An Iterated Graph Laplacian Approach for Ranking on Manifolds

Xueyuan Zhou
Department of Computer Science
University of Chicago
Chicago, IL
zhouxy@cs.uchicago.edu

Mikhail Belkin
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH
mbelkin@cse.ohio-state.edu

Nathan Srebro
Toyota Technological Institute at Chicago
Chicago, IL
nati@uchicago.edu

ABSTRACT

Ranking is one of the key problems in information retrieval. Recently, there has been significant interest in a class of ranking algorithms based on the assumption that data is sampled from a low dimensional manifold embedded in a higher dimensional Euclidean space.

In this paper, we study a popular graph Laplacian based ranking algorithm [23] using an analytical method, which provides theoretical insights into the ranking algorithm going beyond the intuitive idea of "diffusion." Our analysis shows that the algorithm is sensitive to a commonly used parameter due to the use of symmetric normalized graph Laplacian. We also show that the ranking function may diverge to infinity at the query point in the limit of infinite samples. To address these issues, we propose an improved ranking algorithm on manifolds using Green's function of an iterated unnormalized graph Laplacian, which is more robust and density adaptive, as well as pointwise continuous in the limit of infinite samples.

We also for the first time in the ranking literature empirically explore two variants from a family of twice normalized graph Laplacians. Experimental results on text and image data support our analysis, which also suggest the potential value of twice normalized graph Laplacians in practice.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEV**]: Information Search and Retrieval—*Retrieval models*

General Terms

Theory

Keywords

Iterated graph Laplacian, Green's function

KDD'11, August 21–24, 2011, San Diego, California, USA.

1. INTRODUCTION

Information retrieval is becoming progressively more important with the fast growing amounts of digital data, and as Internet search engines become necessary tools for finding useful information among huge amounts of data on the Web. In retrieval, ranking is a key step to provide a "relevance" score for the search results so that the most "relevant" results are presented to users first.

In recent years, the graph Laplacian has become an important object in manifold related machine learning and data mining [5, 26, 1, 23]. The ranking on manifolds algorithm in [23] estimates a real-valued function on the whole data set by "diffusion" based on a graph Laplacian, which can be used as a density adaptive ranking score for all data points. One key advantage of this family of graph Laplacian based ranking algorithms is that they are density adaptive, combining local information encoded in the local neighborhood graph and global information from the graph Laplacian into a final ranking function.

Given the success of Laplacian based methods in machine learning and data mining, there is still relatively little analytical analysis of these algorithms. One example is [13], which contains a surprising finding for a family of popular graph Laplacian based semi-supervised learning methods. It turns out that when labeled data is fixed and unlabeled data increases, the estimator on unlabeled points degenerates to a constant. The essential cause of this degenerate behavior is that the solution space is too rich, which leads to overfitting. See [24] for more discussions and a solution based on iterated Laplacians.

The semi-supervised problem setting is related to the ranking problem in [23], which motivates the analysis of these algorithms from a functional analysis point of view to obtain new theoretical insights going beyond the intuitive "diffusion" analogy. For instance, we can ask the following questions about ranking: what is the ranking function space? what are its properties? and, what should the ranking function be in the limit of infinite data? These questions are not just basic theoretical foundations of ranking, but are also crucial to the applications of ranking algorithms in practice.

In this paper, we try to answer these questions. Our contributions are as follows: first, we study a graph Laplacian based ranking on manifolds algorithm [23] from a functional analysis point of view. We show that the algorithm is sensitive to a commonly used parameter due to its use of the symmetric normalized graph Laplacian. This is not the case

when the unnormalized graph Laplacian is used. This observation is different from the case of spectral clustering considered in [21], where the normalized graph Laplacian turns out to have better theoretical properties.

We also show that the ranking function diverges to infinity at the query point in the limit of infinite samples in higher dimensions. A close relation between the Green's function and the reproducing kernel based on the graph Laplacians is discussed, which connects the graph Laplacian based ranking algorithms to kernel methods.

Second, we propose an approach based on the iterated graph Laplacian, using unnormalized graph Laplacians. Compared to the existing method [23], our method has several preferable properties: it is more robust, more density adaptive, and behaves well as the amount of unlabeled data increases.

Finally, we test two interesting variants from a family of twice normalized graph Laplacians, which also give competitive results on ranking and suggest their potential values in practice. To the best of our knowledge, this is the first time they are used in ranking.

We review the graph Laplacian in section (2), including the finite sample case and their continuous limits. Then in section (3), we study an existing graph Laplacian based ranking on manifolds algorithm [23]. Based on the analysis, we suggest using an iterated unnormalized graph Laplacian for ranking in section (4). We provide experimental results on text and image benchmarks.

1.1 Problem Setup

Given a random sample set $X = \{x_1, x_2, \cdots, x_n\}$, where x_i is a random vector, drawn i.i.d. from a fixed unknown smooth probability distribution $p(x)$ $(0 < a \leqslant p(x) \leqslant b < +\infty)$. We assume the density is supported on Ω, which is either a compact subset of $\mathbb{R}^d$ or a d-dimensional Riemannian submanifold of $\mathbb{R}^N$. The problem of ranking considered in this paper is to find a ranking score function $f(x) : x \mapsto \mathbb{R}$, providing a "similarity" measure between the query point x_i and all other points $x_j \in X/\{x_i\}$, such that the smaller the value $|f(x_i) - f(x_j)|$ is, the more similar x_i and x_j are. Let the initial query be x_1, and the ranking function value at x_1, i.e., $f(x_1)$ be known, which is typically set as $f(x_1) = +1$. The analysis in this paper also applies to multiple query points. By a convenient abuse of notation, $f(x)$ means both the continuous function value at x in the limit of infinite points and the x^{th} element of the column vector f such that $f_x = f(x)$ in the case of finite X.

2. GRAPH LAPLACIAN

In this section, we review two graph Laplacians that will be used later, including both their discrete forms and their continuous limits.

