Learning a Kernel Matrix for Nonlinear Dimensionality Reduction

Kilian Q. Weinberger KILIANW@CIS.UPENN.EDU
Fei Sha FEISHA@CIS.UPENN.EDU
Lawrence K. Saul LSAUL@CIS.UPENN.EDU
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA

Abstract

We investigate how to learn a kernel matrix for high dimensional data that lies on or near a low dimensional manifold. Noting that the kernel matrix implicitly maps the data into a nonlinear feature space, we show how to discover a mapping that "unfolds" the underlying manifold from which the data was sampled. The kernel matrix is constructed by maximizing the variance in feature space subject to local constraints that preserve the angles and distances between nearest neighbors. The main optimization involves an instance of semidefinite programming—a fundamentally different computation than previous algorithms for manifold learning, such as Isomap and locally linear embedding. The optimized kernels perform better than polynomial and Gaussian kernels for problems in manifold learning, but worse for problems in large margin classification. We explain these results in terms of the geometric properties of different kernels and comment on various interpretations of other manifold learning algorithms as kernel methods.

1. Introduction

Kernel methods (Schölkopf & Smola, 2002) have proven to be extremely powerful in many areas of machine learning. The so-called "kernel trick" is by now widely appreciated: a canonical algorithm (e.g., the linear perceptron, principal component analysis) is reformulated in terms of Gram matrices, then generalized to nonlinear problems by substituting a kernel function for the inner product. Well beyond this familiar recipe, however, the field continues to develop as researchers devise novel types of kernels, exploiting prior knowledge in particular domains and insights from computational learning theory and convex optimization. Indeed, much work revolves around the simple question: *"How to choose the kernel?"* The answers are diverse, reflecting the tremendous variety of problems to which kernel methods have been applied. Kernels based on string matching (Lodhi et al., 2004) and weighted transducers (Cortes et al., 2003) have been proposed for problems in bioinformatics, text, and speech processing. Other specialized kernels have been constructed for problems in pattern recognition involving symmetries and invariances (Burges, 1999). Most recently, kernel matrices have been learned by semidefinite programming for large margin classification (Graepel, 2002; Lanckriet et al., 2004).

In this paper, we revisit the problem of nonlinear dimensionality reduction and its solution by kernel principal component analysis (PCA) (Schölkopf et al., 1998). Our specific interest lies in the application of kernel PCA to high dimensional data whose basic modes of variability are described by a low dimensional manifold. The goal of nonlinear dimensionality reduction in these applications is to discover the underlying manifold (Tenenbaum et al., 2000; Roweis & Saul, 2000). For problems of this nature, we show how to learn a kernel matrix whose implicit mapping into feature space "unfolds" the manifold from which the data was sampled. The main optimization of our algorithm involves an instance of semidefinite programming, but unlike earlier work in learning kernel matrices (Graepel, 2002; Lanckriet et al., 2004), the setting here is completely unsupervised.

The problem of manifold learning has recently attracted a great deal of attention (Tenenbaum et al., 2000; Roweis & Saul, 2000; Belkin & Niyogi, 2003; Saul & Roweis, 2003), and a number of authors (Bengio et al., 2004; Ham et al., 2004) have developed connections between manifold learning algorithms and kernel PCA. In contrast to previous work, however,

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

our paper does not serve to reinterpret pre-existing algorithms such as Isomap and locally linear embedding as instances of kernel PCA. Instead, we propose a novel optimization (based on semidefinite programming) that bridges the literature on kernel methods and manifold learning in a rather different way. The algorithm we describe can be viewed from several complementary perspectives. This paper focuses mainly on its interpretation as a kernel method, while a companion paper (Weinberger & Saul, 2004) focuses on its application to the unsupervised learning of image manifolds.

2. Kernel PCA

Schölkopf, Smola, and Müller (1998) introduced kernel PCA as a nonlinear generalization of PCA (Jolliffe, 1986). The generalization is obtained by mapping the original inputs into a higher (and possibly infinite) dimensional feature space $\mathcal{F}$ before extracting the principal components. In particular, consider inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{R}^D$ and features $\mathbf{\Phi}(\mathbf{x}_1), \ldots, \mathbf{\Phi}(\mathbf{x}_N) \in \mathcal{F}$ computed by some mapping $\mathbf{\Phi} : \mathcal{R}^D \rightarrow \mathcal{F}$. Kernel PCA is based on the insight that the principal components in $\mathcal{F}$ can be computed for mappings $\mathbf{\Phi}(\mathbf{x})$ that are only implicitly defined by specifying the inner product in feature space—that is, the kernel function $K(\mathbf{x}, \mathbf{y}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y})$.

