A REVIEW OF THE CONDITIONAL GRADIENT METHOD AND A FEW APPLICATIONS

TINA TORABI AND ALIREZA YAZDANI

Abstract. This study explores the application of the conditional gradient method and its various adaptations as effective optimization strategies in the areas of Matrix Completion and feature selection, which are crucial in machine learning, statistics, and signal processing. The report not only highlights the effectiveness of the Conditional Gradient Method in addressing these vital challenges but also offers an in-depth comparative analysis of its different variants.

1. Introduction

The rapid growth of data-driven applications in various fields has given rise to a variety of optimization problems that demand efficient and scalable solutions. Among these, Matrix Completion and feature selection problems play a pivotal role in machine learning, statistics, and signal processing. This report presents an exploration of the conditional gradient method and its variants as a powerful optimization technique for tackling both of these problems. First, consider a scenario where a matrix is given with only a subset of its entries known, and the task is to infer the missing entries. This problem, in its raw form, is ill-posed due to the absence of initial information about the matrix. However, matrices derived from real-world data frequently demonstrate inherent low-dimensional structures, offering a means to effectively constrain the solution space. In recommender systems, when two users share a common interest in certain products, it is probable that they will exhibit a similar level of interest when encountering a new product, reflecting the existence of a low-rank structure in the user-item interaction matrix. Low-rank matrix completion is a recurring problem in a variety of applications including recommendation systems [6], computer vision [10], and signal processing [8]. The Netflix Prize Challenge [1] serves as an essential example and a motivating application for the study of low-rank matrix completion problems. Launched by Netflix, this competition aimed to improve the accuracy of predictions about how much a user would enjoy a film based on their previous ratings. The underlying challenge was to fill in a massive, sparsely populated matrix representing user ratings for various movies. Each entry in this matrix corresponded to a user's rating for a particular movie, but a vast majority of these entries were missing, as no user had rated all the movies in the database. The question now arises: *is it feasible to complete a matrix with partially observed entries given its rank is small?* In Section 4, we will rigorously formulate the low-rank matrix completion problem and discuss a well-established heuristic for solving it, as proposed by Fazel et al [4]. We will then compare two specific approaches in solving this problem using the conditional gradient method, by presenting and evaluating numerical results.

Now consider a complex machine learning project with a dataset that encompasses a wide array of features. In this high-dimensional data space, not all features contribute equally to the predictive accuracy of the model. Some may be crucial, while others could be redundant or even detrimental to the model's performance. This is where the concept of feature selection becomes indispensable. Among various feature selection techniques, Lasso (Least Absolute Shrinkage and Selection Operator), introduced by Tibshirani [14], excels in scenarios with high-dimensional datasets. In Section 3, we will address this problem using the conditional gradient algorithm and provide numerical results to illustrate its practical applications.

Notation. Let $\mathbb{R}^{m\times n}$ denote the space of all $m \times n$ real matrices. For a matrix $X \in \mathbb{R}^{m\times n}$, X_{ij} represents the entry in the i-th row and j-th column. Throughout this paper, $\langle X, Y\rangle$ denotes the inner product between two matrices X and Y, defined as

$$\langle X, Y\rangle = \mathrm{Tr}(X^\top Y) \;=\; \sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}Y_{ij}$$

, where $X^\top$ represents the transpose of X. The Frobenius norm, denoted as $\|\cdot\|_F$, is the norm associated with this inner product, $\|X\|_F \;=\; \mathrm{Tr}(X^\top X)$.

The i-th largest singular value of X is represented by $\sigma_i(X)$ which is equal to the square root of the i-th largest eigenvalue of $XX^\top$. The largest singular value of a matrix, denoted as $\|X\|$, is its operator norm or induced 2-norm, which can be represented as $\|X\| := \sigma_1(X)$. The rank of X, denoted as r, corresponds to the number of non-zero singular values of X. Finally, the sum of the singular values of X defines its nuclear norm, noted as $\|X\|_* := \sum_{i=1}^{r}\sigma_i(X)$, which is also known under various terminologies, including the Schatten 1-norm, Ky Fan r-norm, and trace class norm.

1.1. Preliminaries.

We will first review some preliminaries needed to establish the methods we mention later.

Definition 1 (Convex envelope). *Given a function $f : \mathcal{C} \to \mathbb{R}$, a function g is called the (largest) convex envelope of f if and only if g is convex and*

$$g(x) \le f(x) \quad \forall x \in \mathcal{C}.$$

That is, g is a convex function below f that is the closest point-wise to f.

In the realm of vector spaces, consider a vector x in $\mathbb{R}^n$. We can express this vector as a linear combination of a set of base vectors, or 'atoms', from a subset $\mathcal{A}$ in $\mathbb{R}^n$.

This expression takes the form:

$$x = \sum_{a \in \mathcal{A}} c_a a, \quad c_a \geq 0 \quad \forall a \in \mathcal{A}.$$

Here, each c_a is a non-negative scalar signifying the extent to which each atom a contributes to the composition of x.

Let $\hat{\mathcal{A}} := \text{clconv}(\mathcal{A} \cup \{0\})$ denote the closed convex hull of an atomic set $\mathcal{A} \subseteq \mathbb{R}^n$ adjoined with the origin. These atoms are crucial in representing x as a positive linear combination. We define the gauge function for this purpose:

$$\begin{aligned}\gamma_{\hat{\mathcal{A}}}(x) &= \inf_{c_a} \left\{ \sum_{a \in \hat{\mathcal{A}}} c_a \mid x = \sum_{a \in \hat{\mathcal{A}}} c_a a,\ c_a \geq 0\ \forall a \in \hat{\mathcal{A}} \right\} \\ &= \inf\{\lambda \geq 0 | x \in \lambda \hat{\mathcal{A}}\}\end{aligned}$$

This function calculates the minimum total weight needed for such a decomposition of x. Atoms that have a positive influence in this minimal representation are considered significant.

