Journal of Machine Learning Research 9 (2008) 1711-1739 Submitted 8/06; Revised 3/07; Published 8/08

Regularization on Graphs with Function-adapted Diffusion Processes

Arthur D. Szlam ASZLAM@MATH.UCLA.EDU
Department of Mathematics
U.C.L.A., Box 951555
Los Angeles, CA 90095-1555

Mauro Maggioni MAURO.MAGGIONI@DUKE.EDU
Department of Mathematics and Computer Science
Duke University, Box 90320
Durham, NC, 27708

Ronald R. Coifman COIFMAN@MATH.YALE.EDU
Program in Applied Mathematics
Department of Mathematics
Yale University, Box 208283
New Haven,CT,06510

Editor: Zoubin Ghahrmani

Abstract

Harmonic analysis and diffusion on discrete data has been shown to lead to state-of-the-art algorithms for machine learning tasks, especially in the context of semi-supervised and transductive learning. The success of these algorithms rests on the assumption that the function(s) to be studied (learned, interpolated, etc.) are smooth with respect to the geometry of the data. In this paper we present a method for modifying the given geometry so the function(s) to be studied are smoother with respect to the modified geometry, and thus more amenable to treatment using harmonic analysis methods. Among the many possible applications, we consider the problems of image denoising and transductive classification. In both settings, our approach improves on standard diffusion based methods.

Keywords: diffusion processes, diffusion geometry, spectral graph theory, image denoising, transductive learning, semi-supervised learning

1. Introduction

Recently developed techniques in the analysis of data sets and machine learning use the geometry of the data set in order to study functions on it. In particular the idea of analyzing the data set and functions on it intrinsically has lead to novel algorithms with state-of-the-art performance in various problems in machine learning (Szummer and Jaakkola, 2001; Zhu et al., 2003; Zhou and Schlkopf, 2005; Belkin and Niyogi, 2003a; Mahadevan and Maggioni, 2007; Maggioni and Mhaskar, 2007). They are based on the construction of a diffusion, or an averaging operator K on the data set, dependent on its local, fine scale geometry. K, its powers, and the special bases associated to it, such as its eigenfunctions (Belkin and Niyogi, 2003a; Coifman et al., 2005a; Coifman and Lafon, 2006a) or its diffusion wavelets (Coifman and Maggioni, 2006) can be used to study the geometry of and analyze functions on the data set. Among other things, "diffusion analysis" allows us to introduce a notion of smoothness in discrete settings that preserves the relationships between smoothness,

sparsity in a "Fourier" basis, and evolution of heat that are well-known in Euclidean spaces (Zhou and Schlkopf, 2005).

One of the main contributions of this work is the observation that the geometry of the space is not the only important factor to be considered, but that the geometry and the properties of the function f to be studied (denoised/learned) should also affect the smoothing operation of the diffusion. We will therefore modify the geometry of a data set with features from f, and build K on the modified f-adapted data set. The reason for doing this is that perhaps f is not smooth with respect to the geometry of the space, but has structure that is well encoded in the features. Since the harmonic analysis tools we use are robust to complicated geometries, but are most useful on smooth functions, it is reasonable to let the geometry of the data set borrow some of the complexity of the function, and study a smoother function on a more irregular data set. In other words, we attempt to find the geometry so that the functions to be studied are as smooth as possible with respect to that geometry. On the one hand, the result is nonlinear in the sense that it depends on the input function f, in contrast with methods which consider the geometry of the data alone, independently of f. On the other hand, on the modified data set, the smoothing operator K will be linear, and very efficiently computable. One could generalize the constructions proposed to various types of processes (e.g., nonlinear diffusions).

The paper is organized as follows: in Section 2, we review the basic ideas of harmonic analysis on weighted graphs. In Section 3 we introduce the function-adapted diffusion approach, which aims to modify the geometry of a data set so that a function or class of functions which was non-smooth in the original geometry is smooth in the modified geometry, and thus amenable to the harmonic analysis in the new geometry. In Section 4 we demonstrate this approach in the context of the image denoising problem. In addition to giving easy to visualize examples of how the method works, we achieve state of the art results. In Section 5, we demonstrate the approach in the context of transductive learning. While here it is more difficult to interpret our method visually, we test it on a standard database, where it outperforms comparable "geometry only" methods on the majority of the data sets, and in many cases achieves state of the art results. We conclude by considering the under-performance of the method on some data sets, observing that in those examples (most of which are in fact the only artificial ones!), the geometry of the data suffices for learning the function of interests; and our method is superfluous.

2. Diffusion on Graphs Associated with Data-sets

An intrinsic analysis of a data set, modeled as a graph or a manifold, can be developed by considering a natural random walk K on it (Chung, 1997; Szummer and Jaakkola, 2001; Ng et al., 2001; Belkin and Niyogi, 2001; Zha et al., 2001; Lafon, 2004; Coifman et al., 2005a,b). The random walk allows to construct diffusion operators on the data set, as well as associated basis functions. For an initial condition δ_x, $K^t\delta_x(y)$ represents the probability of being at y at time t, conditioned on starting at x.

2.1 Setup and Notation

We consider the following general situation: the space is a finite weighted graph $G = (V,E,W)$, consisting of a set V (vertices), a subset E (edges) of $V \times V$, and a nonnegative function $W : E \to \mathbb{R}^+$ (weights). Without loss of generality we assume that there is an edge from x to $y \in V$, and write $x \sim y$, if and only if $W(x,y) > 0$. Notice that in this work W will usually be symmetric; that is

the edges will be undirected. The techniques we propose however do not require this property, and therefore can be used on data sets in which graph models with directed edges are natural.

We interpret the weight $W(x,y)$ as a measure of similarity between the vertices x and y. A natural filter acting on functions on V can be defined by normalization of the weight matrix as follows: let

$$d(x) = \sum_{y \in V} W(x,y)\,,$$

and let[1] the filter be

$$K(x,y) = d^{-1}(x)W(x,y)\,, \tag{1}$$

so that $\sum_{y\in V} K(x,y) = 1$, and so that multiplication Kf of a vector from the left is a local averaging operation, with locality measured by the similarities W. Multiplication by K can also be interpreted as a generalization of Parzen window type estimators to functions on graphs/manifolds. There are other ways of defining averaging operators. For example one could consider the heat kernel $e^{-t\mathcal{L}}$ where $\mathcal{L}$ is defined in (3) below, see also Chung (1997), or a bi-Markov matrix similar to W (Sinkhorn, 1964; Sinkhorn and Knopp, 1967; Soules, 1991; Linial et al., 1998; Shashua et al., 2005; Zass and Shashua, 2005).

In general K is not column-stochastic,[2] but the operation fK of multiplication on the right by a (row) vector can be thought of as a diffusion of the vector f. This filter can be iterated several times by considering the power K^t.

2.2 Graphs Associated with Data Sets

¿From a data set X we construct a graph G: the vertices of G are the data points in X, and weighted edges are constructed that connect nearby data points, with a weight that measures the similarity between data points. The first step is therefore defining these *local similarities*. This is a step which is data- and application-dependent. It is important to stress the attribute *local*. Similarities between far away data points are not required, and deemed unreliable, since they would not take into account the geometric structure of the data set. Local similarities are assumed to be more reliable, and non-local similarities will be inferred from local similarities through diffusion processes on the graph.

2.2.1 LOCAL SIMILARITIES

Local similarities are collected in a matrix W, whose rows and columns are indexed by X, and whose entry $W(x,y)$ is the similarity between x and y. In the examples we consider here, W will usually be symmetric, that is the edges will be undirected, but these assumptions are not necessary.

If the data set lies in $\mathbb{R}^d$, or in any other metric space with metric ρ, then the most standard construction is to choose a number ("local time") $\sigma > 0$ and let

$$W_\sigma(x,y) = h\left(\frac{\rho(x,y)^2}{\sigma}\right),$$

1. Note that $d(x) = 0$ if and only if x is not connected to any other vertex, in which case we trivially define $d^{-1}(x) = 0$, or simply remove x from the graph.
2. In particular cases K is a scalar multiple of a column-stochastic matrix, for example when D is a multiple of identity, which happens for example if G is regular and all the edges have the same weight.

for some function h with, say, exponential decay at infinity. A common choice is $h(a) = \exp(-a)$. The idea is that we expect that very close data points (with respect to ρ) will be similar, but do not want to assume that far away data points are necessarily different.

Let D be the diagonal matrix with entries given by d as in (2.1). Suppose the data set is, or lies on, a manifold in Euclidean space. In Lafon (2004), Belkin and Niyogi (2005), Hein et al. (2005), von Luxburg et al. (2004) and Singer (2006), it is proved that in this case, the choice of h in the construction of the weight matrix is in some asymptotic sense irrelevant. For a rather generic symmetric function h, say with exponential decay at infinity, $(I - D_\sigma^{-\frac{1}{2}} W_\sigma D_\sigma^{-\frac{1}{2}})/\sigma$, approaches the Laplacian on the manifold, at least in a weak sense, as the number of points goes to infinity and σ goes to zero. Thus this choice of weights is naturally related to the heat equation on the manifold. On the other hand, for many data sets, which either are far from asymptopia or simply do not lie on a manifold, the choice of weights can make a large difference and is not always easy. Even if we use Gaussian weights, the choice of the "local time parameter" σ can be nontrivial.

For each x, one usually limits the maximum number of points y such that $W(x,y) \neq 0$ (or non-negligible). Two common modifications of the construction above are to use either $\rho_\varepsilon(x,y)$ or $\rho_k(x,y)$ instead of ρ, where

$$\rho_\varepsilon(x,y) = \begin{cases} d(x,y) & \text{if } \rho(x,y) \leq \varepsilon; \\ \infty & \text{if } \rho(x,y) > \varepsilon \end{cases},$$

where usually ε is such that $h(\varepsilon^2/\sigma) << 1$, and

$$\rho_k(x,y) = \begin{cases} \rho(x,y) & \text{if } y \in n_k(x); \\ \infty & \text{otherwise.} \end{cases}$$

and $n_k(x)$ is the set of k nearest neighbors of x. This is for two reasons: one, often only very small distances give information about the data points, and two, it is usually only possible to work with very sparse matrices.[3] This truncation causes W to be not symmetric; if symmetry is desired, W may be averaged (arithmetically or geometrically) with its transpose.

