BILGO: Bilateral Greedy Optimization for Large Scale Semidefinite Programming

Zhifeng Hao[a,c], Ganzhao Yuan[a], Bernard Ghanem[b]

[a]*School of Computer Science & Engineering, South China University of Technology*
[b]*King Abdullah University of Science & Technology, KSA*
[c]*Faculty of Computer, Guangdong University of Technology*

Abstract

Many machine learning tasks (e.g. metric and manifold learning problems) can be formulated as convex semidefinite programs. To enable the application of these tasks on a large-scale, scalability and computational efficiency are considered desirable properties for a practical semidefinite programming algorithm. In this paper, we theoretically analyse a new *bilateral greedy optimization*(denoted BILGO) strategy in solving general semidefinite programs on large-scale datasets. As compared to existing methods, BILGO employs a bilateral search strategy during each optimization iteration. In such an iteration, the current semidefinite matrix solution is updated as a bilateral linear combination of the previous solution and a suitable rank-1 matrix, which can be efficiently computed from the leading eigenvector of the descent direction at this iteration. By optimizing for the coefficients of the bilateral combination, BILGO reduces the cost function in every iteration until the KKT conditions are fully satisfied, thus, it tends to converge to a global optimum. In fact, we prove that BILGO converges to the global optimal solution at a rate of $\mathcal{O}(1/k)$, where k is the iteration counter. The algorithm thus successfully combines the efficiency of conventional rank-1 update algorithms and the effectiveness of gradient descent. Moreover, BILGO can be easily extended to handle low rank constraints. To validate the effectiveness and efficiency of BILGO, we apply it to two important machine learning tasks, namely Mahalanobis metric learning and maximum variance unfolding. Extensive experimental results clearly demonstrate that BILGO

Email addresses: mazfhao@scut.edu.cn (Zhifeng Hao), y.ganzhao@scut.edu.cn (Ganzhao Yuan), b.ghanem@kaust.edu.sa (Bernard Ghanem)

can solve large-scale semidefinite programs efficiently.

Keywords: Semidefinite Programming, Low-Rank Optimization, Rank-1 Approximation, Frank-Wolfe algorithm, Leading Eigenvector, Metric Learning, Kernel Learning

1. Introduction

Semidefinite Programming (SDP) is commonly used in machine learning problems, such as metric learning [1, 2], manifold learning [3, 4, 5], linear regression [6], distance matrix completion [7], and solving Lyapunov equations [8]. Given a convex and continuously differentiable function F, where $F : \mathbb{R}^{d\times d} \to \mathbb{R}$, the goal of general semidefinite programming is to find a positive semidefinite (PSD) matrix $\mathbf{M}$ such that the objective $F(\mathbf{M})$ is minimized, as shown in Eq (1).

$$\min_{\mathbf{M}\in\mathbb{R}^{d\times d}} F(\mathbf{M}) \;\; s.t.\; \mathbf{M} \succcurlyeq 0 \tag{1}$$

A plethora of approaches have been proposed in the literature to handle the PSD constraints. **(i)** Classic interior point methods convert the original optimization problem to a series of subproblems with *LogDet* barrier functions. A similar strategy has also been adopted in studying the matrix nearness problem with linear constraints via *LogDet* divergences [9]. **(ii)** Eigenvalue decomposition methods, e.g. the projected gradient descent method [1, 10], iteratively trim the negative eigenvalues in order to maintain positive semidefiniteness. However, full eigenvalue decomposition is an expensive operator especially at large scales where d is large. **(iii)** Block coordinate descent methods, e.g. the method of [11], update one row or column of the solution matrix at a time by solving a subproblem, which enforces the PSD condition via the *Schur Complement Decomposition* theorem. Due to its memory complexity of $\mathcal{O}(d^2)$, this type of SDP solver is not feasible in large scale problems. **(iv)** Low rank methods [12] replace the PSD matrix with its low-rank decomposition and minimize an unconstrained, non-convex objective. Despite their significant computational gains, these methods sacrifice the convexity of the objective and *only* local minima are guaranteed.

From above, we observe that existing SDP methods are either time-consuming, memory-demanding, or suffer from local minima. In this paper, we theoretically analyse a new SDP method that applies a bilateral greedy optimization (BILGO) strategy to efficiently achieve the global minimum of

Eq (1). Inspired by recent work on leading eigenvector updates in SDP (e.g. [13, 14, 15, 16]) and the forward greedy selection algorithm [17], BILGO [18, 19] iteratively updates the current solution by searching for the optimal linear combination of the previous solution and the leading eigenvector of the gradient matrix. As compared to existing SDP methods that perform linear search, BILGO optimizes the linear combination weights *bilaterally*, i.e. it solves for both weights simultaneously. The contributions of this paper are three-fold. **(i)** We prove that each iteration of BILGO ensures a decrease in the objective untill the KKT conditions are fully satisfied, thus, it tends to converge to a global minimum. In fact, for any continuously differentiable convex function, BILGO converges to the global optimal solution at a rate of $\mathcal{O}(1/k)$, where k is the iteration counter. Therefore, the algorithm successfully combines the advantages of existing SDP techniques, achieving effectiveness and efficiency with a single shot. **(ii)** The performance of BILGO can be further enhanced by employing an efficient closed-form bilateral linear search, an efficient Lanczos-based method to compute the leading eigenvector, and a simple extension to handle low-rank matrix constraints, which is a common assumption adopted in machine learning theory and practise for large-scale problems. **(iii)** To validate the effectiveness and efficiency of BILGO, we apply it to two important machine learning tasks, namely Mahalanobis metric learning and maximum variance unfolding. Extensive experimental results clearly demonstrate that BILGO can solve large-scale semidefinite programs efficiently.

The rest of the paper is organized as follows. Section 2 provides a detailed description of the BILGO algorithm, as well as an analysis of its convergence properties. Further computational enhancements are discussed in Section 3. Section 4 illustrates connections between BILGO and existing work. Two applications of BILGO are discussed in section 5. Section 6 presents our extensive experimental results that clearly show BILGO's impressive performance on large scale machine learning benchmarks. Section 7 concludes this work.

2. BILGO algorithm

Mathematical Notation: The inner product between two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined as $\langle \mathbf{A}, \mathbf{B} \rangle = \sum_{ij} \mathbf{A}_{ij}\mathbf{B}_{ij}$, and Frobenius norm of a matrix $\mathbf{A}$ as $\|\mathbf{A}\|_{\text{fro}}$. We use $\mathbf{I}_d$ to denote the identity matrix of size $d \times d$ and Ω to denote the cone consisting of all PSD matrices.

In this section, we describe Bilateral Greedy Optimization (BILGO) method

[18, 19] and analyze its properties. BILGO is an iterative algorithm, which generates a sequence of feasible solutions $\{\mathbf{M}^0, \mathbf{M}^1, \ldots, \mathbf{M}^k, \ldots\} \subset \Omega$ until convergence to the global optimum of Eq (1). One key feature of BILGO lies in its update strategy, i.e. computing $\mathbf{M}^{k+1}$ from $\mathbf{M}^k$. The algorithm design is based on the following optimality analysis. Given a matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$ and by introducing the dual variables $\mathbf{S} \in \mathbb{R}^{d\times d}$ of Eq (1), we easily derive the following KKT optimality conditions.

$$\begin{aligned} \mathbf{M} &\succeq 0 \quad \text{(Feasibility)} \\ \mathbf{S} &\succeq 0 \quad \text{(Non-Negativity)} \\ \nabla F(\mathbf{M}) - \mathbf{S} &= \mathbf{0} \quad \text{(Optimality)} \\ \mathbf{SM} &= 0 \quad \text{(Complementary Slackness)} \end{aligned}$$

Note that the non-negativity constraints are enforced in eigenvalues of the dual variables $\mathbf{S}$ (refer to [15]). Since $F(\mathbf{M})$ is a convex function with respect to $\mathbf{M}$, these KKT conditions are both *necessary* and *sufficient* for global optimality.

Given a matrix $\mathbf{M}$ in the feasible set Ω, a descent direction at $\mathbf{M}$ is another matrix $\mathbf{D}$, such that $\mathbf{M} + \pi\mathbf{D}$ always remains in the feasible set Ω and improves the objective $F(\mathbf{M})$ when $\pi > 0$ is sufficiently small. Figure 1 shows the feasible direction $\mathbf{D}$ at point $\mathbf{M}$. $\mathbf{D}$ is a feasible direction if changing $\mathbf{M}$ by a small amount in the direction $\mathbf{D}$ maintains feasibility. We naturally obtain the following lemma.

Lemma 1. *Let $\mathbf{M} \in \Omega$ be any nonstationary point. If (1) there exists another matrix $\mathbf{N} \in \Omega$ that $\mathbf{D} = \mathbf{N} - \mathbf{M}$; and (2) $\langle \nabla F(\mathbf{M}), \mathbf{D}\rangle < 0$, then $\mathbf{D}$ is a descent direction.*

Proof: Because Ω is a convex set, when $0 < \pi < 1$, $\mathbf{M} + \pi(\mathbf{N} - \mathbf{M}) = (1-\pi)\mathbf{M} + \pi\mathbf{N} \in \Omega$ is also in Ω (1). Moreover, iff $\mathbf{M}$ is non-stationary, the gradient matrix $\nabla F(\mathbf{M}) \neq \mathbf{0}$. Since $F(.)$ is convex and differentiable, the first-order condition for convexity dictates that the following inequality holds: $F(\mathbf{M}+\pi\mathbf{D}) - F(\mathbf{M}) \geq \pi \langle \nabla F(\mathbf{M}), \mathbf{D}\rangle$. Obviously, if $\langle \nabla F(\mathbf{M}), \mathbf{D}\rangle \geq 0$, then $F(\mathbf{M} + \pi\mathbf{D}) \geq F(\mathbf{M})$ and $\mathbf{D}$ is not a descent direction. Therefore, it must be the case that $\langle \nabla F(\mathbf{M}), \mathbf{D}\rangle < 0$. □

Lemma 1 is crucial because it creates the opportunity for finding a descent direction $\mathbf{D}$ that *also* ensures the feasibility of each intermediate solution. As

such, BILGO does not involve any explicit projection onto Ω (e.g. trimming of negative eigenvalues). Such a projection-free method is also known as *conditional gradient descent* or the *Frank Wolfe algorithm* in some work [20]. In what follows, we use $\vec{\mathbf{v}}_{\max}$ to denote the eigenvector of matrix $-\nabla F(\mathbf{M})$ corresponding to its largest eigenvalue. Next, we discuss how to construct a feasible descent direction $\mathbf{D}$ using $\vec{\mathbf{v}}_{\max}$.

