Omega(X-Y)\|_F^2+\lambda_1\|X\|_1.$$

where Y is the given data, $\Omega=\{(i,j)\}$ is the set of observed indices, and $P_\Omega(\cdot)$ projects the loss onto Ω. In [69], the above formulation has been shown to yield empirical improvements over low-rank or sparse regularization alone, for the sparse covariance matrix estimation and graph link prediction.

We compare FWUA with the following methods:

1. The Generalized Forwards-Backwards algorithm (GenFB) [69]. This variant applies proximal steps in a sequential fashion for the ℓ_1 and trace norm.
2. Hybrid Conditional Gradient with Smoothing (HCGS) [3]. Uses Nesterov Smoothing [58] to find $(1/\beta^{(k)})$-smooth surrogates for the objective function where $\{\beta^{(k)}\}$ is an arbitrary sequence that goes to 0.
3. Smoothed Composite Conditional Gradient (SCCG) [65]. Similar to HCGS with fixed smoothing parameter μ for all iterations (instead of $\beta^{(k)}$ schedule).

For all experiments, we terminate any algorithm when it fails to make sufficient progress. Explicitly, we terminate the at iteration k if,

$$\frac{f_{\max}-f_{\min}}{f_{\min}}<10^{-4}$$

where,

$$f_{\max}\coloneqq\max_{i\in\{0,\ldots,9\}}f(X^{(k-i)})\text{ and}$$
$$f_{\min}\coloneqq\min_{i\in\{0,\ldots,9\}}f(X^{(k-i)}).$$

That is, when the relative difference between the maximum and minimum objective values over the past ten iterations becomes sufficiently small, the algorithm terminates. For the

FW based methods (HCGS, SCCG, FWUA), we set the maximum iteration count 1000. For the GenFB algorithm the maximum iteration count is set to 100.

For each problem instance, the same λ_1 value is used by all three methods and this value is tuned, by searching over a grid of parameter values, to yield the best test performance for GenFB. The bound δ for the trace norm, used in all FW variants, is then set to the trace norm of the solution given by GenFB. For SCCG, the smoothing parameter μ is additionally tuned to yield the smallest average objective value. HCGS sets $\beta^{(k)} = 1/\sqrt{k+1}$ as suggested by the authors [3]. Additionally, we compare the limiting behavior for SCCG where $\mu = 0$. This corresponds to a specific subgradient, denoted as SCCG (SG).

Sparse Covariance Estimation

We follow the synthetic experiments described in [69], where the goal is to recover a block diagonal matrix. We consider square matrices where $n = 750 : 250 : 2000$ (here we use MATLAB notation). The true underlying matrix is generated with 5 blocks, where the entries are i.i.d. and uniformly sampled from $[-1, 1]$. Gaussian noise, $\mathcal{N}(0, \sigma^2)$ is then added with $\sigma^2 = 0.2$. For this experiment, all entries are observed.

In Figure 4.2, we remark that since the GenFB algorithm is a regularized algorithm, the intermediate iterates are not feasible for the constrained problem used for Frank-Wolfe. Consequently, only the performance of the solution at convergence is compared.

Graph Link Prediction

Next we consider predicting links in a noisy social graph. The input data is a matrix corresponding to an undirected graph, where the entry $A_{ij} = 1$ indicates that user i and j are friends and $A_{ij} = 0$ otherwise. We consider the Facebook dataset from [53] which consists of a graph with 4,039 nodes and 88,234 edges, and assume 50% of the entries are observed. The goal is to recover the remaining edges in the graph. Additionally, each entry A_{ij} is flipped (0 to 1 or 1 to 0) with probability $\sigma \in \{0, 0.05, 0.1\}$, potentially removing or adding labels to the graph. We report the AUC performance measure of the link prediction on the remaining entries of the graph as well as the average CPU time over 5 random initializations summarized in Table 4.3 across all levels of σ.

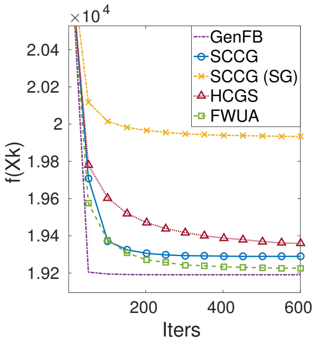
Parameter	Methods	Description	n	Value
λ_1	All methods	Parameter for ℓ_1 norm	all	0.4
λ_2	GenFB	Parameter for trace norm	all	10
δ	FW Variants	Trace norm constraint	750	43.92
			1000	73.17
			1250	100.43
			1500	134.06
			1750	169.40
			2000	201.46
μ	SCCG	Smoothing parameter	all	0.01

Table 4.1: Parameters used for sparse covariance estimation

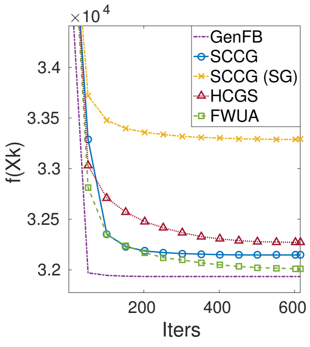
Discussion

For both applications, the results agree with our initial intuition that FWUA can improve the performance of the FW variants while scaling much better than the GenFB algorithm. We observe in the covariance plots in Figure 4.3, the sparsity patterns for HCGS and SCCG are much noisier than FWUA and in Table 4.3, the AUC for SCCG and HCGS methods are lower than GenFB and FWUA.

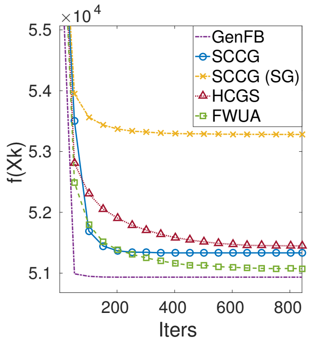
For SCCG, the smoothing parameter μ is closely related to the convergence rate as well as the approximation accuracy of the smooth surrogate. Large values of μ will lead to a better constant in the convergence rate, but this introduces large approximation errors. The initial expectation was that this tradeoff to be smooth, where gradually decreasing μ should gradually worsen the rate of improvement per iteration. In return, small values of μ should yield better final solutions. Instead, we observe an interesting phenomenon for SCCG, where the algorithm makes no progress for many iterations when μ is too small. This delay is highlighted in Figure 4.5. We see that the delay observed increases as μ decreases. This seems to give support to the hypothesis for FWUA where the approximation quality of the smooth surrogate must be closely related to the step size since it appears that SCCG can only make progress using small values of μ once the step sizes become sufficiently small. Since HCGS and SCCG do not factor in step size into the smoothing schedule, we observe that empirically, the solutions returned from these methods are not as competitive as the



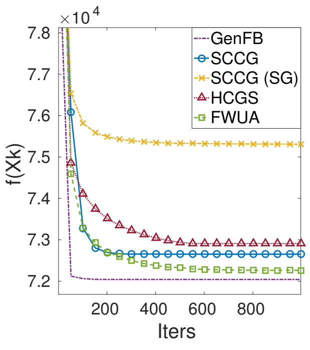
(a) $n = 750$



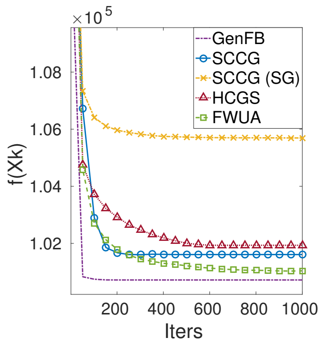
(b) $n = 1000$



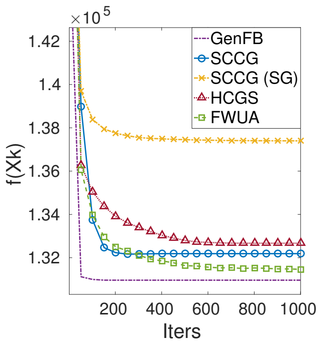
(c) $n = 1250$



(d) $n = 1500$



(e) $n = 1750$



(f) $n = 2000$

Figure 4.2: Objective values vs iteration for various synthetic dataset sizes.

FWUA or GenFB algorithm.

4.5.2 ℓ_1 Loss Matrix Completion

The last experiment we consider is matrix completion with an ℓ_1 loss function on the MovieLens datasets[2]. Here, we consider the objective function below

$$f(X) = \|P_\Omega(X - Y)\|_1 + \lambda\|P_{\Omega^c}(X)\|_F^2 \tag{4.30}$$

which is proposed in [11] for robustness to outliers. Here the regularization penalizes entries in the complement of Ω, potentially preventing overfitting.

We compare with the Robust Low-Rank Matrix Completion (RLRMC) algorithm proposed in [11], which solves a nonconvex fixed rank problem by the smoothing ℓ_1 term. We

[2]https://grouplens.org/datasets/movielens/

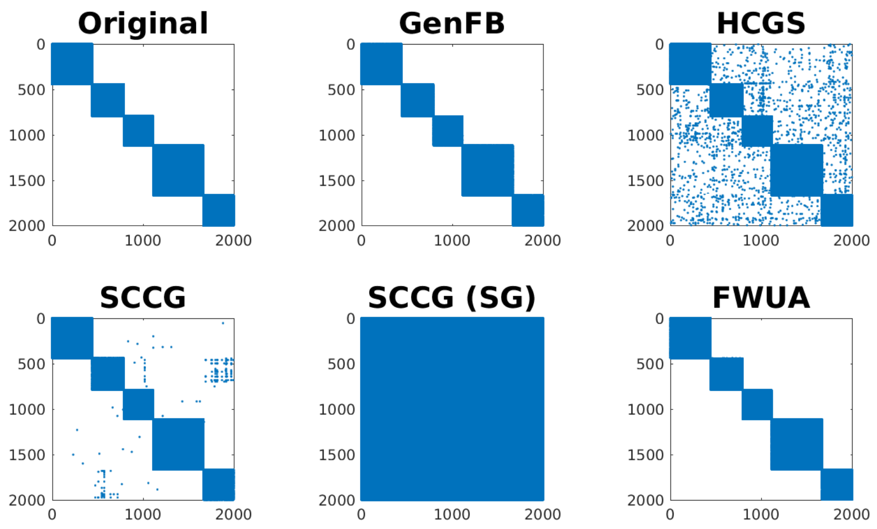


Figure 4.3: An example sparsity pattern at termination, values thresholded at $0.01\|X\|_\infty$.

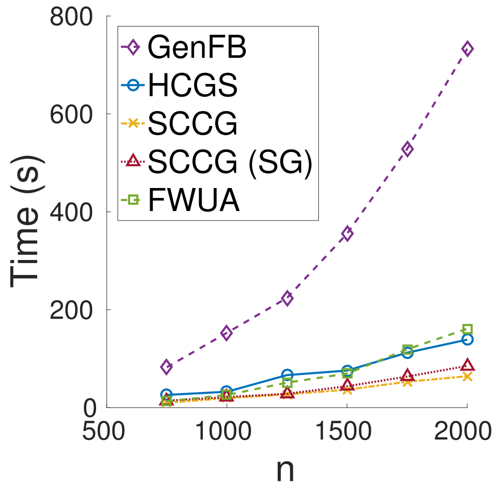


Figure 4.4: CPU time for varying n on the sparse covariance estimation problem.

additionally compare to the Greedy Low-Rank Learning (GLRL) algorithm proposed in [83], which greedily updates the solution with low-rank solutions found by computing the truncated SVD of a subgradient. For scalability, we utilize the rank-drop step subproblems proposed Chapter 3 for the FWUA algorithm to reduce the rank of the intermediate solutions. The gradient of the smoothed objective, $\nabla \tilde{f}^{(k)}$, is used to compute rank-drop steps.

