The Chinese University of Hong Kong
Department of Computer Science and Engineering

Ph.D. – Term Paper

Title:	Further Investigations on Heat Diffusion Models
Name:	Yang Haixuan
Student I.D.:	03499020
Contact Tel. No.:	6201-4825 Email A/C: hxyang@cse.cuhk.edu.hk
Supervisor:	Prof. Irwin King & Prof. Michael R. Lyu
Markers:	Prof. Christopher C. Yang (SEEM) & Prof. Evangeline F.Y. Young
Mode of Study:	Full-time
Submission Date:	November 30, 2006
Term:	6
Fields:	
Presentation Date:	will be confirmed
Time:	will be confirmed
Venue:	will be confirmed

Further Investigations on Heat Diffusion Models

Abstract

In this paper, we investigate the Heat Diffusion Model further in three aspects: its applications in the ranking algorithm on the Web, other graph constructions for its graph inputs, and the volume effects.

In applications on the Web, we propose a ranking algorithm DiffusionRank. On one hand, DiffusionRank is motivated by the heat diffusion phenomena, which can be connected to Web ranking because the activities flow on Web can be imagined as heat flow, the link from a page to another can be imagined as the pipe of an air-conditioner, and heat flow can embody the structure of the underlying Web graph. On the other hand, DiffusionRank is also motivated by the manipulation problem: the PageRank scores of Web pages can be manipulated while the PageRank algorithm has proven to be very effective for ranking Web pages. DiffusionRank is proposed to handle the manipulation problem and to cast a new insight on the Web structure. Theoretically we show that DiffusionRank can serve as a generalization of PageRank when the heat diffusion coefficient γ *tends to infinity. In such a case* $1/\gamma = 0$*, DiffusionRank (PageRank) has the lower ability of anti-manipulation. When the heat diffusion coefficient is set to be zero, DiffusionRank has the highest ability of anti-manipulation, but in such a case* $\gamma = 0$*, the web structure is completely ignored. Consequently, the heat diffusion coefficient is an interesting factor that can control the balance between the extent that we want to model the original Web and the extent that we want to reduce the effect of manipulation. It is found empirically that, when* $\gamma = 1$*, DiffusionRank has a Penicillin-like effect on the link manipulation. Moreover, DiffusionRank can be employed to find group-to-group relations on the Web, can divide the Web graph into several parts, and can find link communities. Experimental results show that the DiffusionRank algorithm achieves the above mentioned advantages.*

In our previous work, a classifier is proposed based on a heat diffusion model on a finite K *nearest neighbor (KNN) graph. To broaden the scope of the previous work, we separate the graph construction procedure and classifier construction procedure by proposing Graph-based Heat Diffusion Classifiers (G-HDC). This separation allows various graph inputs. Besides traditional KNN graph, we propose two other candidate graph inputs for G-HDC: a graph with the shortest edges whose number is the same as the KNN graph, and the Minimum Spanning Tree. These two graphs can be also considered as the representation of the underlying geometry, and the heat diffusion model on them can be considered as the approximation to the way that heat flows on a geometric structure.*

Three graph construction methods lead to three classifiers when applied to the G-HDC. Experiments show that the two classifiers with the two proposed graph inputs can be considered as two other candidate classifiers, which enrich the family of heat diffusion classifiers.

We also propose the volume-based heat diffusion model (VHDM). VHDM is based on a graph model that takes account of unseen input data, giving rise to a closed-form expression. We give related background on heat diffusion models, the theoretical framework of our model, detailed analysis of the formulation, and a VHDC classifier as an example application. Experiments show that VHDC performs uniformly better than the Parzen Window Approach (PWA) and KNN in prediction accuracy, as we expected, and is competitive with recently proposed transductive learning algorithms. The enhanced performance of VHDC over the form without the volume consideration shows the necessity of introducing the volume representation.

Contents

Chapter 1

Introduction

Some successful applications of heat kernels have been reported recently. In [2], a nonlinear dimensionality reduction algorithm was proposed based on a local heat kernel approximation and a graph heat kernel approximation (i.e., the graph Laplacian). For given k points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in an $m-$dimensional space R^m, an adjacency graph was constructed first by either $\epsilon-$neighborhoods or K nearest neighbors. Then the local heat kernel approximation $e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/t}$ was employed to construct the weight of the neighborhood graph. To achieve efficient dimensionality reduction, the minimization of the following smoothness penalty on the graph: $\mathbf{f}^t L\mathbf{f} = \sum_{i,j=1}^k (f(\mathbf{x}_i) - f(\mathbf{x}_j))$ was converted to the generalized eigenvector problem:

$$L\mathbf{f} = \lambda D\mathbf{f},$$

where $L = D - W$ is the graph Laplacian, $W = (W_{ij})$, $W_{ij} = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/t}$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are adjacent and $w_{ij} = 0$ otherwise, and D is a diagonal degree matrix given by $D_{ii} = \sum_i W_{ij}$. Finally the eigenvectors corresponding to the smallest eigenvalues except zero are used for dimensionality reduction.

In [19], a discrete diffusion kernel on graphs and other discrete input space was proposed. When it was applied to a large margin classifier, good performance for categorical data was demonstrated by employing the simple diffusion kernel on the hypercube. The proposed heat diffusion kernel takes the form of $e^{\beta H}$, where $H = (H_{ij})$,

$$H_{ij} = \begin{cases} 1, & i \sim j, \\ -d_i, & i = j, \\ 0, & \text{otherwise}. \end{cases}$$

In [20], a general framework was proposed. The key idea was to begin with a statistical family that was natural for the data being analyzed, and to represent data as points on the statistical manifold associated with the Fisher information metric of this family. The investigation of the heat equation with respect to the Riemannian structure, given by the Fisher metric, led to a family of kernels, which generalized the familiar Gaussian kernel for Euclidean space. When applied to the text classification, where the natural statistical family was the multinomial, a closed form approximation to the heat kernel for a multinomial family was proposed, which yielded significant improvements over the use of Gaussian or linear kernels. The proposed closed form approximation heat kernel on the $m-$dimensional multinomial was given by

$$K(\theta, \theta') = (4\pi t)^{-\frac{m}{2}} e^{-\frac{1}{t}\arccos^2(\sqrt{\theta}\cdot\sqrt{\theta})}.$$

These successful applications of the heat kernel motivate us to investigate the heat equation, since the heat kernel can be explained as a special solution to the heat equation given a special initial condition called the delta function $\delta(\mathbf{x}-\mathbf{y})$. More specifically, $\delta(\mathbf{x}-\mathbf{y})$ describes a unit heat source at position $\mathbf{y}$ with no heat in other positions, in other words, $\delta(\mathbf{x}-\mathbf{y}) = 0$ for $\mathbf{x} \neq \mathbf{y}$ and $\int_{-\infty}^{+\infty} \delta(\mathbf{x}-\mathbf{y}) d\mathbf{x} = 1$. If we let $f_0(\mathbf{x}, 0) = \delta(\mathbf{x}-\mathbf{y})$, then the heat kernel $K_t(\mathbf{x}, \mathbf{y})$ is a solution to the following differential equation on a manifold $\mathcal{M}$:

$$\left\{ \begin{array}{lcl} \frac{\partial f}{\partial t} - \Delta f & = & 0, \\ f(\mathbf{x}, 0) & = & f_0(\mathbf{x}), \end{array} \right. \tag{1.1}$$

where $f(\mathbf{x}, t)$ is the temperature at location $\mathbf{x}$ at time t, beginning with an initial distribution $f_0(\mathbf{x})$ at time zero, and Δf is the *Laplace-Beltrami operator* on a function f. Eq. (2.3) describes the heat flow throughout a geometric manifold with initial conditions. In local coordinates, Δf is given by

$$\Delta f = \frac{1}{\sqrt{\det g}} \sum_j \frac{\partial}{\partial x_j} \left(\sum_i g^{ij} \sqrt{\det g} \frac{\partial f}{\partial x_i} \right)$$

[20]. When the underlying manifold is the familiar $m-$dimensional Euclidean Space, Δf is simplified as $\sum_i \frac{\partial^2 f}{\partial x_i^2}$, and the heat kernel takes the RBF form

$$K_t(\mathbf{x}, \mathbf{y}) = (4\pi t)^{-\frac{m}{2}} e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{4t}}. \tag{1.2}$$

It is therefore observed that the RBF kernel is a special case of the heat kernel (when the underlying manifold is the Euclidean space). The previous researches have shown that the heat kernel is a useful tool. However, the special setting of the initial condition that results in the heat kernel in the heat equation may narrow the scope of the possible applications of Eq. (2.3). In this paper we consider how to construct a classifier directly by employing the solution to the heat equation on a graph in a wider setting of the initial condition. The heat equation on a graph is considered as an approximation to Eq. (2.3).

In [34], we proposed the classifier HDC (the version without considering the volume effect), and proved the following two claims: *KNN* can be considered as a special case of *HDC* (when $\beta \to +\infty$, $N = 1$, and γ is small); and when the window function is a multivariate normal kernel, the Parzen Window Approach [3] can be considered as a special case of *HDC* (when $K = n - 1$, and γ is small). As a result, *KNN* and Parzen Window Approach are special cases of *VHDC*, as *VHDC* generalizes *HDC* (let $V(j) = 1$). In [37], we discussed various graph inputs for *HDC*.

In Chapter 2, we consider the application of the Heat Diffusion Model on the Web.

Different from the work in [34], in Chapter 3, we separate the graph construction procedure and classifier construction procedure by proposing Graph-based Heat Diffusion Classifiers (*G-HDC*). Besides, we consider two other candidate input graphs for *G-HDC* in this paper. This results in two novel classifiers called SKNN-based Heat Diffusion Classifier (*SKNN-HDC*) and MST-based Heat Diffusion Classifier (*MST-HDC*). Moreover, more interpretation and justification are provided to make the Heat Diffusion Model more convincing.

Along another aspect, Different from the work in [34], we consider the volume effects on the Heat Diffusion Model in Chapter 4.

Chapter 2

DiffusionRank: A Possible Penicillin for Web Spamming

2.1 Preliminaries

While the *PageRank* algorithm [24] has proven to be very effective for ranking Web pages, inaccurate *PageRank* results are induced because of two problems: the incomplete information about the Web structure, and the manipulated web pages by people for commercial interests.

The first problem is considered in [35, 36] by predicting the Web Structure as a random graph. In this paper we aim to handle the second problem: the manipulation problem.

The manipulation problem is also called the Web spam, which refers to hyperlinked pages on the World Wide Web that are created with the intention of misleading engines [12]. It is reported in [23] that approximately 70% of all pages in the .biz domain are spam and that about 35% of the pages in the .us domain belong to spam category. The reason for the increasing amount of Web spam is explained in [23]: some web site operators try to influence the positioning of their pages within search results because of the large fraction of web traffic originating from searches and the high potential monetary value of this traffic.

From the viewpoint of the Web site operators who want to increase the ranking value of a particular page for search engines, two methods are being used widely [23, 12]:

- **Link Stuffing.** The creation of extraneous pages which link to a target page. Using link stuffing, web sites can increase the desirability of target pages to search engines using link-based ranking.

- **Keyword Stuffing.** The creation of thousands of popular keywords to a target page, often making the text invisible to humans through ingenious use of color schemes. A search engine will then index the extra keywords, and return the manipulated page as an answer to queries that contain some of the keyword.

