Spectral Dimensionality Reduction

Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux
Jean-François Paiement, Pascal Vincent, and Marie Ouimet
Département d'Informatique et Recherche Opérationnelle
Centre de Recherches Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
http://www.iro.umontreal.ca/∼bengioy

Abstract

In this chapter, we study and put under a common framework a number of non-linear dimensionality reduction methods, such as Locally Linear Embedding, Isomap, Laplacian eigenmaps and kernel PCA, which are based on performing an eigen-decomposition (hence the name "spectral"). That framework also includes classical methods such as PCA and metric multidimensional scaling (MDS). It also includes the data transformation step used in spectral clustering. We show that in all of these cases the learning algorithm estimates the principal eigenfunctions of an operator that depends on the unknown data density and on a kernel that is not necessarily positive semi-definite. This helps to generalize some of these algorithms so as to predict an embedding for out-of-sample examples without having to retrain the model. It also makes it more transparent what these algorithm are minimizing on the empirical data and gives a corresponding notion of generalization error.

1 Introduction

Unsupervised learning algorithms attempt to extract important characteristics of the unknown data distribution from the given examples. High-density regions are such salient features and they can be described by clustering algorithms (where typically each cluster corresponds to a high density "blob") or by manifold learning algorithms (which discover high-density low-dimensional surfaces). A more generic description of the density is given by algorithms that estimate the density function.
In the context of supervised learning (each example is associated with a target label) or semi-supervised learning (a few examples are labeled but most are not), manifold

learning algorithms can be used as pre-processing methods to perform dimensionality reduction. Each input example is then associated with a low-dimensional representation which corresponds to its estimated coordinates on the manifold. Since the manifold learning can be done without using the target labels, it can be applied on all of the input examples (both those labeled and those unlabeled). If there are many unlabeled examples, it has been shown that they can help to learn a more useful low-dimensional representation of the data (Belkin and Niyogi, 2003b). Dimensionality reduction is an interesting alternative to feature selection. Like feature selection it yields a low-dimensional representation which helps to build lower capacity predictors in order to improve generalization. However, unlike feature selection it may preserve information from all the original input variables. In fact, if the data really lie on a low-dimensional manifold, it may preserve almost all of the original information while representing it in a way that eases learning. For example, manifold learning algorithms such as those described in this chapter often have the property of "unfolding" the manifold, i.e. flattening it out, as shown in figure 1. On the other hand, these techniques being purely unsupervised, they may throw away low variance variations that are highly predictive of the target label. In addition to being useful as a preprocessing step for supervised or semi-supervised learning, linear and non-linear dimensionality reduction is often used for data analysis and visualization, e.g. (Vlachos et al., 2002), since visualizing the projected data (two or three dimensions at a time) can help to better understand them.

In the last few years, many unsupervised learning algorithms have been proposed which share the use of an eigen-decomposition for obtaining a lower-dimensional embedding of the data that characterizes a non-linear manifold near which the data would lie: Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000) and Laplacian eigenmaps (Belkin and Niyogi, 2003a). There are also many variants of spectral clustering (Weiss, 1999; Ng, Jordan and Weiss, 2002), in which such an embedding is an intermediate step before obtaining a clustering of the data that can capture flat, elongated and even curved clusters. The two tasks (manifold learning and clustering) are linked because the clusters that spectral clustering manages to capture can be arbitrary curved manifolds (as long as there is enough data to locally capture the curvature of the manifold): clusters and manifold both are zones of high density. An interesting advantage of the family of manifold learning algorithms described in this chapter is that they can easily be applied in the case of non-vectorial data as well as data for which no vectorial representation is available but for which a similarity function between objects can be computed, as in the MDS (multi-dimensional scaling) algorithms (Torgerson, 1952; Cox and Cox, 1994).

There are of course several dimensionality reduction methods that do not fall in the spectral framework described here, but which may have interesting connections nonetheless. For example, the principal curves algorithms (Hastie and Stuetzle, 1989; Kegl and Krzyzak, 2002) have been introduced based on geometric grounds, mostly for 1-dimensional manifolds. Although they optimize a different type of criterion, their spirit is close to that of LLE and Isomap. Another very interesting family of algorithms is the Self-Organizing Map (Kohonen, 1990). With these algorithms, the low dimensional

embedding space is discretized (into topologically organized centers) and one learns the coordinates in the raw high-dimensional space of each of these centers. Another neural network like approach to dimensionality reduction is the auto-associative neural network (Rumelhart, Hinton and Williams, 1986; **?**; Saund, 1989), in which one trains a multi-layer neural network to predict its input, but forcing the intermediate representation of the hidden units to be a compact code. In section 2.7 we discuss in more detail a family of density estimation algorithms that can be written as mixtures of Gaussians with low-rank covariance matrices, having intimate connections with the LLE and Isomap algorithms.
An interesting question that will not be studied further in this chapter is that of selecting the dimensionality of the embedding. This is fundamentally a question of model selection. It could be addressed using traditional model selection methods (such as cross-validation) when the low-dimensional representation is used as input for a supervised learning algorithm. Another approach is that of inferring the dimensionality based on purely unsupervised grounds, using the geometric properties of the empirical data distribution (Kégl, 2003).

1.1 Transduction and Induction

The end result of most inductive machine learning algorithms is a function that minimizes the empirical average of a loss criterion (possibly plus regularization). The function can be applied on new points and for such learning algorithms it is clear that the ideal solution is a function that minimizes the expected value of that loss criterion under the unknown true distribution from which the data was sampled. That expected loss is known as the generalization error.
However, such a characterization was missing for spectral embedding algorithms such as metric Multi-Dimensional Scaling (MDS) (Torgerson, 1952; Cox and Cox, 1994), spectral clustering (see (Weiss, 1999) for a review), Laplacian eigenmaps (Belkin and Niyogi, 2003a), Locally Linear Embedding (LLE) (Roweis and Saul, 2000) and Isomap (Tenenbaum, de Silva and Langford, 2000), which are used either for dimensionality reduction or for clustering. As such these algorithms are therefore really *transduction* algorithms: any test data for which an embedding is desired must be included in the (unlabeled) "training set" on which the algorithm is applied. For example, if the embedding obtained is used as an input representation for a supervised learning algorithm, the input part of the test examples must be provided at the time of learning the embedding. The basic form of these algorithms does not provide a generic function that can be applied to new points in order to obtain an embedding or a cluster membership, and the notion of generalization error that would be implicitly minimized is not clearly defined either.
As a natural consequence of providing a unifying framework for these algorithms, we provide an answer to these questions. A loss criterion for spectral embedding algorithms can be defined. It is a reconstruction error that depends on pairs of examples. Minimizing its average value yields the eigenvectors that provide the classical output of these algorithms, i.e. the embeddings. Minimizing its expected value over the true underlying distribution yields the eigenfunctions of a linear operator (called L here) that is defined

with a similarity function (a kernel, but not necessarily positive semi-definite) and the data-generating density. When the kernel is positive semi-definite and we work with the empirical density there is a direct correspondence between these algorithms and kernel Principal Component Analysis (PCA) (Schölkopf, Smola and Müller, 1998). Our work is also a direct continuation of previous work (Williams and Seeger, 2000) noting that the Nyström formula and the kernel PCA projection (which are equivalent) represent an approximation of the eigenfunctions of the above linear operator. Previous analysis of the convergence of generalization error of kernel PCA (Shawe-Taylor, Cristianini and Kandola, 2002; Shawe-Taylor and Williams, 2003; Zwald, Bousquet and Blanchard, 2004) also help to justify the view that these methods are estimating the convergent limit of some eigenvectors (at least when the kernel is positive semi-definite). The eigenvectors can then be turned into estimators of eigenfunctions, which can therefore be applied to new points, turning the spectral embedding algorithms into function induction algorithms. The Nyström formula obtained this way is well known (Baker, 1977), and will be given in eq. 2 below. This formula has been used previously for estimating extensions of eigenvectors in Gaussian process regression (Williams and Seeger, 2001), and it was noted (Williams and Seeger, 2000) that it corresponds to the projection of a test point computed with kernel PCA.

In order to extend spectral embedding algorithms such as LLE and Isomap to out-of-sample examples, this chapter defines for these spectral embedding algorithms data-dependent kernels k_m that can be applied outside of the training set. See also the independent work (Ham et al., 2003) for a kernel view of LLE and Isomap, but where the kernels are only applied on the training set.

Obtaining an induced function that can be applied to out-of-sample examples is not only interesting from a theoretical point of view, it is also computationally useful. It allows to say something about new examples without having to re-do the kernel computations (building the Gram matrix, normalizing it, and computing the principal eigenvectors, which all take at least time quadratic in the number of training examples). The formula proposed requires time linear in the training set size.

Additional contributions of this chapter include a characterization of the empirically estimated eigenfunctions in terms of eigenvectors in the case where the kernel is not positive semi-definite (which is often the case for MDS and Isomap), a convergence theorem linking the Nyström formula to the eigenfunctions of L, as well as experiments on MDS, Isomap, LLE and spectral clustering / Laplacian eigenmaps showing that the Nyström formula for out-of-sample examples is accurate.

1.2 Notation

To simplify the presentation, we will consider the vector-space versions of these algorithms, in which we start from a data set $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ with $\boldsymbol{x}_i \in \mathbb{R}^n$ sampled i.i.d. from an unknown distribution with density $p(\cdot)$. However, the results in this chapter can be readily extended to the case of arbitrary objects, with $p(\boldsymbol{x})d\boldsymbol{x}$ replaced by $d\mu(\boldsymbol{x})$ with $\mu(\boldsymbol{x})$ an appropriate measure, and the only quantity that is required in the algorithms is the similarity or distance between pairs of objects (e.g. similarity $k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$

below). See for example the treatment of pairwise measurement data for LLE (Saul and Roweis, 2002) and Isomap (Tenenbaum, de Silva and Langford, 2000), and for MDS in section 2.2.
Below we will use the notation

$$E_{\boldsymbol{x}}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$$

for averaging over $p(\boldsymbol{x})$ and

$$\hat{E}_{\boldsymbol{x}}[f(\boldsymbol{x})] = \frac{1}{m}\sum_{i=1}^{m} f(\boldsymbol{x}_i)$$

for averaging over the data in $\boldsymbol{X}$, i.e. over the empirical distribution denoted $\hat{p}(\boldsymbol{x})$. We will denote kernels with $k_m(\boldsymbol{x}, \boldsymbol{y})$ or $\tilde{k}(\boldsymbol{x}, \boldsymbol{y})$, symmetric functions, not always positive semi-definite, that may depend not only on $\boldsymbol{x}$ and $\boldsymbol{y}$ but also on the data $\boldsymbol{X}$. The spectral embedding algorithms construct an affinity matrix $\boldsymbol{K}$, either explicitly through

$$K_{ij} = k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{1}$$

or implicitly through a procedure that takes the data $\boldsymbol{X}$ and computes $\boldsymbol{K}$. We denote by $v_{r,i}$ the i-th coordinate of the r-th eigenvector of $\boldsymbol{K}$ (sorted in order of decreasing eigenvalues), associated with the eigenvalue ℓ_r. With these notations, the Nyström formula discussed above can be written:

$$f_{r,m}(\boldsymbol{x}) = \frac{\sqrt{m}}{\ell_r}\sum_{i=1}^{m} v_{r,i}k_m(\boldsymbol{x}, \boldsymbol{x}_i) \tag{2}$$

where $f_{r,m}$ is the r-th Nyström estimator with m samples. We will show in section 3 that it estimates the r-th eigenfunction of a linear operator and that it provides an embedding for a new example $\boldsymbol{x}$.