Parameter	Methods	σ	Description	Value
λ_1	All methods	0	Parameter for ℓ_1	0.01
		0.05		0.05
		0.01		0.1
λ_2	GenFB	0	Parameter for trace norm	5
		0.05		25
		0.01		50
δ	FW Variants	0	Trace norm constraint	1052.88
		0.05		590.13
		0.01		639.87
μ	SCCG	0 and 0.1	Smoothing parameter	0.001
		0.05		0.01

Table 4.2: Parameters used for graph link prediction

σ		GenFB	SCCG	SCCG (SG)	HCGS	FWUA
0	AUC	0.968	0.949	0.873	0.868	**0.972**
	Time (s)	3746.30	1127.91	288.46	**263.27**	449.89
0.05	AUC	**0.829**	0.820	0.799	0.806	**0.829**
	Time (s)	4024.96	**332.23**	378.68	407.87	408.37
0.10	AUC	0.715	0.708	0.707	0.708	**0.732**
	Time (s)	4006.92	**363.18**	403.27	420.22	481.82

Table 4.3: Graph link prediction averaged over 5 random initializations for the Facebook dataset. Out of sample AUC and CPU-time are reported. Best performers are bolded.

Since the rank-drop steps are only taken if the objective does not worsen, convergence is still guaranteed as long as the k used in the step-size $2/(k+2)$, only counts the number of standard Frank-Wolfe steps taken, i.e., the iteration count does not increase when a rank-drop step is used. The parameters were chosen using a grid search and we report the best results with respect to the smallest RMSE.

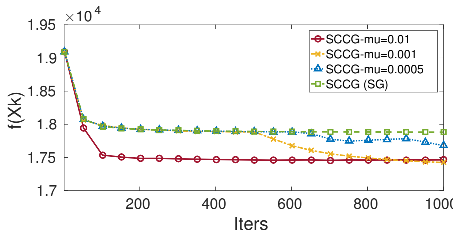


Figure 4.5: An example of the progress with varying μ for sparse covariance estimation with $n = 750$.

Dataset	Parameter	Description	Methods	Value
MovieLens 100k	λ	Parameter for $\|P_{\Omega^c}(X)\|_2^2$	RLRMC	0.001
	r	Fixed rank	RLRMC	2
			GLRL	10
	δ	Trace norm constraint	FWUA	1,200
MovieLens 1M	λ	Parameter for $\|P_{\Omega^c}(X)\|_2^2$	RLRMC	0.001
	r	Fixed rank	RLRMC	5
			GLRL	25
	δ	Trace norm constraint	FWUA	6,400
MovieLens 10M	λ	Parameter for $\|P_{\Omega^c}(X)\|_2^2$	RLRMC	0.005
	r	Fixed rank	RLRMC	10
			GLRL	50
	δ	Trace norm constraint	FWUA	15,000

Table 4.4: Parameters used for matrix completion

Discussion

We observe that FWUA performs better than both RLRMC and GLRL in terms of out of sample RMSE, but RLRMC is much faster. This is not surprising since RLRMC is a nonconvex fixed rank model. We observe that the performance of RLRMC is very sensitive to both the rank and λ parameters, requiring extensive parameter tuning to find reasonable results. Even a small change in rank (e.g. ± 2) can make a dramatic difference in the RMSE. Thus, if we account for the exhaustive parameter tuning required for RLRMC, in practice, the computational time required for the entire model evaluation can far exceed the cost of

Dataset		RLRMC	GLRL	FWUA
ML-100k	RMSE	0.892	0.935	**0.876**
	Time (s)	**10.62**	43.56	65.39
ML-1M	RMSE	0.817	0.917	**0.812**
	Time (s)	**108.97**	1,079.11	862.15
ML-10M	RMSE	0.810	0.901	**0.801**
	Time (s)	**2,197.20**	26,473.89	6,830.31

Table 4.5: Low-rank Matrix Completion averaged over 5 random initializations. Best performers are bolded.

FWUA, where warm starts can be leveraged for tuning the radius of the trace-norm ball. Thus, FWUA can be an attractive alternative when a good estimate of the true rank is not known a priori. We also note that for the large scale example, the number of singular values required in the truncated SVD used by the GLRL updates became very high, leading to scalability issues.

4.6 Conclusion

We propose a variant of the Frank-Wolfe algorithm for a nonsmooth objective, by replacing the first order Taylor approximation with the Chebyshev uniform affine approximation in the FW subproblem. We show that, for nonsmooth matrix estimation problems, this uniform approximation is easy to compute and allows for convergence analysis without assuming a bounded curvature constant. Experimentally we demonstrate that the FWUA algorithm can improve both speed and classification performance in a variety of sparse and low-rank learning tasks, while providing a viable convex alternative for ℓ_1 loss matrix completion when little is known about the underlying data.

Chapter 5

STAR-SVM

5.1 Introduction

A common theme in this thesis is utilizing structure to best leverage small amounts of labeled data. The key idea has been to assign the missing labels that are most consistent with the structural assumptions that we make about the data. For example for the matrix completion task for movie recommendations, we are given a matrix with very few labeled entries and most reviews are unknown. To infer the missing labels, the structural assumption made is that the underlying matrix is low-rank and the optimization becomes finding a low-rank matrix that is consistent with the given labels.

For many applications, understanding the structure of the data is nontrivial and cannot be as easily expressed as the low-rank constraint. It becomes more difficult to define an appropriate regularization measure to capture the structure in the data . In this chapter, we will focus on binary classification problems where the structural assumptions are not simple to encode. The idea we investigate will be to gradually uncover the structure through self-training and utilize support vector machines to determine confidence. We will show empirically that this iterative procedure allows for more accurate labels, particularly in the imbalanced class setting. Moreover, we show that this iterative procedure allows for leveraging warm starts very efficiently, scaling better than many existing SSL algorithms.

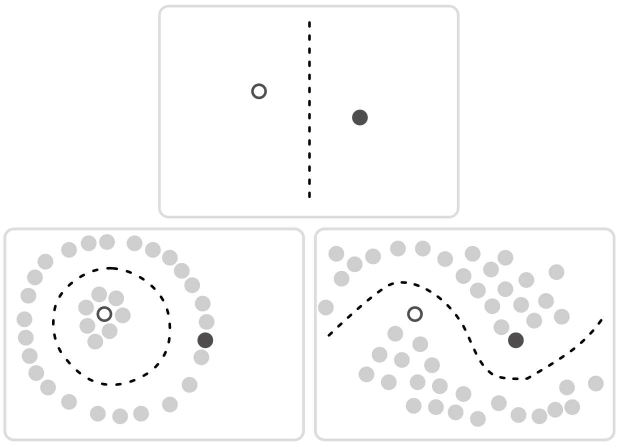


Figure 5.1: The structure given from the unlabeled data greatly helps creating a classifier that generalizes better to unseen data. In the bottom row, we see that different sets of unlabeled data (corresponding to the same labeled dataset in the top row) can shape the classifier in very different ways.

5.2 Background

5.2.1 Support Vector Machines

We begin with a brief overview of Support Vector Machines (SVM). The problems we consider are *binary linear classification problems*, in which we are given a set of feature vectors, $\{x_i\}_{i=1}^n$ where each $x_i \in \mathbb{R}^m$, and a label vector, $y \in \{-1, 1\}^n$, where y_i is the label corresponding to the feature vector x_i. The problem is to find a hyperplane that separates the positive class from the negative class.

The motivation for SVMs relies on the concept of a *margin*.

Definition 5.2.1. *Consider a hyperplane defined by the weights $w \in \mathbb{R}^n$ and a bias term $b \in \mathbb{R}$. Then the* **functional margin with respect to a** **training point** *$(x_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$, denoted as $p(x_i, y_i, w, b)$, is*

$$p(x_i, y_i, w, b) = y_i(w^\top x_i + b).$$

Similarly, the **functional margin with respect to a** **training set** *$\mathcal{S} = \{(x_i, y_i) : i = 1, ...m\}$ is defined as,*

$$p(\mathcal{S}, w, b) = \min_{(x_i,y_i)\in\mathcal{S}} p(x_i, y_i, w, b).$$

If we use $\mathbf{sign}(w^\top x_i + b)$ to assign the label to x_i, then the functional margin will be positive if the point is classified correctly and negative is classified incorrectly. Moreover, we can interpret points that have a large and positive functional margin to be labelings that we are confident in since these points are far from the decision boundary. However, by substituting w with $2w$ and b with $2b$, we see that this corresponds to the same hyperplane but the functional margins have all been doubled. Thus, it may be more useful to consider a normalized version of the functional margin.

Definition 5.2.2. *The* **geometric margin with respect to a training point** $(x_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$*, denoted as* $q(x_i, y_i, w, b)$*, is*

$$q(x_i, y_i, w, b) = y_i \left(\frac{w^\top x_i + b}{\|w\|_2} \right).$$

Analogously, the **geometric margin with respect to a training set** $\mathcal{S} = \{(x_i, y_i) : i = 1, ...m\}$ *is defined as,*

$$q(\mathcal{S}, w, b) = \min_{(x_i, y_i) \in \mathcal{S}} q(x_i, y_i, w, b).$$

The linear SVM problem is to find the separating hyperplane with the maximum geometric margin [9, 24]. This hyperplane can be found by solving the following optimization problem,

$$\begin{aligned} &\max_{w,b,\gamma} \frac{\gamma}{\|w\|_2} \\ &s.t.\ y_i(w^\top x_i + b) \geq \gamma,\ \forall (x_i, y_i) \in \mathcal{S}. \end{aligned} \tag{5.1}$$

The objective maximizes the geometric margin while the constraint ensures that all points are classified correctly and sufficiently far from the hyperplane.

Since the parameter γ measures the functional margin on the training set $\mathcal{S}$, we can simplify (5.1) by normalizing γ. Note that this does not change the problem since w and b can be scaled arbitrarily without affecting the solution. The problem in (5.1) can now be rewritten as,

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|_2^2 \\ &s.t.\ y_i(w^\top x_i + b) \geq 1,\ \forall (x_i, y_i) \in \mathcal{S}. \end{aligned} \tag{5.2}$$

The constraints in (5.2) require all points to be correctly classified. However, most datasets are not linearly separable which would lead to no feasible solutions for (5.1). To handle this case, slack variables $\{\xi_i\}_{i=1}^m$, are introduced to create a *soft margin problem*,

$$\begin{aligned} &\min_{w,b}\ \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^m \xi_i \\ &s.t.\ y_i(w^\top x_i + b) \geq 1 - \xi_i,\ \forall (x_i, y_i) \in \mathcal{S} \\ &\qquad \xi_i \geq 0, i = 1, ..., m. \end{aligned} \tag{5.3}$$

where $C \geq 0$ is a penalty parameter.

While the formulation in (5.3) addresses the feasibility issue, it does not address problems where the data is separable (or nearly separable), but not linearly separable as in Figure 5.1. The key insight is to use *Cover's Theorem*, which states that with high probability, one can map a training set that is not linearly separable to a higher dimensional space where the training set becomes linearly separable [25].

Definition 5.2.3. *A* **feature map** *is a function $\phi : \mathbb{R}^m \to \mathcal{V}$, where $\mathcal{V}$ is a Hilbert space.*

Let M be the dimension of the Hilbert space corresponding to the image of the feature map, ϕ. For SVM, we are typically concerned with $M > m$ to lift the data to a higher dimensional space. As an example, consider Figure 5.2. The data forms a donut shape where the inner circle cannot be linearly separated from the outer circle. However, by considering the radius of each circle, we can easily distinguish the classes. This motivates the following feature map:

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ \|x\|_2 \end{pmatrix}$$

On the right side of Figure 5.2, the data is now linearly separable using the feature map described. We remark that using the radius of the circles alone would have been sufficient to separate such a simple case, but we wish to emphasize that the feature map typically lifts the data to a higher dimension.

