Semi-supervised Learning Based on Joint Diffusion of Graph Functions and Laplacians

Kwang In Kim(✉)

Department of Computer Science, University of Bath, Bath, UK
k.kim@bath.ac.uk

Abstract. We observe the distances between estimated function outputs on data points to create an anisotropic graph Laplacian which, through an iterative process, can itself be regularized. Our algorithm is instantiated as a discrete regularizer on a graph's *diffusivity operator*. This idea is grounded in the theory that regularizing the diffusivity operator corresponds to regularizing the metric on Riemannian manifolds, which further corresponds to regularizing the anisotropic Laplace-Beltrami operator. We show that our discrete regularization framework is consistent in the sense that it converges to (continuous) regularization on underlying data generating manifolds. In semi-supervised learning experiments, across ten standard datasets, our diffusion of Laplacian approach has the lowest average error rate of eight different established and state-of-the-art approaches, which shows the promise of our approach.

Keywords: Semi-supervised learning · Graph Laplacian · Diffusion · Regularization

1 Introduction

In semi-supervised learning, we discover a function f which, from a set of data points and partial labels, propagates the labels to an unlabeled subset of data. To achieve this, methods exploit the underlying 'geometry' of or 'relationships' between points in the input space to help learn f. Many times, the underlying relationships between data points can be represented as a graph, or as a set of data points sampled from an underlying manifold. In these cases, the graph Laplacian is useful as it contains the pairwise relations or similarity w_{ij} between data points in the unlabeled space.

Hence, many existing algorithms use the graph Laplacian as a regularizer to directly enforce smoothness over the estimated function f: When a pair of data nodes or points $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, we can cause their corresponding function evaluations $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ to also be similar by directly trading a training error against a regularization cost. Alternatively, other algorithms use simulations of physical diffusion processes to indirectly promote smoothness on f: At each point $\mathbf{x}$, the evaluated function value $f(\mathbf{x})$ is spread over the neighborhood of $\mathbf{x}$. The strength of this diffusion—the *diffusivity*—between two points $\mathbf{x}_i$ and $\mathbf{x}_j$, is determined by their similarity w_{ij}.

For semi-supervised learning, it has been shown that these two approaches are actually equivalent (e.g. [1]): The solution of direct regularization-based approaches that trade training error and regularization cost is also the solution at the limit case of the diffusion process over time. Thus, the graph Laplacian is both the regularizer and the generator of the diffusion process. As such, we will be using the term 'regularization' interchangeably with 'diffusion'.

In general, due to the ill-posed nature of semi-supervised learning, the estimated f evaluations are noisy, and so the constructed new anisotropic graph Laplacian is also noisy. As such, in this paper, we propose to regard the graph Laplacian itself as an object which can be regularized. This is similar to the tradition in learning algorithms where the f estimate is regularized to prevent over-fitting to noisy data.

Our approach builds upon the analysis of learning over continuous spaces, as is commonly assumed in geometry-based semi-supervised algorithms: As the number of sampled data points increases to infinity, the corresponding graph converges to the underlying manifold M which generated those data samples. In this case, the graph Laplacian converges to the Laplace-Beltrami operator on M. Roughly, applying the Laplace-Beltrami operator to a function f measures the first-order variation of f. This operator lets us regularize f by penalizing the first-order f variation.

Our idea of regularizing the graph Laplacian itself extends in the continuous space to regularizing the Laplace-Beltrami operator itself: We measure the variation of the structure that measures the variation of f. As we will show, this structure is prescribed by the metric on M, rendering our framework into metric regularization. In general, regularizing the metric on M is a difficult problem when the manifold M is only observed through sparse data points, as is typical in practical applications. Using recent results on the equivalence of the metric on M and the continuous *diffusivity* operator [2], we develop a regularization framework that enforces the smoothness of the diffusivity operator as a surrogate. Then, discretizing our continuous space formulation into a finite graph G, we construct an efficient regularization framework for the graph Laplacian.

We show that this discrete regularization framework on G converges to the continuous Laplacian regularization on M based on convergence analysis of the graph Laplacian [3,4] and the equivalence of inducing an anisotropic diffusion process with a new metric on M [2]. In experiments, we demonstrate that the resulting algorithms significantly improves upon existing linear/non-linear, and isotropic/anisotropic diffusion-based semi-supervised learning algorithms, as well as other state-of-the-art algorithms.

Related work. Anisotropic diffusion has been known to be particularly effective in processing two-dimensional images [5] and surfaces [6]. Szlam *et al.* [7] extended these algorithms to high-dimensional graph structured data as discrete approximations of smooth manifolds. The resulting semi-supervised learning and denoising algorithms demonstrated significant improvement over existing isotropic diffusion algorithms and isotropic graph Laplacian-based regularization algorithms. Recently Kim *et al.* [2] further extended and improved upon this framework for non-linear diffusion. Our approach extends both the linear diffusion algorithm of Szlam *et al.* [7] and the non-linear diffusion algorithm of Kim *et al.* [2] by introducing the Laplacian as a new object to be diffused/regularized in addition to the classification function f being diffused. As discussed previously and shown in Sect. 2, regularizing the Laplacian is equivalent to regularizing the metric on manifolds, which we take to instantiate our practical algorithms. From this perspective, our approach can be regarded as an instance of tensor regularization [8–10]. Existing tensor regularization approaches rely on known manifold structure (e.g. images or surfaces) [8,9] or they are specialized on specific graph connectivities (e.g. edges focus on orthogonal transform between node data [10]). In contrast, our algorithm is applicable to any graph structured data without having to access the underlying data generating manifold or to introduce restricting assumptions. Accordingly, one of our main contribution is extending existing tensor regularization approaches to graph-based semi-supervised learning.

Our algorithm refines the original graph Laplacian throughout the diffusion process. A single fixed instance of the graph Laplacian can be regarded as a pseudo-inverse of a similarity kernel matrix (Sects. 2 and 4, and [11,12]). In this sense, our algorithm can also be regarded as an instance of spectral kernel design where the kernel matrix (inverse of Laplacian in semi-supervised learning context) is automatically constructed based on the spectral analysis of the kernel matrix itself [11,12]. Section 4 discusses this comparison, and Sect. 5 presents experimental comparisons [11,12]. This perspective also establishes a connection between our approach to graph denoising and link prediction algorithms. In particular, we show that our linear diffusion algorithm corresponds to an iterative solver of existing graph denoising [13] and ensemble ranking [14] algorithms (Sect. 4). In experiments, we demonstrate that adopting our approaches in a linear diffusion algorithm improves semi-supervised learning performance and, furthermore, non-linearly extending them could lead to even further performance improvement.

