Learning Eigenfunctions of Similarity: Linking Spectral Clustering and Kernel PCA

Yoshua Bengio, Pascal Vincent and Jean-François Paiement
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
{*bengioy,vincentp,paiemeje*}*@iro.umontreal.ca*
http://www.iro.umontreal.ca/∼bengioy

Technical Report 1232,
Département d'Informatique et Recherche Opérationnelle

February 28th, 2003

Abstract

In this paper, we show a direct equivalence between spectral clustering and kernel PCA, and how both are special cases of a more general learning problem, that of learning the principal eigenfunctions of a kernel, when the functions are from a Hilbert space whose inner product is defined with respect to a density model. This suggests a new approach to unsupervised learning in which abstractions (such as manifolds and clusters) that represent the main features of the data density are extracted. Abstractions discovered at one level can be used to build higher-level abstractions. This paper also discusses how these abstractions can be used to recover a quantitative model of the input density, which is at least useful for evaluative and comparative purposes.

1 Introduction

Spectral clustering can give very impressive results and has attracted much interest in the last few years. 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) , a space in which the structure of the data is revealed, and then applying a classical clustering algorithm such as K-means, e.g. as in (Ng, Jordan and Weiss, 2002). What appears almost magical is the way in which sets of points that are on different highly non-linear manifolds can get mapped (in the above first step) to almost linear subspaces (different for each of these manifolds), as shown below in Figure 1. The long-term goal of the research described in this paper is to better understand such mappings and take advantage of this understanding to open the door for new unsupervised learning procedures.

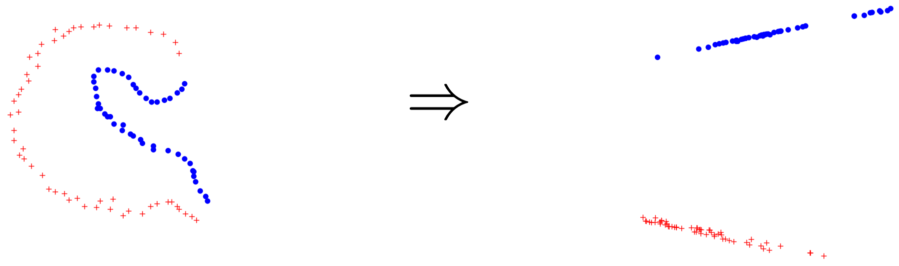

Figure 1: Example of the transformation learned as part of spectral clustering. Input data on the left, transformed data on the right, the colors and cross/circle are only used to show which points get mapped where: the mapping reveals both the clusters and the internal structure of the two manifolds.

One problem with spectral clustering is that the procedure is highly sensitive to the choice of the kernel, for example it is very sensitive to the choice of the spread (variance) of a Gaussian kernel. Another is that the procedure provides an embedding for the training points, not for new points. A very similar method for dimensionality reduction has been proposed in (Belkin and Niyogi, 2002a), based on so-called Laplacian eigenmaps. The above paper proposes to use such transformations in a semi-supervised and transductive setting: the unlabeled test set and the input part of the training set are used to learn a mapping to a more revealing representation, and the transformed training set is used with a supervised learning algorithm.
Kernel PCA is another unsupervised learning method that was proposed earlier and that is based on the simple idea of performing Principal Components Analysis in the feature space of a kernel (Schölkopf and Müller, 1996). We will explain this approach in much more detail in the paper.

In this paper, we show a direct equivalence between the embedding computed in spectral clustering and the mapping computed with kernel PCA, and how both are special cases of a more general learning problem, that of learning the principal eigenfunctions of a kernel, when the functions are from a Hilbert space whose inner product is defined with respect to a density model.

The motivations of this research are the following.

1. Understand better the strengths, weaknesses, and relations between spectral methods that can be used to transform data in a space where its

structure may be more readily apparent.

2. Generalize the transformation found in spectral clustering and Laplacian eigenmaps (which are both based on embeddings of the data examples) so as to obtain mappings that can be applied to new test points and to smoothed variant of the empirical data distribution.

3. Introduce a new multi-layered approach to learning abstractions of the structure in the data density, on top of previously learned abstractions: at each layer we indirectly model the density of the data by transforming it in a space where the notion of similarity better represents the structure of the underlying density of the data.

Note that very interesting links have already been established between kernel PCA and learning eigenfunctions in (Williams and Seeger, 2000). In particular, the eigenvalues and eigenvectors obtained from the eigen-decomposition of the Gram matrix converge to the eigenfunctions of the linear operator defined by the kernel with respect to the data density, as in equation 12 below, as the number of data points increases.

2 Spectral Manifold Learning Methods

2.1 Kernels and Notation

In this paper we will consider symmetric semi-positive definite two-argument functions called *kernels*. We also assume that the linear operator in $\mathcal{L}^2$ corresponding to the kernel is a compact operator, i.e. it maps functions in $\mathcal{L}^2$ to a closed and totally bounded set. Often a kernel K is assumed to be semi-positive definite, and in that case it can be written as a dot product in a "feature space" $\phi(x)$ (see Mercer theorem and a review of learning algorithms based on kernels, e.g. (Schölkopf, Burges and Smola, 1999; Wahba, 1990)):

$$K(x, y) = \sum_i \phi_i(x)\phi_i(y) = \phi(x) \cdot \phi(y). \tag{1}$$

where both x and y are in $\mathbf{R}^d$, while $\phi(x) \in \mathbf{R}^r$, or to allow r not necessarily finite, we write $\phi(x) \in l_2$, the space of bounded sum of squares sequences.
We are given data set $\{x_1, \ldots, x_n\}$ with examples $x_i \in \mathbf{R}^d$. We will associate a density $p(x)$ to the data generating process, either the empirical density or one obtained through a model. We will write $E[.]$ for expectations over that density, or to make it clear over which variable the integral is performed, we will write

$$E_x[f(x)] = \int f(x)p(x)dx.$$

For example, in the next two sections (on spectral clustering and on kernel PCA), we will restrict our attention to the empirical distribution associated with our data set, so we would have

$$E_x[f(x)] = \frac{1}{n} \sum_i f(x_i).$$

2.2 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. Here we follow the treatment of (Ng, Jordan and Weiss, 2002) (see the very impressive figures in the same paper). The most commonly used kernel is the Gaussian kernel:

$$K(x, y) = e^{-||x-y||/\sigma^2}. \tag{2}$$

Note that the choice of σ can strongly influence the results (so this hyper-parameter has to be selected carefully).
Spectral clustering works by first embedding the data points in a space where clusters are more clearly revealed. An example of such embedding is shown in Figure 1. The embedding is obtained as follows. First form the symmetric semi-positive definite Gram matrix M with

$$M_{i,j} = K(x_i, x_j) \tag{3}$$

and then using the row sums

$$D_i = \sum_j M_{i,j}$$

normalize it as such:

$$\tilde{M}_{i,j} = \frac{M_{i,j}}{\sqrt{D_i D_j}}.$$

Note for comparison later in the paper that, equivalently, this normalization of the Gram matrix corresponds (up to a constant) to defining a normalized kernel $\tilde{K}$ as follows:

$$\tilde{K}(x, y) = \frac{K(x, y)}{\sqrt{E_x[K(x, y)] E_y[K(x, y)]}} \tag{4}$$

(where the expectations are over the empirical distribution). Finally compute the m principal eigenvectors of $\tilde{M}$, satisfying

$$\tilde{M}\alpha_k = \lambda_k \alpha_k.$$

Let A be the $m \times n$ matrix of these eigenvectors. The coordinates of the examples within the eigenvectors represent an embedding that has very interesting properties (see Figure 1). The embedding associates the point x_i in $\mathbf{R}^d$ to the i-th column of A, A_i:

$$A_i = (\alpha_{1i}, \alpha_{2i}, \ldots, \alpha_{mi}) \tag{5}$$

Clustering is obtained from these coordinates. In the illustration of Figure 1, the two clusters correspond to groups of points that have an approximately **constant angle**, i.e. they are near one of two lines that start at the origin. Thus, in (Ng, Jordan and Weiss, 2002) it is proposed to first project these coordinates onto the unit sphere before performing K-means clustering. Projection onto the unit sphere maps A_i into $A_i/||A_i||$.
See (Ng, Jordan and Weiss, 2002; Weiss, 1999) for further justification of this procedure and its relation to the graph Laplacian and the min-cut problem.
Does the divisive normalization of the kernel in equation 4 yield a positive definite operator? Let us write $e(x) = E_y[K(x, y)]$, with $e(x)$ strictly positive if $K(x, y)$ is strictly positive-valued. Then equation 4 is rewritten using eq. 1:

$$\tilde{K}(x, y) = \sum_i \frac{\phi_i(x)}{\sqrt{e(x)}} \frac{\phi_i(y)}{\sqrt{e(y)}}$$

so defining $\hat{\phi}(x) \overset{\text{def}}{=} \phi(x)/\sqrt{e(x)} > 0$ we obtain that $\tilde{K}$ is a positive-definite kernel:

$$\tilde{K}(x, y) = \hat{\phi}(x).\hat{\phi}(y).$$

2.3 Kernel PCA

Kernel PCA is another unsupervised learning technique that maps data points to a new space. It generalizes the Principal Components Analysis approach to non-linear transformations using the kernel trick (Schölkopf and Müller, 1996; Schölkopf, Smola and Müller, 1998; Schölkopf, Burges and Smola, 1999). The algorithm implicitly finds the leading eigenvectors and eigenvalues of the (normally centered) second moment (i.e. the covariance) of the data x when it is projected in the "feature space" $\phi(x)$ (see eq. 1):

$$C = E_x[\phi(x)\phi(x)'] \tag{6}$$

Let us define the eigen-vectors of the covariance matrix:

$$Cv_k = \lambda_k v_k.$$

Using the notation of the previous section, the kernel PCA algorithm has the following steps.

- **Training:**

  1. Centering: the kernel K is first "normalized" into $\tilde{K}$ such that the corresponding feature space points $\tilde{\phi}(x_i)$ have zero expected value (under the data empirical distribution):

  $$\tilde{K}(x, y) = K(x, y) - E_x[K(x, y)] - E_y[K(x, y)] + E_x[E_y[K(x, y)]] \tag{7}$$

  See derivation of this expression in the next subsection. A corresponding normalized Gram matrix $\tilde{M}$ is formed.

2. Eigen-decomposition: find the principal eigenvectors α_k and eigenvalues a_k of the Gram matrix $\tilde{M}$ (eq. 3 but using $\tilde{K}$), i.e. solving

$$\tilde{M}\alpha_k = a_k \alpha_k.$$

with eigenvalues a_k and eigenvectors α_k.

- **Test points projection:** to project a test point x on the k-th eigenvector of the covariance matrix, compute

$$\pi_k(x) = v_k \cdot \tilde{\phi}(x) = \sum_i \alpha_{ki} \tilde{K}(x_i, x). \tag{8}$$

2.4 Normalization of the Kernel by Centering

It can be shown that the above normalization of the kernel indeed yields to centering of $\tilde{\phi}(x)$, as follows. We want to define

$$\tilde{\phi}(x) \overset{\text{def}}{=} \phi(x) - E_x[\phi(x)].$$

Thus the corresponding kernel

$$\tilde{K}(x,y) = \tilde{\phi}(x) \cdot \tilde{\phi}(y)$$

is expanded as follows

$$\begin{aligned} \tilde{K}(x,y) &= (\phi(x) - E_x[\phi(x)]) \cdot (\phi(y) - E_y[\phi(y)]) \\ &= K(x,y) - E_x[K(x,y)] - E_y[K(x,y)] + E_x[E_y[K(x,y)]] \end{aligned} \tag{9}$$

2.5 Other Spectral Dimensionality Reduction Methods

Several other dimensionality reduction and manifold discovery methods rely on the solution of an eigen-decomposition problem. For example, Local Linear Embedding (Roweis and Saul, 2000) and Isomap (Tenenbaum, de Silva and Langford, 2000) try to discover a non-linear manifold, while Multidimensional Scaling (Cox and Cox, 1994) looks for a linear manifold (but starting from a matrix of similarities between pairs of points, whereas Principal Components Analysis starts from a set of points and the definition of a dot product). An interesting link between multidimensional scaling and kernel PCA is discussed in (Williams, 2001).

A non-linear manifold discovery method very close to the mapping procedure used in spectral clustering is that of Laplacian eigenmaps (Belkin and Niyogi, 2002a; Belkin and Niyogi, 2002b; He and Niyogi, 2002; Belkin and Niyogi, 2002c), which have been proposed to perform semi-supervised learning: the mapping is obtained through the eigen-decomposition of an affinity matrix, on the input part of both labeled and unlabeled data. The mapped inputs from the labeled data set can then be used to perform supervised learning from a representation that is hoped to be more meaningful, since only the coordinates of

variation that are relevant to the input distribution would be represented in the transformed data. Note the review paper on semi-supervised learning (Seeger, 2001), for other related references. A closely related work, also aiming at semi-supervised learning, is that of (Chapelle, Schölkopf and Weston, 2003), in which a full but shared covariance matrix Parzen windows density estimator is computed in the space of kernel PCA projections.
Note also that it has already been proposed to use kernel PCA as a preprocessing step before doing clustering, in (Christianini, Shawe-Taylor and Kandola, 2002), which also illustrates the closeness of kernel PCA and spectral clustering.

3 Similarity Kernel Eigenfunctions

In this section we introduce the notion of **eigenfunctions of a kernel**, which we will find later to generalize both spectral clustering and kernel PCA. The link between kernel PCA and the eigenfunctions of the kernel has already been introduced in (Williams and Seeger, 2000), where the convergence of the Gram matrix eigenvalues to the kernel operator eigenvalues is shown. In section 3.2, we discuss how one might learn them in the case when the reference density $p(x)$ below is not the empirical density.