Using the feature map in place of the original data transforms the problem as follows,

$$\begin{aligned} &\min_{w,b}\ \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^m \xi_i \\ &s.t.\ y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i,\ \forall (x_i, y_i) \in \mathcal{S} \\ &\qquad \xi_i \geq 0, i = 1, ..., m \end{aligned} \tag{5.4}$$

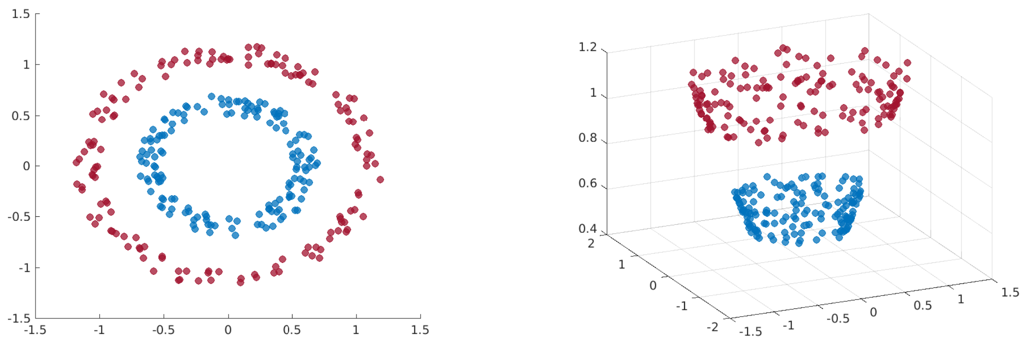


Figure 5.2: On the left, the data is not linearly separable. On the right, the data has been lifted to a higher dimensional space where the data is trivially separable.

The key difference between (5.2) and (5.4) is that the dimension of w is the same as the dimension of $\phi(x)$. Thus, if ϕ maps to a much higher dimensional space $M \gg m$ (possibly infinite dimensional), then (5.4) is much more expensive to solve.

The solution to this problem comes from considering the dual problem. The dual of (5.4), see e.g. [9, 24], is,

$$\begin{aligned} \min_{\alpha} \; & e^\top \alpha - \frac{1}{2}\alpha^\top Y K Y \alpha \\ s.t. \; & 0 \le \alpha_i \le C, i = 1, ..., m \\ & y^\top \alpha = 0 \end{aligned} \tag{5.5}$$

where e is the vector of all ones, $Y = \mathbf{diag}(y)$ is a diagonal matrix where the diagonal entries are the vector y, and,

$$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle.$$

It is important to emphasize that $\alpha \in \mathbb{R}^m$, which is independent of the dimension of the feature map. Moreover, the formulation only requires the inner products of the feature maps, not the feature maps themselves.

Definition 5.2.4. *Given a feature map $\phi : \mathcal{X} \to \mathcal{V}$ where $\mathcal{X} \subseteq \mathbb{R}^m$ and $\mathcal{V}$ is a Hilbert space, a* **kernel** *is a function $k : \mathcal{X} \times \mathcal{X} \to \mathcal{X} \to \mathbb{R}$ which satisfies,*

$$k(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{V}}.$$

For a finite set $X = \{x_1, ..., x_m\}$ where each $x_i \in \mathbb{R}^n$, the associated matrix,

$$K = [k(x_i, x_j)]_{i,j=1}^m$$

is called the **kernel matrix** *or* **Gram matrix**.

If we can evaluate the kernel function efficiently, we can implicitly solve the SVM problem in a very high dimensional feature space without explicitly computing the features in the high dimensional space. This is commonly referred to as the "kernel trick" [34]. In practice, a well known kernel function is typically chosen instead of designing a feature map and computing the kernel afterwards. A common choice we will use is the *Gaussian kernel* defined by the following kernel function,

$$k(x, y) = \mathbf{exp}\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) \tag{5.6}$$

where σ is a parameter.

5.2.2 S^{3}VM

The **S**emi-**S**upervised **S**upport **V**ector **M**achine (S^{3}VM) problem formulates a binary classification problem where the training set is only partially labeled. Explicitly, we are given l labeled points $\{x_i, y_i\}_{i=1}^{\ell}, y_i = \pm 1$, and u unlabeled points, $\{x_i\}_{i=l+1}^{m}$, with $m = l + u$. We will use the sets $\mathcal{L}$ and $\mathcal{U}$ to denote the indices for labeled and unlabeled sets respectively. A natural way to incorporate the unknown labels is to treat them as additional optimization variables. The following optimization problem is then solved to obtain the optimal hyperplane parameters (w, b), as well as the binary labels for the unlabeled set, $y_{\mathcal{U}} = [y_{l+1}, ..., y_m]^\top$,

$$\begin{aligned}
\min_{y_{\mathcal{U}}} \min_{w,b} \;\; & P(w, b, y_{\mathcal{U}}) = \frac{1}{2} w^T w + C \sum_{i \in \mathcal{L}} \xi_i + C^* \sum_{i \in \mathcal{U}} \xi_i \\
s.t. \;\; & y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i, \;\; i = 1, ..., m \\
& \xi_i \geq 0 \\
& y_i \in \{-1, 1\}
\end{aligned} \tag{5.7}$$

where C and C^* are nonnegative regularization parameters for the labeled and unlabeled sets respectively, and ϕ is a suitably chosen feature map. For many approaches, the inner optimization in (5.7) is replaced with its dual as follows,

$$\begin{aligned} \min_{y_{\mathcal{U}}} \max_{\alpha} \; & D(\alpha, y_{\mathcal{U}}) = e^T\alpha - \frac{1}{2}\alpha^T YKY\alpha \\ s.t. \; & y^T\alpha = 0 \\ & 0 \leq \alpha_i \leq C, \; i \in \mathcal{L} \\ & 0 \leq \alpha_j \leq C^*, \; j \in \mathcal{U} \end{aligned} \tag{5.8}$$

where $Y = diag(y)$ and e is the all-ones vector. The resulting problem can be interpreted as finding the maximum margin across all possible labelings. This encourages the algorithm to find a labeling such that the decision boundary passes through a low-density region [13]. The combinatorial nature of $y_{\mathcal{U}}$ assignment makes solving either optimization problem extremely challenging.

5.3 Solving the S^{3}VM Problem

Although S^{3}VM is a natural mathematical formulation to incorporate unlabeled information, the optimization problem becomes a difficult Mixed-Integer Problem (MIP). Thus S^{3}VM loses the desirable convex and continuous optimization properties from the original SVM problem and efficient optimization becomes difficult. A variety of approaches have been attempted, including local combinatorial searches [47], branch and bound techniques [14], semidefinite programming (SDP) [26], concave-convex procedures [23], and convex relaxations [54]. A full survey of the techniques can be found in [15].

The branch-and-bound (BB) approach proposed in [14] shows that the global optimum found by the BB approach at times achieved strong generalization performance, even when the traditional S^{3}VM based methods typically struggle. This suggests that existing relaxation methods may not be a sufficiently accurate approximation to the original problem, and better approximations may lead to better generalization performance.

Instead of using full branch-and-bound, one can consider using a simplier heuristic such as *self-training*. Self-training is one of the earliest semi-supervised learning (SSL) techniques [13] which uses a supervised algorithm to gradually expand the labeled set. Self-training starts by training solely on labeled data. At each iteration, an assessment is made on the predicted labels, using the current decision function for the unlabeled set. Points that are labeled confidently will be added to the labeled set and the supervised method is retrained on the enlarged labeled set, see e.g. [13]. An issue with self-training is that the intermediate supervised learning steps do not incorporate unlabeled information.

Additionally, errors tend to propagate since each supervised learner assumes the labeled set is correct.

In this chapter, we propose a **S**elf-**T**raining with **A**daptive **R**egularization SVM framework (STAR-SVM), which uses self-training to gradually incorporate unlabeled information. We view this as gradually approximating the original nonconvex optimization problem, by a sequence of convex SVM subproblems, which can be readily solved. Additionally, we will show that the labeled set is grown in a way such that each subproblem can be quickly solved from the warm-start solution provided by the previous subproblem. This allows STAR-SVM to find an approximate solution to the nonconvex optimization problem very quickly. Since self-training can be misled by training errors and noisy datasets, self-training alone will unlikely yield a good solution to the original S^{3}VM problem. To address the shortcomings of self-training, we introduce individual regularization parameter C_i for the loss of each data point which automatically adapts to the degradation in confidence at each iteration. A confidence-weight parameter γ is utilized to control the rate at which the confidence declines. Moreover, this allows for a simple way to incorporate class balance information to also improve the S^{3}VM problem in the highly imbalanced class setting.

5.3.1 Motivation

As noted in [14], the exact solution from the nonconvex optimization problem can lead to significant improvements over existing convex relaxations. The proposed method is motivated by utilizing this idea to efficiently find solutions to the nonconvex S^{3}VM formulation, instead of a convex relaxation.

We attempt to solve the combinatorial S^{3}VM optimization problem by successively approximating the problem with convex SVM subproblems using only partial labels. The idea is to gradually grow the labeled set with self-training. Self-training for S^{3}VM problems has been proposed before in [29, 35, 55]. However, none of these methods address a natural shortcoming of self-training, in which the errors tend to propagate, since each iteration assumes the labeled set is totally correct. To address these issues, we introduce the idea of *adaptive regularization* into the self-training framework.

For most S^{3}VM frameworks, there are separate regularization parameters, C and C^*, that bound the optimization variables for the labeled and unlabeled examples respectively. It is suggested to choose C^* to be smaller than C (usually $C^* = 0.1C$) to reflect that we are less confident in the unlabeled examples, [15]. Choosing a smaller constant penalizes the errors for the unlabeled set less, but also allow for less contribution to the decision hyperplane due to the constraints $0 \leq \alpha_i \leq C^*$ in (5.8). Ideally, if we knew beforehand the confidence in each label, we could assign an individual C_i to each example to best reflect the label confidence and better formulate the problem. Utilizing this idea for an S^{3}VM problem is difficult because the labels are unknown, and thus, confidence measures are also unknown. However, when self-training is used, the confidence in the labeling is indicated by the iteration when the example is labeled. The idea is that earlier labels are more reliable than later labels due to error propagation. To reflect the confidence degrading at each iteration, we introduce a confidence weighting parameter, $\gamma \in [0, 1]$. Thus, when example x_i is labeled at iteration k, we adjust the regularization parameter C_i as follows,

$$C_i = \gamma^k C. \tag{5.9}$$

By limiting the contributions of examples we are less confident in, the effect of error propagation will be lessened.

Ignoring the Offset

The work in [71] has shown that, when using kernels with a large feature space, such as the Gaussian RBF kernel, utilizing the offset term, b, does not improve generalization performance for classification problems. Moreover, the effect of removing the offset term is that the optimization problem (5.8) does not have the equality constraint, allowing for simpler solvers such as the one in [72]. For the remainder of the chapter, we will refer to this particular SVM solver as *Training Without Offset SVM* (TWO-SVM).

We give a brief overview of the TWO-SVM method proposed in [72]. Many SVM solvers, such as LibSVM [12], utilize *Sequential Minimization Optimization* (SMO) [66] to efficiently solve the optimization problem. To preserve the equality constraint, the SMO algorithm iteratively chooses pairs of variables, α_i and α_j with $i \neq j$, and optimizes the objective value jointly over these variables called the *working set.* The process is repeated until all pairs of variables are optimal. Without offset, the dual problem no longer has an equality constraint linking the variables α_i. Thus, the algorithm can be simplified by iteratively solving over working sets of size one, that is considering each α_i one at a time. This simplifies the training process allowing for a greedy component-wise gradient descent. At each iteration, an index i^* is identified which achieves the greatest improvement in the dual objective value. Let $\ell(x_i, y_i, \alpha)$ be the hinge-loss function

$$\ell(x_i, y_i, \alpha) := \max\left\{0, 1 - y_i \sum_{j \in \mathcal{L}_k} K(x_i, x_j) y_j \alpha_j\right\}. \tag{5.10}$$

The procedure is continued until the duality gap,

$$\begin{aligned} \text{gap}(\alpha) = \alpha^T YKY\alpha - e^T\alpha + \sum_{i\in\mathcal{L}_k} C_i \ell(x_i, y_i, \alpha) \\ + \sum_{i\in\mathcal{U}_k} C_i^* \ell(x_i, y_i, \alpha) \end{aligned} \tag{5.11}$$

becomes sufficiently small.

