Nonlinear Dimensionality Reduction by Semidefinite Programming and Kernel Matrix Factorization

Kilian Q. Weinberger, Benjamin D. Packer, and Lawrence K. Saul[*]
Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389
{kilianw,lsaul,bpacker}@seas.upenn.edu

Abstract

We describe an algorithm for nonlinear dimensionality reduction based on semidefinite programming and kernel matrix factorization. The algorithm learns a kernel matrix for high dimensional data that lies on or near a low dimensional manifold. In earlier work, the kernel matrix was learned by maximizing the variance in feature space while preserving the distances and angles between nearest neighbors. In this paper, adapting recent ideas from semi-supervised learning on graphs, we show that the full kernel matrix can be very well approximated by a product of smaller matrices. Representing the kernel matrix in this way, we can reformulate the semidefinite program in terms of a much smaller submatrix of inner products between randomly chosen *landmarks*. The new framework leads to order-of-magnitude reductions in computation time and makes it possible to study much larger problems in manifold learning.

1 Introduction

A large family of graph-based algorithms has recently emerged for analyzing high dimensional data that lies or or near a low dimensional manifold [2, 5, 8, 13, 19, 21, 25]. These algorithms derive low dimensional embeddings from the top or bottom eigenvectors of specially constructed matrices. Either directly or indirectly, these matrices can be related to kernel matrices of inner products in a nonlinear feature space [9, 15, 22, 23]. These algorithms can thus be viewed as kernel methods with feature spaces that "unfold" the manifold from which the data was sampled.

In recent work [21, 22], we introduced Semidefinite Embedding (SDE), an algorithm for manifold learning based on semidefinite programming [20]. SDE learns a kernel matrix by maximizing the variance in feature space while preserving the distances and angles between nearest neighbors. It has several interesting properties: the main optimization is convex and guaranteed to preserve certain aspects of the local geometry; the method always yields a semipositive definite kernel matrix; the eigenspectrum of the kernel matrix provides an estimate of the underlying manifold's dimensionality; also, the method does not rely on estimating geodesic distances between faraway points on the manifold. This particular combination of advantages appears unique to SDE.

The main disadvantage of SDE, relative to other algorithms for manifold learning, is the time required to solve large problems in semidefinite programming. Earlier work in SDE was limited to data sets with $n \approx 2000$ examples, and problems of that size typically required several hours of computation on a mid-range desktop computer.

In this paper, we describe a new framework that has allowed us to reproduce our original results in a small fraction of this time, as well as to study much larger problems in manifold learning. We start by showing that for well-sampled manifolds, the entire kernel matrix can be very accurately reconstructed from a much smaller submatrix of inner products between randomly chosen *landmarks*. In particular, letting K denote the full $n \times n$ kernel matrix, we can write:

$$K \approx QLQ^T, \tag{1}$$

where L is the $m \times m$ submatrix of inner products between landmarks (with $m \ll n$) and Q is an $n \times m$ linear transformation derived from solving a sparse set of linear equations. The factorization in eq. (1) enables us to reformulate the semidefinite program in terms of the much smaller matrix L, yielding order-of-magnitude reductions in computation time.

[*]This work was supported by NSF Award 0238323.

The framework in this paper has several interesting connections to previous work in manifold learning and kernel methods. Landmark methods were originally developed to accelerate the multidimensional scaling procedure in Isomap [7]; they were subsequently applied to the fast embedding of sparse similarity graphs [11]. Intuitively, the methods in these papers are based on the idea of triangulation—that is, locating points in a low dimensional space based on their distances to a small set of landmarks. This idea can also be viewed as an application of the Nyström method [24, 12], which is a particular way of extrapolating a full kernel matrix from one of its sub-blocks. It is worth emphasizing that the use of landmarks in this paper is *not* based on this same intuition. SDE does not directly estimate geodesic distances between faraway inputs on the manifold, as in Isomap. As opposed to the Nyström method, our approach is better described as an adaptation of recent ideas for semi-supervised learning on graphs [1, 16, 18, 26, 27]. Our approach is somewhat novel in that we use these ideas not for transductive inference, but for computational savings in a purely *unsupervised* setting. To manage the many constraints that appear in our semidefinite programming problems, we have also adapted certain ideas from the large-scale training of support vector machines [6].