Lemma 2. *If* $\mathbf{M}$ *is not the global minimum of* $F(\mathbf{M})$*, there always exists* $\tau \geq 0$ *such that* $\mathbf{D} = \tau \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M}$ *is a descent direction.*

Proof: Let $\mathbf{N} = \tau \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T$. For any $\tau \geq 0$, $\mathbf{N} \in \Omega$, implying $\mathbf{D}$ satisfies the first condition of Lemma 1. Let $J(\tau) = \langle \mathbf{D}, \nabla F(\mathbf{M})\rangle = \langle \tau \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M}, \nabla F(\mathbf{M})\rangle = \langle \tau \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T, \nabla F(\mathbf{M})\rangle - \langle \mathbf{M}, \nabla F(\mathbf{M})\rangle$. Using the eigenvalue decomposition theorem, we have $-\nabla F(\mathbf{M}) = \sum_i \lambda_i \vec{\mathbf{v}}_i \vec{\mathbf{v}}_i^T$. Notice that $\vec{\mathbf{v}}_i^T \vec{\mathbf{v}}_j = 0$ for all $i \neq j$ and $\vec{\mathbf{v}}_i^T \vec{\mathbf{v}}_j = 1$ for all $i = j$. We thus obtain $J(\tau) = \langle \tau \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T, \sum_i \lambda_i \vec{\mathbf{v}}_i \vec{\mathbf{v}}_i^T\rangle - \langle \mathbf{M}, \sum_i \lambda_i \vec{\mathbf{v}}_i \vec{\mathbf{v}}_i^T\rangle = -\tau\lambda_{\max} - \sum_i \lambda_i \vec{\mathbf{v}}_i^T \mathbf{M} \vec{\mathbf{v}}_i$. Since the KKT optimal conditions imply that $\nabla F(\mathbf{M}) \succeq 0$ when $\mathbf{M}$ is a stationary point and since $\mathbf{M} \succeq 0$, we conclude that $\lambda_{\max} \geq 0$ for any non-stationary point $\mathbf{M}$. (i) When $\lambda_{\max} > 0$, we can always choose any sufficiently large τ so that $J(\tau) < 0$. (ii) When $\lambda_{\max} = 0$, $\nabla F(\mathbf{M}) \succeq 0$, $\langle \nabla F(\mathbf{M}), \mathbf{M}\rangle > 0$, since otherwise $\mathbf{M}$ satisfies the KKT optimal condition, implying that $\mathbf{M}$ is the global optimal solution. Therefore, we obtain

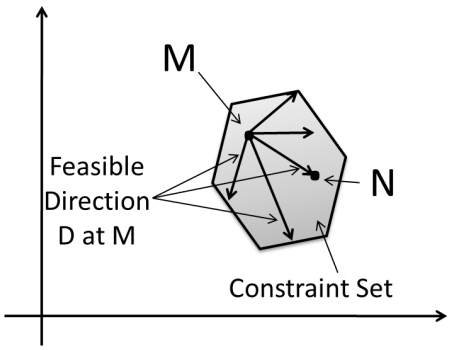


Figure 1: Feasible direction $\mathbf{D}$ at point $\mathbf{M}$

.

$J(\tau) < 0$ for any τ. Based on the second condition of Lemma 1, we conclude that there always exists $\tau \geq 0$ such that $\mathbf{D} = \tau \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T - \mathbf{M}$ is a descent direction. □

Combining Lemmas 1 and 2 above, we prove Theorem 1 that ensures the feasibility of the update rule in each iteration of BILGO $\mathbf{M} \leftarrow \alpha \mathbf{M} + \beta \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T$, when coefficients α and β are chosen appropriately.

Theorem 1. *If* $\mathbf{M}$ *is not the global minimum of* $F(\mathbf{M})$*, there always exist* $0 \leq \alpha \leq 1, \beta \geq 0$ *such that:*

$$F(\alpha \mathbf{M} + \beta \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T) < F(\mathbf{M})$$

Proof: We update the solution $\mathbf{M}$ by $\mathbf{M} \leftarrow \mathbf{M} + \pi \mathbf{D}$, where $0 \leq \pi \leq 1$ and $\mathbf{D}$ is a valid descent direction. Since $\mathbf{D} = \tau \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T - \mathbf{M}$ is a descent direction (refer to Lemma 2), there always exists $\tau \geq 0$ and $0 \leq \pi \leq 1$ such that $F(\mathbf{M}) > F(\mathbf{M} + \pi \mathbf{D}) = F(\mathbf{M} + \pi(\tau \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T - \mathbf{M})) = F((1 - \pi)\mathbf{M} + \pi\tau \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T)$. Since τ is dependent on π, we set α and β as in Eq (2) to obtain $0 \leq \alpha \leq 1$ and $\beta \geq 0$.

$$\beta = \pi\tau, \ \alpha = 1 - \pi \tag{2}$$

□

The overall BILGO method is summarized in Algorithm 1. In each iteration, BILGO utilizes Theorem 1 to update the result matrix. Using the largest vector of the gradient direction as the improvement direction, a bilateral optimization is conducted to construct the final update solution. Specifically, in the k^{th} iteration, BILGO computes the leading eigenvector $\vec{\mathbf{v}}_{\max}^k$ of the gradient descent direction $-\nabla F(\mathbf{M}^k)$, and then finds the bilateral combination of the current solution $\mathbf{M}^k$ and the rank-1 matrix computed using the leading eigenvector, i.e. $\vec{\mathbf{v}}_{\max}^k (\vec{\mathbf{v}}_{\max}^k)^T$. Theorem 1 guarantees that the objective is decreased in each iteration. To define stopping criteria, we analyze the relative change in $\lambda_{\max}^k$ and $\gamma_{slack}^k = \left\| \nabla F(\mathbf{M}^k) \mathbf{M}^k \right\|_{\text{fro}}$, which when sufficiently small indicate that the KKT conditions for global optimality have been satisfied.

Convergence of BILGO

In the rest of this section, we analyze the convergence properties of the BILGO algorithm, when the objective $F(\cdot)$ is continuously differentiable on the convex set Ω with some Lipschitz constant $L \geq 0$. Since Ω is a convex

Algorithm 1 BILGO Method

$k = 0, \mathbf{M}^k = \mathbf{0} \in \mathbb{R}^{d\times d}$, ϵ_1, ϵ_2
while not converge **do**
  Find the largest real eigenvalue $\lambda^k_{\max}$ and its corresponding eigenvector $\vec{\mathbf{v}}^k_{\max}$ of the matrix $-\nabla F(\mathbf{M}^k)$
  if $\frac{\lambda^k_{\max}-\lambda^{k-1}_{\max}}{\lambda^k_{\max}} < \epsilon_1$ and $\frac{\gamma^k_{slack}-\gamma^{k-1}_{slack}}{\gamma^k_{slack}} < \epsilon_1$ **then**
    terminate and output $\mathbf{M}^k$
  end if
  if $\frac{F(\mathbf{M}^k)-F(\mathbf{M}^{k-1})}{F(\mathbf{M}^{k-1})} < \epsilon_2$ **then**
    terminate and output $\mathbf{M}^k$
  end if
  Solve the bilateral search problem to find the optimal $0 \le \alpha \le 1$ and $\beta \ge 0$ to minimize $F\left(\alpha\mathbf{M}^k + \beta\vec{\mathbf{v}}^k_{\max}(\vec{\mathbf{v}}^k_{\max})^T\right)$.
  $\mathbf{M}^{k+1} = \alpha\mathbf{M}^k + \beta\vec{\mathbf{v}}^k_{\max}(\vec{\mathbf{v}}^k_{\max})^T$
  Increment k by 1
end while

set, $F(\cdot)$ is continuously differentiable (Chapter 2 in [21]) over Ω if for any $\mathbf{R}$ and $\mathbf{T}$ in Ω, the following inequality always holds for some $L \ge 0$.

$$0 \le F(\mathbf{R}) - F(\mathbf{T}) - \langle \nabla F(\mathbf{T}), \mathbf{R} - \mathbf{T}\rangle \le \frac{L}{2}\|\mathbf{T} - \mathbf{R}\|^2_{\text{fro}}$$

Lemma 3. *If F is continuously differentiable with some Lipschitz constant $L \ge 0$, for any $\mathbf{M}, \mathbf{N} \in \Omega$ and $\langle \mathbf{N} - \mathbf{M}, \nabla F(\mathbf{M})\rangle < 0$, there exists a positive constant C such that $F(\mathbf{M}+\pi(\mathbf{N}-\mathbf{M})) \le F(\mathbf{M})+\pi\langle \nabla F(\mathbf{M}), \mathbf{N} - \mathbf{M}\rangle+\pi^2 C$.*

Proof: By the definition of continuously differentiable function F, we obtain $F(\mathbf{M}+\pi(\mathbf{N}-\mathbf{M}))-F(\mathbf{M})-\pi\langle \nabla F(\mathbf{M}), \mathbf{N} - \mathbf{M}\rangle \le \frac{L}{2}\|\mathbf{M}-(\mathbf{M} + \pi(\mathbf{N} - \mathbf{M}))\|^2_{\text{fro}}$. Since $\langle \mathbf{N} - \mathbf{M}, \nabla F(\mathbf{M})\rangle < 0$, $\mathbf{M}$ and $\mathbf{M} + \pi(\mathbf{N} - \mathbf{M})$ belong to the descent sequence of the convex function $F(.)$, clearly the diameter of such sequence is bounded. In other words, there exists a constant C such that for $\mathbf{M}, \mathbf{N} \in \Omega$:

$$\max_{\langle \mathbf{N}-\mathbf{M}, \nabla F(\mathbf{M})\rangle<0} \frac{L}{2}\|\mathbf{M} - (\mathbf{M} + \pi(\mathbf{N} - \mathbf{M}))\|^2_{\text{fro}} = \pi^2 C$$

□

Similar to C, a constant C_f in [22] is used to denote the "Nonlinearity Measure" of a convex function over the *simplex set*. Inspired by this work,

we establish the convergence property of BILGO stated in Lemma 4.

Lemma 4. *For a continuously differentiable convex function $F(\cdot)$, one iteration of Algorithm 1 satisfies:*

$$h(\mathbf{M}^{k+1}) \leq h(\mathbf{M}^k) - g(\mathbf{M}^k)^2$$

where $h(\mathbf{M}^k) = \frac{F(\mathbf{M}^k)-F(\mathbf{M}^*)}{4C}$ *and* $g(\mathbf{M}^k) = \frac{\langle \tau \vec{\mathbf{v}}_{\max}\vec{v}_{\max}^T - \mathbf{M}^k, -\nabla F(\mathbf{M}^k)\rangle}{4C}$.