In this section, we will further illustrate that utilizing the offset term is not suitable for iterative semi-supervised learning procedures.

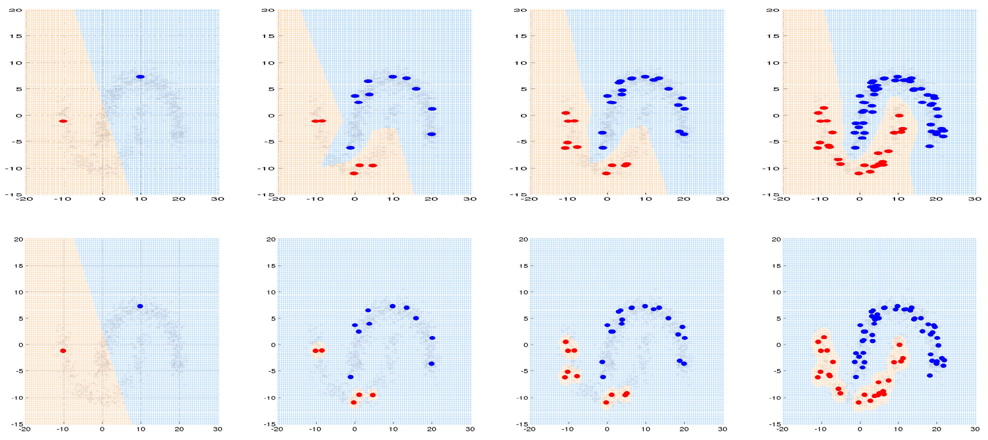


Figure 5.3: We compare the decision boundaries returned from an SVM on several inputs with and without offset. The top row is trained without offset and the bottom row is trained with offset, $\sigma = 1$, and $C = 1$ (the default parameters for LibSVM). We see that when the labeled sets are small, the offset term b biases the decision boundary greatly and overfits the solution.

We train the SVM for a two-moons problem with and without offset using default LibSVM parameters $C = \sigma = 1$ [12]. To demonstrate, in Figure 5.3 we see that the decision boundary obtained using offset conforms very tightly to the given labels. For some datasets, a reasonable decision boundary can be obtained through parameter tuning, but in many SSL problems, the labeled set is not large enough to allow for reasonable parameter tuning. The issue stems from the fact that the offset provides a global bias on the decision function, however, we generally have very limited information on the entire space. When using the offset term, we can see from Figure 5.3 that iteratively assigning new labels based on the decision function with offset can lead to very poor generalization performance. Thus, it seems more fitting for an iterative procedure such as the proposed method to use the offset-free version of the SVM problem.

Consequently we will use the offset-free version of the S^{3}VM problem in the proposed method.

Optimization Formulation

The proposed STAR-SVM framework solves a sequence of optimization problems with C_i^* values sequentially determined by the process. For notational convenience, rather than keeping track of indices in $\mathcal{U}$ which are newly labeled, the index sets will be updated at each iteration. That is, if example x_i is assigned a label at iteration k, then we update the index sets as follows,

$$\begin{aligned}\mathcal{L}_{k+1} &= \mathcal{L}_k \cup \{i\}\\ \mathcal{U}_{k+1} &= \mathcal{U}_k \setminus \{i\}.\end{aligned}$$

We will use the convention that $\mathcal{L}_0$ and $\mathcal{U}_0$ will represent the initial labeled and unlabeled sets respectively.

The primal optimization problem for each STAR-SVM at iteration k is written as,

$$\begin{aligned}\min_{y_\mathcal{U}} \min_{w}\ & P_k(w, y_\mathcal{U}) = \frac{1}{2}w^T w + \sum_{i\in\mathcal{L}_k} C_i\xi_i + \sum_{i\in\mathcal{U}_k} C_i^*\xi_i\\ s.t.\ & y_i(w\cdot\phi(x_i)) \geq 1-\xi_i,\ i=1,...,m\\ & \xi_i \geq 0,\ i=1,...,m\\ & y_i \in \{-1,1\}\end{aligned} \tag{5.12}$$

We can interpret this as finding the labeling that achieves the maximum margin, i.e., finding a labeling that can be separated the best by a decision hyperplane defined by w. The terms, $C_i\xi_i$ and $C_i^*\xi_i$ can be interpreted as the penalty associated with having a label misclassified.

For notational simplicity, we have dropped the dependencies of the regularization parameters C_i and C_i^* on k. The associated dual is written as,

$$\begin{aligned}\min_{y_\mathcal{U}} \max_{\alpha}\ & D_k(\alpha, y_\mathcal{U}) = e^T\alpha - \frac{1}{2}\alpha^T YKY\alpha\\ s.t.\ & 0 \leq \alpha_i \leq C_i,\ i\in\mathcal{L}_k\\ & 0 \leq \alpha_j \leq C_j^*,\ j\in\mathcal{U}_k,\end{aligned} \tag{5.13}$$

where Y is the diagonal matrix of labels, and K is the kernel matrix.

Since no confidence or label information is assumed a priori for points in $\mathcal{U}_0$, we initialize $C_i^* = 0, \forall i \in \mathcal{U}_0$. Thus each subproblem involves training an SVM on the labeled set $\mathcal{L}_k$. In addition, the optimization problems solved at each iteration are standard SVM problems rather than combinatorial optimization problems, which are intractable.

Note that by incorporating the class proportion information into the regularization parameters C_i and using the offset-free SVM formulation, the optimization problem does not have any equality constraint. This is an important point since it makes solving the SVM problem from a warm-start easier. Note that for other formulations, solving the SVM requires working sets with a minimum of two elements to preserve the equality constraints. The SVM subproblems for STAR-SVM can be solved using TWO-SVM, which exploits updating warm-start solutions.

5.3.2 Confidence Score

For self-training to be successful, a reasonable confidence score must be chosen. The confidence score suggested for branching in [14] is the increase in the lower bound objective value if the label were swapped. However, this would involve $2|\mathcal{U}_k|$ SVM solves at each iteration, which is not feasible for large datasets. In the following theorem, we will show that using the magnitude of the functional margin is a reasonable choice to assign confidence.

Without loss of generality, we will partition our variables into labeled and unlabeled sets as follows,

$$K = \begin{pmatrix} K_{\mathcal{L}\mathcal{L}} & K_{\mathcal{L}\mathcal{U}} \\ K_{\mathcal{U}\mathcal{L}} & K_{\mathcal{U}\mathcal{U}} \end{pmatrix}, \; y = \begin{pmatrix} y_{\mathcal{L}} \\ y_{\mathcal{U}} \end{pmatrix}, \; \alpha = \begin{pmatrix} \alpha_{\mathcal{L}} \\ \alpha_{\mathcal{U}} \end{pmatrix}, \tag{5.14}$$

where $\mathcal{L}$ and $\mathcal{U}$ correspond to the rows or columns indexed by $\mathcal{L}_k$ and $\mathcal{U}_k$, respectively.

To measure the confidence of each label, we will measure the change in the objective value if the label is swapped. Intuitively, if the objective value dramatically worsens if a label is swapped, then we should be confident in the original label since the current classifier associates a strong loss with the flipped label. Conversely, if there is not very much difference in objective if the label is swapped, then the classifier is not biased towards either labeling and is likely not confident with the current label. For a given index set $\mathcal{I} \subseteq \mathcal{L}_k$, define the function,

$$\Delta D_k(\alpha, y \mid \mathcal{I}) \coloneqq D_k(\alpha, y) - D_k(\alpha, \tilde{y}) \tag{5.15}$$

where $\tilde{y}_j = y_j$ when $j \notin \mathcal{I}$ and $\tilde{y}_j = -y_j$ when $j \in \mathcal{I}$, that is the vector of labels with indices in $\mathcal{I}$ swapped. $\Delta D(\alpha, y \mid \mathcal{I})$ is the difference between the objective values in (5.13) if the indices in $\mathcal{I}$ have their labels swapped.

Theorem 5.3.1. *Let $\alpha_{\mathcal{L}}$ be a feasible solution to the offset-free dual SVM problem (5.13), trained on the labeled set $\mathcal{L}_k$ with corresponding labels $y_{\mathcal{L}}$. Let $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{i\}$, for any $i \in \mathcal{U}_k$ and let $y_i = \mathbf{sgn}(\sum_{j\in\mathcal{L}_k} K_{i,j} y_j \alpha_j)$. Then for any $\alpha_i \geq 0$,*

$$\Delta D_{k+1}([\alpha_{\mathcal{L}}, \alpha_i], [y_{\mathcal{L}}, y_i] \mid \{i\}) = -2\alpha_i y_i K_{i,\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}} \leq 0.$$

Proof. Let $Y_{\mathcal{L}} = \mathbf{diag}(y_{\mathcal{L}})$. Then we can partition D_{k+1} as follows.

$$D_{k+1}([\alpha_{\mathcal{L}}, \alpha_i], [y_{\mathcal{L}}, y_i]) = -\frac{1}{2}\alpha_{\mathcal{L}}^T Y_{\mathcal{L}} K_{\mathcal{L}\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}} + e^T \alpha_{\mathcal{L}} - \alpha_i y_i K_{i,\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}} - \frac{1}{2}\alpha_i^2 K_{ii} + \alpha_i$$

Assigning $y_i = \mathbf{sgn}(\sum_{j\in\mathcal{L}_k} K_{i,j} y_j \alpha_j)$, we have

$$D_{k+1}([\alpha_{\mathcal{L}}, \alpha_i], [y_{\mathcal{L}}, y_i]) - D_{k+1}([\alpha_{\mathcal{L}}, \alpha_i], [y_{\mathcal{L}}, -y_i]) = -2\alpha_i y_i K_{i,\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}} \tag{5.16}$$

Since $\mathbf{sgn}(y_i) = \mathbf{sgn}(K_{i,\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}})$ and $\alpha_i \geq 0$, the quantity $-2\alpha_i y_i K_{i,\mathcal{L}} Y_{\mathcal{L}} \alpha_{\mathcal{L}} \leq 0$, completing the proof. □

Recall that S^{3}VM finds the local maximizer with the smallest objective value. Theorem 5.3.1 states we can decrease the objective value by simply swapping labels for any points that are inconsistent with the decision hyperplane. Also, we have shown that the magnitude of the functional margin is proportional to the change from swapping labels. Thus, choosing examples that have the largest functional margin corresponds to finding the coordinate which yields the largest rate of change in ΔD_k.

An important aspect of this algorithm we will show is that the intermediate SVM subproblems can be solved very efficiently using warm-starts. Suppose that the label of the i^{th} example is introduced to the labeled set at iteration $k+1$, that is, $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{i\}$. Let $\alpha^*_{(k)}$ be the optimal solution to the inner maximization problem in (5.13) over the labeled set $\mathcal{L}_k$. A natural warm-start solution will be to set $\alpha_i = 0$ (the coefficient corresponding to the newly added example) and to use coefficients given by α^*_k for the indices in $\mathcal{L}_k$. Explicitly, this corresponds to setting $\alpha = [\alpha^*_k, 0]^\top$ as an initial feasible solution to (5.5) for the index set $\mathcal{L}_{k+1}$. In Theorem 5.3.2, we will show that after one iteration of the TWO-SVM algorithm, which greedily updates one index (which will be index i in this case), the duality gap of the SVM dual problem (5.13) can be bounded.