Overview. To explain our approach, we will first introduce regularization of the Laplace-Beltrami operator on continuous manifolds and show that this is equivalent to regularizing a Riemannian metric on M (Sect. 2). However, regularizing a metric on M is not straightforward, so we show that the goal of regularizing a metric on M (or, equivalently, the Laplace-Beltrami operator) can be achieved by regularizing the diffusivity operator on M as a surrogate in the context of anisotropic diffusion (Sect. 3). As such, we introduce an anisotropic diffusion process on M, and also show that by discretizing this process to sampled data we can achieve a practical metric regularization algorithm which still converges to the solution of the continuous case. Finally, with this, we present a new semi-supervised learning algorithm that jointly diffuses classification functions and the diffusivity operator (Sect. 4).

2 Smoothness of the Laplace-Beltrami Operator

We begin by formally defining regularization of the Laplace-Beltrami operator on continuous manifolds, and show its equivalence to metric regularization.

The Laplace-Beltrami operator (or shortly, Laplacian) $\Delta^g : C^\infty(M) \to C^\infty(M)$ on a Riemannian manifold (M, g) with a metric g is a second order differential operator:

$$\Delta^g f = \nabla^{g*} \nabla^g f, \tag{1}$$

where $\nabla^g : C^\infty(M) \to TM$ and $\nabla^{g*} : TM \to C^\infty(M)$ are the gradient and divergence operators, respectively and TM is the tangent bundle of M.

The typical way to regularize a linear operator (such as Laplacian) is to minimize its operator norm, which corresponds to the largest operator eigenvalue. This finds numerous applications including storage allocation [15] and system stability analysis [16]. Understanding the significance of this approach to semi-supervised learning is not straightforward as, for semi-supervised learning, Zhang and Ando [11] actually suggest exactly the opposite: It has been demonstrated that *maximizing* the largest eigenvalues of the Laplacian corresponds to minimizing the upper bound on the generalization error. More precisely, Zhang and Ando proposed decreasing the smallest eigenvalues of the *kernel* matrix K with the pseudo-inverse of the Laplacian L^+ being a special case of K.

Instead, our regularization approach is based on the analysis of the *spatial* behavior of the Laplacian. First, we will explain our method, but interestingly, we will show that the approach of Zhang and Ando [11] leads to a construction which is similar to our linear diffusion framework Sect. 4.

The Laplacian on the Euclidean space $\mathbb{R}^m$ can be written as [3]:

$$[\Delta^{\mathbb{R}^m} f](x) := \sum_{i=1,\ldots,m} \frac{\partial^2}{(\partial x^i)^2} f(x) = \lim_{r \to 0} \frac{1}{C_m r^2} \left([A_B(x) f] - f(x) \right), \tag{2}$$

where

$$[A_B(x) f] = \frac{1}{\mathrm{vol}(B(x,r))} \int_{B(x,r)} f(y) dy, \tag{3}$$

C_m is a constant depending only on m, $B(x,r)$ is a ball of radius r centered at x, and $\mathrm{vol}(B)$ is the volume of B. The equality in Eq. 2 implies that roughly, the Laplacian $\Delta^{\mathbb{R}^m} f$ of a function f evaluated at x measures the deviation of $f(x)$ from its average $[A_B(x) f]$ taken at a local neighborhood. This characterization suggests that the behavior of the Laplacian is determined by the local averaging operator $A_B(x)$. The underlying idea of our approach is to enforce the spatial smoothness of the corresponding averaging operator on M.

As clearly seen from the integral form (Eq. 3), the operator $A_B(x)$ is spatially homogeneous and therefore, its spatial variation is simply zero: Equivalently, the Laplacian is homogeneous in $\mathbb{R}^m$. Now to enforce our underlying idea of spatial smoothness to manifolds, we adopt a generalization of this Laplacian representation to manifolds as proposed by Hein [17], and Coifman and Lafon [18]. Suppose that (M, g) is an n-dimensional submanifold of $\mathbb{R}^m$ with $i : M \to \mathbb{R}^m$ being the corresponding embedding. The local averaging-based Laplacian estimate $\Delta^g_h f$ on (M, g) is defined as

$$[\Delta^g_h f](x) = \frac{1}{h^2}\left(f(x) - \frac{1}{d_h(x)}[A^g_h(x)f]\right), \text{ where } [A^g_h(x)f] = \int_M k_h(x,y)f(y)dV(x), \tag{4}$$

$d_h(x) = [A^g_h(x)\mathbf{1}]$ with $\mathbf{1}$ being a constant function of ones, and dV is the natural volume element of M ($dV(x) = \sqrt{|\det(\mathbf{g})|}dx$ with $\mathbf{g}$ being the coordinate matrix of g). The kernel k_h is defined based on a Gaussian function on $\mathbb{R}^m$:

$$k_h(x,y) = \begin{cases} \frac{1}{h^m}k(\|i(x)-i(y)\|^2_{\mathbb{R}^m}, h^2) & \text{if } \|i(x)-i(y)\|_{\mathbb{R}^m} \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

with $k(a,b) = \exp(-a/b)$.

It has been shown that when M is compact, Δ^g_h converges uniformly to the Laplacian Δ^g [17] as $h \to 0$.[1] From this convergence result and the definition of the average operator A^g_h (Eq. 4), we can see that the spatial variation of Δ^g is entirely determined by the metric g. This is not surprising as the Laplacian itself is indeed, expressed as a function of the metric g: Writing Eq. 1 in local coordinates $\{x^1, \dots, x^n\}$ [19],

$$\begin{aligned}\Delta^g f &= \sum_{i,j=1,\dots,n} -\frac{1}{\sqrt{|\det(\mathbf{g})|}}\partial_j\left([\mathbf{g}^{-1}]_{ij}\sqrt{|\det(\mathbf{g})|}\partial_i f\right) \\ &= \sum_{i,j=1,\dots,n} -[\mathbf{g}^{-1}]_{ij}\partial_j\partial_i f + (\text{lower order terms}),\end{aligned} \tag{6}$$

where $\partial_i := [\partial/\partial_{x^i}]$. Furthermore, as the second equality in Eq. 6 shows, by evaluating Δ^g on a function $x^i x^j$, we can reconstruct the metric $\mathbf{g}$. Therefore, defining a new regularization operator $\Delta^{\overline{g}}$ is equivalent to determining a new metric $\overline{g}$. Unfortunately, this equivalence by itself does not make the regularization of Laplacian easier since in general, regularizing a Riemannian metric is a non-trivial problem.[2] However, in the next section we demonstrate that applying this idea to induce an anisotropic diffusion process on a finite graph G that approximates M, regularizing the metric (and equivalently the Laplacian) can be easily instantiated by enforcing the smoothness of the *local diffusivity operator* on G.