Proof: At the k^{th} iteration, the update rule is $\mathbf{M}^{k+1} = \alpha\mathbf{M}^k + \beta\vec{v}_{\max}\vec{v}_{\max}^T$. For simplicity, we use π and τ instead of α and β in this proof. They are transformable by Eq (2). By Lemma 3, we have: $F(\mathbf{M}^{k+1}) \leq F(\mathbf{M}^k) + \pi\langle \tau\vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M}^k, \nabla F(\mathbf{M}^k)\rangle + \pi^2 C$. This leads to the following inequalities.

$$\begin{aligned}
& h(\mathbf{M}^{k+1}) = \frac{F(\mathbf{M}^{k+1}) - F(\mathbf{M}^*)}{4C} \\
\leq\ & h(\mathbf{M}^k) + \frac{\pi\langle \tau\vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M}^k, \nabla F(\mathbf{M}^k)\rangle}{4C} + \frac{\pi^2}{4} \\
=\ & h(\mathbf{M}^k) - \pi g(\mathbf{M}^k) + \frac{\pi^2}{4} \\
=\ & h(\mathbf{M}^k) + \Big(g(\mathbf{M}^k) - \frac{\pi}{2}\Big)^2 - g(\mathbf{M}^k)^2 \\
\leq\ & h(\mathbf{M}^k) - g(\mathbf{M}^k)^2
\end{aligned}$$

In the last step, we assume $0 \leq g(\mathbf{M}^k) \leq \frac{1}{2}$. This assumption always holds using an appropriate initialization of $\mathbf{M}^0$ and setting τ and π to suitable values. Suppose we choose τ such that $g(\mathbf{M}^0) = \frac{1}{2}$ and $\pi = 1$ in the first iteration, the algorithm is guaranteed to decrease since $h(\mathbf{M}^1) \leq h(\mathbf{M}^0) - \frac{1}{4}$. Moreover, $h(\mathbf{M}^{k+1}) - h(\mathbf{M}^k)$ decreases as k increases, therefore we obtain $-\frac{1}{4} \leq h(\mathbf{M}^{k+1}) - h(\mathbf{M}^k) \leq h(\mathbf{M}^k) - \pi g(\mathbf{M}^k) + \frac{1}{4}\pi^2 - h(\mathbf{M}^k)$ for all $k \geq 1$. Therefore, we have $g(\mathbf{M}^k) \leq \frac{1}{4}\left(\frac{1}{\pi} + \pi\right)$ for all $k \geq 1$. Moreover, we set $\pi = 2g(\mathbf{M}^k)$ [1] to obtain $g(\mathbf{M}^k) \leq \frac{1}{4}\left(\frac{1}{2g(\mathbf{M}^k)} + 2g(\mathbf{M}^k)\right)$. By combining $g(\mathbf{M}^k) \geq 0$ and analysing this simple inequality, we observe $g(\mathbf{M}^k) \leq \frac{1}{2}$ for all $k \geq 0$. □

[1] Here the line search parameter $\pi = 2g(\mathbf{M}^k)$ is a conservative estimation. An exact line search program can be performed to ensure the greatest functional gains.

Using the previous lemmas, we prove Theorem 2, which guarantees that BILGO converges at a rate of $\mathcal{O}(1/k)$, where k is the iteration counter.

Theorem 2. *For a continuously differentiable convex objective $F(\cdot)$, for any $k \geq 1$, any iteration of Algorithm 1 guarantees that: $F(\mathbf{M}^k) - F(\mathbf{M}^*) \leq \frac{4C}{k+3}$*

Proof: Since $F(\cdot)$ is a convex function, $F(\mathbf{M}^k) - F(\mathbf{M}^*) \leq \langle \nabla F(\mathbf{M}^k), \mathbf{M}^k - \mathbf{M}^* \rangle = \langle \nabla F(\mathbf{M}^k), \mathbf{M}^k \rangle - \langle \nabla F(\mathbf{M}^k), \mathbf{M}^* \rangle \leq \langle \nabla F(\mathbf{M}^k), \mathbf{M}^k \rangle - \langle \nabla F(\mathbf{M}^k), \tau \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T \rangle$. The last inequality holds because $\vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T$ is the maximizer of the linear function $\langle -\nabla F(\mathbf{M}^k), \cdot \rangle$, thus $\langle \nabla F(\mathbf{M}^k), \vec{\mathbf{v}}_{\max} \vec{\mathbf{v}}_{\max}^T \rangle \leq \langle \nabla F(\mathbf{M}^k), \frac{\mathbf{M}^*}{\tau} \rangle$. Therefore, we conclude that $g(\mathbf{M}^k) \geq h(\mathbf{M}^k) \geq 0$.

Using Lemma 4, we obtain $h(\mathbf{M}^{k+1}) \leq h(\mathbf{M}^k) - h(\mathbf{M}^k)^2 = h(\mathbf{M}^k) \big(1 - h(\mathbf{M}^k)\big)$. Generally, noting that $1 - \nu \leq \frac{1}{1+\nu}$ for $\nu > -1$, we obtain $h(\mathbf{M}^{k+1}) \leq \frac{h(\mathbf{M}^k)}{1+h(\mathbf{M}^k)} = \frac{1}{1+\frac{1}{h(\mathbf{M}^k)}}$. Solving the recursion problem gives: $h(\mathbf{M}^{k+1}) \leq \frac{1}{k+3}$ for all $k \geq 1$. □

3. Computational Enhancements

In this section, we briefly describe some auxiliary enhancements to BILGO that can further improve its computational footprint.

3.1. Efficient Line Search

The BILGO line search step (Step 10 in Algorithm 1) can be solved by coordinate gradient decent or a bisection search. These traditional methods, however, are only sub-optimal from a convergence perspective. In order to accelerate the line search step, we apply the projective Newton method [23], which is guaranteed to achieve a quadratic convergence rate.

Projective Newton: With respect to α and β, the objective $f(\alpha, \beta) = F(\alpha \mathbf{M}^k + \beta \vec{\mathbf{v}}_{\max}^k (\vec{\mathbf{v}}_{\max}^k)^T)$ is a smooth convex function, and the variables are bounded, i.e. $0 \leq \alpha \leq 1$, $\beta > 0$. By defining the variable vector $\vec{\mathbf{s}} = [\alpha, \beta]^T$ and upper bound $\vec{\mathbf{u}} = [1, \infty]^T$, the line search problem becomes equivalent to finding the optimal $0 \leq \vec{\mathbf{s}} \leq \vec{\mathbf{u}}$ that minimizes the objective $f(\vec{\mathbf{s}})$. By employing the Projective Newton method, we update $\vec{\mathbf{s}}$ by applying: $\vec{\mathbf{s}}_{l+1} = \vec{\mathbf{s}}_l + \gamma(\vec{\mathbf{b}}_l - \vec{\mathbf{s}}_l)$, where $\vec{\mathbf{b}}_l$ is the solution to the box-constrained quadratic program in Eq (3).

$$\min_{0 \leq \vec{\mathbf{b}} \leq \vec{\mathbf{u}}} \frac{1}{2} (\vec{\mathbf{b}} - \vec{\mathbf{s}}_l)^T \nabla^2 f(\vec{\mathbf{s}}_l)(\vec{\mathbf{b}} - \vec{\mathbf{s}}_l) + \nabla f(\vec{\mathbf{s}}_l)^T (\vec{\mathbf{b}} - \vec{\mathbf{s}}_l) \tag{3}$$

Here, $\nabla f(\vec{\mathbf{s}}_l)$ and $\nabla^2 f(\vec{\mathbf{s}}_l)$ are the gradient and Hessian of $f(\cdot)$ evaluated at $\vec{\mathbf{s}}_l$ respectively. γ is a step-size parameter that is often set to 1 in practice. It can be shown that $\vec{\mathbf{b}}_l$ has a closed form solution, since Eq (3) only contains two variables. Additionally, we also observe that when the objective function $F(\mathbf{M})$ is strictly quadratic (e.g. the KTA Mahalanobis metric learning model in Eq. 5), the line search can be obtained exactly in one iteration.

Two Variables Box-Constrained Quadratic Programming: In what follows, we discuss how to solve Eq (3) analytically. We notice that Eq (3) can be formulated as the following two variables box-constrained quadratic problem, as shown in Eq (4):

$$\min_{\vec{\mathbf{b}} \in \mathbb{R}^{2\times 1}} \frac{1}{2}\vec{\mathbf{b}}^T \mathbf{Q} \vec{\mathbf{b}} + \vec{\mathbf{p}}^T \vec{\mathbf{b}}, \quad s.t. \;\; \vec{\mathbf{u}} \geq \vec{\mathbf{b}} \geq 0 \tag{4}$$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ and $\vec{\mathbf{p}} \in \mathbb{R}^{2\times 1}$. Introducing the dual variables $\vec{\mathbf{w}} \in \mathbb{R}^{2\times 1}$ and $\vec{\mathbf{z}} \in \mathbb{R}^{2\times 1}$ for the up bound and low bound constrains, we obtain the Karush-Kuhn-Tucker (KKT) condition for Eq (4) as follows.

$$\begin{aligned} \vec{\mathbf{z}} \geq 0, \;\; \vec{\mathbf{w}} \geq 0, \;\; \vec{\mathbf{b}} \geq 0 \\ \mathbf{Q}\vec{\mathbf{b}} + \vec{\mathbf{p}} - \vec{\mathbf{z}} + \vec{\mathbf{w}} = 0 \\ z_i b_i = 0, i = 1, 2 \\ w_i(u_i - b_i) = 0, i = 1, 2 \end{aligned}$$

When the matrix $\mathbf{Q}$ is rank deficient, Eq (4) reduces to a single variable non-negative quadratic problem which is trivial to compute. When the matrix $\mathbf{Q}$ is full rank, we observe that the optimal solution b^* must satisfy the following conditions for $i = 1, 2$.

$$\begin{aligned} z_i > 0, w_i = 0, b_i = 0 \\ z_i = 0, w_i > 0, b_i = u_i \\ z_i = 0, w_i = 0, 0 < b_i < u_i \end{aligned}$$

Therefore, the optimal solution b^* can be obtained by enumerating all the 9 candidate solutions and picking the one that leads to the smallest objective value in Eq (4).

3.2. Computing the Leading Eigenvector

BILGO needs to find the leading eigenvector of the negative gradient direction (Step 3 in Algorithm 1) in each iteration. Instead of using full-rank decomposition techniques, we develop a novel and efficient Lanczos-based Hessian-free Newton method to accelerate this procedure. Given a symmetric matrix $\mathbf{A} = -\nabla F(\mathbf{M}^k)$, we let $f(\vec{\mathbf{v}}) = \frac{1}{4}\|\vec{\mathbf{v}}\vec{\mathbf{v}}^T - \mathbf{A}\|_{\text{fro}}^2$. $\vec{\mathbf{v}}_{\max}^k$ is the leading eigenvector, if and only if $\vec{\mathbf{v}}_{\max}^k = \frac{\vec{\mathbf{v}^*}}{\|\vec{\mathbf{v}^*}\|}$ and $\vec{\mathbf{v}^*} = \arg\min_{\vec{\mathbf{v}}} f(\vec{\mathbf{v}})$. Although the objective $f(\cdot)$ is non-convex, we use the following proposition to find a local minimum.