A location-dependent approach for selecting the similarity measure is suggested in Zelnik-Manor and Perona (2004). A number m is fixed, and the distances at each point are scaled so the m-th nearest neighbor has distance 1; that is, we let $\rho_x(y,y') = \rho(y,y')/\rho(x,x_m)$, where x_m is the m-th nearest neighbor to x. Now ρ_x depends on x, so in order to make the weight matrix symmetric, they suggest to use the geometric mean of ρ_x and ρ_y in the argument of the exponential, that is, let

$$W_\sigma(x,y) = h\left(\frac{\rho_x(x,y)\rho_y(x,y)}{\sigma}\right), \tag{2}$$

with h, as above, decaying at infinity (typically, $h(a) = \exp(-a)$), or truncated at the k-th nearest neighbor. This is called the self-tuning weight matrix. There is still a timescale in the weights, but a global σ in the self-tuning weights corresponds to some location dependent choice of σ in the standard exponential weights.

3. However, methods of Fast Multipole of Fast Gauss type (Greengard and Rokhlin, 1988) may make it possible to work with dense matrices implicitly, with complexity proportional to the number of points. See Raykar et al. (2005) for a recent reference with applications to machine learning.

2.2.2 THE AVERAGING OPERATOR AND ITS POWERS

Multiplication by the normalized matrix K as in (1) can be iterated to generate a Markov process $\{K^t\}_{t\geq 0}$, and can be used to measure the strength of all the paths between two data points, or the likelihood of getting from one data point to the other if we constrain ourselves to only stepping between very similar data points. For example one defines the diffusion or spectral distance (Bérard et al., 1994; Coifman et al., 2005a; Coifman and Lafon, 2006a) by

$$\mathcal{D}^{(t)}(x,y) = ||K^t(x,\cdot) - K^t(y,\cdot)||_2 = \sqrt{\sum_{z\in X} |K^t(x,z) - K^t(y,z)|^2}\,.$$

The term diffusion distance was introduced in Lafon (2004), Coifman et al. (2005a) and Coifman and Lafon (2006a) and is suggested by the formula above, which expresses $\mathcal{D}^{(t)}$ as some similarity between the probability distributions $K^t(x,\cdot)$ and $K^t(y,\cdot)$, which are obtained by diffusion from x and y according to the diffusion process K. The term spectral distance was introduced in Bérard et al. (1994, see also references therein). It has recently inspired several algorithms in clustering, classification and learning (Belkin and Niyogi, 2003a, 2004; Lafon, 2004; Coifman et al., 2005a; Coifman and Lafon, 2006a; Mahadevan and Maggioni, 2005; Lafon and Lee, to appear, 2006; Maggioni and Mhaskar, 2007).

2.3 Harmonic Analysis

The eigenfunctions $\{\psi_i\}$ of K, satisfying

$$K\psi_i = \lambda_i \psi_i\,,$$

are are related,via multiplication by $D^{-\frac{1}{2}}$, to the eigenfunctions ϕ_i of the graph Laplacian (Chung, 1997), since

$$\mathcal{L} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} - I = D^{\frac{1}{2}} K D^{-\frac{1}{2}} - I\,. \tag{3}$$

They lead to a natural generalization of the Fourier analysis: any function $g \in \mathbb{L}^2(X)$ can be written as $g = \sum_{i\in I} \langle g, \phi_i\rangle \phi_i$, since $\{\phi_i\}$ is an orthonormal basis. The larger is i, the more oscillating the function ϕ_i is, with respect to the geometry given by W, and λ_i measures the frequency of ϕ_i. These eigenfunctions can be used for dimensionality reduction tasks (Lafon, 2004; Belkin and Niyogi, 2003a; Coifman and Lafon, 2006a; Coifman et al., 2005a; Jones et al., 2007a,b).

For a function g on G, define its gradient (Chung, 1997; Zhou and Schlkopf, 2005) as the function on the edges of G defined by

$$\nabla g(x,y) = W(x,y)\left(\frac{g(y)}{\sqrt{d(y)}} - \frac{g(x)}{\sqrt{d(x)}}\right) \tag{4}$$

if there is an edge e connecting x to y and 0 otherwise; then

$$||\nabla g(x)||^2 = \sum_{x\sim y} |\nabla g(x,y)|^2.$$

The smoothness of g can be measured by the Sobolev norm

$$||g||^2_{\mathcal{H}^1} = \sum_x |g(x)|^2 + \sum_x ||\nabla g(x)||^2\,. \tag{5}$$

The first term in this norm measures the size of the function g, and the second term measures the size of the gradient. The smaller $||g||_{\mathcal{H}^1}$, the smoother is g. Just as in the Euclidean case,

$$||g||^2_{\mathcal{H}^1} = ||g|^2_{\mathbb{L}^2(X,d)} - \langle g, \mathcal{L}g \rangle ;$$

thus projecting a function onto the first few terms of its expansion in the eigenfunctions of $\mathcal{L}$ is a smoothing operation.[4]

We see that the relationships between smoothness and frequency forming the core ideas of Euclidean harmonic analysis are remarkably resilient, persisting in very general geometries. These ideas have been applied to a wide range of tasks in the design of computer networks, in parallel computation, clustering (Ng et al., 2001; Belkin and Niyogi, 2001; Zelnik-Manor and Perona, 2004; Kannan et al., 2004; Coifman and Maggioni, 2007), manifold learning (Bérard et al., 1994; Belkin and Niyogi, 2001; Lafon, 2004; Coifman et al., 2005a; Coifman and Lafon, 2006a), image segmentation (Shi and Malik, 2000), classification (Coifman and Maggioni, 2007), regression and function approximation (Belkin and Niyogi, 2004; Mahadevan and Maggioni, 2005; Mahadevan et al., 2006; Mahadevan and Maggioni, 2007; Coifman and Maggioni, 2007).

2.4 Regularization by Diffusion

It is often useful to find the smoothest function $\tilde{f}$ on a data set X with geometry given by W, so that for a given f, $\tilde{f}$ is not too far from f; this task is encountered in problems of denoising and function extension. In the denoising problem, we are given a function $f + \eta$ from $X \to \mathbb{R}$, and η is Gaussian white noise of a given variance, or if one is ambitious, some other possibly stochastic contamination. We must find f. In the function extension or interpolation problem, a relatively large data set is given, but the values of f are known at only relatively few "labeled" points, and the task is to find f on the "unlabeled" points. Both tasks, without any a priori information on f, are impossible; the problems are underdetermined. On the other hand, it is often reasonable to assume f should be smooth, and so we are led to the problem of finding a smooth $\tilde{f}$ close to f.

In Euclidean space, a classical method of mollification is to run the heat equation for a short time with initial condition specified by f. It turns out that the heat equation makes perfect sense on a weighted graph: if f is a function on V, set $f_0 = f$, and $f_{k+1} = Kf$. If $g_k(x) = d^{\frac{1}{2}}(x) f_k(x)$,

$$g_{k+1} - g_k = \mathcal{L} g_k ,$$

so multiplication by K is a step in the evolution of the (density normalized) heat equation. Furthermore, a quick calculation shows this is the gradient descent for the smoothness energy functional $\sum ||\nabla g||^2$. We can thus do "harmonic" interpolation on X by iterating K (Zhu et al., 2003).

We can design more general mollifiers using an expansion on the eigenfunctions $\{\psi_i\}$ of K. For the rest of this section, suppose all inner products are taken against the measure d, that is, $\langle a, b \rangle = \sum a(x)b(x)d(x)$, and so ψ are orthonormal. Then $f = \sum \langle f, \psi_i \rangle \psi_i$ and one can define $\tilde{f}$, a smoothed version of f, by

$$\tilde{f} = \sum_i \alpha_i \langle f, \psi_i \rangle \psi_i$$

4. However it is well known that if f does not have uniform smoothness everywhere, the approximation by eigenfunctions is poor not only in regions of lesser smoothness, but the poor approximation spills to regions of smoothness as well. This lack of localization can be avoided with the multiscale constructions in Coifman and Maggioni (2006) and Maggioni and Mhaskar (2007).

for some sequence $\{\alpha_i\}$ which tends to 0 as $i \to +\infty$; in the interpolation problem, we can attempt to estimate the inner products $\langle f, \psi_i \rangle$, perhaps by least squares. Typical examples for α_i are:

(i) $\alpha_i = 1$ if $i < I$, and 0 otherwise (pure low-pass filter); I usually depends on a priori information on η, for example on the variance of η. This is a band-limited projection (with band I), see for example Belkin (2003).

(ii) $\alpha_i = \lambda_i^t$ for some $t > 0$, this corresponds to setting $\tilde{f} = K^t(f)$, that is, kernel smoothing on the data set, with a data-dependent kernel (Smola and Kondor, 2003; Zhou and Schlkopf, 2005; Chapelle et al., 2006).

(iii) $\alpha_i = P(\lambda_i)$, for some polynomial (or rational function) P, generalizing (ii). See, for example, Maggioni and Mhaskar (2007)

As mentioned, one can interpret $K^t f$ as evolving a heat equation on the graph with an initial condition specified by f. If we would like to balance smoothing by K with fidelity to the original f, we can choose $\beta > 0$ and set $f_0 = f$ and $f_{t+1} = (K f_t + \beta f)/(1+\beta)$; the original function is treated as a heat source. This corresponds at equilibrium to

(iv) $\alpha_i = \beta/(1+\beta-\lambda_i)$.

One can also consider families of nonlinear mollifiers, of the form

$$\tilde{f} = \sum_i m(\langle f, \psi_i \rangle)\psi_i \,,$$

where for example m is a (soft-)thresholding function (Donoho and Johnstone, 1994). In fact, m may be made even dependent on i. While these techniques are classical and well-understood in Euclidean space (mostly in view of applications to signal processing), it is only recently that research in their application to the analysis of functions on data sets has begun (in view of applications to learning tasks, see in particular Maggioni and Mhaskar 2007).

All of these techniques clearly have a regularization effect. This can be easily measured in terms of the Sobolev norm defined in (5): the methods above correspond to removing or damping the components of f (or $f+\eta$) in the subspace spanned by high-frequency ψ_i, which are the ones with larger Sobolev norm.

3. Function-adapted Kernels

The methods above are based on the idea that the function f to be recovered should be smooth with respect to W, but it can happen that an interesting function on data is not smooth with respect to the given geometry on that data. In this case we cannot directly bring to bear the full power of the methods described above. On the other hand, we have seen that these methods are well defined on any weighted graph. We thus propose to modify the geometry of W so that the function(s) to be recovered are as smooth as possible in the modified geometry. Even when f is not smooth, the geometry of W and f can interact in a structured way. We will attempt to incorporate the geometry of the function f (or a family of functions F) in the construction of the weights W; the hope is that we can convert structure to smoothness, and apply the methods of harmonic analysis to a smoother function on a rougher data set. In other words, we want our regularizer to take averages between

points where f has similar structure, in addition to being near to each other in terms of the given geometry of the data.