3 Anisotropic Diffusion on Graphs and Manifolds

Having defined the Laplacian regularization and established its equivalence to metric regularization, we now present continuous anisotropic diffusion on a

[1] Here, we assume for simplicity that the underlying probability distribution p is uniform and M is compact. In general, any compactly supported distribution p can be adopted instead. See [3] for details.

[2] Defining a regularizer on a metric g is not straightforward since, by the construction of the Riemannian connection, $\nabla^g_X g = 0$ for all vector fields X on M. Furthermore, even when the connection (defined on g) is made independent of a new metric $\overline{g}$, evaluating the derivative of a second-order tensor ($\overline{g}$) from finite data points X is a difficult problem in general.

Riemannian manifold (M, g), and its discretization on graphs. We follow the construction of Weickert [5] who applied an anisotropic diffusion process to images as functions on $\mathbb{R}^2$, which has been subsequently extended to functions on manifolds [2,7].

Anisotropic diffusion of a smooth function $f \in C^\infty(M)$ based on a positive definite *diffusivity operator* $D : TM \to TM$ is described as a partial differential equation:

$$\frac{\partial f}{\partial t} = \nabla^{g*} D \nabla^g f. \tag{7}$$

At each point $x \in M$, D is represented (in coordinates) as a positive definite matrix $\mathbf{D}(x)$ which controls the strength and direction of diffusion. If D is an identity operator, Eq. 7 becomes isotropic diffusion. An important feature of the diffusivity operator D is that it uniquely defines a new metric on M [2]: Anisotropic diffusion on (M, g) generated by the operator $\Delta^D := -\nabla^{g*} D \nabla^g$ is equivalent to an isotropic diffusion on a new manifold $(M, \overline{g})$ with an updated metric $\overline{g}$ which is given in coordinates as

$$\overline{\mathbf{g}}(x) = c(x)\mathbf{g}(x)\mathbf{D}^{-1}(x), \tag{8}$$

where $c \in C^\infty(M)$, and $\mathbf{g}(x)$ and $\overline{\mathbf{g}}(x)$ are the coordinate representations (matrices) of g and $\overline{g}$ at x, respectively. In particular, in Riemannian normal coordinates at x (based on g), g becomes Euclidean (i.e. $[\mathbf{g}]_{ij}(x) = \delta_{ij}(x)$), and therefore, $\mathbf{D}(x)$ can be interpreted as a covariance matrix of a Mahalanobis distance. We will henceforth refer Δ^D as an *anisotropic Laplacian*. This operator will be used to analyze the limit case behavior of the discretization of $\Delta^{\overline{g}}$ (see Proposition 1).

In practical applications, the manifold M is not directly observed and instead, a sampled point cloud X is provided ($X \subset i(M) \subset \mathbb{R}^m$). The remainder of this section focuses on a graph Laplacian-based discretization of Eq. 7.

A weighted graph $G = (X, E, W)$ consists of nodes $X = \{\mathbf{x}_1, ..., \mathbf{x}_u\}$, edges $E \subset X \times X$, and weights assigned for each edge in E:

$$w_{ij} = w(e_{ij}) := k_h(i^{-1}(\mathbf{x}_i), i^{-1}(\mathbf{x}_j)), e_{ij} \in E, w_{ij} \in W. \tag{9}$$

The *graph gradient* operator ∇^G is defined as the collection of node function differences along the edges:

$$[\nabla^G f](e_{ij}) = \sqrt{w_{ij}}(f(\mathbf{x}_j) - f(\mathbf{x}_i)). \tag{10}$$

while the *graph divergence* operator ∇^{G*} measures the variation of functions $\{S : E \to \mathbb{R}\}$ on edges:

$$[\nabla^{G*} S](\mathbf{x}_i) = \frac{1}{2d_i} \sum_{j=1,\dots,u} \sqrt{w_{ij}}(S(e_{ji}) - S(e_{ij})), \tag{11}$$

where $d_i = \sum_{j=1}^u w_{ij}$. Given the two operators, the (isotropic) graph Laplacian L is defined as

$$[Lf](\mathbf{x}_i) = [\nabla^{G*} \nabla^G f](\mathbf{x}_i). \tag{12}$$

Similarly to the case of continuous anisotropic diffusion (Eq. 7), an anisotropic diffusion process on G can be introduced by *placing* a positive definite diffusivity operator in-between the gradient and divergence operators: The anisotropic graph Laplacian L^D and the corresponding diffusion process are defined as

$$[L^D f](\mathbf{x}_i) := [\nabla^{G^*} D \nabla^G f](\mathbf{x}_i), \frac{\partial}{\partial t} f = [L^D f], \tag{13}$$

where the *graph diffusivity operator* D is represented as

$$[DS](e_{ij}) = q_{ij} \mathbf{b}_{ij} \langle \mathbf{b}_{ij}, S \rangle \tag{14}$$

with the basis function $\mathbf{b}_{ij}$ being the indicator of e_{ij} and the inner-product $\langle S, T\rangle$ of edge functions S and T defined as $\langle S, T\rangle = \sum_{i,j=1}^{u} S(e_{ij})T(e_{ij})$. It should be noted that by substituting Eq. 14 into Eq. 13, L^D can be constructed by replacing w_{ij} in ∇^G and ∇^{G^*} (Eqs. 10 and 11) with $\overline{w}_{ij} := w_{ij}q_{ij}$ and combining the resulting two operators as in Eq. 12.

As discussed earlier, if X is sampled from (M, g), the graph Laplacian L converges to the Laplace-Beltrami operator Δ^g as $u \to \infty$ [4,20]. Now we generalize this result to anisotropic graph Laplacian:

Proposition 1 (The convergence of L^D *to* $\Delta^{\overline{g}}$). *Assume that (M, g) is an n-dimensional Riemannian submanifold of $\mathbb{R}^m$ and $(M, \overline{g})$ is a new manifold with an updated metric $\overline{g}$. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_u\}$ be a sample from a compactly supported, uniform distribution p on M and the coefficients $\{q_{ij}\}$ of the graph diffusivity operator D are given as*

$$h^m q_{ij} = k(\|\mathbf{x}_i - \mathbf{x}_j\|^2_{\mathbb{R}^m}, h^2) + k(-\|i^{-1}(\mathbf{x}_i) - i^{-1}(\mathbf{x}_j)\|_{\overline{g}^2}, h^2). \tag{15}$$

Then, L^D converges to $\Delta^{\overline{g}}$ almost surely as $u \to \infty$, $h \to 0$, and $uh^{m+2}/\log(u) \to \infty$.