Proposition 1. *For any $\vec{\mathbf{v}} \neq 0$, any local minimum of $f(\vec{\mathbf{v}})$ is a global minimum*[2]*.*

To find a non-zero local minimum, the gradient and Hessian matrix with respect to $\vec{\mathbf{v}}$ can be computed as:

$$\begin{aligned} \nabla f(\vec{\mathbf{v}}) = &\quad \vec{\mathbf{v}}^T\vec{\mathbf{v}}\vec{\mathbf{v}} - \mathbf{A}\vec{\mathbf{v}} \\ \nabla^2 f(\vec{\mathbf{v}}) = &\quad 2\vec{\mathbf{v}}\vec{\mathbf{v}}^T + \vec{\mathbf{v}}^T\vec{\mathbf{v}}\mathbf{I}_d - \mathbf{A} \end{aligned}$$

In order to minimize $f(\vec{\mathbf{v}})$ efficiently, we use the Hessian-free Newton method, which builds a quadratic model to find a search direction that provides a reasonable trade-off between accuracy and computational cost. The proposed method is summarized in Algorithm 2. The fundamental idea here is to compute the approximate Newton direction using the traditional Lanczos [24] or Conjugate Gradient (CG) method. However, since the Hessian $\nabla^2 f(\vec{\mathbf{v}})$ may not be PSD, the algorithm terminates as soon as negative eigenvalues are detected in the Hessian, and the most recently available descent direction is returned. After the Newton direction has been approximated, the algorithm performs an exact line search (with step size μ) using the first order optimiality condition (Step 4 of Algorithm 2). It can be shown that $f(\vec{\mathbf{v}}^t + \mu\vec{\mathbf{d}}^t)$ is a 4^{th} order polynomial in μ. Therefore, the line search step can be obtained by solving a cubic equation and selecting the solution that leads to the greatest descent. Although our proposed method for computing the leading eigenvector is simple greedy gradient descent, it exhibits excellent convergence

[2] This is because the original optimization problem $\frac{1}{4}\|\mathbf{V} - \mathbf{A}\|_{\text{fro}}^2$ is convex and has a unique minimum. When the matrix rank-1 factorization $\mathbf{V} = \vec{\mathbf{v}}\vec{\mathbf{v}}^T$ is used, there is a surjection between solutions using $\vec{\mathbf{v}}$ and those using $\mathbf{V}$.

Algorithm 2 A Lanczos-based Hessian-free Newton method to find leading eigenvector

1: Initialize $\vec{\mathbf{v}}^0$ to a random vector and set $t = 0$
2: **while** not converge **do**
3: Use the iterative Lanczos (or CG) method to approximately solve $\nabla^2 f(\vec{\mathbf{v}}^t)\vec{\mathbf{d}}^t = -\nabla f(\vec{\mathbf{v}}^t)$
4: Solve the cubic equation $\frac{\mathrm{d}f(\vec{\mathbf{v}}^t+\mu\vec{\mathbf{d}}^t)}{\mathrm{d}\mu} = 0$ $(0 < \mu < 1)$ to get a set of candidate steps: *cstep*
5: set μ to the element in *cstep* that leads to the greatest descent
6: Update $\vec{\mathbf{v}}^{t+1} = \vec{\mathbf{v}}^t + \mu\vec{\mathbf{d}}^t$
7: Increment t by 1
8: **end while**

empirically.

3.3. Low-Rank Optimization

As mentioned before, one advantage of BILGO is that it can be easily extended to handle low-rank matrix constraints, as in [17]. At each BILGO iteration, the solution is updated by a rank-1 matrix: $\mathbf{M} \leftarrow \alpha\mathbf{M} + \beta\vec{v}_{\max}\vec{v}_{\max}^T$. By using the low rank representation $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, this update rule becomes equivalent to adding a new column to $\mathbf{L}$, i.e. $\mathbf{L} \leftarrow [\sqrt{\alpha}\mathbf{L}|\sqrt{\beta}\vec{\mathbf{v}}_{\max}]$. Since $\mathbf{L}$ is generated by a greedy algorithm, it is not necessarily the solution which leads to the greatest descent on the objective. Therefore, we need to refine $\mathbf{L}$ to improve the baseline BILGO implementation. However, since the low rank representation $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ is used, the objective $F(\mathbf{L}\mathbf{L}^T)$ is non-convex in general. This may pose some difficulty in optimization. Fortunately, there exist efficient methods to find a local minimum of this non-convex problem. For example, [12] uses a first-order L-bfgs approach that employs a strong Wolfe-Powell line search. Also, the methods in [8, 15] use a Riemannian trust region approach that is based on a second-order model of the objective defined on the manifold. Clearly, these methods can be incorporated into the BILGO algorithm to improve its baseline implementation. The result is an algorithm that is numerically robust and fast.

4. Connections to Existing Work

In this section, we illustrate connections between the BILGO algorithm and prior work.

1. *Connection with Hazan's update rule*: Sparse SDP approximation was initially proposed in [13], applied to nuclear norm regularized problems [14], and then applied to boosting metric learning [16]. Methods of this type mainly use Hazan's update rule to solve an SDP with a trace-one constraint: $\mathbf{M} \leftarrow \mathbf{M} + \pi\mathbf{D} = \mathbf{M} + \pi(\vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M})$. These methods enforce the constraint implicitly. After initializing $\mathbf{M}_0$ with an arbitrary trace-one matrix, these methods converge to the global minimum. In fact, when $\tau = 1$, the BILGO update rule reduces to the aforementioned one. However, Hazan's rule cannot be directly used to solve the unconstrained SDP, since it does not guarantee that $\mathbf{D} = \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T - \mathbf{M}$ is a descent direction for Eq (1) during every iteration. This will occur at a nonstationary point $\mathbf{M}$ where $\langle \mathbf{D}, \nabla F(\mathbf{M})\rangle = -\lambda_{\max} - \sum_i \lambda_i \vec{\mathbf{v}}_i^T \mathbf{M} \vec{\mathbf{v}}_i \geq 0$. In comparison, Theorem 1 guarantees that BILGO *always* generates a descent direction for Eq(1) at *every* iteration.
2. *Connection with Journée et al.'s update rule*: The update rule below is used in [15] to solve SDP problems, i.e. $\mathbf{M} \leftarrow \mathbf{M} + \pi\mathbf{D} = \mathbf{M} + \pi(\vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T)$. This optimization method uses the *smallest* algebraic eigenvalue of the *gradient direction* to monitor the convergence of the algorithm. Here, we point out that **(i)** this update rule does not follow the second condition of Lemma 1, i.e. it may not be able to decrease the objective when $\langle \nabla F(\mathbf{M}), \vec{\mathbf{v}}_{\max}\vec{\mathbf{v}}_{\max}^T\rangle = \lambda_{\max} = 0$. **(ii)** Moreover, such a way of monitoring the convergence of the algorithm may be problematic, because $\lambda_{\max} = 0$ does not necessarily indicate that the algorithm has converged (refer to the experiment in Section 6.1).
3. *Connection with Random Conic Pursuit*: Random Conic Pursuit was proposed in [18] for SDP problems with linear constraints. Their bilateral update rule is similar to that of BILGO: $\mathbf{M} \leftarrow \alpha\mathbf{M} + \beta\mathbf{D} = \alpha\mathbf{M} + \beta\vec{x}\vec{x}^T$, but where $\vec{x}$ is a random vector sampled from a multivariate normal distribution and $\alpha, \beta \geq 0$. Moreover, their proof of convergence rate is based on the assumption that $\beta = \frac{1}{k}$ and $\alpha = 1 - \frac{1}{k}$. It is not necessarily the optimal choice for α, β and $\vec{x}$, but it turns out that their stochastic sampling algorithm can also converges with convergence rate $\mathcal{O}(1/k)$. Actually, the main problem with Random Conic Pursuit is the large hidden constants in the $\mathcal{O}$-notation which makes it slow in practice (refer to the experiment in Section 6.1).
4. *Connection with classical Frank-Wolfe algorithm*: The BILGO optimization is built on the classical Frank-Wolfe algorithm [20] (also known

as conditional gradient descent) framework. Given a non-stationary solution $\mathbf{M}$, classical Frank-Wolfe algorithm finds a descent direction $\mathbf{D} = \mathbf{N} - \mathbf{M}$ by solving $\mathbf{N} = \arg\min_{\mathbf{N}' \in \Omega} \langle \nabla F(\mathbf{N}'), \mathbf{N}' - \mathbf{M} \rangle$. Unlike the constraint set is closed in [20, 22], the constraint set Ω we consider is open. So the optimization problem above is unbounded if we simply consider a full rank matrix $\mathbf{N}$. However, one does not need to use a full-rank matrix to minimize a linear function but a rank-1 matrix suffices. To allow a bounded solution, we restrict $\mathbf{N}$ within a rank-1 unit conic space and show that it can always decrease the objective function. In fact, this is the same property that we use in computing the rank-1 factorization (refer to Footnote 2 in Page 11).

5. *Connection with Shalev-Shwartz et al.'s algorithm*: Shalev-Shwartz et al. proposed an efficient forward greedy selection algorithm [17] for solving large-scale matrix minimization problems with a low-rank constraint. Based on the smoothness and strong convexity of the objective function, they also derived its formal approximation guarantees for the greedy algorithm. Their algorithm mainly focused on a low rank matrix completion problem, i.e. $\min_{\mathbf{X}} \sum_{(i,j)\in\mathbf{\Psi}} (\mathbf{X}_{ij} - \mathbf{Y}_{ij})^2$, s.t. $\text{rank}(\mathbf{X}) \leq k$, where $\mathbf{\Psi}$ is the set of all given entries of $\mathbf{Y}$. It can be shown that this low rank matrix completion can be transformed into the equivalent semidefinite optimization problem below[25]: $\min_{\hat{\mathbf{X}}} \sum_{(i,j)\in\mathbf{\Psi}} (\hat{\mathbf{X}}_{ij} - \hat{\mathbf{Y}}_{ij})^2$, s.t. $\text{rank}(\hat{\mathbf{X}}) \leq k$, where $\hat{\mathbf{X}} = [\mathbf{A}|\mathbf{X}; \mathbf{X}^T|\mathbf{B}]$ and $\hat{\mathbf{Y}} = [\mathbf{0}|\mathbf{Y}; \mathbf{Y}^T|\mathbf{0}]$. Furthermore, '|' and ';' in $[\cdot]$ denote respectively the column-wise and row-wise partitioning indicator, $\mathbf{A}$ and $\mathbf{B}$ are the outputs of the optimization program. What we need above is to make sure the matrix solution is symmetric and hence, right and left eigenvectors are the same. To sum up, the semidefinite optimization problem is more general.
6. *Connection with non-negatively constrained convex programming*: While the semidefinite program in Eq (1) forces non-negativity constraints on the eigenvalues of a matrix, non-negatively constrained convex programming forces non-negativity constraints on the elements of a vector, i.e. $\min f(\vec{w})$, s.t. $\vec{w} \geq 0$, where $f(\cdot)$ is a convex smooth objective function. This optimization problem has been extensively studied with the best-known example of non-negative matrix factorization [26] and quadratic hinge loss support vector machines[27]. Clearly, the bilateral line search, local minimization and convergence analysis can be natu-

rally also extended to this problem of vector space. In every iteration, one can pick the coordinate of the largest real value in the descent direction $-\nabla f(\vec{w})$ and greedily decrease the objective function. Similar to the low-rank factorization $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, one can employ the non-negative representation $\vec{w} = \vec{v} \odot \vec{v}$ and refine $\vec{v}$ using some efficient non-convex optimization approaches [2].

