Convex Perturbations for Scalable Semidefinite Programming

Brian Kulis
UC Berkeley EECS and ICSI
Berkeley, CA 94720

Suvrit Sra
MPI for Biological Cybernetics
72076 Tübingen, Germany

Inderjit Dhillon
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Abstract

Many important machine learning problems are modeled and solved via semidefinite programs; examples include metric learning, nonlinear embedding, and certain clustering problems. Often, off-the-shelf software is invoked for the associated optimization, which can be inappropriate due to excessive computational and storage requirements. In this paper, we introduce the use of convex perturbations for solving semidefinite programs (SDPs), and for a specific perturbation we derive an algorithm 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, and thereby scalable, and c) it can easily exploit the structure of a given SDP (e.g., when the constraint matrices are low-rank, a situation common to several machine learning SDPs). A pleasant byproduct of our method is a fast, kernelized version of the large-margin nearest neighbor metric learning algorithm (Weinberger et al., 2005). We demonstrate that our algorithm is effective in finding fast approximations to large-scale SDPs arising in some machine learning applications.

1 INTRODUCTION

There has been a rapid rise in the use of semidefinite programming in the machine learning community over the last few years. Specific examples include nonlinear dimensionality reduction (Weinberger et al., 2004; Sha and Saul, 2005), kernel matrix learning (Lanckriet et al., 2004), maximum margin matrix factorization (Srebro et al., 2005), graph clustering (Lang, 2005), and metric learning (Weinberger et al., 2005). Often, off-the-shelf software such as SEDUMI (2007) or DSDP (Benson et al., 2000) are used for solving the associated semidefinite programs (SDPs). These software packages are effective in finding high-accuracy SDP solutions, but it is difficult to specialize them to particular SDPs. Moreover, the lack of scalability makes generic software restrictive in the face of large problems. For many machine learning applications high-accuracy solutions are not critical, especially if the solution of the SDP is only an intermediate goal—e.g., metric learning for nearest-neighbor classification. In this paper, we trade high-accuracy for speed and obtain an efficient SDP algorithm that is particularly suited for some machine learning applications.

More specifically, we introduce convex perturbations for semidefinite programming and characterize their relation to the unperturbed original. Instead of minimizing $\mathrm{Tr}(\mathsf{CX})$ subject to constraints on X, we minimize the *perturbed* function $\mathrm{Tr}(\mathsf{CX})-\varepsilon \log\det(\mathsf{X})$. We show that for an appropriate $\bar{\varepsilon} > 0$, 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 develop a *simple* first-order algorithm based on Bregman projections that has many benefits: a) it is simple to implement, typically requiring only a few lines of MATLAB code, b) it takes advantage of the problem structure such as low-rank constraints or a sparse cost matrix (both situations are common for machine learning SDPs), and c) it is scalable as it naturally trades off accuracy for speed—a behavior advantageous for several large-scale machine learning problems. We illustrate our method on varied machine learning problems: maximum variance unfolding (Weinberger et al., 2004), min-balanced cut (Lang, 2005), and large-margin nearest neighbor metric learning (LMNN) (Weinberger et al., 2005). For the LMNN problem in particular, our algorithm can naturally be kernelized, which allows metric learning to be performed over arbitrarily high dimensional spaces as long as a suitable input kernel is defined.

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

1.1 BACKGROUND AND RELATED WORK

Our idea of adding a strictly convex perturbation to a traditional SDP is inspired by Mangasarian and Meyer (1979), who analyzed nonlinear perturbations to linear programs. More recently, Friedlander and Tseng (2007) discussed perturbations for general convex programs; they also presented necessary and sufficient conditions for the solution of the perturbed problem to be a solution to the original problem. Our specific perturbation function, i.e., $-\log\det(\mathsf{X})$, goes beyond purely theoretical guarantees, and is critical for obtaining our first-order algorithm. Mangasarian (1984) used a quadratic perturbation to a linear program, 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 (1991) who extend the perturbation results of Mangasarian and Meyer (1979) to general convex programs by applying them to linearizations of the latter. Tseng (1999) includes several relevant references and also discusses perturbations by separable nonlinear functions; in contrast, our perturbation function is non-separable.