3.1 Hilbert Space and Kernel Decomposition

Consider a Hilbert space $\mathcal{H}$, a set of real-valued functions in $\mathbf{R}^d$ accompanied by an inner product defined with a density $p(x)$:

$$\langle f, g \rangle \stackrel{\text{def}}{=} \int f(x)g(x)p(x)dx. \tag{10}$$

This also defines a norm over functions:

$$||f||^2 \stackrel{\text{def}}{=} \langle f, f \rangle.$$

As discussed in (Williams and Seeger, 2000), the eigenfunctions of the linear operator corresponding to a given semi-positive kernel function $K(x, y)$ are thus defined by the solutions of

$$K f_k = \lambda_k f_k \tag{11}$$

where $f \in \mathcal{H}$, $\lambda_k \in \mathbf{R}$, and Kf is the application of the linear operator K to the function f, i.e.

$$(Kf)(x) \stackrel{\text{def}}{=} \int K(x, y)f(y)p(y)dy. \tag{12}$$

When seen as a linear operator, the kernel K can thus be expanded in terms of a basis formed by its eigenfunctions (Mercer):

$$K = \sum_k \lambda_k f_k f_k'$$

where by convention $|\lambda_1| \geq |\lambda_2| \geq \ldots$ This can also be written as follows:

$$K(x,y) = \sum_{k=1}^{\infty} \lambda_k f_k(x) f_k(y).$$

and because we choose the eigenfunctions to form an orthonormal basis, we have

$$\langle f_k, f_l \rangle = \delta_{k,l}.$$

Section 3.2 shows a criterion which can be minimized in order to learn the principal eigenfunctions.
Note that, as shown in (Williams and Seeger, 2000), when $p(x)$ is the true data generating density and when we want to learn an unknown function f using an approximation g that is a finite linear combination of basis functions, if the unknown function f is assumed to come from a zero-mean Gaussian process prior with covariance $E_f[f(x)f(y)] = K(x,y)$, then the best basis functions, in terms of expected squared error, are the leading eigenfunctions of the linear operator K as defined above.

3.2 Learning the Leading EigenFunctions

Using the Fourier decomposition property, the best approximation of $K(x,y)$ w.r.t. $\mathcal{H}$'s norm using only m terms is the expansion that uses the *first* m terms (with largest eigenvalues):

$$\sum_{k=1}^{m} \lambda_k f_k(x) f_k(y) \approx K(x,y),$$

in the sense that it minimizes the $\mathcal{H}$-norm of the approximation error. In particular, let us consider the principal eigenfunction. It is the norm 1 function f which minimizes

$$J_K(f,\lambda) = \int (K(x,y) - \lambda f(x) f(y))^2 p(x) p(y) dx dy$$

i.e.

$$(f_1, \lambda_1) = \mathrm{argmin_{f,\lambda} J_K(f,\lambda)} \tag{13}$$

under the constraint $||f|| = 1$. This is not very original and can be proven easily, as it is only a generalization to functional spaces of the auto-encoder view of principal component analysis.

Proposition 1 *The principal eigenfunction of the linear operator corresponding to kernel K is the norm-1 function f that minimizes the reconstruction error*

$$J(f,\lambda) = \int (K(x,y) - \lambda f(x) f(y))^2 p(x) p(y) dx dy.$$

Proof

Take the derivative of J w.r.t. the value of f at a particular point z (under some regularity conditions to bring the derivative inside the integral):

$$\frac{\partial J}{\partial f(z)} = 2\int (K(z,y) - \lambda f(z)f(y))\lambda f(y)p(y)dy$$

and set it equal to zero:

$$\int K(z,y)f(y)p(y)dy = \lambda f(z)\int f(y)^2 p(y)dy.$$

Using the constraint $||f||^2 = \langle f,f\rangle = \int f(y)^2p(y)dy = 1$, we obtain the eigenfunction equation:

$$Kf = \lambda f. \tag{14}$$

Note that in practice it maybe inconvenient to minimize J under the constraint that $||f|| = 1$. A practical solution is to minimize with respect to an unconstrained function g,

$$J(g) = \int (K(x,y) - g(x)g(y))^2 p(x)p(y)dxdy.$$

where we can always later decompose g into its norm and a normalized function:

$$g(x) \overset{\text{def}}{=} \sqrt{\lambda} f(x)$$

with $||f|| = 1$ and $\lambda = ||g||^2$. Minimizing J w.r.t. g yields the first-order equation

$$Kg = g||g||^2$$

which when it is solved implies a solution to the eigenfunction equation (plugging the definition of g):

$$\sqrt{\lambda} Kf = \lambda^{3/2} f \quad \Rightarrow Kf = \lambda f$$

for $\lambda > 0$ (which is what we care about here). Note that there are many solutions to this equation (all the eigenfunctions with positive eigenvalue). Before we show that the one that minimizes J is the principal eigenfunction, let us consider the minimization with respect to λ:

$$\frac{\partial J}{\partial \lambda} = 2\int (K(x,y) - \lambda f(x)f(y))f(x)f(y)p(x)p(y)dxdy = 0$$

which gives rise to

$$\lambda = \frac{\int K(x,y)f(x)f(y)p(x)p(y)dxdy}{\int f(x)^2 f(y)^2 p(x)p(y)dxdy}$$

where the denominator is $\int f(x)^2p(x)dx \int f(y)^2p(y)dy = ||f||^2||f||^2 = 1$ and the numerator is $\int f(x)(\int K(x,y)f(y)p(y)dy)p(x)dx$ so the solution is

$$\lambda = \langle f, Kf\rangle. \tag{15}$$

Let us rewrite $J(f,\lambda)$ by expanding the square, using inner product notation and the notations

$$K_x(y) = K(x,y)$$

and

$$E[f] = \int f(x)p(x)dx,$$

so

$$\begin{aligned} J(f,\lambda) &= \int (\int K_x(y)^2 p(y)dy)p(x)dx - 2\lambda \int f(x)(\int K(x,y)f(y)p(y)dy)p(x)dx + \lambda^2 \|f\|^4 \\ &= E[\|K_x\|^2] - 2\lambda\langle f, Kf\rangle + \lambda^2 \\ &= E[\|K_x\|^2] - \lambda^2 \end{aligned} \tag{16}$$

using eq. 15 in the last step. Plugging the above solutions, the k-th eigenfunction would yield an error

$$J(f_k,\lambda_k) = E[\|K_x\|^2] - \lambda_k^2$$

which is minimized for $k = 1$ if $\lambda_1 > \lambda_2$ or more generally by any eigenfunction f_k such that $|\lambda_k| = |\lambda_1|$. Therefore, the principal eigenfunction minimizes J. Q.E.D.