Definition 2 (Support Set). *A subset $\mathcal{S}_{\hat{\mathcal{A}}}(x)$ of $\hat{\mathcal{A}}$ is termed a 'support set' for x relative to $\hat{\mathcal{A}}$ if every atom a in this subset contributes positively in the atomic decomposition of x, as in:*

$$\gamma_{\hat{\mathcal{A}}}(x) = \sum_{a \in \mathcal{S}_{\hat{\mathcal{A}}}} c_a, \quad x = \sum_{a \in \mathcal{S}_{\hat{\mathcal{A}}}} c_a a,$$

where $c_a > 0$ for all $a \in \mathcal{S}_{\hat{\mathcal{A}}}(x)$.

In optimization problems, atoms with larger coefficients in the decomposition signify greater importance in the minimization process. Chen et al. and Chandrasekaran et al. have explored atomic decompositions in contexts like sparse signal decomposition and solving linear inverse problems, respectively. The gauge function is a key element in defining a convex optimization problem for recovering solutions from limited observations. The gauge function can be equated to the Minkowski functional related to the convex hull formed by the atomic set $\mathcal{A}$ and the zero vector. The relationship between the gauge and support functions to the set $\mathcal{A}$ is dual, governed by the *polar inequality*.

Definition 3 (Alignment). *Two vectors x and z in $\mathbb{R}^n$ are 'aligned' with respect to the atomic set $\mathcal{A}$ if they satisfy the polar inequality as an equality.*

In the analysis of atomic decompositions, significant atoms in a decomposition of x are those contained in the set of 'exposed atoms' by a vector z. The convex geometry underlying this concept helps in identifying atoms important for the decomposition.

The nuclear norm of a matrix is derived using the atomic set $\mathcal{A} = \{uv^\top | \|u\|_2 = \|v\|_2 = 1\}$, which comprises normalized rank-1 matrices of dimensions $n \times m$. For matrices X and Z, the norms are defined as:

$$\gamma_{\mathcal{A}}(X) = \|X\|_*, \quad \text{and} \quad \sigma_{\mathcal{A}}(Z) = \sigma_{\max}(Z).$$

Here, $\|X\|_*$ represents the nuclear norm of X (the sum of its singular values), and $\sigma_{\max}(Z)$ is the spectral norm of Z (its maximum singular value). This atomic representation aligns with the idea that the nuclear norm encourages matrices to have a lower rank, consistent with sparsity in the context of rank-1 matrices.

The trace inner product between matrices X and Z is defined as $\langle X, Z \rangle := \text{trace}(X^\top Z)$. The condition for alignment, $\langle X, Z \rangle = \|X\|_1 \cdot \|Z\|_\infty$, is satisfied when X and Z share a synchronized singular value decomposition (SVD). For instance, if X has a rank of r, then their SVDs can be expressed as:

$$X = \sum_{i=1}^{r} c_i u_i v_i^\top, \quad \text{and} \quad Z = \sum_{i=1}^{\min\{m,n\}} s_i u_i v_i^\top,$$

where the singular values are ordered such that:

$$\begin{aligned}&c_1 \geq \cdots \geq c_r > 0, \quad \text{and} \\ &s_1 = \cdots = s_d > s_{d+1} \geq \cdots \geq s_{\min\{m,n\}} \geq 0.\end{aligned}$$

Under this framework, the support set for X and the exposed atoms for Z are identified as:

$$\begin{aligned}\mathcal{S}(X) &= \{u_1 v_1^\top, \ldots, u_r v_r^\top\}, \\ \mathcal{E}_{\mathcal{A}}(Z) &= \{u_1 v_1^\top, \ldots, u_d v_d^\top\}.\end{aligned} \tag{1.1}$$

The inclusion relation, which determines the support as a subset of the exposed atoms, suggests that $d \geq r$. This implies that the singular vectors in Z associated with the singular values $s_1, \ldots, s_d$ encompass the singular values of X. Adding Thm 5.1 from MPF to show why we can solve the dual!

2. Conditional gradient methods

The Conditional Gradient (CG) method, also known as the Frank-Wolfe algorithm, is a technique used for solving optimization problems with a convex objective function and a convex feasible region. Unlike traditional gradient descent methods that require a projection step to ensure the solution remains within the feasible region, the CG method operates by linearizing the objective function at the current point and then moves towards a solution at the vertex of the feasible region that minimizes this linear approximation. In what follows, we will review the CG method, and the algorithm is expressed using the notion of Alignment covered in [2].

Formally, given a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and $\hat{\mathcal{A}}$, the CG method seeks to solve the following problem:

$$\underset{x \in \hat{\mathcal{A}}}{\text{minimize}}\, f(x). \tag{2.1}$$

Algorithm 1 summarizes the iterates of the CG method. Initiating with an arbitrary atom $x^{(0)} \in \hat{\mathcal{A}}$, at each iteration k, the algorithm computes the gradient $\nabla f(x^{(k)})$ at the current point $x^{(k)}$. It then seeks an atom (or a convex combination of atoms) $a^{(k)}$ on the exposed face of $\hat{\mathcal{A}}$, denoted by $\mathcal{F}_A(z^{(k)})$, which maximizes the descent function given by the inner product $\langle a^{(k)}, -\nabla f(x^{(k)}) \rangle$. This step selects an atom $a^{(k)}$ which is $\mathcal{A}$-aligned with $z^{(k)}$. The solution is updated by moving from $x^{(k)}$ towards $a^{(k)}$, employing a convex combination of the two points. This iterative process continues, until the desired convergence is met.