A close relative of perturbation is the method of proximal minimization, which dates back to Martinet (1970) and Rockafellar (1976), and generalizations of which have been discussed by several authors, e.g., Censor and Zenios (1997); Auslender and Teboulle (2006). Here, while minimizing a convex function $f(x)$ one performs the following iteration (for $\varepsilon_k > 0$):

$$x^k \in \underset{x\in\Omega}{\operatorname{argmin}}\big\{f(x) + \varepsilon_k^{-1} d(x, x^{k-1})\big\}, \tag{1.1}$$

where the *proximity-function* $d(x,y)$ is chosen to enforce either strict-convexity or to implicitly handle some difficult constraints. Under appropriate conditions, the sequence of iterates $\{x^k\}$ converges to x^*, an optimum solution to $\min_{x\in\Omega} f(x)$. The key difference between perturbation and proximal minimization is that the former assumes the existence of a fixed ε for which solving (1.1) *once* yields a solution to the original problem. Our methods in this paper further require the perturbation to be *strictly-convex*.

Algorithms: Interior point (IP) methods are amongst the most popular techniques for solving SDPs (Alizadeh, 1995; Nesterov and Nemirovski, 1994). The software package SEDUMI implements an IP code for SDPs, and is popular for small to medium scale problems. The log-det perturbation function is also the barrier-function 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, where *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, where *all* the constraints are enforced via appropriate logarithmic barriers.

In addition to second-order IP methods, the nonlinear programming approach of Burer and Monteiro (2003) and the spectral bundle method of Helmberg and Rendl (2000) 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 Todd (2001); Vandenberghe and Boyd (1996).

2 CONVEX PERTURBATIONS

A standard formulation for an SDP is

$$\begin{aligned} \min_{\mathsf{X}\succeq 0} \quad & \mathrm{Tr}(\mathsf{C}_0\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{C}_i\mathsf{X}) \le b_i, \quad 1 \le i \le m. \end{aligned} \tag{P}$$

Problem (P) is convex, and therefore amenable to a wide variety of optimization techniques. Below we develop a new technique for solving (P) by deriving a scalable first-order method that is also simple to understand and implement. The key insight here is the introduction of a strictly convex perturbation, so that instead of (P) we solve

$$\begin{aligned} \min_{\mathsf{X}\succeq 0} \quad & \mathrm{Tr}(\mathsf{C}_0\mathsf{X}) - \varepsilon\log\det(\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{C}_i\mathsf{X}) \le b_i, \quad 1 \le i \le m, \end{aligned} \tag{PT}$$

where $\varepsilon \in [0, \bar{\varepsilon})$ is a pre-specified constant. The most important ingredient here is the perturbation function $f(\mathsf{X}) = -\log\det(\mathsf{X})$ function—other functions such as $\|\mathsf{X}\|_{\mathrm{F}}^2$ or $\|\mathsf{X}\|_1$ can also be considered, but they do not have the desired algorithmic properties. The log-det perturbation is crucial for adapting a successive projections technique to obtain an efficient algorithm, especially because it enables us to enforce positive-definiteness without any eigenvalue computations.

The perturbation above is applicable to any semidefinite program. However, our algorithm for optimizing (PT) will be particularly useful for SDPs that feature low-rank constraint matrices (i.e., each C_i is low-rank) and for kernelization of the large-margin nearest neighbor metric learning algorithm of Weinberger et al. (2005).

2.1 ANALYSIS

To see how (PT) is helpful for solving (P), we briefly analyze their relationship below. Assuming (P) has a bounded minimum $\bar{\theta}$, consider the following auxiliary problem:

$$\begin{aligned} \min \quad & f(\mathsf{X}) = -\log\det(\mathsf{X}) \\ \text{subject to} \quad & \mathrm{Tr}(\mathsf{C}_i\mathsf{X}) \le b_i, \quad 1 \le i \le m, \\ & \mathrm{Tr}(\mathsf{C}_0\mathsf{X}) \le \bar{\theta}, \quad \mathsf{X} \succeq 0. \end{aligned} \tag{AUX}$$

A feasible solution to (AUX) is optimal for (P), whereby combining (AUX) with (P) yields insight into the behavior of (PT). Theorem 2.1 below (adapted from Mangasarian and Meyer (1979)), captures this relationship.

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 $\overline{\mathsf{Z}}^{pt}, \overline{\boldsymbol{\nu}}^{pt}$, such that $(\overline{\mathsf{X}}, \overline{\mathsf{Z}}^{pt}, \overline{\boldsymbol{\nu}}^{pt})$ is a KKT point of* (PT)*, whence $\overline{\mathsf{X}}$ solves the perturbed problem* (PT).

We sketch the proof in the appendix. This theorem shows that for an appropriate value of ε, a solution of (P) is also a solution of (PT). Since (PT) is strictly convex, solving it allows us to pick a *unique* solution from amongst all the solutions of (P); the following corollary further qualifies this statement.