2.1 Graph Laplacians

Given the sample set X, we can build an undirected weighted graph $G(V, E)$ such that x_i is mapped to the vertex v_i, and the edge weight $w(e_{ij}) = w_{ij}$ is a similarity measure between x_i and x_j. In this paper, we use a symmetric k nearest neighbor (k-NN) graph [12] as $G(V, E)$. A common weight function is

$$w_{ij} = K\left(\frac{\|x_i - x_j\|^2}{t}\right) = e^{-\frac{\|x_i - x_j\|^2}{t}} \tag{1}$$

Denote the connection weight matrix on graph G as W, and let D be a diagonal matrix with $D_{ii} = D(x_i) = \sum_j w_{ij}$. There are several ways of defining a graph Laplacian [20]. We first introduce the unnormalized and symmetric normalized graph Laplacians L_u and L_s, which are closed related to the ranking algorithm we will study. Other interesting graph Laplacians will be introduced later in section (4.2).

$$\begin{aligned} L_u &= D - W \\ L_s &= D^{-1/2} L_u D^{-1/2} = I - D^{-1/2} W D^{-1/2} \end{aligned} \tag{2}$$

Both L_u and L_s are real symmetric and semi-definite matrices. We order the eigenvalues of all graph Laplacians in this paper in an increasing order $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$. For L_u and L_s, the smallest eigenvalues are both zero. This means L_u and L_s are not full rank. From norm point of view, it means $f^T L_u f$ and $f^T L_s f$ are semi-norms with null spaces spanned by their first eigenvectors. The first eigenvector of L_u is constant,

$$v_1 = \left(\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \cdots, \frac{1}{\sqrt{n}}\right)^T \tag{3}$$

while for L_s, a non-constant vector

$$v_1 = \left(\sqrt{D(x_1)}, \sqrt{D(x_2)}, \cdots, \sqrt{D(x_n)}\right)^T \tag{4}$$

where $D(x_i)$ is the degree for vertex x_i in graph G.

2.2 Limits of Graph Laplacians

As we obtain more sample points, the parameter t needs to be decreased in order to let the graph Laplacian to better capture local information. Therefore, the limit analysis of a graph Laplacian involves two limits, the number of data points, $n \to \infty$, and the kernel width, $t \to 0$. See [3, 7, 11] for more on limit analysis of graph Laplacians. On finite sample set X,

$$\frac{1}{n} L_u f(x_i) = \frac{1}{n} \sum_{j=1}^{n} K\left(\frac{||x_i - x_j||^2}{t}\right)(f(x_i) - f(x_j)) \tag{5}$$

where $f(x_i)$ is the i^{th} element of vector f.

In the limit as $n \to \infty$ and $t \to 0$ at a proper rate (see [3, 11]), for a smooth function $f(x) \in C^2$, the limit of $L_u f(x)$ is

$$\frac{1}{n t^{d/2+1}} L_u f(x) \xrightarrow{\text{a.s.}} -p(x)\Delta_{p^2} f(x) \tag{6}$$

where $\Delta_{p^2} f(x) = \frac{1}{p^2(x)} \mathrm{div}[p^2(x) \mathrm{grad} f(x)]$ is called the *weighted Laplacian*, see e.g., [9]. In the limit, the degree function $D(x)$ converges to the underlying density $p(x)$ up to a constant coefficient, see e.g., [10, Chapter 2].

However, the limit of L_s is not a weighted Laplacian of $f(x)$, as shown in [11]

$$\frac{1}{t^{d/2+1}} L_s f(x) \xrightarrow{\text{a.s.}} -\sqrt{p(x)}\Delta_{p^2}\left[\frac{f(x)}{\sqrt{p(x)}}\right] \tag{7}$$

where $1/n$ is canceled by the normalization. This means if we use L_s as an empirical Laplacian in an algorithm, it is not a strictly "diffusion" based algorithm. If we let $F(x) = f(x)/\sqrt{D(x)}$, then the limit is a weighted Laplacian for function $F(x)$, but not $f(x)$. We will see later in section (3.2), this delicate detail has a great influence on ranking.

From the finite sample and infinite limit analyses of the graph Laplacians, it is easy to see that the null space of L_u is

spanned by a constant vector, and in the limit it is spanned by a constant function. While for L_s, the difference is a weight: the null space is spanned by vector $\sqrt{D(x)}$ as in equation (4) in discrete case, and by function $\sqrt{p(x)}$ in the continuous limit. See [2] for the convergence of graph Laplacian eigenvectors to weighted Laplacian eigenfunctions.

3. ANALYSIS OF RANKING ON MANIFOLDS

In [23], the authors proposed a ranking function (column vector) as

$$f = (I - \alpha S)^{-1} y \tag{8}$$

where y is a column vector with the query element y_1 set as 1, and 0 on all other points, $S = D^{-1/2} W D^{-1/2}$, I is an identity matrix, and $\alpha \in [0, 1)$ is a tuning parameter. This algorithm is based on the intuition of "information diffusion." The idea is to propagate the query "information" along the data manifold in a density adaptive way.

Next, we show that it can be rewritten using the symmetric normalized graph Laplacian L_s. By splitting the identity matrix into $(1-\alpha)I + \alpha I$, we can rewrite f as

$$f = \alpha \left[(\frac{1-\alpha}{\alpha}) I + L_s \right]^{-1} y = \alpha (\beta I + L_s)^{-1} y \tag{9}$$

where $\beta = \frac{1-\alpha}{\alpha} > 0$. In ranking, we can drop the global constant α without changing the result. Then the ranking function is equivalent to $f = (\beta I + L_s)^{-1} y$.

A closer look at this problem can reveal that it is also the solution of two other equivalent problems (up to constant coefficients): a discrete Laplace equation and a discrete potential energy minimization problem. See [23] for more details, and see [16] for more on boundary value problems. Next, we further study the ranking function $(\beta I + L_s)^{-1} y$.

3.1 Green's Function and Reproducing Kernel in Ranking

Consider the ranking function $(\beta I + L_s)^{-1} y$. Since only one element of y is nonzero, the ranking function f (column vector) is, in fact, the first column of matrix $(\beta I + L_s)^{-1}$, i.e., the column corresponding to the query point. Thus, this inverse matrix is the key object for the ranking problem.