2 Data-Dependent Kernels for Spectral Embedding Algorithms

The first and foremost observation to make is that many spectral embedding algorithms can be cast in a common framework. The spectral embedding algorithms can be seen to build a $(m \times m)$ similarity matrix $\boldsymbol{K}$ (also called the *Gram* matrix)[1] and compute its principal eigenvectors $\boldsymbol{v}_r = (v_{r,1}, \ldots, v_{r,m})^T$ (one entry per exemple) with eigenvalues ℓ_r (sorted by decreasing order). *The embedding associated with the i-th training example is given by the i-th element of the principal eigenvectors, up to some scaling:*

$$\mathbf{P}(\boldsymbol{x}_i) = (v_{1,i}, v_{2,i}, \ldots, v_{N,i})^T \tag{3}$$

[1]For Laplacian eigenmaps (section 2.4) and LLE (section 2.6), the matrix $\boldsymbol{K}$ discussed here is not the one defined in the original papers on these algorithms, but a transformation of it to reverse the order of eigenvalues, as we see below.

where $N \leq m$ is the desired number of embedding coordinates. The scaling factor depends on the algorithm: for instance, in kernel PCA, MDS and Isomap, $v_{r,i}$ is multiplied by $\sqrt{\ell_r}$, and in LLE it is multiplied by $\sqrt{m}$ to obtain the actual embedding coordinates. In general, we will see that K_{ij} depends not only on $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ but also on the other training examples. Nonetheless, as we show below, it can always be written $K_{ij} = k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where k_m is a "data-dependent" kernel (i.e. it is a function of the m elements of the training set $\boldsymbol{X}$, and not just of its two arguments). In many algorithms a matrix $\tilde{\boldsymbol{K}}$ is first formed from a simpler, often data-independent kernel (such as the Gaussian kernel), and then transformed into $\boldsymbol{K}$. We want to think of the entries of $\boldsymbol{K}$ as being generated by applying k_m to the pairs $(\boldsymbol{x}_i,\boldsymbol{x}_j)$ because this will help us to generalize to new examples not in the training set $\boldsymbol{X}$, and it will help us to think about what happens as m increases.
For each of these methods, by defining a kernel k_m that can be applied outside of the training set, we will be able to generalize the embedding to a new point $\boldsymbol{x}$, via the Nyström formula (eq. 2 above, and section 3.2). This will only require computations of the form $k_m(\boldsymbol{x}, \boldsymbol{x}_i)$ with $\boldsymbol{x}_i$ a training point.

2.1 Kernel Principal Component Analysis

Kernel PCA is an unsupervised manifold learning technique that maps data points to a new space, generally lower-dimensional (but not necessarily). It generalizes the Principal Component Analysis approach to non-linear transformations using the kernel trick (Schölkopf, Smola and Müller, 1996; Schölkopf, Smola and Müller, 1998; Schölkopf, Burges and Smola, 1999). One considers the data mapped into a "feature space", a Hilbert space of possibly infinite dimension such that if $\boldsymbol{x}$ is mapped to $\tilde{\boldsymbol{\phi}}(\boldsymbol{x})$, we have $\langle \tilde{\boldsymbol{\phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\phi}}(\boldsymbol{y}) \rangle = \tilde{k}(\boldsymbol{x}, \boldsymbol{y})$. Here, $\tilde{k}$ must be a positive (semi)-definite kernel, and is often taken as the Gaussian kernel[2], i.e.

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}}. \tag{4}$$

The kernel PCA algorithm consists in performing PCA in the feature space: it implicitly finds the leading eigenvectors and eigenvalues of the covariance of the projection $\tilde{\boldsymbol{\phi}}(\boldsymbol{x})$ of the data. *If the data is centered in feature space* ($\hat{E}_{\boldsymbol{x}}[\tilde{\boldsymbol{\phi}}(\boldsymbol{x})] = 0$), the (empirical) feature space covariance matrix is $\boldsymbol{C} = \hat{E}_{\boldsymbol{x}}[\tilde{\boldsymbol{\phi}}(\boldsymbol{x})\tilde{\boldsymbol{\phi}}(\boldsymbol{x})^T]$. In general, however, the data is not centered, and we need to define a "centered" mapping

$$\boldsymbol{\phi}_m(\boldsymbol{x}) = \tilde{\boldsymbol{\phi}}(\boldsymbol{x}) - \frac{1}{m}\sum_{i=1}^{m} \tilde{\boldsymbol{\phi}}(\boldsymbol{x}_i)$$

and an associated *data-dependent* kernel k_m such that $k_m(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\phi}_m(\boldsymbol{x}), \boldsymbol{\phi}_m(\boldsymbol{y}) \rangle$, which rewrites:

$$k_m(\boldsymbol{x}, \boldsymbol{y}) = \tilde{k}(\boldsymbol{x}, \boldsymbol{y}) - \hat{E}_{\boldsymbol{x}'}[\tilde{k}(\boldsymbol{x}', \boldsymbol{y})] - \hat{E}_{\boldsymbol{y}'}[\tilde{k}(\boldsymbol{x}, \boldsymbol{y}')] + \hat{E}_{\boldsymbol{x}',\boldsymbol{y}'}[\tilde{k}(\boldsymbol{x}', \boldsymbol{y}')]. \tag{5}$$

[2]Using the Gaussian kernel is not always a good idea, as seen in section 5.1, and other nonlinear kernels, such as polynomial kernels, may be more suited.

The empirical covariance matrix $\boldsymbol{C}$ in "feature space" is thus actually defined by

$$\boldsymbol{C} = \hat{E}_{\boldsymbol{x}}[\boldsymbol{\phi}_m(\boldsymbol{x})\boldsymbol{\phi}_m(\boldsymbol{x})^T] \tag{6}$$

with eigenvectors $\boldsymbol{w}_r$ associated with eigenvalues λ_r. As shown in (Schölkopf, Smola and Müller, 1998), this eigen-decomposition of $\boldsymbol{C}$ is related to the one of $\boldsymbol{K}$ (the Gram matrix defined by eq. 1) through $\lambda_r = \ell_r/m$ and

$$\boldsymbol{w}_r = \frac{1}{\sqrt{\ell_r}} \sum_{i=1}^{m} v_{r,i} \boldsymbol{\phi}_m(\boldsymbol{x}_i)$$

where $\boldsymbol{v}_r$ are the eigenvectors of $\boldsymbol{K}$, associated with eigenvalues ℓ_r. As in PCA, one can then obtain the embedding of a training point $\boldsymbol{x}_i$ by the projection of $\boldsymbol{\phi}_m(\boldsymbol{x}_i)$ on the leading eigenvectors $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ of $\boldsymbol{C}$, which yields exactly the embedding of eq. 3, if we multiply $v_{r,i}$ by $\sqrt{\ell_r}$.
Note that, as in PCA, we can also compute the projection $\mathbf{P}(\boldsymbol{x}) = (P_1(\boldsymbol{x}), \ldots, P_N(\boldsymbol{x}))^T$ for a new point $\boldsymbol{x}$, which is written

$$P_r(\boldsymbol{x}) = \langle \boldsymbol{w}_r, \boldsymbol{\phi}_m(\boldsymbol{x}) \rangle = \frac{1}{\sqrt{\ell_r}} \sum_{i=1}^{m} v_{r,i} k_m(\boldsymbol{x}_i, \boldsymbol{x}). \tag{7}$$

This is the key observation that will allow us, in section 3.2, to extend to new points the embedding obtained with other spectral algorithms.

2.2 Multi-Dimensional Scaling

Metric Multi-Dimensional Scaling (MDS) (Torgerson, 1952; Cox and Cox, 1994) starts from a notion of distance $d(\boldsymbol{x}, \boldsymbol{y})$ that is computed between each pair of training examples to fill a matrix $\tilde{K}_{ij} = d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The idea is to find a low-dimensional embedding of the dataset $\boldsymbol{X}$ that preserves the given distances between training points. To do so, the distances are converted to equivalent dot products using the "double-centering" formula, which makes K_{ij} depend not only on $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ but also on all the other examples:

$$K_{ij} = -\frac{1}{2} \left(\tilde{K}_{ij} - \frac{1}{m} S_i - \frac{1}{m} S_j + \frac{1}{m^2} \sum_k S_k \right) \tag{8}$$

where the S_i are the row sums of $\tilde{\boldsymbol{K}}$:

$$S_i = \sum_{j=1}^{m} \tilde{K}_{ij}. \tag{9}$$

Eq. 8 is used to obtain for K_{ij} the centered dot product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the pairwise squared distances given by d^2, just as eq. 5 yields the centered dot product (in feature space) from the pairwise non-centered dot products given by $\tilde{k}$. The embedding

of the example $\boldsymbol{x}_i$ is then given by eq. 3 with $v_{r,i}$ multiplied by $\sqrt{\ell_r}$. If d is the Euclidean distance, this is the same embedding as in classical (linear) PCA.
A corresponding data-dependent kernel which generates the matrix $\boldsymbol{K}$ is:

$$k_m(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2}\left(d^2(\boldsymbol{x}, \boldsymbol{y}) - \hat{E}_{\boldsymbol{x}'}[d^2(\boldsymbol{x}', \boldsymbol{y})] - \hat{E}_{\boldsymbol{y}'}[d^2(\boldsymbol{x}, \boldsymbol{y}')] + \hat{E}_{\boldsymbol{x}',\boldsymbol{y}'}[d^2(\boldsymbol{x}', \boldsymbol{y}')]\right). \quad (10)$$

2.3 Spectral Clustering

Several variants of spectral clustering have been proposed (Weiss, 1999). They can yield impressively good results where traditional clustering looking for "round blobs" in the data, such as k-means, would fail miserably (see figure 1). It is based on two main steps: first embedding the data points in a space in which clusters are more "obvious" (using the eigenvectors of a Gram matrix), and then applying a classical clustering algorithm such as k-means, e.g. as in (Ng, Jordan and Weiss, 2002). To construct the spectral clustering affinity matrix $\boldsymbol{K}$, we first apply a data-independent kernel $\tilde{k}$ such as the Gaussian kernel to each pair of examples: $\tilde{K}_{ij} = \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The matrix $\tilde{\boldsymbol{K}}$ is then normalized, e.g. using "divisive" normalization (Weiss, 1999; Ng, Jordan and Weiss, 2002)[3] :

$$K_{ij} = \frac{\tilde{K}_{ij}}{\sqrt{S_i S_j}} \quad (11)$$

with S_i and S_j defined by eq. 9. To obtain N clusters, the first N principal eigenvectors of $\boldsymbol{K}$ are computed and k-means is applied on the embedding coordinates after normalizing each embedding vector to have unit norm: the r-th coordinate of the i-th example is $v_{r,i}/\sqrt{\sum_{l=1}^{N} v_{l,i}^2}$. Note that if one is interested in the embedding prior to normalization, this embedding should be multiplied by $\sqrt{m}$ to be stable (to keep the same order of magnitude) as m varies.
To generalize spectral clustering to out-of-sample points, we will need a kernel that could have generated that matrix $\boldsymbol{K}$:

$$k_m(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{m} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{\hat{E}_{\boldsymbol{x}'}[\tilde{k}(\boldsymbol{x}', \boldsymbol{y})] \hat{E}_{\boldsymbol{y}'}[\tilde{k}(\boldsymbol{x}, \boldsymbol{y}')]}}. \quad (12)$$

Note that this divisive normalization comes out of the justification of spectral clustering as a relaxed statement of the min-cut problem (Chung, 1997; Spielman and Teng, 1996) (to divide the examples into two groups such as to minimize the sum of the "similarities" between pairs of points straddling the two groups). The additive normalization performed with kernel PCA (eq. 5) makes sense geometrically as a centering in feature space. Both the divisive normalization and the additive normalization procedures make use of a kernel row/column average. It would be interesting to find a similarly pleasing geometric interpretation to the divisive normalization.