Sketch of proof: Equation 15 essentially equates the pairwise distance $\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_D$ induced by D at points $\mathbf{x}_i$ and the corresponding distance measured in the resulting metric $\overline{g}$: We represent $\overline{w}_{ij} (= w_{ij}q_{ij})$ as the Gaussian envelop of the new distance $\|\mathbf{x}_j - \mathbf{x}_i\|_{\overline{g}}$: ($\overline{w}_{ij} = \frac{1}{h^m} k\left(\|\mathbf{x}_i - \mathbf{x}_j\|^2_{\overline{g}}, h^2\right)$). The convergence proof is then completed by (1) the equivalence of anisotropic Laplace-Beltrami operator on (M, g) and the isotropic Laplace-Beltrami operator on $(M, \overline{g})$ and (2) the convergence of the isotropic graph Laplacian to Laplace-Beltrami operator (see *Main Result* of [3]). □

The conditions $h \to 0$ and $uh^{m+2}/\log(u) \to \infty$ in the proposition are required for the convergence of isotropic Laplacian [3]: This implies that the shrink speed of kernel width parameter h has to be controlled in the sense that any ball-shaped region of diameter h contains sufficiently many data points.

This result establishes a connection between the problems of estimating an anisotropic Laplacian L^D on a graph G and a metric $\overline{g}$ on a Riemannian manifold (M, g), and it provides motivation on regularizing the graph diffusivity operator D as a surrogate to $\overline{g}$ in practical estimation problems in general. However, Proposition 1 cannot be used directly in analyzing the behavior our anisotropic diffusion algorithms since, as we shown shortly in Sect. 4, our goal is to *dynamically construct* a new metric depending on the given dataset X and the corresponding labels for a subset of X, rather than restoring a hidden, fixed metric $\overline{g}$. As such, the next section presents the development of a new metric based on the variations of f evaluations.

4 Semi-supervised Learning Based on Non-linear Anisotropic Diffusion on Graphs

Our semi-supervised learning algorithm simultaneously evolves the node-function f as well as the regularizer L^D based on anisotropic diffusion. First, we discuss traditional f-diffusion, and extend it to the diffusion of regularizer L^D. Second, we combine the two diffusion processes (of f and L^D).

Diffusion of f. Suppose that for the first l data points in X, the corresponding labels $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\} \subset \mathbb{R}^c$ are provided where c is the number of classes. Assuming that the class label of the i-th data point is q, $\mathbf{y}_i$ is defined as a row vector of size $1 \times n$ where it has zero everywhere except for the q-th location in which its value is 1. If the regularizer (or diffusion generator) L^D is held fixed, the corresponding (time-discretized) linear diffusion process can be stated as

$$\frac{\mathbf{f}(t+\delta) - \mathbf{f}(t)}{\delta} = -L^D \mathbf{f}(t), \text{ where } \mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_u)]^\top. \tag{16}$$

If D is an identity operator, Eq. 16 becomes isotropic diffusion. Otherwise, it is anisotropic. For instance, Szlam *et al.* [7] proposed tuning the diffusivity based on function evaluations $\mathbf{f}$: On a graph G the diffusivity along the edge e_{ij} becomes stronger or weaker as $\mathbf{f}_i$ and $\mathbf{f}_j$ are similar or different, respectively. This idea can be implemented by directly controlling the diffusivity operator D (Eq. 14):

$$q_{ij} = \exp\left(-\frac{\|\mathbf{f}_i(t_0) - \mathbf{f}_j(t_0)\|^2_{\mathbb{R}^c}}{h'}\right), \tag{17}$$

with a positive number h' being a hyper-parameter. For anisotropic diffusion, Szlam *et al.* [7] suggested initializing $\mathbf{f}(t_0)$ by running few steps of isotropic diffusion. This has significantly improved the convergence speed in our preliminary experiments. For all anisotropic diffusion algorithms compared in the experiments, we initialized the solutions by executing 20 iterations of isotropic diffusion.

Recently, Kim *et al.* [2] have demonstrated that extending the linear diffusion process in Eq. 16 to non-linear one can significantly improve the accuracy of the resulting semi-supervised learning algorithm:

$$\mathbf{f}(t+\delta) = (I - \delta L^D(t))\mathbf{f}(t). \tag{18}$$

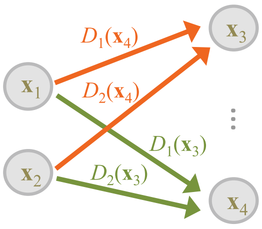

Fig. 1. We regularize the Laplacian on manifolds by enforcing the smoothness of the corresponding diffusivity operator. Instantiating this for graphs, our algorithm enforces the smoothness of the graph diffusivity functions $\{D_1,\ldots,D_u\}$: If $\mathbf{x}_1$ and $\mathbf{x}_2$ are close, the corresponding diffusivity functions $D_1(\cdot)$ and $D_2(\cdot)$ should be *similar*, e.g., $D_1(\mathbf{x}_3) \sim D_2(\mathbf{x}_3)$ and $D_1(\mathbf{x}_4) \sim D_2(\mathbf{x}_4)$.

This process is identical to Eq. 16 except that L^D now depends on t based on Eq. 17.

Diffusion of L^D. Now we apply the diffusion process developed in the previous paragraph to evolve the diffusivity operator D. To facilitate this process, we cast D that originally maps from the spaces of edge functions to itself, to a set of functions on X (Fig. 1): We take inner products of D with basis functions $\{\mathbf{b}_{ij} \otimes \mathbf{b}'_{ij}\}$ (with $\otimes$ and $\mathbf{b}'_{ij}$ being a tensor product and a dual vector of $\mathbf{b}_{ij}$, respectively):

$$D \Rightarrow \{D_1,\ldots,D_u\} \text{ with } D_i(\mathbf{x}_j){:=}\langle \mathbf{b}_{ij}, [D\mathbf{b}_{ij}]\rangle. \tag{19}$$

Now we can construct a sparse matrix $\mathbf{M} \in \mathbb{R}^{u\times u}$ by combining column-wise the elements of $\{D_i\}$, i.e. $\mathbf{M}_{[:,i]} = [D_i(\mathbf{x}_1),\ldots,D_i(\mathbf{x}_u)]^\top$.[3] Here, the sparsity is induced by adopting the h-neighborhood (Eq. 5) or k-nearest neighborhood (NN) used in constructing the isotropic weight matrix $\{w_{ij}\}$. In the experiments, we use k-NN and regard k as a hyper-parameter. The convergence results of Proposition 1 can be easily modified to k-nearest neighborhood case (e.g. see Sect. 2.4 of [17]).