From the viewpoint of the search engine managers, the Web spam is very harmful to the users' evaluations and thus their preference to choosing search engines because people believe that a good search engine should not return irrelevant or low-quality results. There are two methods being employed to combat the Web spam:

- **Machine Learning.** It employed to handle the keyword stuffing. To successfully apply machine learning method, we need to dig out some useful features for Web pages, to mark part of the Web pages as either spam or non-spam, then to apply a supervised learning technique to mark other pages. For example in [23], the authors proposed to use such features as number of words in the page, number of words in the page title, average length of words, amount of anchor text, fraction of visible content, compressibility, fraction of page drawn from globally popular words, and so on. Then the C4.5 decision tree is applied on the data set. Another example is the RankNet, proposed in [5] and investigated further [25]. Although the authors did not claim that RankNet can be employed as a method for spam detection, it has the potential to fulfill such a task if appropriate features are given.

- **Link Analysis.** It is employed to handle the link stuffing. One example is the *TrustRank* [12], a link-based method, in which the link structure is utilized so that human labelled trusted pages can propagate their trust scores trough their links. This paper focuses on the link-based method.

The rest of the materials of this chapter are organized as follows. In the next section, we give a brief literature review on various related ranking techniques. In Section 2.3, we propose the Heat Diffusion Model (HDM) on various cases. In Section 2.5, we describe the data sets that we worked on and the experimental results.

2.2 Literature Review

Although the importance of a Web page is determined by either the textual content of pages or the hyperlink structure or both. As in previous work [11, 16, 15], we focus on ranking methods solely determined by hyperlink structure of the Web graph.

All the mentioned ranking algorithms are established on a graph. For our convenience, we first give some notations. We denote a static graph by $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{(v_i, v_j) \mid \text{there is an edge from } v_i \text{ to } v_j\}$ is the set of all edges. I_i and d_i denote the in-degree and the out-degree of page i respectively.

2.2.1 PageRank

The importance of a Web page is an inherently subjective matter, which depends on the reader interests, knowledge and attitudes [24]. However, the average importance of all readers can be considered as an objective matter. *PageRank* tries to find such average importance based on the Web link structure, which is considered to contain a large amount of statistical data. The intuition behind *PageRank* is that it uses information external to the Web pages themselves—their in-links, and that in-links from "important" pages are more significant than in-links from average pages. Formally presented in [9], the Web is modelled by a directed graph G in the *PageRank* algorithms, and the rank or "importance" x_i for page $v_i \in V$ is defined recursively in terms of pages which point to it:

$$x_i = \sum_{(j,i) \in E} a_{ij} x_j, \tag{2.1}$$

where a_{ij} is assumed to be $1/d_j$ if there is a link from j to i, and 0 otherwise. Or in matrix terms, $\mathbf{x} = \mathbf{A}\mathbf{x}$. When the concept of "random jump" is introduced, the matrix form in Eq. (2.1) is changed to

$$\mathbf{x} = [(1-\alpha)\mathbf{g}\mathbf{1}^T + \alpha \mathbf{A}]\mathbf{x}, \tag{2.2}$$

where the parameter α is the probability of following the actual link from a page, $(1-\alpha)$ is the probability of taking a "random jump", and $\mathbf{g}$ is a stochastic vector (i.e. $\mathbf{1}^T\mathbf{g} = \mathbf{1}$). Typically, $\alpha = 0.85$ and $\mathbf{g} = \frac{1}{n}\mathbf{1}$, where $\mathbf{1}$ is the vector of all ones.

2.2.2 TrustRank

TrustRank [12] is composed of two parts. The first part is the seed selection algorithm, in which the inverse *PageRank* was proposed to help an expert of determining a good node. The second part is to utilize the biased *PageRank*, in which the stochastic distribution $\mathbf{g}$ is set to be shared by all the trusted pages found in the first part. Moreover, the initial input of $\mathbf{x}$ is also set to be $\mathbf{g}$. The justification for the inverse *PageRank* and the solid experiments support its advantage in combating the Web spam.

2.2.3 Manifold Ranking

In [41], the idea of ranking on the data manifolds was proposed. The data points represented as vectors in Euclidean space are considered to be drawn from a manifold. From the data points on such a manifold, an undirected weighted graph is created, then the weight matrix $\mathbf{W}$ is given by the Gaussian Kernel smoothing. Then we can show its next two steps:

Step 1. Normalize $\mathbf{W}$ by $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ in which $\mathbf{D}$ is the diagonal matrix with $(i, i)-$element equal to the sum of the $i-$th row of $\mathbf{W}$.

Step 2. Iterate $\mathbf{f}(t + 1) = \alpha\mathbf{S}\mathbf{f}(t) + (1 - \alpha)\mathbf{y}$ until convergence, where α is a parameter in $[0, 1)$, the $i - th$ element in $\mathbf{y}$ is set to be 1 if the corresponding data is a query, and zero otherwise.

2.2.4 Heat Diffusion

Heat diffusion is a physical phenomena. In a medium, heat always flow from position with high temperature to position with low temperature. Heat kernel is used to describe the amount of heat that one point receives from another point.

Recently, the idea of heat kernel on a manifold is borrowed in applications such as dimension reduction [2] and classification [20, **?**, 34]. In these work, the input data is considered to lie in a special structure. In [20], the authors apply the heat kernel on a continuous manifold to text classifications. In [**?**], the authors apply the heat kernel on discrete structures to categorical data classifications. In [2], the authors employ the heat kernel to construct weights of a neighborhood graph and reduce the dimension of the underlying data. In [34], the authors represent the underlying geometry by a finite neighborhood directed graph, on which a heat kernel is established, and construct classifiers by imitating the heat diffusion on the graph.

All the above topics are related to our work. The readers can find that our model is a generalization of *PageRank* in order to resist Web manipulation, that we inherit the first part of *TrustRank*, that we actually are ranking on the manifold, and that heat diffusion is a main scheme in this paper.

2.3 Heat Diffusion Model

Heat diffusion provides us with another perspective about how we can view the Web and also a way to calculate ranking values. In this paper, the Web pages are considered to be drawn from an unknown manifold, and the link structure forms a directed graph, which is considered as an approximation to the unknown manifold. The heat kernel established on the Web graph is considered as the representation of relationship between Web pages. Then the following problems arise naturally:

1. How can we describe heat diffusion behavior on a graph instead of on a manifold?

2. How can we model the page relationship by the diffusion behavior?

To address Problem 2, we propose a novel *DiffusionRank* method in Section 2.4 using the heat kernel. To address Problem 1, we investigate the HDM in this section. First we show our motivations for the proposed models.

2.3.1 Motivation

There are two points to explain that *PageRank* is susceptible to web spam.

- **Over-democratic.** There is a belief behind *PageRank*—all pages are born equal. This can be seen from the equal voting ability of one page: the sum of each column is equal to one. This equal voting ability of all pages gives the chance for a Web site operator to increase a manipulated page by creating a large number of new pages pointing to this page since all the newly created pages can obtain an equal voting right.
- **Input-independent.** For any given non-zero initial input, the iteration will converge to the same stable distribution corresponding to the maximum eigenvalue 1 of the transition matrix. This input-independent property makes it impossible to set a special initial input (larger values for trusted pages and less values even negative values for spam pages) to avoid web spam.

The input-independent feature of *PageRank* can be further explained as follows. $\mathbf{P} = [(1-\alpha)\mathbf{g}\mathbf{1}^T + \alpha\mathbf{A}]$ is a positive stochastic matrix if $\mathbf{g}$ is set to be a positive stochastic vector (the uniform distribution is one of such settings), and so 1 is its largest eigenvalue and no other eigenvalue whose absolute value is equal to 1, which is guaranteed by the Perron Theorem [22]. Let $\mathbf{y}$ is the eigenvector corresponding to 1, then we have $\mathbf{P}\mathbf{y} = \mathbf{y}$. Let $\{\mathbf{x}_k\}$ is the sequence generated from the iterations $\mathbf{x}_{k+1} = \mathbf{P}\mathbf{x}_k$, and $\mathbf{x}_0$ is the initial input. If $\{\mathbf{x}_k\}$ converges to $\mathbf{x}$, then $\mathbf{x}_{k+1} = \mathbf{P}\mathbf{x}_k$ implies that $\mathbf{x}$ must satisfy $\mathbf{P}\mathbf{x} = \mathbf{x}$. Since 1 is the only maximum eigenvalue, we have $\mathbf{x} = c\mathbf{y}$ where c is a constant, and if both $\mathbf{x}$ and $\mathbf{y}$ are normalized by their sums, then $c = 1$. The above discussions show that *PageRank* is independent of the initial input $\mathbf{x}_0$.

We observe that the heat diffusion model is a natural way to avoid the over-democratic and input-independent feature of *PageRank*. Since heat always flows from a position with higher temperatures to one with lower temperatures, points are not equal as some points are born with high temperatures while others are born with low temperatures. On the other hand, different initial temperature distributions will give rise to different temperature distributions after a fixed time period.

Based on these considerations, we propose the novel *DiffusionRank*. This ranking algorithm is also motivated by the viewpoint for the Web structure. We view all the Web pages as points drawn from a highly complex geometric structure, like a manifold in a high dimensional space. On a manifold, heat can flow from one point to another through the underlying geometric structure in a given time period. Different geometric structures determine different heat diffusion behaviors, and conversely the diffusion behavior can reflect the geometric structure. More specifically, on the manifold, the heat flows from one point to another point, and in a given time period, if one point x receives a large amount of heat from another point y, we can say x and y are connected well, and thus x and y have a high similarity in the sense of a high mutual connection.

In the following, we first show the HDM on a manifold, which is the origin of HDM, but can not be employed to the World Wide Web directly, and so is considered as the ideal case. To connect the ideal case and the practical case, we then establish HDM on a graph as an intermediate case. To model the real world problem, we further build HDM on a random graph as a practical case. Finally we demonstrate the *DiffusionRank* which is derived from the HDM on a random graph.

We note that on a point with unit mass, the temperature and the heat of this point are equivalent, and these two terms are interchangeable in this paper.

2.3.2 Heat Flow On a Known Manifold

If the underlying manifold is known, the heat flow throughout a geometric manifold with initial conditions can be described by the following second order differential equation:

$$\frac{\partial \mathbf{f}(\mathbf{x},t)}{\partial t} - \Delta \mathbf{f}(\mathbf{x},t) \quad = \quad 0, \tag{2.3}$$

where $\mathbf{f}(\mathbf{x},t)$ is the heat at location $\mathbf{x}$ at time t, and $\Delta\mathbf{f}$ is the *Laplace-Beltrami operator* on a function $\mathbf{f}$. The heat diffusion kernel $\mathbf{K}_t(\mathbf{x},\mathbf{y})$ is a special solution to the heat equation with a special initial condition—a unit heat source at position $\mathbf{y}$ when there is no heat in other positions. Based on this, the heat kernel $\mathbf{K}_t(\mathbf{x},\mathbf{y})$ describes the heat distribution at time t diffusing from the initial unit heat source at position $\mathbf{y}$, and thus describes the connectivity (which is considered as a kind of similarity) between $\mathbf{x}$ and $\mathbf{y}$.

However, it is very difficult to restore the World Wide Web as a regular geometry with a known dimension; even the manifold, in which the Web pages lie in, is known, it is very difficult to find the heat kernel $\mathbf{K}_t(\mathbf{x},\mathbf{y})$, which involves solving Eq. (2.3) with the delta function as the initial condition. This motivates us to investigate the heat flow on a graph: the graph is considered as an approximation to the underlying manifold, and so the heat flow on the graph is considered as an approximation to the heat flow on the manifold.