From the perspective of atomic alignment, the Linear Minimization Oracle (LMO) in line 4, picks out an atom

Algorithm 1 The CG method

1: **Input:** $x^{(0)} \in \mathcal{A}, \epsilon \geq 0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad z^{(k)} = -\nabla f(x^{(k)})$
4: $\quad a^{(k)} \in \mathcal{F}_{\mathcal{A}}(z^{(k)})$
5: $\quad$ **if** $\langle a^{(k)} - x^{(k)}, z^{(k)} \rangle < \epsilon$ **then**
6: $\quad\quad$ **break** ▷ break if optimal
7: $\quad$ **end if**
8: $\quad x^{(k+1)} = \theta^{(k)} a^{(k)} + (1 - \theta^{(k)}) x^{(k)}$ ▷ $\theta^{(k)} \in (0, 1)$
9: **end for**
10: **return** $x^{(k)}$

$a^{(k)}$ that aligns with $z^{(k)}$. Specifically, it holds that

$$\langle a^{(k)}, z^{(k)} \rangle = \sigma_A(z^{(k)}) = \gamma_A(a^{(k)}) \cdot \sigma_A(z^{(k)}), \tag{2.2}$$

where the second equality is a consequence of the fact that $a^{(k)}$ is an element of $\mathcal{A}$, hence $\gamma_{\mathcal{A}}(a^{(k)}) = 1$. Line 8 of Algorithm 1 merges the atom $a^{(k)}$ into the cumulative collection of atoms selected during previous iterations. Therefore, the updated iteration $x^{(k)}$ is a composite of these atoms. Thus, the most recent iterate $x^{(k)}$ resides within the convex hull of the faces exposed up to the k-th iteration: $x^{(k)} \in \sum_{i=1}^{k} \bar{\theta}(i) \mathcal{F}_A(z^{(i)})$. In a hypothetical, perfectly efficient execution of the algorithm, the series of exposed faces $\mathcal{F}_A(z^{(k)})$ would be expanding, that is, $\mathcal{F}_A(z^{(k)}) \subseteq \mathcal{F}_A(z^{(k+1)})$, eventually converging to the optimal face $\mathcal{F}_A(z^*)$, where $z^* := -\nabla f(x^*)$. However, in real-world applications, one should not expect such a streamlined process, and the algorithm might accumulate many atoms that are not necessarily related to the optimal face.

We now briefly mention an important convergence result for the CG method from [7].

Theorem 2.1. *(Primal convergence of the Frank-Wolfe algorithm). Let f be an L-smooth convex function and let C be a compact convex set of diameter D. Consider the iterates of Algorithm 1. Then the following holds:*

$$f(x_t) - f(x^*) \leq \frac{2LD^2}{t+2},$$

and hence for any accuracy $\varepsilon > 0$ we have $f(x_t) - f(x^*) \leq \varepsilon$ for all $t \geq \frac{2LD^2}{\varepsilon}$.

3. The Lasso problem

The Lasso problem, short for Least Absolute Shrinkage and Selection Operator, is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces. It is particularly useful in scenarios where we have more predictors than observations or when a few predictors are significant while most are not. The Lasso problem addresses these challenges by shrinking the less important feature's coefficients to zero, effectively performing feature selection.

Formally, consider a data matrix $A \in \mathbb{R}^{n \times p}$. The target variable is represented by $b \in \mathbb{R}^n$. The Lasso problem introduces a regularization parameter λ, which controls the degree of shrinkage applied to the coefficients. The problem is of the form (2.1) and can be expressed as:

$$f(x) = \frac{1}{2}\|Ax - b\|^2 \quad \text{s.t.} \quad \|x\|_1 \leq \lambda \tag{3.1}$$

One way of choosing $\theta^{(k)}$ in line 8 of Algorithm 1 is to set it by line search as follows:

$$\theta^{(k)} = \arg\min_{\theta \in [0,1]} f(x^{(k)} + \theta(a^{(k)} - x^{(k)})). \tag{3.2}$$

We can easily compute it:

$$f(x^{(k)} + \theta(a^{(k)} - x^{(k)})) = \frac{1}{2}\|\theta A(a^{(k)} - x^{(k)}) + Ax^{(k)} - b\|^2 , \tag{3.3}$$

and taking the derivative with respect to γ we can easily deduce that the following step size solves the line search problem:

$$\theta^{(k)} = \frac{\omega_{(k)}^T (b - Ax)}{\|\omega_{(k)}\|^2} , \tag{3.4}$$

where $\omega_{(k)} = A(a^{(k)} - x^{(k)})$.

We have implemented Algorithm 1 to solve the lasso problem as described in Equation (3.1). Our implementation was carried out using the `Python` programming language due to its convenient access to machine learning datasets through libraries like scikit-learn. We conducted experiments using two distinct datasets: one is a synthetic random regression dataset, and the other is the Boston Housing Dataset, which contains housing-related data collected by the U.S. Census Service for the Boston, Massachusetts area. In Figure 3.1, we observe the behavior of the CG gap over the course of multiple iterations for two values of $\lambda = 0.3$ and $\lambda = 0.5$. While the gap is not monotonically decreasing, Algorithm 1 is making progress in minimizing the objective function. Despite some fluctuations, the overall pattern suggests that the algorithm is effectively converging towards an optimal solution.