Using $\mathbf{M}$ as a surrogate, the diffusion of L^D and its time discretization, are stated respectively as:

$$\frac{\partial \mathbf{M}}{\partial t} = -L^D\mathbf{M}, \mathbf{M}(t+\delta) = (I - \delta L^D(t))\mathbf{M}(t). \tag{20}$$

Joint diffusion of f and L^D. The two diffusion processes for $\mathbf{f}$ (Eq. 18) and for L^D (Eq. 20), respectively are both governed by L^D. At the same time, they constantly update L^D. We propose interweaving the two processes: We start with the linear diffusion process of $\mathbf{f}$ (i.e. L^D is fixed at the L). Then, at each N-th iteration of the diffusion, L^D is updated based on Eq. 17 and subsequently by the diffusion process on L^D (Eq. 20). Here we fix N at 20 which provides a moderate tradeoff between the flexibility (non-linearity) of L^D and the computational

[3] We adopt Matlab notation where $A_{[:,i]}$ represents the i-th column of matrix A.

Algorithm 1. *AND*: Semi-supervised learning using combined diffusion of function and Laplacian.

```
Input: Data points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_u\} \subset \mathbb{R}^d$; labels $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\} \subset \mathbb{R}^c$;
       hyper-parameters $T_1$, $T_2$, $h'$, and $k$ (see Sect. 5)
Output: Diffused labels $\mathbf{f}$.

Build an isotropic graph Laplacian $L$ based on Eq. 9
for $t = 1, \ldots, T_1$ do
    Update $\mathbf{f}(t)$ based on Eq. 18.
    At each 20-th iteration Update $L^D(t)$ based on Eq. 17;
    for $t' = t, \ldots, t + T_2$ do
        Update $L^D(t')$ based on Eq. 20;
    end
    Assign labels to $\mathbf{f}^t$ and $L^D(t)$;
    Normalize $L^D(t)$;
end
```

complexity. The numbers of diffusion iterations T_1 and T_2 for $\mathbf{f}$ (Eq. 18) and $\mathbf{M}$ (Eq. 20) are taken as hyper-parameters. At each iteration, the Laplacian matrix L^D is normalized (we use *Random-walk* normalization [3]) and parts of $\mathbf{f}$ and $\mathbf{D}$ are updated based on the provided labels: If the i-th data point is labeled, the i-th row of $\mathbf{f}(t+1)$ is replaced by the corresponding label. Also, if the i-th and j-th data points are labeled, $\mathbf{D}_{ij}$ and $\mathbf{D}_{ji}$ are assigned with 1 (if the class labels of i-th and j-th data points are the same) or 0. Algorithm 1 summarizes the proposed joint diffusion process.

Time complexity. Overall, the time complexity of our algorithm depends on the number u of data points and the size k of the nearest neighborhood. Each $\mathbf{f}$-diffusion iteration requires multiplying the matrix L^D with a vector $\mathbf{f}$ while $\mathbf{M}$-diffusion iteration requires the product of L^D and $\mathbf{M}$. Due to the sparsity of these matrices, this operation can be performed even for relatively large datasets. For the *MNIST* dataset with 60,000 data points, running 100 iterations of the joint diffusion process takes around 30 s in MATLAB on a 3.6 GHz CPU.

Relations to existing work. The relationship between our algorithm and related existing works presented in Sect. 1 is interesting. Equation 20 is non-linear as the diffusion generator Δ^D depends on the object $\mathbf{M}$ being diffused. If we *linearize* Eq. 20 at t_0, i.e. approximating the generator $\Delta^D(t)$ by the fixed time instance $\Delta^D(t_0)$, we can recover a closed form regularization solution (see Zhou *et al.* [1]):

$$L^{D*} \Leftarrow \mathbf{M}^* = \operatorname*{arg\,min}_{\mathbf{M} \in \mathbb{R}^{u \times u}} \|\mathbf{M} - \mathbf{M}(t_0)\|_F^2 + \frac{\delta}{1-\delta} \mathrm{tr}[\mathbf{M}^\top L^D(t_0)\mathbf{M}], \tag{21}$$

where $\|A\|_F$ and $tr[A]$ are the Frobenius norm and the trace of matrix A, respectively. This type of regularization has been used in few applications including graph denoising and link prediction [13], and ensemble ranking [14]. In addition, this perspective facilitates comparison of our framework with Zhang and Ando's approach [11] where the inverse of the graph Laplacian is regarded as an instance of the (similarity) kernel matrix K. Their analysis on the effect of the largest eigenvalues of L on the expected error bound led to an algorithm that takes powers of the Laplacian: L^p with $p \in \mathbb{Z}^+$ is used as the regularizer instead of L. If we take the graph Laplacian L as the object to be diffused in Eq. 20 instead of the anisotropic diffusivity operator $\mathbf{M}$,[4] the resulting diffusion process is equivalent to regularizing the original weight matrix $\{w_{ij}\}$. Now assigning a specific value 0.5 to δ, and linearizing the resulting diffusion equation, we obtain[5]

$$L(t+\delta) = \delta L^2(t). \tag{22}$$

Therefore, Zhang and Ando's algorithm can be regarded as taking a single step of the linearized diffusion on the original isotropic regularizer L before $\mathbf{f}$ is subsequently optimized. Interestingly, taking powers of the Laplacian can also be regarded as constructing high-order regularizers. Extending the convergence result of the graph Laplacian to Laplace-Beltrami operator on manifolds, we obtain the analogy between high-order regularization and iterated Laplacian. When $f \in C^\infty(M)$ and $u \to \infty$ [21],

$$\begin{aligned} -\frac{1}{u} C_n \mathbf{f}^\top L^p \mathbf{f} &\to \int_M f(x)[(\Delta^g)^p f](x) dV(x) \\ &= \begin{cases} \int_M \left((\Delta^g)^{\frac{p}{2}} f(x)\right)^2 dV(x) & \text{if } p \text{ is even} \\ \int_M \left\| [\nabla^g (\Delta^g)^{\frac{p-1}{2}} f](x) \right\|_g^2 dV(x) & \text{otherwise,} \end{cases} \end{aligned} \tag{23}$$

where C_n is a constant depending on the dimensionality n and the equality is the result of Stokes' theorem. Therefore, $\mathbf{f}^\top L^p \mathbf{f}$ approximates high-order variations of f.[6] Zhou and Belkin [21] demonstrated that using high-order regularizers can significantly improve the semi-supervised learning performance as they help avoiding the degenerate case where the function obtained by minimizing combinations of the training error and the regularization cost is everywhere zero except for the labeled data points. Therefore, the linearization of our diffusion framework can be regarded as adopting high-order regularizers. In particular, the linearized analytical solution in Eq. 21 can be regarded as a variation of adopting an infinite-order variation measure, as the solution corresponds to the limit case of Eq. 20.