Reproducing Kernel of Reproducing Kernel Hilbert Space: The function space equipped with a (semi-)norm of the form $f^T Q f$ with a (semi-)definite matrix Q is a reproducing kernel Hilbert space (RKHS)[1], and its reproducing kernel matrix is the pseudoinverse of Q, denoted as Q^+, see e.g., [4, Chapter 6] and [17]. Let $Q = (\beta I + L_s)$, which is positive definite since $\beta > 0$, and its inverse exists. Ranking function f then is just a kernel function centered at query point x_1, i.e., $f(x) = K(x_1, x)$, where $K = (\beta I + L_s)^{-1}$ is the reproducing kernel matrix. We denote $K(x, y)$ as the (x, y) element of the matrix K in discrete case.

The additional term βI is used to complete the semi-norm $f^T L_s f$ to a norm, since L_s is not full rank. If we remove βI ($\beta = 0$), and use the pseudoinverse of L_s, we can still obtain a ranking function f as a kernel function, but it is in the subspace that is orthogonal to the first eigenvector of L_s.

The Green's Function: We know the Green's function of a Laplace operator can be seen as the inverse of a Laplacian, which implies $G_s = (\beta I + L_s)^{-1}$. See [6] for more on discrete Green's function on graphs. By eigenvector expansion we have

$$G_s(x, y) = \sum_{k=1}^{n} \frac{1}{\lambda_k + \beta} v_k(x) v_k(y) = K(x, y) \tag{10}$$

where λ_k is the k^{th} eigenvalue of L_s and v_k is the associated eigenvector. This is the same as kernel function $K(x, y)$ since $K = (\beta I + L_s)^{-1}$. Notice that if we restrict the graph Laplacian to a subgraph, then it has full rank and is invertible. We consider the Green's function over the whole graph, which is used in [23].

3.2 Null Space Effect

For the ranking function $f = (\beta I + L_s)^{-1} y$, parameter β plays an important role in this problem. On one hand, from a numerical computation point of view, βI makes $L_s + \beta I$ invertable since L_s is not full rank. On the other hand it "kills" the null space of L_s, which is spanned by the first eigenvector of L_s. The ranking function (the kernel/Green's function) can be written as follows since $\lambda_1 = 0$.

$$G_s(x, y) = \frac{1}{\beta} v_1(x) v_1(y) + \sum_{i=2}^{n} \frac{1}{\lambda_i + \beta} v_i(x) v_i(y) \tag{11}$$

This means if β is too small such that $\frac{1}{\beta} >> \frac{1}{\lambda_i + \beta}$, the behavior of the ranking function might be determined by $v_1(x)$ alone. In ranking, if $v_1(x)$ is a function that is unrelated to the relevance ordering of samples, we will have an uninformative ranking function. This is the case for L_s when the density $p(x)$ is not uniform, since $v_1(x)$ is non-constant and determined by the denisty. For query x_1, by dropping a constant coefficient $v_1(x_1)$, the ranking function then becomes

$$f(x) = G_s(x_1, x) \approx \frac{v_1(x)}{\beta} \tag{12}$$

This means the ranking function $f = (\beta I + L_s)^{-1} y$ is very sensitive to β. Recall that for L_s, we have $v_1(x) = \sqrt{D(x)}$, where $D(x)$ is the degree function for vertex x on the graph. In this case,

$$f(x) \approx \frac{\sqrt{D(x)}}{\beta} \tag{13}$$

This also implies that, for small β, the ranking on graph G is influenced by methods used to construct G. For instance, parameter k in kNN graphs directly varies degree function $D(x)$. Likewise $D(x)$ is affected by the decision to build either a symmetric or asymmetric kNN graph. These are important issues in practice. From the original "diffusion" based iterative algorithm [23], it is difficult to discover these issues.

When $v_1(x)$ is constant, which is the case for L_u, $v_1(x)/\beta$ will not change the ranking since we only need an ordering. In this case, we can safely drop the constant shift $v_1(x)/\beta$, then the second term in equation (11) determines the ranking. This means L_u will be more robust than L_s in terms of parameter β. This problem is less important in regression since the task there is to estimate a real-valued function, while in ranking, we ultimately need an ordering. Notice that other than the null space effect, β also influences the ranking in "spectra shift" $1/(\lambda_i + \beta)$.

One simple solution is to use L_u instead of L_s. Another solution is to modify the ranking function when using L_s.

[1]When there exists a null space, we mean the subspace orthogonal to its null is an RKHS.

Since this problem is caused by the first eigenvector, associated with the zero eigenvalue, one straightforward solution is to remove the first eigenvector. Then the ranking function becomes

$$f(x) = \sum_{i=2}^{n} \frac{1}{\lambda_i + \beta} v_i(x_1) v_i(x) \tag{14}$$

Function $f(x)$ lives in the subspace that is orthogonal to the first eigenvector, since all the eigenvectors of L_s are orthogonal. This can be implemented by computing the pseudoinverse of L_s, which removes the first eigenvector[2]. Notice that even we remove v_1, the ranking functions using L_s and L_u are still different due to the differences between their eigenvectors.

Therefore, β should not be set to any fixed value for different data sets in practice when using L_s. Instead, β should be chosen by validation. When training data is not available or not enough for validation, β should be set to zero and the pseudoinverse should be used to compute the ranking function. We will test how parameter β changes the ranking results in the experiments.

3.3 Diverging Ranking Function

Since the ranking function can be rewritten as a Green's function of a Laplacian, then it is possible that it is not a pointwise well-defined continuous function in the limit of infinite samples. Instead of an analysis involving generalized functions and Sobolev spaces, we use a simple example to show how this Green's function diverges to infinity at the query point in spaces having dimensions greater than one ($d \geqslant 2$).