[3]Better embeddings for clustering are usually obtained if we define $S_i = \sum_{j \neq i} \tilde{K}_{ij}$: this alternative normalization can also be cast into the general framework developed here, with a slightly different kernel. Also, one could take the row average instead of the row sum, which seems more natural even if it does not change the embedding.

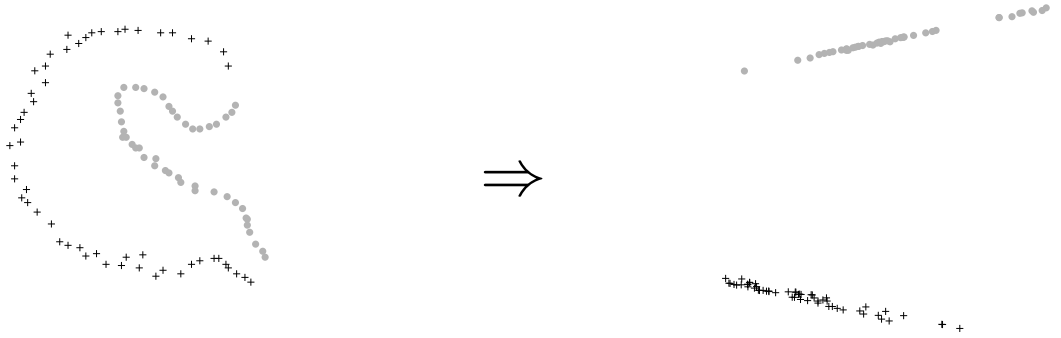


Figure 1: *Example of the transformation learned as part of spectral clustering. Input data on the left, transformed data on the right. Gray level and cross/circle drawing are only used to show which points get mapped where: the mapping reveals both the clusters and the internal structure of the two manifolds.*

2.4 Laplacian Eigenmaps

The Laplacian eigenmaps method is a recently proposed dimensionality reduction procedure (Belkin and Niyogi, 2003a) that was found to be very successful for *semi-supervised learning*, where one uses a large unlabeled dataset to learn the manifold structure, thus reducing the dimensionality of labeled data (which can benefit to supervised learning algorithms). Several variants have been proposed by the authors and we focus here on the latest one, but they all share the same spirit.

The Laplacian operator has a natural interpretation as a smoothness functional: we look for an embedding $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ of the training points such that $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$ is small when i and j are "near" each other, i.e. when $\tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is large (if $\tilde{k}$ can be interpreted as a similarity function). This corresponds to minimizing $\sum_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. It has to be done under a norm constraint, and an appropriate one is that, denoting $\boldsymbol{Y}$ the $(m \times N)$ matrix whose i-th row is $\boldsymbol{y}_i$, we force $\boldsymbol{Y}^T \boldsymbol{S} \boldsymbol{Y} = \boldsymbol{I}$, where $\boldsymbol{S}$ is the diagonal matrix with elements S_i from eq. 9 (row sums of $\tilde{\boldsymbol{K}}$, possibly ignoring diagonal terms). This norm constraint has the advantage of giving an appropriate weight to examples which are "connected" to more other examples. Rearranging this criterion, the solutions to the constrained optimization problem correspond to the following generalized eigenproblem:

$$(\boldsymbol{S} - \tilde{\boldsymbol{K}}) \boldsymbol{z}_r = \sigma_r \boldsymbol{S} \boldsymbol{z}_r \tag{13}$$

with eigenvalues σ_r, and eigenvectors $\boldsymbol{z}_r$ being the columns of $\boldsymbol{Y}$. The solution with smallest (zero) eigenvalue corresponds to the uninteresting solution with constant embedding, so it is discarded. The eigenvalue corresponding to a solution quantifies the above defined smoothness, so we keep the N solutions with smallest non-zero eigenvalues, yielding the desired embedding.

Here, the matrix $\boldsymbol{S} - \tilde{\boldsymbol{K}}$ is the so-called graph Laplacian, and it can be shown (Belkin and Niyogi, 2003a) to be an approximation of the manifold Laplace Beltrami operator, when using the Gaussian kernel or the k-nearest-neighbor kernel for the similarity $\tilde{k}(\cdot, \cdot)$

on the graph. The k-nearest-neighbor kernel is represented by the *symmetric* matrix $\tilde{\boldsymbol{K}}$ whose element (i, j) is 1 if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are *k-nearest-neighbors* ($\boldsymbol{x}_i$ is among the k nearest neighbors of $\boldsymbol{x}_j$ or vice versa) and 0 otherwise. Approximating the Laplace Beltrami operator is motivated by the fact that its eigenfunctions are mappings that optimally preserve the "locality" of data (Belkin and Niyogi, 2003a).

It turns out that the above algorithm results in the same embedding (up to scaling) that is computed with the spectral clustering algorithm from (Shi and Malik, 1997) described in section 2.3: as noted in (Weiss, 1999) (Normalization Lemma 1), an equivalent result (up to a component-wise scaling of the embedding) can be obtained by considering the principal eigenvectors $\boldsymbol{v}_r$ of the normalized matrix $\boldsymbol{K}$ defined in eq. 11. To fit the common framework for spectral embedding in this chapter, we have used the latter formulation. Therefore, the same data-dependent kernel can be defined as for spectral clustering (eq. 12) to generate the matrix $\boldsymbol{K}$, i.e. spectral clustering just adds a clustering step after a Laplacian eigenmaps dimensionality reduction.

2.5 Isomap

Isomap (Tenenbaum, de Silva and Langford, 2000) generalizes MDS (section 2.2) to non-linear manifolds. It is based on replacing the Euclidean distance by an empirical approximation of the geodesic distance on the manifold. We define the *geodesic distance* $\hat{d}(\cdot, \cdot)$ *with respect to a data set* $\boldsymbol{X}$*, a distance* $d(\cdot, \cdot)$ *and a neighborhood* k as follows:

$$\hat{d}(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{\pi}} \sum_{i=1}^{|\boldsymbol{\pi}|} d(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i+1}) \tag{14}$$

where $\boldsymbol{\pi}$ is a sequence of points of length $|\boldsymbol{\pi}| = l \geq 2$ with $\boldsymbol{\pi}_1 = \boldsymbol{x}$, $\boldsymbol{\pi}_l = \boldsymbol{y}$, $\boldsymbol{\pi}_i \in \boldsymbol{X}\ \forall i \in \{2, \ldots, l-1\}$ and $(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{i+1})$ are k-nearest-neighbors of each other. The length $|\boldsymbol{\pi}| = l$ is free in the minimization. The Isomap algorithm obtains the normalized matrix $\boldsymbol{K}$ from which the embedding is derived by transforming the raw pairwise distances matrix as follows: (1) compute the matrix $\tilde{K}_{ij} = \hat{d}^2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ of squared geodesic distances with respect to the data $\boldsymbol{X}$ and (2) apply to this matrix the double-centering transformation (eq. 8), as for MDS. As in MDS, the embedding of $\boldsymbol{x}_i$ is given by eq. 3 with $v_{r,i}$ multiplied by $\sqrt{\ell_r}$. Step (1) can be done in $O(n^3)$ operations very easily (e.g. by Floyd's algorithm), but in (Tenenbaum, de Silva and Langford, 2000) it is suggested to use more efficient algorithms exploiting the sparse structure of the neighborhood graph, such as those presented in (Kumar et al., 1994).

There are several ways to define a kernel that generates $\boldsymbol{K}$ and also generalizes out-of-sample. The solution we have chosen simply computes the geodesic distances without involving the out-of-sample point(s) along the geodesic distance sequence (except for the last distance). This is automatically achieved with the above definition of geodesic distance $\hat{d}$, which only uses the training points to find the shortest path between $\boldsymbol{x}$ and $\boldsymbol{y}$. The double-centering kernel transformation of eq. 10 can then be applied to obtain k_m, using the geodesic distance $\hat{d}$ instead of the MDS distance d.

2.6 Locally Linear Embedding

The Locally Linear Embedding (LLE) algorithm (Roweis and Saul, 2000) looks for an embedding that preserves the local geometry in the neighborhood of each data point. The idea is to find a low-dimensional representation where the reconstruction of a data point from its neighbors is similar to the one in input space. First, a sparse matrix of local predictive weights W_{ij} is computed, such that $\sum_j W_{ij} = 1$, $W_{ii} = 0$, $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest-neighbor of $\boldsymbol{x}_i$ and $\|(\sum_j W_{ij}\boldsymbol{x}_j) - \boldsymbol{x}_i\|^2$ is minimized. To find those weights, for a given training point $\boldsymbol{x}_i$ with neighbors $(\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{ik})$, a local Gram matrix $\boldsymbol{K}^{(i)}$ is computed, such that $K^{(i)}_{rs} = \langle \boldsymbol{y}_{ir} - \boldsymbol{x}_i, \boldsymbol{y}_{is} - \boldsymbol{x}_i \rangle$. To improve the condition of this Gram matrix (to avoid potential issues when solving the linear system below), it is recommended to add a small multiple of the identity matrix:

$$K^{(i)}_{rs} \leftarrow K^{(i)}_{rs} + \delta_{rs}\frac{\Delta^2}{k} Tr(\boldsymbol{K}^{(i)})$$

with Tr the trace operator, δ the Kronecker symbol, and $\Delta^2 \ll 1$. The weights are then obtained by solving the linear system defined by $\sum_r K^{(i)}_{rs} W_{ir} = 1$ for all s, then rescaling the W_{ir} so that they sum to 1 (Saul and Roweis, 2002).

From the weights W_{ij}, the matrix $\tilde{\boldsymbol{K}} = (\boldsymbol{I} - \boldsymbol{W})^T(\boldsymbol{I} - \boldsymbol{W})$ is formed. The embedding is obtained from the lowest eigenvectors of $\tilde{\boldsymbol{K}}$, except for the eigenvector with the smallest eigenvalue, which is uninteresting because it is proportional to $(1, 1, \ldots, 1)$ (and its eigenvalue is 0). Since we want to select the principal eigenvectors, we define our normalized matrix by $\boldsymbol{K} = c\boldsymbol{I} - \tilde{\boldsymbol{K}}$ (c being any real number) and ignore the top eigenvector (although one could apply an additive normalization to remove the components along the $(1, 1, \ldots, 1)$ direction). The LLE embedding for $\boldsymbol{x}_i$ is then given by eq. 3 (multiplied by $\sqrt{m}$), starting at the second eigenvector (since the principal one is constant). If one insists on having a positive semi-definite matrix $\boldsymbol{K}$, one can take for c the largest eigenvalue of $\tilde{\boldsymbol{K}}$ (note that c only changes the eigenvalues additively and has no influence on the embedding of the training set).