The simplest version of this idea is to only choose nonzero weights between points on the same level set of f. Then $||\nabla f||$ (with respect to W) is zero everywhere, and the function to be recovered is as smooth as possible. Of course knowing the level sets of f is far too much to ask for in practice. For example, in the function extension problem, if f has only a few values (e.g., for a classification task), knowing the level sets of f would be equivalent to solving the problem.

If we had some estimate $\tilde{f}$ for f, we could set

$$W^f(x,y) = \exp\left(-\frac{||x-y||^2}{\sigma_1} - \frac{|\tilde{f}(x)-\tilde{f}(y)|^2}{\sigma_2}\right), \tag{6}$$

so that when $\sigma_2 << \sigma_1$, the associated averaging kernel K will average locally, but much more along the (estimated) level sets of f than across them, because points on different level sets now have very weak or no affinity. This is related to ideas in Yaroslavsky (1985); Smith and Brady (1995) and Coifman et al. (2005a).

The estimate $\tilde{f}$ of f is just a simple example of a feature map. More generally, we set

$$W^f(x,y) = h_1\left(-\frac{\rho_1(x,y)^2}{\sigma_1}\right) h_2\left(-\frac{\rho_2(\mathcal{F}(f)(x),\mathcal{F}(f)(y))^2}{\sigma_2}\right), \tag{7}$$

where $\mathcal{F}(f)(x)$ is a set of features associated with f, evaluated at the data point x, ρ_1 is a metric on the data set, ρ_2 is a metric on the set of features, h_1 and h_2 are (usually exponentially) decaying functions, and σ_1 and σ_2 are "local time" parameters in data and feature space respectively. Such a similarity is usually further restricted as described at the end of Section 2.2.1. The idea here is to be less ambitious than (6), and posit affinity between points where we strongly believe f to have the same structure, and not necessarily between every point on an (estimated) level set. The averaging matrix K^f associated with W^f can then be used for regularizing, denoising and learning tasks, as described above. We call such a kernel a function-adapted kernel.

The way the function f affects the construction of K^f will be application- and data- specific, as we shall see in the applications to image denoising and graph transductive learning. For example, in the application to image denoising, $\mathcal{F}(f)(x)$ may be a vector of filter responses applied to the image f at location x. In the application to transductive classification, we are given C functions χ_i, defined by $\chi_i(x) = 1$ if x is labeled as a point in class i, and 0 otherwise (either the point is not labeled, or it is not in class i). We set $f = (\chi_i)_{i=1}^N$. Then $\mathcal{F}(f)(x)$ can be obtained by evaluating $K^t(\chi_i)$ at x, where K is a diffusion operator which only depends on the data set, and not on the χ_i's. In all applications, our idea is simply to to try to choose similarities, with the limited information about the function(s) to be recovered that we are given, so that the function(s) are as regular as possible with respect to the chosen similarities.

4. Application I: Denoising of Images

We apply function-adapted kernels to the task of denoising images. Not only this will be helpful to gain intuition about the ideas in Section 3 in a setting where our methods are easily visualized, but it also leads to state-of-art results.

Gray-scale images are often modeled as real-valued functions, or distributions, on Q, a fine discretization of the square $[0,1]^2$, and they are often analyzed, denoised, compressed, inpainted, deblurred as such, see for example Tschumperle (2002), Perona and Malik (1990), Rudin et al. (1992),

Chan and Shen (2005), Perona and Malik (1990), Tomasi and Manduchi (1998), Elad (2002), Boult et al. (1993), Chin and Yeh (1983), Davis and Rosenfeld (1978), Graham (1961), Huang et al. (1979), Lee (1980), Yin et al. (1996) and references therein. It is well known that images are not smooth as functions from Q to $\mathbb{R}$, and in fact the interesting and most important features are often exactly the non-smooth parts of f. Thus Fourier analysis and the heat equation on Q are not ideally suited for images; much of the work referenced above aims to find partial differential equations whose evolution smooths images without blurring edges and textures.

With the approach described in Section 3, unlike with many PDE-based image processing methods, the machinery of smoothing is divorced from the task of feature extraction. We build a graph $G(I)$ whose vertices are the pixels of the image and whose weights are adapted to the image structure, and use the diffusion on the graph with a fidelity term, as described in Section 2.4 to smooth the image, *considered as a function on the graph*. If we are able to encode image structure in the geometry of the graph in such a way that the image is actually smooth as a function on its graph, then the harmonic analysis on the graph will be well-suited for denoising that image. Of course, we have shifted part of the problem to feature extraction, but we will see that very simple and intuitive techniques produce state of the art results.

4.1 Image-adapted Graphs and Diffusion Kernels

To build the image-adapted graph we first associate a feature vector to each location x in the image I, defined on a square Q. A simple choice of $d+2$ dimensional feature vectors is obtained by setting two of the coordinates of the feature vector to be scaled versions of the coordinates of the corresponding pixel in the image αx, where $\alpha \geq 0$ is a scalar, and $x \in Q$. The remaining d features are the responses to convolution with d different filters $g_1, \cdots, g_d$, evaluated at location x. More formally, we pick a d-vector $g = (g_1, \cdots, g_d)$ of filters (i.e., real valued functions on Q), fix $\alpha \geq 0$, and map Q into $\mathbb{R}^{d+2}$ by a *feature map*

$$\begin{aligned} \mathcal{F}_{g,\alpha}(I) : Q &\to \mathbb{R}^{d+2} \\ x &\mapsto (\alpha x, f * g_1(x), \cdots, f * g_d(x)) \end{aligned}$$

This is an extremely flexible construction, and there are many interesting choices for the filters $\{g_i\}$. One could take a few wavelets or curvelets at different scales, or edge filters, or patches of texture, or some measure of local statistics. Also note there are many other choices of feature maps that are not obtained by convolution, see Section 4.1.2 for examples.

The graph $G(I)$ will have vertices given by $\mathcal{F}_{g,\alpha}(x)$, $x \in Q$. To obtain the weighted edges, set

$$\rho(x,y) = \rho_{g,\alpha}(x,y) = ||\mathcal{F}_{g,\alpha}(f)(x) - \mathcal{F}_{g,\alpha}(f)(y)||,$$

where $||\cdot||$ is a norm (e.g., Euclidean) in $\mathbb{R}^{d+2}$. The parameter α specifies the amount of weight to give to the original 2-d space coordinates of the pixels, and may be 0. Alternatively, instead of using a weight α, one can choose sets $S = S(x) \subset Q$ so that

$$\rho(x,y) = d_{g,S}(x,y) = \begin{cases} \rho_{g,0}(x,y) & \text{if } y \in S(x); \\ \infty & \text{otherwise.} \end{cases} . \tag{8}$$

In the discrete case, if we choose $S(x)$ to be the n nearest neighbors of x in the 2 space coordinates: we will write $\rho_{g,n}$, and if the filters are understood, just ρ_n.

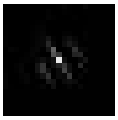

Figure 1: Above: image of Lena, with two locations highlighted. Left: row of the diffusion kernel corresponding to the upper-left highlighted area in the above image. Right: row of the diffusion kernel corresponding to the bottom-left highlighted area in the above image. The diffusion kernel averages according to different patterns in different locations. The averaging pattern on the right is also "non-local", in the sense that the averaging occurs along well-separated stripes, corresponding to the hair in the original picture.

For a fixed choice of metric ρ as above, and a "local time" parameter σ, we construct the similarity matrix W_σ as described in Section 2.2.1, and the associated diffusion kernel K as in (1).

In Figure 3 we explore the local geometry in patch space by projecting the set of patches around a given patch onto the principal components of the set of patches itself. Geometric structures of the set of patches, dependent on local geometry of the image (e.g., texture vs. edge) are apparent. The key feature of these figures is that the gray level intensity value is *smooth* as a function from the set of patches to $\mathbb{R}$, even when the intensity is not smooth in the original spatial coordinates.

We now describe some interesting choices for the feature maps $\mathcal{F}(I)$.

4.1.1 PATCH GRAPH

Let g_N be the set of N^2 filters $\{g_{i,j}\}_{i,j=1,\ldots,N}$, where $g_{i,j}$ is a $N \times N$ matrix with 1 in the i,j entry and 0 elsewhere. Then $\mathcal{F}_{g_N,0}$ is the set of patches of the image embedded in N^2 dimensions. The diffusion one gets from this choice of filters is the NL-means filter of Buades et al. (2005a). "NL"

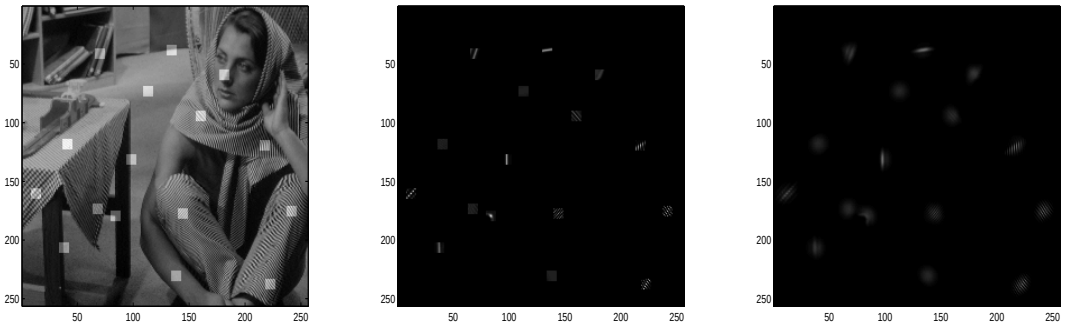

Figure 2: Left to right: image of Barbara, with several locations p_i highlighted; $K^t(p_i,\cdot)$, for $t = 1,2$.

stands for Non-Local; in the paper, they proposed setting $\alpha = 0$. In a later paper they add some locality constraints; see Buades et al. (2005b) and Mahmoudi (2005). We wish to emphasize that smoothing with the NL-means filter is not, in any sort of reasonable limit, a 2-*d* PDE; rather, it is the heat equation on the set of patches of the image!

Note the embedding into 5×5 patches is the same embedding (up to a rotation) as into 5×5 DCT coordinates, and so the weight matrices constructed from these embeddings are the same. On the other hand, if we attenuate small filter responses, the weight matrices for the two filter families will be different.