Theorem 5.3.2. *Let α_k^* be an optimal solution to (5.13) over the set $\mathcal{L}^{(k)}$ with labels $y^{(k)}$. For a feasible solution α, define the dual suboptimality measure as*

$$\nu(\alpha) = D(\alpha^*, y^{(k)}) - D(\alpha, y^{(k)}).$$

Let $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{i\}$ be the updated label set for some index $i \in \mathcal{U}_k$. If $\alpha_0 = [\alpha_k^*, 0]$ is an initial warm-start solution to (5.13) for the set $\mathcal{L}_{k+1}$ with given labels $y^{(k+1)} = [y^{(k)}, y_i]^\top$ for some $y_i \in \{-1, 1\}$, and α' is the updated solution after one iteration of TWO-SVM, then

$$\nu(\alpha') \leq C_i \ell(x_i, y_i, \alpha')$$

where ℓ is the hinge-loss function defined in (5.10).

Proof. As noted in [72], for a given feasible solution α, the duality gap over the training set $(\mathcal{L}_k, y^{(k)})$ is,

$$\begin{aligned}\mathrm{gap}_k(\alpha) = \alpha^T Y K Y \alpha - e^T \alpha + \sum_{i \in \mathcal{L}_k} C_i \ell(x_i, y_i, f) \\ + \sum_{i \in \mathcal{U}_k} C_i^* \ell(x_i, y_i, f)\end{aligned}$$

By strong duality, since $\alpha_{\mathcal{L}}^*$ is the optimal solution to the SVM problem trained on $\mathcal{L}_k$, $\mathrm{gap}_k(\alpha_k^*) = 0$. Thus, using the warm start solution for $\mathcal{L}_{k+1}$, we have that,

$$\mathrm{gap}_{k+1}(\alpha_0) = \mathrm{gap}_{k+1}([\alpha_k^*, 0]) = C_i \ell(x_i, y_i, f). \tag{5.17}$$

If $[\alpha_k^*, 0]^\top$ is an optimal solution to (5.13) over the training set $(\mathcal{L}_{k+1}, y^{(k+1)})$, then the duality gap is 0 and the result is trivially satisfied. Otherwise, the greedy update for TWO-SVM will choose index i to improve. In this case, the updated solution given by one iteration of TWO-SVM is $\alpha' = [\alpha_k^*, \alpha_i]^\top$ for some $0 \leq \alpha_i \leq C_i$.

Consider,

$$\nu(\alpha') = D_{k+1}(\alpha_{k+1}^*, y^{(k+1)}) - D_{k+1}(\alpha', y^{(k+1)}). \tag{5.18}$$

Let $P_{k+1}(w^*, y^{(k+1)})$ be the optimal value to the primal problem for the training set $(\mathcal{L}_{k+1}, y^{(k+1)})$. Let $(w_0, y^{(k+1)})$ be the primal solution associated with the dual solution $(\alpha_0, y^{(k+1)})$. Then,

$$\begin{aligned}&P_{k+1}(w^*, y^{(k+1)}) \\ &\leq P_{k+1}(w_0, y^{(k+1)}) \\ &= D_{k+1}(\alpha_0, y^{(k+1)}) + \mathrm{gap}_{k+1}(\alpha_0)\end{aligned} \tag{5.19}$$

where the last line follows from the definition of the duality gap and strong duality.

Again from strong duality, we have that $P_{k+1}(w^*, y^{(k+1)}) = D_{k+1}(\alpha^*_{k+1}, y^{(k+1)})$. Substituting (5.19) into (5.18), we get,

$$\begin{aligned} &\nu(\alpha') \\ &\leq D_{k+1}(\alpha_0, y^{(k+1)}) + \text{gap}_{k+1}(\alpha_0) - D_{k+1}(\alpha', y^{(k+1)}) \\ &\leq \text{gap}(\alpha_0) \end{aligned}$$

where the last inequality comes from the fact that TWO-SVM updates α_i to increase the objective function, so $D_{k+1}(\alpha', y^{(k+1)}) \geq D_{k+1}(\alpha_0, y^{(k+1)})$. The result follows from above and (5.17). □

This indicates that after one step of the TWO-SVM algorithm, the dual suboptimality measure will be bounded above by a product of the loss and its regularization parameter C_i. Typically, the loss associated with new training points becomes very small as the algorithm sees more training points, and the warm-start solution becomes very close to the optimal solution. This implies that the TWO-SVM algorithm will converge very quickly, possibly after just a single TWO-SVM update.

Moreover, the following lemma from [72] states a relation between the duality gap and the step sizes of the ascent directions.

Lemma 5.3.3. *([72]) Let $\alpha \in [0, C] = [0, C_1] \times [0, C_2] \times ... \times [0, C_n]$ and define the function, $\sigma : (\alpha, I \subseteq \{1, ..., n\}) \to \mathbb{R}$ as,*

$$\sigma(\alpha|I) := \sup_{\substack{\tilde{\alpha} \in [0,C] \\ \tilde{\alpha}_i = \alpha_i, \forall i \notin I}} \langle \nabla D(\alpha), \tilde{\alpha} - \alpha \rangle$$

where $\nabla D(\alpha)$ is the gradient of the SVM dual objective function in (5.8) evaluated at a feasible point α. Then,

$$\sum_{i=1}^{n} \sigma(\alpha|\{i\}) = gap(\alpha).$$

From this lemma we see that by choosing examples that yield small duality gaps, we limit the sum of the ascent direction step sizes. This means that we can find solutions that are closer to the warm-start, allowing for a smooth labeling.

5.3.3 Computational Efficiency

In practice, it is inefficient to only add one label at each iteration, since there are likely several examples which are quite similar and can be added all at once without compromising the performance of the algorithm. In most self-training procedures, a confidence threshold is chosen to accept all labelings that are above this threshold. To avoid introducing new hyperparameters, we propose to use an adaptive thresholding scheme in Algorithm 8. The idea is to threshold the confidence scores, where we gradually decrease this threshold as more points become labeled. The rationale is that the earlier iterations have a very large impact on the classifier and we wish to only accept points with very high confidence. In the later iterations when the labeled set becomes large, we expect that any remaining unlabeled points will be ambiguous and may not strongly belong to either class. Since the regularization parameters C_i diminish as the iteration count grows larger, these points will likely contribute very little to the final classifier and we do not wish to spend as much time rigorously classifying each point compared to the early iterations. We will use the notation $|A|$ to denote the cardinality of the set A.

Algorithm 8 Adaptive Thresholding

```
Input: Vector of confidence scores $g \in \mathbb{R}^{|\mathcal{U}_k|}$
    Original Labeled set $\mathcal{L}_0$
    Original indices of unlabeled points $\mathcal{U}_0$
    Current labeled set $\mathcal{L}_k$,
Output: Indices of confident labels $\mathcal{I}$
1: $\tau \leftarrow \frac{|\mathcal{L}_k|-|\mathcal{L}_0|}{|\mathcal{U}_0|}$
2: $\mathcal{I}^+ \leftarrow \{i : g_i \geq (1-\tau)\max(g)\}$
3: $\mathcal{I}^- \leftarrow \{i : -g_i \geq (1-\tau)\max(-g)\}$
4: $\mathcal{I} \leftarrow \mathcal{I}^+ \cup \mathcal{I}^-$
```

Incorporating Class Proportion Knowledge

For imbalanced datasets, penalizing the regularization parameters, C, for the positive and negative classes separately can improve performance, see, e.g. [43]. When data is highly imbalanced, the positive (minority class) labels may be ignored by the classifier since the loss for the associated class will be small due to the small size of the minority class. Thus, by adjusting the regularization parameters to reflect the class ratio, we also penalize the loss accordingly. For our iterative approach, we can adjust the regularization parameters accordingly to reflect the prior knowledge of the class ratios when the labels are assigned. Let r be the proportion of positive examples (assuming the positive class is the minority class), either provided as a user parameter or estimated from the labeled data, $r = \frac{\sum_{i=1}^{l} \max(0, y_i)}{l}$, where $l = |\mathcal{L}_0|$. Then the associated weights will be defined as,

$$\begin{aligned} C^+ &= rC \\ C^- &= (1-r)C. \end{aligned} \tag{5.20}$$

Combining this with (5.9), the update for C_i at iteration k becomes,

$$C_i = \begin{cases} \gamma^k C^+, & \text{if } y_i = 1 \\ \gamma^k C^-, & \text{if } y_i = -1. \end{cases} \tag{5.21}$$

5.3.4 Algorithm Overview

We now briefly summarize the proposed algorithm, STAR-SVM, in Algorithm 9.

Although STAR-SVM in Algorithm 9 is motivated in the binary class setting, extending this algorithm to a multi-class algorithm can be accomplished using a standard one-vs-rest approach. If there are c classes, we train the STAR-SVM algorithm c times where each trial corresponds to a different class being the positive class and label all other classes the negative class. An unlabeled example x_i, is given the label j, if x_i was labeled positive in the earliest iteration for the trial where j is the positive class (or the latest trial j if x_i is labeled negatively for all trials). This labeling agrees with the assumption that earlier labelings correspond to higher confidence. Although other approaches can be considered, this approach will be used for the subsequent experimental results.

5.4 Experimental Results

In this section, we compare STAR-SVM against leading SSL methods on a variety of imbalanced datasets, both graph and S^{3}VM based methods are considered. The benchmark datasets are provided by UCI [4] and Keel [1]. A wide variety of datasets are chosen to reflect varying structural properties of the data as well as varying class proportion ratios. This experiment will examine the performance of the classifiers in a setting where little is known about the true distributions of the data. We emphasize imbalanced datasets since this is an area where SSL methods typically struggle the most [75, 76]. In addition, we

Algorithm 9 STAR-SVM

```
Input: Data $\mathcal{X}$
  Size of training set $n$
  Labeled indices $\mathcal{L}_0$
  Unlabeled indices $\mathcal{U}_0$
  Labels $y$
  Kernel matrix $K$
  Regularization parameter $C$
  Confidence weight $\gamma$
  Class proportion $r$
Output: $f^*(x) = \sum_{i=1}^{n} K(x, x_i) y_i \alpha_i$
  $k \leftarrow 1$
  repeat
    $\alpha \leftarrow \texttt{TWO-SVM}(\mathcal{X}_{\mathcal{L}}, y_{\mathcal{L}}, C_{\mathcal{L}})$
    $g \leftarrow K_{\mathcal{L}\mathcal{L}} diag(y_{\mathcal{L}})\alpha$
    $\mathcal{I} \leftarrow \texttt{threshold}(g, |\mathcal{L}_0|, |\mathcal{U}_0|, |\mathcal{L}_k|)$
    for $j \in \mathcal{I}$ do
      if $g_j > 0$ then
        $C_j \leftarrow \gamma^k C^+$
      else
        $C_j \leftarrow \gamma^k C^-$
      end if
    end for
    $\mathcal{U}_{k+1} \leftarrow \mathcal{U}_k \setminus \mathcal{I}$
    $y_{\mathcal{I}} \leftarrow \mathbf{sgn}(g_{\mathcal{I}})$
    $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \cup \mathcal{I}$
    $k \leftarrow k + 1$
  until $\mathcal{U}_k = \emptyset$
```

want to emphasize the importance of adapting regularization parameters to incorporate class imbalances. Additionally, since S^{3}VM methods are inherently binary classifiers, any multi-class setting will typically require solving several imbalanced S^{3}VM problems.

5.4.1 Imbalanced Datasets

We demonstrate STAR-SVM's performance across a variety of imbalanced realworld datasets from UCI [4] and Keel [1]. The multi-class data has been transformed into imbalanced binary datasets using the classes suggested in [27]. We summarize the descriptions in Table 5.1.

The methods we compare against are,

1. Gaussian Fields and Harmonic Functions (GFHF) [86], which use the harmonic solution to solve the graph diffusion.

2. Local and Global Consistency (LGC) [84], which use normalized graph Laplacian and soft constraints for given labels.

3. Greedy Gradient Maximum Cut (GGMC) [75, 76], which use alternating minimization over both discrete labels and continuous decision function.