Note that the proof also gives us a criterion in which the norm 1 constraint is eliminated:

$$J_K(g) = \int (K(x,y) - g(x)g(y))^2 p(x)p(y)dxdy \tag{17}$$

which gives a solution g from which we can recover λ and f through $\lambda = \|g\|^2$ and $f = g/\sqrt{\lambda}$.
Note that the function g that we obtain is actually a component of a "feature space" ϕ for K. Indeed, if

$$K(x,y) = \sum_i \lambda_i f_i(x) f_i(y)$$

then writing $\phi_i(x) = \sqrt{\lambda_i} f_i(x)$ gives rise to a dot product decomposition of K,

$$K(x,y) = \phi(x)\cdot\phi(y).$$

Let us now consider learning not only the first but also the leading m eigenfunctions. To simplify notation, let us define the "residual kernel"

$$K_k(x,y) = K(x,y) - \sum_{i=1}^{k} \lambda_k f_k(x) f_k(y), \tag{18}$$

with $K_0 = K$. The corresponding **kernel reconstruction error** is

$$K_k(x,y)^2 = (K(x,y) - \sum_{i=1}^{k} \lambda_k f_k(x) f_k(y))^2. \tag{19}$$

Justified by Proposition 2 below, a general algorithm for learning the first m eigenfunctions (and corresponding eigenvalues) of a linear operator K can thus be written as follows:

- For $k = 1$ to m

$$(f_k, \lambda_k) = \underset{||f||=1}{\text{argmin}_{\text{f},\lambda}} \; J_{K_{k-1}}(f, \lambda)$$

In practice the minimization would have to be performed on a large class of functions or non-parametrically, i.e. we impose some restrictions on the class of functions. A special case of interest is that in which the density $p(x)$ is the empirical density. In that case the minimization of J can be done with numerical analysis methods for finding the eigenvectors of a matrix. However, it might be interesting to consider smooth classes of functions (which can only approximate the above minimization).
Following the reasoning exposed for online learning of the principal components (Diamantras and Kung, 1996), a simpler implementation would not have to wait for the first $m-1$ eigenfunctions before beginning to learn the m-th one. They can all be learned in parallel, using the algorithm to learn the m-th one that assumes that the first $m-1$ are learned: convergence will simply be faster for the leading eigenfunctions. Note that convergence also depends on the ratios of eigenvalues, as usual for PCA and iterative eigen-decomposition algorithms (Diamantras and Kung, 1996).
Like the previous proposition, the following proposition is not very original and generalizes well known results on eigenvectors.

Proposition 2 *Given the first principal $m-1$ eigenfunctions f_i of the linear operator associated with a symmetric function $K(x, y)$, the m-th one can be obtained by minimizing w.r.t. g the expected value of the* kernel reconstruction error*:*

$$\int (K(x,y) - g(x)g(y) - \sum_{i=1}^{k-1} \lambda_i f_i(x) f_i(y))^2 p(x)p(y)dxdy \tag{20}$$

Then we get the m-th eigenvalue $\lambda_m = ||g||^2$ and the m-th eigenfunction $f_m = g/\sqrt{\lambda_m}$.

Proof

Let us consider the reconstruction error using the approximation $g(x)g(y) + \sum_{i=1}^{m-1} \lambda_i f_i(x) f_i(y)$:

$$J_m = \int (K(x,y) - g(x)g(y) - \sum_{i=1}^{m-1} \lambda_i f_i(x) f_i(y))^2 p(x)p(y)dxdy.$$

where $g(x)$ can be decomposed into $g(x) \overset{\text{def}}{=} \lambda' f'(x)$ with $||f'|| = 1$, and (f_i, λ_i) are the first $m-1$ (eigenfunction,eigenvalue) pairs in order of decreasing absolute value of λ_i. We want to prove that g that minimizes J_m is f_m.
The minimization of J_m with respect to λ' gives

$$\frac{\partial J_m}{\partial \lambda'} = 2 \int (K(x,y) - \lambda' f'(x) f'(y) - \sum_{i=1}^{m-1} \lambda_i f_i(x) f_i(y)) f'(x) f'(y) p(x)p(y)dxdy = 0$$

which gives rise to

$$\lambda' = \langle f', Kf' \rangle - \sum_{i=1}^{m-1} \int \lambda_i f_i(x) f_i(y) f'(x) f'(y) p(x) p(y) dx dy \tag{21}$$

We have

$$\begin{aligned} J_m = \quad & J_{m-1} \\ & -2 \int \lambda' f'(x) f'(y) (K(x,y) - \textstyle\sum_{i=1}^{m-1} \lambda_i f_i(x) f_i(y)) p(x) p(y) dx dy \\ & + \int (\lambda' f'(x) f'(y))^2 p(x) p(y) dx dy \end{aligned}$$

which gives

$$J_m = J_{m-1} - \lambda'^2$$

using eq. 21.
The value of λ'^2 should be maximized for the last expression to be minimized (that is what will give rise to the ordering of the eigenfunctions in terms of absolute value).
Take the derivative of J_m w.r.t. the value of f' at a particular point z (under some regularity conditions to bring the derivative inside the integral):

$$\frac{\partial J_m}{\partial f'(z)} = 2 \int (K(z,y) - \lambda' f'(z) f'(y) \sum_{i=1}^{m-1} \lambda_i f_i(z) f_i(y)) \lambda' f'(y) p(y) dy$$

and set it equal to zero:

$$\int K(z,y) f'(y) p(y) dy = \sum_{i=1}^{m-1} \int \lambda_i f_i(z) f_i(y) f'(y) p(y) dy.$$

Using the constraint $||f'||^2 = \langle f', f' \rangle = \int f'(y)^2 p(y) dy = 1$, we obtain:

$$Kf' = \lambda' f' + \sum_{i=1}^{m-1} \int \lambda_i f_i(z) f_i(y) f'(y) p(y) dy. \tag{22}$$

Using the assumption that f_i are orthogonal eigenfunctions for $i < m$, we can thus rewrite eq. 22 as

$$Kf' = \lambda' f' + \sum_{i=1}^{m-1} \lambda_i f_i \langle f', f_i \rangle.$$

Since we can always write the application of K in terms of the eigenfunctions as follows,

$$Kf' = \sum_{i=1}^{\infty} \lambda_i f_i \langle f', f_i \rangle,$$

we obtain

$$\lambda' f' = \lambda_m f_m \langle f', f_m \rangle + \sum_{i=m+1}^{\infty} \lambda_i f_i \langle f', f_i \rangle.$$

Applying Perceval's theorem to obtain the norm on both sides, we get

$$\lambda'^2 = {\lambda_m}^2 \langle f', f_m \rangle^2 + \sum_{i=m+1}^{\infty} {\lambda_i}^2 \langle f', f_i \rangle^2.$$

If the eigenvalues are distinct, we have $\lambda_m > \lambda_i$ for $i > m$, and the last expression is maximized when $\langle f', f_m \rangle = 1$ and $\langle f', f_i \rangle = 0$ for $i > m$, which proves that $f_m = f'_m$ is in fact the m-th eigenfunction of the kernel K and thereby $\lambda_m = \lambda'_m$.
If the eigenvalues are not distinct, then the result can be generalized in the sense that the choice of eigenfunctions is not anymore unique but the eigenfunctions sharing the same eigenvalue form an orthogonal basis for a subspace.
Then since we have assumed $g = \lambda' f'$, after obtaining g through the minimization of J_m, since this minimization yields $\lambda' = \lambda_m$ and $f' = f_m$, and since $||f_m|| = 1$ by definition, we get $\lambda_m = ||g||^2$ and $f_m = g/\sqrt{\lambda_m}$.
Q.E.D.

Discussion
The above proposition is not very surprising since it is analog to the well-known results on eigenvectors. However, it may help us understand a bit better what it means to learn the eigenfunctions of a kernel (and thus as we show below, doing kernel PCA or spectral clustering). Indeed, it highlights the fact that the embedding obtained by these methods is one that **attempts to preserve dot products in the mean-squared error sense**, just like classical multidimensional scaling, but in a high-dimensional "feature space". What are the **dominant dot-products** that this approximation is trying to preserve? Those corresponding to pairs of points which are **colinear in feature space**. This maybe helps to explain why these embedding tend to result in groups of points which are colinear (or locally colinear), as for example in Figure 1. This is interesting but maybe raises the question of whether preserving the largest dot products is always the most appropriate objective. Note that if one uses a Monte-Carlo method to minimize the expectation of the kernel reconstruction error (eq. 19, and see section 5) then it is not difficult to generalize the learning algorithm to other error criteria.