4.1.2 BOOTSTRAPPING A DENOISER; OR DENOISED IMAGES GRAPH

Different denoising methods often pick up different parts of the image structure, and create different characteristic artifacts. Suppose we have obtained denoised images $f_1,...,f_d$, from a noisy image f. To make use of the various sensitivities, and rid ourselves of the artifacts, we could embed pixels $x \in Q$ into $\mathbb{R}^{d+2}$ by $x \mapsto (\alpha x, f_1(x),...,f_d(x))$. In other words we interpret $(f_i(x))_{i=1,...,d}$ as a feature vector at x. This method is an alternative to "cycle spinning"(Coifman and Donoho, 1995), that is, simply averaging the different denoisings.

In practice, we have found that a better choice of feature vector is $f_{\sigma(1)}(x),...,f_{\sigma(d)}(x)$, where σ is a random permutation of $\{1,...,d\}$ depending on x. The idea is to mix up the artifacts from the various denoisings. Note that this would not affect standard averaging, since $\sum f_i(x) = \sum f_{\sigma(i)}$.

4.2 Image Graph Denoising

Once we have the graph W and normalized diffusion K, we use K to denoise the image. The obvious bijection from pixels to vertices in the image graph induces a correspondence between functions on pixels (such as the original image) and functions on the vertices of the graph. In particular the original image can be viewed as a function I on $G(I)$. The functions K^tI are smoothed versions of I with respect to the geometry of $G(I)$. If the graph was simply the standard grid on Q, then K would be nothing other than a discretization of the standard two-dimensional heat kernel, and K^tI would be the classical smoothing of I induced by the Euclidean two-dimensional heat kernel, associated with the classical Gaussian scale space (we refer the reader to Witkin, 1983; Koenderink,

1984; Lindeberg, 1994, and references therein). In our context K^t is associated with a scale space induced by $G(I)$, which is thus a nonlinear scale space (in the sense that it depends on the original

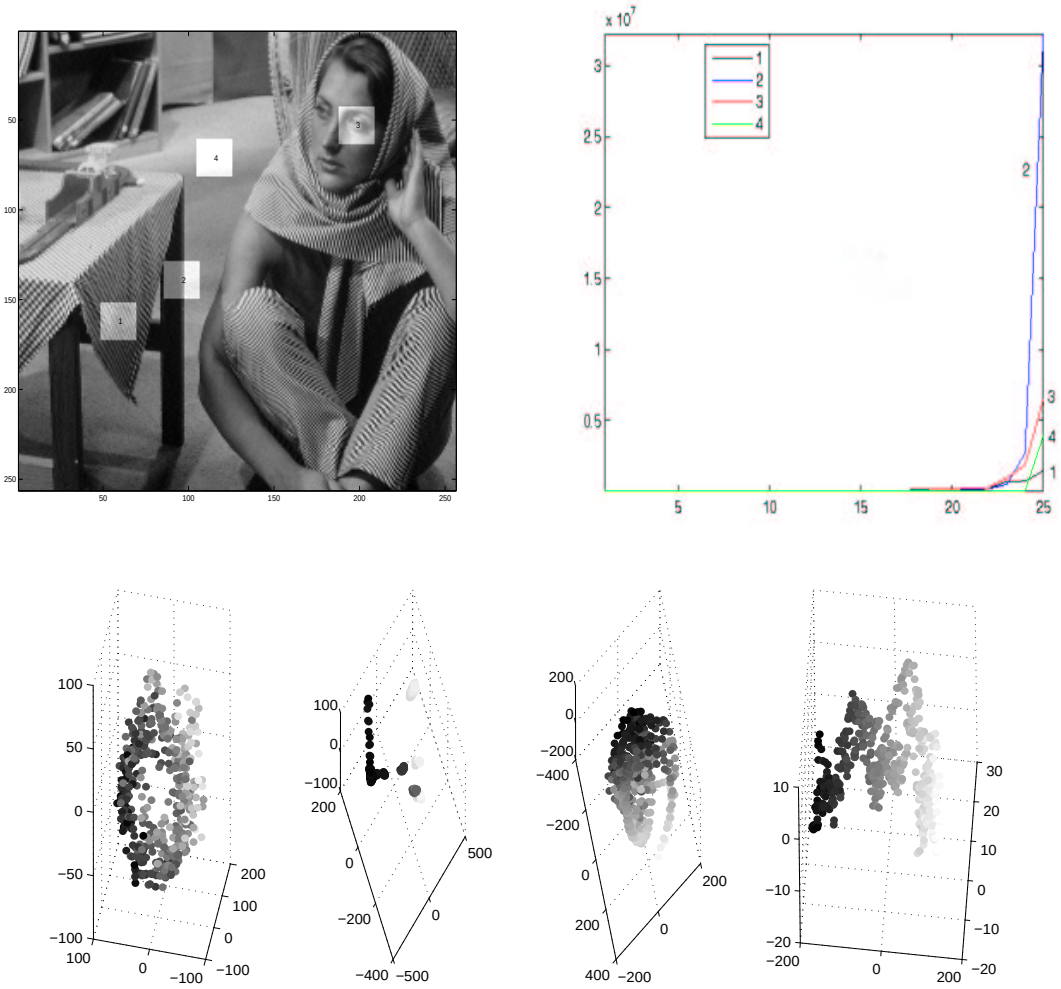

Figure 3: Top left: image of Barbara, with 4 square 10×10 pixel regions highlighted. The 5×5 patches in each region are considered as 25 dimensional vectors, and top right we plot the singular values of their covariance matrix. At the bottom, we project the 25-dimensional points in each region on their top 3 principal components, and the color is the value of the image at each point. In region 1, note how the (approximate) periodicity of the texture in region 1 is reflected in the tubular shape of the projection; in region 2, the portions of the image on different sides of the edge are disconnected in the feature space, and note the higher dimensionality, as measured by the singular values; for region 3, note the higher dimensionality (slower decay of the singular values) compared to regions 1 and 4; for region 4 note the very small dimensionality. Most importantly, note that in each region, the gray level pixel value is smooth as a function of the patches.

image I). In fact $G(I)$, as described above, is often a point cloud in high-dimensional space, where closeness in those high-dimensional space represents similarity of collections of pixels, and/or of their features, in the original two-dimensional domain of I.

We can balance smoothing by K with fidelity to the original noisy function by setting $f_{t+1} = (Kf_t + \beta f)/(1+\beta)$ where $\beta > 0$ is a parameter to be chosen, and large β corresponds to less smoothing and more fidelity to the noisy image. This is a standard technique in PDE based image processing, see Chan and Shen (2005) and references therein. If we consider iteration of K as evolving a heat equation, the fidelity term sets the noisy function as a heat source, with strength determined by β. Note that even though when we smooth in this way, the steady state is no longer the constant function, we still do not usually wish to smooth to equilibrium. We refer the reader to Figure 4 for a summary of the algorithm proposed.

$\tilde{I} \leftarrow$ `DenoiseImage`(I,t)

// Input:
// I : an image
// t : amount of denoising

// Output:
// $\tilde{I}$: a denoised version of I.

1. Construct a graph G associated with I, in any of the ways discussed in Section 4.
2. Compute the associated I-adapted diffusion operator K^I.
3. set $\tilde{I} \leftarrow (K^I)^t I$.

Figure 4: Pseudo-code for denoising an image

4.3 Examples

Figure 5 displays examples of denoising with a diffusion on an image graph. On the top left of the figure we have the noisy image f_0; the noise is $N(0,.0244)$. On the top right of Figure 5, we denoise the image using a 7×7 NL-means type patch embedding as described in Section 4.1.1. We set

$$W(k,j) = e^{-\rho_{81}^{\sim}(k,j)^2/.3}$$

where $\rho_{81}^{\sim}$ is the distance in the embedding, restricted to 81 point balls in the 2-d metric; that is we take $S(k)$ in Equation (8) to be the 81 nearest pixels to pixel k in the 2-d metric. We then normalize $K = D^{-1}W$ and denoise the image by applying K three times with a fidelity term of .07; that is, $f_{t+1} = (Kf_t + .07f_0)/(1.07)$, and the image displayed is f_3. The parameters were chosen by hand.

In the bottom row of Figure 5: on the bottom left, we sum 9 curvelet denoisings. Each curvelet denoisings is a reconstruction of the noisy image f_0 shifted either 1, 2, or 4 pixels in the vertical and/or horizontal directions, using only coefficients with magnitudes greater than 3σ. To demonstrate bootstrapping, or cycle spinning by diffusion, we embed each pixel in $\mathbb{R}^9$ using the 9 curvelet denoisings as coordinates. We set

$$W(k,j) = e^{-\rho_{81}^{\sim}(k,j)^2/.03}$$

Figure 5: 1) Lena with Gaussian noise added. 2) Denoising using a 7×7 patch graph. 3) Denoising using hard thresholding of curvelet coefficients. The image is a simple average over 9 denoisings with different grid shifts. 4) Denoising with a diffusion built from the 9 curvelet denoisings.

where $\tilde{\rho_{81}}$ is the distance in the embedding, and again we take $S(k)$ in Equation (8) to be the 81 nearest pixels to pixel k in the 2-d metric. We then normalize $K = D^{-1}W$ and denoise the image by applying K ten times with a fidelity term of .1; that is $f_{t+1} = (Kf_t + .1f_0)/(1.1)$, and f_{10} is displayed. The results are on the bottom right of Figure 5. We are able to greatly reduce the artifacts from a simple average of the curvelet denoisings.

5. Application II: Graph Transductive Learning

We apply function adapted approach to the transductive learning problem, and give experimental evidence demonstrating that using function adapted weights can improve diffusion based classifiers.

In a transductive learning problem one is given a few "labeled" examples $\tilde{X} \times \tilde{F} = \{(x_1, y_1), \ldots, (x_p, y_p)\}$ and a large number of "unlabeled" examples $X \setminus \tilde{X} = \{x_{p+1}, \ldots, x_n\}$. The goal is to estimate the conditional distributions $F(y|x)$ associated with each available example x (labeled or unlabeled).

For example $\tilde{F}$ may correspond to labels for the points $\tilde{X}$, or the result of a measurement at the points in $\tilde{X}$. The goal is to extend $\tilde{F}$ to a function F defined on the whole X, that is consistent with unseen labels/measurements at points in $X \setminus \tilde{X}$.