The paper is organized as follows. In section 2, we review our earlier work on manifold learning by semidefinite programming. In section 3, we investigate the kernel matrix factorization in eq. (1), deriving the linear transformation that reconstructs other examples from landmarks, and showing how it simplifies the semidefinite program for manifold learning. Section 4 gives experimental results on data sets of images and text. Finally, we conclude in section 5.

2 Semidefinite Embedding

We briefly review the algorithm for SDE; more details are given in previous work [21, 22]. As input, the algorithm takes high dimensional vectors $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$; as output, it produces low dimensional vectors $\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$. The inputs $\vec{x}_i \in \mathcal{R}^D$ are assumed to lie on or near a manifold that can be embedded in d dimensions, where typically $d \ll D$. The goal of the algorithm is to estimate the dimensionality d and to output a faithful embedding that reveals the structure of the manifold.

The main idea behind SDE has been aptly described as "maximum variance unfolding" [17]. The algorithm attempts to maximize the variance of its embedding, subject to the constraint that distances and angles between nearby inputs are preserved. The resulting transformation from inputs to outputs thus looks *locally* like a rotation plus translation—that is, it represents an isometry. To picture such a transformation from $D=3$ to $d=2$ dimensions, one can imagine a flag being unfurled by pulling on its four corners.

The first step of the algorithm is to compute the k-nearest neighbors of each input. A neighborhood-indicator matrix is defined as $\eta_{ij}=1$ if and only if the inputs $\vec{x}_i$ and $\vec{x}_j$ are k-nearest neighbors or if there exists another input of which both are k-nearest neighbors; otherwise $\eta_{ij}=0$. The constraints to preserve distances and angles between k-nearest neighbors can then be written as:

$$\|\vec{y}_i - \vec{y}_j\|^2 = \|\vec{x}_i - \vec{x}_j\|^2\,, \tag{2}$$

for all (i,j) such that $\eta_{ij}=1$. To eliminate a translational degree of freedom in the embedding, the outputs are also constrained to be centered on the origin:

$$\sum_i \vec{y}_i = \vec{0}. \tag{3}$$

Finally, the algorithm attempts to "unfold" the inputs by maximizing the variance

$$\mathrm{var}(\vec{y}) = \sum_i \|\vec{y}_i\|^2 \tag{4}$$

while preserving local distances and angles, as in eq. (2). Maximizing the variance of the embedding turns out to be a useful surrogate for minimizing its dimensionality (which is computationally less tractable).

The above optimization can be formulated as an instance of semidefinite programming [20]. Let $K_{ij} = \vec{y}_i \cdot \vec{y}_j$ denote the Gram (or kernel) matrix of the outputs. As shown in earlier work [21, 22], eqs. (2–4) can be written entirely in terms of the elements of this matrix. We can then learn the kernel matrix K by solving the following semidefinite program.

Maximize trace(K) subject to:
1) $K \succeq 0$.
2) $\Sigma_{ij} K_{ij} = 0$.
3) For all (i,j) such that $\eta_{ij}=1$,
$K_{ii} - 2K_{ij} + K_{jj} = \|\vec{x}_i - \vec{x}_j\|^2$.

As in kernel PCA [15], the embedding is derived from the eigenvalues and eigenvectors of the kernel matrix; in particular, the algorithm outputs $y_{\alpha i} = \sqrt{\lambda_\alpha}\, u_{\alpha i}$, where λ_α and u_α are the top d eigenvalues and eigenvectors. The dimensionality of the embedding, d, is suggested by the number of appreciably non-zero eigenvalues.

In sum, the algorithm has three steps: (i) computing k-nearest neighbors; (ii) computing the kernel matrix;

and (iii) computing its top eigenvectors. The computation time is typically dominated by the semidefinite program to learn the kernel matrix. In earlier work, this step limited us to problems with $n \approx 2000$ examples and $k \leq 5$ nearest neighbors; moreover, problems of this size typically required several hours of computation on a mid-range desktop computer.

3 Kernel Matrix Factorization

In practice, SDE scales poorly to large data sets because it must solve a semidefinite program over $n \times n$ matrices, where n is the number of examples. (Note that the computation time is prohibitive despite polynomial-time guarantees[1] of convergence for semidefinite programming.) In this section, we show that for well-sampled manifolds, the kernel matrix K can be approximately factored as the product of smaller matrices. We then use this representation to derive much simpler semidefinite programs for the optimization in the previous section.