2.3.3 On an Undirected Graph

On an undirected graph G, the edge (v_i, v_j) is considered as a pipe that connects to nodes v_i and v_j. The value $f_i(t)$ describes the heat at node v_i at time t, beginning from an initial distribution of heat given by $f_i(0)$ at time zero. $\mathbf{f}(t)$ (**f**(0)) denotes the vector consisting of $f_i(t)$ ($f_i(0)$).

We establish our model as follows. Suppose, at time t, each node i receives $M(i,j,t,\Delta t)$ amount of heat from its neighbor j during a period of Δt. The heat $M(i,j,t,\Delta t)$ should be proportional to the time period Δt and the heat difference $f_j(t) - f_i(t)$. Moreover, the heat flows from node j to node i through the pipe that connects nodes i and j. Based on this consideration, we assume that $M(i, j, t, \Delta t) = \gamma(f_j(t) - f_i(t))\Delta t$. As a result, the heat difference at node i between time $t + \Delta t$ and time t will be equal to the sum of the heat that it receives from all its neighbors. This is formulated as

$$f_i(t + \Delta t) - f_i(t) = \sum_{j:(j,i)\in E} \gamma(f_j(t) - f_i(t))\Delta t. \tag{2.4}$$

To find a closed form solution to Eq. (2.4), we express it in a matrix form:

$$\frac{\mathbf{f}(t + \Delta t) - \mathbf{f}(t)}{\Delta t} = \gamma \mathbf{H}\mathbf{f}(t), \tag{2.5}$$

where $\mathbf{H} = (H_{ij})$, and

$$H_{ij} = \begin{cases} -d(v_j), & j = i, \\ 1, & (v_j, v_i) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

where $d(v)$ denotes the degree of the node v. In the limit $\Delta t \to 0$, Eq. (2.5) becomes

$$\frac{d}{dt}\mathbf{f}(t) = \gamma \mathbf{H}\mathbf{f}(t). \tag{2.7}$$

Solving Eq. (2.7), we get $\mathbf{f}(t) = e^{\gamma t \mathbf{H}}\mathbf{f}(0)$, especially we have

$$\mathbf{f}(1) = e^{\gamma \mathbf{H}}\mathbf{f}(0), \tag{2.8}$$

where $e^{\gamma H}$ is defined as

$$e^{\gamma \mathbf{H}} = \mathbf{I} + \gamma \mathbf{H} + \frac{\gamma^2}{2!}\mathbf{H}^2 + \frac{\gamma^3}{3!}\mathbf{H}^3 + \cdots . \tag{2.9}$$

The matrix $e^{\gamma \mathbf{H}}$ is called as *Continuous Diffusion Kernel* in the sense that the heat diffusion process continues infinitely many times after the nodes begin to diffuse their heat to their subsequent nodes.

2.3.4 On a Directed Graph

The above heat diffusion model must be modified to fit the situation where the links between Web pages are directed. On one Web page, when the page-maker creates a link (a, b) to another page b, he actually forces the energy flow, for example, people's click-through activities, to that page, and so there is added energy imposed on the link. As a result, heat flows in a one-way manner, only from a to b, not from b to a. Based on such consideration, we modified the heat diffusion model on an undirected graph as follows.

On a directed graph G, the edge (v_i, v_j) is considered as a pipe that connects to nodes v_i and v_j, and is forced by added energy such that heat flows only from v_i to v_j.

Suppose, at time t, each node v_i receives $RH(i,j,t,\Delta t)$ amount of heat from its antecedent v_j during a period of Δt. We have three assumptions:

1. The heat $RH(i,j,t,\Delta t)$ should be proportional to the time period Δt.
2. The heat $RH(i,j,t,\Delta t)$ should be proportional to the the heat at node v_j.
3. The heat $RH(i,j,t,\Delta t)$ is zero if there is no link from v_j to v_i.

As a result, v_i will receive $\sum_{j:(v_j,v_i)\in E} \sigma_j f_j(t)\Delta t$ amount of heat from all its antecedents.

On the other hand, node v_i diffuses $DH(i,t,\Delta t)$ amount of heat to its subsequent nodes. We assume that

1. The heat $DH(i,t,\Delta t)$ should be proportional to the time period Δt.
2. The heat $DH(i,t,\Delta t)$ should be proportional to the the heat at node v_i.
3. Each node has the same ability of diffusing heat. This fits the intuition that a Web surfer only has one choice to find the next page that he wants to browse.
4. The heat $DH(i,t,\Delta t)$ should be uniformly distributed to its subsequent nodes. The real situation is more complex than what we assume, but we have to take this simple assumption in order to make our model concise.

As a result, node v_i will diffuse $\gamma f_i(t)\Delta t/d_i$ amount of heat to any of its subsequent nodes, and each of its subsequent node should receive $\gamma f_i(t)\Delta t/d_i$ amount of heat. Therefore $\sigma_j = \gamma/d_j$.

To sum up, the heat difference at node v_i between time $t+\Delta t$ and time t will be equal to the sum of the heat that it receives, deducted by what it diffuses. This is formulated as

$$f_i(t+\Delta t) - f_i(t) = -\gamma f_i(t)\Delta t + \sum_{j:(v_j,v_i)\in E} \gamma/d_j f_j(t)\Delta t. \tag{2.10}$$

Similarly, we obtain

$$\mathbf{f}(1) = e^{\gamma \mathbf{H}}\mathbf{f}(0), \tag{2.11}$$

where $\mathbf{H} = (H_{ij})$, and

$$H_{ij} = \begin{cases} -1, & j = i, \\ 1/d_j, & (v_j, v_i) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

To find another closed form solution to Eq. (2.10), we let $t = 0$, Eq. (2.11) can then be rewritten as

$$\mathbf{f}(\Delta t) = (\mathbf{I} + \gamma \Delta t \mathbf{H})\mathbf{f}(0). \tag{2.13}$$

Assume $N = 1/\Delta t$ is an integer, we then have

$$\begin{aligned} \mathbf{f}(1) &= \mathbf{f}(N \cdot \Delta t) \\ &= (\mathbf{I} + \gamma \Delta t \mathbf{H})\mathbf{f}((N-1) \cdot \Delta t) \\ &\vdots \\ &= (\mathbf{I} + \gamma \Delta t \mathbf{H})^N \mathbf{f}(0) \\ &= (\mathbf{I} + \tfrac{\gamma}{N}\mathbf{H})^N \mathbf{f}(0). \end{aligned} \tag{2.14}$$

Eq. (2.14) is one closed form solution to Eq. (2.10) in the setting of discrete heat diffusion, where it describes the $N-$step heat distribution at a time period of Δt from time 0 to time 1. The matrix $(\mathbf{I} + \frac{\gamma}{N}\mathbf{H})^N$ is called as *Discrete Diffusion Kernel* in the sense that the heat diffusion process stops after a number of steps, and in each step, nodes diffuse their heat only to their subsequent nodes. In Figure 2.1, we illustrate the heat flow on a graph. There are

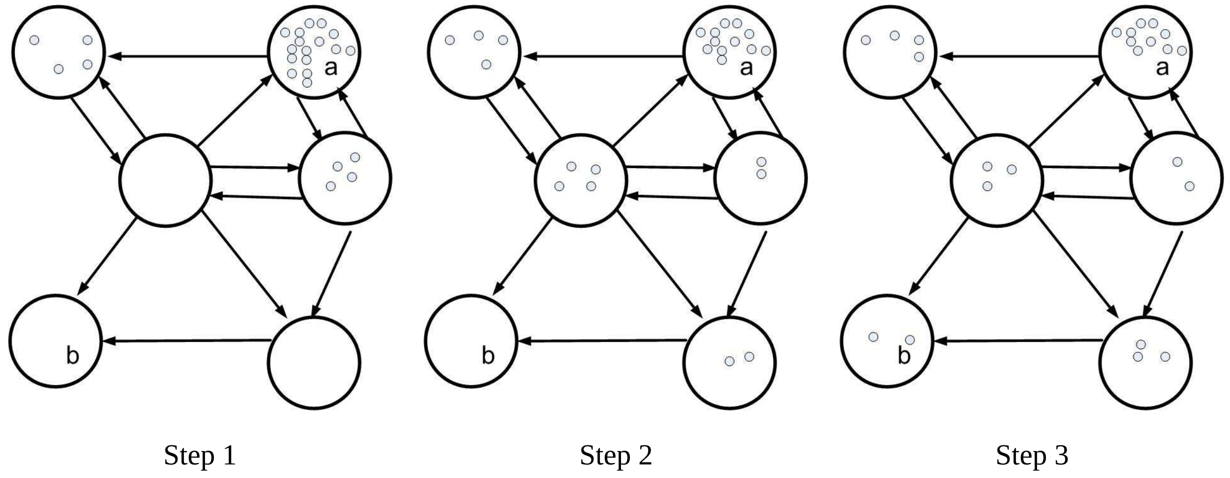


Figure 2.1. Illustration on Heat Diffusion

six nodes in the graph, and there are links between them. We want to show how the heat diffuses from node a to node b. Initially there is some amount of heat only in node a, at Step 1, heat diffuses from a to its two subsequent nodes along the links. In Step 2, the heat diffuses further from nodes with higher heat to nodes with lower heat. At Step 3, node b receives heat from its two antecedents. The heat distribution in Step 3 can reflect the relationship between node a and other nodes, which is caused by the graph structure.

2.3.5 On a Random Directed Graph

For real world applications, the heat diffusion model is not enough because some edges exist in a random way. This can be seen in two viewpoints. The first one is that in Eq. (2.2), the Web graph is actually modelled as a random graph, there is a edge from node v_i to node v_j with a probability of $(1-\alpha)g_j$ (see the item $(1-\alpha)\mathbf{g}\mathbf{1}^T$). The second one is that in [35, 36], the Web structure is predicted as a random graph by its earlier structure found by a crawler. For these reasons, the model would become more flexible if we extend it to random graphs. The definition of a random graph is given below.

Definition 1: A random graph $RG = (V, \mathbf{P} = (p_{ij}))$ is defined as a graph with a vertex set V in which the edges are chosen independently, and for $1 \le i, j \le |V|$ the probability of (v_i, v_j) being an edge is exactly p_{ij}. The original definition of random graphs in [4], is slightly changed to consider the situation of directed graphs. Note that every static graph can be considered as a special random graph in the sense that p_{ij} takes only 0 or 1.

On a random graph $RG = (V, \mathbf{P})$, where $\mathbf{P} = (p_{ij})$ is the probability of the edge (v_i, v_j) exists. In such a random graph, the expected heat difference at node i between time $t + \Delta t$ and time t will be equal to the sum of the expected heat that it receives from all its antecedents, deducted by the expected heat that it diffuses.

