Low-Rank Matrix Learning Using Biconvex Surrogate Minimization

En-Liang Hu and James T. Kwok, *Fellow, IEEE*

***Abstract*— Many machine learning problems involve learning a low-rank positive semidefinite matrix. However, existing solvers for this low-rank semidefinite program (SDP) are often expensive. In this paper, by factorizing the target matrix as a product of two matrices and using a Courant penalty to penalize for their difference, we reformulate the SDP as a biconvex optimization problem. This allows the use of multiconvex optimization techniques to define simple surrogates, which can be minimized easily by block coordinate descent. Moreover, while traditionally this biconvex problem approaches the original problem only when the penalty parameter is infinite, we show that the two problems are equivalent when the penalty parameter is sufficiently large. Experiments on a number of SDP applications in machine learning show that the proposed algorithm is as accurate as other state-of-the-art algorithms, but is much faster, especially on large data sets.**

***Index Terms*— Distance learning, mathematical programming, pattern clustering, recommender systems, semisupervised learning.**

I. Introduction

In this paper, we consider the semidefinite program (SDP)

$$\min_{Z\succeq 0} \tilde{f}(Z) \tag{1}$$

where $Z \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite (PSD), and $\tilde{f}$ is convex differentiable over the PSD cone. Many machine learning problems can be formulated in this way. Examples include sparse principal component analysis (PCA) [1], distance metric learning [2], nonlinear dimensionality reduction [3], kernel learning [4], [5], multitask learning [6], and matrix completion [7].

A standard SDP solver is the interior point method [8], which produces highly accurate solutions. However, each SDP iteration takes $O(n^3)$ time, and thus is not scalable. Moreover, a highly accurate solution is often not needed in practice. Besides the interior point method, methods such as the projected subgradient method [9] have also been used. However, an expensive matrix eigendecomposition is required in each iteration.

Manuscript received October 24, 2017; revised September 20, 2018, January 29, 2019 and June 17, 2019; accepted July 5, 2019. Date of publication August 9, 2019; date of current version October 29, 2019. This work was supported by the National Natural Science Foundation of China under Grant 61663049, Grant 61165012, and Grant 2019010108203008. *(Corresponding author: En-Liang Hu.)*

E.-L. Hu is with the Department of Mathematics, Yunnan Normal University, Cheng Gong Campus, Kunming 6505001-2, China (e-mail: ynel.hu@gmail.com).

J. T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2927819

In many matrix learning applications, the target matrix is low-rank [10], [11]. A popular low-rank SDP solver is the Frank–Wolfe (FW) algorithm [11]. A hybrid accelerated algorithm combining FW with limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) has also been proposed in [12]. On smooth problems, each FW iteration takes $O(N\log(n)/\sqrt{\epsilon})$ arithmetic operations (where N is the number of nonzero entries in $\nabla\tilde{f}(Z_k)$ and Z_k is the iterate at the kth iteration), and the FW algorithm converges to an ϵ-accurate solution in $O(1/\epsilon)$ iterations [11]. This leads to a total of $O(N/\epsilon^{1.5})$ arithmetic operations. When $\nabla\tilde{f}(Z_k)$ is dense, FW can be expensive.

Instead of storing the $n\times n$ matrix Z, one can enforce the low-rank assumption by using a $n\times r$ matrix X such that

$$Z = XX^\top \tag{2}$$

where r is the rank of Z. This is particularly appealing as the computational effort is drastically reduced. For example, matrix-vector multiplications can be performed in $O(nr)$ time [instead of $O(n^2)$]. By using (2), the SDP in (1) is converted to a nonconvex program (NCP) [13], [14]

$$\min_{X} \tilde{f}(XX^\top). \tag{3}$$

A local minimizer $\hat{X}$ of (3) is also a minimizer of (1) when $\text{rank}(\hat{X}) \le r$ [14], [15]. For the standard SDP with a linear objective and linear constraints, the reformulated NCP can be solved with L-BFGS [13]. However, its convergence is unclear. Block coordinate descent (BCD) has also been used to solve some NCP problems in [5], but each block coordinate update needs to have a closed-form solution.

Instead of factorizing Z as $XX^\top$, Hu and Kwok [16] recently considered another optimization approach for the special case of nonparametric kernel learning (NPKL) [17]. Given n samples, let $\mathcal{M}$ be the (must-link) set containing sample pairs that should belong to the same class, and $\mathcal{C}$ be the (cannot-link) set containing pairs that should not belong to the same class. NPKL can be formulated as the following optimization problem:

$$\min_{Z\succeq 0} \text{tr}(ZL) + \frac{\lambda}{2}\sum_{(i,j)\in\mathcal{M}\cup\mathcal{C}\cup\{i=j\}} (Z_{ij}-T_{ij})^2 \tag{4}$$

where $Z=[Z_{ij}]\in\mathbb{R}^{n\times n}$ is the target kernel matrix on the n samples to be learned, L is the graph Laplacian matrix of the data, $T=[T_{ij}]$ with $T_{ij}=1$ if $(i,j)\in\mathcal{M}$ or $i=j$, and 0 if $(i,j)\in\mathcal{C}$, and λ is a tradeoff parameter. The $\text{tr}(ZL)$ term encourages smoothness on the data manifold by aligning Z with L, while the second term measures the difference between

2162-237X © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Z_{ij} and the must-link/cannot-link constraint encoded in T_{ij}. Note that problem (4) is of the form in (1). For this special case, Hu and Kwok [16] reformulated (1) as

$$\min_{X,Y\in\mathbb{R}^{n\times r}} \tilde{f}(XY^\top) \ : \ X = Y. \tag{5}$$

This can then be solved with the alternating direction method of multipliers (ADMM) [18] and converges to an optimal solution.

However, for general $\tilde{f}$, the ADMM's subproblems for (5) can be difficult to solve, and its convergence is still open. In addition, for the machine learning applications to be considered in Section IV, the number of constraints (e.g., pairwise must-link/cannot-link constraints in kernel learning, values of the observed entries in matrix completion, and distances between similar/dissimilar examples in metric learning) can be very large. This increases the size of the ADMM subproblem in [16] and limits its scalability.

In this paper, we also factorize Z as $XY^\top$ as in [16] to take advantage of the low-rank property. However, to alleviate the above-mentioned problems, we replace the strict equality between X and Y by a penalty on their difference. A key observation is that the resultant optimization objective is biconvex (which will be defined in Section II). This allows the use of multiconvex optimization techniques [19] to define simple surrogates, which can be minimized easily by BCD [20]. Empirical results on a variety of SDP applications in machine learning show that the resultant computational gains are substantial.

The rest of this paper is organized as follows. Section II reviews the related work on multiconvex optimization. The proposed algorithm is presented in Section III. Experimental results are reported in Section IV, and the last section gives some concluding remarks.

Notation: The transpose of a vector or matrix is denoted by the superscript ^{T}. For a matrix $A = [A_{ij}]$, $\mathrm{tr}(A)$ is its trace, $\|A\| = (\sum_{ij} A_{ij}^2)^{1/2}$ is its Frobenius norm, and $A \succeq 0$ means that it is symmetric PSD. For two matrices A and B, $\langle A, B\rangle = \mathrm{tr}(A^\top B)$. Moreover, $[\cdot]_+ = \max\{\cdot, 0\}$, and $\nabla_x f(\cdot)$ is the derivative of $f(\cdot)$ with respect to variable x. A convex function f is L-smooth if for any x_1 and x_2, there exists a Lipschitz constant $0 < L < \infty$ such that $\|\nabla_x f(x_1) - \nabla_x f(x_2)\| \le L\|x_1 - x_2\|$.

II. Related Work: Multiconvex Optimization

BCD has long been a popular optimization tool [20]. It alternates approximate minimizations along different coordinate directions or hyperplanes. Generally, each BCD subproblem is much easier to solve than the original optimization problem. Because of its simplicity and efficiency in the big data setting, it has recently witnessed a resurgence of interest in the machine learning community [21]–[26].

When the objective is convex with Lipschitz-continuous gradient, the sequence of objective values obtained by BCD converges to within ϵ of the optimal value at a rate of $O(1/\epsilon)$ [27]. When acceleration (e.g., Nesterov's extrapolation method [28]) is used, this can be improved to $O(1/\sqrt{\epsilon})$ [27]. When the objective is strongly convex, the convergence rate becomes linear ($O(\log(1/\epsilon))$ [27].

For nonconvex objectives, the convergence properties of BCD are open. In particular, Powell [29] provided a simple but intriguing example on which BCD fails to converge to a stationary point. However, for nonconvex problems with multiconvex objective, the following recent result shows that BCD converges.

Definition 1 [30] (Block Multiconvex Set): A set $\chi \in \mathbb{R}^n$ is *block multiconvex* with respect to the partition $\{x^1, \ldots, x^s\}$ if the projection of χ to each component block is convex. In other words, for $i \in \{1, \ldots, s\}$, the set $\chi^i(x^1, \ldots, x^{i-1}, x^{i+1}, \ldots, x^s) \equiv \{x^i \in \mathbb{R}^n : (x^1, \ldots, x^{i-1}, x^i, x^{i+1}, \ldots, x^s) \in \chi\}$ is convex.

Definition 2 [30] (Block Multiconvex Function): A function $H(x)$, in which x is decomposed into s blocks $\{x^1, x^2, \ldots, x^s\}$, is block multiconvex if for each i, $H(x)$ is convex in x^i with all the other blocks fixed.