In order to define a kernel k_m generating $\boldsymbol{K}$, we first denote by $w(\boldsymbol{x}, \boldsymbol{x}_i)$ the weight of $\boldsymbol{x}_i$ in the reconstruction of any point $\boldsymbol{x} \in \mathbb{R}^n$ by its k nearest neighbors in the training set. This is the same reconstruction as above, i.e. the $w(\boldsymbol{x}, \boldsymbol{x}_i)$ are such that they sum to 1, $\|(\sum_i w(\boldsymbol{x}, \boldsymbol{x}_i)\boldsymbol{x}_i) - \boldsymbol{x}\|^2$ is minimized, and $w(\boldsymbol{x}, \boldsymbol{x}_i) = 0$ if $\boldsymbol{x}_i$ is not in the k nearest neighbors of $\boldsymbol{x}$. If $\boldsymbol{x} = \boldsymbol{x}_j \in \boldsymbol{X}$, we have $w(\boldsymbol{x}, \boldsymbol{x}_i) = \delta_{ij}$. Let us now define a kernel k'_m by $k'_m(\boldsymbol{x}_i, \boldsymbol{x}) = k'_m(\boldsymbol{x}, \boldsymbol{x}_i) = w(\boldsymbol{x}, \boldsymbol{x}_i)$ and $k'_m(\boldsymbol{x}, \boldsymbol{y}) = 0$ when neither $\boldsymbol{x}$ nor $\boldsymbol{y}$ is in the training set $\boldsymbol{X}$. Let k''_m be such that $k''_m(\boldsymbol{x}_i, \boldsymbol{x}_j) = W_{ij} + W_{ji} - \sum_k W_{ki}W_{kj}$ and $k''_m(\boldsymbol{x}, \boldsymbol{y}) = 0$ when either $\boldsymbol{x}$ or $\boldsymbol{y}$ is not in $\boldsymbol{X}$. It can be shown that the kernel $k_m = (c-1)k'_m + k''_m$ is then such that

$$k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) = (c-1)\delta_{ij} + W_{ij} + W_{ji} - \sum_k W_{ki}W_{kj} = K_{ij}$$

so that it can be used to generate $\boldsymbol{K}$. There could be other ways to obtain a data-dependent kernel for LLE that can be applied out-of-sample: a justification for using this specific kernel will be given in section 3.1.

As noted independently in (Ham et al., 2003), LLE can thus be seen as performing kernel PCA with a particular kernel matrix. This identification becomes even more accurate when one notes that getting rid of the constant eigenvector (principal eigenvector of $\boldsymbol{K}$) is equivalent to the centering operation in feature space required for kernel PCA (Ham et al., 2003).
It is interesting to note a recent descendant of Laplacian eigenmaps, Isomap and LLE, called Hessian eigenmaps (Donoho and Grimes, 2003), which considers the limit case of the continuum of the manifold, and replaces the Laplacian in Laplacian eigenmaps by a Hessian. Despite attractive theoretical properties, the Hessian eigenmaps algorithm, being based on estimation of second order derivatives (which is difficult with sparse noisy data), has yet to be applied successfully on real-world high-dimensional data.

2.7 Mixtures of Low-Rank Gaussians

Isomap and LLE are two instances of a larger family of unsupervised learning algorithms which characterize the data distribution by a large set of locally linear low-dimensional patches. A simple example of this type of model is a mixture of Gaussians (centered on each example in the standard non-parametric setting) whose covariance matrices are summarized by a few eigenvectors (i.e. principal directions). The mixture of factor analyzers (Ghahramani and Hinton, 1996) is a parametric version of this type of model, in which the EM algorithm is used to estimate the means and the low-rank covariance matrices. A non-parametric version of the mixture of factor analyzers aimed at capturing manifold structure is the Manifold Parzen Windows algorithm (Vincent and Bengio, 2003), which does not require an iterative algorithm for training. With such models, one can obtain a local low-dimensional representation of examples falling near a Gaussian center, but it may be incomparable to the representation obtained for a nearby Gaussian center, because the eigenvectors of the covariance matrices of neighboring Gaussians may not be aligned. In order to perform dimensionality reduction from such models, several algorithms have thus been proposed (Teh and Roweis, 2003; Brand, 2003; Verbeek, Roweis and Vlassis, 2004), which look for a global representation that agrees with each local patch. Although these algorithms do not fall into the "spectral manifold learning" family studied in more detailed in this chapter, they are very close in spirit.

3 Kernel Eigenfunctions for Induction

With the exception of kernel PCA, the spectral manifold learning algorithms presented in section 2 do not provide us with an immediate way to obtain the embedding for a new point $\boldsymbol{x} \notin \boldsymbol{X}$. However, for some of them, extensions have already been proposed. We briefly review them in section 3.1. In section 3.2, we take advantage of the common framework developed in section 2: each algorithm being associated with a data-dependent kernel k_m generating a Gram matrix $\boldsymbol{K}$, we can apply the Nyström formula (eq. 2) to obtain the embedding for a new point $\boldsymbol{x}$.

3.1 Extensions to Spectral Embedding Algorithms

For metric MDS, it is suggested in (Gower, 1968) to solve exactly for the coordinates of the new point such that its distances to the training points are the same in the original input space and in the computed embedding, but in general this requires adding a new dimension. Note also that (Williams, 2001) makes a connection between kernel PCA and metric MDS, remarking that kernel PCA is a form of MDS when the kernel is isotropic. In the following, we will pursue this connection in order to obtain out-of-sample embeddings.

A formula has been proposed (de Silva and Tenenbaum, 2003) to approximate Isomap using only a subset of the examples (the "landmark" points) to compute the eigenvectors. Using the notation of this chapter, that formula is

$$e_r(\boldsymbol{x}) = \frac{1}{2\sqrt{\ell_r}} \sum_i v_{r,i} (\hat{E}_{\boldsymbol{x}'}[\hat{d}^2(\boldsymbol{x}', \boldsymbol{x}_i)] - \hat{d}^2(\boldsymbol{x}_i, \boldsymbol{x})) \tag{15}$$

which is applied to obtain an embedding for the non-landmark examples. One can show (Bengio et al., 2004) that $e_r(\boldsymbol{x})$ is the Nyström formula when $k_m(\boldsymbol{x}, \boldsymbol{y})$ is defined as in section 2.5. Landmark Isomap is thus equivalent to performing Isomap on the landmark points only and then predicting the embedding of the other points using the Nyström formula, which is the solution we also propose in what follows.

For LLE, with the notations of section 2.6, an extension suggested in (Saul and Roweis, 2002) is to take for a new point $\boldsymbol{x}$ the embedding $\mathbf{P}(\boldsymbol{x}) = (P_1(\boldsymbol{x}), \ldots, P_N(\boldsymbol{x}))^T$, where

$$P_r(\boldsymbol{x}) = \sum_{i=1}^{m} P_r(\boldsymbol{x}_i) w(\boldsymbol{x}, \boldsymbol{x}_i).$$

Interestingly, the same embedding can be obtained from the Nyström formula and the kernel k_m defined in section 2.6, when the constant $c \to +\infty$ (Bengio et al., 2004).

3.2 From Eigenvectors to Eigenfunctions

From the common framework developed in section 2, one can see spectral algorithms as performing a kind of kernel PCA with a specific kernel. (Ng, Jordan and Weiss, 2002) had already noted the link between kernel PCA and spectral clustering. Recently, (Ham et al., 2003) have also shown how Isomap, LLE and Laplacian eigenmaps can be interpreted as performing a form of kernel PCA. Here, we propose a similar view, extending the framework to allow negative eigenvalues (which may be the case for Isomap). In addition, those papers did not propose to use this link in order to perform function induction, i.e. obtain an embedding for out-of-sample points. Indeed, since there exists a natural extension to new points for kernel PCA (the projection onto the eigenspaces of the covariance matrix, see eq. 7), it is natural to ask whether it makes sense to use such a formula in the more general setting where the kernel may be data-dependent and may have negative eigenvalues.

As noted in (Williams and Seeger, 2000), the kernel PCA projection formula (eq. 7) is proportional to the so-called Nyström formula (Baker, 1977; Williams and Seeger, 2000) (eq. 2), which has been used successfully to "predict" the value of an eigenvector on a new data point, in order to speed-up kernel methods computations by focusing the heavier computations (the eigen-decomposition) on a subset of examples (Williams and Seeger, 2001). The use of this formula can be justified by considering the convergence of eigenvectors and eigenvalues, as the number of examples increases (Baker, 1977; Koltchinskii, 1998; Koltchinskii and Giné, 2000; Williams and Seeger, 2000). In particular, (Shawe-Taylor, Cristianini and Kandola, 2002; Shawe-Taylor and Williams, 2003; Zwald, Bousquet and Blanchard, 2004) give bounds on the kernel PCA convergence error (in the sense of the projection error with respect to the subspace spanned by the eigenvectors), using concentration inequalities.

Based on this kernel PCA convergence results, we conjecture that in the limit, each eigenvector would converge to an eigenfunction for a linear operator (defined below), in the sense that the i-th element of the r-th eigenvector converges to the application of the r-th eigenfunction to $\boldsymbol{x}_i$. Proposition 2 below formalizes this statement and provides sufficient conditions for such a convergence.

In the following we will assume that the (possibly data-dependent) kernel k_m is *bounded* (i.e. $\exists k_{max}$, $\forall \boldsymbol{x}, \boldsymbol{y}$ $|k_m(\boldsymbol{x}, \boldsymbol{y})| < k_{max}$) and has a discrete spectrum, i.e. that it can be written as a discrete expansion

$$k_m(\boldsymbol{x}, \boldsymbol{y}) = \sum_{r=1}^{\infty} \alpha_{r,m} \psi_{r,m}(\boldsymbol{x}) \psi_{r,m}(\boldsymbol{y}).$$

Consider the space $\mathcal{H}_p$ of continuous functions f on $\mathbb{R}^n$ that are square integrable as follows:

$$\int f^2(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} < \infty$$

with the data-generating density function $p(\boldsymbol{x})$. One must note that we actually do not work on functions but on equivalence classes: we say two continuous functions f and g belong to the same equivalence class (with respect to p) if and only if $\int (f(\boldsymbol{x}) - g(\boldsymbol{x}))^2 p(\boldsymbol{x}) d\boldsymbol{x} = 0$ (if p is strictly positive, then each equivalence class contains only one function).

We will assume that k_m converges uniformly in its arguments (in some probabilistic manner, e.g. almost surely or in probability) to its limit k as $m \to \infty$. We will associate with each k_m a linear operator L_m and with k a linear operator L, such that for any $f \in \mathcal{H}_p$,

$$L_m f = \frac{1}{m} \sum_{i=1}^{m} k_m(\cdot, \boldsymbol{x}_i) f(\boldsymbol{x}_i) \tag{16}$$

and

$$Lf = \int k(\cdot, \boldsymbol{y}) f(\boldsymbol{y}) p(\boldsymbol{y}) d\boldsymbol{y} \tag{17}$$

which makes sense because we work in a space of functions defined everywhere. Furthermore, as $k_m(\cdot, \boldsymbol{y})$ and $k(\cdot, \boldsymbol{y})$ are square-integrable in the sense defined above, for

each f and each m, the functions $L_m f$ and Lf are square-integrable in the sense defined above. We will show that the Nyström formula (eq. 2) gives the eigenfunctions of L_m (proposition 1), that their value on the training examples corresponds to the spectral embedding, and that they converge to the eigenfunctions of L (proposition 2). These results will hold even if k_m has negative eigenvalues.
The eigensystems of interest are thus the following:

$$Lf_r = \lambda_r f_r \tag{18}$$

and

$$L_m f_{r,m} = \lambda_{r,m} f_{r,m} \tag{19}$$

where (λ_r, f_r) and $(\lambda_{r,m}, f_{r,m})$ are the corresponding eigenvalues and eigenfunctions.
Note that when eq. 19 is evaluated only at the $\boldsymbol{x}_i \in \boldsymbol{X}$, the set of equations reduces to the eigensystem

$$\boldsymbol{K}\boldsymbol{v}_r = m\lambda_{r,m}\boldsymbol{v}_r.$$

The following proposition gives a more complete characterization of the eigenfunctions of L_m, even in the case where eigenvalues may be negative. The next two propositions formalize the link already made in (Williams and Seeger, 2000) between the Nyström formula and eigenfunctions of L.