Unfortunately, more rigorous comparison between our approach and that of Zhang and Ando's algorithm [11] and equivalently of Zhou and Belkin [21] is not straightforward since (1) in our diffusion algorithms we do not iterate the diffusion process infinite times (i.e. $t \to \infty$) but terminate it at a given time step

[4] Essentially L^D and $\mathbf{M}$ contain the same information.

[5] When in general, $\delta \neq 0.5$, iterating p-times, the linearized diffusion generates an operator polynomial. This corresponds to using a (selected) combination of differential operators up to $\lceil p/2 \rceil$-th order.

[6] For simplicity, here we regard that f is a one-dimensional function.

T_1 which is an hyper-parameter taking a similar role as δ in Eq. 21; (2) Furthermore, our diffusion is non-linear where the regularizer itself is dynamically evolved. In this case, the solution cannot be straightforwardly represented in closed-form. In the experiments, we demonstrate that this non-linear diffusion of L^D leads to a significant improvement over the linearized diffusion case and equivalently, over the analytic solution of Eq. 21. Also, an important advantage of the diffusion formulation (Eq. 20) over the analytical solution (Eq. 21) is that, diffusion formulation facilitates a sparse framework: At time 0, the sparsity of $\mathbf{M}$ is controlled by the k-nearest neighbor size. As the diffusion progresses, the matrix $\mathbf{M}$ tends to be denser. In this case, the solution $\mathbf{M}$ can be directly sparsified by assigning selected elements of $\mathbf{M}$ with zero. In our experiments, we sparsified $\mathbf{M}$ by replacing it with $\mathbf{M} \circ (I^w)^2$ where $\circ$ is the Hadamard product and $I^w_{ij} = 1$ if $w_{ij} > 0$ and $I^w_{ij} = 0$ otherwise. In contrast, the analytical solution (Eq. 21) leads to a dense matrix, and therefore it cannot be straightforwardly applied to large-scale problems.

5 Experimental Results

Setup. We evaluate the performance of our Laplacian regularization framework with *AND* (Algorithm 1): the anisotropic non-linear diffusion of the diffusivity operator L^D. To facilitate the comparison of linear and non-linear diffusion processes on L^D, we also implemented *ALD*: the linearization of *AND*, where the evolution of L^D is performed only once at the beginning of the $\mathbf{f}$-diffusion and held fixed thereafter. To assess the role of Laplacian diffusion itself, we compared with three different diffusion-based semi-supervised learning algorithms. The isotropic diffusion algorithm (*Iso*) uses a fixed L throughout the optimization, while the linear anisotropic diffusion algorithm (*AL* [7]) constructs an anisotropic Laplacian L^D which is fixed throughout the subsequent $\mathbf{f}$ optimization. The non-linear anisotropic diffusion algorithm (*AN* [2]) updates L^D during the $\mathbf{f}$-optimization but L^D itself is not diffused. For all experiments, the initial graphs were constructed based on k-NNs using a Gaussian weight function (Eq. 5).

We also conduct comparison with existing semi-supervised learning algorithms including Zhou *et al.*'s local and global consistency algorithm (*LGC* [1]), Zhang and Ando's algorithm that takes powers of Laplacian (*p-L* [11]), and Wang and Zhang's linear neighborhood propagation algorithm (*LNP* [22]). *LGC* is equivalent to constructing an analytical solution as the limit case of *Iso* while *p-L* implements the high-order regularization, which is equivalent to Zhou and Belkin's iterated Laplacian-based algorithm [21]. All four anisotropic diffusion-based algorithms compared in the experiments construct L^D based on evaluations of $\mathbf{f}$. However, in general, the regularizer can be constructed based on any a priori knowledge available on the problem. *LNP* calculates a new Laplacian by representing each point in X based on a convex combination of its neighbors and assigns the corresponding combination coefficients as edge weights $\{w_{ij}\}$.

All algorithms evaluated in this section requires the nearest neighbor size k which was regarded as a hyper-parameter. Also, all algorithms except for *LNP*,

Table 1. Performances of different semi-supervised learning algorithms: Error rates (standard deviations). The three best results are highlighted with **boldface blue**, *italic green*, and *plain orange* fonts, respectively. The last row is a percent error difference from the lowest possible error, averaged across all datasets. The intuition is that a technique achieving 100 % would be best performing in all datasets

	LGC [1]		*p-L* [11]		*LNP* [22]		*Iso*		*AL* [7]		*AN* [2]		*ALD* (ours)		*AND* (ours)	
USPS	7.03	(2.30)	4.07	(1.28)	8.71	(1.82)	7.52	(1.45)	6.00	(2.20)	4.35	(1.38)	*4.09*	(0.47)	**3.28**	(0.47)
COIL	7.65	(2.56)	9.95	(2.82)	**6.31**	(2.03)	8.24	(1.31)	8.45	(1.24)	7.53	(0.74)	7.63	(0.93)	*6.76*	(1.61)
$COIL_2$	1.53	(1.60)	0.80	(0.86)	2.53	(1.18)	0.45	(0.32)	0.79	(0.35)	*0.39*	(0.29)	0.76	(0.26)	**0.33**	(0.21)
PCMAC	12.48	(1.68)	*9.56*	(1.04)	13.52	(1.40)	12.34	(1.89)	13.54	(1.14)	13.46	(2.21)	10.98	(0.82)	**8.96**	(2.00)
Text	25.59	(3.44)	**22.33**	(1.89)	36.31	(3.38)	28.72	(2.13)	27.01	(1.51)	26.87	(2.00)	23.89	(2.28)	*23.09*	(2.07)
ETH	10.92	(1.14)	10.05	(0.93)	13.34	(2.28)	11.62	(1.03)	11.27	(1.18)	*10.04*	(0.88)	**9.96**	(0.77)	10.11	(0.85)
Cal101	53.41	(0.56)	53.41	(0.56)	65.52	(0.59)	53.22	(0.64)	52.36	(0.71)	**52.23**	(0.87)	*52.26*	(0.76)	*52.26*	(0.75)
MNIST	4.50	(0.49)	*3.66*	(0.16)	5.44	(0.47)	5.08	(0.25)	4.92	(0.24)	3.72	(0.28)	4.72	(0.20)	**3.65**	(0.18)
MPEG7	2.93	(0.27)	2.90	(0.08)	N/A	(N/A)	3.33	(0.56)	3.11	(0.52)	2.87	(0.53)	*2.71*	(0.19)	**2.69**	(0.12)
SwL	**2.29**	(0.38)	**2.29**	(0.52)	N/A	(N/A)	2.45	(0.47)	2.54	(0.24)	2.45	(0.44)	2.35	(0.21)	2.38	(0.29)
Avg. %	159.72		124.21		231.72		135.11		140.30		*115.71*		123.82		**101.60**	