4. Matrix Completion

Consider $\Omega \in \mathbb{R}^{m \times n}$ is a binary matrix, i.e., a matrix whose entries are either zero or one. Next, assume an observation matrix $B \in \mathbb{R}^{m \times n}$ in which only those entries are known whose corresponding entries in Ω are one. The objective is to estimate the complete B with the assumption that we have an upper bound on its rank, denoted by $r \leq \min\{m, n\}$. This matrix completion problem can be formulated as an optimization problem constrained by rank:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ X - B\|^2 \quad \text{s.t.} \quad \text{rank}(X) \leq r. \tag{4.1}$$

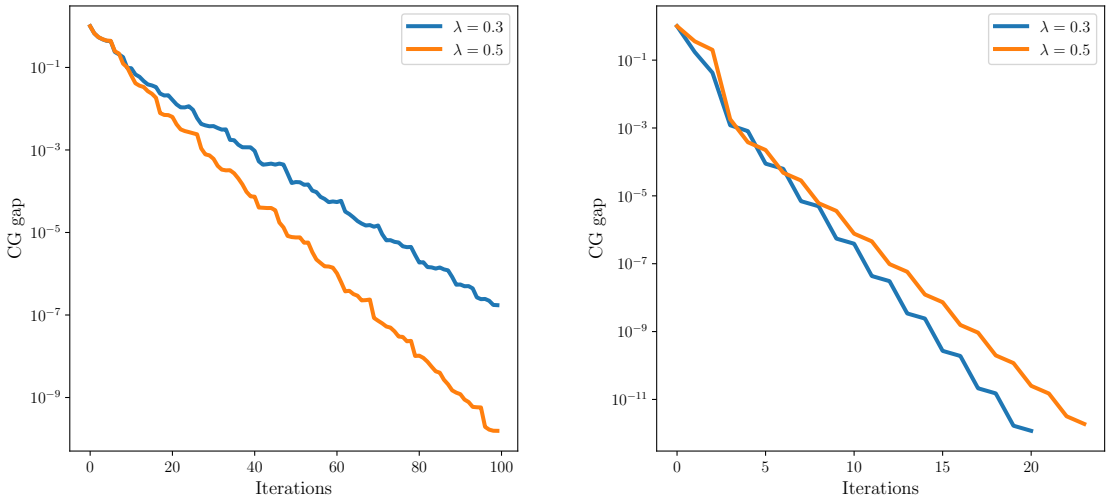
However, due to the NP-hard nature of rank minimization [10], this formulation proves to be intractable. We instead, consider the convex surrogate for the rank constraint [3] as follows:

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ X - B\|^2 \quad \text{s.t.} \quad \|X\|_* \leq \tau. \tag{4.2}$$

Theorem (Fazel02)[4]**.** Consider the function $f(X) = \text{rank}(X)$ defined over the set

$$S := \{X \in \mathbb{R}^{m \times n} \mid \|X\|_2 \leq 1\},$$

the largest convex envelope of f is the nuclear norm $\|X\|_*$. This theorem is applicable specifically to matrices X within



(A) Random regression dataset

(B) Boston Housing dataset

FIGURE 3.1. Optimality gap of the CG method applied to the lasso problem (3.1) for different λ values. The algorithm's performance is assessed using both randomly generated data (left) and the Boston Housing dataset (right).

the unit sphere. It does not provide information for matrices X outside this sphere. Indeed, for such matrices, the value of the convex envelope tends towards infinity.

For an extended set $S' := \{X \in \mathbb{R}^{m\times n} \mid \|X\|_2 \leq M\}$ with $M > 0$, a scaling approach can be employed. In this scenario, the convex envelope of f on S' is given by $\frac{1}{M}\|X\|_*$. The foundation for proving this theorem lies in the concept of the convex conjugate.

Adjusting the parameter τ controls the rank of a solution X to problem (4.2). With adequate data and an appropriate choice of τ, it is reasonable to assume that any solution X will closely approximate the target matrix X. Adjusting the parameter τ controls the rank of a solution X to problem (4.2). With adequate data and an appropriate choice of τ, it is reasonable to assume that any solution X will closely approximate the target matrix X. The convex optimization problem as described in Equation (4.2) is an appropriate model for matrix completion, particularly when the number of observations $|E|$ is approximately $\tilde{O}(r(m+n))$. Here, $\tilde{O}$ excludes logarithmic factors. For further information, refer to the study detailed in [11].

Convex optimization methods have become fundamental tools for addressing the low-rank matrix completion problem. Gradient-based optimization methods have been adapted to tackle matrix completion by incorporating projections onto the set of matrices that agree with the observed entries. Algorithms such as gradient descent or conjugate gradient are employed where the gradient is taken with respect to a differentiable loss function measuring the discrepancy between the observed matrix entries and the model predictions [15]. The Alternating Direction Method of Multipliers (ADMM) is a powerful algorithm that decomposes an optimization problem into simpler subproblems, which can be solved iteratively. For matrix completion, ADMM can be used to split the problem into a nuclear norm minimization step and a data consistency step, solving each subproblem in an alternating fashion [13]. Robust PCA extends matrix completion methods to handle outliers by decomposing the observed matrix into a low-rank component and a sparse error matrix. This is particularly useful when the observed data is corrupted with gross errors or outliers [16]. While the focus here is on convex methods, it's worth noting recent advancements in non-convex optimization approaches. These methods often operate directly on the low-rank factors of the matrix and can achieve better scalability and practical performance in some cases [12].

4.1. **DualCG for Matrix Completion.** Using Algorithm 1 for solving the matrix completion problem 4.1 results in the formation of atoms $a(k)$ through the outer products of unit-norm vector pairs $(u(k), v(k))$, chosen such that $\langle u, Z^{(k)}v\rangle = \tau \cdot \sigma_{\max}(Z^{(k)})$. The primary challenge in this methodology is the handling of these atoms. One approach is to accumulate these atoms into a dense matrix $X(k)$ at every iteration, or alternatively, they can be stored as a sequence of vector pairs $\{(u(i), v(i))\}_{i=1}^{k}$. Both of these methods, unfortunately, require a lot of memory, which makes them less feasible for solving large-scale problems.