4. Laplacian SVM (LapSVM) [7], which adds graph regularization term to SVM framework.

5. WellSVM [54], which is a convex relaxation to S^{3}VM.

The code for each method is provided by the authors.

For most SSL problems, the labeled set is insufficient to perform reliable parameter tuning. Moreover, as noted in [15], some methods benefit more from parameter tuning than others. Thus, to ensure fairness and to simulate a realistic scenario, parameter tuning is not applied to the datasets in Table 5.1. Instead, all methods used the same set of parameters summarized in the Table 5.2.

For LapSVM, the results reported here ignore the offset term as well. Despite the fact that the original LapSVM solver includes a bias term, we have found that the results for LapSVM have improved dramatically when the offset is ignored for the classification.

For each dataset, $n = 20$ labeled samples are randomly chosen, with at least one labeled example from each class. For each dataset, the experiments are repeated using

Dataset Name	Source	Dimension	Points	Num. Positive
Ecoli	UCI	7	336	35
Crime	UCI	100	1994	100
Libras	UCI	7	360	90
Oil	[48]	49	937	41
Optical Digits	UCI	64	5620	554
Wine	UCI	11	4898	183
Satellite Image	UCI	36	6435	626
Vowel	Keel	10	989	90
Shuttle0vs4	Keel	9	1829	123
Page Block	Keel	10	5472	559
Glass4	Keel	9	214	13
Cleveland0vs4	Keel	13	173	13
Contraceptive	Keel	9	1473	333
Euthyroid	UCI	42	3163	293
Spectrometer	UCI	93	531	45

Table 5.1: Imbalanced Datset Description

Parameter	Value	Description
k	6	k used for k-NN graph construction
kernel type	RBF	kernel function used
σ	1	RBF kernel width
γ_A	1	kernel regularization weight for LapSVM
γ_I	0.1	graph regularization weight for LapSVM
γ	0.7	confidence weight for STAR-SVM
C	1	SVM regularization parameter

Table 5.2: Parameters

thirty different random initial labeled sets. The imbalance ratio r is estimated from the given labels. The average performances are reported. We also tried experiments with $n \in \{2, 5, 10, 50\}$, but the trends are similar throughout.

For a given classifier, let TP, FP, TN, FN to be the number of true positives, false positives, true negatives, and false negatives reported by the classifier on a test set. We report the following imbalanced classification metrics, F_1-score, G-mean, and AUC. F_1-score and G-mean are defined as follows,

$$F_1 \coloneqq 2\frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \text{ and G-mean} \coloneqq \sqrt{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}},$$

where the F_1 score is the harmonic mean of the precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) classifier, and G-mean is the geometric mean of the precision and recall of the classifier.

If the classifier outputs a score instead of a class label, then we can infer labels by thresholding the score vector. Explicitly, suppose we are given a vector of scores $s \in \mathbb{R}^n$, where s_i corresponds to the score associated with training point x_i. Given a threshold value $T \in \mathbb{R}$, we can assign a label $\hat{y}_i = 1$ if $s_i > T$ and $\hat{y}_i = -1$ otherwise. For each value of T, we can define a *true positive rate*, which is the recall with labels thresholded at the value T, and a *false positive*, denoted $FPR(T) = \frac{FP}{FP+TN}$, where the quantities FP and TN are associated with the labeling inferred by the threshold at T. The *receiver operating characteristic* ROC-curve plots the true positive rate vs. the false positive rate across all threshold value of T. The *area under curve* (AUC) is then defined as the area under the ROC curve.

For imbalanced datasets, we will report the F_1-score, Geometric Mean, and AUC which are more suitable for imbalanced classification [42].

5.4.2 Computing Time Comparisons

For the experiments run on the imbalanced datasets, we report the average CPU time required to run each algorithm. All algorithms are implemented in MATLAB, however, LapSVM and WellSVM utilize subroutines written in C++.

In Figure 5.4, we see that STAR-SVM scales much better as the datasets increase in size. Experimentally, after the first few iterations, each TWO-SVM solve terminates quickly.

Dataset Name	GFHF	LGC	GGMC	LapSVM	WellSVM	STAR-SVM
Ecoli	$\mathbf{0.54 \pm 0.18}$	0.05 ± 0.17	0.48 ± 0.05	$\mathbf{0.50 \pm 0.15}$	$\mathbf{0.48 \pm 0.19}$	$\mathbf{0.53 \pm 0.05}$
Crime	0.12 ± 0.06	0.15 ± 0.10	0.19 ± 0.07	0.21 ± 0.08	$\mathbf{0.35 \pm 0.14}$	0.29 ± 0.10
Libras	0.40 ± 0.18	$\mathbf{0.49 \pm 0.17}$	0.26 ± 0.10	$\mathbf{0.47 \pm 0.19}$	0.31 ± 0.23	$\mathbf{0.49 \pm 0.16}$
Oil	0.08 ± 0.01	$\mathbf{0.17 \pm 0.13}$	0.12 ± 0.09	$\mathbf{0.16 \pm 0.13}$	0.08 ± 0.05	$\mathbf{0.22 \pm 0.14}$
Optical Digits	0.01 ± 0.01	$\mathbf{0.72 \pm 0.21}$	0.39 ± 0.12	$\mathbf{0.64 \pm 0.17}$	0.23 ± 0.24	0.47 ± 0.08
Wine	0.01 ± 0.02	0.02 ± 0.02	0.07 ± 0.02	$\mathbf{0.10 \pm 0.06}$	$\mathbf{0.14 \pm 0.08}$	$\mathbf{0.10 \pm 0.05}$
Satellite Image	0.08 ± 0.12	0.28 ± 0.22	0.28 ± 0.07	0.36 ± 0.08	0.24 ± 0.23	$\mathbf{0.44 \pm 0.11}$
Vowel	0.49 ± 0.13	0.48 ± 0.16	0.41 ± 0.12	$\mathbf{0.55 \pm 0.09}$	$\mathbf{0.57 \pm 0.13}$	0.51 ± 0.09
Shuttle0vs4	0.87 ± 0.14	0.88 ± 0.16	0.94 ± 0.01	0.89 ± 0.09	0.76 ± 0.16	$\mathbf{1.00 \pm 0.00}$
Page Block	0.47 ± 0.20	0.50 ± 0.17	0.24 ± 0.07	$\mathbf{0.60 \pm 0.08}$	0.43 ± 0.22	$\mathbf{0.52 \pm 0.14}$
Glass4	$\mathbf{0.42 \pm 0.11}$	0.06 ± 0.10	0.40 ± 0.11	$\mathbf{0.45 \pm 0.11}$	0.35 ± 0.17	$\mathbf{0.49 \pm 0.11}$
Cleveland0vs4	$\mathbf{0.41 \pm 0.26}$	0.17 ± 0.20	$\mathbf{0.40 \pm 0.24}$	$\mathbf{0.44 \pm 0.22}$	$\mathbf{0.40 \pm 0.28}$	$\mathbf{0.50 \pm 0.21}$
Contraceptive	0.21 ± 0.08	0.18 ± 0.09	0.29 ± 0.05	0.32 ± 0.05	0.29 ± 0.08	$\mathbf{0.36 \pm 0.04}$
Euthyroid	0.13 ± 0.08	0.13 ± 0.09	0.17 ± 0.05	$\mathbf{0.22 \pm 0.07}$	0.07 ± 0.06	$\mathbf{0.23 \pm 0.06}$
Spectrometer	$\mathbf{0.55 \pm 0.19}$	0.37 ± 0.28	0.39 ± 0.19	0.50 ± 0.18	0.40 ± 0.15	$\mathbf{0.62 \pm 0.15}$
Average	0.319	0.310	0.335	0.428	0.339	**0.451**
Avg. Rank	4.267	4.267	4.400	2.400	4.067	**1.600**

Table 5.3: F_1 Score with 20 labeled examples. Top performers are bolded in each row.

Dataset Name	GFHF	LGC	GGMC	LapSVM	WellSVM	STAR-SVM
Ecoli	0.72 ± 0.21	0.07 ± 0.21	0.82 ± 0.10	0.76 ± 0.26	0.67 ± 0.22	$\mathbf{0.87 \pm 0.02}$
Crime	0.26 ± 0.09	0.32 ± 0.15	0.49 ± 0.20	0.47 ± 0.19	$\mathbf{0.59 \pm 0.21}$	$\mathbf{0.53 \pm 0.13}$
Libras	0.59 ± 0.24	$\mathbf{0.67 \pm 0.22}$	$\mathbf{0.66 \pm 0.18}$	$\mathbf{0.70 \pm 0.24}$	0.49 ± 0.26	$\mathbf{0.68 \pm 0.19}$
Oil	0.03 ± 0.10	0.32 ± 0.17	$\mathbf{0.41 \pm 0.16}$	0.39 ± 0.19	0.28 ± 0.12	$\mathbf{0.47 \pm 0.13}$
Optical Digits	0.07 ± 0.02	$\mathbf{0.87 \pm 0.17}$	0.77 ± 0.11	$\mathbf{0.91 \pm 0.05}$	0.39 ± 0.29	0.78 ± 0.07
Wine	0.06 ± 0.07	0.07 ± 0.07	$\mathbf{0.49 \pm 0.07}$	0.39 ± 0.14	0.38 ± 0.14	$\mathbf{0.44 \pm 0.13}$
Satellite Image	0.16 ± 0.16	0.43 ± 0.27	$\mathbf{0.67 \pm 0.12}$	$\mathbf{0.75 \pm 0.13}$	0.42 ± 0.34	$\mathbf{0.76 \pm 0.14}$
Vowel	0.66 ± 0.15	0.64 ± 0.18	$\mathbf{0.78 \pm 0.11}$	$\mathbf{0.83 \pm 0.10}$	$\mathbf{0.75 \pm 0.14}$	0.76 ± 0.10
Shuttle0vs4	0.91 ± 0.13	$\mathbf{0.94 \pm 0.15}$	$\mathbf{1.00 \pm 0.00}$	0.90 ± 0.08	0.91 ± 0.07	$\mathbf{1.00 \pm 0.00}$
Page Block	0.58 ± 0.19	0.63 ± 0.18	0.60 ± 0.10	$\mathbf{0.81 \pm 0.08}$	0.62 ± 0.22	0.69 ± 0.14
Glass4	0.66 ± 0.14	0.10 ± 0.17	0.75 ± 0.09	$\mathbf{0.78 \pm 0.12}$	0.59 ± 0.22	$\mathbf{0.82 \pm 0.10}$
Cleveland0vs4	0.56 ± 0.32	0.24 ± 0.26	0.63 ± 0.34	$\mathbf{0.68 \pm 0.32}$	0.53 ± 0.33	$\mathbf{0.71 \pm 0.28}$
Contraceptive	0.38 ± 0.09	0.34 ± 0.11	0.48 ± 0.06	0.51 ± 0.06	0.47 ± 0.09	$\mathbf{0.55 \pm 0.04}$
Euthyroid	0.37 ± 0.15	0.29 ± 0.15	0.46 ± 0.12	$\mathbf{0.51 \pm 0.12}$	0.25 ± 0.12	$\mathbf{0.56 \pm 0.13}$
Spectrometer	$\mathbf{0.67 \pm 0.18}$	0.44 ± 0.29	0.63 ± 0.22	$\mathbf{0.70 \pm 0.18}$	0.62 ± 0.14	$\mathbf{0.75 \pm 0.14}$
Average	0.445	0.425	0.643	0.673	0.531	**0.691**
Avg. Rank	4.933	4.667	2.933	2.333	4.600	**1.533**