Proposition 1 *L_m has in its image $N \leq m$ eigenfunctions of the form:*

$$f_{r,m}(\boldsymbol{x}) = \frac{\sqrt{m}}{\ell_r} \sum_{i=1}^{m} v_{r,i} k_m(\boldsymbol{x}, \boldsymbol{x}_i) \tag{20}$$

with corresponding non-zero eigenvalues $\lambda_{r,m} = \frac{\ell_r}{m}$, where $\boldsymbol{v}_r = (v_{r,1}, \ldots, v_{r,m})^T$ is the r-th eigenvector of the Gram matrix $\boldsymbol{K}$, associated with the eigenvalue ℓ_r.
For $\boldsymbol{x}_i \in \boldsymbol{X}$ these functions coincide with the corresponding eigenvectors, in the sense that $f_{r,m}(\boldsymbol{x}_i) = \sqrt{m} v_{r,i}$.

Proof
First, we show that the $f_{r,m}$ defined by eq. 20 coincide with the eigenvectors of $\boldsymbol{K}$ at $\boldsymbol{x}_i \in \boldsymbol{X}$. For $f_{r,m}$ associated with a non-zero eigenvalue,

$$f_{r,m}(\boldsymbol{x}_i) = \frac{\sqrt{m}}{\ell_r} \sum_{j=1}^{m} \boldsymbol{v}_{r,j} k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sqrt{m}}{\ell_r} \ell_r v_{r,i} = \sqrt{m} v_{r,i}. \tag{21}$$

The $\boldsymbol{v}_r$ being orthonormal the $f_{r,m}$ (for different values of r) are therefore different from each other.
Then for any $\boldsymbol{x} \in \mathbb{R}^n$

$$(L_m f_{r,m})(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} k_m(\boldsymbol{x}, \boldsymbol{x}_i) f_{r,m}(\boldsymbol{x}_i) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} k_m(\boldsymbol{x}, \boldsymbol{x}_i) v_{r,i} = \frac{\ell_r}{m} f_{r,m}(\boldsymbol{x}) \tag{22}$$

which shows that $f_{r,m}$ is an eigenfunction of L_m with eigenvalue $\lambda_{r,m} = \ell_r/m$. $\square$

Discussion
The previous result shows that the Nyström formula generalizes the spectral embedding outside of the training set. This means the embedding $\mathbf{P}(\boldsymbol{x}) = (P_1(\boldsymbol{x}), \ldots, P_N(\boldsymbol{x}))^T$ for a new point $\boldsymbol{x}$ is given (up to some scaling) by

$$P_r(\boldsymbol{x}) = \frac{f_{r,m}(\boldsymbol{x})}{\sqrt{m}} = \frac{1}{\ell_r} \sum_{i=1}^{m} v_{r,i} k_m(\boldsymbol{x}, \boldsymbol{x}_i) \tag{23}$$

where the scaling is the same as the one described in section 2 (so that the embedding obtained on the training set is coherent with the one obtained from the eigenvectors, thanks to eq. 21).
However, there could be many possible generalizations. To justify the use of this particular generalization, the following proposition helps to understand the convergence of these functions as m increases. We would like the out-of-sample embedding predictions obtained with the Nyström formula to be somehow close to the asymptotic embedding (the embedding one would obtain as $m \to \infty$).
Note also that the convergence of eigenvectors to eigenfunctions shown in (Baker, 1977) applies to data $\boldsymbol{x}_i$ which are deterministically chosen to span a domain, whereas here the $\boldsymbol{x}_i$ form a random sample from an unknown distribution.

Proposition 2 *If $k_m = k$ is bounded and not data-dependent, then the eigenfunctions $f_{r,m}$ of L_m associated with non-zero eigenvalues of multiplicity 1 converge to the corresponding eigenfunctions of L (almost surely, and up to the sign).*
For k_m data-dependent but bounded (almost surely, and independently of m) and converging uniformly to k, if the eigen-decomposition of the Gram matrix $(k_m(\boldsymbol{x}_i, \boldsymbol{x}_j))$ converges[4] to the eigen-decomposition of the Gram matrix $(k(\boldsymbol{x}_i, \boldsymbol{x}_j))$ then a similar result holds: the eigenfunctions $f_{r,m}$ of L_m associated with non-zero eigenvalues of multiplicity 1 converge to the corresponding eigenfunctions of L (almost surely, and up to the sign).

Proof
In the following, we will denote by $\hat{f} \in \mathcal{H}_{\hat{p}}$ the restriction of a function $f \in \mathcal{H}_p$ to the training set $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, and by $\hat{L}_m$ the operator in $\mathcal{H}_{\hat{p}}$ defined as in eq. 16, which has the same eigenvalues and eigenfunctions as L_m (except the eigenfunctions are restricted to $\boldsymbol{X}$). We start with the case where $k_m = k$. We first take advantage of (Koltchinskii and Giné, 2000), theorem 3.1, that shows that the distance between the eigenvalue spectra of $\hat{L}_m$ and L converges to 0 almost surely. We then use theorem 2.1 from (Koltchinskii, 1998), which is stated as follows. Let k be a symmetric kernel such that $E[|k(X,X)|] < +\infty$ and $E[k^2(X,Y)] < +\infty$ (so that the operator L defined by eq. 17 is Hilbert-Schmidt and k can be written $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in I} \mu_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{y})$ with I a discrete set). Suppose that $\mathcal{F}$ is a class of measurable functions such that there exists

[4]The convergences should be almost sure, otherwise the result may hold with a different kind of probabilistic convergence, e.g. in probability.

$F \in \mathcal{H}_p$ verifying $|f(\boldsymbol{x})| \leq F(\boldsymbol{x})$ for all $f \in \mathcal{F}$. Moreover, suppose that for all $i \in I$, $\{f\psi_i : f \in \mathcal{F}\} \in GC(p)$, where $GC(p)$ denotes the set of p-Glivenko-Cantelli classes (see, e.g., (van der Vaart and Wellner, 1996)). Then, for all non-zero eigenvalue λ_r

$$\sup_{f,g \in \mathcal{F}} \left| \langle P_r(\hat{L}_m)\hat{f}, \hat{g} \rangle_{\mathcal{H}_{\hat{p}}} - \langle P_r(L)f, g \rangle_{\mathcal{H}_p} \right| \to 0 \tag{24}$$

almost surely when $m \to +\infty$, with $P_r(L)$ the projection on the r-th eigenspace of L, and $P_r(\hat{L}_m)$, with probability 1 and for m sufficiently large, the projection on the corresponding eigenspace of $\hat{L}_m$ (for more details see (Koltchinskii, 1998)).
Let us consider the r-th eigenspace of L (of dimension 1 because we have considered eigenvalues of multiplicity 1), i.e. the eigenspace spanned by the eigenfunction f_r: the r-th eigenspace of $\hat{L}_m$ is also 1-dimensional, almost surely (because of the convergence of the spectrum), and spanned by $f_{r,m}$. Let $\boldsymbol{x} \in \mathbb{R}^n$ be any point in the input space, and $\mathcal{F} = \{h_{\boldsymbol{x}}\}$ with $h_{\boldsymbol{x}} = k(\boldsymbol{x}, \cdot) \in \mathcal{H}_p$. For any $i \in I$

$$\left| \frac{1}{m} \sum_{j=1}^{m} h_{\boldsymbol{x}}(\boldsymbol{x}_j)\psi_i(\boldsymbol{x}_j) - \int h_{\boldsymbol{x}}(\boldsymbol{y})\psi_i(\boldsymbol{y})p(\boldsymbol{y})d\boldsymbol{y} \right| \to 0$$

almost surely (thanks to the strong law of large numbers), so that $\mathcal{F}$ verifies the hypothesis needed to apply the theorem above. In addition,

$$\langle P_r(L)h_{\boldsymbol{x}}, h_{\boldsymbol{x}} \rangle_{\mathcal{H}_p} = \langle \langle h_{\boldsymbol{x}}, f_r \rangle f_r, h_{\boldsymbol{x}} \rangle = \langle h_{\boldsymbol{x}}, f_r \rangle^2_{\mathcal{H}_p} = (Lf_r)(\boldsymbol{x})^2 = \lambda_r^2 f_r(\boldsymbol{x})^2$$

and similarly, using eq. 22, we have with probability 1 and for m large enough:

$$\langle P_r(\hat{L}_m)\hat{h}_{\boldsymbol{x}}, \hat{h}_{\boldsymbol{x}} \rangle_{\mathcal{H}_{\hat{p}}} = \langle \hat{h}_{\boldsymbol{x}}, \hat{f}_{r,m} \rangle^2_{\mathcal{H}_{\hat{p}}} = (L_m f_{r,m})(\boldsymbol{x})^2 = \lambda_{r,m}^2 f_{r,m}(\boldsymbol{x})^2. \tag{25}$$

The conclusion of the theorem thus tells us that

$$\left| \lambda_{r,m}^2 f_{r,m}(\boldsymbol{x})^2 - \lambda_r^2 f_r(\boldsymbol{x})^2 \right| \to 0$$

almost surely. Since we have the convergence of the eigenvalues, this implies

$$\left| f_{r,m}(\boldsymbol{x})^2 - f_r(\boldsymbol{x})^2 \right| \to 0 \tag{26}$$

almost surely, which shows the (simple) convergence of the eigenfunctions, up to the sign. To get the convergence in $\mathcal{H}_p$, we need to show that $\| |f_{r,m}| - |f_r| \|_{\mathcal{H}_p} \to 0$, i.e.