7. *Connection with Sören Laue's algorithm*: We are also aware of the recent proposed hybrid algorithm for convex semidefinite optimization [19]. Both approaches suggest using bilateral line search and local minimization. Both proofs look different, but are actually equivalent. Firstly, one of the linear search variables α in BILGO is further constrained to be $\alpha \leq 1$. Although we can still further constraint it, such constraint may not be needed. This is because with a descent direction $\mathbf{D}$ in Theorem 1, the convex optimization will find a non-negative step length $\pi(\pi \geq 0)$ which will naturally leads to $\alpha = 1 - \pi \leq 1$ (refer to the experiment in Section 6.1). Secondly, it appears that their convergence analysis works by constructing a duality gap that is an upper bound on the primal error, and our analysis is based on the KKT optimal condition theory of convex optimization. However, it is known that the relaxed complementary slackness condition as part of KKT can be shown to be equivalent to a duality gap. Generally speaking, both proofs are in fact equivalent to each other. However, our research on the similar problem is independent with theirs. We think our theoretical analysis here is of independent interest.

5. Applications

In this section, we discuss two important applications of the bilateral greedy optimization: Mahalanobis metric learning and maximum variance unfolding for manifold learning.

5.1. Mahalanobis Metric Learning

For Mahalanobis metric learning, we mainly focus on two learning models: the Kernel Target Alignment (KTA) model [28] and the Large Margin Nearest Neighbor (LMNN) model [29].

KTA model: Let c be the number of classes and let $\mathbf{Y} = [\vec{\mathbf{y}}_i | \dots | \vec{\mathbf{y}}_m]$ denote the class labels assigned to all training data $\mathbf{X} = \{\vec{\mathbf{x}}_1^T, \dots, \vec{\mathbf{x}}_m^T\}$. Each $\mathbf{y}_i = (y_i^1 \dots y_i^c)^T \in \{0,1\}^c$ is a binary vector of c elements. The model compares two kernel matrices, one is based on class labels: $\mathbf{K}_D = \mathbf{Y}\mathbf{Y}^T$,

the other on the distance metric $\mathbf{K}_X = \mathbf{X}\mathbf{M}\mathbf{X}^T$. The loss model computes the dissimilarity between two zero-mean Gaussian distributions with covariance matrices $\mathbf{K}_D$ and $\mathbf{K}_X$ respectively. Here we use the Frobenius norm to measure the distance between $\mathbf{K}_D$ and $\mathbf{K}_X$. By employing the information-theoretic regularizer [9] $\|\mathbf{M} - \mathbf{I_d}\|^2_{\text{fro}}$, the objective function of the SDP can be written as in Eq (5):

$$F_{KTA}(\mathbf{M}) = \min_{\mathbf{M} \succeq 0} \|\mathbf{M} - \mathbf{I_d}\|^2_{\text{fro}} + \lambda_{reg} \|\mathbf{X}^T\mathbf{M}\mathbf{X} - \mathbf{Y}^T\mathbf{Y}\|^2_{\text{fro}} \tag{5}$$

The gradient of F_{KTA} with respect to $\mathbf{M}$ can be computed respectively as in Eq (6):

$$\frac{\partial F_{KTA}}{\partial \mathbf{M}} = 2\lambda_{reg}\mathbf{X^T}\mathbf{X}\mathbf{M}\mathbf{X^T}\mathbf{X} + 2\mathbf{M} - 2\mathbf{I_d} - 2\lambda_{reg}\mathbf{X^T}\mathbf{Y}\mathbf{Y^T}\mathbf{X} \tag{6}$$

To allow efficient low-rank optimization, we use the change of variable $\mathbf{M} = \mathbf{L}\mathbf{L^T}$ and reformulate the SDP problem $F_{KTA}(\mathbf{M})$ as a non-convex optimization problem $F_{KTA}(\mathbf{L}\mathbf{L^T})$. Once the gradient of F_{KTA} with respect to $\mathbf{L}$ has been computed, as in Eq (7), we can utilize a first-order local search strategy (see section 3.3) to refine the solution $\mathbf{L}$.

$$\frac{\partial F_{KTA}}{\partial \mathbf{L}} = 4\lambda_{reg}\mathbf{X^T}\mathbf{X}\mathbf{L}\mathbf{L^T}\mathbf{X^T}\mathbf{X}\mathbf{L} + 4\mathbf{L}\mathbf{L^T}\mathbf{L} - 4\mathbf{L} - 4\lambda_{reg}\mathbf{X^T}\mathbf{Y}\mathbf{Y^T}\mathbf{X}\mathbf{L} \tag{7}$$

LMNN model: LMNN metric learning has two objectives. The first is to minimize the average distance between instances and their target neighbors. The second goal is to constrain impostors to be further away from target neighbors, thus, pushing them out of the local neighborhood. There exist a distant constraint set $\mathcal{R}$ and a neighbor constraint set $\mathcal{S}$, where $\forall(i,j,k)(\vec{x}_i, \vec{x}_j, \vec{x}_k) \in \mathcal{R}$ and $\forall(i,j)(\vec{x}_i, \vec{x}_j) \in \mathcal{S}$, $y_i = y_j$ and $y_i \neq y_k$. After defining $\mathbf{X}_{ij} = (\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^T$, we write the distance between $\vec{x}_i$ and $\vec{x}_j$ as $d_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^T\mathbf{M}(\vec{x}_i - \vec{x}_j) = tr(\mathbf{M}\mathbf{X}_{ij})$. Using a quadratic hinge loss function, the optimization problem and the gradient with respect to $\mathbf{M}$ can be written as shown in Eq (8) and Eq(9), where $\mathcal{ST}$ denotes the support triple set which generates an l_2 loss value greater than zero, i.e. $(\vec{x}_i, \vec{x}_j, \vec{x}_k)$ belongs to $\mathcal{ST}$ if $1 - tr(\mathbf{M}\mathbf{X}_{ik}) + tr(\mathbf{M}\mathbf{X}_{jk}) > 0$.

$$F_{LMNN}(\mathbf{M}) = \min_{\mathbf{M}\succeq 0} \sum_{(\vec{x}_i,\vec{x}_j)\in\mathcal{S}} tr(\mathbf{M}X_{ij}) + \lambda_{reg} \sum_{(\vec{x}_i,\vec{x}_j,\vec{x}_k)\in\mathcal{R}} \max(0, 1 - tr(\mathbf{M}X_{ik}) + tr(\mathbf{M}X_{jk}))^2 \quad (8)$$

$$\frac{\partial F_{LMNN}}{\partial \mathbf{M}} = \sum_{(\vec{x}_i,\vec{x}_j)\in\mathcal{S}} \mathbf{X}_{ij} + 2\lambda_{reg} \sum_{(\vec{x}_i,\vec{x}_j,\vec{x}_k)\in\mathcal{ST}} (1 - tr(\mathbf{M}\mathbf{X}_{ik}) + tr(\mathbf{M}\mathbf{X}_{jk}))(\mathbf{X}_{ik} - \mathbf{X}_{ij}) \quad (9)$$

Again, when the change of variable $\mathbf{M} = \mathbf{L}\mathbf{L}^{\mathbf{T}}$ is used, the gradient of F_{LMNN} with respect to $\mathbf{L}$ can be computed as shown in Eq (10).

$$\frac{\partial F_{LMNN}}{\partial \mathbf{L}} = \sum_{(\vec{x}_i,\vec{x}_j)\in\mathcal{S}} 2\mathbf{X}_{ij}\mathbf{L} + 4\lambda_{reg} \sum_{(\vec{x}_i,\vec{x}_j,\vec{x}_k)\in\mathcal{ST}} (1 - tr(\mathbf{L}^{\mathbf{T}}\mathbf{X}_{ik}\mathbf{L}) + tr(\mathbf{L}^{\mathbf{T}}\mathbf{X}_{jk}\mathbf{L}))(\mathbf{X}_{ik}\mathbf{L} - \mathbf{X}_{ij}\mathbf{L}) \quad (10)$$

5.2. Maximum Variance Unfolding

Maximum Variance Unfolding (MVU) or Semidefinite Embedding (SDE) [3, 4, 5] is among the state of the art manifold learning algorithms and experimentally proven to be the best method to unfold a manifold to its intrinsic dimension. The main intuition behind MVU is to exploit the local linearity of manifolds and create a mapping that preserves local neighborhoods at every point of the underlying manifold. It creates a mapping from the high dimensional input vectors to some low dimensional Euclidean vector space in the following steps. Firstly, a neighborhood graph is created. Each input is connected with its k-nearest input vectors (according to Euclidean distance metric) and all k-nearest neighbors are connected with each other. The neighborhood graph is 'unfolded' with the help of semidefinite programming. The low-dimensional embedding is finally obtained by application of multidimensional scaling on the learned inner product matrix. Specifically, given a 0/1 binary indicator matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$, a Euclidean distance matrix

$\mathbf{D} \in \mathbb{R}^{n\times n}$ and $\mathbf{W} \in \mathbb{R}^{n\times n}$, MUV can be formulated as the optimization problem[3] shown in Eq (11).

$$F_{MVU}(\mathbf{M}) = \min_{\mathbf{M}\succeq 0} \|\mathbf{U} \odot (\widehat{\mathbf{M}}\vec{e}^T + \vec{e}\widehat{\mathbf{M}}^T - 2\mathbf{M} - \mathbf{D})\|_{\text{fro}}^2 - \nu tr(\mathbf{WM}) \quad (11)$$