Since the probability of the link (v_j, v_i) is p_{ji}, the expected heat flow from node j to node i should be multiplied by p_{ji}, and so we have

$$f_i(t+\Delta t) - f_i(t) = -\gamma\ f_i(t)\,\Delta t + \sum_{j:(v_j,v_i)\in E} \frac{\gamma p_{ji} f_j(t)\Delta t}{RD^+(v_j)}, \tag{2.15}$$

where $RD^+(v_i)$ is the expected out-degree of node v_i, it is defined as $\sum_k p_{ik}$. Similarly we have

$$\mathbf{f}(1) = e^{\gamma \mathbf{R}}\mathbf{f}(0), \tag{2.16}$$

where $\mathbf{R} = (R_{ij})$, and

$$R_{ij} = \begin{cases} -1, & j = i; \\ p_{ji}/RD^+(v_j), & j \neq i. \end{cases} \tag{2.17}$$

By assuming that $N = 1/\Delta t$ is an integer, we also have

$$\mathbf{f}(1) = (\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N \mathbf{f}(0). \tag{2.18}$$

The matrix $(\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N$ in Eq. (2.18) and matrix $e^{\gamma\mathbf{R}}$ in Eq. (2.16) are called *Discrete Diffusion Kernel* on the random graph and the *Continuous Diffusion Kernel* on the random graph respectively. Based on the Heat Diffusion Models and their solutions, we can establish the *DiffusionRank* on undirected graphs, directed graphs, and random graphs. In next section, we mainly focus on random graphs.

2.4 DiffusionRank

For a random graph, the matrix $(\mathbf{I}+\frac{\gamma}{N}\mathbf{R})^N$ or $e^{\gamma\mathbf{R}}$ can measure the similarity relationship between nodes. Let $f_i(0)=1, f_j(0)=0$ if $j\neq i$, then the vector $\mathbf{f}(0)$ represent the unit heat at node v_i while all other nodes has zero heat. For such $\mathbf{f}(0)$ in a random graph, we can find the heat distribution at time 1 by using Eq. (2.16) or Eq. (2.18). The heat distribution is exactly the $i-$th row of the matrix of $(\mathbf{I}+\frac{\gamma}{N}\mathbf{R})^N$ or $e^{\gamma\mathbf{R}}$. So the ith-row jth-column element h_{ij} in the matrix $(\mathbf{I}+\gamma\Delta t\mathbf{R})^N$ or $e^{\gamma\mathbf{R}}$ means the amount of heat that v_i can receive from v_j from time 0 to 1. Thus the value h_{ij} can be used to measure the similarity from v_j to v_i. For a static graph, similarly the matrix $(I+\frac{\gamma}{N}\mathbf{H})^N$ or $e^{\gamma\mathbf{H}}$ can measure the similarity relationship between nodes.

The intuition behind is that the amount $h(i,j)$ of heat that a page v_i receives from a unit heat in a page v_j in a unit time embodies the extent of the link connections from page v_j to page v_i. Roughly speaking, when there are more uncrossed paths from v_j to v_i, v_i will receive more heat from v_j; when the path length from v_j to v_i is shorter, v_i will receive more heat from v_j; when the pipe connecting v_j and v_i is wide, the heat will flow quickly. The final heat that v_i receives will depend on various paths from v_j to v_i, their length and width.

2.4.1 Algorithm

For the ranking task, we adopt the heat kernel on a random graph. Formally the *DiffusionRank* is described as follows.

In algorithm 1, the element U_{ij} in the inverse transition matrix **U** is defined to be $1/I_j$ if there is a link from i to j, and zero otherwise. This trusted pages selection procedure by inverse *PageRank* is completely borrowed from *TrustRank* [12] except for a fix number of the size of the trusted set. Although the inverse *PageRank* is not perfect in its ability of determining the maximum coverage , it is appealing because of its polynomial execution time and its reasonable intuition—we actually inverse the original link when we try to build the seed set from those pages that point to many pages that in turn point to many pages and so on. In the algorithm, the underlying random graph is set as $\mathbf{P}=\alpha_B\cdot\mathbf{A}+(1-\alpha_B)\cdot\frac{1}{n}\cdot\mathbf{1}_{n\times n}$, which is induced by the Web graph. As a result, $\mathbf{R}=-\mathbf{I}+\mathbf{P}$.

2.4.2 Advantages

Next we show the four advantages for *DiffusionRank*.

Two closed forms

First, its solutions have two forms, both of which are closed form. One takes the discrete form, and has the advantage of fast computing while the other takes the continuous form, and has the advantage of being easily analyzed in theoretical aspects. The theoretical advantage has been shown in the proof of theorem in next section.

Group-group relations

Second, it can be naturally employed to detect the group-group relation. For example, if group 2 contains pages ($j_1, j_2, \ldots, j_s$), and if group 1 contains pages ($i_1, i_2, \ldots, i_t$), then the sum $\sum_{u,v} h_{i_u,j_v}$ denotes the total amounts of heat that group 1 receives from group 2, where h_{i_u,j_v} is the i_u-th row j_v-th column element of the heat kernel. More specifically, we first need to set $f(0)$ for such an application as follows.

In $\mathbf{f}(0) = (f_1(0), f_2(0), \ldots, f_n(0))^T$, if $i \in \{j_1, j_2, \ldots, j_s\}$, then $f_i(0) = 1$, and 0 otherwise. Then we employ Eq. (2.16) to calculate $\mathbf{f}(1) = (f_1(1), f_2(1), \ldots, f_n(1))^T$, finally we sum those $f_j(1)$ where $j \in \{i_1, i_2, \ldots, i_t\}$.

Figure 2.2 (a) shows the results generated by the *DiffusionRank*. We consider five groups—five departments in our Engineering Faculty: CSE, MAE, EE, IE, and SE. γ is set to be 1, the numbers in Figure 2.2 (a) are the amount of heat that they diffuse to each other. These results are normalized by the total number of each group, and the edges are ignored if the values are less than 0.000001. The group-to-group relations are therefore detected, for example, we can see that the most strong overall tie is from EE to IE. While it is a natural application for *DiffusionRank* because of the easy interpretation by the amount heat from one group to another group, it is difficult to apply other ranking techniques to such an application because they lack such a physical meaning.

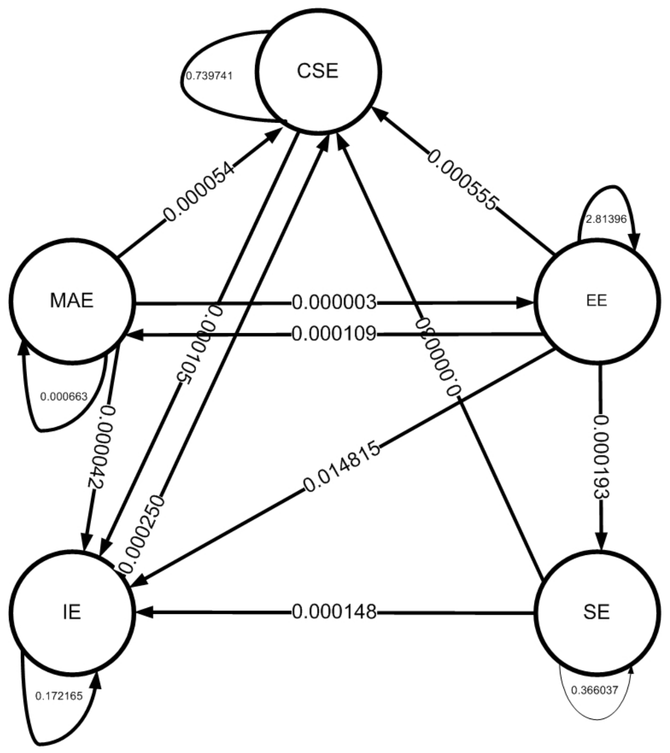
Graph cut

Third, it can be used to partition the Web graph into several parts. A quick example is shown below.

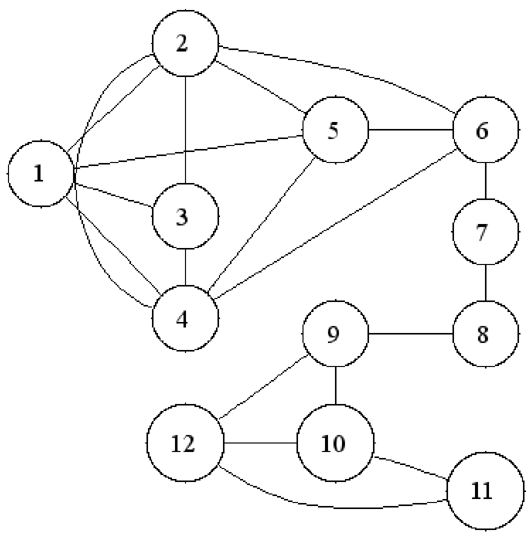
The graph in Figure 2.2 (b) is an undirected graph, and so we employ the Eq. (2.8). If we know that node 1 belongs to one community and that node 12 belongs to another community, then we can put one unit positive heat source on node 1 and one unit negative heat source on node 12. After time 1, if we set $\gamma = 0.5$, the heat distribution is [0.25, 0.16, 0.17, 0.16, 0.15, 0.09, 0.01, -0.04, -0.18 -0.21, -0.21, -0.34], and if we set $\gamma = 1$, it will be [0.17, 0.16, 0.17, 0.16, 0.16, 0.12, 0.02, -0.07, -0.18, -0.22, -0.24, -0.24]. In both settings, we can easily divide the graph into two parts: $\{1, 2, 3, 4, 5, 6, 7\}$ with positive temperatures and $\{8, 9, 10, 11, 12\}$ with negative temperatures. For directed graphs and random graphs, similarly we can cut them by employing corresponding heat solution.

Anti-manipulation

Fourth, it can be used to anti-manipulation. Let group 2 contains trusted Web pages ($j_1, j_2, \ldots, j_s$), then for each page i, $\sum_v h_{i,j_v}$ is the heat that page i receives from the group 2, and can be computed by Eq. (2.14) in the case of



(a) Group to Group Relations



(b) An undirected graph

Figure 2.2. Two graphs

a static graph or Eq. (2.18) in the case of a random graph, in which $f(0)$ is set to be a special initial heat distribution so that the trusted Web pages have unit heat while all the others have zero heat. In doing so, manipulated Web page will get a lower rank unless it has strong in-links from the trusted Web pages directly or indirectly. The situation is quite different for *PageRank* because *PageRank* is input-independent as we have shown in Section 2.3.1.

Based on the fact that the connection from a trusted page to a "bad" page should be weak: less uncross paths, longer distance and narrower pipe, we can say *DiffusionRank* can resist web spam if we can select trusted pages. It is fortunate, the trusted pages selection method in [12]—the first part of *TrustRank* can help us to fulfill this task.

In fact, the more general setting for *DiffusionRank* is $\mathbf{P} = \alpha_B \cdot \mathbf{A} + (1-\alpha_B) \cdot \frac{1}{n} \cdot \mathbf{g} \cdot \mathbf{1}^T$, and as a result, $\mathbf{R} = -\mathbf{I} + \mathbf{P}$. By such a setting, *DiffusionRank* also generalize *TrustRank* when γ tends to infinity and when g is set in the same way as *TrustRank*. However, the second part of *TrustRank* is not adopted by us. In our model, $\mathbf{g}$ should be the true "teleportation" determined by the user's browse habits, popularity distribution over all the Web pages, and so on; $\mathbf{P}$ should be the true model of the random nature of the World Wide Web. Setting $\mathbf{g}$ according to the trusted pages will not be consistent with the basic idea of Heat Diffusion on a random graph. We simply set $\mathbf{g} = \mathbf{1}$ only because we can not find it without any priori knowledge.