Assume we remove the first eigenvector and ignore the density, then the ranking function is the Green's function of a regular Laplacian Δ. In $\mathbb{R}^3$ or higher dimensions, the Green's function of a regular Laplacian is

$$G(x, y) = \frac{1}{\|x - y\|_{\mathbb{R}^d}} \tag{15}$$

where $|G(x, x)| = \infty$. In our case, $G_s(x, x) = K(x, x) = \infty$ means if we enforce $y_1 = 1 = G_s(x_1, x_1)$, the diverging $G_s(x_1, x_1)$ will cause the ranking function to be 0 at all other points by a normalization as the following.

$$f(x) = \frac{G_s(x_1, x)}{G_s(x_1, x_1)} = \begin{cases} 1, \text{when } x = x_1 \\ 0, \text{when } x \neq x_1 \end{cases} \tag{16}$$

In $\mathbb{R}^2$, we also have $|G(x, x)| = \infty$, but the Green's function has a different form. Thus, this problem happens for graph Laplacians in spaces having dimensions greater than one[3]. More importantly, this problem happens to all versions of graph Laplacians, including L_u, L_s and all the graph Laplacians that will be introduced later. For L_s, the bounded density weight will not change this fact.

One trivial solution to this problem is that there is no need to maintain $f(x_1) = y_1 = 1$, since we only need an ordering, instead of a real-valued function $f(x)$ as in regression. In the finite sample case, it is possible to obtain a real-valued ranking function (vector) since $G_s(x, x)$ will always be finite. However, due to the sharp increase of $G_s(x, x)$ in high dimensions as we have more and more samples, the random noise might easily damage the ranking.

In order to solve the diverging problem of the ranking function in the limit of infinite samples, we need to find the Green's function of a "higher order" Laplacian, in a higher order Sobolev space, see e.g., [19]. This idea can be implemented by the Green's function of an iterated graph Laplacian defined as the following,

$$L_u^m = \sum_{k=1}^{n} \lambda_k^m v_k v_k^T$$

where λ_k and v_k are the eigenpair of L_u and $m \geqslant 0$. Similarly, we can also define the corresponding iterated graph Laplacian for L_s and other version of graph Laplacians. Since L_u is real and symmetric, for integer m, L_u^m is exactly the power of matrix L_u.

4. RANKING BY GREEN'S FUNCTION

Based on previous analyses, the null space effect of L_s suggests that we should use a graph Laplacian that has a constant first eigenvector, or remove the first eigenvector of L_s in the ranking function. The diverging ranking function problem suggests to use the Green's function of an iterated graph Laplacian. In this section, we use L_u as an example to show an improved solution, then discuss the improved method on a Gaussian mixture, and at last we introduce several other interesting graph Laplacians.

The Green's function of the iterated Laplacian L_u^m, with a possible additional term βI but allowing $\beta = 0$ (meaning we remove the first eigenvector by pseudoinverse) is defined as [4]

$$G_u^m(x, y) = \sum_{i=1}^{n} \left(\frac{1}{\lambda_i + \beta} \right)^m v_i(x) v_i(y) \tag{17}$$

When $\beta = 0$, the pseudoinverse should be used to remove $v_1(x)$. It is easy to see that in matrix form $G_u^m = [(\beta I + L_u)^{-1}]^m$. In the limit of infinite samples, as long as $2m > d$, $f(x) = G_u^m(x_1, x)$ is a continuous function by the Sobolev embedding theory. See e.g. [19] for more on Laplacians and Sobolev spaces. The ranking function then becomes the following.

$$f = \left[(\beta I + L_u)^{-1} \right]^m y, \quad \beta \geqslant 0. \tag{18}$$

Besides β, the other important parameter is m, which defines a spectra transform of the form $g(\lambda) = \lambda^m$ on L_u. For a weighted Laplacian, the smaller λ_i is the smoother v_i will be. This means the eigenvectors are in decreasing smoothness order when we order λ_i in an increasing order. For the Green's function, eigenvalues are $1/\lambda_k^m$ when $\beta = 0$. Then the power transform of the spectra acts as a low pass filter, smoothing the Green's function as m increases. This idea can be quantitatively described by Sobolev norms, see [19, Chapter 5].

Overall, the two parameters $\beta \geqslant 0$ and $m \geqslant 0$ act as a spectra transform to the Green's function, where m acts as a scaling between different "frequency" levels, while β acts as a shift along a fixed "frequency" level. This effect is similar to the Fourier analysis and multiple resolution analysis, since the eigenvectors of a graph Laplacian not only form

[2]This can be implemented by pseudoinverse "pinv" of L_s in Matlab by setting a small threshold for eigenvalue cutoff.

[3]Intrinsic dimension $d \geqslant 2$ on manifolds.

[4]It is also possible to define $G_u^m(x, y) = \sum_{i=1}^{n} 1/(\lambda_i^m + \beta) v_i(x) v_i(y)$.

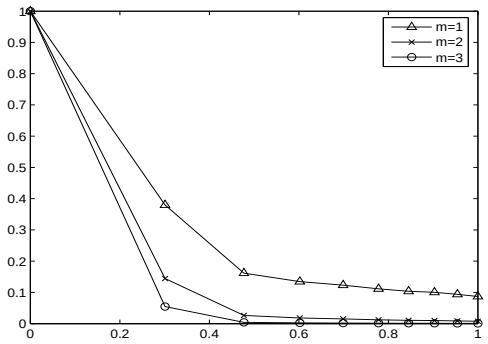

Figure 1: Low-pass filter effect: re-scaled top 10 eigenvalues (λ_i^{-m}) of G_u^m on a mixture of two Gaussians in 2D (without $\lambda_1 = 0$). Horizontal axis is $\log_{10}(i)$, vertical axis is λ_i^{-m}, and $\beta = 0$.

an orthonormal basis, but also have an intrinsic smoothness ordering. The Green's function of an iterated graph Laplacian also has a similar spectral density as the Matérn model [18] in a Gaussian process.

4.1 Empirical Study on Gaussian Mixture

In order to have an intuitive understanding of the effect of m and β, we generate an artificial data set and plot the transformed spectra and ranking function in Figure (1), Figure (2) and Figure (3). The data set includes a mixture of two Gaussians of unit variance on $\mathbb{R}^2$ centered at $(\pm 1.5, 0)$, and an additional uniform over $[-3, 3] \times [-3, 3]$ to avoid empty regions. The query point is at $(-2, -1.8)$.