Kernel PCA can be used to obtain low dimensional representations of high dimensional inputs. For this, it suffices to compute the dominant eigenvectors of the kernel matrix $K_{ij} = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$. The kernel matrix can be expressed in terms of its eigenvalues λ_α and eigenvectors $\mathbf{v}_\alpha$ as $K = \sum_\alpha \lambda_\alpha \mathbf{v}_\alpha \mathbf{v}_\alpha^{\mathrm{T}}$. Assuming the eigenvalues are sorted from largest to smallest, the d-dimensional embedding that best preserves inner products in feature space is obtained by mapping the input $\mathbf{x}_i \in \mathcal{R}^D$ to the vector $\mathbf{y}_i = (\sqrt{\lambda_1} v_{1i}, \ldots, \sqrt{\lambda_d} v_{di})$.

The main freedom in kernel PCA lies in choosing the kernel function $K(\mathbf{x}, \mathbf{y})$ or otherwise specifying the kernel matrix K_{ij}. Some widely used kernels are the linear, polynomial and Gaussian kernels, given by:

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}, \qquad (1)$$

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^p, \qquad (2)$$

$$K(\mathbf{x}, \mathbf{y}) = e^{\frac{-|\mathbf{x}-\mathbf{y}|^2}{2\sigma^2}}. \qquad (3)$$

The linear kernel simply identifies the feature space with the input space. Implicitly, the polynomial kernel maps the inputs into a feature space of dimensionality $O(D^p)$, while the Gaussian kernel maps the inputs onto the surface of an infinite-dimensional sphere.

The dominant eigenvalues of the kernel matrix K_{ij} measure the variance along the principal components in feature space, provided that the features are centered on the origin. The features can always be centered by subtracting out their mean—namely, by the transformation $\mathbf{\Phi}(\mathbf{x}_i) \leftarrow \mathbf{\Phi}(\mathbf{x}_i) - \frac{1}{N}\sum_j \mathbf{\Phi}(\mathbf{x}_j)$. When the mapping $\mathbf{\Phi}(\mathbf{x})$ is only implicitly specified by the kernel function, the "centering" transformation can be applied directly to the kernel matrix. In particular, recomputing the inner products $K_{ij} = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$ from the centered features gives:

$$K_{ij} \leftarrow K_{ij} - \frac{2}{N}\sum_k K_{kj} + \frac{1}{N^2}\sum_{k\ell} K_{k\ell}. \qquad (4)$$

For a centered kernel matrix, the relative weight of the leading d eigenvalues, obtained by dividing their sum by the trace, measures the relative variance captured by the leading d eigenvectors. When this ratio is nearly unity, the data can be viewed as inhabiting a d-dimensional subspace of the feature space, or equivalently, a d-dimensional manifold of the input space.

3. Learning the kernel matrix

The choice of the kernel plays an important role in kernel PCA, in that different kernels are bound to reveal (or conceal) different types of low dimensional structure. In this section, we show how to learn a kernel matrix that reveals when high dimensional inputs lie on or near a low dimensional manifold. As in earlier work on support vector machines (SVMs) (Graepel, 2002; Lanckriet et al., 2004), we will cast the problem of learning the kernel matrix as an instance of semidefinite programming. The similarity ends there, however, as the optimization criteria for nonlinear dimensionality reduction differ substantially from the criteria for large margin classification. We describe the constraints on the optimization in section 3.1, the objective function in section 3.2, and the optimization itself in section 3.3.

3.1. Constraints

Semipositive definiteness
The kernel matrix K is constrained by three criteria. The first is semipositive definiteness, a condition required to interpret the kernel matrix as storing the inner products of vectors in a Hilbert space. We thus constrain the optimization over K to the cone of symmetric matrices with nonnegative eigenvalues. Though not a linear constraint, the cone of semipositive definite matrices defines a *convex* domain for the overall optimization.

Centering
The second constraint is that the kernel matrix stores

the inner products of features that are centered on the origin, or:

$$\sum_i \mathbf{\Phi}(\mathbf{x}_i) = 0 \tag{5}$$

As described in section 2, this condition enables us to interpret the eigenvalues of the kernel matrix as measures of variance along principal components in feature space. Eq. (5) can be expressed in terms of the kernel matrix as:

$$0 = \left|\sum_i \mathbf{\Phi}(\mathbf{x}_i)\right|^2 = \sum_{ij} \mathbf{\Phi}(\mathbf{x}_i)\cdot\mathbf{\Phi}(\mathbf{x}_j) = \sum_{ij} K_{ij}. \tag{6}$$

Note that this is a linear constraint on the elements of the kernel matrix, thus preserving the convexity of the domain of optimization.