requires determining the scale of the input kernel h (Eq. 5). Adopting the experimental convention of Bühler and Hein [23], h was adaptively determined for each point $\mathbf{x}_i \in X$ such that h_i becomes half of the mean distance from $\mathbf{x}_i$ to its k-nearest neighbors. For all diffusion-based algorithms (*Iso*, *AL*, *AN*, *ALD*, and *AND*), the number of iterations T_1 and the time step size δ have to be determined. Our preliminary experiments have demonstrated that the performance of these algorithms are not significantly affected by δ when it is smaller than around 0.2. We fix δ at 0.1 throughout the entire experiments. Except for the linear diffusion *Iso*, all diffusion algorithms also require determining the output kernel scale h' (Eq. 17). In addition, the proposed diffusion of Laplacian algorithms (*ALD* and *AND*) have the number of Laplacian diffusion T_2 as a hyper-parameter. *LGC* has a regularization hyper-parameter which has a similar role to T_1 in the diffusion-based algorithms. *LNP* has a similar hyper-parameter α (See [22]). In addition, *p-L* has the power p as an additional hyper-parameter. Throughout the entire experiments, each training dataset is divided into equally-sized sub-training and validation datasets and the hyper-parameters were tuned based on measuring the validation error. For all experiments with diffusion algorithms, we tune the hyper-parameters of *Iso* first. Then, the parameters of the remaining diffusion algorithms were chosen by restricting the search space of k only at the vicinity of the optimal value (based on the validation error) for *Iso*. Furthermore, the search space of h' for *AN*, *ALD*, and *AND* were similarly determined based on the optimal value of *AL*. This resulted in the total number of *ALD* and *AND* parameter evaluations being only around three times larger than that of *AL*.

We evaluate the performances of all algorithms on ten different datasets: *USPS*, *COIL*, $COIL_2$, and *Text* datasets are commonly used in assessing the performances of semi-supervised learning algorithms. These sets are obtained from Chapelle *et al.*'s benchmark datasets repository. See [24] for details of experimental settings. The experiments for *MNIST* [25] and *PCMAC* are prepared in the same way.

The Swedish leaf dataset (*SwL*) consists of 1,125 images of 15 different tree species [26]. Following Ling and Jacobs [27] and Kim *et al.* [2], we use Fourier descriptors to represent each object and use 50 labels per class. We use the code and dataset kindly shared by [26]. The *MPEG7* shape dataset [28] contains 70 different object silhouettes, each of which has 200 instances. Following [29], we used the pairwise distance matrix obtained as the results of shape matching [30] to construct the initial Laplacian L. For both *SwL* and *MPEG7*, each data point is not explicitly represented. Instead, only the relationships between data points are presented as pairwise distances. For these datasets, the results for *LNP* are not available as it requires explicit data representation. The *ETH-80* dataset consists of 8 different object classes each of which has 410 instances. Each data point is represented by a histogram of oriented gradients, with 50 labels set following conventions [31]. We use the preprocessed dataset kindly provided by [31]. The Caltech101 dataset (*Cal101*) contains 8,677 images of 101 different object categories. We use a combination of GIST, pyramid of histogram of oriented gradients, and local binary patterns as features, as kindly shared by [32]. For all datasets, the experiments were repeated 5 times with random label selections and the results were averaged.

Results. Table 1 summarizes the results. Overall, the baseline isotropic diffusion algorithm (*Iso*) has been significantly improved by anisotropic diffusion algorithms as well as other state-of-the-art algorithms. The linear neighborhood propagation algorithm (*LNP*) and local and global consistency (*LGC*) are the best on *COIL* and *SwL*, respectively. Except for *SwL*, *p-L* constantly outperformed these algorithms, demonstrating the effectiveness of high-order regularization. The performance of *ALD* is roughly on par with *p-L*. This is in accordance with the observation that *ALD* corresponds to an iterative algorithm that optimizes the criteria adopted by *p-L* (Sect. 4). On the other hand, the significantly improved performance of *AND* over *ALD* demonstrates the effect of non-linear Laplacian diffusion.

The general tendency observed among the four anisotropic diffusion algorithms (*AL*, *AN*, *ALD*, and *AND*) is that non-linear diffusion and Laplacian diffusion improve over both linear diffusion and function-only diffusion, respectively. A notable exception is *Cal101* which highlights the main limitations of non-linear diffusion based approaches. While *AN*, *ALD*, *AND* showed the three best performances, all anisotropic diffusion algorithms did not show any noticeable improvement from the isotropic case (*Iso*). This is because the initial labeling based on isotropic diffusion is very noisy (more than 50 % error rate); hence, refining the graph Laplacian based on it as a starting point did not lead to a better diffusion process. Similarly, the results of *LGC* and *p-L* are only comparable to *Iso* while *LNP* is considerably worse.

For *Text* dataset, *p-L* was the best followed by *ALD* and *AND*. These results are obtained based on automatically tuned hyper-parameters. If we use ground-truth hyper-parameters (i.e. keeping the minimal test error during hyper-parameter search), the performance of *AND* is more than 10 % better than *p-L*. This reveals another main limitation of our algorithm *AND*: When the hyper-parameters are tuned properly, *AND* can lead to significant improvements over the linear (Laplacian) diffusion algorithms and over the corresponding limit case *p-Lap*. However, *AND* has one more hyper-parameter than *p-L*, and so with limited labeled data this can lead to worse performance through overfitting. Similarly, for *MNIST* dataset, the performance of *AND* is only marginally better than *p-L*, but when using ground-truth hyper-parameters the error rate of *AND* shows a 14 % improvement over *p-L*.