In Figure (1), we re-scale the eigenvalues of G_u^m to make the spectra of different m values are comparable. As m increases, we can see that the low frequency components (eigenvectors with smaller indices) have larger and larger weights in the final ranking function. This makes the ranking function smoother and smoother, but still in a data dependent way. This low-pass filter effect has a different behavior from that in [7], where a Markov random walk explanation applies. The eigenvalues here connect closely to the Sobolev spaces [19] and decrease much faster.

For a good ranking function, the top of the ranking list should be points that are, on one hand near the query point in terms of the Euclidean distance in the chosen feature space. On the other hand, points on the top of the list should shift towards the nearby high density regions in order to obtain more relevant results on average. In Figure (2), the contour lines of the Green's function using both L_u and L_s and the top 100 ranked points are shown, where we can see two distinct features.

First, the null space effect for L_s is clear. In the upper row of Figure (2), different β values change the behavior of L_s-based ranking function dramatically. When $\beta = 10^{-4}$, the Green's function using L_s is almost the same as eigenvector v_1 of L_s, which is the square root of the density of a mixture of two Gaussians. The top 100 ranked points are those on the same contour lines, which is completely uninformative for ranking. For "pinv" ($\beta = 0$), the pseudoinverse of L_s does not include v_1, so the Green's function is density adaptive and the top 100 ranked points are reasonable neighbors in $\mathbb{R}^2$. When $\beta = 1$, the Green's function acts as a Gaussian in $\mathbb{R}^2$, capturing no density information. When $\beta = 0.01$, the results are reasonable and similar to "pinv" case. This problem does not happen to L_u (lower row), since its first eigenvector is constant. The results are much stable for different β values. In the contour plot for L_u when $\beta = 10^{-4}$, even the Green's function is almost flat as a constant (indicated by its contour line values), but a reasonable ordering can still be recovered. Any constant shift will not change the ordering, which is essential for ranking. Although L_s can still be useful with a proper β value, in practice choosing a sensitive parameter can be difficult.

Notice that the slight difference of the ranking functions for different β using L_u is caused by the weight $1/(\lambda_i + \beta)$. Overall, the effect of β is twofold: β not only plays an important role in the first eigenvector as in $1/\beta$, but also changes other eigenvector components as in $1/(\lambda_i + \beta)$.

Second, as m increases, the top 100 ranked points are shifted towards the nearby high density regions, as shown in Figure (3). This will increase the relevant results on average in practice. Results using L_u also have a similar behavior.

4.2 Different Graph Laplacians

In the experiments, we also tested several other interesting versions of graph Laplacians, which are introduced here. Given the weight matrix W constructed from the sample set X, there are three empirical graph Laplacians associated with W: L_u, L_s, and L_r. We introduced L_u and L_s in section (2), and $L_r = D^{-1}L_u = I - D^{-1}W$. It is easy to see that L_r is a weighted version of L_u. Since L_r is not symmetric, above analyses will not apply directly to L_r. Therefore, we only test it empirically in the experiments.

There exists another way of defining graph Laplacians, which involves a normalization of W using a parameter $\alpha \in \mathbb{R}$ [7]. First, the weight matrix W is normalized to another weight matrix

$$\tilde{W}_\alpha = D^{-\alpha} W D^{-\alpha} \tag{19}$$

Then we can define the associated degree matrix $\tilde{D}_\alpha$ for $\tilde{W}_\alpha$, and define each graph Laplacian accordingly as before. For each α, again there are three empirical graph Laplacian $\tilde{L}_u^\alpha$, $\tilde{L}_s^\alpha$ and $\tilde{L}_r^\alpha$. When $\alpha = 1/2$, denote $\tilde{L}_u^{1/2}$ as L_p[5], and when $\alpha = 1$, denote $\tilde{L}_u^1$ as L_g. In the limit of infinite samples [10, Corollary 2.40]

$$\frac{1}{nt^{d/2+1}} L_p f(x) \xrightarrow{\text{a.s.}} -\frac{1}{p(x)} \text{div}[p(x)\text{grad} f(x)] \tag{20}$$

In this case, the measure hidden inside of the weighted Laplacian is $p(x)$, instead of $p^2(x)$ as for L_u. This graph Laplacian might be preferred in practice since the random samples are drawn from $p(x)$ instead of $p^2(x)$. For L_g, we have

$$\frac{1}{nt^{d/2+1}} L_g f(x) \xrightarrow{\text{a.s.}} -\frac{1}{p(x)} \Delta f(x) \tag{21}$$

In this case, the limit has no density drifting term, but with a density outside of the Laplace operator. We will test the empirical performances of these two graph Laplacians in the experiments.

[5]Notice that $f^T L_p f$ is different from $f^T L_s f$, since $f^T L_p f = \frac{1}{2}\sum_{i,j} w_{ij}/\sqrt{D(x_i)D(x_j)}(f(x_i) - f(x_j))^2$ and $f^T L_s f = \frac{1}{2}\sum_{i,j} w_{ij}(f(x_i)/\sqrt{D(x_i)} - f(x_j)/\sqrt{D(x_j)})^2$.