$$\int g_{r,m}(\boldsymbol{x})d\boldsymbol{x} \to 0 \tag{27}$$

with $g_{r,m}(\boldsymbol{x}) = (|f_{r,m}(\boldsymbol{x})| - |f_r(\boldsymbol{x})|)^2 p(\boldsymbol{x})$. We will need to show that both $f_{r,m}$ and f_r are bounded (independently of m). Since f_r is an eigenfunction of L, we have $|\lambda_r f_r(\boldsymbol{x})| = |(Lf_r)(\boldsymbol{x})| = |\int k(\boldsymbol{x}, \boldsymbol{y}) f_r(\boldsymbol{y}) p(\boldsymbol{y}) d\boldsymbol{y}| \leq c |\int f_r(\boldsymbol{y}) p(\boldsymbol{y}) d\boldsymbol{y}|$, so that $|f_r(\boldsymbol{x})| \leq c'_r$. For $f_{r,m}$, we have

$$|\lambda_{r,m} f_{r,m}(\boldsymbol{x})| = |(L_m f_{r,m}(\boldsymbol{x}))| \quad = \quad \left| \frac{1}{m} \sum_{i=1}^{m} k(\boldsymbol{x}, \boldsymbol{x}_i) \sqrt{m} v_{r,i} \right| \leq \frac{c}{\sqrt{m}} \sum_{i=1}^{m} |v_{r,i}| \leq c$$

because the maximum of $\sum_{i=1}^{m} |v_{r,i}|$ subject to $\|\boldsymbol{v}_r\|^2 = 1$ is $\sqrt{m}$, so that $|f_{r,m}(\boldsymbol{x})| \leq c_r''$ almost surely (thanks to the convergence of $\lambda_{r,m}$). Therefore, we have that (almost surely) $g_{r,m}(\boldsymbol{x}) \leq (c_r' + c_r'')^2 p(\boldsymbol{x})$ which is an integrable function, and from eq. 26, $g_{r,m}(\boldsymbol{x}) \to 0$ for all $\boldsymbol{x}$. The theorem of dominated convergence can thus be applied, which proves eq. 27 is true (almost surely), and there is convergence of the eigenfunctions in $\mathcal{H}_p$.
If k_m is data-dependent but converges, in a way such that the eigen-decomposition of the Gram matrix $(k_m(\boldsymbol{x}_i, \boldsymbol{x}_j))$ converges to the eigen-decomposition of the Gram matrix $(k(\boldsymbol{x}_i, \boldsymbol{x}_j))$, with k the limit of k_m, we want to apply the same reasoning. Because of the convergence of the eigen-decomposition of the Gram matrix, eq. 24 still holds. However, eq. 25 has to be replaced with a limit, because $h_{\boldsymbol{x}} = k(\boldsymbol{x}, \cdot) \neq k_m(\boldsymbol{x}, \cdot)$. This limit still allows to write eq. 26 (possibly with a different form of probabilistic convergence, depending on the convergence of k_m to k), and the same result is obtained. $\square$

Discussion
Kernel PCA has already been shown to be a stable and convergent algorithm (Shawe-Taylor, Cristianini and Kandola, 2002; Shawe-Taylor and Williams, 2003; Zwald, Bousquet and Blanchard, 2004). These papers characterize the rate of convergence of the projection error on the subspace spanned by the first N eigenvectors of the feature space covariance matrix. When we perform the PCA or kernel PCA projection on an out-of-sample point, we are taking advantage of the above convergence and stability properties: we trust that a principal eigenvector of the empirical covariance matrix estimates well a corresponding eigenvector of the true covariance matrix. Another justification for applying the Nyström formula outside of the training examples is therefore, as already noted earlier and in (Williams and Seeger, 2000), *in the case where k_m is positive semi-definite*, that it corresponds to the kernel PCA projection (on a corresponding eigenvector of the feature space covariance matrix $\boldsymbol{C}$).
Clearly, we thus have with the Nyström formula a method to *generalize spectral embedding algorithms to out-of-sample examples,* whereas the original spectral embedding methods only provide the transformed coordinates of training points (i.e. an embedding of the training points). The experiments described in section 5 show empirically the good generalization of this out-of-sample embedding. Note however that it is not always clear whether the assumptions needed to apply proposition 2 are verified or not (especially because of the data-dependency of k_m). This proposition mainly gives an intuition of what a spectral embedding technique is doing (estimating eigenfunctions of a linear operator) in the case of ideal convergence.
(Williams and Seeger, 2000) have shown an interesting justification for estimating the eigenfunctions of L. When an unknown function f is to be estimated with an approximation g that is a finite linear combination of basis functions, if f is assumed to come from a zero-mean Gaussian process prior with covariance $E_f[f(\boldsymbol{x})f(\boldsymbol{y})] = k(\boldsymbol{x}, \boldsymbol{y})$, then the best choices of basis functions, in terms of expected squared error, are (up to rotation/scaling) the leading eigenfunctions of the linear operator L defined by eq 17.

4 Learning Criterion for the Leading Eigenfunctions

Using an expansion into orthonormal bases (e.g. generalized Fourier decomposition in the case where p is continuous), the best approximation of $k(\boldsymbol{x}, \boldsymbol{y})$ (in the sense of minimizing expected squared error) using only N terms is the expansion that uses the *first* N eigenfunctions of L (in the order of decreasing eigenvalues):

$$\sum_{r=1}^{N} \lambda_r f_r(\boldsymbol{x}) f_r(\boldsymbol{y}) \approx k(\boldsymbol{x}, \boldsymbol{y}).$$

This simple observation allows us to define a *loss criterion for spectral embedding algorithms*, something that was lacking up to now for such algorithms. The limit of this loss converges toward an expected loss whose minimization gives rise to the eigenfunctions of L. One could thus conceivably estimate this generalization error using the average of the loss on a test set. That criterion is simply the kernel reconstruction error

$$R_{k_m}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{r=1}^{N} \lambda_{r,m} f_{r,m}(\boldsymbol{x}_i) f_{r,m}(\boldsymbol{x}_j) \right)^2 .$$

Proposition 3 *The spectral embedding for a continous kernel k with discrete spectrum is the solution of a sequential minimization problem, iteratively minimizing for $N = 1, 2, \ldots$ the expected value of the loss criterion*

$$R_k(\boldsymbol{x}, \boldsymbol{y}) = \left(k(\boldsymbol{x}, \boldsymbol{y}) - \sum_{r=1}^{N} \lambda_r f_r(\boldsymbol{x}) f_r(\boldsymbol{y}) \right)^2 .$$

First, with $\{(f_r, \lambda_r)\}_{k=1}^{N-1}$ already obtained, one gets recursively (λ_N, f_N) by minimizing

$$J_N(\lambda', f') = \int \left(k(\boldsymbol{x}, \boldsymbol{y}) - \lambda' f'(\boldsymbol{x}) f'(\boldsymbol{y}) - \sum_{r=1}^{N-1} \lambda_r f_r(\boldsymbol{x}) f_r(\boldsymbol{y}) \right)^2 p(\boldsymbol{x}) p(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \qquad (28)$$

where by convention we scale f' such that $\int f'(\boldsymbol{x})^2 p(\boldsymbol{x}) = 1$ (any other scaling can be transferred into λ').
Secondly, if the same hypothesis on k_m as in proposition 2 are verified, the Monte-Carlo average of the criterion R_{k_m}

$$\frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \left(k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{r=1}^{N} \lambda_{r,m} f_{r,m}(\boldsymbol{x}_i) f_{r,m}(\boldsymbol{x}_j) \right)^2$$

converges in probability to the asymptotic expectation of R_k.

Sketch of proof
The first part of the proposition concerning the sequential minimization of the loss criterion follows from classical linear algebra (Strang, 1980; Kreyszig, 1990). It is an

extension of the well-known result stating that the best rank N approximation (for the Frobenius norm) of a symmetric matrix is given by its expansion in terms of its first N eigenvectors. A proof can be found in (Bengio et al., 2003).
To prove the second part, a reasoning similar to the one in the proof of proposition 2 can be done, in order to obtain that (in probability, when $m \to \infty$)

$$\int \left(\lambda_{r,m} f_{r,m}(\boldsymbol{x}) f_{r,m}(\boldsymbol{y}) - \lambda_r f_r(\boldsymbol{x}) f_r(\boldsymbol{y})\right) p(\boldsymbol{x}) p(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \to 0$$

which, combined with the central limit theorem, leads to the desired convergence. □

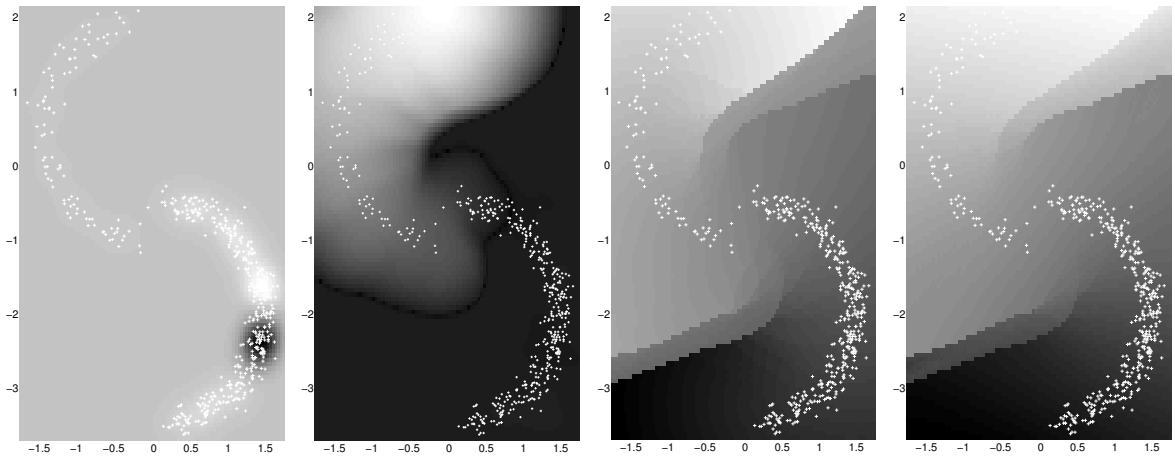
Discussion
Note that the empirical criterion is indifferent to the value of the solutions $f_{r,m}$ outside of the training set. Therefore, although the Nyström formula gives a possible solution to the empirical criterion, there are other solutions. Remember that the task we consider is that of estimating the eigenfunctions of L, i.e. approximating a similarity function k where it matters according to the unknown density p. Solutions other than the Nyström formula might also converge to the eigenfunctions of L. For example one could use a non-parametric estimator (such as a neural network) to estimate the eigenfunctions. Even if such a solution does not yield the exact eigenvectors on the training examples (i.e. does not yield the lowest possible error on the training set), it might still be a good solution in terms of generalization, in the sense of good approximation of the eigenfunctions of L. It would be interesting to investigate whether the Nyström formula achieves the fastest possible rate of convergence to the eigenfunctions of L.

5 Experiments

5.1 Toy Data Example

We first show on a toy dataset what kind of structure can be discovered from the eigenfunctions defined in the previous sections. In figure 2, we display with gray levels the value[5] of the first eigenfunction computed for kernel PCA, spectral clustering (same as Laplacian eigenmaps), Isomap and LLE, on a toy dataset of 500 examples (remember that the first eigenfunction is proportional to the first coordinate of the projection, as seen in eq. 23). This toy dataset is formed of two clusters (white dots) with a similar form, one (top-left) with 100 points, and the other (bottom-right) with 400 points. The clusters are connected (through a low-density region) in such a way that the data lie approximately on a one-dimensional manifold. Our C++ code for performing those spectral dimensionality reduction algorithms can be found in the PLearn library (`http://plearn.org`).
Although such a toy example does not provide a deep understanding of what these algorithms do, it reveals several characteristics worth pointing out. First of all, kernel

[5]In the case of spectral clustering, this is the logarithm of the eigenfunction that is displayed, so as to be able to actually see its variations.