Isometry
The final constraints on the kernel matrix reflect our goals for nonlinear dimensionality reduction. In particular, we are interested in the setting where the inputs lie on or near a low dimensional manifold, and the goals of kernel PCA are to detect the dimensionality of this underlying manifold and discover its modes of variability. We imagine that this manifold is isometric to an open connected subset of Euclidean space (Tenenbaum et al., 2000; Donoho & Grimes, 2003), and the problem of learning the kernel matrix is to discover how the inner products between inputs transform under this mapping. An isometry[1] is a smooth invertible mapping that looks locally like a rotation plus translation, thus preserving local (and hence geodesic) distances. Thus, in our application of kernel PCA to manifold learning, the final constraints we impose on the kernel matrix are to restrict the (implicitly defined) mappings between inputs and features from fully general nonlinear transformations to the special class of isometries.

How is this done? We begin by extending the notion of isometry to *discretely sampled* manifolds, in particular to sets of inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and features $\{\mathbf{\Phi}(\mathbf{x}_1), \ldots, \mathbf{\Phi}(\mathbf{x}_N)\}$ in one-to-one correspondence. Let the $N\times N$ binary matrix η indicate a neighborhood relation on both sets, such that we regard $\mathbf{x}_j$ as a neighbor of $\mathbf{x}_i$ if and only if $\eta_{ij}=1$ (and similarly, for $\mathbf{\Phi}(\mathbf{x}_j)$ and $\mathbf{\Phi}(\mathbf{x}_i)$). We will say that the inputs $\mathbf{x}_i$ and features $\mathbf{\Phi}(x_i)$ are *locally isometric* under the neighborhood relation η if for every point $\mathbf{x}_i$, there exists a rotation and translation that maps $\mathbf{x}_i$ and its neighbors precisely onto $\mathbf{\Phi}(x_i)$ and its neighbors.

The above definition translates naturally into various sets of linear constraints on the elements of the kernel matrix K_{ij}. Note that the local isometry between neighborhoods will exist if and only if the distances and angles between points and their neighbors are preserved. Thus, whenever both $\mathbf{x}_j$ and $\mathbf{x}_k$ are neighbors of $\mathbf{x}_i$ (that is, $\eta_{ij}\eta_{ik}=1$), for local isometry we must have that:

$$(\mathbf{\Phi}(\mathbf{x}_i)-\mathbf{\Phi}(\mathbf{x}_j))\cdot(\mathbf{\Phi}(\mathbf{x}_i)-\mathbf{\Phi}(\mathbf{x}_k)) = (\mathbf{x}_i-\mathbf{x}_j)\cdot(\mathbf{x}_i-\mathbf{x}_k)\,. \tag{7}$$

Eq. (7) is sufficient for local isometry because the triangle formed by any point and its neighbors is determined up to rotation and translation by specifying the lengths of two sides and the angle between them. In fact, such a triangle is similarly determined by specifying the lengths of all its sides. Thus, we can also say that the inputs and features are locally isometric under η if whenever $\mathbf{x}_i$ and $\mathbf{x}_j$ are themselves neighbors (that is, $\eta_{ij}=1$) or are common neighbors of another input (that is, $[\eta^T\eta]_{ij}>0$), we have:

$$|\mathbf{\Phi}(\mathbf{x}_i)-\mathbf{\Phi}(\mathbf{x}_j)|^2 = |\mathbf{x}_i-\mathbf{x}_j|^2\,. \tag{8}$$

This is an equivalent characterization of local isometry as eq. (7), but expressed only in terms of pairwise distances. Finally, we can express these constraints purely in terms of dot products. Let $G_{ij}=\mathbf{x}_i\cdot\mathbf{x}_j$ denote the Gram matrix of the inputs, and recall that the kernel matrix $K_{ij}=\mathbf{\Phi}(\mathbf{x}_i)\cdot\mathbf{\Phi}(\mathbf{x}_j)$ represents the Gram matrix of the features. Then eq. (8) can be written as:

$$K_{ii}+K_{jj}-K_{ij}-K_{ji} = G_{ii}+G_{jj}-G_{ij}-G_{ji}. \tag{9}$$

Eq. (9) constrains how the inner products between nearby inputs are allowed to transform under a locally isometric mapping. We impose these constraints to ensure that the mapping defined (implicitly) by the kernel matrix is an isometry. Note that these constraints are also linear in the elements of the kernel matrix, thus preserving the convexity of the domain of optimization.