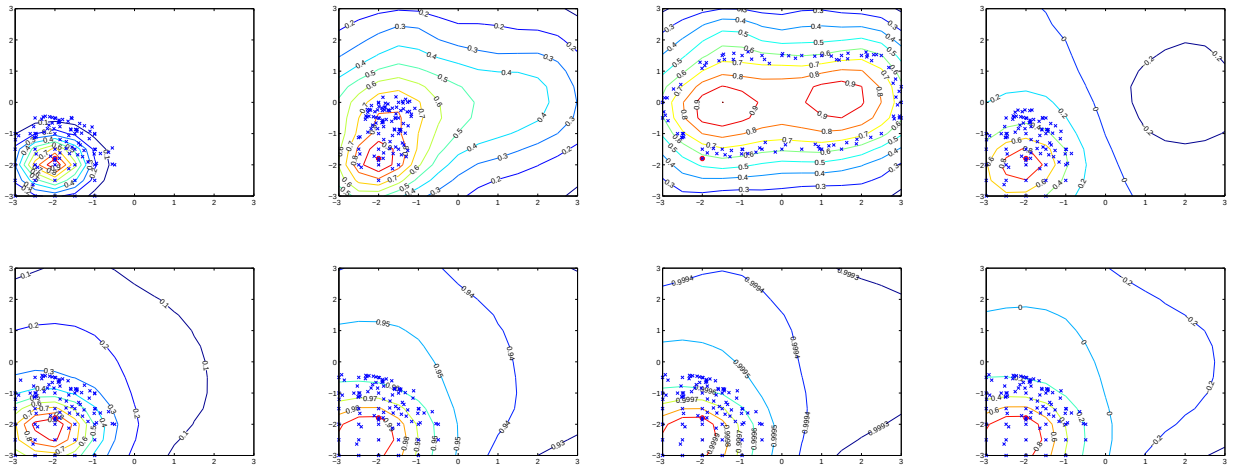

Figure 2: Sensitivity of β on L_s(top row) and L_u(lower row): Green's function at $(-2,-1.8)$ over a mixture of two Gaussians at $(\pm 1.5, 0)$ with $m = 1$. Left to right: $\beta = 1$, $\beta = 10^{-2}$, $\beta = 10^{-4}$, pinv ($\beta = 0$).

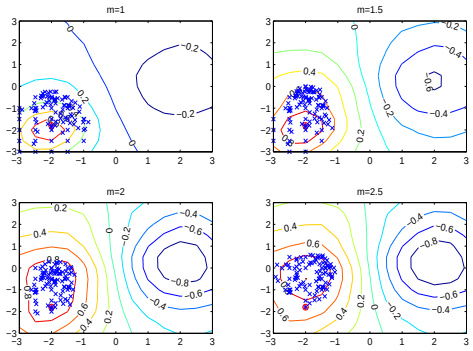

Figure 3: Green's function with different m values using L_s.

5. EMPIRICAL STUDY

In this section, we test the ranking function based on the iterated graph Laplacians on several text and image benchmarks.

5.1 Experiment Setup

Data Sets: Benchmarks include COIL20 images[6] [14], MNIST image digits[7], 20 Newsgroups text data[8], and Scene image data set used in [15].

The COIL20 data set contains 20 objects. The images of each object were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images. The total number of the images in this data set is 1440, and the size of each image used in this paper is 32×32 pixels, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional raw pixel vector. The MNIST digits image data set consists of 10 handwritten digits, and each image is represented by a 784-dimensional raw pixel vector. In this paper, we use a subset of 2000 images of the original data set, with 200 images for each digit. The 20 Newsgroups data set includes documents of 20 different classes, and each document is represented as a 61188-dimensional TFIDF vector. A subset of 2000 documents is used, with 100 from each group. The Scene data set contains 8 outdoor scene categories. A subset of 2000 images is used, with 125 from each category. After computing the Gist descriptor for each image using online code[9] with 4×4 blocks, each image is represented as a 512-dimensional vector. For COIL20, MNIST, and Scene data sets, the Gaussian weight is used to compute the weight on weighted graphs, while for 20 Newsgroups, the Cosine weight is used.

Evaluation Criteria: The main evaluation criteria in this paper is mean average precision (mAP). The average precision (AP) is the area below the recall-precision curves. The AP is a number between 0 and 1 with 1 meaning the first M returned points are all the relevant data in the whole data set. The mean AP is the average of AP for a set of queries. In this paper, we use each sample point of the data set as a query, then take the average over all the AP scores. We also use the precision-scope curves to show the performance on the top of the ranked list on Scene data. Since in image retrieval, results on the top of the returned list are much more important than others to users.

5.2 Experimental Results

There are four parameters for ranking using the iterated graph Laplacians: k and t for the graph construction, and β and m for the ranking function. For graph Laplacian based machine learning and data mining methods, graph construction related parameter k in kNN graphs and t in Gaussian weights have been studied both theoretically and empirically, see e.g., [26] and the reference therein. In order to capture the local information of the underlying manifold, both k and t should work together to generate a good graph. For example, a small k with a wide range of t, or a large k

[6]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[7]http://yann.lecun.com/exdb/mnist/
[8]http://www.zjucadcg.cn/dengcai/Data/TextData.html
[9]http://people.csail.mit.edu/torralba/code/spatialenvelope/

Table 1: mAP on different data sets, with different β values. "pinv" corresponds to results from ranking functions with the first eigenvector removed, as in (14).

β	10^{-5}	10^{-4}	10^{-3}	$\mathbf{10^{-2}}$	10^{-1}	1	pinv		10^{-5}	10^{-4}	10^{-3}	$\mathbf{10^{-2}}$	10^{-1}	1	pinv
	MNIST, $k=10$, $t=10^6$, $m=1$								COIL20, $k=10$, $t=10$, $m=1$						
L_u	0.57	0.57	0.57	**0.57**	0.56	0.51	0.57		0.80	0.80	0.80	**0.81**	0.81	0.77	0.66
L_s	0.11	0.12	0.18	**0.50**	0.54	0.48	0.57		0.36	0.48	0.78	**0.80**	0.78	0.75	0.65
	20News, $k=10$, $m=1$								Scene, $k=10$, $t=0.10$, $m=1$						
L_u	0.47	0.47	0.47	**0.46**	0.44	0.39	0.47		0.57	0.57	0.57	**0.57**	0.56	0.52	0.57
L_s	0.07	0.07	0.10	**0.33**	0.43	0.38	0.47		0.14	0.14	0.18	**0.45**	0.54	0.50	0.56

but with a relatively small t. In this paper, we focus on β and m.

Parameter β: We first compare the original ranking algorithm in [23] using L_s as in equation (18) with different values of β and $m=1$, to the algorithm using other graph Laplacians. The mAP scores for L_s and L_u with different β values are reported in Table (1). We can see that the ranking algorithm using L_s is very sensitive to parameter β. This confirms our null space effect analysis in section (3.2). The commonly used value of β, which is approximately 0.01, is far from being a good choice for L_s (except the special data set COIL20). Contrary to L_s, ranking results using L_u are much more stable on β.