This framework is of interest in applications where it is easy to collect samples, that is, X is large, however it is expensive to assign a label or make a measurement at X, so only a few labels/measurements are available, namely at the points in $\tilde{X}$. The points in $X \setminus \tilde{X}$, albeit unlabeled, can be used to infer properties of the structure of the space (or underlying process/probability distribution) that is potentially useful in order to extend $\tilde{F}$ to F. Data sets with internal structures or geometry are in fact ubiquitous.

If F is smooth with respect to the data, an intrinsic analysis on the data set, such as the one possible by the use of diffusion processes and the associated Fourier and multi-scale analyses, fits very well in the transductive learning framework. In several papers a diffusion process constructed on X has been used for finding F directly (Zhou and Schlkopf, 2005; Zhu et al., 2003; Kondor and Lafferty, 2002) and indirectly, by using adapted basis functions on X constructed from the diffusion, such as the eigenfunctions of the Laplacian (Coifman and Lafon, 2006a,b; Lafon, 2004; Coifman et al., 2005a,b; Belkin and Niyogi, 2003b; Maggioni and Mhaskar, 2007), or diffusion wavelets (Coifman and Maggioni, 2006; Mahadevan and Maggioni, 2007; Maggioni and Mahadevan, 2006; Mahadevan and Maggioni, 2005; Maggioni and Mahadevan, 2005).

We will try to modify the geometry of the unlabeled data so that F is as smooth as possible with respect to the modified geometry. We will use the function adapted approach to try to learn the correct modification.

5.1 Diffusion for Classification

We consider here the case of classification, that is, F takes only a small number of values (compared to the cardinality of X), say $\{1, \ldots, k\}$. Let C_i, $i \in \{1, \ldots k\}$, be the classes, let C_i^{lab} be the labeled data points in the ith class, that is, $C_i = \{x \in \tilde{X} : \tilde{F} = i\}$, and let χ_i^{lab} be the characteristic function of those C_i, that is, $\chi_i^{lab}(x) = 1$ if $x \in C_i$, and $\chi_i^{lab}(x) = 0$ otherwise.

A simple classification algorithm can be obtained as follows (Szummer and Jaakkola, 2001):

(i) Build a geometric diffusion K on the graph defined by the data points X, as described in Section 2.2.1.

(ii) Use a power of K to smooth the functions χ_i^{lab}, exactly as in the denoising algorithm described above, obtaining functions $\overline{\chi_i^{lab}}$:

$$\overline{\chi_i^{lab}} = K^t \chi_i^{lab} .$$

The parameter t can be chosen by cross-validation.

(iii) Assign each point x to the class

$$\mathrm{argmax}_i \overline{\chi_i^{lab}}(x) .$$

This algorithm takes into account the influence of the labeled points on the unlabeled point to be classified, where the measure of influence is based on the weighted connectivity of the whole data set. If we average with a power of the kernel we have constructed, we count the number and strength of all the paths of length t to the various classes from a given data point. As a consequence, this method is more resistant to noise than, for example, a simple nearest neighbors (or also a geodesic nearest neighbors) method, as changing the location or class of a small number of data points does not change the structure of the whole network, while it can change the class label of a few nearest neighbors.

For each i, the "initial condition" for the heat flow given by χ_i^{lab} considers all the unlabeled points to be the same as labeled points not in C_i. Since we are solving many one-vs-all problems, this is reasonable; but one also may want to set the initial condition $\chi_i^{lab}(x) = 1$ for $x \in C_i^{lab}$, $\chi_i^{lab}(x) = -1$ for $x \in C_j^{lab}$, $j \neq i$, and $\chi_i^{lab}(x) = 0$ for all other x. It can be very useful to change the initial condition to a boundary condition by resetting the values of the labeled points after each application of the kernel. For large powers, this is equivalent to the harmonic classifier of Zhu et al. (2003), where the χ_i^{lab} is extended to the "harmonic" function with given boundary values on the labeled set. Just as in the image denoising examples, it is often the case that one does not want to run such a harmonic classifier to equilibrium, and we may want to find the correct number of iterations of smoothing by K and updating the boundary values by cross validation.

We can also use the eigenfunctions of K (which are also those of the Laplacian $\mathcal{L}$) to extend the classes. Belkin (2003) suggests using least squares fitting in the embedding defined by the first few eigenfunctions $\phi_1, \ldots, \phi_N$ of K. Since the values at the unlabeled points are unknown, we regress only to the labeled points; so for each χ_i^{lab}, we need to solve

$$\operatorname{argmin}_{\{a_l\}} \sum_{x \text{ labeled}} \left| \sum_{l=1}^{N} a_{il} \phi_l(x) - \chi_i^{lab}(x) \right|^2,$$

and extend the χ_i^{lab} to

$$\overline{\chi_i^{lab}} = \sum_{l=1}^{N} a_{il} \phi_i.$$

The parameter N controls the bias-variance tradeoff: smaller N implies larger bias of the model (larger smoothness)[5] and decreases the variance, while larger N has the opposite effect. Large N thus corresponds to small t in the iteration of K.

5.2 Function Adapted Diffusion for Classification

If the structure of the classes is very simple with respect to the geometry of the data set, then smoothness with respect to this geometry is precisely what is necessary to generalize from the labeled data. However, it is possible that the classes have additional structure on top of the underlying data set, which will not be preserved by smoothing geometrically. In particular at the boundaries between classes we would like to filter in such a way that the "edges" of the class function are preserved. We will modify the diffusion so it flows faster along class boundaries and slower across them, by using function-adapted kernels as in (7). Of course, we do not know the class boundaries: the func-

5. On the other hand, extending with small numbers of eigenfunctions creates "ripples"; that is, the Gibbs phenomenon. Techniques for avoiding the Gibbs phenomenon are discussed in Maggioni and Mhaskar (2007).

$F \leftarrow$ `ClassifyWithAdaptedDiffusion`$(X, \tilde{X}, \{\chi_i\}_{i=1,\ldots,N}, t_1, \beta, t_2)$

// Input:
// $X := \{x_i\}$: a data set
// $\tilde{X}$: a subset of X, representing the labeled set
// $\{\chi_i\}_{i=1,\ldots,N}$: set of characteristic functions of the classes, defined on $\tilde{X}$
// β : weight of the tuning parameter

// Output:
// C : function on X, such that $C(x)$ is the class to which $x \in X$ is estimated to belong.

1. Construct a weighted graph G associated with X, in any of the ways discussed.
2. Compute the associated diffusion operator K as in (1).
3. Compute guesses at the soft class functions $\overline{\chi_i}$ using any of the methods in Section 5.1, or any other method, and for multi-class problems, set
$$c_i(x) = \frac{\overline{\chi_i}(x)}{\sum_i |\overline{\chi_i(x)}|}.$$
4. Using the c_i as features, or $\overline{\chi_i}$ for two class problems, construct a new graph with kernel K' from the similarities as in Equation (7), with $\sigma_2 = \beta\sigma_1$.
5. Finally, find $C(x)$ using any of the methods in Section 5.1 and the kernel K'

Figure 6: Pseudo-code for learning of a function based on diffusion on graphs

tions $\{\chi_i\}$ are initially given on a (typically small) subset $\tilde{X}$ of X, and hence a similarity cannot be immediately defined in a way similar to (7).

We use a bootstrapping technique. We first use one of the algorithms above, which only uses similarities between data points ("geometry"), to generate the functions $\overline{\chi_i}$. We then use these functions to design a function-adapted kernel, by setting

$$\mathcal{F}(\{\chi_i\})(x) := (c_i(x))_{i=1,\ldots,k},$$

and then define a kernel as in (7). Here the c_i's are normalized confidence functions defined by

$$c_i(x) = \frac{\overline{\chi_i}(x)}{\sum_i |\overline{\chi_i(x)}|}.$$

In this way, if several classes claim a data point with some confidence, the diffusion will tend to average more among other points which have the same ownership situation when determining the value of a function at that data point. The normalization, besides having a clear probabilistic interpretation when the $\overline{\chi_i}$ are positive, also achieves the effect of not slowing the diffusion when there is only one possible class that a point could be in, for example, if a data point is surrounded by points of a single class, but is relatively far from all of them.

We summarize the algorithm in Figure 6. In the examples below we simply let ρ_2 be the metric of $\mathbb{R}^k$, and also let $h_2(a) = h_1(a) = e^{-a}$. The ratio β between σ_2 and σ_1, however, is important,

since it measures the trade-off between the importance given to the geometry of X and that of the set of estimates $\{(\overline{\chi_i}(x))_{i=1,\ldots,k}\}_{x\in X} \subseteq \mathbb{R}^k$.

We wish to emphasize the similarity between this technique and those described in Section 4 and especially Section 4.1.2. We allow the geometry of the data set to absorb some of the complexity of the classes, and use diffusion analysis techniques on the modified data set. The parallel with image denoising should not be unexpected: the goal of a function-adapted kernel is to strengthen averaging along level lines, and this is as desirable in image denoising as in transductive learning.

We remark that even if the c_i's are good estimates of the classes, they are not necessarily good choices for extra coordinates: for example, consider a two class problem, and a function c which has the correct sign on each class, but oscillates wildly. On the other hand, functions which are poor estimates of the classes could be excellent extra coordinates as long as they oscillate slowly parallel to the class boundaries. Our experience suggests, consistently with these considerations, that the safest choices for extra coordinates are very smooth estimates of the classes. In particular, of the three methods of class extension mentioned above, the eigenfunction method is often not a good choice for extra coordinates because of oscillation phenomena; see the examples in Section 5.4.

5.3 Relationship Between Our Methods and Previous Work

In Coifman et al. (2005a) the idea of using the estimated classes to warp the diffusion is introduced. They suggest, for each class C_n, building the modified weight matrix $W_n(i,j) = W(i,j)\overline{\chi_n^{lab}}(i)\overline{\chi_n^{lab}}(j)$, normalizing each W_n, and using the W_n to diffuse the classes. Our approach refines and generalizes theirs, by collecting all the class information into a modification of the metric used to build the kernel, rather than modifying the kernel directly. The tradeoff between geometry of the data and geometry of the (estimated/diffused) labels is made explicit and controllable.

In Zhu et al. (2003) it is proposed to adjust the graph weights to reflect prior knowledge. However, their approach is different than the one presented here. Suppose we have a two class problem. They add to each node of the graph a "dongle" node with transition probability β, which they leave as a parameter to be determined. They then run the harmonic classifier (Zhu et al., 2003) with the confidence function (ranging from 1 to -1) from a prior classifier as the boundary conditions on all the dongle nodes. Thus their method sets a tension between the *values* of the prior classifier and the harmonic classifier. Our method does not suggest values for the soft classes based on the prior classifier; rather, it uses this information to suggest modifications to the graph weights between unlabeled points.