Table 5.4: G-mean for 20 labeled examples

Dataset Name	GFHF	LGC	GGMC	LapSVM	WellSVM	STAR-SVM
Ecoli	**0.91 ± 0.02**	0.67 ± 0.19	0.90 ± 0.01	**0.92 ± 0.03**	**0.93 ± 0.04**	**0.94 ± 0.02**
Crime	0.70 ± 0.10	0.73 ± 0.04	0.32 ± 0.04	0.69 ± 0.12	**0.85 ± 0.08**	0.76 ± 0.05
Libras	**0.85 ± 0.18**	**0.87 ± 0.08**	0.70 ± 0.12	**0.86 ± 0.11**	**0.81 ± 0.10**	**0.86 ± 0.08**
Oil	0.50 ± 0.02	**0.76 ± 0.06**	0.48 ± 0.05	**0.72 ± 0.10**	0.72 ± 0.05	**0.76 ± 0.04**
Optical Digits	0.69 ± 0.18	0.95 ± 0.06	0.95 ± 0.01	**0.98 ± 0.02**	0.72 ± 0.18	0.84 ± 0.07
Wine	0.56 ± 0.09	**0.67 ± 0.05**	0.42 ± 0.07	**0.65 ± 0.08**	**0.69 ± 0.07**	**0.66 ± 0.07**
Satellite Image	**0.88 ± 0.05**	0.64 ± 0.16	**0.88 ± 0.04**	**0.88 ± 0.06**	0.68 ± 0.21	0.79 ± 0.10
Vowel	**0.92 ± 0.07**	0.86 ± 0.09	0.88 ± 0.08	**0.92 ± 0.05**	**0.93 ± 0.04**	0.89 ± 0.05
Shuttle0vs4	**1.00 ± 0.00**	**1.00 ± 0.00**	0.98 ± 0.01	**1.00 ± 0.00**	0.99 ± 0.02	**1.00 ± 0.00**
Page Block	**0.90 ± 0.05**	0.87 ± 0.05	0.75 ± 0.11	**0.92 ± 0.03**	0.85 ± 0.07	**0.90 ± 0.04**
Glass4	0.88 ± 0.09	0.87 ± 0.03	0.66 ± 0.11	**0.95 ± 0.05**	0.93 ± 0.04	**0.96 ± 0.02**
Cleveland0vs4	0.92 ± 0.06	0.89 ± 0.02	0.56 ± 0.19	0.92 ± 0.06	**0.96 ± 0.02**	0.93 ± 0.04
Contraceptive	0.53 ± 0.05	0.53 ± 0.04	0.54 ± 0.04	0.53 ± 0.05	**0.59 ± 0.07**	**0.57 ± 0.05**
Euthyroid	**0.57 ± 0.07**	0.50 ± 0.06	**0.62 ± 0.05**	0.55 ± 0.07	0.57 ± 0.06	**0.57 ± 0.09**
Spectrometer	0.73 ± 0.13	0.84 ± 0.09	0.44 ± 0.06	**0.92 ± 0.07**	0.88 ± 0.06	**0.92 ± 0.08**
Average	0.769	0.776	0.672	**0.828**	0.808	0.823
Avg. Rank	3.933	4.133	4.733	2.867	2.933	**2.400**

Table 5.5: AUC for 20 labeled examples

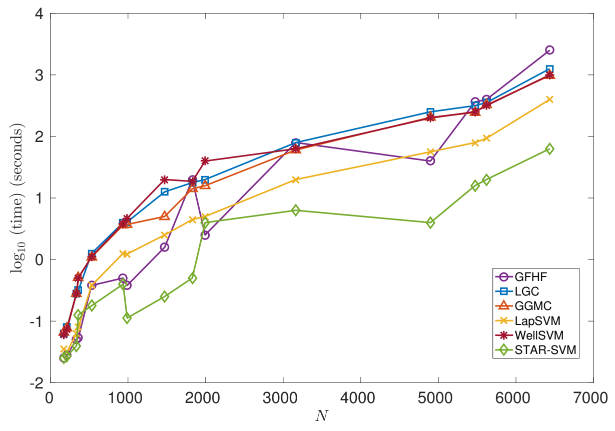


Figure 5.4: CPU times for imbalanced datasets.

5.4.3 Discussion

As observed in Tables 5.3-5.5, STAR-SVM shows very strong performance across all metrics. We observe that for most datasets, one of the graph-based methods (GFHF, LGC, and GGMC) typically struggles. This highlights the sensitivity of the graph construction, which can be very challenging to tune correctly with very limited data.

5.5 Conclusions

In this chapter, we propose a method STAR-SVM that is competitive against state-of-the-art SSL methods on various real-world datasets. We have shown that by gradually adjusting the regularization parameters, the optimization problems adaptively incorporate structure from the unlabeled data even in the presence of noisy datasets. Furthermore, we have shown theoretically and experimentally that growing the labeled set in this fashion allows for smooth variations of the classifier, yielding very quick convergence in solving SVM subproblems using a warm-start. The strong performance of STAR-SVM, in a setting where little is known about the data, suggests that nonconvex objective functions can be a promising direction for future S^{3}VM algorithms.

Chapter 6

Conclusion

In this thesis, we have provided new algorithms to computationally improve a few semi-supervised learning problems. Since the amount of unlabeled data is growing exceptionally quickly, there is a tremendous need for scalable algorithms. In particular, it is impractical to rely purely on manually annotated labels, and algorithms that can leverage the structure revealed by the vast amount of unlabeled data collected can be very appealing.

Low-rank matrix completion problems are one of the core ingredients in modern recommender systems. Since the matrices scale with the number of users and the number of products, it is very important that there exist scalable algorithms to find the low-rank representations. Even constant factor improvements in computation time can lead to more frequent retraining schedules, allowing for more accurate representations in real-world settings. We observed that even though the Frank-Wolfe algorithm can greatly reduce the iteration cost compared to proximal methods [46], the algorithm typically would return a very high rank solution incurring a much higher computation time along with much higher memory requirements.

For simultaneously sparse and low-rank matrix estimation problems, we observed that there is a clear tradeoff between scalable approaches and accurate approaches. While certain Frank-Wolfe methods existed using smooth surrogates, their methods failed to converge to reasonably accurate solutions as observed by noisy sparsity patterns and poor predictive performance [17]. On the other extreme, approaches utilizing proximal gradient descent scaled very poorly due to the full SVD required, but was able to yield high quality solutions in practice [69]. Since many problems such as graph link prediction and sparse covariance estimation require a sparse and low-rank structure, it is tremendously important to find a better algorithm to quickly find high quality solutions.

The final set of problems we addressed was for semi-supervised classification tasks. For many classification tasks, we do not have access to a large amount of labeled data, e.g. handwritten digits, medical diagnoses, credit card fraud, etc. By leveraging different structural assumptions, S^{3}VM and semi-supervised graph based methods have been very successful at returning classifiers with very little labeled data. However, S^{3}VM and graph-based methods appear to struggle on imbalanced classification tasks and the performance is very sensitive to the initial labeling.

Below, we give a brief summary of the specific problems encountered with the current landscape of semi-supervised learning, as well as solutions that we have proposed in the thesis.

6.1 Rank-Drop Steps

For convex approaches to the *low-rank matrix completion* problem, we showed that a few existing convex methods fail to scale. For the proximal gradient descent or projected gradient descent, the necessity of a full SVD, requiring $O(mn^2)$ operations at each iteration, completely prevents these algorithms from scaling to very large problems. While Frank-Wolfe methods partially solves this problem by only requiring computing the leading singular vector pair, it unfortunately has a much slower convergence rate, often yielding very high rank intermediate solutions. The high-rank iterates introduced large storage requirements as well as expensive gradient computations, negating the benefit of the cheap iteration cost.

To address this issue, we proposed *rank-drop steps* to find low-rank intermediate solutions, leveraging the cheap iteration cost of Frank-Wolfe without the computational burden associated with high-rank intermediate iterates. We accomplished this by introducing two subproblems specifically created to decrease the rank of the current iterate. By considering the interior and exterior cases, we showed that the subproblems can be efficiently solved, yielding promising rank-drop solutions. In the experiments considered, the observed rank decreased dramatically compared to the standard Frank-Wolfe, and showed improved empirical computational performance over the existing Frank-Wolfe variants.

6.2 Frank-Wolfe with Uniform Affine Approximations

There are many practical applications that require a representation which is both sparse and low-rank. Using only sparse or low-rank regularizers alone for the graph link prediction problem, for example, can yield very poor predictions. Unfortunately, there are not many efficient algorithms to solve problems with both sparse and low-rank regularizers. We observed that the alternating proximal approach proposed in [69] scaled very poorly even on modest sized datasets and many existing Frank-Wolfe based approaches returned suboptimal solutions due to poor linear approximations. By utilizing uniform affine approximations, we showed that it is possible to leverage the simple Frank-Wolfe subproblems without introducing large approximation errors by defining an appropriate affine approximation that adapts to the Frank-Wolfe step-size schedule. Combining this idea with the rank-drop steps, we showed that this method can scale to very large problems and outperforms the existing nonsmooth Frank-Wolfe variants.

6.2.1 Self-Training with Adaptive Regularization

The Semi-Supervised Support Vector Machine (S^{3}VM) solves a nonconvex, mixed-integer program. Due to the difficulty in solving the problem, convex approximations have typically been used. However, existing approaches have suffered from poor scalability and struggle on certain datasets, compared to graph based counterparts. The poor predictive performance suggests that for some datasets, the convex relaxation may not be a sufficiently accurate approximation to the problem. We presented a self-training approach with self-adapting regularization parameters for S^{3}VM formulations. At each iteration, the regularization parameters are adapted to better reflect label confidence, class proportion, and to gradually include more unlabeled points. We showed that updating the S^{3}VM framework iteratively in this fashion, the sequence of SVM subproblems can be solved very efficiently and the solution generated by this sequence yields superior performance compared to leading semi-supervised learning methods. Moreover, by iteratively updating the weights of the regularization parameters, we could incorporate label imbalances and greatly outperform many existing semi-supervised learning methods on imbalance datasets. Finally, we note that the incremental approach to increasing the training set also means that the kernel matrix can be incrementally built as well, only requiring the active support vectors from each iteration. This contrasts with existing S^{3}VM and graph-based methods which require the full dense kernel matrix at the start of training.

6.3 Future Work

There are still many natural questions that we can pursue:

- Empirically, it seemed that both the in-face step and the rank-drop step converged faster than the regular Frank-Wolfe step. However, in the convergence proofs for both the rank-drop step and the in-face step, the convergence rate is worse than the Frank-Wolfe algorithm by a factor of 2. *Is it possible to improve the analysis or determine conditions where we can theoretically guarantee a better convergence rate for either variant?*

- The convergence rate given for the nonsmooth Frank-Wolfe algorithm proposed (FWUA) appears overly pessimistic to account for the approximation errors added by each component, incurring a constant factor of mn. The empirical performance seems to suggest that the convergence rate can be improved. *Is it possible to improve the constant factor of the convergence so it does not scale quadratically with the dimension?*

- The STAR-SVM algorithm was motivated by the assumption that the convex relaxation was not a sufficiently accurate relaxation to the original S^{3}VM problem. *Is it possible to identify conditions where the convex relaxation is guaranteed to be insufficient?*

References

[1] J Alcalá, A Fernández, J Luengo, J Derrac, S García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(255-287):11, 2010.

[2] Zeyuan Allen-Zhu, Elad Hazan, Wei Hu, and Yuanzhi Li. Linear convergence of a frank-wolfe type algorithm over trace-norm balls. *arXiv preprint arXiv:1708.02105*, 2017.

[3] Andreas Argyriou, Marco Signoretto, and Johan AK Suykens. Hybrid conditional gradient-smoothing algorithms with applications to sparse and low rank regularization. In *Regularization, Optimization, Kernels, and Support Vector Machines*, pages 53–82. Chapman and Hall/CRC, 2014.

[4] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[5] Erik J Balder. On subdifferential calculus. *Lecture notes, Universiteit Utrecht.[51, 52]*, 2010.

[6] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[7] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, 2004.