The "storage-efficient" algorithm we will discuss now is a specialized dual CG method (DualCG) tailored for matrix-completion problems presented in 2. This variant is similar to the methodology proposed by Yurtsever et al [17] called SketchyCG which we will review in the next subsection.

The DualCG method sidesteps the explicit use of the primal variable within the CG iteration cycle. Instead of finding the primal optimal solution x^*, the DualCG attempts to find the dual optimal solution $z^* = -\nabla f(x^*)$. The primal solution x^* is later reconstructed in Line 16,

leveraging the $\mathcal{A}$-alignment between x^* and z^* in accordance with Theorem 5.1 [2].

Initially, the algorithm takes as input a binary matrix Ω, which specifies the observed entries of the matrix to be completed, and a matrix B containing these observed values. The integer ℓ denotes the target rank for the approximation. The algorithm starts by initializing matrix $R^{(k)}$ through the Hadamard product of Ω and B, capturing the observed values. During each iteration indexed by k, the algorithm updates the matrix $Z^{(k)}$ using the Hadamard product of Ω and $R^{(k)}$. Subsequently, it performs a singular value decomposition (SVD) on $Z^{(k)}$ to extract the leading singular vectors u and v. A rank-one update, scaled by a factor τ and denoted as τuv^T, is then calculated. The algorithm proceeds to compute the difference $\Delta R^{(k)}$ between the rank-one update and the current approximation matrix $Q^{(k)}$. The inner product $\rho^{(k)}$ between $\Delta R^{(k)}$ and $R^{(k)}$ is calculated as the optimality gap. If the convergence criterion is not met, the algorithm determines the step size $\theta^{(k)}$ through an exact line search that minimizes the Frobenius norm of $\Delta R^{(k)}$. The matrices $R^{(k)}$ and $Q^{(k)}$ are then updated by applying the computed step size to $\Delta R^{(k)}$.

Once the iterative process concludes, a final SVD is performed on $Z^{(k)}$ to obtain the top ℓ singular vectors, forming the matrices U, V, and Σ. Finally, Line 16 represents the recovery of the primal solution. Note that the optimization problem in Line 16 can be rewritten as **TINA:is this in the right place????**

$$S = \arg\min_S \{\frac{1}{2}\|\mathbf{UV}S - B\|\} \tag{4.3}$$

Algorithm 2 The DualCG method for problem (4.2).

1: **Input:** Ω, B, ℓ
2: $R^{(k)} \leftarrow \Omega \circ B$; $Q^{(k)} \leftarrow 0$
3: **for** $k = 1, 2, \ldots$ **do**
4: $\quad Z^{(k)} \leftarrow \Omega \circ R^{(k)}$
5: $\quad (u, v) \leftarrow \text{svds}(Z^{(k)}, 1)$
6: $\quad \Delta R^{(k)} \leftarrow \Omega \circ (\tau uv^T) - Q^{(k)}$
7: $\quad \rho^{(k)} \leftarrow \langle \Delta R^{(k)}, R^{(k)} \rangle$ ▷ Optimality gap
8: $\quad$ **if** $\rho^{(k)} < \varepsilon$ **then**
9: $\quad\quad$ **break**
10: $\quad$ **end if**
11: $\quad \theta^{(k)} \leftarrow \min\{1, \rho^{(k)}/\|\Delta R^{(k)}\|_F^2\}$
12: $\quad R^{(k+1)} \leftarrow R^{(k)} - \theta^{(k)} \Delta R^{(k)}$
13: $\quad Q^{(k+1)} \leftarrow Q^{(k)} + \theta^{(k)} \Delta R^{(k)}$
14: **end for**
15: $(U, \Sigma, V) \leftarrow \text{svds}(Z^{(k)}, \ell)$ ▷ Top ℓ SVs
16: $S \leftarrow \arg\min_S \left\{ \frac{1}{2}\|\Omega \circ USV^T - B\|_2^2 \mid \text{tr}(S) \le \tau, S \ge 0 \right\}$
17: **return** (U, S, V)

4.2. **SketchyCG Method for Matrix Completion.** In this subsection, we summarize a randomized sketching algorithm, called SketchyCG, proposed by Yurtsever et al [17] that allows us to compute a rank-r approximation of B with optimal storage. The key steps of this algorithm are as follows:

(1) **Initialization:** The algorithm begins by initializing two random matrices $\Omega \in \mathbb{R}^{n\times k}$ and $\Psi \in \mathbb{R}^{m\times \ell}$, where k and ℓ are dimensions chosen based on the target rank r of the approximation. The initial guess for the solution, z_0, is set to zero.
(2) **Iteration:** At each iteration, the algorithm computes an update direction using the Lanczos or randomized SVD methods to find the maximum singular vector pair (u_t, v_t) of the gradient of the objective function. This pair is used to compute an update step h_t.
(3) **Learning Rate and Updates:** A learning rate η_t is computed, which decreases over iterations. The current solution estimate z_t, and the sketches Y_t and W_t are updated using h_t and the singular vectors.
(4) **Stopping Criterion:** The algorithm continues until the stopping criterion is met, which is based on the suboptimality ϵ of the current solution estimate.
(5) **Reconstruction:** Once the algorithm terminates, it reconstructs a rank-r approximation of the solution from the final sketches Y_t and W_t.

The memory requirements for storing the dual variable z_t, together with the random matrices (Ω, Ψ) and the sketch (Y, W), are on the order of $\Theta(d + r(m + n))$. This implies that there is no necessity to fully create an $m \times n$ matrix at any phase of the calculation process which showcases the memory efficiency of this approach.