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

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

i.e., X^* is the maximum-determinant *solution to* (P)[1].

Proof. Let $\overline{\mathsf{X}}$ be any solution of (P). Because X^* is *the* solution to (PT), by Theorem 2.1, X^* is also optimal for (P). Thus, $\operatorname{Tr}(\mathsf{C}_0\mathsf{X}^*) = \operatorname{Tr}(\mathsf{C}_0\overline{\mathsf{X}})$, and since X^* solves (PT), we further have $\operatorname{Tr}(\mathsf{C}_0\mathsf{X}^*) - \log\det(\mathsf{X}^*) \leq \operatorname{Tr}(\mathsf{C}_0\overline{\mathsf{X}}) - \log\det(\overline{\mathsf{X}})$. Therefore, $-\log\det(\mathsf{X}^*) \leq -\log\det(\overline{\mathsf{X}})$, or equivalently $\det(\mathsf{X}^*) \geq \det(\overline{\mathsf{X}})$. □

A question that arises is what happens if the hypotheses of Theorem 2.1 are not satisfied—for example, when (AUX) fails to have a KKT point due to infeasible constraints. Detecting such situations *a priori* can be difficult, and it is valuable to see how different the perturbed problem is compared to the original. In practice we do not know $\bar{\varepsilon}$ exactly, when solving an SDP[2]. Fortunately, we can show that the ε we choose yields a solution "close" to the true solution, even if $\varepsilon > \bar{\varepsilon}$. The theorem below (adapted from Friedlander and Tseng (2007)) shows that under fairly mild assumptions, the solution to the perturbed problem is within $O(\varepsilon)$ of the unperturbed solution.

Theorem 2.3. *Let X be feasible and $\overline{\mathsf{X}} \in \overline{S}$ be optimal for (P). Suppose there exist $\tau > 0$ and $\gamma > 0$ such that*

$$\operatorname{Tr}(\mathsf{C}_0\mathsf{X}) - \operatorname{Tr}(\mathsf{C}_0\overline{\mathsf{X}}) \geq \tau \operatorname{dist}(\mathsf{X}, \overline{S})^{\gamma}, \qquad (2.1)$$

where $\operatorname{dist}(\mathsf{X}, \overline{S}) = \min_{\overline{\mathsf{X}} \in \overline{S}} \|\overline{\mathsf{X}} - \mathsf{X}\|_F$. For any $\bar{\varepsilon} > 0$,

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

where X^* is the optimal solution to (PT) and $\overline{\mathsf{X}}(\varepsilon) = \operatorname{argmin}_{\overline{\mathsf{X}} \in \overline{S}} \|\overline{\mathsf{X}} - \mathsf{X}^*\|_F$.

[1]With $f(\mathsf{X}) = \|\mathsf{X}\|_{\mathrm{F}}^2$ as the perturbation, one obtains the minimum Frobenius-norm solution to (P).

[2]For specific SDPs, one can obtain intervals bounding ε. For ML applications ε can be selected via cross-validation.

We sketch the proof of this theorem in the appendix. This theorem shows that the solutions to the perturbed and unperturbed problems are close when $\|\overline{\mathsf{X}}(\varepsilon)^{-1}\|_{\mathrm{F}}$ is small, which in turn occurs when $\overline{\mathsf{X}}(\varepsilon)$ is well-conditioned.

2.2 ALGORITHM

There are several potential methods for optimizing (PT). However, the use of the log-det perturbation function lends itself well to the use of a row-action technique, namely Bregman's method (Censor and Zenios, 1997), which enforces constraints one by one yielding a simple and scalable algorithm. At each step, the algorithm makes a "Bregman projection" to enforce the chosen constraint, while simultaneously making an appropriate correction (if needed). Under mild assumptions, this method provably converges to the globally optimal solution.

The most critical aspect of the method is the Bregman-projection, which needs to be implemented efficiently for the algorithm to be practical. Below, we show how to implement this step for projecting onto a single affine equality or inequality constraint. For low-rank constraint matrices C_i, the Bregman projection for $h(\mathsf{X}) = \operatorname{Tr}(\mathsf{C}_0\mathsf{X}) - \varepsilon \log\det(\mathsf{X})$ can be performed in $O(n^2)$ time as a simple low-rank update to the current solution.