For such an application of *DiffusionRank*, the computation complexity for *Discrete Diffusion Kernel* is the same as that for *PageRank* in cases of both a static graph and a random graph. This can be seen in Eq. (2.14) and Eq. (2.18), by which we need N iterations and for each iteration we need a multiplication operation between a matrix and a vector, while in Eq. (2.1) and Eq. (2.2) we also need a multiplication operation between a matrix and a vector for each iteration.

2.4.3 The Physical Meaning of γ

γ plays an important role in the anti-manipulation effect of *DiffusionRank*. γ is explained as the thermal conductivity—the heat diffusion coefficient. If it has a high value, heat will diffuse very quickly. Conversely, if it is small, heat will diffuse slowly. In the extreme case, if it is infinitely large, then heat will diffuse from one node to other nodes immediately, and this is exactly the case corresponding to *PageRank*. Next, we will interpret it mathematically.

Theorem 1: When γ tends to infinity and $\mathbf{f}(0)$ is not the zero vector, $e^{\gamma\mathbf{R}}\mathbf{f}(0)$ is proportional to the stable distribution produced by *PageRank*.

Let $\mathbf{g} = \frac{1}{n}\mathbf{1}$. By the Perron Theorem [22], we have shown that 1 is the largest eigenvalue of $\mathbf{P} = [(1-\alpha)\mathbf{g}\mathbf{1}^T + \alpha\mathbf{A}]$, and that no other eigenvalue whose absolute value is equal to 1. Let $\mathbf{x}$ be the stable distribution, and so $\mathbf{Px} = \mathbf{x}$. $\mathbf{x}$ is the eigenvector corresponding to the eigenvalue 1. Assume the $n-1$ other eigenvalues of $\mathbf{P}$ are $|\lambda_2| < 1, \ldots, |\lambda_n| < 1$, we can find an invertible matrix $\mathbf{S} = \begin{pmatrix} \mathbf{x} & \mathbf{S}_1 \end{pmatrix}$ such that

$$\mathbf{S}^{-1}\mathbf{PS} = \begin{pmatrix} 1 & * & * & * \\ 0 & \lambda_2 & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & \lambda_n \end{pmatrix}. \tag{2.19}$$

Since

$$e^{\gamma\mathbf{R}} = e^{\gamma(-\mathbf{I}+\mathbf{P})} = S^{-1}\begin{pmatrix} 1 & * & * & * \\ 0 & e^{\gamma(\lambda_2-1)} & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & e^{\gamma(\lambda_n-1)} \end{pmatrix}S, \tag{2.20}$$

all eigenvalues of the matrix $e^{\gamma\mathbf{R}}$ are $1, e^{\gamma(\lambda_2-1)}, \ldots, e^{\gamma(\lambda_n-1)}$. When $\gamma \to \infty$, they become $1, 0, \ldots, 0$, which means that 1 is the only nonzero eigenvalue of $e^{\gamma\mathbf{R}}$ when $\gamma \to \infty$.

We can see that when $\gamma \to \infty$, $e^{\gamma\mathbf{R}}e^{\gamma\mathbf{R}}\mathbf{f}(0) = e^{\gamma\mathbf{R}}\mathbf{f}(0)$, and so $e^{\gamma\mathbf{R}}\mathbf{f}(0)$ is an eigenvector of $e^{\gamma\mathbf{R}}$ when $\gamma \to \infty$. On the other hand, $e^{\gamma\mathbf{R}}x = (\mathbf{I} + \gamma\mathbf{R} + \frac{\gamma^2}{2!}\mathbf{R}^2 + \frac{\gamma^3}{3!}\mathbf{R}^3 + \ldots)\mathbf{x} = \mathbf{Ix} + \gamma\mathbf{Rx} + \frac{\gamma^2}{2!}\mathbf{R}^2\mathbf{x} + \frac{\gamma^3}{3!}\mathbf{R}^3\mathbf{x} + \ldots = \mathbf{x}$ since $\mathbf{Rx} = (-\mathbf{I}+\mathbf{P})\mathbf{x} = -\mathbf{x} + \mathbf{x} = \mathbf{0}$, and hence $\mathbf{x}$ is the eigenvector of $e^{\gamma\mathbf{R}}$ for any γ. Therefore both $\mathbf{x}$

and $e^{\gamma\mathbf{R}}\mathbf{f}(0)$ are the eigenvectors corresponding the unique eigenvalue 1 of $e^{\gamma\mathbf{R}}$ when $\gamma \to \infty$, and consequently $\mathbf{x} = ce^{\gamma\mathbf{R}}\mathbf{f}(0)$.

By this theorem, we see that *DiffusionRank* is a generalization of *PageRank*. When $\gamma = 0$, the ranking value is most robust to manipulation since no heat is diffused and the system is unchangeable, but the Web structure is completely ignored since $e^{\gamma\mathbf{R}}\mathbf{f}(0) = e^{0\mathbf{R}}\mathbf{f}(0) = \mathbf{I}\mathbf{f}(0) = \mathbf{f}(0)$; when $\gamma = \infty$, *DiffusionRank* becomes *PageRank*, it can be manipulated easily. We expect an appropriate setting of γ that can balance both. For this, we have not theoretical result, but in practice we find that $\gamma = 1$ works well in Section 2.5. Next we discuss how to determine the number of iterations if we employ the discrete heat kernel.

2.4.4 The Number of Iterations

While we enjoy the advantage of the concise form of the exponential heat kernel, it is better for us to calculate *DiffusionRank* by employing Eq. (2.18) in an iterative way. Then the problem about determining the N—the number of iterations arises. We can formulate the problem as follows.

For a given threshold ϵ, find N such that $||((\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N - e^{\gamma\mathbf{R}})\mathbf{f}(0)|| < \epsilon$ for any $\mathbf{f}(0)$ whose sum is one.

Since it is difficult to solve this problem, we propose a heuristic method motivated by the following observations. When $\mathbf{R} = -\mathbf{I} + \mathbf{P}$, by Eq. (2.19), we have $(\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N = (\mathbf{I} + \frac{\gamma}{N}(-\mathbf{I} + \mathbf{P}))^N =$

$$\mathbf{S}^{-1}\begin{pmatrix} 1 & * & * & * \\ 0 & (1+\frac{\gamma(\lambda_2-1)}{N})^N & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & (1+\frac{\gamma(\lambda_n-1)}{N})^N \end{pmatrix}\mathbf{S}, \tag{2.21}$$

Compare Eq. (2.20) and Eq. (2.21), we observe that the eigenvalues of $(\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N - e^{\gamma\mathbf{R}}$ are $1 + \frac{\gamma(\lambda_n-1)}{N})^N - e^{\gamma(\lambda_n-1)}$. We propose a heuristic method to determine N so that the difference between the eigenvalues are less than a threshold for only positive λs.

We also observe that if $\gamma = 1, \lambda < 1$, then $|(1 + \frac{\gamma(\lambda-1)}{N})^N - e^{\gamma(\lambda-1)}| < 0.005$ if $N \geq 100$, and $|(1 + \frac{\gamma(\lambda-1)}{N})^N - e^{\gamma(\lambda-1)}| < 0.01$ if $N \geq 30$. So we can set $N = 30$, or $N = 100$, or others according to different accuracy requirements. In this paper, we use the relatively accurate setting $N = 100$ to make the real eigenvalues in $(\mathbf{I} + \frac{\gamma}{N}\mathbf{R})^N - e^{\gamma\mathbf{R}}$ less than 0.005.

2.5 Experiments

Since the ranking methods in [41] are mainly aimed to data manifolds, and the biased vector in the general personalized *PageRank* in [41] is unknown in the Web graph setting, we do not include the manifold ranking in the comparisons. Next we show the data, the methodology, the setting, and the results.

2.5.1 Data Preparation

Our input data consist of a toy graph, a middle-size real-world graph, and a large-size real-world graph. The toy graph is shown in the left part in Figure 2.3. The graph below it shows node 1 is being manipulated by adding new nodes $A, B, C, \ldots$ such that they all point to node 1, and node 1 points to them all.

The data of two real Web graph were obtained from the domain *cuhk.edu.hk*. The total number of pages found are 18542 in the middle-size graph, and 607170 in the large-size graph respectively. The middle-size graph is a subgraph of the large-size graph, and they were obtained by the same crawler: one is recorded by the crawler in its earlier time, and the other is obtained when the crawler stopped.

2.5.2 Methodology

The algorithms we run include *PageRank*, *TrustRank* and *DiffusionRank*. All the rank values are multiplied by the number of nodes so that the sum of the rank values is equal to the number of nodes. By this normalization, we can compare the results on graphs with different sizes since the average rank value is one for any graph after such normalization. We will need value difference and pairwise order difference as comparison measures. Their definitions are as follows.

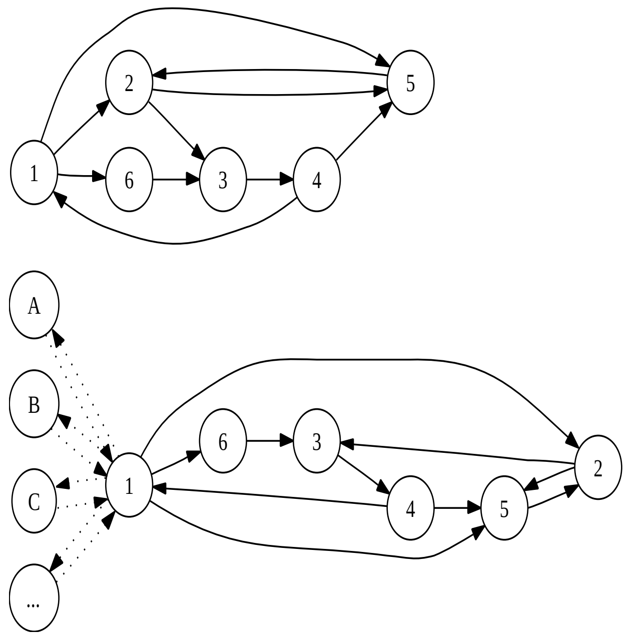
Value Difference. The value difference between $\mathbf{A} =$
$\{A_i\}_{i=1}^n$ and $\mathbf{B} = \{B_i\}_{i=1}^n$ is measured as $\sum_{i=1}^n |A_i - B_i|$.

Pairwise Order Difference. The order difference between $\mathbf{A}$ and $\mathbf{B}$ is measured as the number of significant order differences between $\mathbf{A}$ and $\mathbf{B}$. The pair $(A[i], A[j])$ and $(B[i], B[j])$ is considered as a significant order difference if one of the following cases happen: both $A[i] > [<]A[j] + 0.1$ and $B[i] \leq [\geq]A[j]$; both $A[i] \leq [\geq]A[j]$ and $B[i] > [<]A[j] + 0.1$.

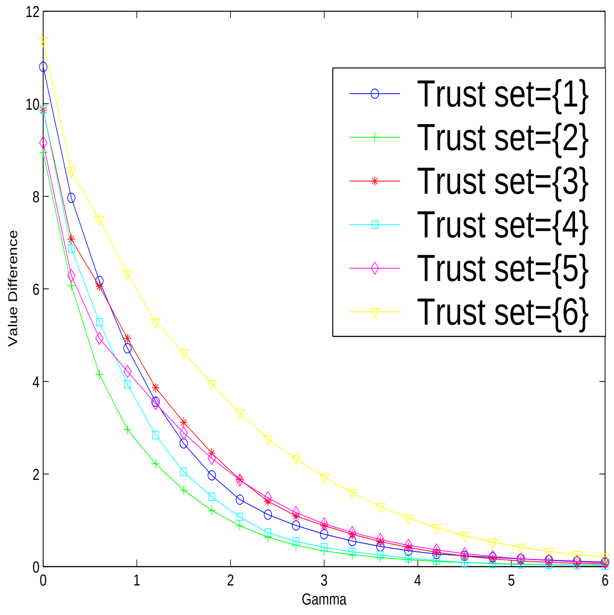
Note that the the average ranking value is one after our normalization. Since all the ranking algorithms we compared work in an iterative way, there may exit a large amount small errors caused by computation accuracy. Therefore we set the significant order difference in order to reduce the effect of small errors.