3.1 Sketch of algorithm

We begin by sketching the basic argument behind the factorization in eq. (1). The argument has three steps. First, we derive a linear transformation for approximately reconstructing the entire data set of high dimensional inputs $\{\vec{x}_i\}_{i=1}^n$ from m randomly chosen inputs designated as *landmarks*. In particular, denoting these landmarks by $\{\vec{\mu}_\alpha\}_{\alpha=1}^m$, the reconstructed inputs $\{\hat{x}_i\}_{i=1}^n$ are given by the linear transformation:

$$\hat{x}_i = \sum_\alpha Q_{i\alpha}\vec{\mu}_\alpha. \tag{5}$$

The linear transformation Q is derived from a sparse weighted graph in which each node represents an input and the weights are used to propagate the positions of the m landmarks to the remaining $n-m$ nodes. The situation is analogous to semi-supervised learning on large graphs [1, 16, 18, 26, 27], where nodes represent labeled or unlabeled examples and transductive inferences are made by diffusion through the graph. In our setting, the landmarks correspond to labeled examples, the reconstructed inputs to unlabeled examples, and the vectors $\vec{\mu}_\alpha$ to the actual labels.

Next, we show that the *same linear transformation can be used to reconstruct the unfolded data set*—that is, after the mapping from inputs $\{\vec{x}_i\}_{i=1}^n$ to outputs $\{\vec{y}_i\}_{i=1}^n$. In particular, denoting the unfolded landmarks by $\{\vec{\ell}_\alpha\}_{\alpha=1}^m$ and the reconstructed outputs by $\{\hat{y}_i\}_{i=1}^n$, we argue that $\vec{y}_i \approx \hat{y}_i$, where:

$$\hat{y}_i = \sum_\alpha Q_{i\alpha}\vec{\ell}_\alpha. \tag{6}$$

The connection between eqs. (5–6) will follow from the particular construction of the weighted graph that yields the linear transformation Q. This weighted graph is derived by appealing to the symmetries of linear reconstruction coefficients; it is based on a similar intuition as the algorithm for manifold learning by locally linear embedding (LLE) [13, 14].

Finally, the kernel matrix factorization in eq. (1) follows if we make the approximation

$$K_{ij} = \vec{y}_i \cdot \vec{y}_j \approx \hat{y}_i \cdot \hat{y}_j. \tag{7}$$

In particular, substituting eq. (6) into eq. (7) gives the approximate factorization $K \approx QLQ^T$, where $L_{\alpha\beta} = \vec{\ell}_\alpha \cdot \vec{\ell}_\beta$ is the submatrix of inner products between (unfolded) landmark positions.

3.2 Reconstructing from landmarks

To derive the linear transformation Q in eqs. (5–6), we assume the high dimensional inputs $\{\vec{x}_i\}_{i=1}^n$ are well sampled from a low dimensional manifold. In the neighborhood of any point, this manifold can be locally approximated by a linear subspace. Thus, to a good approximation, we can hope to reconstruct each input by a weighted sum of its r-nearest neighbors for some small r. (The value of r is analogous but not necessarily equal to the value of k used to define neighborhoods in the previous section.) Reconstruction weights can be found by minimizing the error function:

$$\mathcal{E}(W) = \sum_i \left\| \vec{x}_i - \sum\nolimits_j W_{ij}\vec{x}_j \right\|^2, \tag{8}$$

subject to the constraint that $\sum_j W_{ij} = 1$ for all j, and where $W_{ij} = 0$ if $\vec{x}_j$ is not an r-nearest neighbor of $\vec{x}_i$. The sum constraint on the rows of W ensures that the reconstruction weights are invariant to the choice of the origin in the input space. A small regularizer for weight decay can also be added to this error function if it does not already have a unique global minimum.

Without loss of generality, we now identify the first m inputs $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$ as landmarks $\{\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_m\}$ and ask the following question: is it possible to reconstruct (at least approximately) the remaining inputs given just the landmarks $\vec{\mu}_\alpha$ and the weights W_{ij}? For sufficiently large m, a unique reconstruction can be obtained by minimizing eq. (8) with respect to $\{\vec{x}_i\}_{i>m}$.