5. Results

We now provide numerical results comparing the DualCG and SketchyCG methods for solving the matrix completion problem. We implemented the DualCG in both `MATLAB` and `Julia` and implemented SketchyCG in `Julia`. For a range of problem sizes where $m = n$, we create the binary mask Ω with a 10% non-zero element density. We then form the observation matrix B by element-wise multiplication of Ω with $(UV^\top + 0.1 \cdot N)$, where $U \in R^{m\times r}$ and $V \in R^{n\times r}$ are matrices, and N is a matrix with elements independently and identically distributed (i.i.d.) according to a standard Gaussian distribution. We set the "true rank" r to be $\lceil m/100 \rceil$. In all our experiments, the top singular value of the final dual solution estimate $Z^{(k)}$ had a unique multiplicity. As a result of 1.1, Line 12 of DualCG Algorithm 2 is made trivial. Table 5.1 presents a comparative analysis of the performance metrics of the DualCG and SketchyCG methods for different problem sizes. Since the dataset is random, we present the average of five experiments for each problem size. We quantify the performance using the Normalized Mean Absolute Error (NMAE) used in [9], defined as:

$$\text{NMAE} = \frac{1}{(B_{\max} - B_{\min})|\Omega|} \sum_{(i,j)\in\Omega} |X_{i,j}^{\text{method}} - B_{i,j}|$$

where $B_{\max}$ and $B_{\min}$ are the lower and upper bounds for the observed data. Our observations in 5.1 indicate that, with a fixed number of iterations, DualCG tends to outperform SketchyCG in minimizing the error for smaller

Algorithm 3 SketchyCG for problem (4.2)

Input: Data for (4.2); ϵ; target rank r
Output: Rank-r approximate solution $\hat{X} = U\Sigma V^*$ of (4.2) in factored form

1: **function** SketchyCGM
2: SketchInit(m, n, r) ▷ Initialize Sketch
3: $z \leftarrow 0$
4: **for** $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
5: $(u, v) \leftarrow \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(z)))$
6: $h \leftarrow \mathcal{A}(-\alpha u v^*)$
7: **if** $\langle z - h, \nabla f(z)\rangle \leq \epsilon$ **then**
8: **break** ▷ Stop
9: **end if**
10: $\eta \leftarrow 2/(t+2)$
11: $z \leftarrow (1-\eta)z + \eta h$
12: SketchCGMUpdate$(-\alpha u, v, \eta)$
13: **end for**
14: $(U, \Sigma, V) \leftarrow$ SketchReconstruct( )
15: **return** (U, Σ, V)
16: **end function**

Methods for Sketch object

17: **function** SketchInit$(m,\ n,\ r)$
18: $k \leftarrow 2r+1$ and $\ell \leftarrow 4r+3$
19: $\Omega \leftarrow \texttt{randn}(n, k)$ and $\Psi \leftarrow \texttt{randn}(\ell, m)$
20: $Y \leftarrow \texttt{zeros}(m, k)$ and $W \leftarrow \texttt{zeros}(\ell, n)$
21: **end function**
22: **function** SketchCGMUpdate$(u,\ v,\ \eta)$
23: $Y \leftarrow (1-\eta)Y + \eta u(v^*\Omega)$
24: $W \leftarrow (1-\eta)W + \eta(\Psi u)v^*$
25: **end function**
26: **function** SketchReconstruct( )
27: $Q \leftarrow \texttt{orth}(Y)$ ▷ Orthobasis for range of Y
28: $B \leftarrow (\Psi Q)\backslash W$ ▷ Solve family of least-squares
29: $(U, \Sigma, V) \leftarrow \texttt{svds}(B, r)$ ▷ Top r SVs
30: **return** (QU, Σ, V)
31: **end function**

matrices. However, as the matrix size increases, the performance gap between the two methods diminishes, showing a convergence in their efficacy. This might suggest that DualCG reaches convergence in terms of optimality gap with fewer iterations. Additionally, SketchyCG demonstrates a speed advantage over DualCG, particularly noticeable with larger matrices, suggesting a computational efficiency that scales better with size. This could be attributed to the optimization problem addressed at Line 12 in Algorithm 2. Considering its superior performance in terms of error and computation time, SketchyCG emerges as the most promising option. It is essential to focus on the convergence speed, specifically the rate at which the optimality gap decreases. The graph in Figure **??** Figure HERE PLZ depicts the relationship between the optimality gap and the number of iterations for the $m = 250$ problem outlined in Table 5.1. Remarkably, SketchyCG displays more favorable performance during the initial iterations, whereas the DualCG method showcases superior results in the later steps. In the context of a substantial

TABLE 5.1. Performance of the DualCG and SketchyCG methods for the random matrix-completion problem. The estimated rank of the final solution is determined by identifying the minimum number of singular values required to encompass 90% of its Frobenius norm. The time measurement is expressed in seconds.

size	Dual CG			Sketchy CG		
$m = n$	rank	time	NMAE	rank	time	NMAE
100	1	0.21	0.0282	1	0.3	0.0710
250	1	0.83	0.0526	1	0.6	0.0460
500	1	3.15	0.0572	1	1.3	0.0480
1,000	1	13.17	0.0608	1	4.2	0.0520
2,000	1	42.51	0.0670	1	13.7	0.0630

problem like the Netflix problem, which involves millions of users and thousands of movie titles, the convergence speed emerges as a crucial factor for deciding on the algorithm. Therefore, we recommend utilizing the DualCG method, as it demonstrates superior results in the optimality gap.