2.5.3 Experimental Set-up

The experiments on the middle-size graph and the large-size graphs are conducted on the workstation whose hardware model is Nix Dual Intel Xeon 2.2GHz, whose RAM is 1GB, and whose OS is Linux Kernel 2.4.18-27smp



(a)



(b)

Figure 2.3. Left: the toy graph consisting of six nodes, and node 1 is being manipulated by adding new nodes $A, B, C, \ldots$; Right: the approximation tendency to PageRank by DiffusionRank

(RedHat7.3). In calculating *DiffusionRank*, we employ the discrete diffusion kernel in Eq. (2.14) and Eq. (2.18) for such graphs. The related tasks are implemented using C language.

While in the toy graph, we employ the continuous diffusion kernel in Eq. (2.11) and Eq. (2.16), and implement related tasks using Matlab.

For nodes that have zero out-degree (dangling nodes), we employ the method in the modified *PageRank* algorithm [18], in which dangling nodes of are considered to have random links uniformly to each node.

We set $\alpha = \alpha_I = \alpha_B = 0.85$ in the all algorithms. We also set $\mathbf{g}$ to be the uniform distribution in both *PageRank* and *DiffusionRank*. For *DiffusionRank*, we set $\gamma = 1$. According to the discussions in Section 2.4.3 and Section 2.4.4, we set the iteration number to be $M_B = 100$ in *DiffusionRank*, and for accuracy consideration, the iteration number in all the algorithms are set to be 100.

2.5.4 Approximation of PageRank

We show that when γ tends to infinity, the value differences between *DiffusionRank* and *PageRank* tend to zero. Figure 2.3 (b) shows the approximation property of *DiffusionRank*, as proved in Theorem 1, on the toy graph. The horizontal axis of figure 2.3 marks the γ value, and vertical axis corresponds to the value difference between *DiffusionRank* and *PageRank*. All the possible trusted sets with $L = 1$ are considered, they are shown in the figure. For $L > 1$, the results should be the linear combination of some of these curves because of the linearity of the solutions to heat equations. On other graphs, the situations are similar, and we omit them.

2.5.5 Results of Anti-manipulation

In this section, we show how the rank values change as the intensity of manipulation increases. We measure the intensity of manipulation by the number of new added points that point to the manipulated point. The horizontal axes of Figure 2.4 stand for the numbers of new added points, and vertical axes show the corresponding rank values of the manipulated nodes.

To be clear, we consider all the six situations. Every node in Figure 2.3 (a) is manipulated respectively, and its corresponding values for *PageRank*, *TrustRank* (TR), *DiffusionRank* (DR) are shown in the one of six sub-figures in Figure 2.4. The vertical axes show which node is being manipulated.

In each sub-figure, the trusted sets are computed below. Since the inverse *pagerank* yields the results $[1.26, 0.85, 1.31, 1.36, 0.51, 0.71]$. Let $L = 1$. If the manipulated node is not 4, then the trusted set is $\{4\}$, and otherwise $\{3\}$.

We observe that in all the cases, rank values of the manipulated node for DiffusionRank grow slowest as the number of the new added nodes increases. On the middle-size graph and the large-size graph, this con-

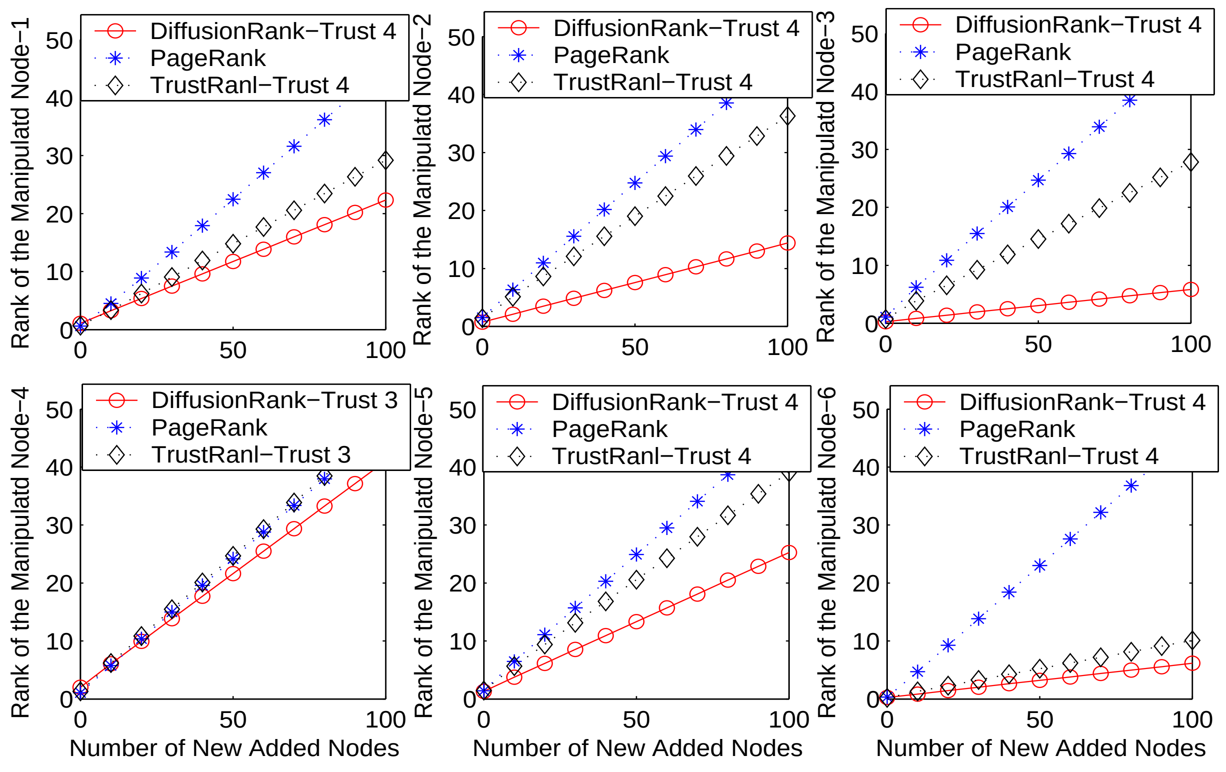


Figure 2.4. The rank values of the manipulated nodes on the toy graph

clusion is also true, see Figure 2.5. In the left sub-figure, we show the rank values of the manipulated page (www.cse.cuhk.edu.hk/˜hxyang). $L = 1$. We choose four trusted sets, on which we test *DiffusionRank* and *TrustRank*, the results are denoted by DiffusionRank$i$ and TrustRank$i$ ($i = 0, 1, 2, 3$ denotes the four trusted set). Moreover, we show the results for *DiffusionRank* when we have no trusted set, and we trust all the pages before some of them are manipulated. We also test the order difference between the ranking order A before the page is manipulated and the ranking order PA after the page is manipulated. Because after manipulation, the number of pages changes, we only compare the common part of A and PA. This experiment is used to test the stability of all these algorithms. The less the order difference, the stabler the algorithm, in the sense that only a smaller part of the order relations is affected by the manipulation.

Figure 2.6 shows that the rank values change when we add new nodes that point to the manipulated node. We give several γ setting in the left panel of the figure. We find that when $\gamma = 1$, the least order difference is achieved by DiffusionRank. It is interesting to point that as γ increases, the order difference will increase first; after reaching a maximum value, it will decrease, and finally it tends to the *PageRank* results. We show this tendency in the right panel of the Figure 2.6, in which we choose three different setting—the number of manipulated nodes are 2000, 5000, and 10000 respectively. From this figure, we can see that when $\gamma < 2$, the values are less than those for

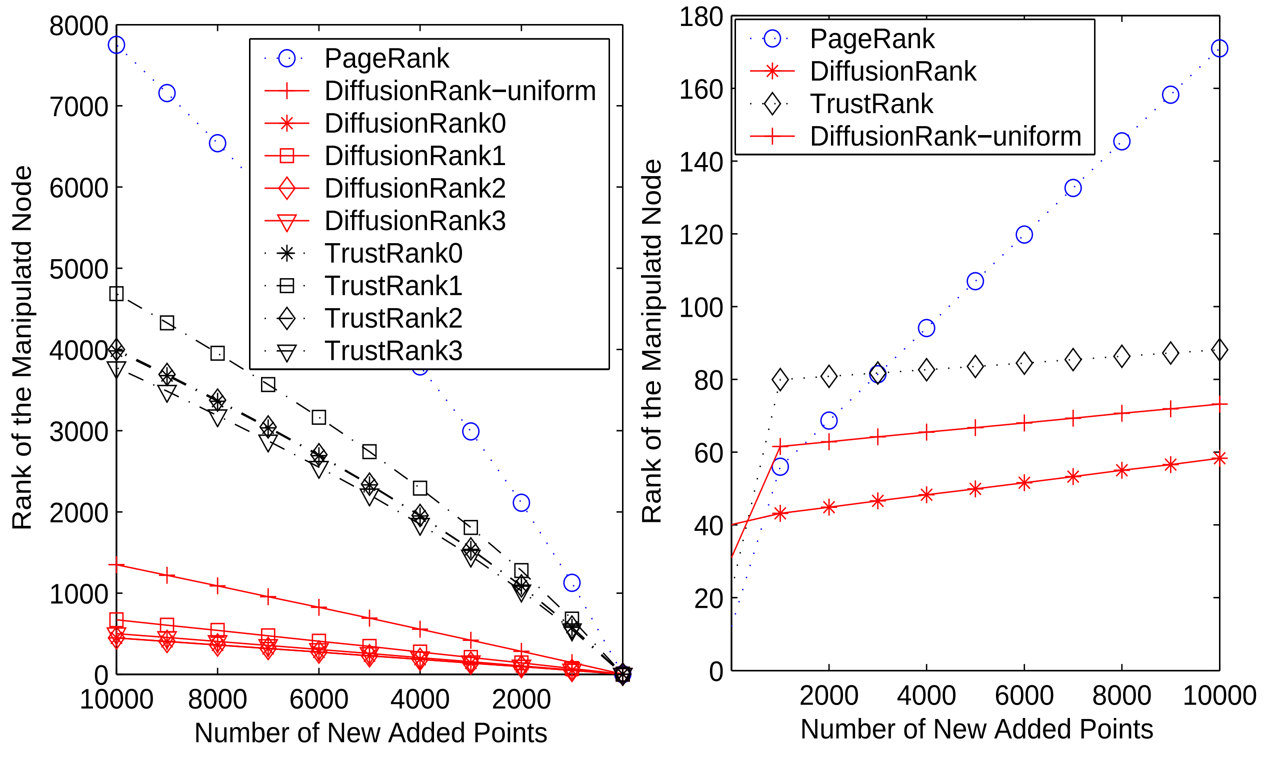


Figure 2.5. The rank values of the manipulated nodes on the middle-size graph (left) and the large-size graph (right)

PageRank, and that when $\gamma > 20$, the difference between *PageRank* and *DiffusionRank* is very small.