2.2.1 Projection onto a Single Constraint

Following Davis et al. (2007) we can derive the updates necessary for projecting the current solution X_t onto a single constraint (via a Bregman projection) to form X_{t+1}. Given the constraint $\operatorname{Tr}(\mathsf{C}_i\mathsf{X}_t) = b_i$, the Bregman projection to update from X_t to X_{t+1} is given by

$$\mathsf{X}_{t+1} = (\mathsf{X}_t^{-1} - (\alpha/\varepsilon)\mathsf{C}_i)^{-1}, \qquad (2.2)$$

where α is chosen so that $\operatorname{Tr}(\mathsf{C}_i\mathsf{X}_{t+1}) = b_i$. For general constraint matrices C_i, computing α and performing the update can be expensive. But when C_i is low-rank, the computation simplifies considerably. For a rank-one equality constraint of the form $\operatorname{Tr}(\mathsf{X}_t \boldsymbol{z}_i \boldsymbol{z}_i^T) = b_i$, the Bregman projection is given by

$$\mathsf{X}_{t+1} = \mathsf{X}_t + \frac{\frac{\alpha}{\varepsilon}\mathsf{X}_t \boldsymbol{z}_i \boldsymbol{z}_i^T \mathsf{X}_t}{1 - \frac{\alpha}{\varepsilon}\boldsymbol{z}_i^T \mathsf{X}_t \boldsymbol{z}_i}, \qquad (2.3)$$

which follows by applying the Sherman-Morrison-Woodbury formula to (2.2). We then solve the equation $\operatorname{Tr}(\mathsf{X}_{t+1}\boldsymbol{z}_i\boldsymbol{z}_i^T) = \boldsymbol{z}_i^T \mathsf{X}_{t+1} \boldsymbol{z}_i = b_i$ for α, which results in

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

We then use this choice of α to update X_{t+1}. After solving for α, we update X_{t+1} via (2.3); note that this update is a rank-one update, and can be performed in $O(n^2)$ time. Similar projections exist for rank-two and rank-three constraint matrices.

Algorithm 1 First-order SDP method

```
Input: $\{C_i, b_i\}_{i=1}^m$: input constraints, $C_0$: input matrix
  $\varepsilon$: tradeoff parameter
Output: $X$: output PSD matrix
 1. Initialize $X$ and $\lambda_i$ such that $X \succ 0$,
    $\lambda_i \geq 0$ for inequality constraints, and $X = \varepsilon(C_0 + \sum_i \lambda_i C_i)^{-1}$.
 2. repeat
    2.1. Pick a constraint (e.g., the most violated constraint) $(C_i, b_i)$.
    2.2. Compute projection parameter for $\alpha$.
         {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. $X \leftarrow (X - \frac{\alpha}{\varepsilon} C_i)^{-1}$.
 3. until convergence
```

If the constraint is an inequality constraint, then additionally a correction must be enforced to ensure that the corresponding dual variable remains non-negative. Let λ_i be 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 (2.3) with α' in place of α. Note that the dual variables λ_i and the starting matrix X_0 are initialized so that $\nabla h(X_0) = -\sum_i \lambda_i C_i$, $X_0 \succ 0$, 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 Censor and Zenios (1997). 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 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.

3 APPLICATIONS

We now discuss some of the machine learning applications of the algorithm. In particular, SDPs with low-rank constraint matrices are especially relevant. We also show how the kernelization of LMNN falls out naturally as a consequence of our algorithm.

3.1 EXAMPLE SDPS

Nonlinear Embedding: Semidefinite embedding (Weinberger et al., 2004) (maximum variance unfolding) is a nonlinear dimensionality reduction problem that aims to find a low-dimensional embedding of the input data so that the variance in the data is maximized while the distances among a set of nearest neighbors S are 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:

$$\begin{aligned} \max_{X \succeq 0} \quad & \mathrm{Tr}(X) \\ \text{subject to} \quad & X_{ii} + X_{jj} - 2X_{ij} = D_{ij},\ (i,j) \in S \\ & \boldsymbol{e}^T X \boldsymbol{e} = 0. \end{aligned}$$

A related problem is the robust Euclidean embedding problem (Cayton and Dasgupta, 2006), which seeks to find the closest (in vector ℓ_1-norm) Euclidean distance matrix D to a given input dissimilarity matrix D_0. Appropriate manipulation 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 (Lang, 2005) has been used for 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:

$$\begin{aligned} \min_{X \succeq 0} \quad & \mathrm{Tr}(LX) \\ \text{subject to} \quad & \mathrm{diag}(X) = \boldsymbol{e}, \quad \boldsymbol{e}^T X \boldsymbol{e} = 0. \end{aligned} \tag{3.1}$$