The simplest choice for neighborhoods is to let η_{ij} indicate whether input $\mathbf{x}_j$ is one of the k nearest neighbors of input $\mathbf{x}_i$ computed using Euclidean distance. In this case, eq. (9) specifies $O(Nk^2)$ constraints that fix the distances between each point and its nearest neighbors, as well as the pairwise distances between nearest neighbors. Provided that $k \ll N$, however, the kernel matrix is unlikely to be fully specified by these constraints since it contains $O(N^2)$ elements.

3.2. Objective function

In the previous section, we showed how to restrict the kernel matrices so that the features $\mathbf{\Phi}(\mathbf{x}_i)$ could be regarded as images of a locally isometric mapping. The

[1] Formally, two Riemannian manifolds are said to be isometric if there is a diffeomorphism such that the metric on one pulls back to the metric on the other.

goal of nonlinear dimensionality reduction is to discover the *particular* isometric mapping that "unfolds" the underlying manifold of inputs into a Euclidean space of its intrinsic dimensionality. This intrinsic dimensionality may be much lower than the extrinsic dimensionality of the input space. To unfold the manifold, we need to construct an objective function over "locally isometric" kernel matrices that favors this type of dimensionality reduction.

To this end, imagine each input $\mathbf{x}_i$ as a steel ball connected to its k nearest neighbors by rigid rods. (For simplicity, we assume that the graph formed this way is fully connected; if not, then each connected component of the graph should be analyzed separately.) The effect of the rigid rods is to "lock" the neighborhoods in place, fixing the distances and angles between nearest neighbors. The lattice of steel balls formed in this way can be viewed as a discretized version of the underlying manifold. Now imagine that we pull the steel balls as far apart as possible, recording their final positions by $\mathbf{\Phi}(\mathbf{x}_i)$. The discretized manifold will remain connected—due to the constraints imposed by the rigid rods—but it will also flatten, increasing the variance captured by its leading principal components. (For a continuous analogy, imagine pulling on the ends of a string; a string with any slack in it occupies at least two dimensions, whereas a taut, fully extended string occupies just one.)

We can formalize this intuition as an optimization over semipositive definite matrices. The constraints imposed by the rigid rods are, in fact, precisely the constraints imposed by eq. (9). An objective function that measures the pairwise distances between steel balls is given by:

$$\mathcal{T} = \frac{1}{2N}\sum_{ij}|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)|^2 \qquad (10)$$

It is easy to see that this function is bounded above due to the constraints on distances between neighbors imposed by the rigid rods. Suppose the distance between any two neighbors is bounded by some maximal distance τ. Providing the graph is connected, then for any two points, there exists a path along the graph of distance at most $N\tau$, which (by the triangle inequality) provides an upper bound on the Euclidean distance between the points that appears in eq. (10). This results in an upper bound on the objective function of order $O(N^3\tau^2)$.

Eq. (10) can be expressed in terms of the elements of the kernel matrix by expanding the right hand side:

$$\mathcal{T} \;=\; \frac{1}{2N}\sum_{ij}(K_{ii} + K_{jj} - 2K_{ij}) \;=\; \mathrm{Tr}(K). \qquad (11)$$

The last step in eq. (11) follows from the centering constraint in eq. (6). Thus the objective function for the optimization is simply the trace of the kernel matrix. Maximizing the trace also corresponds to maximizing the variance in feature space.

3.3. Semidefinite embedding (SDE)

The constraints and objective function from the previous sections define an instance of semidefinite programming (SDP) (Vandenberghe & Boyd, 1996). Specifically, the goal is to optimize a linear function of the elements in a semipositive definite matrix subject to linear equality constraints. Collecting the constraints and objective function, we have the following optimization:

Maximize: $\mathrm{Tr}(K)$ **subject to:**

1. $K \succeq 0$.
2. $\sum_{ij} K_{ij} = 0$.
3. $K_{ii}+K_{jj}-K_{ij}-K_{ji} \;=\; G_{ii}+G_{jj}-G_{ij}-G_{ji}$ **for all** i, j **such that** $\eta_{ij}=1$ **or** $\left[\eta^T\eta\right]_{ij}>0$**.**

The optimization is convex and does not suffer from local optima. There are several general-purpose toolboxes and polynomial-time solvers available for problems in semidefinite programming. The results in this paper were obtained using the SeDuMi toolbox (Sturm, 1999) in MATLAB. Once the kernel matrix is computed, a nonlinear embedding can be obtained from its leading eigenvectors, as described in section 2. Because the kernel matrices in this approach are optimized by semidefinite programming, we will refer to this particular form of kernel PCA as *Semidefinite Embedding* (SDE).