4 Links between the Methods

In this section we show that that finding the eigenfunctions of the kernel function includes as a special case both the embedding found in spectral clustering and that found by Kernel PCA.

Proposition 3 *If we choose for $p(x)$ (the weighing function in the Hilbert space inner product of eq. 10) the empirical distribution of the data, then the embedding A_{ik} obtained with spectral clustering (see eq. 5) is equivalent to values of the eigenfunctions: $A_{ik} = f_k(x_i)$ where f_k is the k-th principal eigenfunction of* *the kernel.*

Proof

As shown in Proposition 1, finding function f and scalar minimizing

$$\int (\tilde{K}(x,y) - \lambda f(x)f(y))^2 p(x)p(y)dxdy$$

such that $||f|| = 1$ yields a solution that satisfies the eigenfunction equation

$$\int \tilde{K}(x,y)f(y)p(y)dy = \lambda f(x)$$

with λ the (possibly repeated) maximum norm eigenvalue, i.e. we obtain $f = f_1$ and $\lambda = \lambda_1$ respectively the principal eigenfunction (or one of them if the maximum eigenvalue is repeated) and its corresponding eigenvalue.
Here $\tilde{K}$ refers to a possibly normalized kernel, e.g. such as may be defined in eq. 4 for spectral clustering.
Using the empirical density and considering the values of x at the data points x_i, the above equation becomes (for all x_i):

$$\frac{1}{n}\sum_j \tilde{K}(x_i, x_j)f(x_j) = \lambda f(x_i).$$

Let us write $u_j = f(x_j)$ and $\tilde{M}_{ij} = \tilde{K}(x_i, x_j)$, then the above can be written

$$\tilde{M}u = n\lambda u.$$

The spectral clustering method thus solves the same eigenvalue problem (up to scaling the eigenvalue by n) and thus we obtain for the principal eigenvector the expected result:

$$A_{i1} = f_1(x_i).$$

To obtain the result for the other coordinates, i.e. other eigenvalues, simply consider the "residual kernel" K_k as in eq. 18 and recursively apply the same reasoning to obtain that $A_{i2} = f_2(x_i)$, $A_{i3} = f_3(x_i)$, etc...
Q.E.D.

Discussion

What do we learn from this proposition? Firstly, there is an equivalence between the principal eigenvectors of the Gram matrix and the principal eigenfunctions of a the kernel, when the Hilbert space is defined with an inner product of the form of eq. 10, and the density in the inner product is the empirical density. (Williams and Seeger, 2000) had already shown a related result when the number of data points goes to infinity. Why are these results interesting? For one, it

suggests generalizations of the transformation performed for spectral clustering in which one uses a smoother density $p(x)$, e.g. obtained through a parametric or non-parametric model.
Another nice fallout of this analysis is that it provides a simple way to **generalize the embedding to a mapping**: whereas the methods used with spectral clustering and Laplacian eigenmaps only give the transformed coordinates of training points (i.e. an embedding of the training points), it is easy to obtain the transformed coordinates of a new point, once it is realized that the transformed coordinates are simply the values of the principal eigenfunctions. Let us first consider the easiest case, where $p(x)$ is the empirical distribution. Then Proposition 4 allows us to write

$$f_k(x) = \sum_i \alpha_{ki} \tilde{K}(x_i, x)$$

where α_k is the k-th principal eigenvector of the normalized Gram matrix $\tilde{M}$, with $\tilde{M}_{ij} = \tilde{K}(x_i, x_j)$. When $p(x)$ is not the empirical distribution, Propositions 1 and 2 provide a criterion that can be minimized in order to learn the principal eigenfunctions f_k.

Proposition 4 *The test point projection $\pi_k(x)$ (eq. 8) on the k-th principal component obtained by kernel PCA with normalized kernel $\tilde{K}(x,y)$ is equal to*

$$\pi_k(x) = \lambda_k f_k(x)$$

where λ_k and $f_k(x)$ are respectively the k-th leading eigenvalue and eigenfunction of $\tilde{K}$, when the Hilbert space inner product weighing function $p(x)$ is the empirical density.

Proof

Let us start from the eigenfunction equation 11 on kernel $\tilde{K}$ and apply the linear operator $\tilde{K}$ on both sides:

$$\tilde{K}\tilde{K}f_k = \lambda_k \tilde{K} f_k.$$

which can be written

$$\int \tilde{K}(x,y) \int \tilde{K}(y,z) f_k(z) p(z) p(y) dz dy = \lambda_k \int \tilde{K}(x,y) f_k(y) p(y) dy$$

or changing the order of integrals on the left-hand side:

$$\int f_k(z) \left(\int \tilde{K}(x,y)\tilde{K}(y,z) p(y) dy \right) p(z) dz = \lambda_k \int \tilde{K}(x,y) f_k(y) p(y) dy$$

Let us now plug-in the definition of $\tilde{K}(x,y) = \sum_i \tilde{\phi}_i(x)\tilde{\phi}_i(y)$:

$$\int f_k(z) \left(\int \sum_i \tilde{\phi}_i(x)\tilde{\phi}_i(y) \sum_j \tilde{\phi}_j(y)\tilde{\phi}_j(z) p(y) dy \right) p(z) dz = \lambda_k \int \sum_i \tilde{\phi}_i(x)\tilde{\phi}_i(y) f_k(y) p(y) dy.$$

In this expression we can see the element (i, j) of the feature space covariance matrix C (eq. 6):

$$C_{ij} = \int \tilde{\phi}_i(y)\tilde{\phi}_j(y)p(y)dy$$

and we obtain (plugging this definition on the left hand side and pulling sums out of integrals)

$$\sum_i \tilde{\phi}_i(x) \sum_j C_{ij} \int \tilde{\phi}_j(z) f_k(z)p(z)dz = \lambda_k \sum_i \tilde{\phi}_i(x) \int f_k(y)\tilde{\phi}_i(y)p(y)dy$$

or

$$\tilde{\phi}(x) \cdot (C\langle f_k, \tilde{\phi}\rangle) = \tilde{\phi}(x) \cdot (\lambda_k \langle f_k, \tilde{\phi}\rangle)$$

where $\langle f_k, \tilde{\phi}\rangle$ is the feature space vector with elements $\int f_k(y)\tilde{\phi}_i(y)p(y)dy$. Since this is true for all x, it must be that in the region where $\tilde{\phi}(x)$ takes its values,

$$Cv_k = \lambda_k v_k$$

where $v_k = \langle f_k, \tilde{\phi}\rangle$ and it is also the k-th eigenvector of the covariance matrix C. Finally, the kernel PCA test projection on that eigenvector is

$$\begin{aligned} \pi_k(x) &= v_k \cdot \tilde{\phi}(x) \\ &= (\int f_k(y)\tilde{\phi}(y)p(y)dy) \cdot \tilde{\phi}(x) \\ &= \int f_k(y)\tilde{\phi}(y) \cdot \tilde{\phi}(x)p(y)dy \\ &= \int f_k(y)K(x, y)p(y)dy \\ &= \lambda_k f_k(x) \end{aligned} \tag{23}$$

Q.E.D.