When $\mathbf{W} = \mathbf{I_n}$, it is plain MVU. When $\mathbf{W} = \mathbf{Y}$ [4], it is the 'colored' variant of MVU, which produces low-dimensional representations subject to class labels or side information [4]. We let $\mathbf{M} = \mathbf{LL}^T$, $\vec{o} = diag(\mathbf{M})$ and write the gradient of $F_{MVU}(\mathbf{M})$ with respect to $\mathbf{M}$ and $\mathbf{L}$ as shown in Eq (12) and Eq (13). [5]

$$\begin{aligned}\frac{\partial F_{MVU}}{\partial \mathbf{M}} =& (4\widehat{\vec{e}\mathbf{U}o} + 4\widehat{\vec{o}\mathbf{U}e}) - (4\mathbf{U}\widehat{o} + 4\widehat{o}\mathbf{U}) - (4\widehat{\widehat{\mathbf{UD}}} - 4\mathbf{U}\odot\mathbf{D}) - \\ & (8\widehat{\widehat{\mathbf{UM}}} - 8\mathbf{U}\odot\mathbf{M}) - \nu\mathbf{W} \quad (12)\\ \frac{\partial F_{MVU}}{\partial \mathbf{L}} =& (8\widehat{\vec{e}\mathbf{U}o} + 8\widehat{\vec{o}\mathbf{U}e} - 8\mathbf{U}\widehat{o} - 8\widehat{o}\mathbf{U} - 8\widehat{\widehat{\mathbf{UD}}} + 8\mathbf{U}\odot\mathbf{D} - 16\widehat{\widehat{\mathbf{ULL}^T}} + \\ & 16\mathbf{U}\odot\mathbf{LL}^T)\mathbf{L} - 2\nu\mathbf{WL} \quad (13)\end{aligned}$$

This formulation allows us to run the L-bfgs algorithm and iteratively improve the result. After the algorithm terminates, the final configuration $\mathbf{L}$ is used as the optimal solution to the program.

6. Experimental results

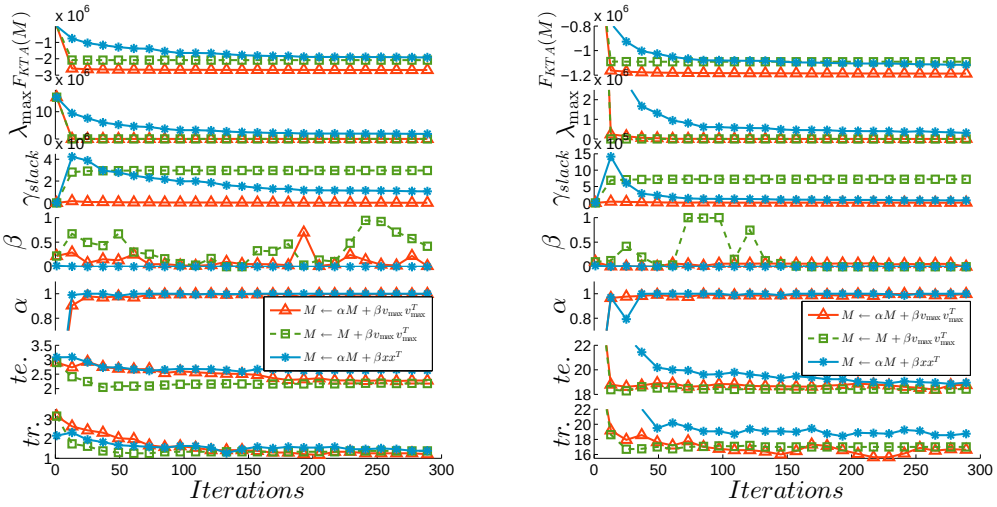
In this section, we demonstrate the effectiveness and efficiency of BILGO algorithm on the two learning tasks: Mahalanobis metric learning and maximum variance unfolding. All algorithms are implemented in Matlab on an Intel 2.50 GHz CPU with 4GB RAM. We test on 12 well-known real-world benchmark datasets[6], which contain high dimensional ($d \geq 10^3$)

[3] Here $\odot$ denotes the Hadamard product. $\vec{e}$ denotes a column vector having all elements equal to one. $\widehat{\ }$ is the diagonal operator, it is equlevent to the 'diag' Matlab function: when Δ is a matrix, $\widehat{\Delta}$ denotes a column vector formed from the main diagonal of Δ, when Δ is a vector, $\widehat{\Delta}$ denotes a diagonal matrix with Δ in the main diagonal entries.

[4] $\mathbf{Y}$ is the label matrix defined earlier in Section 5.1

[5] Here we use the property of Hadamard product: $\vec{x}^T(\mathbf{A}\odot\mathbf{B})\vec{y} = tr(\widehat{\vec{x}}\mathbf{A}\widehat{\vec{y}}\mathbf{B}^T)$.

[6] `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`



(a) on 'w1a' dataset

(b) on 'a1a' dataset

Figure 2: Convergence behavior of BILGO, the Journée et al.'s method and Random Conic Pursuit method on the 'w1a' and 'a1a' dataset.

data vectors and are usually large scale (more than 10^4 elements each). Some Matlab codes and sample datasets for tests are available at: `http://yuanganzhao.weebly.com/`.

6.1. Global Convergence

In this section, we verify the global convergence property of BILGO. We demonstrate the asymptotic behavior of BILGO using the Kernel Target Alignment metric learning model (BILGO-KTA) on 'w1a' and 'a1a' dataset. In Figure 2, we compare the Journée et al.'s update rule [15] and Random Conic Pursuit [18] against that of BILGO-KTA. We plot the values $\{F_{KTA}(\mathbf{M}^k)$, γ_{slack}^k, $\lambda_{\max}^k$, α, β, testing accuracy, training accuracy$\}$ at every iteration k (k = 1,...,300). For Random Conic Pursuit, we use Matlab function 'mvnrnd' to sample a random vector $\vec{x} \in \mathbb{R}^{d\times 1}$ from the multivariate normal distribution with mean zero and covariance $\mathbf{\Sigma}$. Here $\mathbf{\Sigma} = (1-\chi)\mathbf{M}^k + \chi\mathbf{I}_d$, where χ is set to 0.5 in our experiments. For BILGO, we only constrain α to be $\alpha \geq 0$ instead of $1 \geq \alpha \geq 0$. The results in Figure 2 lead to several interesting observations. Firstly, we observe that Journée et al.'s method gets stuck at a critical point where $\lambda_{\max} \approx 0$ and $\gamma_{slack} \gg 0$, while BILGO-KTA converges to the global minimum where the KKT conditions are fully sat-

isfied, i.e. $\lambda_{\max} \approx 0, \gamma_{slack} \approx 0$. Secondly, although Random Conic Pursuit selects the rank-1 conic randomly, it can still decrease the objective function iteratively and tends to converge the global minimum with both $\lambda_{\max}$ and γ_{slack} decreasing to 0. This phenomenon is more pronounced on 'a1a' dataset. We attribute this phenomenon to the bilateral strategy used in Random Conic Pursuit. However, Random Conic Pursuit converges much slower than BILGO-KTA. Thirdly, we observe that as BILGO iterates, α is approaching to 1 and β is approaching to 0. Moreover, we find that $\alpha \leq 1$ always holds. This observation indicates that further constraining $\alpha \leq 1$ is not needed in practice. Finally, as for efficacy of the learning tasks, we find that BILGO achieves the lowest objective, thus it results in the best training accuracy. However, BILGO does not necessarily obtain the best testing accuracy while Journée et al.'s approach does. The regularization function of the learning task is responsible for these results.

6.2. Accuracy and Efficiency on Metric Learning

In this section, we demonstrate the accuracy and efficiency of BILGO by applying it to metric learning tasks. We use BILGO-KTA and BILGO-LMNN to denote the BILGO solver for the two metric learning optimization problems respectively. Moreover, we use "L-bfgs + exact line search" as it is suggested in [2] to improve the intermediate result in every *5* iterations of BILGO-KTA, giving rise to its local search version BILGO-KTA-LS.

After learning the optimal Mahalanobis distance function from the training set, we use a kNN classifier (k=3) to classify each test sample. The hyper-parameter λ_{reg} is set by a typical two-fold cross-validation procedure searching over the values $\lambda_{reg} = \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$.

We compare BILGO-KTA metric learning method against three full rank methods: ITML[7] [9], Boost-Metric[8] [16], LMNN[9] [29], and two low rank methods: SURF [10] [6] and KTA-Lbfgs[11]. We use the default stopping criterion for each of these methods. BILGO is terminated when the relative change in $\{\lambda_{\max}, \gamma_{slack}\}$ or $F(\cdot)$ is small enough. We find $\epsilon_1 = 10^{-3}$ and

[7] `http://www.cs.utexas.edu/~pjain/itml/`

[8] `http://code.google.com/p/boosting/`

[9] `http://www.cse.wustl.edu/~kilian/`

[10] `http://www.montefiore.ulg.ac.be/~meyer/`

[11] This is a metric learning algorithm that minimizes the objective in Eq (5) using the local-search L-bfgs method described in [12] with a random initial solution.

Table 1: Comparison of running times (in seconds) with state-of-the-art metric learning solvers. OOM indicates "out of memory", and OOT indicates "out of time".

Data Set	SURF	ITML	Boost-Metric	LMNN	BILGO-LMNN	KTA-Lbfgs	BILGO-KTA	BILGO-KTA-LS
splice	8±3	2.8±1	8±4	18±10	15±6	0.8±0.5	**0.2**±0.1	**0.2**±0.1
isolet	**10**±2	230±112	302±140	510±121	107±16	157±20	56±10	107±12
optdigits	1±0.5	0.5±0.1	17±3	17±5	33±9	1±0.3	0.8±0.2	**0.3**±0.1
dna	23±4	10±4	83±30	232±50	202±32	3±1	**2±2**	**2**±1
a1a	15±3	15±6	450±213	138±36	310±23	10±4	8±2	**6±2**
protein	35±11	100±11	1749±351	2310±412	76±23	51±16	50±14	**31±4**
mushrooms	7±3	6±3	79±22	14±4	50±14	5±2	3±1	**2±1**
w1a	26±6	20±6	2588±153	830±112	213±35	18±1	15±3	**8±2**
usps	35±5	450±87	578±121	120±21	53±11	39±9	**25±4**	28±5
mnist	612±87	740±81	OOT	1286±174	612±77	601±121	361±31	**301±31**
gisette	411±58	OOT	OOT	OOT	489±64	138±35	89±13	**58±9**
realsim	992±243	OOM	OOM	OOM	2545±312	108±31	220±56	**68±14**

$\epsilon_2 = 10^{-5}$ to be a good trade-off between accuracy and runtime. Since SURF uses a low rank representation, we have $\mathbf{M} = \mathbf{L}\mathbf{L}^T, \mathbf{L} \in \mathbb{R}^{d\times r}$. To make a fair comparison, we set $r = \max(15, k)$, where k is the number of iterations required by BILGO-KTA-LS to converge. k takes on values between 8 and 30 in our experiments [12]. We select 2000 pairs/triplets of constraints for training for all algorithms except the KTA-based algorithms. The experimental results are reported in Table 1 and Table 2. Several conclusions can be drawn here.