4. Experimental Results

Experiments were performed on several data sets to evaluate the learning algorithm described in section 3. Though the SDE kernels were expressly optimized for problems in manifold learning, we also evaluated their performance for large margin classification.

4.1. Nonlinear dimensionality reduction

We performed kernel PCA with linear, polynomial, Gaussian, and SDE kernels on data sets where we knew or suspected that the high dimensional inputs were sampled from a low dimensional manifold. Where necessary, kernel matrices were centered before computing principal components, as in eq. (4).

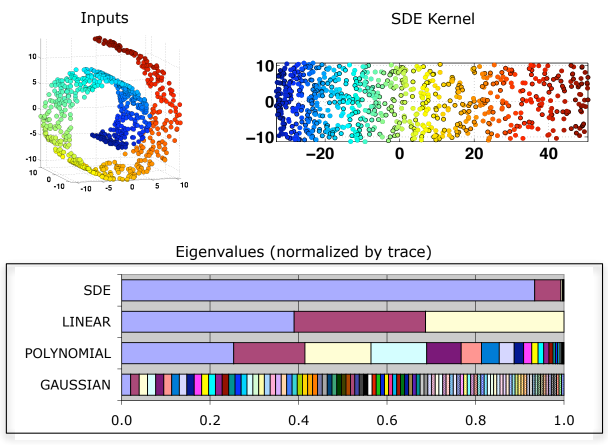


Figure 1. Top: results of SDE applied to $N = 800$ inputs sampled from a "Swiss roll" (top left). The inputs had $D = 23$ dimensions, of which 20 were filled with small amounts of noise (not shown). The two dimensional plot shows the embedding from kernel PCA with the SDE kernel. *Bottom:* eigenvalues of different kernel matrices, normalized by their trace. Only the eigenvalues from SDE indicate the correct intrinsic dimensionality ($d = 2$) of the Swiss roll.

In the first experiment, the inputs were sampled from a three dimensional "Swiss roll", a data set commonly used to evaluate algorithms in manifold learning (Tenenbaum et al., 2000). Fig. 1 shows the original inputs (top left), the embedding discovered by SDE with $k = 4$ nearest neighbors (top right), and the eigenvalue spectra from several different kernel matrices (bottom). The color coding of the embedding reveals that the Swiss roll has been successfully unraveled. Note that the kernel matrix learned by SDE has two dominant eigenvalues, indicating the correct underlying dimensionality of the Swiss roll, whereas the eigenspectra of other kernel matrices fail to reveal this structure. In particular, the linear kernel matrix has three dominant eigenvalues, reflecting the extrinsic dimensionality of the swiss roll, while the eigenspectra of the polynomial ($p = 4$) and Gaussian ($\sigma = 1.45$) kernel matrices[2] indicate that the variances of their features $\mathbf{\Phi}(\mathbf{x}_i)$ are spread across a far greater number of dimensions than the original inputs $\mathbf{x}_i$.

The second experiment was performed on a data set consisting of $N = 400$ color images of a teapot viewed from different angles in the plane. With a resolution of 76×101 and three bytes of color information per pixel,

[2]For all the data sets in this section, we set the width parameter σ of the Gaussian kernel to the estimated standard deviation within neighborhoods of size $k = 4$, thus reflecting the same length scale used in SDE.

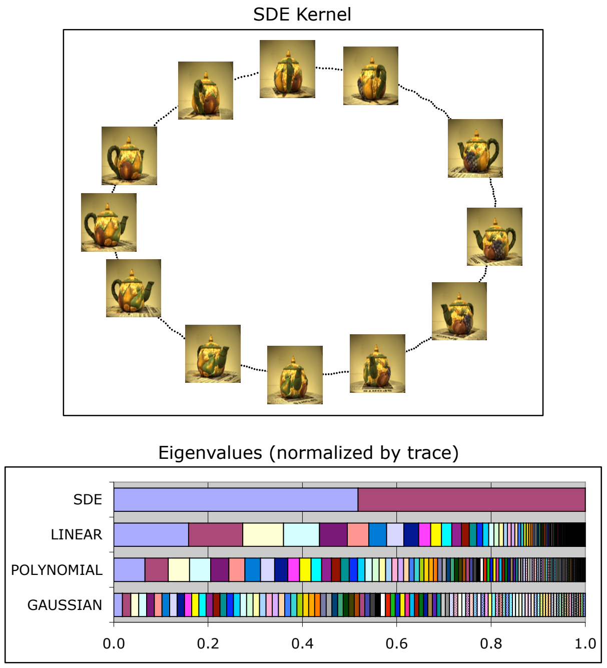


Figure 2. Results of SDE applied to $N = 400$ images of a teapot viewed from different angles in the plane, under a full 360 degrees of rotation. The images were represented by inputs in a $D = 23028$ dimensional vector space. SDE faithfully represents the 360 degrees of rotation by a circle. The eigenvalues of different kernel matrices are also shown, normalized by their trace.