[1] For the examples in this paper, we used the SDP solver CSDP v4.9 [4] with time complexity of $O(n^3+c^3)$ per iteration for sparse problems with $n \times n$ target matrices and c constraints. It seems, however, that large constant factors can also be associated with these complexity estimates.

To this end, we rewrite the reconstruction error as a function of the inputs, in the form:

$$\mathcal{E}(X) = \sum_{ij} \Phi_{ij}\, \vec{x}_i \cdot \vec{x}_j, \tag{9}$$

where $\Phi = (I_n - W)^T (I_n - W)$ and I_n is the $n \times n$ identity matrix. It is useful to partition the matrix Φ into blocks distinguishing the m landmarks from the other (unknown) inputs:

$$\Phi \quad = \quad \left(\begin{array}{cc} \overbrace{\quad \Phi^{\ell\ell} \quad}^{m} & \overbrace{\quad \Phi^{\ell u} \quad}^{n-m} \\ \Phi^{u\ell} & \Phi^{uu} \end{array} \right) \tag{10}$$

In terms of this matrix, the solution with minimum reconstruction error is given by the linear transformation in eq. (5), where:

$$Q = \left(\begin{array}{c} I_m \\ (\Phi^{uu})^{-1} \Phi^{ul} \end{array} \right). \tag{11}$$

An example of this minimum error reconstruction is shown in Fig. 1. The first two panels show $n = 10000$ inputs sampled from a Swiss roll and their approximate reconstructions from eq. (5) and eq. (11) using $r = 12$ nearest neighbors and $m = 40$ landmarks.

Intuitively, we can imagine the matrix Φ_{ij} in eq. (9) as defining a sparse weighted graph connecting nearby inputs. The linear transformation reconstructing inputs from landmarks is then analogous to the manner in which many semi-supervised algorithms on graphs propagate information from labeled to unlabeled examples.

To justify eq. (6), we now imagine that the data set has been unfolded in a way that preserves distances and angles between nearby inputs. As noted in previous work [13, 14], the weights W_{ij} that minimize the reconstruction error in eq. (8) are invariant to translations and rotations of each input and its r-nearest neighbors. Thus, roughly speaking, if the unfolding looks locally like a rotation plus translation, then the same weights W_{ij} that reconstruct the inputs $\vec{x}_i$ from their neighbors should also reconstruct the outputs $\vec{y}_i$ from theirs. This line of reasoning yields eq. (6). It also suggests that if we could somehow learn to faithfully embed just the landmarks in a lower dimensional space, the remainder of the inputs could be unfolded by a simple matrix multiplication.

3.3 Embedding the landmarks

It is straightforward to reformulate the semidefinite program (SDP) for the kernel matrix $K_{ij} = \vec{y}_i \cdot \vec{y}_j$ in section 2 in terms of the smaller matrix $L_{\alpha\beta} = \vec{\ell}_\alpha \cdot \vec{\ell}_\beta$. In particular, appealing to the factorization $K \approx QLQ^T$, we consider the following SDP:

Maximize trace(QLQ^T) subject to:
1) $L \succeq 0$.
2) $\Sigma_{ij}(QLQ^T)_{ij} = 0$.
3) For all (i,j) **such that** $\eta_{ij} = 1$,
$(QLQ^T)_{ii} - 2(QLQ^T)_{ij} + (QLQ^T)_{jj} \leq \|\vec{x}_i - \vec{x}_j\|^2$.

This optimization is nearly but not quite identical to the previous SDP up to the substitution $K \approx QLQ^T$. The only difference is that we have changed the equality constraints in eq. (2) to inequalities. The SDP in section 2 is guaranteed to be feasible since all the constraints are satisfied by taking $K_{ij} = \vec{x}_i \cdot \vec{x}_j$ (assuming the inputs are centered on the origin). Because the matrix factorization in eq. (1) is only approximate, however, here we must relax the distance constraints to preserve feasibility. Changing the equalities to inequalities is the simplest possible relaxation; the trivial solution $L_{\alpha\beta} = 0$ then provides a guarantee of feasibility. In practice, this relaxation does not appear to change the solutions of the SDP in a significant way; the variance maximization inherent to the objective function tends to saturate the pairwise distance constraints, even if they are not enforced as strict equalities.