(a) *Kernel PCA* (b) *Spectral clustering* (c) *Isomap* (d) *LLE*

Figure 2: *First eigenfunction (gray levels) for various non-linear spectral dimensionality reduction algorithms, on a toy dataset of 500 samples (white dots). Kernel PCA and spectral clustering use the same Gaussian kernel (eq. 4) with bandwidth* $\sigma = 0.2$*, while Isomap and LLE use 20 neighbors.*

PCA should not be used with a Gaussian kernel in order to discover a low-dimensional non-linear manifold. Indeed, one may think kernel PCA does some kind of "local" PCA within neighborhoods of size of the order of the Gaussian kernel's bandwidth, but it is *not* the case. It actually tends to discriminate smaller regions of the data as the bandwidth decreases (and a high bandwidth makes it equivalent to linear PCA). The spectral clustering / Laplacian eigenmaps eigenfunction is more satisfying, in that it obviously reveals the clustered nature of our data (see footnote 5). Note that, even though in this case only one eigenfunction may be enough to discriminate between the two clusters, one should in general compute as many eigenfunctions as desired clusters (because each eigenfunction will tend to map to the same point all clusters but one, e.g. in figure 2(b) almost all points in the bottom-right cluster are given the same value). As for Isomap and LLE, they give very similar results, showing they correctly captured the underlying one-dimensional non-linear manifold. Although it cannot be seen in this particular example, one should keep in mind that LLE, because it is based only on local computations, does not respect as well as Isomap the global structure of the data.

5.2 Discovering Structure in Face Images

Experiments in this section are done over a subset of the database of 698 synthetic face images available at `http://isomap.stanford.edu`. By selecting only images whose illumination is between 180 and 200, this yields a dataset of 113 examples in 4096 dimensions, which approximately form a 2-dimensional manifold (the two degrees of freedom are the rotation angles of the camera). The Isomap algorithm with 10 nearest

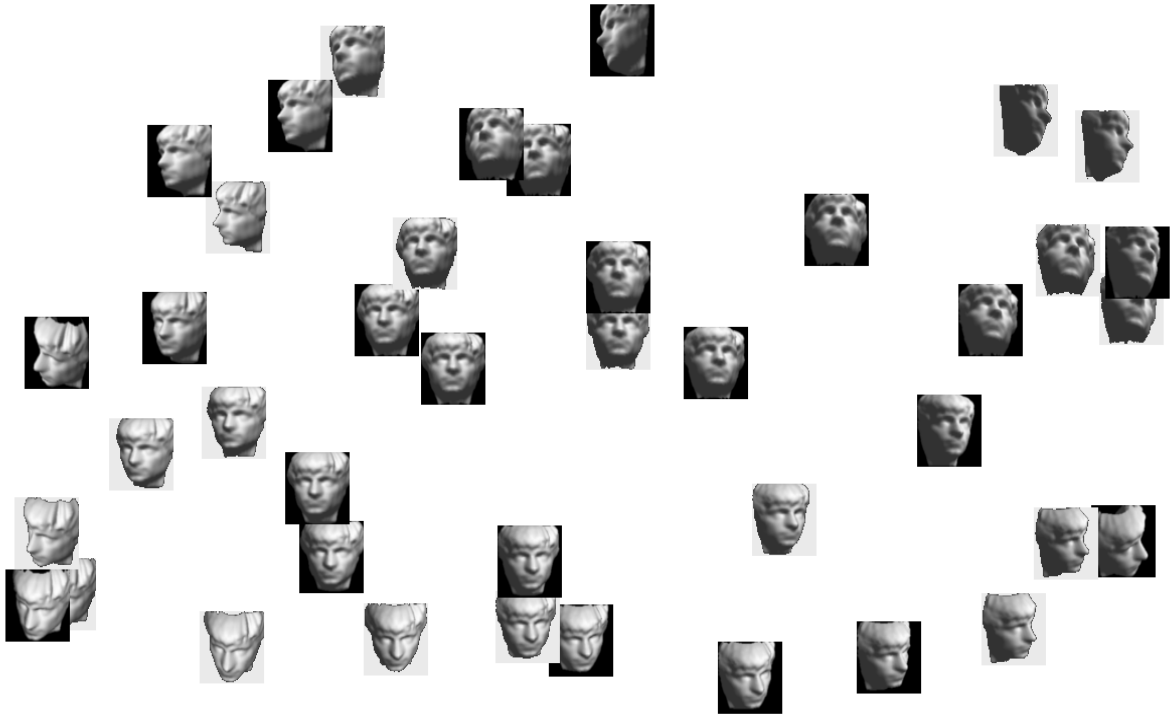


Figure 3: *2-dimensional embedding learned by Isomap from 70 high-dimensional synthetic faces. A few faces from the train set (black background) and from the test set (light gray background) are projected using the Nyström formula.*

neighbors is run on the first 70 examples, while the remaining 43 are projected by the Nyström formula. The embedding thus obtained (figure 3) clearly demonstrates that Isomap captured the intrinsic 2-dimensional manifold, and that the Nyström formula generalizes well. This is a typical example where such a non-linear spectral embedding algorithm can prove very useful for data visualization as well as for dimensionality reduction.

5.3 Generalization Performance of Function Induction

Here we want to test one aspect of the theoretical results: does the function induction achieved with the Nyström formula work well? We would like to know if the embedding that it predicts on a new point $\boldsymbol{x}$ is close to the embedding that would have been obtained on $\boldsymbol{x}$ if it had been in the training set. However, how do we evaluate the "error" thus obtained? Our idea is to compare it to the variations in embedding that result from small perturbations of the training set (such as replacing a subset of the examples by others from the same distribution).
For this purpose we consider splits of the data in three sets, $\boldsymbol{X} = \boldsymbol{F} \cup \boldsymbol{R}_1 \cup \boldsymbol{R}_2$ and

training either with $\boldsymbol{F} \cup \boldsymbol{R}_1$ or $\boldsymbol{F} \cup \boldsymbol{R}_2$, comparing the embeddings on $\boldsymbol{F}$. For each algorithm described in section 2, we apply the following procedure:

1. We choose $\boldsymbol{F} \subset \boldsymbol{X}$ with $q = |\boldsymbol{F}|$ samples. The remaining $m - q$ samples in $\boldsymbol{X}/\boldsymbol{F}$ are split into two equal size subsets $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$. We train (obtain the eigenvectors) over $\boldsymbol{F} \cup \boldsymbol{R}_1$ and $\boldsymbol{F} \cup \boldsymbol{R}_2$ and we calculate the Euclidean distance between the aligned embeddings obtained for each $\boldsymbol{x}_i \in \boldsymbol{F}$. When eigenvalues are close, the estimated eigenvectors are unstable and can rotate in the subspace they span. Thus we estimate an alignment (by linear regression) between the two embeddings using the points in $\boldsymbol{F}$.

2. For each sample $\boldsymbol{x}_i \in \boldsymbol{F}$, we also train over $\{\boldsymbol{F} \cup \boldsymbol{R}_1\}/\{\boldsymbol{x}_i\}$. We apply the Nyström formula to out-of-sample points to find the predicted embedding of $\boldsymbol{x}_i$ and calculate the Euclidean distance between this embedding and the one obtained when training with $\boldsymbol{F} \cup \boldsymbol{R}_1$, i.e. with $\boldsymbol{x}_i$ in the training set (in this case no alignment is done since the influence of adding a single point is very limited).

3. We calculate the mean **difference** δ (and its standard error) between the distance obtained in step 1 and the one obtained in step 2 for each sample $\boldsymbol{x}_i \in \boldsymbol{F}$, and we repeat this experiment for various sizes of $\boldsymbol{F}$.

The results obtained for MDS, Isomap, spectral clustering and LLE are shown in figure 4 for different values of $|\boldsymbol{R}_1|/m$ (i.e the fraction of points exchanged). The vertical axis is δ, the difference between perturbation error and induction error. The horizontal zero line corresponds to no difference between the embedding error due to induction (vs transduction) and the embedding error due to training set perturbation. For values of δ above the zero line, the embedding error due to perturbation is greater than the embedding error due to out-of-sample prediction. Clearly, the in-sample (transduction) vs out-of-sample (induction) difference is of the same order of magnitude as the change in embedding due to exchanging a small fraction of the data (1 to 5%).
Experiments are done over the same faces database as in the previous section, but with the whole set of 698 images. Similar results have been obtained over other databases such as Ionosphere[6] and Swissroll[7]. Each algorithm generates a two-dimensional embedding of the images, following the experiments reported for Isomap. The number of neighbors is 10 for Isomap and LLE, and a Gaussian kernel with a bandwidth of 0.01 is used for spectral clustering / Laplacian eigenmaps. 95% confidence intervals are drawn beside each mean difference of error on the figure.
As expected, the mean difference between the two distances is almost monotonically increasing as the number $|\boldsymbol{R}_1|$ of substituted training samples grows, mostly because the training set embedding variability increases. We find in most cases that the out-of-sample error is less than or comparable to the training set embedding instability when around 2% of the training examples are substituted randomly.

[6] `http://www.ics.uci.edu/~mlearn/MLSummary.html`
[7] `http://www.cs.toronto.edu/~roweis/lle/`

6 Conclusion

Manifold learning and dimensionality reduction are powerful machine learning tools for which much progress has been achieved in recent years. This chapter sheds light on a family of such algorithms, involving spectral embedding, which are all based on the eigen-decomposition of a similarity matrix.

Spectral embedding algorithms such as spectral clustering, Isomap, LLE, metric MDS, and Laplacian eigenmaps are very interesting dimensionality reduction or clustering methods. However, they lacked up to now a notion of generalization that would allow to easily extend the embedding out-of-sample without again solving an eigensystem. This chapter has shown with various arguments that the well known Nyström formula can be used for this purpose, and that it thus represents the result of a *function induction* process. These arguments also help us to understand that *these methods do essentially the same thing, but with respect to different kernels*: they *estimate the eigenfunctions of a linear operator L* associated with a kernel and with the underlying distribution of the data. This analysis also shows that these methods are *minimizing an empirical loss*, and that the solutions toward which they converge are the minimizers of a corresponding expected loss, which thus defines what good generalization should mean, for these methods. It shows that these unsupervised learning algorithms can be extended into function induction algorithms. The Nyström formula is a possible extension but it does not exclude other extensions which might be better or worse estimators of the eigenfunctions of the asymptotic linear operator L. When the kernels are positive semi-definite, these methods can also be immediately seen as performing kernel PCA. Note that Isomap generally yields a Gram matrix with negative eigenvalues, and users of MDS, spectral clustering or Laplacian eigenmaps may want to use a kernel that is not guaranteed to be positive semi-definite. The analysis in this chapter can still be applied in that case, even though the kernel PCA analogy does not hold anymore. This is important to note because recent work (Laub and Müller, 2003) has shown that the coordinates corresponding to large negative eigenvalues can carry very significant semantics about the underlying objects. In fact, it is proposed in (Laub and Müller, 2003) to perform dimensionality reduction by projecting on the eigenvectors corresponding to the largest eigenvalues **in magnitude** (i.e. irrespective of sign).

In this chapter we have given theorems that provide justification for the Nyström formula in the general case of data-dependent kernels which may not be positive-definite. However, these theorems rely on strong assumptions which may not hold for particular spectral manifold learning algorithms. To help assess the practical validity of the Nyström formula for predicting the embedding of out-of-sample points, we have performed a series of comparative experiments.

The experiments performed here have shown empirically on several data sets that (i) those spectral embedding algorithms capture different kinds of non-linearity in the data, (ii) they can be useful for both data visualization and dimensionality reduction, and (iii) the predicted out-of-sample embedding is generally not far from the one that would be obtained by including the test point in the training set, the difference being of the same order as the effect of small perturbations of the training set.