As the central focus of our work, we delve deeper into assessing the performance of the DualCG method using the aforementioned random dataset. Specifically, our experiments aim to understand the impact of problem size, the nuclear norm constraint τ, and the rank of the final solution ℓ. In these cases, the algorithm is executed for 10,000 iterations in each scenario. We depict the variation of NMAE with respect to the nuclear norm constraint τ for different problem sizes and the final rank of the solution ℓ in Figure 5.1. As we observe, NMAE does not change monotonically with τ. The impact of τ on the solution has two primary aspects: firstly, it must be small in Line 6 of Algorithm 2 to prevent substantial changes in ΔR during each update. This ensures a sufficiently small optimality gap for convergence within a limited number of iterations. Put differently, at each iteration, we introduce an atom to our solution, but this atom may not necessarily be in the support set of the optimal solution. Therefore, we aim to avoid giving the atom a large weight, making it difficult to ignore in future iterations. Secondly, τ must be large enough to recover all the singular values of the observed data B, presenting a challenge in fine-tuning this parameter. For the case when $m = 100$, where the true rank of the problem is one, there is no need to explore different ℓ values. However, for the problem size with $m = 500$, where the true rank is five, we can vary ℓ from one to five for further investigation. As observed, increasing ℓ leads to lower NMAE due to the approximation of observed data with a solution closer to its original true rank. Increasing ℓ does not affect the dual problem itself but only influences Line 12 in the DualCG algorithm, where another optimization problem is solved to recover the primal solution. Notably, larger ℓ values make it more challenging to solve the mentioned optimization problem.

The algorithm originally outputs matrices, and visualizing these outputs to ensure the proper functioning of the algorithms poses some difficulty. Therefore, we attempt

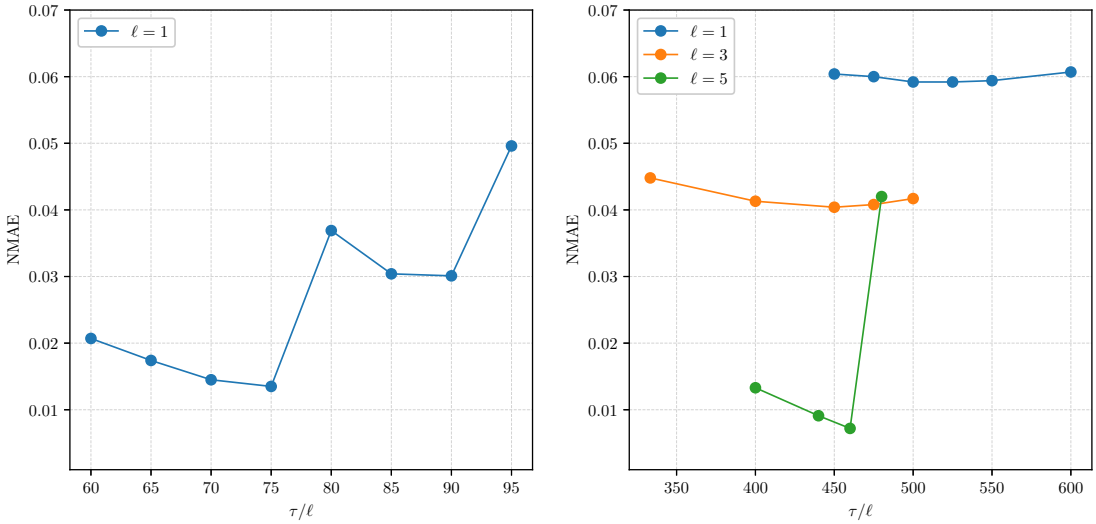


FIGURE 5.1. NMAE against the nuclear norm constraint τ for different ℓ values in the DualCG algorithm. τ is scaled with ℓ solely for aesthetic reasons. **Left:** The problem size is $m = 100$. **Right:** The problem size is $m = 500$.

to reconstruct corrupted images as an alternative evaluation method. Initially, we reduce the rank of the image by computing its Singular Value Decomposition (SVD) and retaining only a selected number of top singular values. Subsequently, we introduce sparsity by removing some pixels from the image. Finally, we apply the DualCG method to the corrupted image for reconstruction. Figure 5.2 displays an original low-rank (20) image on the left, the sparse image with only 60% observed data in the middle, and the reconstructed low-rank image on the right. The reconstruction is performed using the DualCG method with parameters $\tau = 90$ and $\ell = 20$.

We finally test the DualCG and sketchyCG algorithms on some real-world data. The dataset we use is the data called MovieLens 10K datasets obtained from [5] for ??? users rating ??? movie titles from 1 to 5. The sparsity of this data set is ???. Figure 5.3 shows optimality gap convergence results applied to MovieLens 10K dataset, fixing the hyperparameters $\ell = 1$ and $\tau = 20$. The results show that using DualCGM, the optimality gap converges in much fewer iterations in comparison to SketchyCG. Although the trend in SketchyCG seems to be much smoother. The NMAE for the DualCG and SketchyCG were 0.0512 and 0.0534 respectively.

We also applied the DualCG algorithm to the extension of MovieLens 10K dataset, called MovieLens 100K dateset. This dataset has The only

6. Conclusion

In this work, we have provided a comprehensive review of the conditional gradient method, commonly referred to as the Frank-Wolfe algorithm, and its variants. These methods have been shown to be particularly robust in addressing a variety of optimization challenges. Specifically, we focused on two significant applications: the lasso problem, which is central to sparse regression in statistics, and the matrix completion problem, a key component in collaborative filtering and recommendation systems.

The conditional gradient method, with its inherent simplicity and efficiency, has proven to be highly effective for solving lasso problems, especially in high-dimensional settings where traditional methods struggle.

Similarly, in the realm of matrix completion, the conditional gradient method has demonstrated remarkable performance. The ability of the conditional gradient method to handle large-scale matrices efficiently makes it an attractive choice for such applications.

Furthermore, we explored storage-efficient variants of the conditional gradient method. Our comparative analysis revealed that these storage-efficient methods not only maintain the efficacy of the standard algorithm but also significantly reduce memory usage, making them ideal for large-scale problems.

Additionally, our study delved into the convergence properties of these methods. We demonstrated that the conditional gradient method offers guaranteed convergence rates, which is crucial for ensuring the reliability and predictability of the algorithm in practical applications.