To summarize, the overall procedure for unfolding the inputs $\vec{x}_i$ based on the kernel matrix factorization in eq. (1) is as follows: (i) compute reconstruction weights W_{ij} that minimize the error function in eq. (8); (ii) choose landmarks and compute the linear transformation Q in eq. (11); (iii) solve the SDP for the landmark kernel matrix L; (iv) derive a low dimensional embedding for the landmarks $\vec{\ell}_\alpha$ from the eigenvectors and eigenvalues of L; and (v) reconstruct the outputs $\vec{y}_i$ from eq. (6). The free parameters of the algorithm are the number of nearest neighbors r used to derive locally linear reconstructions, the number of nearest neighbors k used to generate distance constraints in the SDP, and the number of landmarks m (which also constrains the rank of the kernel matrix). In what follows, we will refer to this algorithm as landmark SDE, or simply ℓSDE.

ℓSDE can be much faster than SDE because its main optimization is performed over $m \times m$ matrices, where $m \ll n$. The computation time in semidefinite programming, however, depends not only on the matrix size, but also on the number of constraints. An apparent difficulty is that SDE and ℓSDE have the same number of constraints; moreover, the constraints in the latter are not sparse, so that a naive implementation of ℓSDE can actually be much slower than SDE. This difficulty is surmounted in practice by solving the semidefinite program for ℓSDE while only explicitly monitoring a small fraction of the original constraints. To start, we feed an initial subset of constraints to

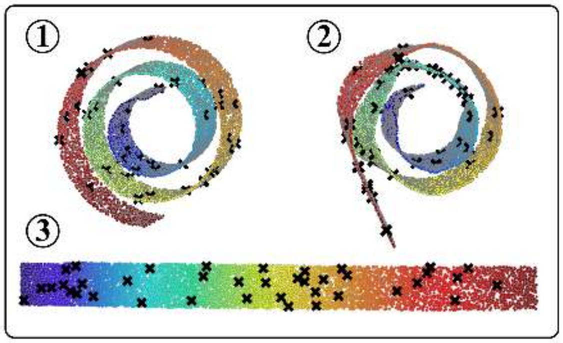

Figure 1: (1) $n = 10000$ inputs sampled from a Swiss roll; (2) linear reconstruction from $r = 12$ nearest neighbors and $m = 40$ landmarks (denoted by black x's); (3) embedding from ℓSDE, with distance and angle constraints to $k = 4$ nearest neighbors, computed in 16 minutes.

the SDP solver, consisting only of the semidefiniteness constraint, the centering constraint, and the distance constraints between landmarks and their nearest neighbors. If a solution is then found that violates some of the unmonitored constraints, these are added to the problem, which is solved again. The process is repeated until all the constraints are satisfied. Note that this incremental scheme is made possible by the relaxation of the distance constraints from equalities to inequalities. As in the large-scale training of support vector machines [6], it seems that many of the constraints in ℓSDE are redundant, and simple heuristics to prune these constraints can yield order-of-magnitude speedups. (Note, however, that the centering and semidefiniteness constraints in ℓSDE are always enforced.)

4 Experimental Results

Experiments were performed in MATLAB to evaluate the performance of ℓSDE on various data sets. The SDPs were solved with the CSDP (v4.9) optimization toolbox [4]. Of particular concern was the speed and accuracy of ℓSDE relative to earlier implementations of SDE.

The first data set, shown in the top left panel of Fig. 1, consisted of $n = 10000$ inputs sampled from a three dimensional "Swiss roll". The other panels of Fig. 1 show the input reconstruction from $m = 40$ landmarks and $r = 12$ nearest neighbors, as well as the embeddingobtained in ℓSDE by constraining distances and angles to $k = 4$ nearest neighbors. The computation took 16 minutes on a mid-range desktop computer. Table 2 shows that only 1205 out of 43182 constraints had to be explicitly enforced by the SDP solver to find a feasible solution. Interestingly, similarly faithful embeddings were obtained in shorter times using as few as $m = 10$ landmarks, though the input reconstructions in these cases were of considerably worse quality. Also worth mentioning is that adding low variance Gaussian noise to the inputs had no significant impact on the algorithm's performance.

