Scalable Semidefinite Programming using Convex Perturbations

Brian Kulis*, Suvrit Sra†*, Stefanie Jegelka†, Inderjit S. Dhillon*
*Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{kulis,suvrit,inderjit}@cs.utexas.edu
†Max Planck Institute for Biological Cybernetics
Spemannstr 38
72076 Tübingen, Germany
{suvrit.sra,stefanie.jegelka}@tuebingen.mpg.de

UTCS Technical Report TR-07-47

September 13, 2007

Abstract

Several important machine learning problems can be modeled and solved via semidefinite programs. Often, researchers invoke off-the-shelf software for the associated optimization, which can be inappropriate for many applications due to computational and storage requirements. In this paper, we introduce the use of convex perturbations for semidefinite programs (SDPs). Using a particular perturbation function, we arrive at an algorithm for SDPs that has several advantages over existing techniques: a) it is simple, requiring only a few lines of MATLAB, b) it is a first-order method which makes it scalable, c) it can easily exploit the structure of a particular SDP to gain efficiency (e.g., when the constraint matrices are low-rank). We demonstrate on several machine learning applications that the proposed algorithm is effective in finding fast approximations to large-scale SDPs.

1 Introduction

There has been a rapid rise in the use of semidefinite programming in the machine learning community over the last few years. Problems such as nonlinear dimensionality reduction [24, 18], learning kernel matrices [12], maximum margin matrix factorization [19], graph clustering [13], and many others have been formulated and solved as semidefinite programs (SDPs). Off-the-shelf software, such as SEDUMI [17] or DSDP [2], are commonly used for solving these problems. These software packages are effective in finding high-accuracy SDP solutions, though it is difficult to specialize them for particular SDPs and the scalability of these methods makes them restrictive for large problems. Furthermore, it is often the case for machine learning applications that high-accuracy solutions are not necessary, especially if the solution of the SDP is only an intermediate goal in the overall machine learning problem (e.g., metric learning for nearest-neighbor classification [23]).

In this paper we introduce convex perturbations for semidefinite programming and characterize their relation to the unperturbed original. In particular, instead of minimizing $\mathrm{Tr}(\mathsf{CX})$ subject to constraints on X, we minimize $\mathrm{Tr}(\mathsf{CX}) - \varepsilon \log\det(\mathsf{X})$. We show that for an appropriate $\overline{\varepsilon} > 0$, solving the perturbed problem for any $\varepsilon \leq \bar{\varepsilon}$ yields a solution to the original problem—in fact, it yields the *maximum determinant* solution. Subsequently we exploit the strict convexity of the perturbed problem to develop a *simple* first-order algorithm based on Bregman projections. Our algorithm scales well with the size of the data and may be preferable to off-the-shelf software, especially for large-scale inputs. A pleasant benefit of our algorithm's simplicity is that it can be implemented with only a few lines of MATLAB code. Furthermore, it is easy to take advantage of the problem structure, particularly when the constraints are low-rank or the cost matrix is sparse. Being a first-order approach, our method is more suitable for problems where low to medium accuracy solutions are acceptable, e.g., in large-scale machine learning problems. We illustrate our method by applying it to varied problems in machine learning: maximum variance unfolding [24], min-balanced cut [13], and large-margin nearest neighbor metric learning [23].

1.1 Related Work

Perturbation: The idea of adding a scaled perturbation function, $\varepsilon f(x)$, to the objective bears a close connection to Tikhonov regularization and exact penalty terms. Mangasarian and Meyer [15] analyzed non-linear (including convex and non-convex) perturbations for linear programs, and our extension to SDPs is inspired by their work. In [14], Mangasarian developed an efficient method for linear programming based on a quadratic perturbation, and characterized the solution of the perturbed problem as the least ℓ_2-norm solution to the original linear program. In a similar vein, we characterize the solution of our perturbed problem as the maximum determinant solution to the original SDP. Other relevant references related to perturbations include the work of Ferris and Mangasarian [9] who extend the perturbation results of [15] to general convex programs by applying them to linearizations of the latter. Tseng [21] includes several relevant references and also discusses perturbations by separable nonlinear functions (e.g., $f(\boldsymbol{x}) = \sum_i x_i^2$); in contrast, our perturbation function $f(\mathsf{X}) = -\log\det(\mathsf{X})$ is non-separable in the entries of X. More recently, Friedlander and Tseng [10] discussed perturbations for general convex programs, wherein they also discuss necessary and sufficient conditions for the solution of the perturbed problem to be a solution to the original problem.

Algorithms: Interior point (IP) methods are amongst the most popular techniques for solving SDPs, especially after [1, 16] demonstrated their applicability. The software package SEDUMI implements an IP code for SDPs, and is currently the method of choice for small to medium scale problems. The perturbation function $f(\mathsf{X}) = -\log\det(\mathsf{X})$ is the log-barrier used by many interior point SDP solvers, however, fundamental differences exist between our approach and IP methods. We solve the perturbed problem as a constrained optimization problem with a fixed ε, wherein *only* positive-definiteness is enforced via the log-det barrier (the other constraints are tackled differently). In contrast, IP methods recast the original problem as an unconstrained problem with varying ε, wherein *all* the constraints are enforced via appropriate logarithmic barriers.

In addition to second-order IP methods, the nonlinear programming approach of Burer and Monteiro [4] and the spectral bundle method of Helmberg and Rendl [11] are popular for solving SDPs. Our method presents a new approach that is simpler than existing techniques, both conceptually, as well as from an implementation perspective—surprisingly, without sacrificing too much accuracy. For additional details and references on semidefinite programming, we refer the reader to [20, 22].