Consider the problem

$$\min_{x\in\chi} H(x^1, x^2, \ldots, x^s) \tag{6}$$

where H is differentiable and block multiconvex. x is decomposed into s blocks $\{x^1, x^2, \ldots, x^s\}$, and χ is a closed and block multiconvex subset of $\mathbb{R}^n$. Using BCD with the Gauss–Seidel update, H is cyclically minimized over each of $\{x^1, \ldots, x^s\}$ while fixing the remaining blocks at their last updated values. Specifically, let x_k^i be the value of x^i after the kth iteration, and $H_k^i(x^i) \equiv H(x_k^1, \ldots, x_k^{i-1}, x^i, x_{k-1}^{i+1}, \ldots, x_{k-1}^s)$. Xu and Yin [20] proposed three BCD update schemes that minimize different surrogate functions dominating the original objective. It can be shown that all these converge to a stationary point.

1) *Standard:*

$$x_k^i = \arg\min_{x^i\in\chi_k^i} H_k^i(x^i) \tag{7}$$

where $\chi_k^i = \chi^i(x_k^1, \ldots, x_k^{i-1}, x_{k-1}^{i+1}, \ldots, x_{k-1}^s)$.

2) *Proximal:* This includes an additional quadratic proximal regularization term in the surrogate objective

$$x_k^i = \arg\min_{x^i\in\chi_k^i} H_k^i(x^i) + \frac{L_{k-1}^i}{2}\left\|x^i - x_{k-1}^i\right\|^2 \tag{8}$$

where L_{k-1}^i is uniformly lower bounded from zero and uniformly upper bounded.

3) *Prox-Linear:* This scheme approximates $H_k^i(x_k^i)$ by the quadratic function $H_k^i(\hat{x}_{k-1}^i) + \langle \hat{g}_{k-1}^i, x_k^i - \hat{x}_{k-1}^i\rangle + \frac{L_{k-1}^i}{2}\left\|x_k^i - \hat{x}_{k-1}^i\right\|^2$ around the extrapolated iterate

$$\hat{x}_{k-1}^i = x_{k-1}^i + \omega_{k-1}^i\left(x_{k-1}^i - x_{k-2}^i\right) \tag{9}$$

where $\hat{g}_{k-1}^i = \nabla H_k^i(\hat{x}_{k-1}^i)$, and $\omega_{k-1}^i \ge 0$ is an extrapolation weight. The iterates are then obtained as

$$x_k^i = \arg\min_{x^i\in\chi_k^i} \langle \hat{g}_{k-1}^i, x^i - \hat{x}_{k-1}^i\rangle + \frac{L_{k-1}^i}{2}\left\|x^i - \hat{x}_{k-1}^i\right\|^2. \tag{10}$$

Due to multiconvexity of H, subproblems (7), (8), and (10) are all convex. Empirically, scheme (10) produces solutions of

lower objective values [19]. Moreover, subproblems (7) and (8) may sometimes be difficult to solve, whereas the gradient descent (GD) step in (10) is always simple and often faster than solving (7) and (8).

III. Biconvex Surrogate Minimization

In this section, similar to [16], we also factorize Z in (1) as $XY^\top$, where $X, Y \in \mathbb{R}^{n\times r}$. However, the strict equality between X and Y in (5) is replaced by a penalty on their difference, leading to the optimization problem

$$\min_{X,Y\in\mathbb{R}^{n\times r}} F(X,Y) \equiv f(X,Y) + \frac{\gamma}{2}\|X-Y\|^2 \tag{11}$$

where

$$f(X,Y) = \tilde{f}(XY^\top) \tag{12}$$

and $\gamma > 0$ is a penalty parameter (whose setting will be discussed in Section III-C). The second term in (11) is the classic quadratic (or Courant) penalty for the constraint $X = Y$ [31].

In general, problem (11) (in penalty form) approaches the original problem (5) (in constrained form) only when γ is infinite. However, we will show in Section III-B that when γ is finite but sufficiently large, the X and Y solutions obtained are the same, and thus any stationary point of (11) is also a stationary point of (3).

A. Optimizing (11)

First, we make the following assumptions on f in (12).

Assumption 1: $f(X,Y)$ is lower bounded.

Assumption 2: For an arbitrary constant $U \in \mathbb{R}^{n\times r}$, $f(\cdot, U)$ is L^X-smooth, $f(U,\cdot)$ is L^Y-smooth.

The following proposition shows that the objective in (11) is biconvex if $\tilde{f}$ in (1) is convex.

Definition 3: A function $g(x,y)$ is biconvex if g is multiconvex with respect to x and y.

Proposition 1: If $\tilde{f}$ in (1) is convex, then f in (12) and F in (11) are biconvex.

Let L^X_{k-1} and L^Y_{k-1} be the local Lipschitz constants of $\nabla_X f(X, Y_{k-1})$ and $\nabla_Y f(X_k, Y)$, respectively. These can be estimated by line search or backtracking [32]. Obviously, $L^X_{k-1}+\gamma$ and $L^Y_{k-1}+\gamma$ are then the local Lipschitz constants of $\nabla_X F(X, Y_{k-1})$ and $\nabla_Y F(X_k, Y)$, respectively. Since F is biconvex (and thus multiconvex), using the prox-linear scheme in (10), we have

$$X_k = \arg\min_X Q_X(X; \hat{X}_{k-1}, Y_{k-1}) \tag{13}$$

$$Y_k = \arg\min_Y Q_Y(Y; \hat{Y}_{k-1}, X_k) \tag{14}$$

where

$$\begin{aligned} Q_X(X;\hat{X}_{k-1},Y_{k-1}) &= \langle \nabla_X F(\hat{X}_{k-1},Y_{k-1}), X-\hat{X}_{k-1}\rangle \\ &\quad + \frac{L^X_{k-1}+\gamma}{2}\|X-\hat{X}_{k-1}\|^2 \\ Q_Y(Y;X_k,\hat{Y}_{k-1}) &= \langle \nabla_Y F(X_k,\hat{Y}_{k-1}), Y-\hat{Y}_{k-1}\rangle \\ &\quad + \frac{L^Y_{k-1}+\gamma}{2}\|Y-\hat{Y}_{k-1}\|^2 \end{aligned}$$

and $\hat{X}_{k-1}$ and $\hat{Y}_{k-1}$ are defined as in (9)

$$\hat{X}_{k-1} = X_{k-1} + \omega^X_{k-1}(X_{k-1}-X_{k-2}) \tag{15}$$

$$\hat{Y}_{k-1} = Y_{k-1} + \omega^Y_{k-1}(Y_{k-1}-Y_{k-2}) \tag{16}$$

with $\omega^X_{k-1}, \omega^Y_{k-1} \geq 0$ being the extrapolation weights. Following [19], the extrapolation weights are set as:

$$\omega^X_{k-1} = \min\left\{\omega_{k-1}, \delta_\omega\sqrt{L^X_{k-2}/L^X_{k-1}}\right\} \tag{17}$$

$$\omega^Y_{k-1} = \min\left\{\omega_{k-1}, \delta_\omega\sqrt{L^Y_{k-2}/L^Y_{k-1}}\right\} \tag{18}$$

where $0 < \delta_\omega < 1$, $\omega_{k-1} = (t_{k-1}-1/t_k)$, with $t_k = (1/2)(1+(1+4t^2_{k-1})^{1/2})$ and $t_0 = 1$.

On setting the derivatives of Q_X and Q_Y to zero, we obtain the following simple closed-form solutions for X_k and Y_k.

Proposition 2: The closed-form solutions for (13) and (14) are

$$X_k = \hat{X}_{k-1} - \frac{1}{L^X_{k-1}+\gamma}\nabla_X F(\hat{X}_{k-1}, Y_{k-1}) \tag{19}$$

and

$$Y_k = \hat{Y}_{k-1} - \frac{1}{L^Y_{k-1}+\gamma}\nabla_Y F(X_k, \hat{Y}_{k-1}) \tag{20}$$

respectively.

Remark 1: In (19), X_k can be viewed as performing GD on F (with the Y-component fixed at Y_{k-1}) at $\hat{X}_{k-1}$. Similarly, Y_k in (20) can be viewed as performing GD on F (with the X-component fixed at X_k) at $\hat{Y}_{k-1}$. Thus, minimizing f can be viewed as alternating GD based on the iterates $\hat{X}_{k-1}$ and $\hat{Y}_{k-1}$. When extrapolation is not used (i.e., $\omega^X_{k-1} = \omega^Y_{k-1} = 0$), $\hat{X}_{k-1}$ reduces to X_{k-1}, $\hat{Y}_{k-1}$ reduces to Y_{k-1}, and Proposition 2 reduces to

$$X_k = X_{k-1} - \frac{1}{L^X_{k-1}+\gamma}\nabla_X F(X_{k-1}Y_{k-1}) \tag{21}$$

$$Y_k = Y_{k-1} - \frac{1}{L^Y_{k-1}+\gamma}\nabla_Y F(X_k, Y_{k-1}). \tag{22}$$

This is the same as performing standard alternating GD on F.