Metric Learning: Various metric learning algorithms have been posed as SDPs. In particular, the method of large-margin nearest neighbors (LMNN) (Weinberger et al., 2005) guarantees that distances between nearest neighbors in the same class are much smaller than distances between points in different classes. The resulting SDP has rank-3 constraints. The method of Weinberger et al. (2005) attempts to find a Mahalanobis distance matrix A such that two neighboring 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 $C_0 = \sum_{ij} \eta_{ij} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$. Then the corresponding SDP to be solved is formalized as:

$$\begin{aligned} \min_{A,\xi} \quad & \mathrm{Tr}(C_0 A) + \gamma \sum_{ijl} \eta_{ij}(1 - y_{jl})\xi_{ijl} \\ \text{subject to } & d_A(\boldsymbol{x}_i, \boldsymbol{x}_l) - d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1 - \xi_{ijl}, \\ & \xi_{ijl} \geq 0, \quad A \succeq 0. \end{aligned} \tag{3.2}$$

Collaborative Filtering: The maximum margin matrix factorization SDP for collaborative filtering (Srebro et al., 2005) 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.

3.2 KERNELIZATION OF LMNN

We now show an intriguing application of our algorithm, namely, the kernelization of LMNN metric learning. We note that another recent result (Chatpatanasiri et al., 2008) has discussed kernelization of some metric learning algorithms including LMNN, though via a different analysis independent of convex perturbations. Recall the matrix update from Algorithm 1: $\mathsf{A} \leftarrow (\mathsf{A} - (\alpha/\varepsilon)\mathsf{C}_i)^{-1}$. In the case of LMNN, we implicitly maintain $\mathsf{A} \succeq 0$ and $\xi_{ijl} \geq 0$ via our perturbation, so all constraints are of the form $d_{\mathsf{A}}(\boldsymbol{x}_i, \boldsymbol{x}_l) - d_{\mathsf{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1 - \xi_{ijl}$ and the the corresponding constraint matrices C_i are rank-three. The resulting update (after repeated applications of Sherman-Morrison) is:

$$\mathsf{A} \leftarrow \mathsf{A} + \mathsf{A}\big(\beta_1 \boldsymbol{w}_{ij}\boldsymbol{w}_{ij}^T + \beta_2 \boldsymbol{w}_{il}\boldsymbol{w}_{il}^T + \beta_3 \boldsymbol{w}_{ij}\boldsymbol{w}_{il}^T + \beta_3 \boldsymbol{w}_{il}\boldsymbol{w}_{ij}^T\big)\mathsf{A},$$

where $\boldsymbol{w}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$ and $\beta_1, \beta_2, \beta_3$ are obtained after solving for the projection parameter α, which is the root of a third-order polynomial. By multiplying the update on the left by the matrix X^T of data points ($\mathsf{X} = [\boldsymbol{x}_1 \boldsymbol{x}_2 ... \boldsymbol{x}_n]$) and on the right by X, and letting $\mathsf{K} = \mathsf{X}^T\mathsf{A}\mathsf{X}$, the update becomes

$$\mathsf{K} \leftarrow \mathsf{K} + \mathsf{K}\big(\beta_1 \boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^T + \beta_2 \boldsymbol{d}_{il}\boldsymbol{d}_{il}^T + \beta_3 \boldsymbol{d}_{ij}\boldsymbol{d}_{il}^T + \beta_3 \boldsymbol{d}_{il}\boldsymbol{d}_{ij}^T\big)\mathsf{K},$$

where $\boldsymbol{d}_{ij} = (\boldsymbol{e}_i - \boldsymbol{e}_j)$. Thus, we maintain K instead of A; note that the size of K is independent of the dimensionality of the data points. Furthermore, the slack variables are updated as $\xi_{ijl} \leftarrow \xi_{ijl} - \beta_4\gamma\xi_{ijl}^2$, where β_4 is also computed after solving the projection parameter.

The next step is to compute the initial matrix $\mathsf{K}_0 = \mathsf{X}^T\mathsf{A}_0\mathsf{X}$. Setting $\lambda_i = 0$ initially for all constraints, the resulting initialization for our method is given by $\mathsf{A}_0 = \varepsilon(\sum_{ij}\eta_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T)^{-1}$. By adding εI to the sum in the inverse to guarantee positive definiteness, we can once again repeatedly apply Sherman-Morrison-Woodbury to compute $\mathsf{X}^T\mathsf{A}_0\mathsf{X}$ using only inner products.