the images were represented as points in a $D = 23028$ dimensional vector space. Though very high dimensional, the images in this data set are effectively parameterized by a single degree of freedom—namely, the angle of rotation. The low dimensional embedding of these images by SDE and the eigenvalue spectra of different kernel matrices are shown in Fig. 2. The kernel matrix learned by SDE (with $k = 4$ nearest neighbors) concentrates the variance of the feature space in two dimensions and maps the images to a circle, a highly intuitive representation of the full 360 degrees of rotation. By contrast, the linear, polynomial ($p = 4$), and Gaussian ($\sigma = 1541$) kernel matrices have eigenvalue spectra that do not reflect the low intrinsic dimensionality of the data set.

Why does kernel PCA with the Gaussian kernel perform so differently on these data sets when its width parameter σ reflects the same length scale as neighborhoods in SDE? Note that the Gaussian kernel computes a nearly zero inner product ($K_{ij} \approx 0$) in feature space for inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ that do not belong to the same or closely overlapping neighborhoods. It follows from these inner products that the feature vectors

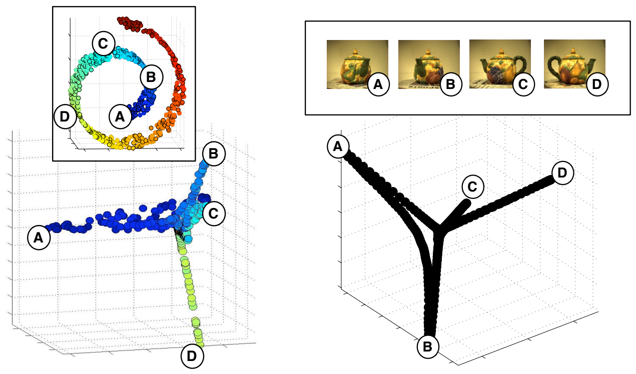


Figure 3. Embeddings from kernel PCA with the Gaussian kernel on the Swiss roll and teapot data sets in Figs. 1 and 2. The first three principal components are shown. In both cases, different patches of the manifolds are mapped to orthogonal parts of feature space.

$\mathbf{\Phi}(\mathbf{x}_i)$ and $\mathbf{\Phi}(\mathbf{x}_j)$ must be nearly orthogonal. As a result, the different patches of the manifold are mapped into orthogonal regions of the feature space: see Fig. 3. Thus, rather than unfolding the manifold, the Gaussian kernel leads to an embedding whose dimensionality is equal to the number of non-overlapping patches of length scale σ. This explains the generally poor performance of the Gaussian kernel for manifold learning (as well as its generally good performance for large margin classification, discussed in section 4.2).

Another experiment on the data set of teapot images was performed by restricting the angle of rotation to 180 degrees. The results are shown in Fig. 4. Interestingly, in this case, the eigenspectra of the linear, polynomial, and Gaussian kernel matrices are not qualitatively different. By contrast, the SDE kernel matrix now has only one dominant eigenvalue, indicating that the variability for this subset of images is controlled by a single (non-cyclic) degree of freedom.

As a final experiment, we compared the performance of the different kernels on a real-world data set described by an underlying manifold. The data set (Hull, 1994) consisted of $N = 953$ grayscale images at 16×16 resolution of handwritten twos and threes (in roughly equal proportion). In this data set, it is possible to find a relatively smooth "morph" between any pair of images, and a relatively small number of degrees of freedom describe the possible modes of variability (e.g. writing styles for handwritten digits). Fig 5 shows the results. Note that the kernel matrix learned by SDE concentrates the variance in a significantly fewer number of dimensions, suggesting it has constructed a more appropriate feature map for nonlinear dimensionality reduction.