2 Problem Formulation

A standard formulation for semidefinite programming (SDP) is

$$\begin{aligned} \min \quad & \mathrm{Tr}(\mathsf{C}\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) \leq b_i, \quad 1 \leq i \leq m \\ & \mathsf{X} \succeq 0. \end{aligned} \tag{P}$$

Problem (P) is a convex optimization problem, and as mentioned previously, for simplicity and scalability we wish to develop a first-order method for solving large-scale problems. The key insight that enables us to develop a first-order method is the introduction of a strictly convex perturbation to (P). Formally, instead of (P) we propose to solve

$$\begin{aligned} \min \quad & \mathrm{Tr}(\mathsf{CX}) - \varepsilon \log\det(\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) \leq b_i, \quad 1 \leq i \leq m \\ & \mathsf{X} \succeq 0, \end{aligned} \tag{PT}$$

where $\varepsilon \in [0, \bar{\varepsilon}]$ is some pre-specified constant, and $f(\mathsf{X}) = -\log\det(\mathsf{X})$ is the perturbation function. Note that other perturbation functions such as $\|\mathsf{X}\|_{\mathrm{F}}^2$ or $\mathrm{Tr}(\mathsf{X}\log\mathsf{X})$ can also be considered, but initially we restrict our attention to the log-det perturbation. This perturbation brings in two crucial advantages: i) it makes the problem strictly convex, thereby permitting us to adapt a successive projections technique for an efficient solution, and ii) it enables our algorithm to easily enforce positive-definiteness of X because $-\log\det(\mathsf{X})$ is a "natural" barrier function for positive definite matrices.[1]

Remark: We solve the constrained optimization problem (PT) directly by successively enforcing each linear inequality constraint. The positive-definiteness constraint is implicitly enforced by the log-det term. Interior Point methods instead the following problem (or equivalently the dual formulation thereof)

$$\min_{\mathsf{X}} \quad \mathrm{Tr}(\mathsf{CX}) - \varepsilon_t \log\det(\mathsf{X}) - \gamma_t \sum_{i=1}^{m} \log([\mathrm{Tr}(\mathsf{A}_i\mathsf{X}) - b_i]_+),$$

where ε_t and γ_t are varied according to a prescribed schedule. In the limit, as $t \to \infty$, these methods are guaranteed to return a solution lying strictly in the interior of the feasible set. From this formulation it is easy to see that IP methods can become computationally demanding when the number of constraints is very large, a difficulty that our method is able to circumvent by going through the constraints one by one.

2.1 Analysis

To analyze the relationship between (P) and (PT) consider the following auxiliary problem:

$$\begin{aligned} \min \quad & f(\mathsf{X}) = -\log\det(\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) \leq b_i, \quad 1 \leq i \leq m, \\ & \mathrm{Tr}(\mathsf{CX}) \leq \bar{\theta}, \quad \mathsf{X} \succeq 0, \end{aligned} \tag{AUX}$$

where $\bar{\theta}$ is the minimum value achieved by (P) (assuming it has a bounded minimum). Problem (AUX) optimizes the convex perturbation $f(\mathsf{X})$ over the set of optimal solutions of (P). This relation is formalized by Theorem 2.1 below (adapted from [15]), which applies to not only $f(\mathsf{X}) = -\log\det(\mathsf{X})$, but also to other convex perturbation functions such as $f(\mathsf{X}) = \frac{1}{2}\|\mathsf{X}\|_{\mathrm{F}}^2$ and $f(\mathsf{X}) = \mathrm{Tr}(\mathsf{X}\log\mathsf{X})$.

Theorem 2.1. *Let $\overline{S} \neq \emptyset$ be the set of optimal solutions of* (P)*. Further, assume that f is differentiable on $\overline{S}$, strong duality holds for* (P)*, and that* (AUX) *has a KKT point. Then, there exists an $\overline{\mathsf{X}} \in \overline{S}$ and an $\bar{\varepsilon} > 0$, such that for each $\varepsilon \in [0, \bar{\varepsilon}]$ there exist $\overline{\mathsf{Z}}^{pt}, \bar{\boldsymbol{\nu}}^{pt}$, such that $(\overline{\mathsf{X}}, \overline{\mathsf{Z}}^{pt}, \bar{\boldsymbol{\nu}}^{pt})$ is a KKT point of* (PT)*, whence $\overline{\mathsf{X}}$ solves the perturbed problem* (PT).

[1] Note that the log-det barrier function forces the solution to lie in the interior of the semidefinite cone. However, as with interior point methods, this does not cause any practical difficulties when dealing with SDPs.

Proof. Let $\mathcal{A}(\bar{\boldsymbol{\nu}}) = \sum_i \bar{\nu}_i \mathsf{A}_i$, and $(\overline{\mathsf{X}}, \overline{\mathsf{Z}}, \bar{\boldsymbol{\nu}}, \gamma)$ be a KKT point of (AUX), whereby,

$$\begin{aligned} \nabla f(\overline{\mathsf{X}}) + \mathcal{A}(\bar{\boldsymbol{\nu}}) + \gamma C - \overline{\mathsf{Z}} &= 0, \\ \mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) &= \bar{\theta}, \\ \mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) &\le b_i, \quad 1 \le i \le m, \\ \bar{\nu}_i\big(\mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) - b_i\big) &= 0, \quad 1 \le i \le m, \\ \overline{\mathsf{X}}, \overline{\mathsf{Z}} \succeq 0, \bar{\boldsymbol{\nu}} &\ge 0, \gamma \ge 0. \end{aligned} \tag{2.1}$$

Note that the equality $\mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) = \bar{\theta}$ must hold for a KKT point of (AUX), because any feasible point of (AUX) is also feasible for (P), and if $\mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) \le \bar{\theta}$ is not satisfied with strict equality, it would contradict the assumption that $\bar{\theta}$ is the minimum value of (P). Thus $\overline{\mathsf{X}} \in \overline{S}$.

Since $\overline{\mathsf{X}}$ is an optimal solution of (P), there is a KKT point $(\overline{\mathsf{X}}, \mathsf{Z}, \boldsymbol{\nu})$ of (P) satisfying

$$\begin{aligned} \mathsf{C} + \mathcal{A}(\boldsymbol{\nu}) - \mathsf{Z} &= 0 \\ \mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) &\le b_i, \quad 1 \le i \le m, \\ \nu_i(\mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) - b_i) &= 0, \quad 1 \le i \le m, \\ \overline{\mathsf{X}}, \mathsf{Z} \succeq 0, \boldsymbol{\nu} &\ge 0. \end{aligned} \tag{2.2}$$

We now combine (2.1) and (2.2) to construct a KKT point of (PT). Consider the following two cases (γ is the dual variable corresponding to the $\mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) \le \bar{\theta}$ constraint):