Corollary 1: X_k can be rewritten as the convex combination

$$X_k = \alpha\hat{X}^{gd}_{k-1} + (1-\alpha)Y_{k-1} \tag{23}$$

where $\alpha = L^X_{k-1}/L^X_{k-1}+\gamma$, and

$$\hat{X}^{gd}_{k-1} = \hat{X}_{k-1} - \frac{1}{L^X_{k-1}}\nabla_X f(\hat{X}_{k-1}, Y_{k-1}) \tag{24}$$

is the GD update of $f(\cdot, Y_{k-1})$ at $\hat{X}_{k-1}$. Similarly, Y_k can be rewritten as the convex combination

$$Y_k = \beta\hat{Y}^{gd}_{k-1} + (1-\beta)X_k \tag{25}$$

where $\beta = L^Y_{k-1}/L^Y_{k-1}+\gamma$, and

$$\hat{Y}^{gd}_{k-1} = \hat{Y}_{k-1} - \frac{1}{L^Y_{k-1}}\nabla_Y f(X_k, \hat{Y}_{k-1}) \tag{26}$$

is the GD update of $f(X_k,\cdot)$ at $\hat{Y}_{k-1}$.

The theoretical procedure for optimizing (11) is shown in Algorithm 1. Although existing algorithms require closed-form coordinate descent update or solving complicated ADMM subproblems, here only simple descent steps are needed.

Algorithm 1 Theoretical BiS Minimization for (11).

Initialization: $X_{-1} = X_0$, $Y_{-1} = Y_0$, $\gamma > 0$.
for $k = 1, 2, \ldots$ **do**
  obtain $\hat{X}_{k-1}$ from X_{k-1} and X_{k-2} using (15);
  obtain $\hat{Y}_{k-1}$ from Y_{k-1} and Y_{k-2} using (16);
  obtain X_k from $\hat{X}_{k-1}$ and Y_{k-1} using (23);
  obtain Y_k from $\hat{Y}_{k-1}$ and X_k using (25);
  if converged **then**
    return $Z = Y_k Y_k^\top$ (or $X_k Y_k^\top$).
  end if
end for

B. Convergence

In the following, we consider $\omega_{k-1}^X \in [0, \delta_\omega (L_{k-2}^X / L_{k-1}^X)^{1/2}]$ and $\omega_{k-1}^Y \in [0, \delta_\omega (L_{k-2}^Y / L_{k-1}^Y)^{1/2}]$ for some $0 < \delta_\omega < 1$. Note that this includes the setting in (17) and (18).

Proposition 3 (Lemma 2.2, [19]): The sequence $\{(X_k, Y_k)\}$ generated by Algorithm 1 converges to a limit point $(\bar{X}, \bar{Y})$, where $\bar{X} = \lim_{k\to\infty} X_k$ and $\bar{Y} = \lim_{k\to\infty} Y_k$. Moreover, $(\bar{X}, \bar{Y})$ is a stationary point of (11).

Define

$$\mu_k^X \equiv \left[L_{k-1}^X + \frac{2\mathrm{tr}[(Y_{k-1} - \hat{X}_{k-1})^\top \nabla_X f(\hat{X}_{k-1}, Y_{k-1})]}{\left\| Y_{k-1} - \hat{X}_{k-1} \right\|^2} \right]_+ \tag{27}$$

$$\mu_k^Y \equiv \left[L_{k-1}^Y + \frac{2\mathrm{tr}[(X_k - \hat{Y}_{k-1})^\top \nabla_Y f(X_k, \hat{Y}_{k-1})]}{\left\| X_k - \hat{Y}_{k-1} \right\|^2} \right]_+ . \tag{28}$$

The following lemma shows that $\{\mu_k^X\}$ and $\{\mu_k^Y\}$ converge.

Lemma 1: The sequences $\{\mu_k^X\}$ in (27) and $\{\mu_k^Y\}$ in (28) converge.

In the sequel, we denote the limits by

$$\mu^{\bar{X}} \equiv \lim_{k\to\infty} \mu_k^X, \quad \mu^{\bar{Y}} \equiv \lim_{k\to\infty} \mu_k^Y \tag{29}$$

respectively. As convergence implies boundedness, we obtain from the above lemma that the sequences $\{\mu_k^X\}$ and $\{\mu_k^Y\}$ are bounded, and $\min\{\mu_k^X, \mu_k^Y\}$ exists. The following theorem shows that when γ is larger than this minimum, the $\bar{X}$ and $\bar{Y}$ solutions obtained in Algorithm 1 are equal. In other words, the equality constraint in (5) is exactly satisfied.

Theorem 1: If $\gamma > \min\{\mu_k^X, \mu_k^Y\}$, then $\bar{X} = \bar{Y}$.

Instead of using the limits $\mu^{\bar{X}}$ and $\mu^{\bar{Y}}$, the following corollary uses another condition on γ, which is easier to attain in practice.

Corollary 2: If $\gamma > \max_k\{\min\{\mu_k^X, \mu_k^Y\}\}$, then $\bar{X} = \bar{Y}$.

Without loss of generality, we assume that $f(X, Y) = f(Y, X)$. The condition on γ can then be further simplified.

Theorem 2: Assume that $f(X, Y) = f(Y, X)$. If $\gamma > (1/4)\min\{L^X, L^Y\}$, then $\bar{X} = \bar{Y}$.

Remark 2: Although these corollaries suggest that γ cannot be too small, note that neither can γ be too large. Otherwise, as can be seen from Proposition 2, the stepsize will be very small.

As F is biconvex (and thus multiconvex), stronger convergence results can be obtained when F satisfies the so-called Kurdyka–Łojasiewicz inequality [19]. Specifically, Algorithm 1 can converge to a stationary point with a finite, linear, or sublinear local convergence rate. Interested readers are referred to [19] for details.

C. Setting of the Penalty Parameter

The proposed formulation requires setting the penalty parameter γ in (11). As shown in Theorem 1, when γ is sufficiently large, $\bar{X} = \bar{Y}$ and is also equal to the solution in (5). However, the limits $\mu^{\bar{X}}$ and $\mu^{\bar{Y}}$ [defined in (29)] in Theorem 1 are hard to determine in advance.

In this section, we propose two heuristics to set a good γ. The first heuristic is inspired from Theorem 1 and varies γ in each iteration as

$$\text{(heuristic 1)} \quad \gamma_k = \mu_k^X + \varepsilon \tag{30}$$

where $\varepsilon > 0$ is a small constant. When $k \to \infty$, we expect that the $\{\mu_k^X\}$ sequence converges, and then $\gamma \equiv \lim_{k\to\infty} \gamma_k = \mu^{\bar{X}} + \varepsilon$. Thus, $\gamma > \min\{\mu^{\bar{X}}, \mu^{\bar{Y}}\}$. The second heuristic follows from Corollary 2 and varies γ as:

$$\text{(heuristic 2)} \quad \gamma_k = \max\{\mu_1^X, \mu_2^X, \ldots, \mu_k^X\}. \tag{31}$$

Using the two heuristics to set γ, a practical version of biconvex surrogate (BiS) is shown in Algorithm 2.

Algorithm 2 Practical BiS Minimization for (11).

Initialization: $X_{-1} = X_0$, $Y_{-1} = Y_0$.
for $k = 1, 2, \ldots$ **do**
  obtain $\hat{X}_{k-1}$ from X_{k-1} and X_{k-2} using (15);
  obtain $\hat{Y}_{k-1}$ from Y_{k-1} and Y_{k-2} using (16);
  set γ as (30) or (31);
  obtain X_k from $\hat{X}_{k-1}$ and Y_{k-1} using (23);
  obtain Y_k from $\hat{Y}_{k-1}$ and X_k using (25);
  if converged **then**
    return $Z = Y_k Y_k^\top$ (or $X_k Y_k^\top$).
  end if
end for

The effectiveness of these two heuristics will be empirically studied in Section IV-A1.

IV. Experiments

In this section, we perform experiments on three low-rank SDP applications in machine learning, namely, NPKL (Section IV-A), matrix completion (Section IV-B), and metric learning (Section IV-C).

The following algorithms will be compared.

1) The proposed BiS (Algorithm 2) with $\delta_\omega = 0.99$ in (17) and (18).
2) A variant of Algorithm 2 but without extrapolation (i.e., $\omega_{k-1}^X = \omega_{k-1}^Y = 0$). As discussed in Remark 1, this is the same as performing alternating GD on $F(X, Y)$ in (11).
3) *FW*: The accelerated FW algorithm [12].