After these investigations, we find that

1. For all these ranking algorithms, the larger the graph, the easier that the pages can be manipulated.

2. In all the graphs we tested, *DiffusionRank* is most robust to manipulation both in value difference and order difference.

3. $\gamma = 1$ works well in all these cases.

4. The trust set selection algorithm proposed in [12] is effective for both *TrustRank* and *DiffusionRank*.

5. The difference between the *PageRank* value and the *DiffusionRank* (*TrustRank*) value can help us to detect pages being manipulated. The larger the difference, the more probably the page is manipulated. The remaining part of this Section is devoted to this finding.

From the last item, we are encouraged to develop a simple manipulation detection algorithm. For a given threshold τ, if the difference between P*ageRank* value for a particular page and its *DiffusionRank* value is greater than τ, then we consider that the page is probably being manipulated. *TrustRank* can be also employed to fulfil such task in the same way.

2.5.6 Manipulation Detection

To test this idea, we random choose 100 pages, and manipulated them all in different extents. Then we draw the Recall-Precision curves as follows: for a given recall rate ν, find the maximum threshold τ such that the recall rate is exactly ν, and then calculate the ratio of the number of nodes being manipulated and the number of nodes whose PageRank-DiffusionRank [PageRank-TrustRank] difference is greater than τ. The higher precision rate under the same recall rate means that the less “good” nodes are mixed with the ‘bad” nodes.

The upper panel in Figure 2.7 shows the results on the middle-size graph, and the lower panel shows the results on the large-size graph. If no technique is employed, one have to guess the manipulated pages in a random way, by which the precision will be $100/18542 \approx 0.0054 = 0.54\%$ on the middle-size graph, and $100/667170 \approx 0.000165 = 0.0165\%$ on the large-size graph. We also draw the curve of the random detection rates on both graphs.

We observe that both *DiffusionRank* and *TrustRank* work excellent on the detection precision on the middle-size graph. Compared to the random detection rate, they also work well on the large-size graph. From size of the areas below the curves, we find *DiffusionRank* performs slightly better than *TrustRank*.

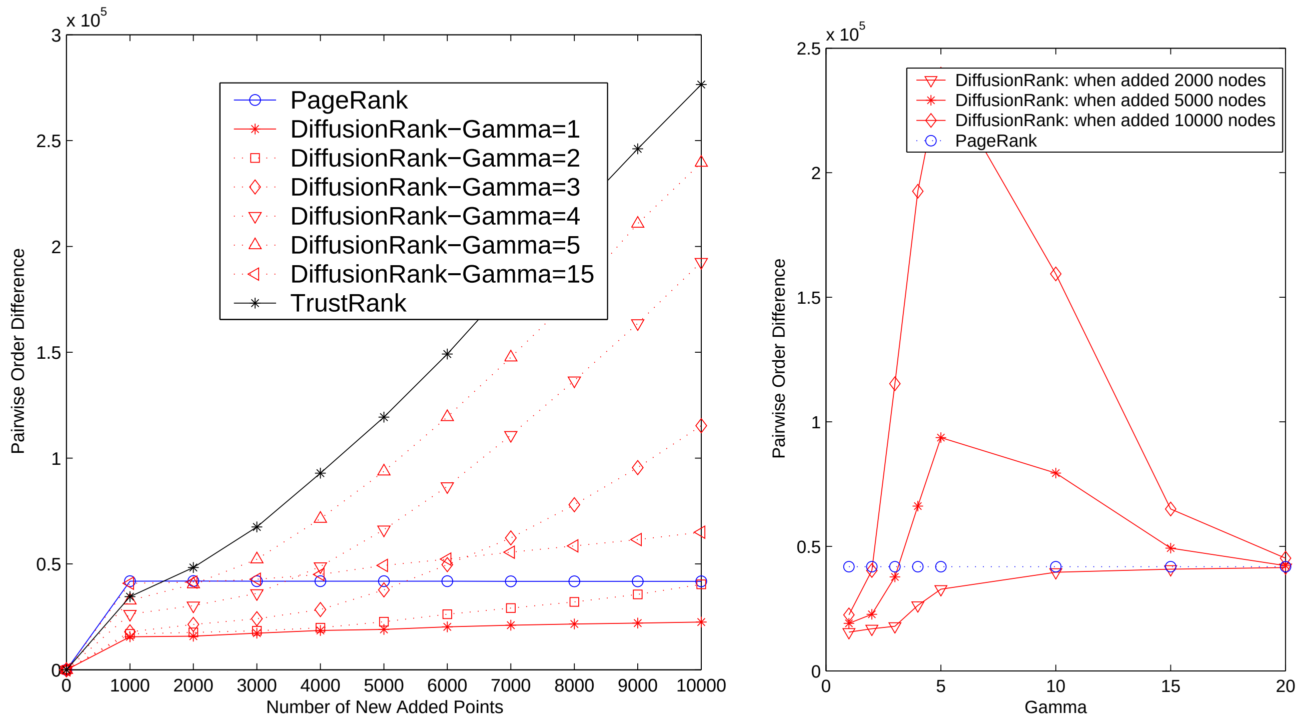


Figure 2.6. Left: Pairwise order difference on the middle-size graph, the least it is, the more stable the algorithm. Right: the tendency of varying γ

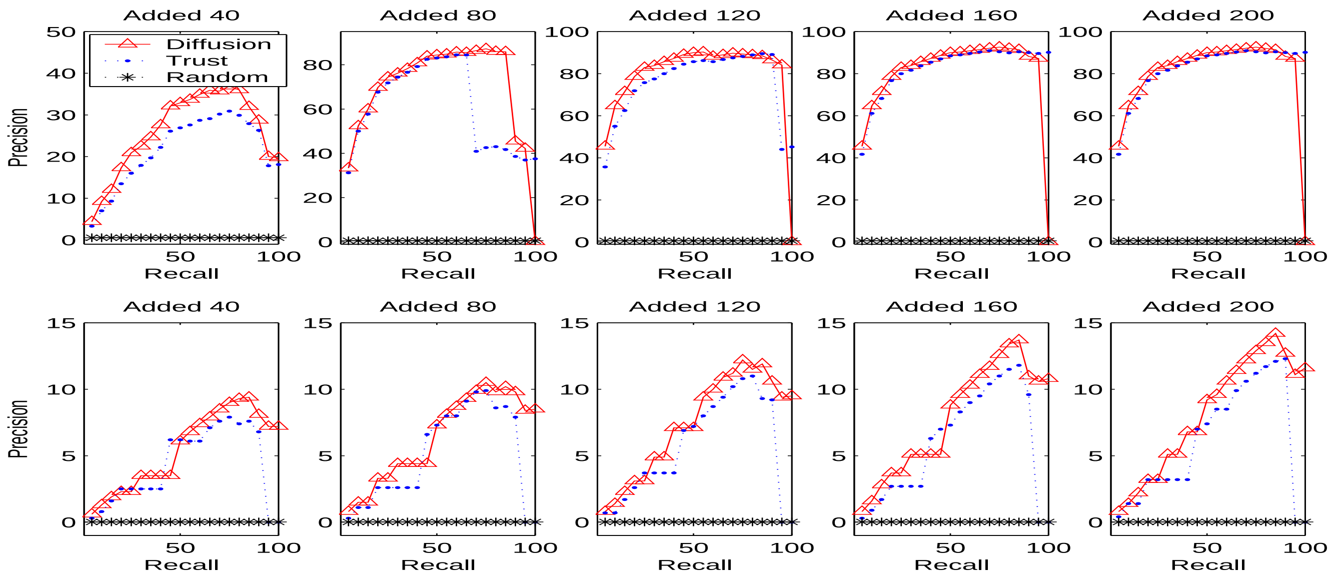


Figure 2.7. Precision vs Recall when $L = 50$: the larger the area below the curve, the better.

Algorithm 1 DiffusionRank Function

Input

$\mathbf{A}$: transition matrix

$\mathbf{U}$: the inverse transition matrix

α_I: decay factor for the inverse PageRank

α_B: decay factor for PageRank

M_I: number of iterations for the inverse PageRank

L: the number of trusted pages

γ: the thermal conductivity coefficent

output

$\mathbf{h}^*$: DiffusionRank scores

```
1:  $\mathbf{s} = \mathbf{1}$
2:  for $i = 1$ TO $M_I$ do
3:      $\mathbf{s} = \alpha_I \cdot \mathbf{U} \cdot \mathbf{s} + (1 - \alpha_I) \cdot \frac{1}{n} \cdot \mathbf{1}$
4:  end for
5:  Sort $\mathbf{s}$ in a decreasing order: $\pi = Rank(\{1, \dots, n\}, \mathbf{s})$
6:  $\mathbf{d} = \mathbf{0}$
7:  for $i = 1$ TO $L$ do
8:      if $\pi(i)$ is evaluated as a trusted page if the  then
9:          $\mathbf{d}(\pi(i)) = 1$
10:     else
11:         $L = L + 1$
12:     end if
13: end for
14: $\mathbf{d} = \mathbf{d}/|\mathbf{d}|$
15: $\mathbf{h} = \mathbf{d}$
16: Find the iteration number $M_B$ according to $\lambda$
17: for $i = 1$ TO $M_B$ do
18:     $\mathbf{h}^* = (1 - \frac{\gamma}{M_B})\mathbf{h} + \frac{\gamma}{M_B}(\alpha_B \cdot \mathbf{A} \cdot \mathbf{h} + (1 - \alpha_B) \cdot \frac{1}{n} \cdot \mathbf{1})$
19: end for
20: RETURN $\mathbf{h}^*$
```

Chapter 3

Graph-based Heat Diffusion Classifiers

This chapter is organized as follows. In Section 4.2, we establish a heat diffusion model based on a graph. In Section 3.2, we establish the Graph-based Heat Diffusion Classifier (*G-HDC*). In Section 3.3, we present two other candidate graph inputs for *G-HDC*. In Section 4.1, we interpret more about the model than what we did in [34]. In Section 4.6, we show and discuss our experimental results.

3.1 Heat Diffusion Model on a Graph

First, we give our notation for the heat diffusion model. Consider a directed weighted graph $G = (V, E, W)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{(v_i, v_j) \mid \text{there is an edge from } v_i \text{ to } v_j\}$ is the set of all edges, and $W = (w_{ij})$ is the weight matrix. In contrast to the normal undirected weighed graph, the edge (v_i, v_j) is considered as a pipe that connects to nodes i and j, and the weight w_{ij} is considered as the length of the pipe (v_i, v_j). The value $f_i(t)$ describes the temperature at node i at time t, beginning from an initial distribution of temperature given by $f_i(0)$ at time zero.

We establish our model as follows. Suppose, at time t, each node i receives an amount $M(i, j, t, \Delta t)$ of heat from its neighbor j during a period of Δt. The heat $M(i, j, t, \Delta t)$ should be proportional to the time period Δt and the temperature difference $f_j(t) - f_i(t)$. Moreover, the heat flows from node j to node i through the pipe that connects nodes i and j, and therefore the heat diffuses in the pipe in the same way as it does in the m-dimensional Euclidean space, as described in Eq. (3.7). Based on the above consideration, we assume that $M(i, j, t, \Delta t) = \alpha \cdot e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))\Delta t$.