Case 1: $\gamma = 0$. For any $\varepsilon \ge 0$, $(\overline{\mathsf{X}}, \varepsilon\overline{\mathsf{Z}} + \mathsf{Z}, \varepsilon\bar{\boldsymbol{\nu}} + \boldsymbol{\nu})$ is a KKT point of (PT). This is easily verified by multiplying (2.1) by ε and adding the result to (2.2). Formally,

$$\begin{aligned} \varepsilon\nabla f(\overline{\mathsf{X}}) + \mathcal{A}(\varepsilon\bar{\boldsymbol{\nu}} + \boldsymbol{\nu}) + \mathsf{C} - (\varepsilon\overline{\mathsf{Z}} + \mathsf{Z}) &= 0, \\ \mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) &\le b_i, \quad 1 \le i \le m, \\ (\varepsilon\bar{\nu}_i + \nu_i)\big(\mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) - b_i\big) &= 0, \quad 1 \le i \le m, \\ \overline{\mathsf{X}}, \varepsilon\overline{\mathsf{Z}} + \mathsf{Z} \succeq 0, (\varepsilon\bar{\boldsymbol{\nu}} + \boldsymbol{\nu}) &\ge 0, \end{aligned}$$

are the KKT conditions for (PT). Note that $\overline{\mathsf{Z}}^{pt} = \varepsilon\overline{\mathsf{Z}} + \mathsf{Z}$ and $\bar{\boldsymbol{\nu}}^{pt} = \varepsilon\bar{\boldsymbol{\nu}} + \boldsymbol{\nu}$.

Case 2: $\gamma > 0$. For any $\lambda \in [0, 1]$, $(\overline{\mathsf{X}}, \overline{\mathsf{Z}}^{pt}, \boldsymbol{\nu}^{pt})$ is a KKT point of (PT), with $\varepsilon = \lambda/\gamma$, $\overline{\mathsf{Z}}^{pt} = (1-\lambda)\mathsf{Z} + \frac{\lambda}{\gamma}\overline{\mathsf{Z}}$, and $\bar{\boldsymbol{\nu}}^{pt} = (1-\lambda)\boldsymbol{\nu} + \frac{\lambda}{\gamma}\bar{\boldsymbol{\nu}}$. As for Case 1, this is easily verified by multiplying (2.1) by λ/γ, (2.2) by $1 - \lambda$, and adding the two. Note that $\bar{\varepsilon} = 1/\gamma$.

Finally, since the objective function of (PT) is strictly convex and we assume strong duality holds for (P), strong duality also holds for (PT). Thus, the KKT conditions are sufficient for $\overline{\mathsf{X}}$ to be the minimum of (PT). $\square$

Theorem 2.1 above shows that a solution of (P) is also a solution of (PT) for an appropriate ε. Since the latter problem has a strictly convex objective, it has a unique minimum, whereby, solving it automatically yields a solution to (P). In other words, solving (PT) leads to a solution of (P) because the perturbation function allows us to pick a *unique* solution from amongst all the solutions to (P). This statement is formalized by the following corollary to Theorem 2.1.

Corollary 2.2. *Assume the conditions of Theorem 2.1 hold, so that $\varepsilon \in [0, \bar{\varepsilon})$. Let $\overline{S} \ne \emptyset$ be the set of optimal solutions of* (P) *and X^* the solution to* (PT). *Then,*

$$\mathsf{X}^* = \underset{\overline{\mathsf{X}} \in \overline{S}}{\mathrm{argmax}} \quad \det(\overline{\mathsf{X}}),$$

i.e., X^* is the maximum-determinant *solution to* (P).

Proof. Let $\overline{\mathsf{X}}$ be any solution of (P). Because X^* is *the* solution to (PT), by Theorem 2.1 X^* it is also optimal for (P). Thus, $\mathrm{Tr}(\mathsf{C}\mathsf{X}^*) = \mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}})$, and since X^* solves (PT), we further have $\mathrm{Tr}(\mathsf{C}\mathsf{X}^*) - \log\det(\mathsf{X}^*) \leq \mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) - \log\det(\overline{\mathsf{X}})$. Therefore, $-\log\det(\mathsf{X}^*) \leq -\log\det(\overline{\mathsf{X}})$, or equivalently $\det(\mathsf{X}^*) \geq \det(\overline{\mathsf{X}})$. $\square$

Remark: If we used $f(\mathsf{X}) = \frac{1}{2}\|\mathsf{X}\|_\mathrm{F}^2$ as the perturbation function, we would obtain the minimum Frobenius-norm solution to the original SDP. See Section 3.2 for details on this alternative.

2.2 Error-bounds

The hypotheses of Theorem 2.1 may not always be satisfied. In particular, there could be problem instances for which (AUX) might not possess a KKT point (due to the rigid constraint that forces the interior of the feasible set to be empty). Detecting such a situation *a priori* can be difficult and it is valuable to assess how different the perturbed problem can be in comparison to the original. Furthermore, in practice we do not know $\bar{\varepsilon}$ when solving an SDP, so we hope that the ε that we choose will yield a solution that is "close" to the solution of the unperturbed problem even if $\varepsilon > \bar{\varepsilon}$. The theorem below (adapted from [10]) shows that under fairly mild assumptions, the solution to the perturbed problem is close to the solution of the unperturbed problem when the unperturbed solution is well-conditioned. We formalize this as follows:

Theorem 2.3. *Let $\mathsf{X}^* \in \overline{S}$ denote an optimal solution to (P). Suppose there exist $\tau > 0$ and $\gamma > 1$ such that*

$$\mathrm{Tr}(\mathsf{C}\mathsf{X}) - \mathrm{Tr}(\mathsf{C}\mathsf{X}^*) \geq \tau\,\mathrm{dist}(\mathsf{X}, \overline{S})^\gamma \quad \textit{for all } \mathsf{X} \textit{ feasible for } (\mathrm{P}), \tag{2.3}$$

where $\mathrm{dist}(\mathsf{X}, \overline{S}) = \min_{\mathsf{X}^* \in \overline{S}} \|\mathsf{X} - \mathsf{X}^*\|_F$. Then, for any $\bar{\varepsilon} > 0$, there exists $\bar{\tau} > 0$, such that

$$\mathrm{dist}(\mathsf{X}(\varepsilon), \overline{S})^{\gamma-1} \leq \bar{\tau}\varepsilon, \quad \textit{for all} \quad \varepsilon \in (0, \bar{\varepsilon}],$$

where $\mathsf{X}(\varepsilon)$ is the optimal solution to (PT).