TABLE I
ADULT DATA SETS USED IN THE NPKL EXPERIMENT. EACH DATA SAMPLE HAS 123 FEATURES

	a1a	a2a	a3a	a4a	a5a	a6a	a7a	a8a	a9a
number of samples	1,605	2,265	3,185	4,781	6,414	11,220	16,100	22,696	32,561
number of must-link pairs	963	1,359	1,911	2,869	3,848	6,732	9,660	13,618	19,537
number of cannot-link pairs	963	1,359	1,911	2,869	3,848	6,732	9,660	13,618	19,537

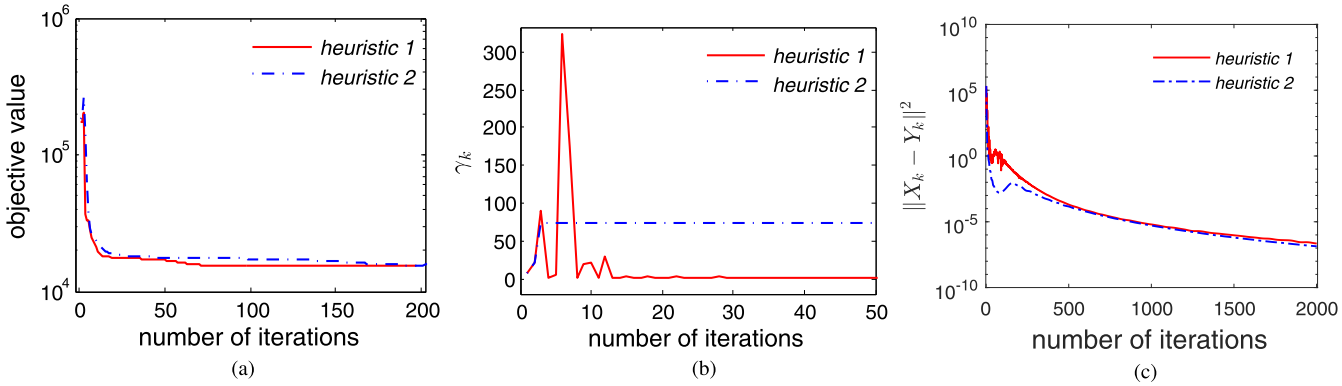

Fig. 1. Convergence of the BiS algorithm with different settings of γ_k [heuristic 1: γ_k in (30) and heuristic 2: γ_k in (31)]. (a) Objective value. (b) γ_k. (c) Residual value $||X_k - Y_k||^2$.

4) *GD:* This transforms the SDP in (1) to the NCP in (3), and then optimizes using GD, with stepsize determined based on the line search as in [33].
5) *Accelerated GD (AGD):* This is similar to GD, except that Nesterov's AGD [34], [35] is used.
6) *ADMM*: This transforms the SDP in (1) to the formulation in (5), and then optimize using ADMM.

All these are implemented in MATLAB. The FW algorithm is stopped when the relative change of objective values in successive iterations is smaller than 10^{-4}. For the other algorithms, they are stopped when the objective value is smaller than the final objective value obtained by FW (or when the number of iterations reaches 2000). As for the rank of X or Y, we follow [17] and [37] and set its value to be the largest r satisfying $r(r+1) \leq m$, where m is the total number of observed data [i.e., m is the number of must-link and cannot-link pairs in Section IV-A, the number of observed entries in Section IV-B, and the number of triplets (similar–dissimilar patterns) in Section IV-C, respectively]. In BiS and its variant without extrapolation, the Lipschitz constants L_k^X and L_k^Y are obtained by line search (in particular, exact line search as the objective is quadratic for the problems considered here). Experiments are performed on a PC with a 3.07-GHz CPU and 24-GB RAM.

A. Nonparametric Kernel Learning

We consider the NPKL formulation in (4). As in [36], the learned kernel matrix is then used for kernelized k-means clustering (with the number of clusters k equal to the number of classes).

Clustering performance is measured by the Rand index $(a + b/0.5\ n(n-1))$ [37], where a is the number of pairs belonging to the same class and are placed in the same cluster by k-means and b is the number of sample pairs belonging to different classes and are placed in different clusters. The denominator $0.5\ n(n-1)$ is the total number of sample pairs. The higher the Rand index, the better the performance.

Experiments are performed on the adult data sets[1] (Table I) that have been commonly used for benchmarking NPKL algorithms. We randomly sample $0.6n$ must-link and $0.6n$ cannot-link pairs. The experiment is repeated 20 times with random restarts for k-means.

1) Setting of γ_k: In this section, we first investigate the effectiveness of the two heuristics to set γ_k (Section III-C). Experiments are performed on the a9a data set. As shown in Fig. 1(a), the algorithm converges in both γ_k settings, but heuristic 1 has faster convergence. This is due to that the γ_k value obtained by heuristic 2 is nondecreasing and tends to be much larger than that by heuristic 1 [Fig. 1(b)]. As discussed in Remark 2, a larger γ_k leads to smaller stepsize and slower convergence. Hence, we will use heuristic 1 in the sequel. Fig. 1(c) shows the progress of $\|X_k - Y_k\|^2$ with iterations. Again, it converges to zero under both γ_k settings.

2) Clustering Performance: Table II shows the Rand indices obtained on all the adult data sets. As can be seen, all algorithms (except ADMM) have comparable clustering performance.

3) Speed: For GD, AGD, BiS, and its variant without extrapolation, the most expensive operation is on estimating the local Lipschitz constants L_{k-1}^X and L_{k-1}^Y. This involves computing the objective and its gradient. Note that when Z in (4) is decomposed as $XY^\top$, the objective in (4) is quadratic with respect to both X and Y (so are the objectives (33) and (35) for the problems considered in Sections IV-B and IV-C,

[1]Downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

TABLE II

RAND INDICES (%) ON THE ADULT DATA SETS. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRED t-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED

	FW	*GD*	*AGD*	*ADMM*	*BiS (w/o extrapolation)*	*BiS*
a1a	96.55 ± 0.70	96.55 ± 0.69	**96.56 ± 0.69**	96.25 ± 0.81	96.51 ± 0.72	**96.56 ± 0.73**
a2a	**95.69 ± 0.54**	91.03 ± 14.43	95.63 ± 0.47	95.31 ± 0.45	95.61 ± 0.48	95.61 ± 0.56
a3a	94.12 ± 0.84	94.04 ± 0.71	94.14 ± 0.75	93.55 ± 0.88	94.19 ± 0.69	**94.22 ± 0.73**
a4a	**94.09 ± 0.69**	93.98 ± 0.57	94.00 ± 0.50	93.29 ± 0.59	94.08 ± 0.53	93.98 ± 0.48
a5a	**94.27 ± 0.32**	94.23 ± 0.31	94.16 ± 0.34	93.58 ± 0.19	94.16 ± 0.34	94.21 ± 0.34
a6a	**94.53 ± 0.33**	94.44 ± 0.38	94.49 ± 0.37	93.80 ± 0.37	94.46 ± 0.37	94.41 ± 0.35
a7a	94.54 ± 0.29	94.53 ± 0.33	94.54 ± 0.30	93.88 ± 0.35	94.48 ± 0.33	**94.56 ± 0.32**
a8a	**94.58 ± 0.14**	94.49 ± 0.11	94.48 ± 0.13	93.87 ± 0.15	94.44 ± 0.11	94.47 ± 0.18
a9a	**94.68 ± 0.18**	94.54 ± 0.15	94.61 ± 0.18	94.04 ± 0.18	94.57 ± 0.18	94.62 ± 0.12

TABLE III

CPU TIME (IN SECONDS) ON THE ADULT DATA SETS. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRED t-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED

	FW	*GD*	*AGD*	*ADMM*	*BiS (w/o extrapolation)*	*BiS*
a1a	16.2 ± 0.29	35.8 ± 0.63	7.8 ± 0.25	142.8 ± 0.95	24.3 ± 0.67	**3.5 ± 0.15**
a2a	26.2 ± 0.50	57.5 ± 1.39	11.3 ± 0.35	199.8 ± 1.13	30.8 ± 1.00	**4.7 ± 0.14**
a3a	47.9 ± 0.78	87.8 ± 2.48	13.1 ± 0.26	279.7 ± 2.39	50.1 ± 1.30	**5.9 ± 0.12**
a4a	67.4 ± 1.21	121.1 ± 2.72	17.4 ± 0.25	415.9 ± 5.88	67.6 ± 2.30	**9.1 ± 0.23**
a5a	87.1 ± 1.46	159.2 ± 2.90	21.7 ± 0.31	435.0 ± 6.78	99.9 ± 3.78	**11.9 ± 0.33**
a6a	182.0 ± 2.80	263.1 ± 4.67	36.6 ± 0.57	723.6 ± 13.93	218.4 ± 5.81	**28.9 ± 0.53**
a7a	233.2 ± 4.96	396.0 ± 8.54	56.5 ± 0.86	1575.8 ± 87.24	259.0 ± 4.81	**35.6 ± 0.67**
a8a	327.6 ± 9.65	536.0 ± 17.09	81.1 ± 2.47	3409.5 ± 201.07	412.9 ± 17.84	**52.2 ± 1.56**
a9a	473.0 ± 7.25	765.2 ± 17.73	114.1 ± 2.25	5769.6 ± 463.8	546.7 ± 12.46	**71.8 ± 1.24**

respectively). Computing L_{k-1}^X and L_{k-1}^Y then takes $\mathcal{O}(nr^2)$ time. In comparison, the per-iter complexity for the ADMM procedure in [16] is $O(rn + \sum_{i=1}^n (|\mathcal{T}_i|^3 + r^2 + r|\mathcal{T}_i|)) = O(nr + nr^2 + \sum_{i=1}^n (|\mathcal{T}_i|^3 + r|\mathcal{T}_i|))$. Hence, the ADMM iteration is more expensive than those of BiS and its variant.

Table III shows the CPU time used by various algorithms. As can be seen, BiS is consistently the fastest. Fig. 2 shows convergence of the training objective in (4) on the (largest) a9a data set. Plots for the other data sets are similar and thus not shown. Clearly, BiS converges much faster than the others. In particular, note that the convergence of ADMM is fast in terms of the number of iterations, but slow when measured with respect to time, as its per-iteration time complexity is much higher.