Finally, we must be able to compute $d_{\mathsf{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ using the matrix K returned by the optimization algorithm, and the original kernel function $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. For this, we follow the method described in Davis et al. (2007), which shows how the distance function in kernel space can be computed by unrolling the updates and maintaining an appropriate matrix of coefficients. See Davis et al. (2007) for further details.

Note that this procedure solves the perturbed SDP for LMNN *exactly*, but in kernel space. As a result, we do not sacrifice convexity—this stands in contrast to the method of Torresani and Lee (2007) that sacrifices convexity in order to kernelize LMNN.

4 EXPERIMENTS

In Section 4.1, we compare our approach to existing first-order and second-order SDP software on the min balanced cut problem to assess the viability of our algorithm for some benchmark SDPs. Here we observe empirically how close the solutions of the perturbed problems are to the unperturbed SDPs. In Section 4.2, we present results on LMNN. We demonstrate that a non-kernelized version of our method is faster than the original method of Weinberger et al. (2005) on several data sets; then we show applications of high-dimensional metric learning with our kernelized-LMNN on a computer vision task and standard UCI datasets. In all experiments, we set $\varepsilon = 10^{-1}$; we found that the solutions obtained by our algorithm are very close to the globally optimal solutions.

4.1 MIN BALANCED CUT AND MVU

In this section we show results for two machine learning SDPs—min-balanced cut and semidefinite embedding or maximum variance unfolding (MVU). These SDPs have low-rank constraints, making them appropriate for our method. We compare with SEDUMI (2007), which implements both first-order and second-order solvers. Our software and SEDUMI use 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 (reasonable) default parameters for SEDUMI.

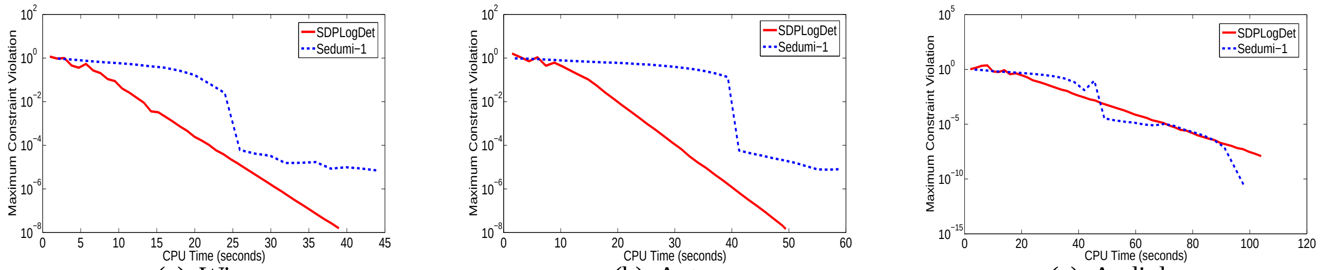
Table 1 lists accuracy results for min-balanced cut on some of the UCI data sets[3] tested over the first-order SEDUMI algorithm (SEDUMI-1), the second-order SEDUMI algorithm (SEDUMI-2), and our approach (SDPLogDet). The graphs for this SDP were constructed using the Gram matrix of the data points and scaled so that edge weights were between 0 and 1. The accuracy of our method in terms of the final objective function value is very close to that of both the first and second-order SEDUMI solvers, generally differing only in the third significant digit; 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}. These results show that our method yields reasonable approximate solutions.

For these datasets we found that SEDUMI-2 was in general the fastest, followed by SDPLogDet, and finally, SEDUMI-1. Since the convergence criteria were different among the algorithms, to compare timings we ran the algorithms until they reached a maximum violation of 10^{-8} and measured maximum violation as a function of the CPU time of the algorithms. In Figure 1, we show comparisons between SEDUMI-1 and SDPLogDet on three of the UCI data sets. SDPLogDet shows consistent linear convergence, and typically has faster convergence during early iterations as compared to SEDUMI-1. This demonstrates our method's ability to reach low to medium accuracy solutions quickly in comparison to an existing state-of-the-art first-order solver;