Proof. For any $\varepsilon > 0$, let $\mathsf{X}^*(\varepsilon) = \mathrm{argmin}_{\mathsf{X}^* \in \overline{S}} \|\mathsf{X}(\varepsilon) - \mathsf{X}^*\|_\mathrm{F}$, so that

$$\mathrm{Tr}(\mathsf{C}\mathsf{X}^*(\varepsilon)) - \varepsilon\log\det(\mathsf{X}^*(\varepsilon)) \geq \mathrm{Tr}(\mathsf{C}\mathsf{X}(\varepsilon)) - \varepsilon\log\det(\mathsf{X}(\varepsilon)),$$

since $\mathsf{X}(\varepsilon)$ is the optimal solution to (PT). Now, using (2.3) we obtain

$$\mathrm{Tr}(\mathsf{C}\mathsf{X}(\varepsilon)) - \varepsilon\log\det(\mathsf{X}(\varepsilon)) \geq \mathrm{Tr}(\mathsf{C}\mathsf{X}^*(\varepsilon)) + \tau\|\mathsf{X}(\varepsilon) - \mathsf{X}^*(\varepsilon)\|_\mathrm{F}^\gamma - \varepsilon\log\det(\mathsf{X}(\varepsilon)),$$

which implies

$$\tau\|\mathsf{X}(\varepsilon) - \mathsf{X}^*(\varepsilon)\|_\mathrm{F}^\gamma \leq \varepsilon\big(\log\det(\mathsf{X}(\varepsilon)) - \log\det(\mathsf{X}^*(\varepsilon))\big).$$

Exploiting the concavity of $\log\det$, we have

$$\log\det(\mathsf{X}(\varepsilon)) - \log\det(\mathsf{X}^*(\varepsilon)) \leq \big\langle(\mathsf{X}^*(\varepsilon))^{-1}, \mathsf{X}(\varepsilon) - \mathsf{X}^*(\varepsilon)\big\rangle \leq \|(\mathsf{X}^*(\varepsilon))^{-1}\|_\mathrm{F}\|\mathsf{X}(\varepsilon) - \mathsf{X}^*(\varepsilon)\|_\mathrm{F},$$

which we can combine with the former inequality to finally obtain

$$\tau\|\mathsf{X}(\varepsilon) - \mathsf{X}^*(\varepsilon)\|_\mathrm{F}^{\gamma-1} \leq \varepsilon\|(\mathsf{X}^*(\varepsilon))^{-1}\|_\mathrm{F}.$$

Setting $\bar{\tau} = \frac{\|(\mathsf{X}^*(\varepsilon))^{-1}\|_F}{\tau}$ completes the proof. $\square$

From the theorem above, we see that the solutions to the perturbed and unperturbed problems are close when $\|(\mathsf{X}^*(\varepsilon))^{-1}\|_F$ is small (which is true when the eigenvalues of $\mathsf{X}^*(\varepsilon)$ are large). We refer the reader to [10] for more discussion and references related to error-bounds such as the one described above.

3 Algorithms

There are many potential methods for optimizing (PT). However, the use of the log-det perturbation function lends itself well to Bregman's method, a general technique for minimizing a strictly convex function subject to linear equality and inequality constraints [6]. This method proceeds iteratively by choosing a single constraint at each iteration, projecting the current solution onto that constraint, and performing an appropriate correction. The projection done at each step is not an orthogonal projection, but rather a *Bregman projection*, which is tailored to the particular convex function to be minimized. Under mild assumptions, this method provably converges to the globally optimal solution. Equivalently, we may view Bregman's method as a dual-coordinate ascent procedure, where we choose a single dual variable (corresponding to one constraint), fix all other dual variables, and maximize the dual with respect to the chosen dual variable.

Below, we sketch out the computation involved in projecting the current solution onto a single affine equality or inequality constraint. As we will see, for low-rank constraint matrices A_i, the Bregman projection for $h(\mathsf{X}) = \mathrm{Tr}(\mathsf{CX}) - \varepsilon \log\det(\mathsf{X})$ can be performed in $O(n^2)$ time as a simple low-rank update to the current solution.

We further discuss the Frobenius perturbation in Section 3.2, where $f(\mathsf{X}) = \frac{1}{2}\|\mathsf{X}\|_{\mathrm{F}}^2$, and we briefly describe methods for optimizing SDPs with the Frobenius perturbation.

3.1 Projection onto a Single Constraint

The gradient of h with respect to X is given by $\nabla h(\mathsf{X}) = \mathsf{C} - \varepsilon \mathsf{X}^{-1}$. Bregman's method chooses a single constraint i at every iteration, and projects the current matrix X_t onto that constraint via a Bregman projection to form X_{t+1}. For equality constraints, the Bregman projection is performed by solving the following system of nonlinear equations for α and X_{t+1} (this is derived by differentiating the Lagrangian with respect to X and the dual variables):

$$\begin{aligned} \nabla h(\mathsf{X}_{t+1}) &= \nabla h(\mathsf{X}_t) + \alpha \mathsf{A}_i \\ \mathrm{Tr}(\mathsf{X}_{t+1}\mathsf{A}_i) &= b_i. \end{aligned} \tag{3.1}$$

Simplifying the top equation yields $\mathsf{X}_{t+1} = (\mathsf{X}_t^{-1} - \frac{\alpha}{\varepsilon}\mathsf{A}_i)^{-1}$. When A_i is of low-rank, then the Sherman-Morrison-Woodbury inverse formula may be applied to this update. For example, if $\mathsf{A}_i = \boldsymbol{z}_i\boldsymbol{z}_i^T$, then the top equation simplifies to

$$\mathsf{X}_{t+1} = \mathsf{X}_t + \frac{\frac{\alpha}{\varepsilon}\mathsf{X}_t\boldsymbol{z}_i\boldsymbol{z}_i^T\mathsf{X}_t}{1 - \frac{\alpha}{\varepsilon}\boldsymbol{z}_i^T\mathsf{X}_t\boldsymbol{z}_i}. \tag{3.2}$$

Given that $\mathrm{Tr}(\mathsf{X}_{t+1}\boldsymbol{z}_i\boldsymbol{z}_i^T) = \boldsymbol{z}_i^T\mathsf{X}_{t+1}\boldsymbol{z}_i = b_i$, we can solve for the projection parameter α in closed form as:

$$\alpha = \frac{\varepsilon(b_i - \boldsymbol{z}_i^T\mathsf{X}_t\boldsymbol{z}_i)}{b_i \cdot \boldsymbol{z}_i^T\mathsf{X}_t\boldsymbol{z}_i}.$$

We then use this choice of α to update X_{t+1}. After solving for α, we update X_{t+1} via (3.2); note that this update is a rank-one update, and can be performed in $O(n^2)$ time.