In general, automatically tuning the hyper-parameters in semi-supervised learning is an open problem especially due to the limited number of labeled data points. Nevertheless, our two algorithms *ALD* and *AND* are included in the three best algorithms for most datasets even with automatically tuned parameters. The effectiveness of our algorithms may be improved in interactive scenarios, where users inspect different parameter combinations and chooses ones best-suited to their problem of interest.

6 Conclusions

The success of anisotropic diffusion in semi-supervised learning problems suggests that adaptively tuning the regularizer to the problem and data can be beneficial. However, existing algorithms did not regularize the regularizer itself. We demonstrated that this idea does lead to improved semi-supervised learning performance. Our framework builds upon the equivalence of inducing anisotropic diffusion and metric construction on Riemannian manifolds: Regularizing the regularizer boils down to regularizing the Riemannian metric tensor on manifolds. Instead of directly regularizing the metric tensor (which is a very difficult problem when the manifold itself is not directly available), we regularize the diffusivity operator on a graph. Our analysis shows that the resulting discrete regularizer converges to the analytical regularizer on metric on Riemannian manifolds. The resulting linear and non-linear anisotropic Laplacian diffusion algorithms significantly outperforms classical diffusion-based algorithms as well as the state-of-the-art semi-supervised learning algorithms.

Acknowledgment. We thank James Tompkin for fruitful discussions and comments, and EPSRC for grant EP/M00533X/1.

References

1. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS, pp. 321–328 (2003)
2. Kim, K.I., Tompkin, J., Pfister, H., Theobalt, C.: Context-guided diffusion for label propagation on graphs. In: Proceedings of the IEEE ICCV, pp. 2776–2784 (2015)
3. Hein, M., Audibert, J.Y., von Luxburg, U.: Graph Laplacians and their convergence on random neighborhood graphs. JMLR **8**, 1325–1368 (2007)
4. Belkin, M., Niyogi, P.: Towards a theoretical foundation for Laplacian-based manifold methods. J. Comput. Syst. Sci. **74**(8), 1289–1308 (2005)

5. Weickert, J.: Anisotropic Diffusion in Image Processing. ECMI Series. Teubner-Verlag, Stuttgart (1998)
6. Clarenz, U., Diewald, U., Rumpf, M.: Anisotropic geometric diffusion in surface processing. In: Proceedings of the Visualization, pp. 397–405 (2000)
7. Szlam, A.D., Maggioni, M., Coifman, R.R.: Regularization on graphs with function-adapted diffusion processes. JMLR **9**, 1711–1739 (2008)
8. Tschumperlé, D., Deriche, R.: Diffusion tensor regularization with constraints preservation. In: Proceedings of the IEEE CVPR, pp. I948–I953 (2001)
9. Castaño-Moraga, C.A., Lenglet, C., Deriche, R., Ruiz-Alzola, J.: A Riemannian approach to anisotropic filtering of tensor fields. Sig. Process. **87**(2), 263–276 (2007)
10. Singer, A., Wu, H.T.: Vector diffusion maps and the connection Laplacian. Commun. Pure Appl. Math. **65**(8), 1067–1144 (2012)
11. Zhang, T., Ando, R.: Analysis of spectral kernel design based semi-supervised learning. In: NIPS, pp. 1601–1608 (2006)
12. Johnson, R., Zhang, T.: Graph-based semi-supervised learning and spectral kernel design. IEEE Trans. Inf. Theor. **54**(1), 275–288 (2008)
13. Kim, K.I., Tompkin, J., Theobald, M., Kautz, J., Theobalt, C.: Match graph construction for large image databases. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7572, pp. 272–285. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33718-5_20
14. Szummer, M., Yilmaz, E.: Semi-supervised learning to rank with preference regularization. In: Proceedings of the ACM CIKM, pp. 269–278 (2011)
15. Fan, M.K.H., Nekooie, B.: On minimizing the largest eigenvalue of a symmetric matrix. Linear Algebra Appl. **214**, 225–246 (1995)
16. Boyd, S., Yang, Q.: Structured and simultaneous Lyapunov functions for system stability problems. Int. J. Control **49**(6), 2215–2240 (1989)
17. Hein, M.: Geometrical Aspects of Statistical Learning Theory. Ph.d. thesis, Fachbereich Informatik, Technische Universität Darmstadt, Germany (2005)
18. Coifman, R.R., Lafon, S.: Diffusion maps. Appl. Comput. Harmonic Anal. **21**(1), 5–30 (2006)
19. Rosenberg, S.: The Laplacian on a Riemannian Manifold. Cambridge University Press, Cambridge (1997)
20. Hein, M., Audibert, J.-Y., von Luxburg, U.: From graphs to manifolds – weak and strong pointwise consistency of graph Laplacians. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 470–485. Springer, Heidelberg (2005)
21. Zhou, X., Belkin, M.: Semi-supervised learning by higher order regularization. In: JMLR W&CP (Proceedings of AISTATS), pp. 892–900 (2011)
22. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: Proceedings of ICML, pp. 985–992 (2006)
23. Bühler, T., Hein, M.: Spectral clustering based on the graph p-Laplacian. In: Proceedings of ICML, pp. 81–88 (2009)
24. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2010). Datasets: http://olivier.chapelle.cc/ssl-book/benchmarks.html
25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
26. Söderkvist, O.: Computer Vision Classification of Leaves from Swedish Trees. Ph.d. thesis, Master thesis, Linköping University, Sweden (2001). Datasets: http://www.dabi.temple.edu/~hbling/code_data.htm
27. Ling, H., Jacobs, D.W.: Shape classification using the inner-distance. IEEE T-PAMI **29**(2), 286–299 (2007)

28. Latecki, L.J., Lakämper, R., Eckhardt, U.: Shape descriptors for non-rigid shapes with a single closed contour. In: Proceedings of the IEEE CVPR, pp. 424–429 (2000)
29. Donoser, M., Bischof, H.: Diffusion processes for retrieval revisited. In: Proceedings of the IEEE CVPR, pp. 1320–1327 (2013)
30. Gopalan, R., Turaga, P., Chellappa, R.: Diffusion processes for retrieval revisited. In: Proceedings of the ECCV, pp. 286–299 (2010). Datasets: http://www.dabi.temple.edu/~hbling/code_data.htm
31. Ebert, S., Fritz, M., Schiele, B.: RALF: a reinforced active learning formulation for object class recognition. In: Proceedings of the IEEE CVPR, pp. 3626–3633 (2012)
32. Dai, D., Gool, L.V.: Ensemble projection for semi-supervised image classification. In: Proceedings of the IEEE ICCV, pp. 2072–2079 (2013). Datasets: http://people.ee.ethz.ch/~daid/EnPro/