Discussion

What do we learn from this second proposition? again, we find an equivalence between the eigenfunctions of the kernel (in an appropriate Hilbert space) and the mapping computed through kernel PCA. By combining this with the first proposition, we trivially obtain an equivalence between the mappings computed for spectral clustering and for kernel PCA, up to *scaling by the eigenvalues*, and up to a *different normalization of the kernel*.
In addition, we again find the possibility of generalizing, from kernel PCA to the case when the density defining the inner product of the Hilbert space is not the empirical density but a smoother density. Finally, by stating the problem in terms of eigenfunctions, we open the door to other generalizations discussed below, in which the eigenfunctions are only approximately estimated (allowing to impose further smoothness constraints or other domain-specific constraints),

and the possibility of estimating the eigenfunctions through a stochastic minimization process that does not require to explicitly compute and store the Gram matrix.
The links thus discovered leave open other questions. For example, is there a "geometric" meaning to the divisive normalization of the kernel used with spectral clustering (equation 4)? This 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 in with kernel PCA (equation 7) makes sense geometrically as a centering in feature space. Both normalization procedures make use of the kernel row/column average $E_x[K(x, y)]$.

5 Unsupervised Stochastic Learning of Eigenfunctions

Based on the previous considerations, we can outline a general stochastic algorithm for unsupervised learning through learning of a kernel's eigenfunctions. One interesting advantage of this approach is that it does not require to store the whole Gram matrix, since it uses stochastic sampling of data pairs to approximate the integral of eq. 20 (and minimize it). Future experiments should compare this approach to the use of the Nystrom method (Williams and Seeger, 2000), in which a fixed subsample is chosen, small enough to handle its Gram matrix, so as to be amenable to conventional (but numerically more powerful) eigenvector / eigenvalue computation routines.
The outline of the algorithm is thus the following:

- Choose a primary data density model (which could be the empirical distribution, but we favor smoother models, e.g. a Parzen windows density) $p(x)$.
- Choose a kernel K (e.g. Gaussian with fixed variance σ).
- Choose a normalization method for the kernel (e.g. divisive as in eq. 4 or subtractive as in eq. 7), to obtain $\tilde{K}$.
- Choose a flexible but smooth class of functions $\mathcal{F}$ over which one can easily tune a function using gradient-based methods (e.g. neural networks).
- Choose a number k of dimensions for the transformed space (number of eigenfunctions to learn).
- Apply the eigenfunction learning algorithm described in section 3.2 to obtain a (smoothed and approximate) estimation of the first k eigenfunctions of $\tilde{K}$, minimizing J over the class $\mathcal{F}$. Note that this can be done "on-line" without having to store the Gram matrix, sampling i.i.d. pairs of examples from $p(x)$.

- Optionally estimate out-of-sample likelihood using the method described in the next section (sec. 6).

When sampling pairs x and y from $p(x,y) = p(x)p(y)$, many pairs may yield very little information on the eigenfunctions, those pairs such that $K(x,y)$ is near a constant (e.g. near zero for the Gaussian kernel, when $||x-y||^2/\sigma^2$ is large). For this reason it might be very useful to approximate J_m using importance sampling rather than straightforward Monte-Carlo sampling. The minimum variance proposal distribution for importance sampling would be one that picks x and y in proportion to

$$(K(x,y) - \sum_i g_i(x)g_i(y))^2 p(x)p(y),$$

where $g_i(x)$ is our current guess at $\sqrt{\lambda_i} f_i(x)$, i.e our current guess at $\phi_i(x)$. If the approximation $\sum_i g_i(x)g_i(y)$ is "local" in the sense that it easily assigns small values to regions of (x,y) space in which x is far from y and $K(x,y)$ is tiny (because that is the first thing and easiest thing that would be learned), then a reasonable approximation to the ideal proposal distribution is one that simply gives low probability to y, given x, when $K(x,y)$ is small. For example, it would be reasonable to use in the case of the Gaussian kernel the proposal distribution

$$q(x,y) = p(x)q(y|x) = p(x)\frac{e^{-0.5||x-y||^2/\sigma^2}}{(2\pi)^{d/2}\sigma^d}$$

which samples x from $p(x)$ and y from a Gaussian centered at x with variance σ^2.
An outline of the algorithm when $p(x)$ is not the empirical distribution is given in Algorithm 1 at the end of this document.

6 Estimating the Likelihood and Mapping Back in Input Space

The above unsupervised learning algorithm hopefully allows to reveal salient characteristics of the data distribution, such as clusters and manifolds, through a data-dependent mapping $f(x)$. Implicitly, the algorithm tells us something about the density and we would like to be able to formalize that link, by representing explicitly a corresponding density model $p_f(x)$. Note that we start from a rough estimate of the density, $p(x)$, that is used to define our Hilbert space inner product, and the procedures discussed here allow to hopefully obtain a better estimate, $p_f(x)$.
The goal of this section is thus to show how one could estimate the training or out-of-sample likelihood of data using an already learned mapping $f(x)$. This could be particularly important in order to compare quantitatively the value added by the representation of the data in $f(x)$, against traditional unsupervised learning algorithms, many of which can be compared on a common yardstick through the average out-of-sample log-likelihood, i.e. here the average of

$\log p_f(x_i)$ over data x_i not used to learn the model. How do we recover a density model $p_f(x)$ from $f(x)$?
Suppose that we have already learned a map $f(x)$ from $\mathbf{R}^d$ to $\mathbf{R}^k$ with $k \leq d$ (future work should consider the case $k > d$). Then under smoothness and topological assumptions on f (to be formalized), there exist two functions g and h which together allow to *approximately reconstruct* x using $f(x)$ as follows:

$$\hat{x} = h(f(x), g(x)).$$

Intuitively, $h(x)$ captures the "complement" of $f(x)$, i.e. the variations of x not captured by f, such that the concatenation (f, g) is a bijective and invertible mapping. One can approximate such an inversion by looking for g and h which minimize a reconstruction error, e.g.

$$\min_{g,h} \int (h(f(x), g(x)) - x)^2 p(x) dx.$$

If we had such a mapping then we could use the integral change of variable identities to recover the distribution of x from a model of the distribution of

$$y = (f(x), g(x)).$$

The density transformation that gives the density $\hat{p}_f$ of $\hat{x}$ given the density p_y of y is

$$\hat{p}_f(\hat{x}) = p_y(y)|h'|$$

where $|h'|$ denotes the absolute value of the determinant of the Jacobian matrix $h'_{ij} = \frac{\partial h_i}{\partial y_j}$.
However, in practice the above minimization might not be perfect (especially if $p(x)$ is not the empirical distribution), i.e. a residual error will remain and $x \neq \hat{x}$.
In that case we can still recover $p_f(x)$ up to an arbitrarily small sampling error by estimating a convolution of $\hat{p}_f(\hat{x})$. For example, we can model the reconstruction errors

$$\epsilon = x - \hat{x}$$

with a noise model $N(\epsilon)$ (e.g. an isotropic Gaussian $N(0, \sigma^2)$ whose variance σ^2 is equal to the average of the reconstruction error $||x - \hat{x}||^2$). From this we can express the density $p_f(x)$ with $\hat{p}_f(\hat{x})$ convolved with $N(\epsilon)$:

$$p_f(x) = \int \hat{p}_f(x - \epsilon) N(\epsilon) d\epsilon. \tag{24}$$

If the noise variance is small, this is a very local integral which should be "easy" to perform numerically, e.g. by straightforward Monte-Carlo, sampling ϵ from $N(\epsilon)$.