An interesting parallel can be drawn between the spectral embedding algorithms and the view of PCA as finding the principal eigenvectors of a matrix obtained from the data. The present chapter parallels for spectral embedding the view of PCA as an estimator of the principal directions of the covariance matrix of the underlying unknown distribution, thus introducing a convenient notion of generalization, relating to an unknown distribution.
Finally, a better understanding of these methods opens the door to new and potentially much more powerful unsupervised learning algorithms. Several directions remain to be explored:

1. Using a smoother distribution than the empirical distribution to define the linear operator L_m. Intuitively, a distribution that is closer to the true underlying distribution would have a greater chance of yielding better generalization, in the sense of better estimating eigenfunctions of L. This relates to putting priors on certain parameters of the density, e.g. as in (Rosales and Frey, 2003).

2. All of these methods are capturing salient features of the unknown underlying density. Can one use the representation learned through the estimated eigenfunctions in order to construct a good density estimator? Looking at figure 1 suggests that modeling the density in the transformed space (right hand side) should be much easier (e.g. would require fewer Gaussians in a Gaussian mixture) than in the original space.

3. These transformations discover abstract structures such as clusters and manifolds. It might be possible to learn even more abstract (and less local) structures, starting from these representations. Ultimately, the goal would be to learn higher-level abstractions on top of lower-level abstractions by iterating the unsupervised learning process in multiple "layers".

Looking for extensions such as these is important because all of the manifold learning algorithms studied here suffer from the following fundamental weakness: they are using mostly the neighbors around each example to capture the local structure of the manifold, i.e. the manifold is seen as a combination of linear patches around each training example. This is very clear in LLE and Isomap, which have a simple geometric interpretation, and it is also clear in non-spectral methods such as Manifold Parzen Windows (Vincent and Bengio, 2003) and other mixtures of factor analyzers (Ghahramani and Hinton, 1996). In low dimension or when the manifold is smooth enough, there may be enough examples locally to characterize the plane tangent to the manifold. However, when the manifold has high curvature with respect to the amount of training data (which can easily be the case, especially with high-dimensional data), it is hopeless to try to capture the local tangent directions based only on local information. Clearly, this is the topic for future work, addressing a fundamental question about generalization in high-dimensional data, and for which the traditional non-parametric approaches may be insufficient.

Acknowledgments

The authors would like to thank Léon Bottou, Christian Léger, Sam Roweis, Yann Le Cun, and Yves Grandvalet for helpful discussions, and the following funding organizations: NSERC, MITACS, IRIS, and the Canada Research Chairs.

References

Baker, C. (1977). *The numerical treatment of integral equations*. Clarendon Press, Oxford.

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

Belkin, M. and Niyogi, P. (2003b). Using manifold structure for partially labeled classification. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N., and Ouimet, M. (2004). Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press.

Bengio, Y., Vincent, P., Paiement, J., Delalleau, O., Ouimet, M., and Le Roux, N. (2003). Spectral clustering and kernel PCA are learning eigenfunctions. Technical Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal.

Brand, M. (2003). Charting a manifold. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Chung, F. (1997). Spectral graph theory. In *CBMS Regional Conference Series*, volume 92. American Mathematical Society.

Cox, T. and Cox, M. (1994). *Multidimensional Scaling*. Chapman & Hall, London.

de Silva, V. and Tenenbaum, J. (2003). Global versus local methods in nonlinear dimensionality reduction. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 705–712, Cambridge, MA. MIT Press.

Donoho, D. and Grimes, C. (2003). Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. Technical Report 2003-08, Dept. Statistics, Stanford University.

Ghahramani, Z. and Hinton, G. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Dpt. of Comp. Sci., Univ. of Toronto.

Gower, J. (1968). Adding a point to vector diagrams in multivariate analysis. *Biometrika*, 55(3):582–585.

Ham, J., Lee, D., Mika, S., and Schölkopf, B. (2003). A kernel view of the dimensionality reduction of manifolds. Technical Report TR-110, Max Planck Institute for Biological Cybernetics, Germany.

Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84:502–516.

Kégl, B. (2003). Intrinsic dimension estimation using packing numbers. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 681–688. MIT Press, Cambridge, MA.

Kegl, B. and Krzyzak, A. (2002). Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):59–74.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Koltchinskii, V. (1998). Asymptotics of spectral projections of some random matrices approximating integral operators. In Eberlein, Hahn, and Talagrand, editors, *Progress in Probability*, volume 43, pages 191–227, Basel. Birkhauser.

Koltchinskii, V. and Giné, E. (2000). Random matrix approximation of spectra of integral operators. *Bernoulli*, 6(1):113–167.

Kreyszig, E. (1990). *Introductory Functional Analysis with Applications*. John Wiley & Sons, Inc., New York, NY.

Kumar, V., Grama, A., Gupta, A., and Karypis, G. (1994). *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin Cummings, Redwood City, CA.

Laub, J. and Müller, K.-R. (2003). Feature discovery: unraveling hidden structure in non-metric pairwise data. Technical report, Fraunhofer FIRST.IDA, Germany.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Rosales, R. and Frey, B. (2003). Learning generative models of affinity matrices. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 485–492, San Francisco, CA. Morgan Kaufmann Publishers.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge.

Saul, L. and Roweis, S. (2002). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155.

Saund, E. (1989). Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314.

Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1999). *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA.

Schölkopf, B., Smola, A., and Müller, K.-R. (1996). Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

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

Shawe-Taylor, J., Cristianini, N., and Kandola, J. (2002). On the concentration of spectral properties. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*. MIT Press.

Shawe-Taylor, J. and Williams, C. (2003). The stability of kernel principal components analysis and its relation to the process eigenspectrum. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731–737.

Spielman, D. and Teng, S. (1996). Spectral partitioning works: planar graphs and finite element meshes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*.

Strang, G. (1980). *Linear Algebra and Its Applications*. Academic Press, New York.

Teh, Y. W. and Roweis, S. (2003). Automatic alignment of local representations. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Torgerson, W. (1952). Multidimensional scaling, 1: Theory and method. *Psychometrika*, 17:401–419.

van der Vaart, A. and Wellner, J. (1996). *Weak Convergence and Empirical Processes with applications to Statistics*. Springer, New York.

Verbeek, J. J., Roweis, S. T., and Vlassis, N. (2004). Non-linear cca and pca by alignment of local models. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA. MIT Press.

Vincent, P. and Bengio, Y. (2003). Manifold parzen windows. In S. Becker, S. T. and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 825–832, Cambridge, MA. MIT Press.

Vlachos, M., Domeniconi, C., Gunopulos, D., Kollios, G., and Koudas, N. (2002). Non-linear dimensionality reduction techniques for classification and visualization. In *Proc. of 8th SIGKDD*, Edmonton, Canada.

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982.

Williams, C. (2001). On a connection between kernel pca and metric multidimensional scaling. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 675–681. MIT Press.

Williams, C. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA. MIT Press.

Zwald, L., Bousquet, O., and Blanchard, G. (2004). Statistical properties of kernel principal component analysis. In Shawe-Taylor, J. and Singer, Y., editors, *Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004. Proceedings*, volume 3120 of *Lecture Notes in Computer Science*, pages 594–608. Springer-Verlag.

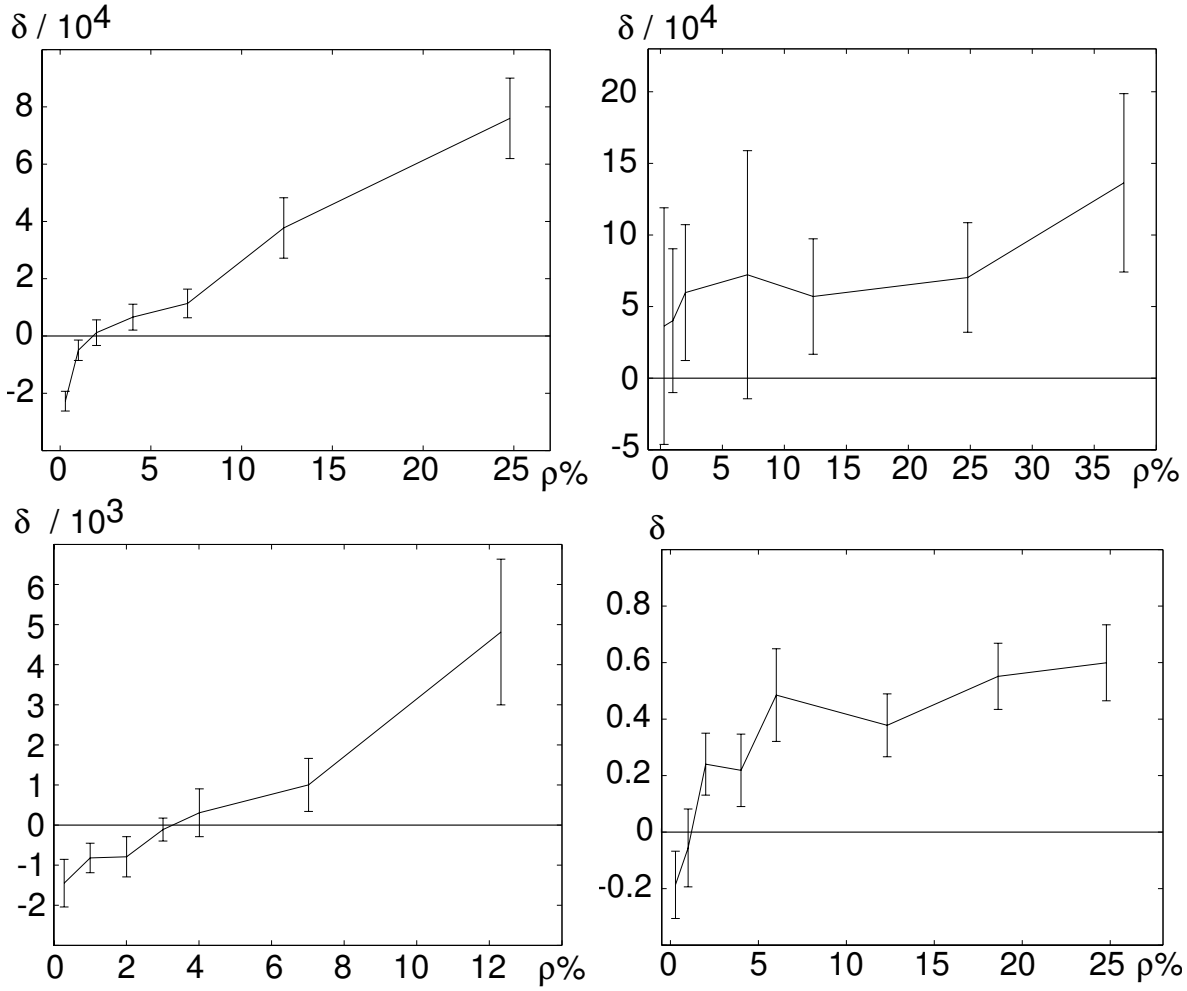


Figure 4: *δ (training set variability minus out-of-sample error), w.r.t. ρ (proportion of substituted training samples) on the "Faces" dataset ($m = 698$), obtained with a two-dimensional embedding. Top left: MDS. Top right: spectral clustering or Laplacian eigenmaps. Bottom left: Isomap. Bottom right: LLE. Error bars are 95% confidence intervals. Exchanging about 2% of the training examples has an effect comparable to using the Nyström formula.*