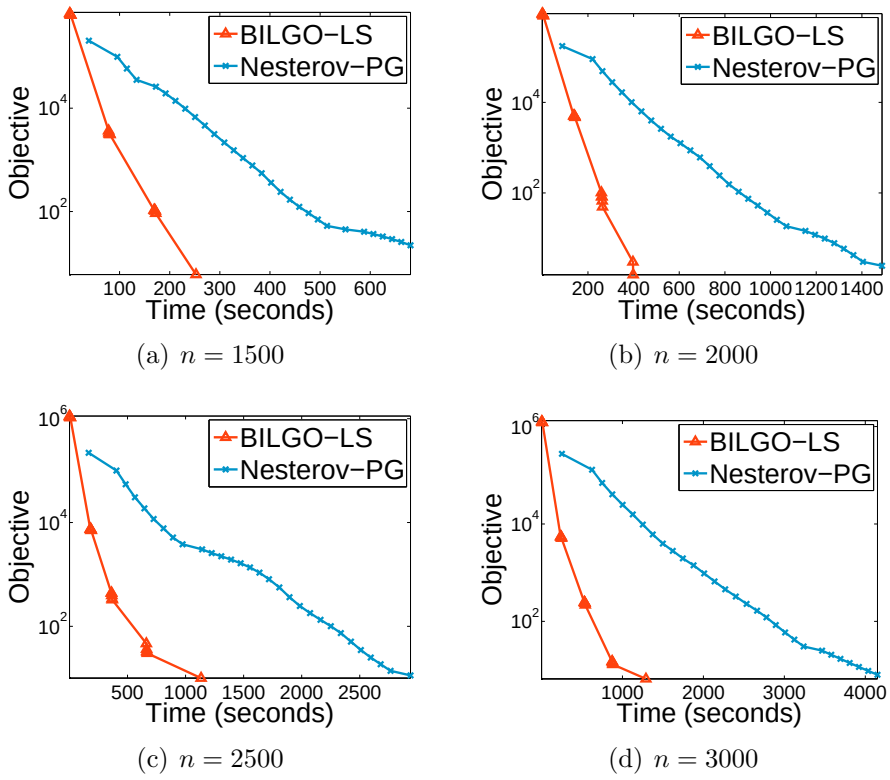
1. As for efficiency, it is demonstrated in Table 1 that BILGO significantly outperforms the full rank methods. While ITML remains competitive in efficiency on low dimensional datasets, it cannot utilize the low rank property of the solution and thus rapidly becomes intractable as the dimensionality d grows (e.g. in the realsim dataset). Moreover, the greedy BILGO-LMNN method gives comparable results to the low rank method SURF, which suffers from slow convergence, if the randomized initial solution is far from the global solution. Since BILGO implicitly stores $\mathbf{M}$ using the low rank representation $\mathbf{L}$ and uses a sparse eigenvalue solver, it usually obtains a good solution in a few iterations (also see Figure 2).

[12] Since the required rank of BILGO-KTA-LS is often small, which may not be suitable for SURF, we assume r is lower-bounded by 15.

Table 2: Comparison of error rates with state-of-the-art metric learning solvers. OOM indicates "out of memory", and OOT indicates "out of time".

Data Set	SURF	ITML	Boost-Metric	LMNN	BILGO-LMNN	KTA-Lbfgs	BILGO-KTA	BILGO-KTA-LS
splice	21.5±1.1	32.3±1.8	16.8±1.0	19.6±1.2	18.1±1.9	19.4±1.4	18.9±1.0	20.4±1.6
isolet	5.5±1.3	7.8±03.2	9.4±0.7	8.5±1.1	7.8±2.1	5.4±1.2	5.9±1.5	5.4±1.2
optdigits	2.4±0.3	1.9±0.1	1.8±0.1	1.4±0.0	2.4±0.2	2.3±0.5	2.4±0.1	2.5±0.5
dna	8.2±1.0	9.8±0.9	6.2±1.1	4.8±1.0	5.2±0.5	4.8±0.4	5.9±0.5	4.8±0.2
a1a	18.3±1.9	18.3±0.3	18.1±1.2	20.3±2.1	18.3±1.2	19.0±1.0	17.9±0.1	18.2±1.0
protein	38.2±3.3	41.3±2.8	36.1±2.4	39.1±3.2	40.1±2.3	37.9±2.3	36.3±3.3	36.9±2.3
mushrooms	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.1±0.1	0.0±0.0	0.0±0.0
w1a	2.1±0.1	1.5±0.0	1.8±0.0	2.4±0.1	1.2±0.1	2.1±0.0	1.5±0.1	2.1±0.0
usps	4.6±1.0	5.0±1.5	5.0±0.0	4.3±0.1	5.2±1.0	5.0±0.9	4.9±1.1	5.1±0.9
mnist	6.3±0.4	3.5±0.4	OOT	7.3±2.3	6.1±0.8	5.1±1.2	5.1±0.1	5.3±1.2
gisette	5.0±1.0	OOT	OOT	OOT	4.5±1.0	4.0±1.1	4.3±1.2	4.0±1.1
realsim	4.0±1.5	OOM	OOM	OOM	4.2±0.5	3.8±1.0	3.0±1.5	3.7±1.1

2. In terms of accuracy, we observe in Table 2 that LMNN is still one of the most accurate metric learning solvers. However, it lacks stability, especially for large scale datasets, mainly due to the large number of constraints. As opposed to LMNN that picks and cycles through a subset of metric constraints, the KTA model can handle all the constraints implicitly. That explains why KTA implementations (BILGO-KTA and BILGO-KTA-LS) are faster and more stable. We observe that BILGO-KTA-LS and BILGO-KTA perform with consistently high stability in both problem domains.
3. We study the impact of the local search on a metric learning classifier. Although BILGO-KTA-LS iteratively reduces the objective, it does not necessarily achieve better accuracy than BILGO-KTA. Thus, an approximate solution is often good enough for a metric learning task. Moreover, BILGO-KTA-LS often takes less time to converge than BILGO-KTA, because the use of L-bfgs enables its convergence rate to be super-linear.
4. Finally, we study the impact of the greedy strategy on a metric learning classifier. Note that BILGO-KTA-LS and KTA-Lbfgs use the same objective function and stopping criterion. We observe that BILGO-KTA-LS is about 2 times faster than KTA-Lbfgs. This is the case because KTA-Lbfgs is initialized randomly leading to slow convergence, while BILGO-KTA-LS benefits greatly from the efficient matrix-free sparse eigenvalue solver and quickly obtains a good solution.



(a) $n = 1500$ (b) $n = 2000$

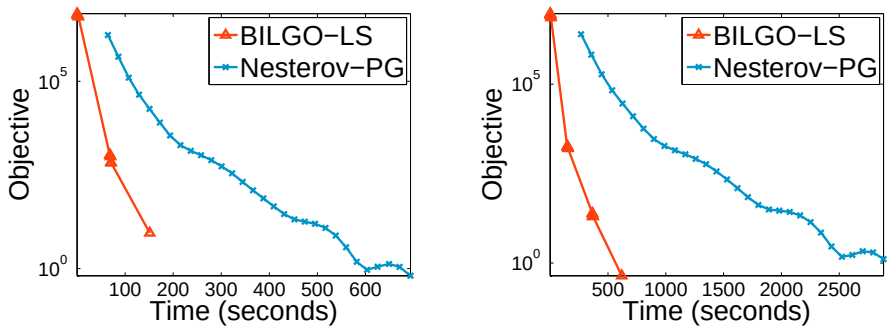
(c) $n = 2500$ (d) $n = 3000$

Figure 3: Convergence behavior of BILGO (red) and Nesterov's projective gradient (blue) on the 'w1a' dataset.

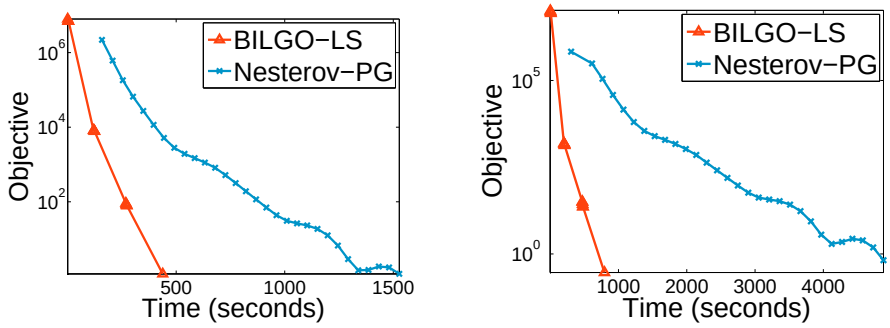
6.3. Efficiency on Maximum Variance Unfolding

In this section, we demonstrate the efficiency of BILGO by applying it to maximum variance unfolding (MVU) for manifold learning. The MVU problem, as we have mentioned in the introduction section, can be solved by interior point method or projected gradient descent [1, 10]. However, the interior point method will become rapidly intractable as the number of training instances become large due to its time complexity of $\mathcal{O}(n^{6.5})$. Therefore, we drop the comparisons with the interior point method and only compare against Nesterov's first order optimal method [30, 10], which is known to achieve a much faster convergence rate than the traditional methods such as subgradient or naïve projected gradient descent methods. So, comparison with the BILGO algorithms is meaningful.

We verify the efficiency of BILGO-LS by demonstrating the asymptotic



(a) $n = 1500$ (b) $n = 2000$



(c) $n = 2500$ (d) $n = 3000$

Figure 4: Convergence behavior of BILGO (red) and Nesterov's projective gradient (blue) on the 'mnist' dataset.

behavior of both BILGO-LS (BILGO with local search) and Nesterov-PG (Nesterov's projective gradient method) on two datasets: 'w1a' and 'mnist'. To generate data, we randomly sample n points from the datasets. To generate the indicator matrix $\mathbf{U}$, we use the 3-NN graph. We have verified that the 3-NN graph derived from our data is connected. We solve the plain MVU optimization problem using both methods. For BILGO-LS, we use low-rank optimization to accelerate the algorithm. In every 5 iterations, a local-search L-bfgs is performed to refine the solution $\mathbf{L}$. The computational results of both methods are shown in Figure 3 and Figure 4. We observe that the objective values of BILGO-LS converge more quickly than those of Nesterov-PG. For a small-scale MVU problem, Nesterov-PG is comparable to BILGO-LS. However, since full eigenvalue decomposition is an expensive operator especially for high dimensional matrix spaces, Nesterov-PG shows poor performance for large scale problems. We also report the objectives and

Table 3: Comparisons with Nesterov's projective gradient method. The results separated by '/' are objective and time (in seconds), respectively.

	w1a		mnist	
n	BILGO-LS	Nesterov-PG	BILGO-LS	Nesterov-PG
500	4.95/23.67	9.23/28.07	0.35/20.57	0.11/22.96
1000	0.48/124.81	4.01/207.01	0.29/123.28	0.48/203.57
1500	6.04/252.35	22.24/680.23	0.25/251.10	0.64/693.59
2000	1.56/398.39	2.45/1488.12	1.16/437.90	1.16/1527.46
2500	10.05/1133.15	11.33/2936.65	0.44/619.67	1.29/2882.82
3000	6.60/1290.15	8.00/4150.07	0.29/795.64	0.65/4880.54

run-times for both BILGO-LS and Nesterov-PG, as n is increased in Table 3. Here, the improvement of BILGO algorithm over that of Nesterov-PG, in terms of efficiency, is more obvious. For example, for an MVU problem of size 3000 (on 'mnist' dataset), BILGO-LS is six times faster than Nesterov-PG while also achieving a lower objective value.