word	**four nearest neighbors**
one	two, three, four, six
may	won't, cannot, would, will
men	passengers, soldiers, officers, lawmakers
iraq	states, israel, china, noriega
drugs	computers, missiles, equipment, programs
january	july, october, august, march
germany	canada, africa, arabia, marks
recession	environment, yen, season, afternoon
california	minnesota, arizona, florida, georgia
republican	democratic, strong, conservative, phone
government	pentagon, airline, army, bush

Table 1: Selected words and their four nearest neighbors (in order of increasing distance) after nonlinear dimensionality reduction by ℓSDE. The $d = 5$ dimensional embedding of $D = 60000$ dimensional bigram distributions was computed by ℓSDE in 35 minutes (with $n = 2000$, $k = 4$, $r = 12$, and $m = 30$).

The second data set was created from the $n = 2000$ most common words in the ARPA North American Business News corpus. Each of these words was represented by its discrete probability distribution over the $D = 60000$ words that could possibly follow it. The distributions were estimated from a maximum likelihood bigram model. The embedding of these high dimensional distributions was performed by ℓSDE (with $k = 4$, $r = 12$, and $m = 30$) in about 35 minutes; the variance of the embedding, as revealed by the eigenvalue spectrum of the landmark kernel matrix, was essentially confined to $d = 5$ dimensions. Table 1 shows a selection of words and their four nearest neighbors in the low dimensional embedding. Despite the massive dimensionality reduction from $D = 60000$ to $d = 5$, many semantically meaningful neighborhoods are seen to be preserved.

The third experiment was performed on $n = 400$ color images of a teapot viewed from different angles in the plane. Each vectorized image had a dimensionality of $D = 23028$, resulting from 3 bytes of color information for each of 76×101 pixels. In previous work [22] it was shown that SDE represents the angular mode of variability in this data set by an almost perfect circle. Fig. 2 compares embeddings from ℓSDE ($k = 4$, $r = 12$, $m = 20$) with normal SDE ($k = 4$) and LLE ($r = 12$). The eigenvalue spectrum of ℓSDE is very

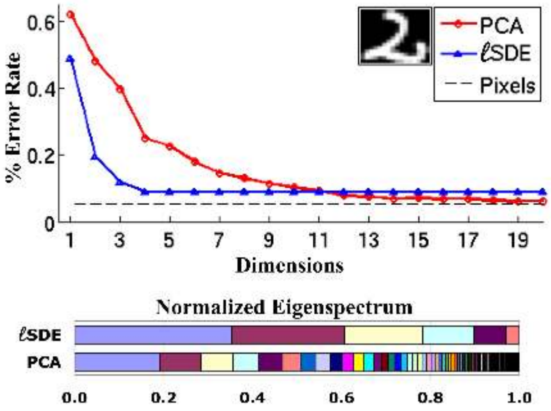

Figure 3: *Top:* Error rate of five-nearest-neighbors classification on the test set of USPS handwritten digits. The error rate is plotted against the dimensionality of embeddings from PCA and ℓSDE (with $k=4$, $r=12$, $m=10$). It can be seen that ℓSDE preserves the neighborhood structure of the digits fairly well with only a few dimensions. *Bottom:* Normalized eigenvalue spectra from ℓSDE and PCA. The latter reveals many more dimensions with appreciable variance.

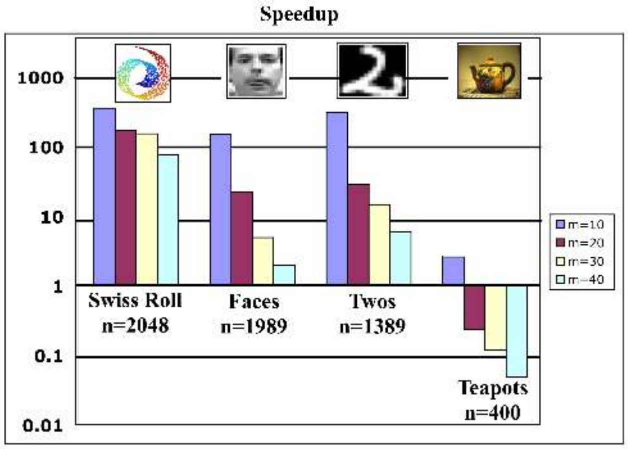

Figure 4: Relative speedup of ℓSDE versus SDE on data sets with different numbers of examples (n) and landmarks (m). Speedups of two orders of magnitude are observed on larger data sets. On small data sets, however, SDE can be faster than ℓSDE.