5.4 Examples

We present experiments that demonstrate the use of function-adapted weights for transductive classification. We find that on many standard benchmark data sets, classification rate is improved using function-adapted weights instead of "geometry only" weights in diffusion based classifiers.

We use the data sets of Chapelle et al. (2006) and the first 10,000 images in the MNIST data set. At the time this article was written, the respective data sets are available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html` and `http://yann.lecun.com/exdb/mnist/`, with an extensive review of the performance of existing algorithms available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.pdf`, and at `http://yann.lecun.com/exdb/mnist/`.

All the data sets were reduced to 50 dimensions by principal components analysis. In addition, we smooth the MNIST images by convolving 2 times with an averaging filter (a 3×3 all ones matrix). The convolutions are necessary if we want the MNIST data set to resemble a Riemannian manifold; this is because if one takes an image with sharp edges and considers a smooth family of smooth diffeomorphisms of $[0,1] \times [0,1]$, the set of images obtained under the family of diffeomorphisms is not necessarily a (differentiable) manifold (see Donoho and Grimes 2002, and also Wakin et al. 2005). However, if the image does not have edges, then the family of morphed images is a manifold.[6]

We do the following:

x1. Choose 100 points as labeled. Each of the benchmark data sets of Chapelle et al., has 12 splits into 100 labeled and 1400 unlabeled points; we use these splits. In the MNIST data set we label points 1001 through 1100 for the first split, 1101 to 1200 for the second split, etc, and used 12 splits in total. Denote the labeled points by L, let C_i the ith class, and let χ_i^{lab} be 1 on the labeled points in the ith class, -1 on the labeled points of the other classes, and 0 elsewhere.

x2. Construct a Gaussian kernel W with k nearest neighbors, $\sigma = 1$, and normalized so the jth neighbor determines unit distance in the self tuning normalization (Equation 2), where $\{k, j\}$ is one of $\{9,4\}$, $\{13,9\}$, $\{15,9\}$, or $\{21,15\}$.

x3. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where $\overline{\chi_i^{lab}}(x)$ are constructed using the harmonic classifier with the number of iterations chosen by leave-20-out cross validation from 1 to 250. More explicitly: set $g_i^0 = \chi_i^{lab}$. Set $g_i^N(x) = (Kg_i^{N-1})(x)$ if $x \notin L$, $g_i^N(x) = 1$ if $x \in C_i \bigcap L$, and $g_i^N(x) = 0$ if $x \in L \backslash C_i$, and K is W normalized to be averaging. Finally, set $\overline{\chi_i^{lab}} = g_i^N(x)$, where N is chosen by leave-10-out cross validation between 1 and 250 (C_i and L are of course reduced for the cross validation).

x4. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where the $\overline{\chi_i^{lab}}(x)$ are constructed using least squares regression in the (graph Laplacian normalized) eigenfunction embedding, with the number of eigenfunctions cross validated; that is, for each χ_i^{lab}, we solve

$$\text{argmin}_{\{a_l\}} \sum_{x \text{ labeled}} \left| \sum_{l=1}^{N} a_{il} \phi_l(x) - \chi_i(x) \right|^2,$$

and extend the χ_i^{lab} to

$$\overline{\chi_i^{lab}} = \sum_{l=1}^{N} a_{il} \phi_i.$$

The ϕ are the eigenfunctions of $\mathcal{L}$, which is W normalized as a graph Laplacian, and N is chosen by leave-10-out cross validation.

6. For the most simple example, consider a set of $n \times n$ images where each image has a single pixel set to 1, and every other pixel set to 0. As we translate the on pixel across the grid, the difference between each image and its neighbor is in a new direction in R^{n^2}, and thus there is no reasonable tangent. The same thing is true for translates of a more complicated binary image, and translates of any image with an edge. One could complain that this is an artifact of the discrete grid, but it is easy to convince yourself that the set of translates of a characteristic function in $L^2(\mathbb{R})$ does not have a tangent anywhere- the tangent direction of the curve defined by the translates of a function is exactly the derivative of the function.

x5. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where $\overline{\chi_i^{lab}}(x)$ are constructed by smoothing χ_i^{lab} with K. More explicitly: set $g_i^0 = \chi_i^{lab}$. Set $g_i^N = W g_i^{N-1}$, where K is W normalized to be averaging; and finally, let $\overline{\chi_i^{lab}} = g_i^N(x)$, where N is chosen by leave-10-out cross validation between 1 and 250 (C_i and L are of course reduced for the cross validation).

We also classify the unlabeled points using a function-adapted kernel. Using the $\overline{\chi_i^{lab}}$ from the harmonic classifier at steady state ($N = 250$), we do the following:

x6. If the problem has more than two classes, set

$$c_i(x) = \frac{g_i^{250}(x)}{\sum_i |g_i^{250}(x)|},$$

else, set $c_i(x) = g_i^{250}(x)$

x7. Using the c_i as extra coordinates, build a new weights $\tilde{W}$. The extra coordinates are normalized to have average norm equal to the average norm of the original spatial coordinates; and then multiplied by the factor β, where β is determined by cross validation from $\{1,2,4,8\}$. The modified weights are constructed using the nearest neighbors from the original weight matrix, exactly as in the image processing examples.

x8. Use the function dependent $\tilde{K}$ to estimate the classes as in (x3).

x9. Use the function dependent $\tilde{L}$ to estimate the classes as in (x4).

x10. Use the function dependent $\tilde{K}$ to estimate the classes as in (x5).

We also repeat these experiments using the smoothed classes as an initial guess, and using the eigenfunction extended classes as initial guess. The results are reported in the Figures 7, 8, and 9. Excepting the data sets g241c, gc241n, and BCI, there is an almost universal improvement in classification rate using function-adapted weights instead of "geometry only" weights over all choices of parameters and all methods of initial soft class estimation.

In addition to showing that function adapted weights often improve classification using diffusion based methods, the results we obtain are very competitive and in many cases better than all other methods listed in the extensive comparative results presented in Chapelle et al. (2006), also available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.pdf`. In Figure 10 we attempt a comparison. For every data set, we report the performance of the best classifier (with model selection, and cross-validated performance) among all the ones considered in Chapelle et al. (2006). We also report the performance of our best classifier, among the ones we considered, corresponding to different choices of the two parameters for the self-tuning nearest-neighbor graph and initial smoothing (but with other parameters cross-validated). This comparison is unfair in many respects, for us in that we give the best choice over the two graph parameters (out of the four pairs we tested) and choice of initial class estimation (three tested), and against us considering the large number of algorithms listed in Chapelle et al. (2006). Nevertheless it demonstrates that the proposed algorithms on 3 out of 7 data sets can outperform all the algorithms considered in Chapelle et al. (2006).

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.2	2.9	2.5	2.6	2.2
USPS	4.9	4.1	5.0	4.1	4.2	3.6
BCI	45.9	45.5	44.9	44.7	47.4	48.7
g241c	31.5	31.0	34.2	32.7	23.1	41.3
COIL	14.3	12.0	13.4	11.1	16.8	15.1
gc241n	25.5	24.7	27.1	25.9	13.9	35.7
text	25.5	23.7	26.3	24.0	26.4	25.4
MNIST	9.4	8.5	9.0	7.9	9.4	8.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.2	2.7	2.1	2.6	2.2
USPS	5.2	4.2	5.2	4.0	4.0	3.3
BCI	47.6	47.4	45.0	45.5	48.2	48.6
g241c	30.7	31.2	33.3	32.0	21.7	31.7
COIL	17.2	16.7	16.0	15.1	21.9	19.0
gc241n	23.1	21.6	25.3	22.8	11.1	24.0
text	25.2	23.0	25.5	23.3	26.9	24.0
MNIST	10.0	9.2	10.1	8.7	9.7	8.5

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.3	2.8	2.2	2.6	1.9
USPS	5.0	4.0	5.2	3.9	3.9	3.3
BCI	48.2	48.0	45.9	46.1	47.6	47.9
g241c	30.5	30.4	32.8	31.2	21.2	29.7
COIL	18.0	17.0	16.2	15.2	22.9	19.9
gc241n	24.5	21.7	26.2	23.1	11.1	17.7
text	25.1	22.4	25.7	22.3	25.6	22.9
MNIST	10.3	9.2	10.0	8.9	9.6	8.3

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.6	2.9	2.6	2.0	2.1
USPS	5.6	4.7	5.6	4.4	4.4	3.7
BCI	48.2	48.5	46.3	46.7	48.9	48.5
g241c	28.5	28.2	32.1	29.4	18.0	23.6
COIL	19.8	19.3	19.2	17.9	26.3	24.1
gc241n	21.8	20.5	24.6	21.7	9.2	14.2
text	25.1	22.3	25.6	22.7	25.4	23.2
MNIST	10.8	10.0	10.7	9.7	10.8	10.0

Figure 7: Various classification error percentages. Each pair of columns corresponds to a smoothing method; the right column in each pair uses function adapted weights, with c_i determined by the harmonic classifier. KS stands for kernel smoothing as in (x5), FAKS for function adapted kernel smoothing as in (x10), HC for harmonic classifier as in (x3), FAHC for function adapted harmonic classifier as in (x8), EF for eigenfunctions as in (x4), and FAEF for function adapted eigenfunctions as in (x9). The Gaussian kernel had k neighbors, and the jth neighbor determined unit distance in the self-tuning construction, where counterclockwise, from the top left, $\{k, j\}$ is $\{9,4\}$, $\{13,9\}$, $\{15,9\}$, and $\{21,15\}$. Notice that excepting the data sets g241c, gc241n, and BCI, there is an almost universal improvement in classification error with function-adapted weights.