7 Open Questions and Future Work

7.1 Multi-Layered Similarity Learning

Can we improve on the unsupervised learning algorithm of section 5 by discovering higher-level abstractions? Much previous work, especially related to artificial neural networks has been based on the idea of discovering abstractions by building on top of already discovered abstractions, in a multi-layered fashion (Hinton, 1986; Hinton, 2002).
Consider simply applying the procedure of section 5 iteratively, hopefully thus gradually building up higher level abstractions to describe the structure of the input density. Since the algorithm presented in the previous section allows to obtain a presumably "better" density estimate $p_f(x)$ from a rougher one $p(x)$ (this remains to be shown!), we might hope that repeating the same procedure would yield an even better estimate.
More precisely, at each iteration, one would use the unsupervised learning procedure to map the data $x^{(i)}$ at level i into the data $x^{(i+1)}$ at the next level:

$$x^{(i+1)} = f^{(i)}(x^{(i)}).$$

In addition, to be consistent with the above discussion, one would like to sample from $p_{f^{(i)}}$ at level i of the unsupervised learning procedure.

7.2 Relation Between Kernel and Density

What is the *right* choice of kernel?
Intuitively, the right choice of kernel is related to the right notion of similarity between data points, which is itself related to the density of the data.
For example, if we use a Gaussian kernel with a variance parameter σ that is too small with respect to the typical nearest neighbor then all the points align along the principal direction in features space (not revealing any structure at all in the data). Similarly, if the density is very different in differents parts of space, the procedure with a Gaussian kernel and a global σ does not work very well (because a value of σ that is appropriate in one region is not appropriate in the other and vice-versa).
Another example that may help drive our intuition in the right direction is the special case where the data is distributed according to a single anisotropric Gaussian with a covariance matrix Σ. In that case, an appropriate similarity function might be one similar to the Mahalanobis distance, e.g.

$$K(x, y) = e^{-0.5(x-y)'\Sigma^{-1}(x-y)}.$$

The general question we would like to answer is the following: given an arbitrary density model $p(x)$, what is the most appropriate similarity kernel (for use in one of the spectral projection techniques)?
An interesting direction to look into is the work by Amari et al. on the geometry of curved manifolds which define a local metric. They have exploited this idea

to propose improved kernels for SVMs (Amari and Wu, 1999), based on the result that the local Riemannian metric tensor induced by the feature space transformation $\phi(x)$ is given by the positive definite matrix

$$\frac{\partial}{\partial x_i}\frac{\partial}{\partial y_i}K(x,y)\mid_{x=y} = \frac{\partial\phi(x)}{\partial x_i}\cdot\frac{\partial\phi(y)}{\partial y_i}$$

where x_i and y_i here denote the i-th coordinate respectively of x and y.
Previous litterature on the link between spectral clustering and the Laplacian operator might be helpful to understand what the appropriate kernel should be. The Laplacian operator (Belkin and Niyogi, 2002a) yields the Gaussian kernel with variance t and corresponds to the solution of the heat equation after t units of time in an isotropic environment. An intuition is that the generalization we are seeking corresponds to the solution obtained with a density-dependent, anisotropic environment.
Another link already mentionned earlier between the data density and the kernel is also studied in (Williams and Seeger, 2000): it is shown that if the function to be approximated is sampled from a Gaussian process prior with covariance kernel $K(x, y)$, then bases that minimize the expected squared error are the leading eigenfunctions of the linear operator K (defined as in 12, i.e. with respect to $p(x)$). Related to this work is the proposal in (Girolami, 2001) to approximate the density directly as a linear combination of eigenfunctions of the kernel: kernel PCA is thus extended to provide the discrete expansion coefficients for a non-parametric orthogonal series density estimator.
Another related relevant recent work is that of (Chapelle, Schölkopf and Weston, 2003), in which the input data empirical distribution is used to adapt a metric to be used for learning an SVM with labeled data, i.e. it is an application to semi-supervised learning. The above paper considers a single shared covariance matrix in a Parzen windows density estimate performed in the kernel PCA space. This idea is linked to a flurry of other recent work on learning the local structure of the density through local covariance matrices, as in (Vincent and Bengio, 2003), and in the case of mixtures of factor analyzers, (Teh and Roweis, 2003) and (Brand, 2003) (which also aim at discovering an underlying low-dimensional manifold).

7.2.1 Fisher Kernel

Yann Le Cun (personal communication) has proposed to use the Fisher kernel here, since it can be derived from a density (unfortunately, it should be a parametric density). The Fisher kernel is defined so, for a parametric model density $p_\theta(x)$:

$$K(x,y) \propto \frac{\partial \log p_\theta(x)}{\partial\theta}' F^{-1}\frac{\partial \log p_\theta(y)}{\partial\theta}$$

where F is the Fisher matrix (which is a Hessian matrix and also a covariance matrix):

$$Fij = \int \frac{\partial \log p_\theta(x)}{\partial\theta_i}\frac{\partial \log p_\theta(x)}{\partial\theta_j}p_\theta(x)dx.$$

What is interesting is that if one tries this formula with the Gaussian density with free parameter μ and fixed covariance matrix Σ, one gets

$$(x-\mu)'\Sigma^{-1}(y-\mu).$$

Note that if one considers that this is a dot product $\langle x, y\rangle$ in some vector space, the corresponding distance is

$$||x-y||^2 = \langle x, x\rangle + \langle y, y\rangle - 2\langle x, y\rangle = (x-y)'\Sigma^{-1}(x-y),$$

which is the Mahalonabis distance. Since a reasonable way to define a kernel is with $\exp(-distance^2)$, one possible kernel is

$$K(x, y) = \exp(-(x-y)'\Sigma^{-1}(x-y)).$$

This is very nice, but the main problem with all this is that at this point we don't understand **why one should use the Fisher kernel**, why we should use the exponential of the distance rather than the dot product, etc... Another issue is that in general we may not have a nice parametric model p_θ but rather a non-parametric model. That might be circumvented if the non-parametric model can be expressed parametrically, e.g. as a mixture of Gaussians. What if we included the covariance matrix (or matrices in the case of a Gaussian mixture) in the free parameters?