similar to that of SDE, revealing that the variance of the embedding is concentrated in two dimensions. The results from ℓSDE do not exactly reproduce the results from SDE on this data set, but the difference becomes smaller with increasing number of landmarks (at the expense of more computation time). Actually, as shown in Fig. 4, ℓSDE (which took 79 seconds) is slower than SDE on this particular data set. The increase in computation time has two simple explanations that seem peculiar to this data set. First, this data set is rather small, and ℓSDE incurs some overhead in its setup that is only negligible for large data sets. Second, this data set of images has a particular cyclic structure that is easily "broken" if the monitored constraints are not sampled evenly. Thus, this particular data set is not well-suited to the incremental scheme for adding unenforced constraints in ℓSDE; a large number of SDP reruns are required, resulting in a longer overall computation time than SDE. (See Table 2.)

The final experiment was performed on the entire data set of $n=9298$ USPS handwritten digits [10]. The inputs were 16×16 pixel grayscale images of the scanned digits. Table 2 shows that only 690 out of 61735 inequality constraints needed to be explicitly monitored by the SDP solver for ℓSDE to find a feasible solution. This made it possible to obtain an embedding in 40 minutes (with $k=4$, $r=12$, $m=10$), whereas earlier implementations of SDE could not handle problems of this size. To evaluate the embeddings from ℓSDE, we compared their nearest neighbor classification error rates to those of PCA. The top plot in Fig. 3 shows the classification error rate (using five nearest neighbors in the training images to classify test images) versus the dimensionality of the embeddings from ℓSDE and PCA. The error rate from ℓSDE drops very rapidly with dimensionality, nearly matching the error rate on the actual images with only $d=3$ dimensions. By contrast, PCA requires $d=12$ dimensions to overtake the performace of ℓSDE. The bar plot at the bottom of Fig. 3 shows the normalized eigenvalue spectra from both ℓSDE and PCA. From this plot, it is clear that ℓSDE concentrates the variance of its embedding in many fewer dimensions than PCA.

When does ℓSDE outperform SDE? Figure 4 shows the speedup of ℓSDE versus SDE on several data sets. Not surprisingly, the relative speedup grows in proportion with the size of the data set. Small data sets (with $n<500$) can generally be unfolded faster by SDE, while larger data sets (with $500<n<2000$) can be unfolded up to 400 times faster by ℓSDE. For even larger data sets, only ℓSDE remains a viable option.

5 Conclusion

In this paper, we have developed a much faster algorithm for manifold learning by semidefinite programming. There are many aspects of the algorithm that we are still investigating, including the interplay between the number and placement of landmarks, the definition of local neighborhoods, and the quality of