One interesting observation is that, when we do not have enough training data to choose β by validation, using the pseudoinverse by setting $\beta = 0$ also gives competitive results. This is true for both L_u and L_s, as shown in "pinv" column in Table (1).

We also tested L_r, L_p and L_g, the results of which are not reported here due to the limited space. The performance of L_r is even worse than L_s, which is potentially due to the asymmetric normalization. The ranking performance of L_p and L_g is comparable to that of L_u when other parameters are chosen as in Table (1).

Parameter m: How parameter m changes the ranking function depends on the underlying density as shown in Figure (2), and also depends on how we normalize the graph Laplacian. In Table (2), mAP on different data sets with different m values is reported, where 5-fold cross validation is used to choose parameter k, t, and β. We choose $k \in \{5, 10, 50, 100, 500, 1000, |X|\}$, with $|X|$ meaning complete graphs, $\beta \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$, on MNIST $t \in \{10^0, 10^1, 10^2, 10^3, 10^4, 10^5\} \times 10^5$, and on Scene $t \in \{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.

We can see that results with $m > 1$ on the three data sets can be better than ranking with $m = 1$, which is particularly clear for L_u, and also generally true for L_p and L_g, except L_p on MNIST. But most values of $m > 1$ are not helpful for L_s and L_r. This is probably due to the fact that the normalization step complicates the corresponding iterated Laplacians, which is particularly true considering that the limit of L_s is not a weighted Laplacian.

Notice that the results in Table (2) are obtained by choosing the best parameters by validation, including β. This means we already eliminated the null space effect in Table (2). In large scale computing, validation might not be available.

Different Graph Laplacians: Among different versions of graph Laplacians, we find that the results of three "unnormalized" graph Laplacians (L_u, L_p, and L_g) are roughly comparable, as shown in Table (2). To the best of our knowledge, this is the first time that L_p and L_g are tested in ranking. One interesting result is that on MNIST, the ranking using L_p with $m = 1$ is much better than all other graph Laplacians. Based on Table (2), if we have to choose one graph Laplacian to use, then L_p is the best choice. The overall performance of L_p suggests the potential value of L_p and the family of twice normalized graph Laplacian $\tilde{L}^u_\alpha$. This is interesting since among all the weighed graph Laplacians $\tilde{L}^u_\alpha$, L_p is the only "natural" one (see e.g. [25]), whose weight corresponds to the probability density $p(x)$, i.e., $\int_\Omega p(x)dx = 1$. Notice that for L_u, the weight is $p^2(x)$ and $\int_\Omega p^2(x)dx \neq 1$ generally.

Table 2: mAP on different data sets, with different m values. Results in bold are the best for each graph Laplacian, choosing the smallest m in case of a tie.

m		1	2	4	8	16	32
MNIST	L_u	0.63	**0.65**	0.65	0.65	0.62	0.59
	L_s	**0.65**	0.65	0.64	0.59	0.63	0.62
	L_r	**0.65**	0.62	0.58	0.58	0.61	0.54
	L_p	**0.69**	0.68	0.64	0.66	0.65	0.65
	L_g	0.62	**0.65**	0.65	0.63	0.64	0.63
20News	L_u	0.45	0.49	0.51	0.49	**0.52**	0.47
	L_s	0.47	**0.50**	0.44	0.47	0.46	0.37
	L_r	0.47	**0.49**	0.41	0.43	0.40	0.24
	L_p	0.44	0.48	**0.51**	0.47	0.51	0.48
	L_g	0.43	0.47	**0.50**	0.47	0.50	0.47
Scene	L_u	0.56	0.58	**0.59**	0.57	0.57	0.57
	L_s	**0.57**	0.55	0.53	0.55	0.56	0.47
	L_r	**0.55**	0.51	0.52	0.52	0.47	0.45
	L_p	0.56	0.58	**0.59**	0.56	0.59	0.59
	L_g	0.56	0.57	**0.59**	0.57	0.59	0.58

Content Based Image Retrieval: A natural application of ranking is content based image retrieval. We test different graph Laplacians on Scene image data set, and the precision-scope curves for L_u, L_s and L_r are shown in Figure (4). Due to the limited space, we only report one typical result with a fixed parameter setting. However, results of several different k and t combinations follow a similar pattern: L_u is better than L_s, and L_s is better than L_r.

Study on COIL20: The construction of COIL20 can be seen as an idealization of a real world scenario. Consider all the images of the same object on the Internet, for instance the Eiffel Tower. It is likely different visitors take the photo

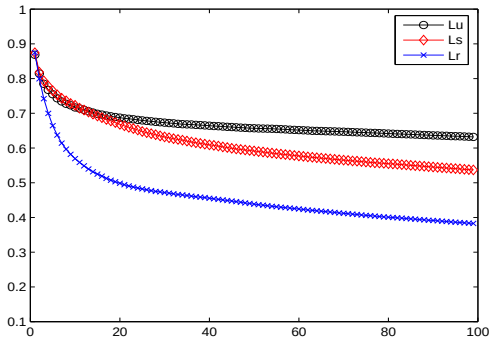

Figure 4: Typical precision-scope curves on Scene images using different graph Laplacians ($k = 10$, $\beta = 0.01$, $m = 1$, and $t = 0.1$).

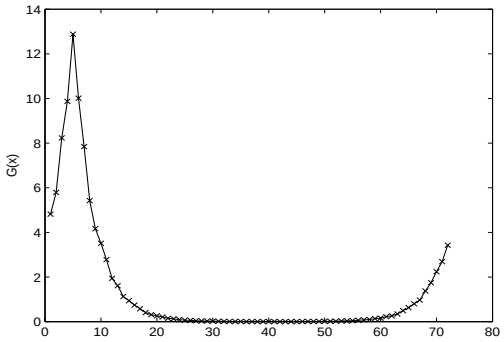

Figure 5: Green's function at $i = 5$ over the first class of COIL20.

of the tower from different angles. Then this is a similar setting as COIL20. Compared to the equal angle rotation and fixed shooting radius in COIL20, the main differences are random angle rotations and radiuses in real world. Therefore, we report more detailed experiment results on COIL20.