B. Matrix Completion

In this section, we consider the matrix completion problem in collaborative filtering. Given a partially observed matrix $\tilde{T}$ (with the observed entries indexed by the set Ω), we try to find a low-rank matrix $U \in \mathbb{R}^{n\times u}$ that well approximates the observed entries. This can formulated as the following optimization problem [38]:

$$\min_U \|U\|_* + \frac{\lambda}{2} \sum_{(i,j)\in\Omega} (U_{ij} - \tilde{T}_{ij})^2 \tag{32}$$

where $\|\cdot\|_*$ is the nuclear norm and λ is a tradeoff parameter. Equivalently, (32) can be rewritten as the SDP [12]

$$\min_{Z\succeq 0} \operatorname{tr}(Z) + \frac{\lambda}{2} \sum_{(i,j)\in\Omega} (Z_{ij} - T_{ij})^2 \ : \ \operatorname{rank}(Z) \le k \tag{33}$$

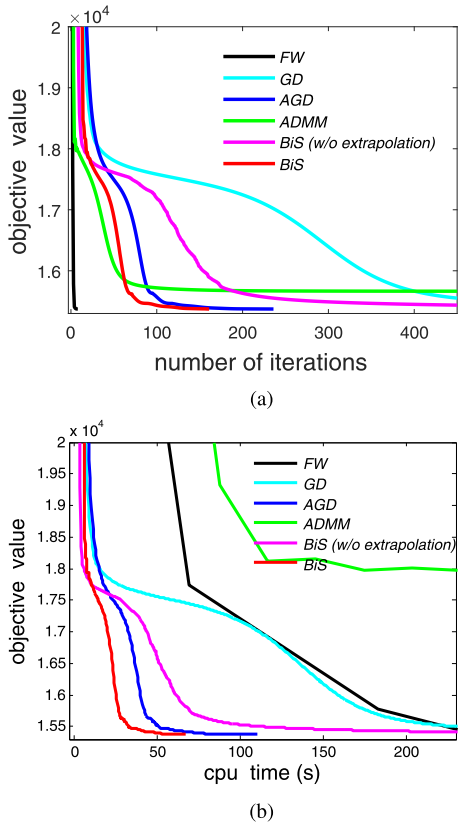

Fig. 2. Convergence of the objective with respect to number of iterations (top) and time (bottom) on the a9a data set. (a) Objective value versus iterations. (b) Objective value versus CPU time.

where $T = \begin{bmatrix} 0 & \tilde{T} \\ \tilde{T}^\top & 0 \end{bmatrix}$, $Z = \begin{bmatrix} V & U \\ U^\top & W \end{bmatrix}$, and V and W are some symmetric matrices.

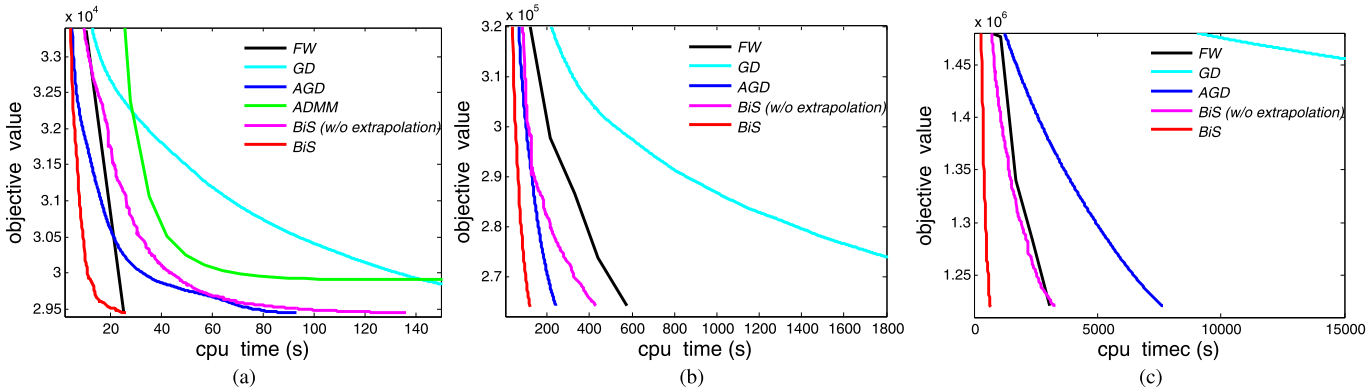

Fig. 3. Objective versus CPU time on the MovieLens data sets. (a) MovieLens-100K. (b) MovieLens-1M. (c) MovieLens-10M.

TABLE IV
MOVIELENS DATA SETS

	#users	#movies	#ratings for training	#ratings for testing
100K	943	1,682	79,993	20,007
1M	6,040	3,706	799,522	200,192
10M	69,878	10,677	7,999,549	2,000,505

TABLE V
TESTING RMSE ON THE MOVIELENS DATA SETS. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRED T-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED. NOTE THAT ADMM CANNOT BE RUN ON THE LARGER 1M AND 10M DATA SETS BECAUSE IT IS TOO SLOW

	100K	1M	10M
FW	**0.935±0.007**	0.872±0.001	**0.818±0.000**
GD	0.940±0.007	0.884±0.003	0.874±0.002
AGD	0.942±0.009	0.880±0.530	0.824±0.001
ADMM	0.987±.0066	-	-
BiS (w/o extrapolation)	0.940±0.009	0.874±0.004	0.819±0.001
BiS	0.936±0.006	**0.868±0.003**	**0.818±0.001**

Following [12], we perform experiments on the MovieLens data sets[2] (Table IV), which have been commonly used for evaluating recommender systems. They contain ratings $\{1, 2, \ldots, 5\}$ assigned by various users on movies. For each user, 80% of the observed ratings are used for training, and the rest for testing. This is repeated ten times and the average performance is reported.

Table V shows the root-mean-squared error (RMSE) on the test set. As can be seen, the proposed BiS is the most accurate on the larger 1 M and 10 M data sets, and very competitive on the smallest 100 K data set. Table VI shows the CPU time, and Fig. 3 shows convergence of the training objective versus CPU time. As can be seen, BiS converges much faster than the other methods.

C. Metric Learning

In this section, we consider the metric learning problem. Let the (squared) Mahalanobis distance between patterns x_i and x_j be $d_M^2(x_i, x_j) = (x_i - x_j)^\top M(x_i - x_j)$, where M is a PSD matrix. A good M should ensure that the distance between a pair of similar patterns (specified in the set S) be smaller than

TABLE VI
CPU TIME (IN SECONDS) ON THE MOVIELENS DATA SETS. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRED T-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED. NOTE THAT ADMM CANNOT BE RUN ON THE LARGER 1M AND 10M DATA SETS

	100K	1M	10M
FW	**24.3±2.9**	563.6±40.9	3427±434
GD	233.9±14.3	3205.8±437.1	19277±545
AGD	53.5±22.9	294.4±55.0	9574±1549
ADMM	2717.1±145.6	-	-
BiS (w/o extrapolation)	138.6±65.2	424.1±79.0	4310±694
BiS	31.5±19.9	**111.2±19.5**	**786±104**

the distance between a pair of dissimilar patterns (contained in the set D). Given a triplet $\tau = (i, j, k)$ where $(x_i, x_j) \in \mathcal{S}$ and $(x_i, x_k) \in \mathcal{D}$, this can be formulated as [3]

$$1 + d_M^2(x_i, x_j) \le d_M^2(x_i, x_k)$$

or, equivalently, $1 + \mathrm{tr}(MX_\tau) \le 0$, where $X_\tau = (x_i - x_j)(x_i - x_j)^\top - (x_i - x_k)(x_i - x_k)^\top$. Let $\mathcal{T}$ be the set of all these triplets. We then have the following optimization problem:

$$\min_{M \succeq 0} \mathrm{tr}(M) + \lambda \sum_{\tau \in \mathcal{T}} [(1 + \mathrm{tr}(MX_\tau)]_+ \tag{34}$$

where $\mathrm{tr}(M)$ (equal to the nuclear norm of the PSD matrix M) is a low-rank regularizer and λ is a tradeoff parameter. The objective in (34) is nonsmooth. This can be simplified to the following smooth optimization problem by using Nesterov's smoothing technique [39]:

$$\min_{M \succeq 0} \mathrm{tr}(M) + \lambda g_\mu(M) \tag{35}$$

where $g_\mu(M) = \max_{0 \le v_\tau \le 1} \sum_{\tau \in \mathcal{T}} v_\tau (1 + \mathrm{tr}(X_\tau^\top M)) - (\mu/2) v_\tau^2$ is Lipschitz-smooth and μ is a smoothing parameter. This can then be solved by accelerated proximal gradient (APG) algorithm [29]. However, each APG iteration requires a projection onto the PSD cone, which can be expensive.

Experiment is performed on the ORL data set.[3] It contains face images from 40 subjects. Each subject has ten images, each of size 112×92 (thus, $112 \times 92 = 10304$-D). We use half of the image set to learn a metric M, and then classify the remaining images by a three-nearest neighbor classifier with the learned metric. We do not compare with

[2]http://grouplens.org/data sets/movielens/

[3]http://www.face-rec.org/databases/

TABLE VII

CLASSIFICATION ACCURACIES (%) USING THE THREE-NEAREST NEIGHBOR CLASSIFIER ON THE ORL DATA SET. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRED t-TEST WITH 95% CONFIDENCE) ARE HIGHLIGHTED

GD	95.36 ± 4.20	*AGD*	95.82 ± 3.73
BiS (w/o extrapolation)	95.25 ± 2.35	*BiS*	**96.19 ± 5.20**
ADMM	75.06 ± 6.42		

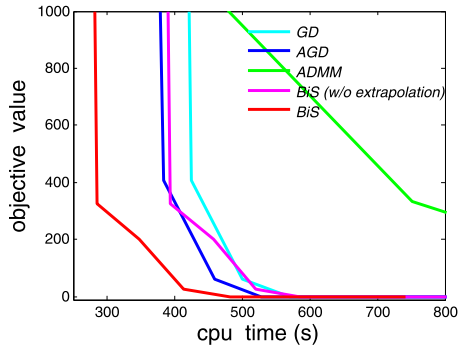

Fig. 4. Objective versus CPU time on the ORL data set.