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.4	2.9	2.4	2.6	2.1
USPS	4.9	4.6	5.0	4.6	4.2	3.3
BCI	45.9	47.0	44.9	45.3	47.4	47.8
g241c	31.5	29.3	34.2	29.2	23.1	33.1
COIL	14.3	13.3	13.4	12.4	16.9	16.8
gc241n	25.5	21.3	27.1	22.5	13.9	23.0
text	25.5	24.5	26.3	25.0	26.4	24.6
MNIST	9.4	7.9	9.0	7.7	9.4	7.3

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.2	2.7	2.1	2.6	2.1
USPS	5.2	4.3	5.2	4.0	4.0	3.5
BCI	47.6	48.7	45.0	46.5	48.2	49.1
g241c	30.7	27.9	33.3	27.7	21.7	28.1
COIL	17.2	17.6	16.0	15.5	22.5	20.3
gc241n	23.1	17.9	25.3	19.3	11.1	21.0
text	25.2	23.8	25.5	23.7	26.9	24.5
MNIST	10.0	8.2	10.1	8.2	9.7	7.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.5	2.8	2.2	2.6	1.9
USPS	5.0	4.0	5.2	3.9	3.9	3.4
BCI	48.2	48.6	45.9	46.5	47.6	48.1
g241c	30.5	26.9	32.8	27.9	21.2	27.3
COIL	18.0	17.6	16.2	15.8	22.3	21.0
gc241n	24.5	19.7	26.2	20.8	11.1	19.5
text	25.1	22.8	25.7	23.3	25.6	23.4
MNIST	10.3	8.3	10.0	7.9	9.6	7.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.6	2.9	2.6	2.0	2.1
USPS	5.6	4.9	5.6	4.2	4.4	4.2
BCI	48.2	49.0	46.3	47.1	48.9	49.0
g241c	28.5	26.0	32.1	26.5	18.0	22.8
COIL	19.8	19.4	19.2	18.3	26.6	23.1
gc241n	21.8	16.5	24.6	17.4	9.2	14.3
text	25.1	22.9	25.6	23.0	25.4	22.8
MNIST	10.8	9.6	10.7	9.2	10.8	8.2

Figure 8: Various classification results, c_i determined by smoothing by K. The table is otherwise organized as in Figure 7.

6. Some Comments on the Benchmarks where Our Methods Do Not Work Well

If the class structure is trivial with respect to the geometry of the data as presented, then anisotropy will be unhelpful. This is the case for two of the benchmark data sets, g241c and g241n. In g241c, which has been constructed by generating two Gaussian clouds, and labeling each point by which cloud it came from, the best possible strategy (knowing the generative model) is to assign a point

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.9	2.9	2.6	2.6	2.4
USPS	4.9	4.1	5.0	3.8	4.2	4.1
BCI	45.9	47.1	44.9	46.0	47.4	48.7
g241c	31.5	25.3	34.2	26.7	23.1	23.7
COIL	14.3	13.0	13.4	12.0	16.5	16.6
gc241n	25.5	16.7	27.1	18.2	13.9	14.1
text	25.5	25.1	26.3	25.6	26.4	25.4
MNIST	9.4	7.4	9.0	6.9	9.4	7.9

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.0	2.7	2.1	2.6	2.3
USPS	5.2	3.8	5.2	3.6	4.0	3.4
BCI	47.6	48.1	45.0	46.9	48.2	48.5
g241c	30.7	23.8	33.3	24.7	21.7	21.6
COIL	17.2	17.5	16.0	15.4	22.0	21.5
gc241n	23.1	13.0	25.3	14.1	11.1	11.5
text	25.2	24.8	25.5	24.9	26.9	27.3
MNIST	10.0	7.8	10.1	7.3	9.7	7.4

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.5	2.8	2.2	2.6	2.2
USPS	5.0	4.1	5.2	3.5	3.9	3.2
BCI	48.2	47.5	45.9	45.7	47.6	47.9
g241c	30.5	23.1	32.8	24.1	21.2	21.2
COIL	18.0	17.5	16.2	16.1	22.8	22.1
gc241n	24.5	13.2	26.2	13.9	11.1	11.1
text	25.1	24.3	25.7	24.3	25.6	25.9
MNIST	10.3	8.1	10.0	7.5	9.6	8.6

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.7	2.9	2.5	2.0	2.2
USPS	5.6	4.6	5.6	4.1	4.4	3.6
BCI	48.2	49.0	46.3	47.4	48.9	49.7
g241c	28.5	19.8	32.1	21.5	18.0	18.0
COIL	19.8	19.8	19.2	18.8	26.7	25.8
gc241n	21.8	11.0	24.6	12.0	9.2	9.2
text	25.1	24.1	25.6	24.0	25.4	24.9
MNIST	10.8	8.9	10.7	7.9	10.8	9.4

Figure 9: Various classification results, c_i determined by smoothing by eigenfunctions of $\mathcal{L}$. The table is otherwise organized as in Figure 7.

	FAKS	**FAHC**	**FAEF**	**Best of other methods**
digit1	**2.0**	**2.1**	***1.9***	2.4 (Data-Dep. Reg.)
USPS	**4.0**	**3.9**	***3.3***	4.7 (LapRLS, Disc. Reg.)
BCI	45.5	45.3	47.8	**31.4** (LapRLS)
g241c	19.8	21.5	18.0	**13.5** (Cluster-Kernel)
COIL	12.0	11.1	15.1	**9.6** (Disc. Reg.)
gc241n	11.0	12.0	9.2	**5.0** (ClusterKernel)
text	***22.3***	***22.3***	**22.8**	23.6 (LapSVM)

Figure 10: Classification errors, in percent. In the rightmost column we chose, for each data set, the best performing method with model selection, among all those discussed in Chapelle et al. (2006). In each of the remaining columns we report the performance of each of the smoothing methods described above, but with the best settings of parameters for constructing the nearest neighbor graph and type of initial class guesses, among those considered in other tables (but all other smoothing parameters, including those for the initial guesses, cross validated). The aim of this rather unfair comparison is to highlight the potential of the methods on the different data sets.

to the cluster center it is nearest to. The boundary between the classes is exactly at the bottleneck between the two clusters; in other words, the geometry/metric of the data as initially presented leads to the optimal classifier, and thus modifying the geometry by the cluster guesses can only do harm. This is clearly visible if one looks at the eigenfunctions of the data set: the sign of the second eigenfunction at a given point is an excellent guess as to which cluster that point belongs to, and in fact in our experiments, often two was the optimal number of eigenfunctions. See figure 11. g241n

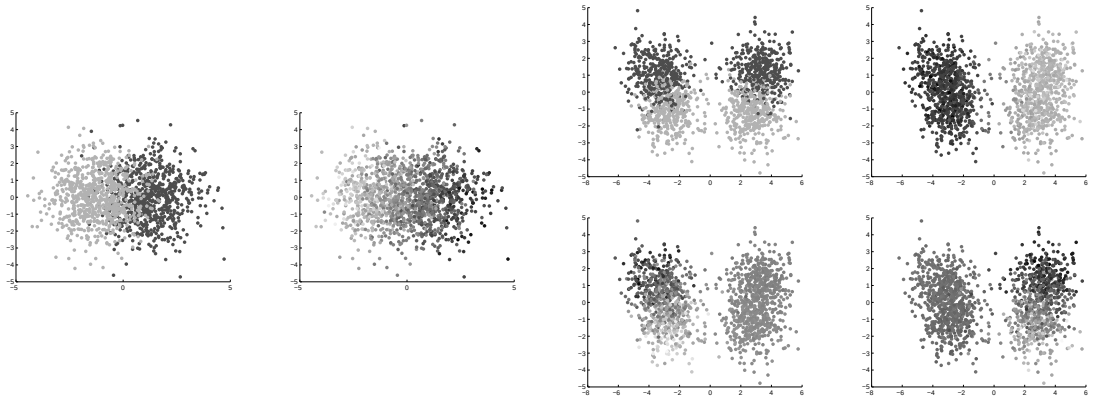

Figure 11: Panel on the left. On the left the lighter and darker points are the two classes for g241c. On the right is the second eigenfunction. Panel on the right. On the top left the lighter and darker points are the two classes for g241n. On the top right is the second eigenfunction, then on the bottom the third and fourth eigenfunctions.

is very similar; it is generated by four Gaussians. However, two pairs of centers are close together, and the pairs are relatively farther apart. The classes split across the two fine scale clusters in each coarse scale cluster as in g241c. In this data set, the ideal strategy is to decide which coarse cluster a point is in, and then the problem is exactly as above. In particular, the optimal strategy is given by the geometry of the data as presented. This is again reflected in the simplicity of the classes with respect to eigenfunctions 2, 3, and 4; see figure 11.

While in some sense these situations are very reasonable, it is our experience that in many natural problems the geometry of the data is not so simple with respect to the classes, and function-adapted kernels help build better classifiers.

Our method also was not useful for the BCI example. Here the problem was simply that the initial guess at the classes was too poor.

7. Computational Considerations

Let N be the cardinality of the data set X, which is endowed with some metric ρ. The first and most computationally intensive part of the algorithms proposed is the construction of the graph and corresponding weights. The approach we use is direct, in the sense that we explicitly store the similarity matrix W. For each point $x \in X$, we need to find the points in an ε-ball, or the k nearest neighbors of x. This problem can be solved trivially, for any metric ρ, in $O(dN^2)$ computations. It is of course highly desirable to reduce this cost, and this requires more efficient ways of computing near (or

nearest) neighbors. This problem is known to be hard even in Euclidean space $\mathbb{R}^d$, as d increases. The literature on the subject is vast, rather than a long list of papers, we point the interested reader to Datar et al. (2004) and references therein. The very short summary is that for approximate versions of the k-nearest neighbor problem, there exist algorithms which are subquadratic in N, and in fact pretty close to linear. The neighbor search is in fact the most expensive part of the algorithm: once for each point x we know its neighbors, we compute the similarities W (this is $O(k)$ for the k neighbors of each point), and create the $N \times N$ sparse matrix W (which contains kN non-zero entries). The computation of K from W is also trivial, requiring $O(N)$ with a very small constant. Apply K^t to a function f on X is very fast as well (for $t << N$, as is the case in the algorithm we propose), because of the sparsity of K, and takes $O(tkN)$ computations.

This should be compared with the $O(N^2)$ or $O(N^3)$ algorithms needed for other kernel methods, involving the computations of many eigenfunctions of the kernel, or of the Green's function $(I - K)^{-1}$.

Note that in most of the image denoising applications we have presented, because of the 2-d locality constraints we put on the neighbor searches, the number of operation is linear in the number N of pixels, with a rather small constant. In higher dimensions, for all of our examples, we use the nearest neighbor searcher provided in the TSTool package, available at `http://www.physik3.gwdg.de/tstool/`. The entire processing of an image as in the examples 256×256 takes about 7 seconds on a laptop with a 2.2Ghz dual core Intel processor (the code is not parallelized though, so it runs on one core only), and 2Gb of RAM (the memory used during processing is approximately 200Mb).

8. Future Work

We mention several directions for further study. The first one is to use a transductive learning approach to tackle image processing problems like denoising and inpainting. One has at one's disposal an endless supply of clean images to use as the "unlabeled data", and it seems that there is much to be gained by using the structure of this data.