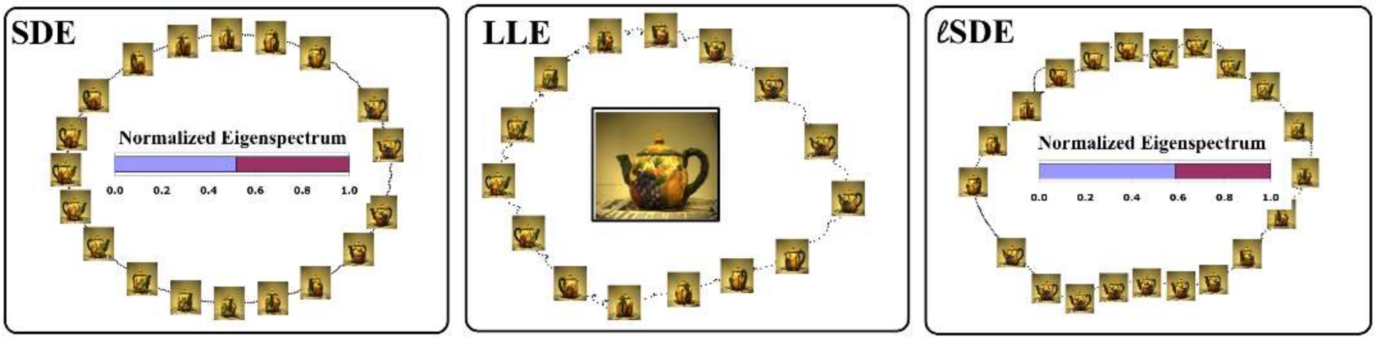

Figure 2: Comparison of embeddings from SDE, LLE and ℓSDE for $n = 400$ color images of a rotating teapot. The vectorized images had dimension $D = 23028$. LLE (with $r = 12$) and ℓSDE (with $k = 4$, $r = 12$, $m = 20$) yield similar but slightly more irregular results than SDE (with $k = 4$). The normalized eigenspectra in SDE and ℓSDE (i.e., the eigenspectra divided by the trace of their kernel matrices) reveal that the variances of their embeddings are concentrated in two dimensions; the eigenspectrum from LLE does not reveal this sort of information.

data set	n	m	constraints	monitored	time (secs)
teapots	400	20	1599	565	79
bigrams	2000	30	11170	1396	2103
USPS digits	9298	10	61735	690	2420
Swiss roll	10000	20	43182	1205	968

Table 2: Total number of constraints versus number of constraints explicitly monitored by the SDP solver for ℓSDE on several data sets. The numbers of inputs (n) and landmarks (m) are also shown, along with computation times. The speedup of ℓSDE is largely derived from omitting redundant constraints.

the resulting reconstructions and embeddings. Nevertheless, our initial results are promising and show that manifold learning by semidefinite programming can scale to much larger data sets than we originally imagined in earlier work [21, 22].

Beyond the practical applications of ℓSDE, the framework in this paper is interesting in the way it combines ideas from several different lines of recent work. ℓSDE is based on the same appeals to symmetry at the heart of LLE [13, 14] and SDE [21, 22]. The linear reconstructions that yield the factorization of the kernel matrix in eq. (1) are also reminiscent of semi-supervised algorithms for propagating labeled information through large graphs of unlabeled examples [1, 16, 18, 26, 27]. Finally, though based on a somewhat different intuition, the computational gains of ℓSDE are similar to those obtained by landmark methods for Isomap [7].

While we have applied ℓSDE (in minutes) to data sets with as many as $n = 10000$ examples, there exist many larger data sets for which the algorithm remains impractical. Further insights are therefore required. In related work, we have developed a simple out-of-sample extension for SDE, analogous to similar extensions for other spectral methods [3]. Algorithmic advances may also emerge from the dual formulation of "maximum variance unfolding" [17], which is related to the problem of computing fastest mixing Markov chains on graphs. We are hopeful that a combination of complementary approaches will lead to even faster and more powerful algorithms for manifold learning by semidefinite programming.

Acknowledgments

We are grateful to Ali Jadbabaie (University of Pennsylvania) for several discussions about semidefinite programming and to the anonymous reviewers for many useful comments.

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory (COLT 2004)*, pages 624–638, Banff, Canada, 2004.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] Y. Bengio, J-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[4] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software 11(1):613-623*, 1999.

[5] M. Brand. Charting a manifold. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 985–992, Cambridge, MA, 2003. MIT Press.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[7] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728, Cambridge, MA, 2003. MIT Press.

[8] D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003.

[9] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 369–376, Banff, Canada, 2004.

[10] J. J. Hull. A database for handwritten text recognition research. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[11] J. C. Platt. Fast embedding of sparse similarity graphs. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[12] J. C. Platt. FastMap, MetricMap, and landmark MDS are all nyström algorithms. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, WI, January 2005.

[13] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[14] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[15] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[16] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and Kernel Workshop*, Washington D.C., 2003.

[17] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, submitted.

[18] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[20] L. Vandenberghe and S. P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

[21] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, volume 2, pages 988–995, Washington D.C., 2004.

[22] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.

[23] C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 675–681, Cambridge, MA, 2001. MIT Press.

[24] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA, 2001. MIT Press.

[25] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction by local tangent space alignment. *SIAM Journal of Scientific Computing*, in press.

[26] D. Zhou, O. Bousquet, T. N. Lai, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 321–328, Cambridge, MA, 2004. MIT Press.

[27] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 912–919, Washington D.C., 2003.