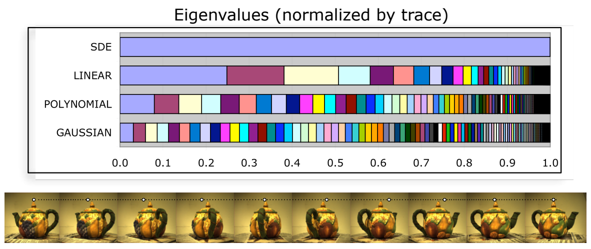


Figure 4. Results of kernel PCA applied to $N = 200$ images of a teapot viewed from different angles in the plane, under 180 degrees of rotation. The eigenvalues of different kernel matrices are shown, normalized by their trace. The one dimensional embedding from SDE is also shown.

4.2. Large margin classification

We also evaluated the use of SDE kernel matrices for large margin classification by SVMs. Several training and test sets for problems in binary classification were created from the USPS data set of handwritten digits (Hull, 1994). Each training and test set had 810 and 90 examples, respectively. For each experiment, the SDE kernel matrices were learned (using $k = 4$ nearest neighbors) on the combined training and test data sets, ignoring the target labels. The results were compared to those obtained from linear, polynomial ($p = 2$), and Gaussian ($\sigma = 1$) kernels. Table 1 shows that the SDE kernels performed quite poorly in this capacity, even worse than the linear kernels.

Fig. 6 offers an explanation of this poor performance. The SDE kernel can only be expected to perform well for large margin classification if the decision boundary on the unfolded manifold is approximately linear. There is no a priori reason, however, to expect this type of linear separability. The example in Fig. 6 shows a particular binary labeling of inputs on the Swiss roll for which the decision boundary is much simpler in the input space than on the unfolded manifold. A similar effect seems to be occurring in the large margin classification of handwritten digits. Thus, the strength of SDE for nonlinear dimensionality reduction is generally a weakness for large margin classification. By contrast, the polynomial and Gaussian kernels lead to more powerful classifiers precisely because they map inputs to higher dimensional regions of feature space.

5. Related and ongoing work

SDE can be viewed as an unsupervised counterpart to the work of Graepel (2002) and Lanckriet et al (2004), who proposed learning kernel matrices by semidefinite programming for large margin classification. The kernel matrices learned by SDE differ from those usually

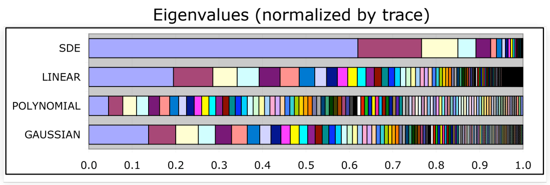


Figure 5. Results of kernel PCA applied to $N = 953$ images of handwritten digits. The eigenvalues of different kernel matrices are shown, normalized by their trace.

Digits	Linear	Polynomial	Gaussian	SDE
1 vs 2	0.00	0.00	0.14	0.59
1 vs 3	0.23	0.00	0.35	1.73
2 vs 8	2.18	1.12	0.63	3.37
8 vs 9	1.00	0.54	0.29	1.27

Table 1. Percent error rates for SVM classification using different kernels on test sets of handwritten digits. Each line represents the average of 10 experiments with different 90/10 splits of training and testing data. Here, the SDE kernel performs much worse than the other kernels.

employed in SVMs, in that they aim to map inputs into an (effectively) *lower* dimensional feature space. This explains both our positive results in nonlinear dimensionality reduction (section 4.1), as well as our negative results in large margin classification (section 4.2).

SDE can also be viewed as an alternative to manifold learning algorithms such as Isomap (Tenenbaum et al., 2000), locally linear embedding (LLE) (Roweis & Saul, 2000; Saul & Roweis, 2003), hessian LLE (hLLE) (Donoho & Grimes, 2003), and Laplacian eigenmaps (Belkin & Niyogi, 2003). All these algorithms share a similar structure, creating a graph based on nearest neighbors, computing an $N \times N$ matrix from geometric properties of the inputs, and constructing an embedding from the eigenvectors with the largest or smallest nonnegative eigenvalues. A more detailed discussion of differences between these algorithms is given in a companion paper (Weinberger &

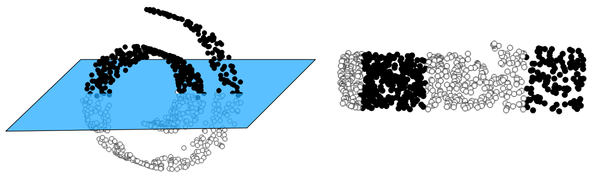
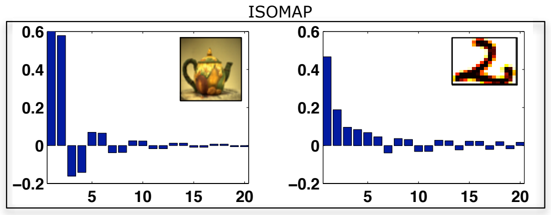


Figure 6. Left: linearly separable inputs (in black versus white) sampled from a a Swiss roll. *Right:* unfolding the manifold leads to a more complicated decision boundary.