In summary, the conditional gradient method and its variants stand out as powerful tools in optimization. Their ability to handle large-scale problems with efficiency and their robustness in various applications underscore their importance in both theoretical research and practical implementation. Future work may involve extending these methods and exploring their integration with other optimization techniques.

References

[1] James Bennett and Stan Lanning. The netflix prize. 2007.

[2] Zhenan Fan, Halyun Jeong, Yifan Sun, and Michael P. Friedlander. Atomic decomposition via polar alignment: The geometry of structured optimization. *Foundations and Trends in Optimization*, 3(4):280–366, 2020.

[3] M. Fazel, H. Hindi, and S.P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 6, pages 4734–4739 vol.6, 2001.

[4] Maryam Fazel. *Matrix Rank Minimization with Applications*. Phd thesis, Stanford University, Redwood City, 2002.

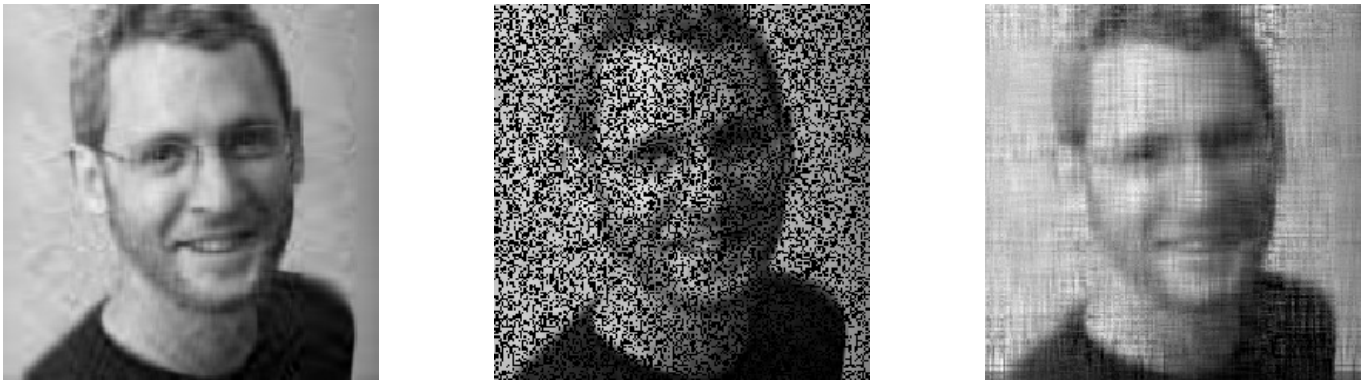


FIGURE 5.2. Low rank image reconstruction using the DualCG method. **Left:** Original low rank (= 20) image. **Middle:** Sparse Image with 40% of entries missing. **Right:** Reconstructed image with $\tau = 90$ and $\ell = 20$.

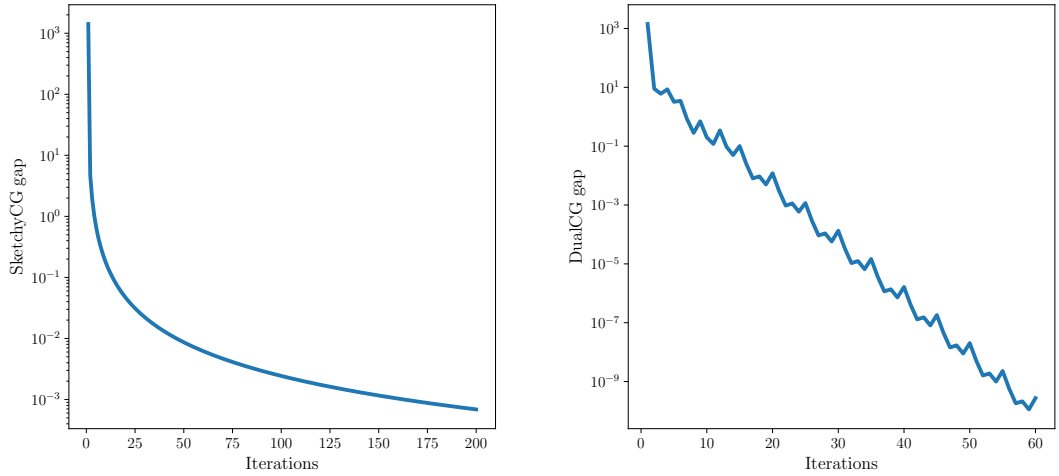


FIGURE 5.3. Optimality gap of the SketchyCG and DualCG algorithms when applied to the matrix completion problem using the MovieLens 10K dataset.

[5] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[6] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013.

[7] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 427–435, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[8] Xiao Peng Li, Lei Huang, Hing Cheung So, and Bo Zhao. A survey on matrix completion: Perspective of signal processing. *arXiv: Signal Processing*, 2019.

[9] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128:321–353, 2009.

[10] Behnaz Rezaei and Sarah Ostadabbas. Background subtraction via fast robust matrix completion. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1871–1879, 2017.

[11] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[12] Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via non-convex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016.

[13] Rahman Taleghani and Maziar Salahi. An admm-factorization algorithm for low rank matrix completion. *Applications and Applied Mathematics: An International Journal (AAM)*, 14(2):Article 34, 2019.

[14] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[15] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. 2009.

[16] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[17] Alp Yurtsever, Madeleine Udell, Joel Tropp, and Volkan Cevher. Sketchy Decisions: Convex Low-Rank Matrix Optimization with Optimal Storage. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1188–1196. PMLR, 20–22 Apr 2017.

Tina Torabi, Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, British Columbia, Canada

Email address: torabit@math.ubc.ca

Alireza Yazdani, Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, British Columbia, Canada

Email address: ayazdani@math.ubc.ca