FW, as extracting the leading singular vectors from the dense 10304×10304 matrix is computationally expensive. The experiment is repeated ten times and the average performance is reported.

Table VII shows the testing accuracies obtained with the learned metric M. Fig. 4 shows the convergence of the training objective with time. Again, the proposed BiS is both accurate and fast.

V. CONCLUSION

In this paper, we proposed an efficient algorithm for learning low-rank PSD matrices. With the addition of a Courant penalty, we reformulated this as a biconvex optimization problem. After linearization, the surrogate objective is simple and can be easily optimized using BCD. Although traditionally this penalty-based formulation approaches the original optimization problem only when the penalty parameter is infinite, we showed that the Courant penalty reduces to zero when the penalty is sufficiently large (but finite), and the solution obtained from the biconvex problem also solves the original problem. This reformulation opens a door to solve many similar low-rank matrix learning problems that also involve an unknown $XX^\top$ and allows us to borrow the multiconvex optimization tool in [19]. Experiments on a number of low-rank PSD matrix learning problems in machine learning, namely, NPKL, matrix completion, and metric learning, show that the proposed algorithm is accurate and faster than existing algorithms.

APPENDIX A
PROOFS

A. Proof of Lemma 1

Proof: The sequence $\{(X_k, Y_k)\}$ converges to $(\bar{X}, \bar{Y})$ by Proposition 3. Since $\lim_{k\to\infty} \hat{X}_k = \lim_{k\to\infty} X_k = \bar{X}$ and $\lim_{k\to\infty} \hat{Y}_k = \lim_{k\to\infty} Y_k = \bar{Y}$, we have

$$\begin{aligned}
&\lim_{k\to\infty} \mu_k^X \\
&= \left[\lim_{k\to\infty} L_{k-1}^X \right. \\
&\qquad \left. + \lim_{k\to\infty} \frac{2\mathrm{tr}[(Y_{k-1} - \hat{X}_{k-1})^\top \nabla_X f(\hat{X}_{k-1}, Y_{k-1})]}{\|Y_{k-1} - \hat{X}_{k-1}\|^2}\right]_+ \\
&= \left[L^{\bar{X}} + \frac{2\mathrm{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})]}{\|\bar{Y} - \bar{X}\|^2}\right]_+
\end{aligned}$$

where $L^{\bar{X}}$ is a nonnegative constant satisfying

$$f(X, \bar{Y}) \le f(\bar{X}, \bar{Y}) + \langle \nabla_X f(\bar{X}, \bar{Y}), X - \bar{X}\rangle + \frac{L^{\bar{X}}}{2}\|X - \bar{X}\|^2.$$

Similarly

$$\lim_{k\to\infty} \mu_k^Y = \left[L^{\bar{Y}} + \frac{2\mathrm{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})]}{\|\bar{X} - \bar{Y}\|^2}\right]_+$$

where $L^{\bar{Y}}$ is a nonnegative constant satisfying

$$f(\bar{X}, Y) \le f(\bar{X}, \bar{Y}) + \langle \nabla_Y f(\bar{X}, \bar{Y}), Y - \bar{Y}\rangle + \frac{L^{\bar{Y}}}{2}\|Y - \bar{Y}\|^2.$$

■

B. Proof of Proposition 1

Proof: Let $\alpha_1, \alpha_2 \ge 0$ such that $\alpha_1 + \alpha_2 = 1$, $X, Y_1, Y_2 \in \mathbb{R}^{n\times r}$. We have

$$\begin{aligned}
f(X, \alpha_1 Y_1 + \alpha_2 Y_2) &= \tilde{f}(X(\alpha_1 Y_1 + \alpha_2 Y_2)^\top) \\
&= \tilde{f}\left(\alpha_1 X Y_1^\top + \alpha_2 X Y_2^\top\right) \\
&\le \alpha_1 \tilde{f}\left(X Y_1^\top\right) + \alpha_2 \tilde{f}\left(X Y_2^\top\right) \\
&= \alpha_1 f(X, Y_1) + \alpha_2 f(X, Y_2)
\end{aligned}$$

where the inequality uses convexity of $\tilde{f}(Z)$. Hence, f is convex with respect to Y. Similarily, f is convex with respect to X, and so f is biconvex. As $\gamma/2\|X - Y\|^2$ in (11) is biconvex, F is biconvex. ■

C. Proof of Corollary 1

Proof: From (13) , we have

$$\begin{aligned}
X_k &= \arg\min_X Q_X(X; \hat{X}_{k-1}, Y_{k-1}) \\
&= \arg\min_X \langle \nabla_X F(\hat{X}_{k-1}, Y_{k-1}), X - \hat{X}_{k-1}\rangle \\
&\quad + \frac{L_{k-1}^X + \gamma}{2}\|X - \hat{X}_{k-1}\|^2 \\
&= \frac{1}{L_{k-1}^X + \gamma}\left[L_{k-1}^X \hat{X}_{k-1} - \nabla_X f(\hat{X}_{k-1}, Y_{k-1}) + \gamma Y_{k-1}\right] \\
&= \frac{L_{k-1}^X}{L_{k-1}^X + \gamma}\left[\hat{X}_{k-1} - \frac{1}{L_{k-1}^X}\nabla_X f(\hat{X}_{k-1}, Y_{k-1})\right] \\
&\quad + \frac{\gamma}{L_{k-1}^X + \gamma} Y_{k-1} \\
&= \alpha \hat{X}_{k-1}^{gd} + (1 - \alpha) Y_{k-1}. \qquad (36)
\end{aligned}$$

Similarily, from (14), we have

$$\begin{aligned} Y_k &= \arg\min_Y Q_Y(Y; X_k, \hat{Y}_{k-1}) \\ &= \arg\min_Y \langle \nabla_Y F(X_k, \hat{Y}_{k-1}), Y - \hat{Y}_{k-1}\rangle \\ &\quad + \frac{L_{k-1}^Y + \gamma}{2}\|Y - \hat{Y}_{k-1}\|^2 \\ &= \frac{1}{L_{k-1}^Y + \gamma}\left[L_{k-1}^Y \hat{Y}_{k-1} - \nabla_Y f(X_k, \hat{Y}_{k-1}) + \gamma X_k\right] \\ &= \frac{L_{k-1}^Y}{L_{k-1}^Y + \gamma}\left[\hat{Y}_{k-1} - \frac{1}{L_{k-1}^Y}\nabla_Y f(X_k, \hat{Y}_{k-1})\right] \\ &\quad + \frac{\gamma}{L_{k-1}^Y + \gamma} X_k \\ &= \beta \hat{Y}_{k-1}^{gd} + (1-\beta) X_k. \end{aligned} \tag{37}$$

■

D. Proof of Theorem 1

Proof: First, we show that $\bar{X} = \bar{Y}$ if $\gamma \geq \mu^{\bar{Y}}$. The following lemma holds immediately by using (15) and (16).

Lemma 2: When Algorithm 2 converges

$$\begin{aligned} \lim_{k\to\infty} \hat{X}_{k-1} &= \lim_{k\to\infty}\left[X_{k-1} + \omega_{k-1}^X (X_{k-1} - X_{k-2})\right] \\ &= \lim_{k\to\infty} X_{k-1} = \lim_{k\to\infty} X_k \\ &= \bar{X} \end{aligned}$$

and similarly

$$\lim_{k\to\infty} \hat{Y}_{k-1} = \lim_{k\to\infty} Y_{k-1} = \lim_{k\to\infty} Y_k = \bar{Y}.$$

Lemma 3: If $\gamma > \mu_k^Y$, then $\|\hat{Y}_{k-1} - Y_k\|^2 > \|X_k - Y_k\|^2$.