7.3 How to estimate a likelihood when $k > d$?

The technique proposed in section 6 only works when there are less projected dimensions than original dimensions. What could be done otherwise?
It is always possible to write instead a density model as the normalization of an energy function computed only on the projected data. For example, let us assume we have a "simple" density model $p_u(u)$ for the projected data $u = f(x)$. Then our model of the original data x could be

$$p_f(x) = \frac{p_u(f(x))}{Z}$$

where Z is the partition function, the integral of $p_u(f(x))$. It is not clear whether this integral can be approximated accurately, even with a large number of samples, because $p_u(f(x))$ may have many modes (in x) and x is generally high-dimensional (contrast with the integral of eq. 24).
Another option is to generalize the procedure of eq. 24, taking into account the fact that when $k > d$, the data u does not span the whole of $\mathbf{R}^k$, only a manifold of dimension at most d. The local structure of this manifold is given by the singular value decomposition of the (rectangular, now) Jacobian matrix $\frac{\partial f_i(x)}{\partial x_j}$, since $f(x)$ can only vary in the directions locally tangent to this manifold. Note also that importance sampling might be used to estimate the conditional density of u given that u is on that manifold, by sampling x's (according to any given distribution, say $p(x)$), and generating points $u = f(x)$ that only lie on the manifold.

8 Conclusion

Spectral methods discussed in this paper empirically appear to allow capturing such salient features of a data set as its main clusters and submanifolds. This is unlike previous manifold learning methods like LLE and Isomap which assume a single manifold and have not been designed to say something about the modes of the distribution.
However, there is much that remains to be understood about these methods. For example, what is the role of normalization? how should the kernel be chosen? and more fundamentally, why are these algorithms doing what they are doing? to this question there are already partial answers, and this paper may have contributed a little bit to this understanding, but the picture is far from clear.
Finally, a better understanding of these methods opens the door to new and potentially much more powerful unsupervised learning algorithms. Several directions have been proposed here:

1. Using a smoother distribution than the empirical distribution to define the inner product. But why, fundamentally, might this be helpful?

2. Learning a density function from the mapping in order to compute likelihoods. Many questions remain to be studied there.

3. Learning higher-level abstractions on top of lower-level abstractions by iterating the unsupervised learning process in multiple "layers". Preliminary experiments on toy data suggests that this idea works, but why should it work? that remains to be shown.

4. Using the data to define the kernel. Another paper is in preparation that attempts to answer that question.

Acknowledgments

The authors would like to thank Yann Le Cun, Yves Grandvalet, Chris Williams, Michael Jordan and Bernhard Schölkopf for helpful feedback, and the following funding organizations: NSERC, MITACS, IRIS, and the Canada Research Chairs.

References

Amari, S. and Wu, S. (1999). Improving Support Vector Machine classifiers by modifying kernel functions. *Neural Networks*, 12:783–789. 21

Belkin, M. and Niyogi, P. (2002a). Laplacian eigenmaps and spectral techniques for embedding and clustering. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press. 2, 6, 21

Belkin, M. and Niyogi, P. (2002b). Laplacian eigenmaps for dimensionality reduction and data representation. Technical Report TR-2002-01, University of Chicago, Computer Science. 6

Belkin, M. and Niyogi, P. (2002c). Semi-supervised learning on manifolds. Technical Report TR-2002-12, University of Chicago, Computer Science. 6

Brand, M. (2003). Charting a manifold. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15. The MIT Press. 21

Chapelle, O., Schölkopf, B., and Weston, J. (2003). Semi-supervised learning through principal directions estimation. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15. The MIT Press. 7, 21

Christianini, N., Shawe-Taylor, J., and Kandola, J. (2002). Spectral kernel methods for clustering. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14. The MIT Press. 7

Chung, F. (1997). Spectral graph theory. In *CBMS Regional Conference Series*, volume 92. American Mathematical Society. 17

Cox, T. and Cox, M. (1994). *Multidimensional Scaling*. Chapman & Hall, London. 6

Diamantras, K. and Kung, S. (1996). *Principal Components Neural Networks: theory and applications*. Wiley. 11

Girolami, M. (2001). Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Computation*, 14(3):669–688. 21

He, X. and Niyogi, P. (2002). Locality preserving projections (lpp). Technical Report TR-2002-09, University of Chicago, Computer Science. 6

Hinton, G. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986. Lawrence Erlbaum, Hillsdale. 20

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800. 20

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press. 1, 4, 5

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. 6

Schölkopf, B., A. S. and Müller, K.-R. (1996). Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany. 2, 5

Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1999). *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA. 3, 5

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319. 5

Seeger, M. (2001). Learning with labeled and unlabeled data. Technical report, Edinburgh University. 7

Spielman, D. and Teng, S. (1996). Spectral partitionning works: planar graphs and finite element meshes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*. 17

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*, volume 15. The MIT Press. 21

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. 6

Vincent, P. and Bengio, Y. (2003). Manfold parzen windows. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15. The MIT Press. 21

Wahba, G. (1990). Spline models for observational data. In *CBMS-NSF Regional Conference Series in Applied Mathematics*, volume 59, Philadelphia, PA. Society for Industrial and Applied Mathematics (SIAM). 3

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982. 4, 5

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*, volume 13, pages 675–681. MIT Press. 6

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. 3, 7, 8, 14, 17, 21

Algorithm 1: Outline of a gradient-based online-algorithm for learning the leading eigenfunctions when p is not the empirical distribution.
Input: kernel K, number of desired eigenfunctions m, original density $p(x)$, conditional density for proposal distribution $q(y|x)$, type of kernel normalization (e.g. subtractive or divisive), class of functions $\mathcal{F}$ for the scaled eigenfunctions $g_i(x)$, class of functions $\mathcal{E}$ for the kernel expected value $e(x) = E_y[K(x, y)]$.
Output: estimated eigenvalues λ_i and estimated eigenfunctions $f_i(x) = g_i(x)/\sqrt{\lambda_i}$.

```
while (not stopping criterion)
   sample $x$ from $p(x)$
   sample $y$ from $q(y|x)$
   $w \leftarrow p(y)/p(x)$
   compute $K(x, y)$
   clear parameter gradients for function $e$
   compute $e(x)$
   assign gradient $2w(e(x) - K(x, y))$ to $e$'s output
   back-propagate gradients from $e$'s output to $e$'s parameters
   compute $e(y)$
   assign gradient $2w(e(y) - K(x, y))$ to $e$'s output
   back-propagate gradients from $e$'s output to $e$'s parameters
   update $e$'s parameters with one step of online gradient
   compute normalized $\tilde{K}(x, y)$ from $K(x, y)$ using $e(x)$ and $e(y)$
   compute $g_i(x)$ ($i = 1$ to $m$)
   update $\lambda_i$'s with a moving average of $g_i(x)^2$ ($i = 1$ to $m$)
   compute reconstructions $\hat{K}_i \leftarrow \hat{K}_{i-1} + g_i(x)g_i(y)$ ($i = 1$ to $m$)
   compute residues $\epsilon_i \leftarrow (\hat{K}_i - \tilde{K}(x, y))$ ($i = 1$ to $m$)
   assign gradient $2w\epsilon_i g_i(y)$ to $g_i(x)$'s output ($i = 1$ to $m$)
   assign gradient $2w\epsilon_i g_i(x)$ to $g_i(y)$'s output ($i = 1$ to $m$)
   back-propagate gradients from $g_i(x)$'s output to $g$'s parameters
   back-propagate gradients from $g_i(y)$'s output to $g$'s parameters
   update $g$'s parameters with one step of online gradient
   update online gradient learning rate
```