[8] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[9] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[10] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[11] Léopold Cambier and P-A Absil. Robust low-rank matrix completion by riemannian optimization. *SIAM Journal on Scientific Computing*, 38(5):S440–S460, 2016.

[12] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[13] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, 2006.

[14] Olivier Chapelle, Vikas Sindhwani, and S Sathiya Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in neural information processing systems*, pages 217–224, 2006.

[15] Olivier Chapelle, Vikas Sindhwani, and Sathiya S Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

[16] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. 2004.

[17] Edward Cheung and Yuying Li. Nonsmooth frank-wolfe using uniform affine approximations. *arXiv preprint arXiv:1710.05776*, 2017.

[18] Edward Cheung and Yuying Li. Projection free rank-drop steps. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1539–1545, 2017.

[19] Edward Cheung and Yuying Li. Self-training with adaptive regularization for s3vm. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3633–3640. IEEE, 2017.

[20] Moody T Chu, RE Funderlic, and Gene H Golub. Rank modifications of semidefinite matrices associated with a secant update formula. *SIAM Journal on Matrix Analysis and Applications*, 20(2):428–436, 1998.

[21] Moody T Chu, Robert E Funderlic, and Gene H Golub. On a variational formulation of the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1082–1092, 1997.

[22] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63, 2010.

[23] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *The Journal of Machine Learning Research*, 7:1687–1712, 2006.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[26] Tijl De Bie and Nello Cristianini. Semi-supervised learning using semi-definite programming. *Semi-supervised learning. MIT Press, Cambridge-Massachussets*, 32, 2006.

[27] Zejin Ding. Diversified ensemble classifiers for highly imbalanced data learning and their application in bioinformatics. 2011.

[28] Eugen Egerváry. On rank-diminishing operations and their applications to the solution of linear equations. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 11(5):376–386, 1960.

[29] Wael Emara, Mehmed Kantardzic Marcel Karnstedt, Kai-Uwe Sattler, Dirk Habich, and Wolfgang Lehner. An approach for incremental semi-supervised svm. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 539–544. IEEE, 2007.

[30] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734–4739. IEEE, 2001.

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

[32] Robert M Freund and Paul Grigas. New analysis and results for the frank–wolfe method. *Mathematical Programming*, 155(1-2):199–230, 2016.

[33] Robert M Freund, Paul Grigas, and Rahul Mazumder. An extended frank-wolfe method with" in-face" directions, and its application to low-rank matrix completion. *arXiv preprint arXiv:1511.02204*, 2015.

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[35] Fei Gao, Jingyuan Mei, Jinping Sun, Jun Wang, Erfu Yang, and Amir Hussain. A novel classification algorithm based on incremental semi-supervised support vector machine. *PloS one*, 10(8):e0135709, 2015.

[36] Dan Garber. Faster projection-free convex optimization over the spectrahedron. In *Advances in Neural Information Processing Systems*, pages 874–882, 2016.

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

[38] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[39] Russell A Gordon. *Real Analysis: A first course*. Addison Wesley, 2001.

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

[41] Elad Hazan. Sparse approximate solutions to semidefinite programs. *Lecture Notes in Computer Science*, 4957:306–316, 2008.

[42] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[43] He He and Ali Ghodsi. Rare class classification by support vector machine. In *ICPR*, pages 548–551, 2010.

[44] Cho-Jui Hsieh and Peder A Olsen. Nuclear norm minimization via active subspace selection. In *ICML*, pages 575–583, 2014.

[45] Martin Jaggi. *Sparse Convex Optimization Methods for Machine Learning*. PhD thesis, ETH Zurich, October 2011.

[46] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[47] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[48] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[49] J Kuczyński and H 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.

[50] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for frank-wolfe algorithms. *arXiv preprint arXiv:1312.7864*, 2013.

[51] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

[52] Rasmus Munk Larsen. Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series*, 27(537), 1998.

[53] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[54] Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled svms. *The Journal of Machine Learning Research*, 14(1):2151–2188, 2013.

[55] Yuanqing Li, Cuntai Guan, Huiqi Li, and Zhengyang Chin. A self-training semi-supervised svm algorithm and its application in an eeg-based brain computer interface speller system. *Pattern Recognition Letters*, 29(9):1285–1294, 2008.

[56] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010.

[57] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[58] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[59] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[60] Guillaume Obozinski, Laurent Jacob, and Jean-Philippe Vert. Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*, 2011.

[61] Gergely Odor, Yen-Huan Li, Alp Yurtsever, Ya-Ping Hsieh, Quoc Tran-Dinh, Marwa El Halabi, and Volkan Cevher. Frank-wolfe works for non-lipschitz continuous gradient objectives: Scalable poisson phase retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6230–6234. Ieee, 2016.

[62] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[63] Javier Pena and Daniel Rodriguez. Polytope conditioning and linear convergence of the frank-wolfe algorithm. *arXiv preprint arXiv:1512.06142*, 2015.

[64] Javier Pena, Daniel Rodríguez, and Negar Soheili. On the von neumann and frank–wolfe algorithms with away steps. *SIAM Journal on Optimization*, 26(1):499–512, 2016.

[65] Federico Pierucci, Zaid Harchaoui, and Jérôme Malick. A smoothing approach for composite conditional gradient with nonsmooth loss. 2014.

[66] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[67] Nikhil Rao, Parikshit Shah, and Stephen Wright. Forward–backward greedy algorithms for atomic norm regularization. *IEEE Transactions on Signal Processing*, 63(21):5798–5811, 2015.

[68] Sathya N Ravi, Maxwell D Collins, and Vikas Singh. A deterministic nonsmooth frank wolfe algorithm with coreset guarantees. *arXiv preprint arXiv:1708.06714*, 2017.

[69] Emile Richard, Pierre-André Savalle, and Nicolas Vayatis. Estimation of simultaneously sparse and low rank matrices. *arXiv preprint arXiv:1206.6474*, 2012.

[70] W So. Facial structures of schatten p-norms. *Linear and Multilinear Algebra*, 27(3):207–212, 1990.

[71] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[72] Ingo Steinwart, Don Hush, and Clint Scovel. Training svms without offset. *The Journal of Machine Learning Research*, 12:141–202, 2011.

[73] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[74] Silvia Villa, Lorenzo Rosasco, Sofia Mosci, and Alessandro Verri. Proximal methods for the latent group lasso penalty. *Computational Optimization and Applications*, 58(2):381–407, 2014.

[75] Jun Wang, Tony Jebara, and Shih-Fu Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM, 2008.

[76] Jun Wang, Tony Jebara, and Shih-Fu Chang. Semi-supervised learning using greedy max-cut. *The Journal of Machine Learning Research*, 14(1):771–800, 2013.

[77] G Alistair Watson. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170:33–45, 1992.

[78] Leon L Wegge. Mean value theorem for convex functions. *Journal of Mathematical Economics*, 1(2):207–208, 1974.

[79] DJ White. Extension of the frank-wolfe algorithm to concave nondifferentiable objective functions. *Journal of optimization theory and applications*, 78(2):283–301, 1993.

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

[81] Quanming Yao, James Kwok, et al. Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity. *arXiv preprint arXiv:1606.03841*, 2016.

[82] Quanming Yao and James T. Kwok. Accelerated inexact soft-impute for fast large-scale matrix completion. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4002–4008. AAAI Press, 2015.

[83] Quanming Yao and James T Kwok. Greedy learning of generalized low-rank models. In *IJCAI*, pages 2294–2300, 2016.

[84] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, pages 321–328, 2003.

[85] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3, 2006.

[86] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

APPENDICES

Appendix A

Useful Properties of Matrix Norms

Property A.0.1. *Let $A \in \mathbb{R}^{m\times n}$ be a rank-r matrix. The Frobenius norm has the following equivalent characterizations:*

(i) $\|A\|_F = \sqrt{\sum_{ij} A_{ij}^2}$.

(ii) $\|A\|_F = \sqrt{\mathbf{tr}(A^\top A)}$.

(iii) $\|A\|_F = \sqrt{\sum_{i=1}^r \sigma_i(A)}$.

Proof. (i) and (ii) are equivalent immediately follows algebraically. To see that (ii) and (iii) are equivalent, consider the SVD $A = U\Sigma V^\top$. Then,

$$\begin{aligned}\|A\|_F &= \sqrt{\mathbf{tr}(A^\top A)} \\ &= \sqrt{\mathbf{tr}(V\Sigma U^\top U\Sigma V^\top)} \\ &= \sqrt{\mathbf{tr}(V^\top V\Sigma U^\top U)} \\ &= \sqrt{\mathbf{tr}(\Sigma)} \\ &= \sqrt{\sum_{i=1}^r \sigma_i(A)}.\end{aligned}$$

□

Property A.0.2. *The Frobenius norm is orthogonally invariant. Explicitly, if $B \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{m\times m}$ are orthogonal matrices, then $\|BAC\|_F = \|A\|_F$.*

Proof. This follows immediately from the fact that $\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i(A)}$ since the singular values are orthogonally invariant. $\square$

Property A.0.3. *If A is a rank-one matrix, then,*

$$\|A\|_F = \|A\|_{tr} = \|A\|_{sp}.$$

Proof. Note that the Frobenius norm, the trace norm, and the spectral norm are the ℓ_2, ℓ_1, and ℓ_∞ norms on the vector of singular values. Since A is rank one, there is only one singular value and all these norms agree on $\mathbb{R}$. $\square$

Appendix B

Omitted Proofs

B.1 Proof of Theorem 2.6.1

The proof will require the following Lemma from [5].

Lemma B.1.1. *[5] Let $\{f_s\}_{\mathcal{S}}$ be a collection of functions indexed by some set $\mathcal{S}$ and define $f(x) = \sup_{s \in \mathcal{S}} f_s(x)$ where each f_s is convex and continuous on* $\mathbf{dom}(f)$*. Additionally, assume that the set $\mathcal{A} \coloneqq \{\beta : f_\beta(x) = f(x)\}$ is compact (in some metric), and the function $\alpha \mapsto f_\alpha$ is upper semi-continuous for each x, then,*

$$\partial f(x) = \mathbf{cl}\left\{\mathbf{conv}\left(\bigcup_{s : f_s(x) = f(x)} \partial f_s(x)\right)\right\}.$$

Proof. (Theorem 2.6.1) First, note that,

$$\underset{s:\|s\|\leq\delta}{\arg\min}\langle z, s\rangle = -\delta \underset{\hat{s}:\|\hat{s}\|\leq 1}{\arg\max}\langle -z, \hat{s}\rangle.$$

Thus by definition of the dual norm (2.1), if $s \in \arg\min_{s':\|s'\|\leq\delta}\langle z, s'\rangle$, then $\langle -z, s\rangle = -\delta\|z\|_*$.

Let $\mathcal{S} = \{s : \|s\| \leq 1\}$ and denote $h_s(x) = \langle s, x\rangle$. Then we can consider the dual norm as a maximum of functions h_s over the set $\mathcal{S}$. Hence, to find the values of s that achieve the maximum, we can utilize Lemma B.1.1. Then the subdifferential is just the convex hull of the union of the subdifferentials of active functions at x (the functions that attain the maximum at x).

We know that,

$$\partial\|z\|_* = \mathbf{cl}\left\{\mathbf{conv}\left(\bigcup_{s:h_s(z)=\|z\|_*} \partial h_s(z)\right)\right\}.$$

However, $\partial h_s(z) = s$, so we can simplify this set to,

$$\partial\|z\|_* = \bigcup_{s:h_s(z)=\|z\|_*} s$$

where we note that since h_s is linear, the set $\bigcup_{s:h_s(z)=\|z\|_*} s$ is already closed and convex.

We can now conclude that $\partial\|z\|_*$ is the convex hull of all s such that $\|s\| \leq 1$ and $\langle z, s\rangle = \|z\|_*$ and more specifically,

$$\partial\|z\|_* = \underset{s:\|s\|\leq 1}{\arg\max}\langle z, s\rangle.$$

$\square$