If instead the constraint is an inequality constraint, then a correction must be enforced to make sure that the corresponding dual variable remains non-negative. Let λ_i correspond to the dual variable for constraint i. After solving for α as in the equality case, we set $\alpha' = \min(\lambda_i, \alpha)$ and $\lambda_i = \lambda_i - \alpha'$. Finally, we update to X_{t+1} using (3.2) with α' in place of α. Note that the dual variables λ_i and the starting matrix X_0 are initialized so that $\nabla h(\mathsf{X}_0) = -\sum_i \lambda_i \mathsf{A}_i$, $\mathsf{X}_0 \succ 0$, and $\lambda_i \geq 0$ for all inequality constraints.

This general approach is summarized as Algorithm 1, which converges to the globally optimal solution to (PT). For further details on the convergence of Bregman's projection method, see [6]. In practice, when choosing a constraint at each iteration, we choose the constraint that is the most violated. For low-rank constraints, determining the most violated constraint can usually be performed efficiently; for example, if

Algorithm 1 Semidefinite programming with Bregman's method

```
Input: $\{\mathsf{A}_i, b_i\}_{i=1}^m$: input constraints, $\mathsf{C}$: input matrix, $\varepsilon$: tradeoff parameter
Output: $\mathsf{X}$: output PSD matrix
1. Initialize $\mathsf{X}$ and $\lambda_i$ such that $\mathsf{X} \succ 0$, $\lambda_i \geq 0$ for inequality constraints, and $\mathsf{X} = \varepsilon(\mathsf{C} + \sum_i \lambda_i \mathsf{A}_i)^{-1}$.
2. repeat
    2.1. Pick a constraint (e.g., the most violated constraint) $(\mathsf{A}_i, b_i)$.
    2.2. Solve (3.1) for $\alpha$.    {Can be done in closed form for rank-1 to rank-4 constraints}
    2.3. If constraint $i$ is an inequality constraint, $\alpha \leftarrow \min(\lambda_i, \alpha)$, $\lambda_i \leftarrow \lambda_i - \alpha$.
    2.4. $\mathsf{X} \leftarrow (\mathsf{X} - \frac{\alpha}{\varepsilon} A_i)^{-1}$.
3. until convergence
```

each constraint can be evaluated in constant time, then the most violated constraint can be found in $O(m)$ time, which is generally much less than the cost of a single projection.

Note that when the constraint matrices A_i are high-rank (greater than 4), it is not possible to solve for α in closed-form; this is because the equation for solving for α with rank-k constraint matrices involves finding the roots of a polynomial of degree k. In such higher-rank cases, we can perform *approximate Bregman projections* using the secant method to calculate an approximate α [6]. Updating X also becomes commensurately more expensive for higher-rank constraints.

3.2 Frobenius norm perturbation

So far we have mainly focused our attention on the log-det perturbation for SDPs. In this section, we briefly digress to look at two simple approaches for handling the Frobenius norm perturbation, i.e., $f(\mathsf{X}) = \frac{1}{2}\|\mathsf{X}\|_{\mathrm{F}}^2$. Bregman projections alone are not applicable for the Frobenius perturbation; they cannot be applied to the semi-definiteness constraint due to its non-polyhedral nature. With log-det, the domain of the perturbation function is the set of positive-definite matrices, and so positive definiteness is automatically enforced. For the Frobenius perturbation, we must find a way to explicitly maintain positive definiteness.

To tackle this new difficulty imposed by the semidefiniteness constraint we propose two methods below. The first approach invokes Dykstra's method [8], while the second one calls upon the gradient ascent scheme of [3]. Both these approaches are based upon replacing the original SDP by its perturbed version, which is then interpreted as a *least-squares* SDP. To the best of our knowledge, both these approaches to solving SDPs are new.

3.2.1 Least-squares SDP

With $f(\mathsf{X}) = \frac{1}{2}\|\mathsf{X}\|_{\mathrm{F}}^2$, Problem (P) becomes

$$\begin{aligned} \min \quad & \mathrm{Tr}(\mathsf{CX}) + \varepsilon \tfrac{1}{2}\|\mathsf{X}\|_{\mathrm{F}}^2, \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) \leq b_i, \quad 1 \leq i \leq m \\ & \mathsf{X} \succeq 0, \end{aligned}$$

which may be rewritten as the least-squares SDP problem

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathsf{X} + \varepsilon^{-1}\mathsf{C}\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) \leq b_i, \quad 1 \leq i \leq m \\ & \mathsf{X} \succeq 0. \end{aligned} \tag{PT2}$$

3.2.2 Solving (PT2) via Dykstra's Method

Dykstra's method is similar to Bregman's method, except that it is able to handle arbitrary convex constraint sets, though at the same time being restricted to only quadratic objective functions. Intuitively, Dykstra's method cycles through the constraints like Bregman's method. At each step, it first deflects the current iterate by a small amount before projecting onto the associated constraint. After the projection, it accumulates the difference between the projected and un-projected variables into the deflection term, which is then used again the next time around when projecting onto the same constraint. By maintaining these extra history variables (analogous to the dual variables maintained by Bregman's method), Dykstra's method ensures convergence to the optimum, rather than just to a point in the intersection of the convex constraints.

For the perturbed problem (PT2), the resulting adaptation of Dysktra's method is provided by Algorithm 2 below. For mathematical details pertinent to Dykstra's algorithm, the reader is referred to [7].