Proof: Using (37)

$$\begin{aligned} &\hat{Y}_{k-1} - Y_k \\ &= \hat{Y}_{k-1} - \frac{1}{L_{k-1}^Y + \gamma}\left[L_{k-1}^Y \hat{Y}_{k-1} - \nabla_Y f(X_k, \hat{Y}_{k-1}) + \gamma X_k\right] \\ &= \frac{1}{L_{k-1}^Y + \gamma}[\gamma(\hat{Y}_{k-1} - X_k) + \nabla_Y f(X_k, \hat{Y}_{k-1})]. \end{aligned}$$

Similarly, from (37)

$$\begin{aligned} &X_k - Y_k \\ &= X_k - \frac{1}{L_{k-1}^Y + \gamma}\left[L_{k-1}^Y \hat{Y}_{k-1} - \nabla_Y f(X_k, \hat{Y}_{k-1}) + \gamma X_k\right] \\ &= \frac{-1}{L_{k-1}^Y + \gamma}\left[L_{k-1}^Y (\hat{Y}_{k-1} - X_k) - \nabla_Y f(X_k, \hat{Y}_{k-1})\right]. \end{aligned}$$

Let $A = \hat{Y}_{k-1} - X_k$ and $B = \nabla_Y f(X_k, \hat{Y}_{k-1})$. Then

$$\begin{aligned} &\|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2 \\ &= \frac{1}{(L_{k-1}^Y + \gamma)^2}\left[\|\gamma A + B\|^2 - \|L_{k-1}^Y A - B\|^2\right] \\ &= \frac{1}{L_{k-1}^Y + \gamma}\left[(\gamma - L_{k-1}^Y)\mathrm{tr}(A^\top A) + 2\mathrm{tr}(A^\top B)\right]. \end{aligned}$$

Hence, from (28)

$$\begin{aligned} &\gamma > \mu_k^Y \\ &\Leftrightarrow \gamma > L_{k-1}^Y - 2\frac{\mathrm{tr}(A^\top B)}{\mathrm{tr}(A^\top A)} \\ &\Leftrightarrow (\gamma - L_{k-1}^Y)\mathrm{tr}(A^\top A) + 2\mathrm{tr}(A^\top B) > 0 \\ &\Leftrightarrow \frac{1}{L_{k-1}^Y + \gamma}\left[(\gamma - L_{k-1}^Y)\mathrm{tr}(A^\top A) + 2\mathrm{tr}(A^\top B)\right] > 0 \\ &\Leftrightarrow \|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2 > 0. \end{aligned}$$

■

Lemma 4: If $\gamma > \mu_k^Y$, then $\bar{X} = \bar{Y}$.

Proof: From Lemma 3, we have $\|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2 > 0$, that is,

$$(\gamma - \mu_k^Y)(\|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2) > 0.$$

By the sign-preserving property[4]

$$\lim_{k\to\infty} (\gamma - \mu_k^Y)(\|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2) \geq 0.$$

Hence, from Lemma 2, we have

$$\begin{aligned} &\lim_{k\to\infty} (\gamma - \mu_k^Y)(\|\hat{Y}_{k-1} - Y_k\|^2 - \|X_k - Y_k\|^2) \geq 0 \\ &\Leftrightarrow (\gamma - \mu^{\bar{Y}})(\|\bar{Y} - \bar{Y}\|^2 - \|\bar{X} - \bar{Y}\|^2) \geq 0 \\ &\Leftrightarrow -(\gamma - \mu^{\bar{Y}})(\|\bar{X} - \bar{Y}\|^2) \geq 0 \end{aligned}$$

and so $\bar{X} = \bar{Y}$ if $\gamma > \mu^{\bar{Y}}$ (by the sign-preserving property of $\gamma > \mu_k^Y$). ■

Next, we show that $\bar{X} = \bar{Y}$ if $\gamma > \mu^{\bar{X}}$.

Lemma 5: If $\gamma > \mu_k^X$, then $\|\hat{X}_{k-1} - X_k\|^2 > \|X_k - Y_{k-1}\|^2$.

Proof: From (36)

$$X_k = \frac{1}{L_{k-1}^X + \gamma}\left[L_{k-1}^X \hat{X}_{k-1} - \nabla_X f(\hat{X}_{k-1}, Y_{k-1}) + \gamma Y_{k-1}\right].$$

We have

$$\begin{aligned} &\hat{X}_{k-1} - X_k \\ &= \hat{X}_{k-1} - \frac{1}{L_{k-1}^X + \gamma}\left(L_{k-1}^X \hat{X}_{k-1} - \nabla_X f(\hat{X}_{k-1}, Y_{k-1}) + \gamma Y_{k-1}\right) \\ &= \frac{1}{L_{k-1}^X + \gamma}[\gamma(\hat{X}_{k-1} - Y_{k-1}) + \nabla_X f(\hat{X}_{k-1}, Y_{k-1})] \end{aligned}$$

and

$$\begin{aligned} &Y_{k-1} - X_k \\ &= Y_{k-1} - \frac{1}{L_{k-1}^X + \gamma}\left(L_{k-1}^X \hat{X}_{k-1} - \nabla_X f(\hat{X}_{k-1}, Y_{k-1}) + \gamma Y_{k-1}\right) \\ &= \frac{-1}{L_{k-1}^X + \gamma}\left[L_{k-1}^X (\hat{X}_{k-1} - Y_{k-1}) - \nabla_X f(\hat{X}_{k-1}, Y_{k-1})\right]. \end{aligned}$$

Let $A = \hat{X}_{k-1} - Y_{k-1}$ and $B = \nabla_X f(\hat{X}_{k-1}, Y_{k-1})$. Then

$$\begin{aligned} &\|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2 \\ &\quad = \frac{1}{(L_{k-1}^X + \gamma)^2}\left[\|\gamma A + B\|^2 - \|L_{k-1}^X A - B\|^2\right] \\ &\quad = \frac{1}{L_{k-1}^X + \gamma}\left[(\gamma - L_{k-1}^X)\mathrm{tr}(A^\top A) + 2\mathrm{tr}(A^\top B)\right]. \end{aligned}$$

[4]In other words, if $\lim_{k\to\infty} a_k = a$ and $a_k \geq 0$ for all k, then $a \geq 0$.

Hence, from (27)

$$\begin{aligned}
\gamma &> \mu_k^X \\
&\Leftrightarrow \gamma > L_{k-1}^X - 2\frac{\text{tr}(A^\top B)}{\text{tr}(A^\top A)} \\
&\Leftrightarrow \big(\gamma - L_{k-1}^X\big)\text{tr}(A^\top A) + 2\text{tr}(A^\top B) > 0 \\
&\Leftrightarrow \frac{1}{L_{k-1}^X + \gamma}\big[\big(\gamma - L_{k-1}^X\big)\text{tr}(A^\top A) + 2\text{tr}(A^\top B)\big] > 0 \\
&\Leftrightarrow \|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2 > 0.
\end{aligned}$$

■

Lemma 6: If $\gamma > \mu_k^X$, then $\bar{X} = \bar{Y}$.

Proof: From Lemma 5, when $\gamma > \mu_k^X$, $\|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2 > 0$, that is,

$$(\gamma - \mu_k^X)(\|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2) > 0.$$

By the sign-preserving property, we have

$$\lim_{k\to\infty} \big(\gamma - \mu_k^X\big)(\|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2) \geq 0.$$

Hence, from Lemma 2, we have

$$\begin{aligned}
&\lim_{k\to\infty} \big(\gamma - \mu_k^X\big)(\|\hat{X}_{k-1} - X_k\|^2 - \|X_k - Y_{k-1}\|^2) \geq 0 \\
&\quad\Leftrightarrow (\gamma - \mu^{\bar{X}})(\|\bar{X} - \bar{X}\|^2 - \|\bar{X} - \bar{Y}\|^2) \geq 0 \\
&\quad\Leftrightarrow -(\gamma - \mu^{\bar{X}})(\|\bar{X} - \bar{Y}\|^2) \geq 0
\end{aligned}$$

so $\bar{X} = \bar{Y}$ if $\gamma > \mu^{\bar{X}}$ (by the sign-preserving property of $\gamma > \mu_k^X$). ■

Result follows on combining Lemmas 4 and 6. ■

Appendix B
Proof of Corollary 2

Proof: By combining Lemmas 4 and 6, result follows on using $\gamma > \max_k\{\min\{\mu_k^X, \mu_k^Y\}\} > \min\{\mu_k^X, \mu_k^Y\}$. ■

Appendix C
Proof of Theorem 2

Proof: $f(X, Y) = f(Y, X)$ implies

$$\nabla_X f(X, Y) = \nabla_X f(Y, X) \tag{38}$$

$$\nabla_Y f(X, Y) = \nabla_Y f(Y, X). \tag{39}$$

First, we introduce some lemmas.

Lemma 7: If $(\bar{X}, \bar{Y})$ is a stationary point of (11), then

$$\nabla_X f(\bar{X}, \bar{Y}) = -\nabla_Y f(\bar{X}, \bar{Y}). \tag{40}$$

Proof: Since $(\bar{X}, \bar{Y})$ is a stationary point of (11), $\nabla_X F(\bar{X}, \bar{Y}) = 0$, $\nabla_Y F(\bar{X}, \bar{Y}) = 0$, and so

$$\nabla_X f(\bar{X}, \bar{Y}) = \gamma\,(\bar{Y} - \bar{X}) \tag{41}$$

$$\nabla_Y f(\bar{X}, \bar{Y}) = \gamma\,(\bar{X} - \bar{Y}). \tag{42}$$

Hence, $\nabla_X f(\bar{X}, \bar{Y}) = -\nabla_Y f(\bar{X}, \bar{Y})$, and result follows. ■

Lemma 8: Assume that $f(X, Y) = f(Y, X)$, $(\bar{X}, \bar{Y})$ is a stationary point of (11). Then, $\bar{X} = \bar{Y}$ if $\gamma > (1/4)L^X$.