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

	Objective Function Value			Maximum Constraint Violation		
Data Set	SEDUMI-2	SEDUMI-1	SDPLogDet	SEDUMI-2	SEDUMI-1	SDPLogDet
Iris	1.020×10^4	1.020×10^4	1.022×10^4	2.005×10^{-11}	1.269×10^{-10}	9.633×10^{-4}
Wine	5.657×10^3	5.657×10^3	5.672×10^3	8.854×10^{-12}	3.761×10^{-12}	9.953×10^{-4}
Ionosphere	6.755×10^3	6.755×10^3	6.766×10^3	1.486×10^{-11}	1.683×10^{-5}	9.532×10^{-4}
Soybean	1.239×10^5	1.188×10^5	1.239×10^5	1.125×10^{-11}	4.143×10^{-2}	9.937×10^{-4}
Autos	3.394×10^3	3.394×10^3	3.415×10^3	4.379×10^{-12}	2.123×10^{-10}	9.851×10^{-4}
Audiology	9.709×10^3	9.709×10^3	9.729×10^3	3.633×10^{-11}	2.402×10^{-11}	9.814×10^{-4}
Breast Cancer	2.509×10^4	2.509×10^4	2.511×10^4	8.455×10^{-12}	3.050×10^{-10}	8.768×10^{-4}
Colic	2.279×10^4	2.279×10^4	2.282×10^4	7.387×10^{-11}	4.343×10^{-7}	9.796×10^{-4}
Dermatology	2.622×10^4	2.622×10^4	2.623×10^4	3.136×10^{-12}	5.259×10^{-6}	9.510×10^{-4}

Table 1: Accuracy Results on Min Balanced Cut



(a) Wine (b) Autos (c) Audiology

Figure 1: Convergence comparison between SDPLogDet and SEDUMI-1 over three example UCI data sets for the min balanced cut problem. SDPLogDet shows consistent linear convergence, while SEDUMI-1 converges less consistently and has slower convergence during early iterations.

additional speed can be gained by more algorithmic refinements, and remains a part of our ongoing efforts.

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

Table 2: UCI Data Sets

The key advantage to using a first-order SDP algorithm as opposed to a second-order method is in scalability to very large data sets. In an additional experiment, we compared memory consumption of the methods for the maximum variance unfolding problem. 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 (the number of variables to solve for in the SDP is $n(n+1)/2$); in these experiments, $k = 5$. SEDUMI-2 requires the most memory, and was unsuccessful in running on problems where n was greater than 1500. Interestingly, SEDUMI-1 also required significant memory overhead, and does not scale to problems larger than $n = 1500$. DSDP, another second-order method (Benson et al., 2000), scales to $n = 3000$ (i.e., 4.5 million variables), whereas SDPLogDet 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 even larger SDPs.

n	SEDUMI-1	SEDUMI-2	DSDP	SDPLogDet
100	7	13	3	1
500	165	222	67	5
1000	779	881	263	35
1500	1821	1930	586	52
2000	—	—	1033	92
2500	—	—	1608	144
3000	—	—	2170	207
3500	—	—	—	253

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

4.2 METRIC LEARNING EXPERIMENTS

We now compare our solver (SDPLogDet) to specialized software for the LMNN problem, which employs subgradient methods; note that SEDUMI could not be used for LMNN due to the large number of constraints. Also note that, unlike the algorithm of Weinberger et al. (2005) (or the more recent highly-tuned version (Weinberger and Saul, 2008)), we have not specialized our SDP algorithm in any way for LMNN. We test on standard UCI data sets. In all experiments, the parameter γ is tuned via cross-validation.

	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**

Table 4: 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.

We use a 70/30 training/test split, and use a k-nearest neighbor classifier ($k = 3$) for classification. Table 4 compares Weinberger et al.'s implementation of LMNN with our proposed algorithm averaged over 10 runs. The running times are in seconds, and we provide the baseline Euclidean test error (i.e., test error with no metric learning) for comparison. Our test error results are comparable to Weinberger et al., but with faster running times.

Next we consider K-LMNN, our kernelized version of LMNN. For the following experiment, the data is represented in kernelized form (no explicit vector representation). As a result, the non-kernelized LMNN algorithms cannot be applied, and so we will compare to software for information-theoretic metric learning (ITML) (Davis et al., 2007), which can be kernelized. Again, we use k-nearest neighbor classification.

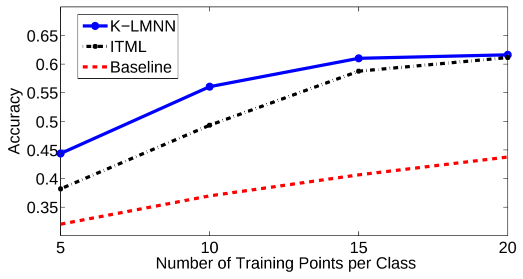


Figure 2: Comparison of nearest neighbor methods for the Caltech-101 data set. K-LMNN outperforms ITML for 5–20 training examples per class.