Algorithm 2 Semidefinite programming with Dykstra's method

```
Input: ${\mathsf{A}_i, b_i}_{i=1}^m$: input constraints, $\mathsf{C}$: input matrix, $\varepsilon$: tradeoff parameter
Output: $\mathsf{X}$: output PSD matrix
1. Initialize $\mathsf{X}_{\text{old}} = -\varepsilon^{-1}\mathsf{C}$, $\Lambda_i = 0$ for $i = 1, \ldots, m+1$.
2. repeat
    2.1. For $i = 1$ to $m+1$ (each constraint):
        $\mathsf{X}_{\text{new}} \leftarrow \mathcal{P}_i(\mathsf{X}_{\text{old}} + \Lambda_i)$
        $\Lambda_i \leftarrow \mathsf{X}_{\text{old}} - \mathsf{X}_{\text{new}} + \Lambda_i$
        {$\mathcal{P}_i(\mathsf{X})$ denotes orthogonal projection of $\mathsf{X}$ onto $i$-th constraint}
3. until convergence
```

Note that in Algorithm 2 we have (arbitrarily) numbered the semi-definiteness constraint to be the $(m+1)$-th constraint. The orthogonal projections should naturally be implemented to exploit the sparsity of the input problem. The *history* matrices Λ_i should also not be stored explicitly to avoid excess storage.

3.2.3 Solving (PT2) via Projected Gradients

Boyd and Xiao [3] remarked in their paper that their method for the least-squares SDP could also be solved via a standard SDP package, though this approach is inefficient. Here we are suggesting the opposite, i.e., using our Frobenius norm perturbation, several SDPs (in particular those with negative-definite cost matrices C) can be solved by converting them into least-squares SDP problems that can be easily tackled by the projected gradient approach of [3].

After some simple algebra, it can be shown that the dual of (PT2) is given by

$$\begin{aligned} \min \quad \psi(\mathsf{Z}, \boldsymbol{\mu}) &= \frac{1}{2}\|\mathsf{Z} - \epsilon^{-1}\mathsf{C} - \mathsf{A}(\boldsymbol{\mu})\|_{\mathrm{F}}^2 + \frac{1}{2\epsilon^2}\|\mathsf{C}\|_{\mathrm{F}}^2 - \boldsymbol{\mu}^T\boldsymbol{b} \\ \text{subject to} \quad & \mathsf{Z} \succeq 0, \quad \boldsymbol{\mu} \geq 0, \end{aligned} \tag{3.3}$$

where $\mathsf{A}(\boldsymbol{\mu}) = \sum_i \mu_i \mathsf{A}_i$. The dual problem (3.3) can be solved by the following simple projected (sub)gradient method (illustrated as Algorithm 3). See [3] for more details.

If Problem (PT2) is strictly feasible, then for small enough step-size γ, Algorithm 3 is guaranteed to converge, i.e., X and $\boldsymbol{\mu}$ converge to their optimal values. As usual, too small a value of γ can lead to slow-convergence, whereby depending upon the problem one must select it appropriately. Further note that in practice, several implementation details need to be handled for making Algorithm 3 efficient. A discussion of such issues may also be found in [3].

Algorithm 3 Semidefinite programming using Projected Gradients

```
Input: $\{\mathsf{A}_i, b_i\}_{i=1}^m$: input constraints, $\mathsf{C}$: input matrix, $\gamma$: step size parameter
Output: $\mathsf{X}$: output PSD matrix
1. Initialize $\mathsf{X}_0 = -\varepsilon^{-1}\mathsf{C}$, $\boldsymbol{\mu} = \mathbf{0}$.
2. repeat
   Let $\mathsf{X} = (\mathsf{X}_0 - \mathsf{A}(\boldsymbol{\mu}))_+$            {Projection onto semidefinite cone}
   Projected gradient update for $\boldsymbol{\mu}$
     2.1. Evaluate $\partial\psi/\partial\mu_i = \mathrm{Tr}(\mathsf{A}_i\mathsf{X}) - b_i$
     2.2. Let $\mu_i = \big(\mu_i + \gamma(\partial\psi/\partial\mu_i)\big)_+$
3. until convergence
```

3.3 Examples

We briefly highlight a few SDPs from machine learning that are well-suited to our algorithmic framework, as they feature low-rank constraint matrices.

Nonlinear Embedding: Semidefinite embedding [24] (also called maximum variance unfolding) is a non-linear dimensionality reduction problem which aims to find a low-dimensional embedding of the input data such that the variance in the data is maximized while the distances among a set of nearest neighbors S is maintained, and a centering constraint is enforced. The total number of constraints is nk, where n is the number of data points and k is the number of nearest neighbors, and all constraints are rank-one. Given a set S of neighbor pairs, each with a target distance D_{ij}, the semidefinite embedding problem can be formalized as:

$$\begin{array}{ll} \max_{\mathsf{X}} & \mathrm{Tr}(\mathsf{X}) \\ \text{subject to} & \mathsf{X}_{ii} + \mathsf{X}_{jj} - 2\mathsf{X}_{ij} = \mathsf{D}_{ij},\ (i,j) \in S \\ & \boldsymbol{e}^T\mathsf{X}\boldsymbol{e} = 0 \\ & \mathsf{X} \succeq 0. \end{array}$$

A related problem is the robust Euclidean embedding problem [5], which seeks to find the closest (squared) Euclidean distance matrix D to some given input dissimilarity matrix D_0 under the elementwise ℓ_1 loss: $\|D - D_0\|_1$. Appropriate manipulation of this objective transforms it into an SDP with rank-two constraint matrices.

Graph Cuts: Several graph cut problems can be relaxed as SDPs. For example, the minimum balanced cut problem [13] has been successfully used when finding balanced clusters of skewed-degree distribution graphs (such as power-law graphs). A relaxation to the minimum balanced cut problem may be posed as an SDP with $|V| + 1$ rank-one constraints, with $|V|$ the number of vertices in the graph. Given a graph Laplacian L, the min balanced cut problem can be expressed as:

$$\begin{array}{ll} \min_{\mathsf{X}} & \mathrm{Tr}(\mathsf{L}\mathsf{X}) \\ \text{subject to} & \mathrm{diag}(\mathsf{X}) = \boldsymbol{e} \\ & \boldsymbol{e}^T\mathsf{X}\boldsymbol{e} = 0 \\ & \mathsf{X} \succeq 0. \end{array}$$

Metric Learning: Various metric learning algorithms have been posed as SDPs. In particular, the method of large-margin nearest neighbors (LMNN) [23] guarantees that distances between nearest neighbors in the same class is much smaller than distances between points in different classes. The resulting SDP has rank-3 constraints. The method of [23] attempts to find a Mahalanobis distance matrix A such that two neighboring