When $k = 1000$, $t = 1$, $\beta = 0.01$, and $m = 1$, we can obtain a perfect mAP on COIL20 data set, using either L_s or L_u. This setting is a little different from what is usually used in practice, i.e., k is small to capture the local manifold structure. On one hand a large k and a small t still can capture the local information, on the other hand, we believe a large k is helpful to the numerical stability by capturing global information in this case[10].

The Green's function with $k = 1000$, $t = 1$, $m = 1$ and $\beta = 0$ for class $Y = 1$ is shown in Figure (5)[11], where x axis is the index of the first class in COIL20. If we see it as a periodic function by connecting two ends of the interval (x axis), the Green's function or reproducing kernel function at index 5 is very similar to the reproducing kernel function over a 1D circle, e.g., [22, Chapter 2]. This empirically shows that each class of COIL20 is likely distributed on an intrinsic 1D circle. If we order the images according to this Green's function, the original rotation order can be recovered correctly. Notice that $G(x_1, x)$ on images of other classes are several order smaller, so we omit that part of the ranking function in the figure.

Numerical Issues: One key numerical issue in computing the Green's function of an iterated graph Laplacian is that we should compute the Green's function matrix G_u of L_u first, then raise the power of G_u to obtain G_u^m, instead of computing the inverse of L_u^m directly. This is due to the fact that the first several eigenvalues of L_u is very small, therefore, raising the power of L_u can easily make those eigenvalues much smaller, which increases the condition number of L_u^m dramatically.

The most expensive step to compute the Green's function of a graph Laplacian is to invert a $n \times n$ matrix, which costs $O(n^3)$. However, it is possible to take advantage of the sparsity of the problem to obtain a lower cost. Consider the ranking function of the form $f = [(\beta I + L_u)^{-1}]^m y$. For $\beta > 0$, since $(\beta I + L_u)$ is sparse when m is small, solving $(\beta I + L_u)^m f = y$ can cost as low as $O(kn)$, where k is the sparsity of the coefficient matrix. For $\beta = 0$, we need to compute L_u^{-1} (pseudoinverse), which can be rewritten as $[D(I - P)]^{-1} = (I - P)^{-1} D^{-1}$, where $P = D^{-1}W$, and $(I - P)^{-1} = \sum_{i=1}^{\infty} P^i$ can be potentially compressed and computed efficiently, see e.g., [8, Section 6.2].

[10]Notice that on other data sets a smaller k is preferred.

[11]This is the first class of COIL20, the toy duck.

6. SUMMARY

In this paper, we analyzed a popular graph Laplacian based ranking algorithm introduced in [23] from a functional analysis point of view. From the theoretical analyses and experimental results, our findings for ranking on manifolds are as follows:

1. The ranking algorithm based on L_s is sensitive to the commonly used parameter β. However, this is not the case for the unnormalized Laplacian L_u.

2. When choosing β by validation is not feasible, the pseudoinverse of the graph Laplacian ($\beta = 0$) can also give competitive results.

3. Using iterated graph Laplacian has a solid theoretical foundation and can improve ranking results in practice.

4. L_p gives competitive results, which suggests the potential value of using the "unnormalized" twice normalized graph Laplacian $\tilde{L}_u^{\alpha}$ in practice.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for helping to improve the paper.

8. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comp*, 15(6):1373–1396, 2003.

[2] M. Belkin and P. Niyogi. Convergence of Laplacian Eigenmaps. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 129–136. MIT Press, Cambridge, MA, 2007.

[3] M. Belkin and P. Niyogi. Towards a Theoretical Foundation for Laplacian-Based Manifold Methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[4] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2003.

[5] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[6] F. Chung and S.-T. Yau. Discrete Green's functions. *J. Combinatorial Theory (A)*, 91:191–214, 2000.

[7] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[8] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[9] A. Grigor'yan. Heat kernels on weighted manifolds and applications. *Cont. Math.*, 398:93–191, 2006.

[10] M. Hein. *Geometrical aspects of statistical learning theory*. PhD thesis, Wissenschaftlicher Mitarbeiter am Max-Planck-Institut für biologische Kybernetik in Tübingen in der Abteilung, 2005.

[11] M. Hein, J.-Y. Audibert, and U. V. Luxburg. Graph Laplacians and their Convergence on Random Neighborhood Graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

[12] M. Maier, M. Hein, and U. von Luxburg. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science*, 410(19):1749–1764, 2009.

[13] B. Nadler, N. Srebro, and X. Zhou. Semi-Supervised Learning with the Graph Laplacian: The Limit of Infinite Unlabelled Data. In *Twenty-Third Annual Conference on Neural Information Processing Systems*, 2009.

[14] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96, February 1996.

[15] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[16] B. D. Reddy. *Introductory Function Analysis, with Applications to Boundary Value Problems and Finite Elements*. Springer, 1997.

[17] A. Smola and R. Kondor. Kernels and regularization on graphs. In *16th Annual Conference on Learning Theory and 7th Kernel Workshop*. Springer Verlag, 2003.

[18] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, New York., 1999.

[19] M. E. Taylor. *Partial Differential Equations I: Basic Theory*. Springer, New York, 1996.

[20] U. von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[21] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Ann. Statist.*, 36(2):555–586, 2008.

[22] G. Wahba. *Spline Models for Observational Data*. Volume 59 of CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia: SIAM, 1990.

[23] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on Data Manifolds. In *Advances in Neural Information Processing Systems (NIPS) 16*, pages 169–176. MIT Press, 2004.

[24] X. Zhou and M. Belkin. Semi-supervised Learning by Higher Order Regularization. In G. Gordon, D. Dunson, and M. Dudík, editors, *The 14th International Conference on Artificial Intelligence and Statistics*, 2011.

[25] X. Zhou and N. Srebro. Error Analysis of Laplacian Eigenmaps for Semi-supervised Learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *The 14th International Conference on Artificial Intelligence and Statistics*, 2011.

[26] X. Zhu. Semi-Supervised Learning Literature Survey. Technical report, 2008. University of Wisconsin Madison, Computer Sciences TR 1530.