We learned a metric using ITML and K-LMNN for the Caltech-101 data set (Fei-Fei et al., 2004), a common benchmark for object classification in computer vision. This data set contains 3969 images in 101 categories. We set $k = 7$ for classification and, as is standard for this data set, measured performance over an increasing number of training examples per class (5, 10, 15, 20). As our initial kernel function, we used the kernel of Zhang et al. (2006) that achieves 40 percent accuracy over this data set with 15 training examples per class. Both K-LMNN and ITML (averaged over 5 runs) significantly outperform the baseline (Euclidean) k-nearest neighbor classifier, and K-LMNN performs better than ITML on average. In fact, K-LMNN seems to be competitive with the best classification results on this data set for single-kernel methods (see Zhang et al. (2006)).

Additionally, we ran K-LMNN on the UCI datasets using a Gaussian kernel. Some example test-errors are reported in Table 5. We see that for some datasets, e.g., ionosphere, using K-LMNN leads to significant improvement over non-kernelized metric learning. One would expect that applying such a kernel function would help in cases where the data is not linearly separable.

Data Set	LMNN	K-LMNN
Iris	.021	.021
Ionosphere	.119	.072
Balance-Scale	.182	.127
Breast-Cancer	.299	.271

Table 5: Metric learning test errors on UCI datasets

5 CONCLUSIONS

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. We also showed how the perturbations naturally yield kernelization of a popular metric learning algorithm. 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.

Acknowledgements This research was supported by NSF grants CCF-0431257 and CCF-072887. We thank Stefanie Jegelka for initial discussions on the results in this paper.

References

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

A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM J. Optim.*, 16(3):697–725, 2006.

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.

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.

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

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

R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. On kernelization of supervised Mahalanobis distance learners. ArXiv, 2008. `http://arxiv.org/pdf/0804.1441`.

J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object cateories. In *Workshop on Generative Model Based Vision*, Washington, D.C., June 2004.

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

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

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

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

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

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

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

B. Martinet. Régularisation d'inéquations variationelles par approximations successives. *RAIRO Rech. Opér.*, 4(R3), 1970.

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

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14, 1976.

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

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

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

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

L. Torresani and K. Lee. Large margin component analysis. In *NIPS*, 2007.

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

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

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

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

K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *ICML*, 2008.

H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

A PROOF SKETCHES

Proof. (Theorem 2.1). Let $\mathcal{A}(\overline{\boldsymbol{\nu}}) = \sum_i \overline{\nu}_i \mathsf{A}_i$, and $(\overline{\mathsf{X}}, \overline{\mathsf{Z}}, \overline{\boldsymbol{\nu}}, \gamma)$ be a KKT point of (AUX). Then $\overline{\mathsf{X}}, \overline{\mathsf{Z}} \succeq 0, \overline{\boldsymbol{\nu}} \geq 0, \gamma \geq 0$, and

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

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 $\overline{\mathsf{X}}, \mathsf{Z} \succeq 0, \boldsymbol{\nu} \geq 0$, and

$$\mathsf{C} + \mathcal{A}(\boldsymbol{\nu}) - \mathsf{Z} = 0$$
$$\mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) \leq b_i, \quad \nu_i(\mathrm{Tr}(\mathsf{A}_i\overline{\mathsf{X}}) - b_i) = 0, \quad 1 \leq i \leq m, \tag{A.2}$$

We combine (A.1) and (A.2) to get a KKT point of (PT). Consider the two cases $\gamma = 0$ and $\gamma > 0$ (γ is the dual variable corresponding to the $\mathrm{Tr}(\mathsf{C}\overline{\mathsf{X}}) \leq \bar{\theta}$ constraint):

Case 1: $\gamma = 0$. For any $\varepsilon \geq 0$, $(\overline{\mathsf{X}}, \varepsilon\overline{\mathsf{Z}} + \mathsf{Z}, \varepsilon\overline{\boldsymbol{\nu}} + \boldsymbol{\nu})$ is a KKT point of (PT). This is easily verified by multiplying (A.1) by ε and adding the result to (A.2).

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 $\overline{\boldsymbol{\nu}}^{pt} = (1-\lambda)\boldsymbol{\nu} + \frac{\lambda}{\gamma}\overline{\boldsymbol{\nu}}$. As for Case 1, this is easily verified by multiplying (A.1) by λ/γ, (A.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). $\square$

Proof. (Theorem 2.3) 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 (2.1) 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. $\square$