Table 1: Accuracy Results on Min Balanced Cut

	Trace Value		Max. Violation	
Data Set	SEDUMI	SDPLogDet	SEDUMI	SDPLogDet
Iris	1.020×10^4	1.021×10^4	9.252×10^{-11}	9.633×10^{-4}
Wine	5.657×10^3	5.672×10^3	8.290×10^{-10}	9.995×10^{-4}
Ionosphere	6.755×10^3	6.766×10^3	9.912×10^{-4}	9.912×10^{-4}
Soybean	1.239×10^5	1.239×10^5	2.978×10^{-11}	9.937×10^{-4}
Diabetes	1.704×10^4	1.711×10^4	3.785×10^{-11}	9.454×10^{-4}

points in the same class have distances much smaller than two points in different classes. Let $\eta_{ij} = 1$ if points i and j are neighbors (and 0 otherwise), $y_{ij} = 1$ if the labels of points i and j match (and 0 otherwise), ξ_{ijl} correspond to slack variables for the constraints, and $\mathsf{C}_0 = \sum_{ij} \eta_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$. Then the corresponding SDP to be solved is formalized as:

$$\begin{aligned} \min_{\mathsf{A},\xi} \quad & \mathrm{Tr}(\mathsf{C}_0\mathsf{A}) + \gamma \sum_{ijl} \eta_{ij}(1 - y_{jl})\xi_{ijl} \\ \text{subject to } & d_{\mathsf{A}}(\boldsymbol{x}_i, \boldsymbol{x}_l) - d_{\mathsf{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1 - \xi_{ijl}, \\ & \xi_{ijl} \geq 0, \quad \mathsf{A} \succeq 0. \end{aligned} \tag{3.4}$$

Collaborative Filtering: The maximum margin matrix factorization SDP for collaborative filtering [19] has simple, low-rank constraints. Other first-order methods have been proposed; however, these methods work on a slightly different (non-convex) optimization problem, which may lead to poor local optima.

4 Experiments

We now present initial results comparing our proposed semidefinite programming algorithm (SDPLogDet) to existing SDP software. We considered several SDPs while experimenting with our software: semidefinite embedding, LMNN, min balanced cut, and robust Euclidean embedding. These SDPs all have constraints that are rank-one, rank-two, or rank-three, making them appropriate for our method. We compare with DSDP [2] and SEDUMI [17], both standard off-the-shelf SDP solvers, as well as Weinberger et al.'s [23] implementation of LMNN (obtained from the authors)—a special-purpose sub-gradient algorithm that outperforms standard solvers on the LMNN problem—and the sub-gradient algorithm for robust Euclidean embedding described in [5]. All software uses the MATLAB interface, with code written in C and MATLAB. It is difficult to compare various software packages for SDPs, especially given the number of tunable parameters, so in all experiments, we used the default parameters for SEDUMI and DSDP. We tested the above algorithms and problems on standard UCI data sets[2], as well as on synthetic data sets. In all experiments, we set $\varepsilon = 10^{-1}$; we observed empirically that smaller ε led to slower convergence, and that with $\varepsilon = 10^{-1}$, the solutions obtained by our algorithm were very close to the globally optimal solutions. Furthermore, we demonstrate that our algorithm performs well when the SDP is ultimately used for a machine learning task; in particular, we see that classification results for metric learning using the output of our algorithm are as good as those obtained using existing methods.

4.1 Accuracy and Scalability of the Proposed Method

We first determine the accuracy and scalability of the proposed methods on some simple SDPs: min balanced cut and semidefinite embedding. In order to judge the accuracy of our method, Table 1 lists results for min balanced cut on some of the UCI data sets tested. The graphs for this SDP were constructed using

[2]Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

the Gram matrix of the data points (and scaled so that edge weights were between 0 and 1). We see that the accuracy of our method in terms of the final objective function value is very close to the accuracy of SEDUMI; further improvements may be gained by setting ε to be smaller or running more iterations of our algorithm. Our maximum violation is higher since we set our convergence criterion to stop when the max violation was smaller than 10^{-3}. As expected, we exhibited slower convergence (total time taken to converge) than SEDUMI for this SDP (our method is a first-order algorithm).

Table 2: UCI Data Sets

Name	No. of points	No. of dims
Iris	150	4
Wine	178	13
Ionosphere	351	34
Soybean	683	35
Diabetes	768	8

Table 3: Memory overhead (in megabytes) for performing semi-definite embedding. A '—' indicates that the method could not run due to memory requirements.

n	SEDUMI	DSDP	SDPLogDet
100	13	3	1
500	222	67	5
1000	881	263	35
1500	1930	586	52
2000	—	1033	92
2500	—	1608	144
3000	—	2170	207
3500	—	—	253

The key advantage to using our method on these SDPs is its scalability to very large data sets. In an additional experiment, we found that for the splice data set, which is a UCI dataset with 3190 data points, SEDUMI could not run on our machine due to its excessive memory consumption (over 2GB). On the other hand, the SDPLogDet algorithm requires approximately 80MB for this data, and ran successfully.

The scalability of the algorithms is even more drastic for SDPs with a greater number of constraints. While the minimum balanced cut SDP has n constraints, the semidefinite embedding problem has nk constraints, where k is the number of nearest neighbors. In Table 3, we show the maximum memory overhead needed for performing semi-definite embedding on synthetic data. The value n refers to the number of rows/columns of the semi-definite matrix; in these experiments, $k = 5$. SEDUMI requires the most memory, and was unsuccessful in running on problems where n was greater than 1500. DSDP scales to $n = 3000$, whereas LogDet requires no memory overhead beyond the storage of the semi-definite matrix and the constraints. Thus it is feasible to scale the proposed method to very large SDPs.

Table 4 lists preliminary results for the min-balanced cut as achieved using an implementation of our Dykstra based SDP algorithm (with Frobenius norm perturbation). The initial results are promising, though it still remains a challenge to decide when to terminate the Dykstra iterations.

4.2 Comparison to Sub-Gradient Methods

We compared our solver to specialized software for the LMNN problem, which employs sub-gradient methods. Table 5 compares Weinberger et al.'s implementation of LMNN with our proposed algorithm. Here we are considering the use of SDPs as one of the steps in an overall machine learning problem (in this case,

Table 4: Min-balanced cut with Dykstra based SDP solver. Relative eigenvalues violation (REV) = $\sum_j |\lambda_j^-(\mathsf{X})| / \sum_i |\lambda_i(\mathsf{X})|$ measures the departure from positive-definiteness.