The second one is to more closely mimic the function regularization in image processing in the context of transductive learning. In this paper, our diffusions regularize in big steps; also our method is linear (on a modified space). Even though there is no differential structure on our data sets, it seems that by using small time increments and using some sort of constrained nearest neighbor search so that we do not have to rebuild the whole graph after each matrix iteration, we can use truly nonlinear diffusions to regularize our class functions.

Another research direction is towards understanding how to construct and use efficiently basis functions which are associated to function-adapted diffusion kernels. The use of the low-frequency eigenfunctions of the operator, and the associated Fourier analysis of functions on the set has been considered in several works, as cited above, while the construction and use of multiscale basis functions, which correspond to a generalized wavelet analysis on data sets (Coifman and Maggioni, 2006; Szlam et al., 2005; Maggioni et al., 2005), has been used so far for approximation problems in machine learning (Maggioni and Mahadevan, 2006; Mahadevan and Maggioni, 2007) but has potential in many other applications. One can consider the approach that uses diffusion kernels directly, as in this paper, as a sort of "PDE approach" (even if in fact the discreteness and roughness of the sets considered usually brings us quite afar from PDEs on continua), while one can investigate "dual" approaches based on representations and bases functions.

9. Conclusions

We have introduced a general approach for associating graphs and diffusion processes to data sets and functions on such data sets. This framework is very flexible, and we have shown two particular applications, denoising of images and transductive learning, which traditionally are considered very different and have been tackled with very different techniques. We show that in fact they are very similar problems and results at least as good as the state-of-the-art can be obtained within the single framework of function-adapted diffusion kernels.

Acknowledgments

The authors would like to thank Francis Woolfe and Triet Le for helpful suggestions on how to improve the manuscript, and to James C. Bremer and Yoel Shkolnisky for developing code for some of the algorithms. MM is grateful for partial support by NSF DMS-0650413 and ONR N00014-07-1-0625 313-4224.

References

M. Belkin. *Problems of learning on manifolds*. PhD thesis, University of Chicago, 2003.

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. *Advances in NIPS*, 15, 2003a.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 585–591. MIT Press, Cambridge, 2001.

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

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(Invited Special Issue on Clustering):209–239, 2004. TR-2001-30, Univ. Chicago, CS Dept., 2001.

Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *COLT*, pages 486–500, 2005.

P. Bérard, G. Besson, and S. Gallot. Embedding Riemannian manifolds by their heat kernel. *Geom. and Fun. Anal.*, 4(4):374–398, 1994.

T. Boult, R.A. Melter, F. Skorina, and I. Stojmenovic. G-neighbors. *Proc. SPIE Conf. Vision Geom. II*, pages 96–109, 1993.

A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model. Simul.*, 4(2):490–530 (electronic), 2005a. ISSN 1540-3459.

A. Buades, B. Coll, and J. M. Morel. Denoising image sequences does not require motion estimation. *CMLA Preprint*, (12), 2005b.

T. F. Chan and J. Shen. *Image processing and analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. ISBN 0-89871-589-X. Variational, PDE, wavelet, and stochastic methods.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL http://www.kyb.tuebingen.mpg.de/ssl-book.

R.T. Chin and C.L. Yeh. Quantitative evaluation of some edge-preserving noise-smoothing techniques. *Computer Vision, Graphics, and Image Processing*, 23:67–91, 1983.

F. R. K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1997. ISBN 0-8218-0315-8.

R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. Technical report, Department of Statistics, 1995. URL citeseer.ist.psu.edu/coifman95translationinvariant.html.

R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, 102(21):7426–7431, 2005a. doi: 10.1073/pnas.0500334102. URL http://www.pnas.org/cgi/content/abstract/102/21/7426.

R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, 102(21):7432–7438, 2005b. doi: 10.1073/pnas.0500334102.

R.R. Coifman and S. Lafon. Diffusion maps. *Appl. Comp. Harm. Anal.*, 21(1):5–30, 2006a.

R.R. Coifman and S. Lafon. Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. *Appl. Comp. Harm. Anal.*, 21(1):31–52, 2006b.

R.R. Coifman and M. Maggioni. Multiscale data analysis with diffusion wavelets. *Proc. SIAM Bioinf. Workshop, Minneapolis*, April 2007. Tech. Rep. YALE/DCS/TR-1335, 2005.

R.R. Coifman and M. Maggioni. Diffusion wavelets. *Appl. Comp. Harm. Anal.*, 21(1):53–94, July 2006. (Tech. Rep. YALE/DCS/TR-1303, Yale Univ., Sep. 2004).

M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-885-7. doi: http://doi.acm.org/10.1145/997817.997857.

L.S. Davis and A. Rosenfeld. Noise cleaning by iterated local averaging. *IEEE Tran. on Systems, Man, and Cybernetics*, 8:705–710, 1978.

D. L. Donoho and C. Grimes. When does isomap recover natural parameterization of families of articulated images? Technical Report Tech. Rep. 2002-27, Department of Statistics, Stanford University, August 2002.

D. L Donoho and IM Johnstone. Ideal denoising in an orthonormal basis chosen from a library of bases. Technical report, Stanford University, 1994.

M. Elad. the origin of the bilateral filter and ways to improve it, 2002. URL citeseer.ist.psu.edu/elad02origin.html.

R.E. Graham. Snow-removal - a noise-stripping process for picture signals. *IRE Trans. on Inf. Th.*, 8:129–144, 1961.

L. Greengard and V. Rokhlin. *The rapid evaluation of potential fields in particle systems*. MIT Press, 1988.

Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *COLT*, pages 470–485, 2005.

T.S. Huang, G.J. Yang, and G.Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

P.W. Jones, M. Maggioni, and R. Schul. Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels. *Proc. Nat. Acad. Sci.*, 2007a. to appear.

P.W. Jones, M. Maggioni, and R. Schul. Universal local manifold parametrizations via heat kernels and eigenfunctions of the Laplacian. *submitted*, 2007b. http://arxiv.org/abs/0709.1975.

R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *J. ACM*, 51(3): 497–515 (electronic), 2004. ISSN 0004-5411.

J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, Jan 1984.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.

S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, Dept of Mathematics & Applied Mathematics, 2004.

Stephane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization. *To appear in IEEE Pattern Analysis and Machine Intelligence*, to appear, 2006.

J.S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2(2):165–168, 1980.

T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 644–652, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-962-9. doi: http://doi.acm.org/10.1145/276698.276880.

M. Maggioni and S. Mahadevan. Multiscale diffusion bases for policy iteration in markov decision processes. *submitted*, 2006. in preparation.

M. Maggioni and S. Mahadevan. Fast direct policy evaluation using multiscale analysis of markov diffusion processes. In *University of Massachusetts, Department of Computer Science Technical Report TR-2005-39; accepted at ICML 2006*, 2005.

M. Maggioni and H. Mhaskar. Diffusion polynomial frames on metric measure spaces. *ACHA*, 2007. in press.

M. Maggioni, J.C. Bremer Jr., R.R. Coifman, and A.D. Szlam. Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. volume 5914, page 59141M. SPIE, 2005. URL http://link.aip.org/link/?PSI/5914/59141M/1.

S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *University of Massachusetts, Department of Computer Science Technical Report TR-2005-38; Proc. NIPS 2005*, 2005.

S. Mahadevan and M. Maggioni. Proto-value functions: A spectral framework for solving markov decision processes. *JMLR*, 8:2169–2231, 2007.

S. Mahadevan, K. Ferguson, S. Osentoski, and M. Maggioni. Simultaneous learning of representation and control in continuous domains. In *AAAI*. AAAI Press, 2006.

G. Mahmoudi, M.; Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. *IEEE Signal Processing Letters*, 12(12):839–842, 2005.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001. URL citeseer.ist.psu.edu/ng01spectral.html.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

V.C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov. Fast computation of sums of gaussians in high dimensions. Technical Report CS-TR-4767, Department of Computer Science, University of Maryland, CollegePark, 2005.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992. ISSN 0167-2789. doi: http://dx.doi.org/10.1016/0167-2789(92)90242-F.

A. Shashua, R. Zass, and T. Hazan. Multiway clustering using supersymmetric nonnegative tensor factorization. Technical report, Hebrew University, Computer Science, Sep 2005.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Tran PAMI*, 22(8):888–905, 2000.

A. Singer. From graph to manifold Laplacian: the convergence rate. *Appl. Comp. Harm. Anal.*, 21 (1):128–134, July 2006.

R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35(2):876–879, 1964.

R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–349, 1967.

S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995. URL citeseer.ist.psu.edu/smith95susan.html.

A. Smola and R. Kondor. Kernels and regularization on graphs, 2003. URL citeseer.ist.psu.edu/smola03kernels.html.

G. W. Soules. The rate of convergence of sinkhorn balancing. *Linear Algebra and its Applications*, 150(3):3–38, 1991.

A.D. Szlam, M. Maggioni, R.R. Coifman, and J.C. Bremer Jr. Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions. volume 5914-1, page 59141D. SPIE, 2005. URL http://link.aip.org/link/?PSI/5914/59141D/1.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001. URL citeseer.ist.psu.edu/szummer02partially.html. http://www.ai.mit.edu/people/szummer/.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proc. IEEE Inter. Conf. Comp. Vis.*, 1998.

D. Tschumperle. *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Universite de Nice-Sophia Antipolis, 2002.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. Technical Report TR-134, Max Planck Insitute for Biological Cybernetics, 2004.

M. Wakin, D. Donoho, H. Choi, and R. Baraniuk. The Multiscale Structure of Non-Differentiable Image Manifolds. In *Optics & Photonics*, San Diego, CA, July 2005.

A. P. Witkin. Scale-space filtering. In *Proc. 8th int. Joint Conf. Art. Intell.*, pages 1019–1022, 1983. Karlsruhe, Germany.

L. P. Yaroslavsky. *Digital Picture Processing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985. ISBN 0387119345.

L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted median filters: a tutorial. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 43(3):155–192, 1996.

R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *International Conference on Computer Vision (ICCV)*, Oct 2005.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *Eighteenth Annual Conference on Neural Information Processing Systems, (NIPS)*, 2004.

H. Zha, C. Ding, M. Gu, X. He, and H.D. Simon. Spectral relaxation for *k*-means clustering. In *NIPS 2001*, pages 1057–1064. MIT Press, Cambridge, 2001.

D. Zhou and B. Schlkopf. Regularization on discrete spaces. pages 361–368, Berlin, Germany, 08 2005. Springer.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions, 2003. URL citeseer.ist.psu.edu/zhu03semisupervised.html.