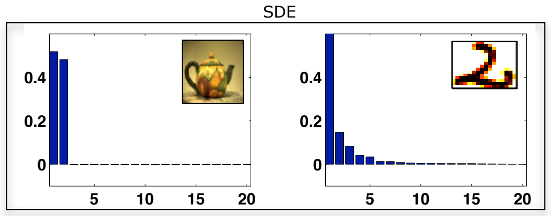


Figure 7. The twenty eigenvalues of leading *magnitude* from Isomap and SDE on the data sets of teapot images and handwritten digits. Note that the similarity matrices constructed by Isomap have negative eigenvalues.

Saul, 2004). Here, we comment mainly on their various interpretations as kernel methods (Ham et al., 2004). In general, these other methods give rise to matrices whose geometric properties as kernels are less robust or not as well understood. For example, unlike SDE, the similarity matrix constructed by Isomap from finite data sets can have negative eigenvalues. In some cases, moreover, these negative eigenvalues can be appreciable in magnitude to the dominant positive ones: see Fig. 7. Unlike both SDE and Isomap, LLE and hLLE construct matrices whose *bottom* eigenvectors yield low dimensional embeddings; to interpret these matrices as kernels, their eigenvalues must be “flipped”, either by inverting the matrix itself or by subtracting it from a large multiple of the identity matrix. Moreover, it does not appear that these eigenvalues can be used to estimate the intrinsic dimensionality of underlying manifolds (Saul & Roweis, 2003). A kernel can be derived from the discrete Laplacian by noting its role in the heat diffusion equation, but the intuition gained from this analogy, in terms of diffusion times through a network, does not relate directly to the geometric properties of the kernel matrix. SDE stands apart from these methods in its explicit construction of a semipositive definite kernel matrix that preserves the geometric properties of the inputs up to local isometry and whose eigenvalues indicate the dimensionality of the underlying manifold.

We are pursuing many directions in ongoing work. The first is to develop faster and potentially distributed (Biswas & Ye, 2003) methods for solving the instance of semidefinite programming in SDE and for out-of-sample extensions (Bengio et al., 2004). Thus far we have been using generic solvers with a relatively high time complexity, relying on toolboxes that do not exploit any special structure in our problem. We are also investigating many variations on the objective function and constraints in SDE—for example, to allow some slack in the preservation of local distances and angles, or to learn embeddings onto spheres. Finally, we are performing more extensive comparisons with other methods in nonlinear dimensionality reduction. Not surprisingly, perhaps, all these directions reflect a more general trend toward the convergence of research in kernel methods and manifold learning.

Acknowledgements

The authors are grateful to S. Boyd and Y. Ye (Stanford) for useful discussions of semidefinite programming and to the anonymous reviewers for many helpful comments.

References

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

Bengio, Y., Paiement, J.-F., & Vincent, P. (2004). Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Biswas, P., & Ye, Y. (2003). A distributed method for solving semideinite programs arising from ad hoc wireless sensor network localization. Stanford University, Department of Electrical Engineering, working paper.

Burges, C. J. C. (1999). Geometry and invariance in kernel based methods. *Advances in Kernel Methods—-Support Vector Learning*. Cambridge, MA: MIT Press.

Cortes, C., Haffner, P., & Mohri, M. (2003). Rational kernels. *Advances in Neural Information Processing Systems 15* (pp. 617–624). Cambridge, MA: MIT Press.

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

Graepel, T. (2002). Kernel matrix completion by semidefinite programming. *Proceedings of the International Conference on Artificial Neural Networks* (pp. 694–699). Springer-Verlag.

Ham, J., Lee, D. D., Mika, S., & Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*. Banff, Canada.

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

Jolliffe, I. T. (1986). *Principal component analysis*. New York: Springer-Verlag.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lodhi, H., Saunders, C., Cristianini, N., & Watkins, C. (2004). String matching kernels for text classification. *Journal of Machine Learning Research*. in press.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

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

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization overy symmetric cones. *Optimization Methods and Software*, *11-12*, 625–653.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Vandenberghe, L., & Boyd, S. P. (1996). Semidefinite programming. *SIAM Review*, *38(1)*, 49–95.

Weinberger, K. Q., & Saul, L. K. (2004). Unsupervised learning of image manifolds by semidefinite programming. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*. Washington D.C.