6.4. Hessian-Free Newton for computing the leading eigenvector

Finally, we evaluate the efficiency of our Hessian-free Newton method in computing the leading eigenvector of a large-scale matrix. Two versions of this method (denoted 'Newton-Lanczos' and 'Newton-CG' respectively) are tested in this experiment. For comparison, we include the popular Matlab function 'eigs' [31]. We simply initialize $v^0 =$rand$(d, 1)$ in MATLAB and stop the algorithm when the relative change in f is less than 10^{-12}. We use the default stopping criterion for 'eigs'. The matrix $\mathbf{A}$ is generated randomly and the three algorithms {'eigs', 'Newton-CG', 'Newton-Lanczos'} are used to estimate the leading eigenvector of $\mathbf{A}$. Given the true eigenvector $\tilde{\mathbf{v}}_{\max}$ and the output $\tilde{\mathbf{v}}$ from the algorithms, the accuracy is defined as $1 - \frac{\tilde{\mathbf{v}}_{\max}^T \mathbf{A} \tilde{\mathbf{v}}_{\max}}{\tilde{\mathbf{v}}^T \mathbf{A} \tilde{\mathbf{v}}}$. The results in Table 4 show that the Newton-Lanczos is consistently more than two times faster than Newton-CG. Newton-Lanczos is also reasonably faster than the Matlab solver for large dimensional matrices.

7. Conclusions and Future Work

In this paper, we theoretically analyse a new *bilateral greedy optimization*(denoted BILGO) strategy in solving general semidefinite programs on large-scale datasets. Utilizing a new bilateral greedy optimization strategy, BILGO is capable of efficiently finding the global minimum of the SDP problem. When the objective is continuously differentiable, BILGO enjoys a sublinear convergence rate. However, exploiting a well designed local-search

Table 4: Comparisons with 'eigs'. The results separated by '/' are accuracy and time (in seconds), respectively.

d	eigs	Newton-CG	Newton-Lanczos
1000	0.00e-00/00.23	2.17e-16/00.72	2.77e-16/00.41
2000	0.00e-00/01.35	1.11e-15/02.80	1.55e-15/01.30
3000	1.58e-16/03.91	6.61e-15/06.19	0.00e-00/03.51
4000	3.57e-15/16.01	0.00e-00/14.62	3.16e-15/11.00
5000	1.71e-15/09.82	2.08e-14/19.38	0.00e-00/09.44
6000	0.00e-00/17.35	1.24e-14/41.65	1.67e-15/12.72
7000	4.13e-16/37.31	7.22e-15/49.76	0.00e-00/26.66
8000	0.00e-00/39.42	2.07e-14/75.19	8.87e-15/27.03
9000	3.82e-15/65.56	5.10e-15/120.1	0.00e-00/46.17

low-rank optimization strategy, BILGO can provide huge savings in computational cost and achieve a superlinear convergence rate. We apply BILGO to two important machine learning tasks: Mahalanobis metric learning and maximum variance unfolding. Extensive experiments show that BILGO is effective in efficiently solving large-scale machine learning tasks (e.g. metric and manifold learning) that can be formulated as SDP problems.

Our future work is most likely to proceed along three directions. Firstly, past study [32, 33] has shown that many non-smooth and linear constrained optimization problems with an appropriate simple primal-dual minimax structure can be solved by Nesterov's smoothing technique. We plan to apply the bilateral and low-rank optimization to solve non-smooth and linear constrained semidefinite programming problems [18]. Secondly, BILGO is a greedy monotone algorithm with sublinear convergence rate, it is equally interesting to study the theoretical behavior of the additional 'away' step [34, 22] in BILGO and consider boosting BILGO to achieve linear convergence rate. Finally, we are also interested in extending the rank-1 matrix update scheme to linear support vector machines [27] and tensor subspace analysis [35].

Acknowledgments

We would like to thank Dr. Zhenjie Zhang for his helpful discussions and suggestions on this paper. Hao and Yuan are supported by NSF-China (61070033, 61100148), NSF-Guangdong (9251009001000005, S2011040004804).

References

[1] A. Globerson, S. Roweis, Metric learning by collapsing classes, in: NIPS, 2006, pp. 451–458.

[2] G. Yuan, Z. Zhang, B. Ghanem, Z. Hao, Low-rank quadratic semidefinite programming, Neurocomputing 106 (0) (2013) 51–60.

[3] K. Q. Weinberger, F. Sha, L. K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, in: ICML, 2004, pp. 106–113.

[4] L. Song, A. J. Smola, K. M. Borgwardt, A. Gretton, Colored maximum variance unfolding, in: NIPS, 2007, pp. 1385–1392.

[5] X.-M. Wu, A. M.-C. So, Z. Li, S.-Y. R. Li, Fast graph laplacian regularized kernel learning via semidefinite-quadratic-linear programming, in: NIPS, 2009, pp. 1964–1972.

[6] G. Meyer, S. Bonnabel, R. Sepulchre, Regression on fixed-rank positive semidefinite matrices: A riemannian approach, Journal of Machine Learning Research (JMLR) 12 (2011) 593–625.

[7] B. Mishra, G. Meyer, R. Sepulchre, Low-rank optimization for distance matrix completion, in: Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 2011.

[8] B. Vandereycken, S. Vandewalle, A riemannian optimization approach for computing low-rank solutions of lyapunov equations, SIAM Journal on Matrix Analysis and Applications (SIMAX) 31 (5) (2010) 2553–2579.

[9] J. V. Davis, B. Kulis, P. Jain, S. Sra, I. S. Dhillon, Information-theoretic metric learning, in: ICML, 2007, pp. 209–216.

[10] J. Liu, S. Ji, J. Ye, Multi-task feature learning via efficient $l_{2,1}$-norm minimization, in: UAI, 2009, pp. 339–348.

[11] Z. Wen, D. Goldfarb, K. Scheinberg, Block coordinate descent methods for semidefinite programming, Handbook on Semidefinite, Conic and Polynomial Optimization (2012) 533–564.

[12] S. Burer, R. D. C. Monteiro, Local minima and convergence in low-rank semidefinite programming, Mathematical Programming 103 (2005) 427–444.

[13] E. Hazan, Sparse approximate solutions to semidefinite programs, in: Proceedings of Latin American conference on Theoretical INformatics (LATIN), 2008, pp. 306–316.

[14] M. Jaggi, M. Sulovsky, A simple algorithm for nuclear norm regularized problems, in: ICML, 2010, pp. 471–478.

[15] M. Journée, F. Bach, P.-A. Absil, R. Sepulchre, Low-rank optimization on the cone of positive semidefinite matrices, SIAM Journal on Optimization (SIOPT) 20 (2010) 2327–2351.

[16] C. Shen, J. Kim, L. Wang, A. van den Hengel, Positive semidefinite metric learning using boosting-like algorithms, The Journal of Machine Learning Research (JMLR) 98888 (2012) 1007–1036.

[17] S. Shalev-Shwartz, A. Gonen, O. Shamir, Large-scale convex minimization with a low-rank constraint, in: ICML, 2011, pp. 329–336.

[18] A. Kleiner, A. Rahimi, M. I. Jordan, Random conic pursuit for semidefinite programming, in: NIPS, 2010, pp. 1135–1143.

[19] S. Laue, A hybrid algorithm for convex semidefinite optimization, in: ICML, 2012.

[20] D. P. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scientific, 1999.

[21] Y. Nesterov, Introductory lectures on convex optimization: a basic course, Kluwer Academic Publishers, 2004.

[22] K. L. Clarkson, Coresets, sparse greedy approximation, and the frank-wolfe algorithm, ACM Transactions on Algorithms (TALG) 6 (4) (2010) 63:1–63:30.

[23] J. C. Dunn, Newton's method and the goldstein step-length rule for constrained minimization problems, SIAM Journal on Control and Optimization (SICON) (1980) 659–674.

[24] C. C. Paige, M. A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM Journal on Numerical Analysis (SINUM) 12 (1975) 617–629.

[25] E. J. Candès, B. Recht, Exact matrix completion via convex optimization, Foundations of Computational Mathematics (FoCM) 9 (6) (2009) 717–772.

[26] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.

[27] C. Hsieh, K. Chang, C. Lin, S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear svm, in: ICML, 2008, pp. 408–415.

[28] S. Wang, R. Jin, An information geometry approach for distance metric learning., in: AISTATS, Vol. 5, 2009, pp. 591–598.

[29] K. Weinberger, L. Saul, Distance metric learning for large margin nearest neighbor classification, The Journal of Machine Learning Research (JMLR) 10 (2009) 207–244.

[30] Y. E. Nesterov, Introductory lectures on convex optimization: a basic course, Vol. 87 of Applied Optimization, Kluwer Academic Publishers, Boston, 2003.

[31] D. C. Sorensen, Implicit application of polynomial filters in a k-step arnoldi method, SIAM Journal on Matrix Analysis and Applications (SIMAX) 13 (1992) 357–385.

[32] Y. Nesterov, Smooth minimization of non-smooth functions, Mathematical Programming 103 (1) (2005) 127–152.

[33] Y. Nesterov, Primal-dual subgradient methods for convex problems, Mathematical programming 120 (1) (2009) 221–259.

[34] S. Damla Ahipasaoglu, P. Sun, M. J. Todd, Linear convergence of a modified frank–wolfe algorithm for computing minimum-volume enclosing ellipsoids, Optimisation Methods and Software 23 (1) (2008) 5–19.

[35] Z. Hao, L. He, B. Chen, X. Yang, A linear support higher-order tensor machine for classification, IEEE Transactions on Image Processing (TIP), 2013 (To appear).doi:10.1109/TIP.2013.2253485.