Dataset	Obj.	Maxviol.	REV	Time (s)
Iris	1.022e4	4.53e-7	.021	2.3
Wine	5.645e3	6.04e-12	.026	5.3
Soybean	1.236e5	3.37e-9	.065	34.3
Ionosphere	6.755e3	2.21e-9	.092	66
Balance	1.332e5	1.47e-5	.025	47.9
Autos	3.408e3	4.27e-7	.054	7.1
Audiology	9.705e3	1.32e-9	.033	14.4
Breast-cancer	2.508e4	3.17e-10	.029	19.3
Colic	2.284e4	3.30e-10	.037	24.9
Dermatology	2.623e4	1.09e-10	.030	29.1

Table 5: Comparisons of Large-Margin Nearest Neighbors: Test Error and Running Times. SEDUMI cannot be used for this SDP due to the large number of constraints. Our method gives comparable test error and superior running times.

	Test Error			Running Time (secs)	
Data Set	Euclidean	Weinberger et al.	SDPLogDet	Weinberger et al.	SDPLogDet
Iris	.031	.024	.021	1.24	**.119**
Wine	.306	.038	.036	8.77	**.323**
Ionosphere	.164	.123	.119	**9.74**	10.54
Soybean	.122	.082	.079	21.95	**13.25**
Diabetes	.311	.296	.298	47.50	**16.08**

metric learning). The running times are in seconds and represent the average time for the optimization to complete, and we provide the baseline Euclidean test error (i.e., test error with no metric learning) for comparison. Overall, we see that, on all data sets, our test error results are comparable to Weinberger et al., and that on four of the five data sets, the running time is faster.

As we are ultimately interested in test error for this learning problem, in Figure 1, we plot the test error after each projection on the wine data set. For this figure, our algorithm was run on wine until the maximum violation was 10^{-12}. Interestingly, the test error is lowest after approximately 110 projections, suggesting that solving this problem to optimality will not always lead to the lowest test error. It therefore may be sufficient to terminate early, thus highlighting another advantage to using first-order methods for machine learning problems.

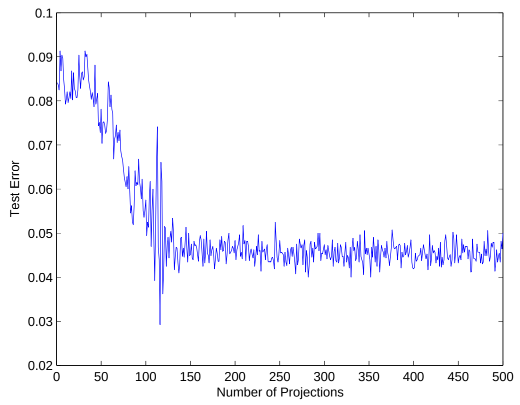
On many SDPs, the large number of constraints make it infeasible to run standard off-the-shelf software such as SEDUMI or DSDP. For example, on the robust Euclidean embedding problem, SEDUMI did not scale to data sets larger than 100 points, which is consistent with the observations in [5].

5 Conclusions and Future Work

In this paper we presented a perturbation based approach to solving SDPs, where we replaced the original linear objective function by a strictly convex objective function. We developed a scalable first-order algorithm based on Bregman projections for solving the perturbed problem that obtains the maximum determinant solution to the original unperturbed problem. Our experimental results are encouraging and they show that despite its simplicity, our method achieves solutions competitive to other SDP methods. Furthermore, due to its modest memory requirements our method is highly scalable, as compared to several standard SDP packages.

Two further directions of future work are open. First is the potential of further improvements to our

Figure 1: Test error as a function of the number of projections on the wine data set



algorithm by either using some second-order information or by a combination with methods such as conjugate gradients. The second is the exploitation of low-rank optimization techniques, particularly for SDPs where the optimal solution can be of much smaller rank than the problem dimensionality.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Opt.*, 5:13–51, 1995.

[2] S. J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Opt.*, 10(2):443–461, 2000.

[3] S. Boyd and L. Xiao. Least-squares covariance matrix adjustment. *SIAM J. Matrix Anal. Appl.*, 27(2):532–546, 2005.

[4] S. Burer and R. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Prog. Ser. B*, 95:329–357, 2003.

[5] L. Cayton and S. Dasgupta. Robust Euclidean embedding. In *ICML*, 2006.

[6] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[7] F. R. Deutsch. *Best Approximation in Inner Product Spaces*. Springer Verlag, first edition, 2001.

[8] R. L. Dykstra. An algorithm for restricted least squares regression. *J. Amer. Statist. Assoc.*, 78, 1983.

[9] M. C. Ferris and O. L. Mangasarian. Finite perturbation of convex programs. *Applied Mathematics and Optimization*, 23:263–273, 1991.

[10] M. P. Friedlander and P. Tseng. Exact regularization of convex programs. *SIAM J. Opt.*, 2007.

[11] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Opt.*, 10:673–696, 2000.

[12] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

[13] K. Lang. Fixing two weaknesses of the spectral method. In *NIPS*, 2005.

[14] O. L. Mangasarian. Normal solution of linear programs. *Math. Prog. Study*, 22:206–216, 1984.

[15] O. L. Mangasarian and R. R. Meyer. Nonlinear Perturbation of Linear Programs. *SIAM J. Cont. & Opt.*, 17(6): 745–752, 1979.

[16] Y. Nesterov and A. Nemirovski. *Interior Point Polynomial Algorithms in Convex Programming*. Number 13 in SIAM Studies in Applied Mathematics. SIAM, 1994.

[17] Sedumi. SeDuMi: Package for optimization over symmetric cones. http://sedumi.mcmaster.ca, 2007.

[18] F. Sha and L. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *ICML*, 2005.

[19] N. Srebro, J. Rennie, and T. Jaakkola. Maximum Margin Matrix Factorizations. In *NIPS*, 2005.

[20] M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[21] Paul Tseng. Convergence and error bounds for perturbation of linear programs. *Computational Optimization and Applications*, 13(1–3):221–230, April 1999.

[22] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[23] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2005.

[24] W. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML*, 2004.