Proof: By contradiction, assume $\bar{X} \neq \bar{Y}$ when $\gamma > (1/4)L^X$. From (42), we have

$$\gamma\,\|\bar{X} - \bar{Y}\|^2 = \text{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})$$

such that

$$\begin{aligned}
\gamma &= \frac{2\text{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})]}{2\|\bar{X} - \bar{Y}\|^2} \\
&= \frac{\text{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})] + \text{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})]}{2\|\bar{X} - \bar{Y}\|^2} \\
&= \frac{\text{tr}[(\bar{X} - \bar{Y})^\top \nabla_Y f(\bar{X}, \bar{Y})] - \text{tr}[(\bar{X} - \bar{Y})^\top \nabla_X f(\bar{X}, \bar{Y})]}{2\|\bar{X} - \bar{Y}\|^2} \\
&\leq \frac{f(\bar{X}, \bar{X}) - f(\bar{X}, \bar{Y}) - \text{tr}[(\bar{X} - \bar{Y})^\top \nabla_X f(\bar{X}, \bar{Y})]}{2\|\bar{X} - \bar{Y}\|^2} \\
&= \frac{f(\bar{X}, \bar{X}) - f(\bar{Y}, \bar{X}) - \text{tr}[(\bar{X} - \bar{Y})^\top \nabla_X f(\bar{Y}, \bar{X})]}{2\|\bar{X} - \bar{Y}\|^2} \\
&\leq \frac{\frac{1}{2}L^X\|\bar{X} - \bar{Y}\|^2}{2\|\bar{X} - \bar{Y}\|^2} \\
&= \frac{1}{4}L^X.
\end{aligned} \tag{43}$$

The third equality uses Lemma 7, the first inequality is obtained from the first-order condition of the convexity of $f(\bar{X}, Y)$ with respect to Y, the fourth equality uses (38), the last inequality is obtained from the assumption that $f(\cdot, \bar{X})$ is L^X-smooth.

From (43), we have $\gamma \leq (1/4)L^X$, which contradicts with the starting assumption $\gamma > (1/4)L^X$. ■

Lemma 9: Assume that $(\bar{X}, \bar{Y})$ is a stationary point of (11). Then, $\bar{X} = \bar{Y}$ if $\gamma > (1/4)L^Y$.

Proof: By contradiction, assume $\bar{X} \neq \bar{Y}$ when $\gamma > (1/4)L^Y$. From (41), we have

$$\gamma\,\|\bar{Y} - \bar{X}\|^2 = \text{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})$$

such that

$$\begin{aligned}
\gamma &= \frac{2\text{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})]}{2\|\bar{Y} - \bar{X}\|^2} \\
&= \frac{\text{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})] + \text{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})]}{2\|\bar{Y} - \bar{X}\|^2} \\
&= \frac{\text{tr}[(\bar{Y} - \bar{X})^\top \nabla_X f(\bar{X}, \bar{Y})] - \text{tr}[(\bar{Y} - \bar{X})^\top \nabla_Y f(\bar{X}, \bar{Y})]}{2\|\bar{Y} - \bar{X}\|^2} \\
&\leq \frac{f(\bar{Y}, \bar{Y}) - f(\bar{X}, \bar{Y}) - \text{tr}[(\bar{Y} - \bar{X})^\top \nabla_Y f(\bar{X}, \bar{Y})]}{2\|\bar{Y} - \bar{X}\|^2} \\
&= \frac{f(\bar{Y}, \bar{Y}) - f(\bar{Y}, \bar{X}) - \text{tr}[(\bar{Y} - \bar{X})^\top \nabla_Y f(\bar{Y}, \bar{X})]}{2\|\bar{Y} - \bar{X}\|^2} \\
&\leq \frac{\frac{1}{2}L^Y\|\bar{Y} - \bar{X}\|^2}{2\|\bar{Y} - \bar{X}\|^2} \\
&= \frac{1}{4}L^Y.
\end{aligned} \tag{44}$$

The third equality uses Lemma 7, the first inequality is obtained from the first-order condition of the convexity of $f(X, \bar{Y})$ with respect to X, the fourth equality uses (39), the last inequality is obtained from the assumption that $f(\bar{X}, \cdot)$ is L^Y-smooth.

From (44), we have $\gamma \leq (1/4)L^Y$, which contradicts with the starting assumption $\gamma > (1/4)L^Y$. ■

Theorem 2 follows on combining Lemmas 8 and 9. ■

REFERENCES

[1] A. D'Aspremont, E. L. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet, "A direct formulation for sparse PCA using semidefinite programming," *SIAM Rev.*, vol. 49, no. 3, pp. 41–48, Sep. 2007.

[2] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 505–512.

[3] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1473–1480.

[4] G. R. C. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Jan. 2004.

[5] E.-L. Hu, B. Wang, and S.-C. Chen, "BCDNPKL: Scalable non-parametric kernel learning using block coordinate descent," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, Jun. 2011, pp. 209–216.

[6] G. Obozinski, B. Taskar, and M. I. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *Statist. Comput.*, vol. 20, no. 2, pp. 231–252, Apr. 2010.

[7] N. Srebro, J. Rennie, and T. Jaakola, "Maximum-margin matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1329–1336.

[8] Y. Nesterov and A. Nemirovski, *Interior-point Polynomial Algorithms in Convex Programming*. Philadelphia, PA, USA: SIAM, 1994.

[9] Y. Nesterov, "Smoothing technique and its applications in semidefinite optimization," *Math. Program.*, vol. 110, no. 2, pp. 245–259, Jul. 2007.

[10] Q. Yao and J. T. Kwok, "Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jan. 2017, p. 179.

[11] M. Jaggi, "Convex optimization without projection steps," 2011, *arXiv:1108.1170*. [Online]. Available: https://arxiv.org/abs/1108.1170

[12] S. Laue, "A hybrid algorithm for convex semidefinite optimization," in *Proc. 29th Int. Conf. Mach. Learn.*, Jun. 2012, pp. 1–8.

[13] S. Burer and R. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Program.*, vol. 95, no. 2, pp. 329–357, Feb. 2003.

[14] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, "Low-rank optimization on the cone of positive semidefinite matrices," *SIAM J. Optim.*, vol. 20, no. 5, pp. 2327–2351, May 2010.

[15] L. Grippo, L. Palagi, and V. Piccialli, "Necessary and sufficient global optimality conditions for NLP reformulations of linear SDP problems," *J. Global Optim.*, vol. 44, pp. 339–348, Jul. 2009.

[16] E.-L. Hu and J. T. Kwok, "Efficient kernel learning from side information using ADMM," in *Proc. Int. Joint Conf. Artif. Intell.*, Jun. 2013, pp. 306–316.

[17] Z. Li and J. Liu, "Constrained clustering by spectral regularization," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 421–428.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, Jul. 2011.

[19] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

[20] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, Jun. 2015.

[21] C. Scherrer, A. Tewari, M. Halappanavar, and D. Haglin, "Feature clustering for accelerating parallel coordinate descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 28–36.

[22] R. Mazumder, J. H. Friedman, and T. Hastie, "SparseNet: Coordinate descent with nonconvex penalties," *J. Amer. Stat. Assoc.*, vol. 106, no. 495, pp. 1125–1138, 2011.

[23] P. Richtarik and M. Takac, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Math. Program.*, vol. 144, nos. 1–2, pp. 1–38, Apr. 2014.

[24] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 408–415.

[25] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *Proc. 17th Int. Conf. Knowl. Discovery Data Mining*, San Diego, CA, USA, Aug. 2011, pp. 1064–1072.

[26] O. Fercoq and P. Richtárik, "Optimization in high dimensions via accelerated, parallel and proximal coordinate descent," *SIAM Rev.*, vol. 58, no. 4, pp. 739–771, 2016.

[27] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2037–2060, 2013.

[28] Y. Nesterov, *Introductory Lectures Convex Optimization: A Basic Course*. Norwell, MA, USA: Kluwer, 2004.

[29] M. J. D. Powell, "On search directions for minimization algorithms," *Math. Program.*, vol. 4, no. 1, pp. 193–201, Dec. 1973.

[30] Y. Xu, "Block coordinate descent for regularized multi-convex optimization," Ph.D. dissertation, Rice Univ., Houston, TX , USA, 2012.

[31] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer-Verlag, 2006.

[32] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[33] S. Burer and C. Choi, "Computational enhancements in low-rank semidefinite programming," *Optim. Methods Softw.*, vol. 21, no. 3, pp. 493–512, 2006.

[34] S. Ghadimi and G. Lan, "Accelerated gradient methods for nonconvex nonlinear and stochastic programming," *Math. Program.*, vol. 156, nos. 1–2, pp. 59–99, Mar. 2016.

[35] H. Li and Z. Lin, "Accelerated proximal gradient methods for nonconvex programming," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 379–387.

[36] J. Zhuang, I. Tsang, and S. Hoi, "A family of simple non-parametric kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 1313–1347, Apr. 2011.

[37] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Stat. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.

[38] S. Shalev-Shwartz, A. Gonen, and O. Shamir, "Large-scale convex minimization with a low-rank constraint," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 329–336.

[39] Y. Nesterov, "Smooth minimization of nonsmooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, May 2005.

En-Liang Hu received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010.

He was a Research Assistant and then became a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He is currently an Associate Professor with the Department of Mathematics, Yunnan Normal University, Cheng Gong Campus, Kunming, China. His current research interests include machine learning, data mining, and optimization.

James T. Kwok (M'98-SM'07-F'17) received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong.

He is currently a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.

Dr. Kwok served/is serving as a Governing Board Member. He served/is serving as the Vice President for Publications of the Asia Pacific Neural Network Society. He also served/is serving as the Area Chair of major machine learning/AI conferences. He served/is serving as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Neurocomputing*, and *International Journal of Data Science and Analytics*. He was a recipient of the IEEE Outstanding 2004 Paper Award, the Second Class Award in Natural Sciences by the Ministry of Education, and the People's Republic of China, in 2008.