As a result, the heat difference at node i between time $t + \Delta t$ and time t will be equal to the sum of the heat that it receives from all its neighbors and small patches around these neighbhors. This is formulated as

$$\frac{f_i(t+\Delta t)-f_i(t)}{\Delta t}=\alpha\sum_{(j,i)\in E}e^{-w_{ij}^2/\beta}(f_j(t)-f_i(t)) \tag{3.1}$$

To find a closed form solution to Eq. (4.3), we express it as a matrix form:

$$\frac{f(t+\Delta t)-f(t)}{\Delta t}=\alpha Hf(t), \tag{3.2}$$

where $H=(H_{ij})$, and

$$H_{ij}=\begin{cases}-\sum_{k:(k,i)\in E}e^{-w_{ik}^2/\beta} & j=i;\\ e^{-w_{ij}^2/\beta}, & (j,i)\in E;\\ 0, & \text{otherwise}.\end{cases} \tag{3.3}$$

In the limit $\Delta t\to 0$, Eq. (3.2) becomes

$$\frac{d}{dt}f(t)=\alpha Hf(t), \tag{3.4}$$

Solving Eq. (3.4), we get

$$f(t)=e^{\alpha tH}f(0)=e^{\gamma H}f(0), \tag{3.5}$$

where $\gamma=\alpha t$, and $e^{\gamma H}$ is defined as

$$e^{\gamma H}=I+\gamma H+\frac{\gamma^2}{2!}H^2+\frac{\gamma^3}{3!}H^3+\cdots. \tag{3.6}$$

The matrix $e^{\gamma H}$ is called the *diffusion kernel* in the sense that the heat diffusion process continues infinitely many times from the initial heat diffusion.

To make the Heat Diffusion Classifier in [34] more flexible, in the next section, we separate the classifier and its KNN graph input, and consider a general graph input instead of the specific KNN graph.

3.2 Graph-based Heat Diffusion Classifiers (G-HDC)

Based on the closed form solution in Eq. (4.5), we establish a classifier by simulating the heat diffusion based on the graph, as described follows.

Assume that there are c classes, namely, C_1,$C_2,\ldots,C_c$. Let the labelled data set contain M samples, represented by $(\mathbf{x}_i,k_i)$ $(i=1,2,\ldots,M)$, which means that the data point $\mathbf{x}_i$ belongs to class C_{k_i}. Suppose the labelled data set contain M_k points in class C_k so that $\sum_k M_k=M$. Let an unlabelled data set contains N unlabelled samples, represented by $\mathbf{x}_i$ $(i=M+1,M+2,\ldots,M+N)$.

For a give graph that can model the data relation, we apply the heat diffusion model to the graph. For the purpose of classification, for each class C_k in turn, we set the initial heat at the labelled data in class C_k to be one and all other data to be zero, then calculate the amount of heat that each unlabelled data receives from the labelled data in class C_k. Finally, we assign the unlabelled data to the class from which it receives most heat. More specifically, we describe the resulting Graph-based Heat Diffusion Classifier as follows.

[Step 1: Construct graph] Define graph G over all data points both in the training data set and in the unlabelled data set by a graph construction algorithm.

[Step 2: Compute the Heat Kernel] Using Eq. (4.4) and Eq. (4.6), find the heat kernel $e^{\gamma H}$.

[Step 3: Compute the Heat Distribution] Let

$$f^k(0) = (x_1^k, x_2^k, \ldots, x_M^k, \underbrace{0, 0, \ldots, 0}_{N})^T,$$

$k = 1, 2, \ldots, c$, where $x_i^k = 1$ if $C_{k_i} = C_k$, and $x_i^k = 0$ otherwise. Then we obtain c results for $f(t)$, namely, $f^k(t) = e^{\gamma H} f^k(0)$, $k = 1, 2, \ldots, c$. $f^k(0)$ means that all the data points in class C_k have unit heat at the initial time, while other data points have no heat, and the corresponding result $f^k(t)$ means that the heat distribution at time t is caused by the initial temperature distribution $f^k(0)$.

[Step 4: Classify the data] For $l = 1, 2, \ldots, N$, compare the p-th ($p = M+l$) components of $f^1(t), f^2(t), \ldots, f^c(t)$, and choose class C_k such that $f_p^k(t) = \max_{q=1}^c f_p^q(t)$, i.e., choose the class that distributes the most heat to the unlabelled data $\mathbf{x}_p$, then classify the unlabelled data $\mathbf{x}_p$ to class C_k.

Then we consider how to choose the input graph to *G-HDC*.

3.3 Candidate Graphs for G-HDC

In the case that the underlying geometry is unknown or its heat kernel cannot be approximated in the same way as used by [20], it is natural to approximate the unseen manifold by a graph, and to establish a heat diffusion model on the approximation graph rather than on the underlying geometry. The graph embodies the discrete structure of the nonlinear manifold. By doing so, we can imitate the way that heat flows through a nonlinear manifold. Below we consider three graph approximations.

3.3.1 KNN Graph

The KNN graph construction algorithm is commonly used in the literature [2, 31, 27, 28]. It is shown that *PWA* (Parzen Window Approach [3] when the window function is a multivariate normal kernel) and *KNN* (K-Nearest-Neighbors) are actually special cases of the classifier proposed in [34], in which the traditional KNN graph construction algorithm is slightly changed as shown below.

Define graph G over all data points by connecting points $\mathbf{x}_j$ and $\mathbf{x}_i$ from $\mathbf{x}_j$ to $\mathbf{x}_i$ if $\mathbf{x}_j$ is one of the K nearest neighbors of $\mathbf{x}_i$, measured by the Euclidean distance. Let $d(i, j)$ be the Euclidean distance between point $\mathbf{x}_i$ and point $\mathbf{x}_j$. Set edge weight w_{ij} equal to $d(i, j)$ if $\mathbf{x}_i$ is one of the K nearest neighbors of $\mathbf{x}_j$.

Note that there are $K * (M + N)$ directed edges in the resulting graph. Despite its success of the KNN graph, there is a room for other graph construction algorithms. Next we propose two other candidates.

3.3.2 SKNN-Graph

When the data lies on a low-dimensional nonlinear manifold that is embedded into a high-dimensional Euclidean space, the straight-line Euclidean distance may be not accurate because of the nonlinearity of the manifold. For example, on the surface of a sphere, the distance between two points is better measured by the geodesic path. In intuition, the smaller the strait-line Euclidean distance in a manifold, the more accurate the distance will be. This is shown in the Figure 3.1. Since AB is shorter than AC and AD, AB is more accurate than AC and AD as an approximation to its geodesic path. Based on such consideration, to make full use of accurate information (shorter

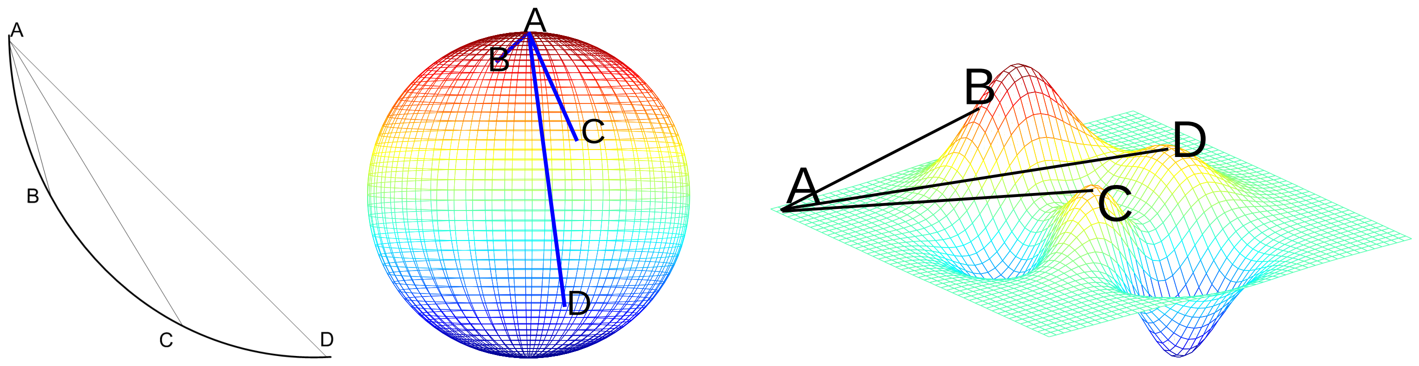


Figure 3.1. Illustrations on a manifold on which the shorter line is more accurate.

edges), we propose to construct the SKNN graph with the Shortest edges whose number is the same as the KNN graph: replace the $K * (M + N)$ edges in the KNN graph with the smallest $K * (M + N)/2$ undirected edges, which amounts to $K * (M + N)$ directed edges.

3.3.3 Minimum Spanning Tree

Given a connected, undirected weighted graph, a spanning tree of that graph is a subgraph which is a tree and connects all the vertices together. A spanning tree is called a minimum spanning tree (MST) if its weight is less than or equal to the weight of every other spanning tree. The MST, constructed from the complete graph consisting of all the data, is selected as a candidate input of *G-HDC* because of the following reasons:

1. In KNN and SKNN, we need to adjust the parameter K, while in MST, we reduce one such kind of parameter.

2. Both KNN and SKNN cannot guarantee the connection of the resulting graph, while MST is a connected graph.

However, MST cannot replace the KNN and SKNN in any case because there is no cycle in a MST, and thus it lacks the ability of modelling the complex geometry. For example, a point on a sphere will diffuse heat to another point through various paths. However, a tree-like graph cannot model this multi-connection since there is only one path between two points.

As an example of these three candidate graph input, in Figure 4.2(a), we show 2000 points on a 2-dimensional spiral manifold which is embedded into 3-dimensional space. In Figure 4.2(b) and 4.2(d), we show KNN graph approximation ($K = 6$) and SKNN graph approximation ($K = 6$) of the spiral manifold, which contains 200 points drawn from the 2000 points in figure 4.2(a). In Figure 4.2(c) we show MST graph approximation. In *G-HDC,* Step 1 consists of one of the above graph construction algorithms. In Step 2 and 3, the heat diffuses from the labelled data to the unlabelled data along the graph, and consequently, the heat flows along the spiral manifold. In Step 4, if the unlabelled data is closer to one class of points in the sense that it receives more heat in total from data in such class, it is classified into this class, otherwise the other class.

3.3.4 Advantages and Disadvantages

The three input graphs have their own advantages and disadvantages, as described below.

The KNN Graph is democratic to each node because each node has exactly the same number of nodes that point to it. Moreover, the resulting classifier *KNN-HDC* is a generalization of KNN. However, the KNN graph may not be connected, and longer edges may be chosen in the KNN graph while shorter edges are removed.

The SKNN Graph loses the advantages of being democratic to each node, and may not be connected. However, it fits the ideas that shorter edges are set more important than longer edges in a manifold.

For Minimum Spanning Tree, we enjoy its advantages of being connected. Moreover, the resulting matrix H is a sparse matrix (it contains only $2n$ nonzero elements), which helps faster calculation of $e^{\gamma H}$ using Eq. (4.6). Also we reduce a parameter K while we need to ascertain K in KNN and SKNN.

3.4 Interpretation

In [34], the local heat diffusion and the connection with other models such as KNN and Parzen Window Approach have been introduced. In this section, we will provide more justification for the proposed *G-HDC*.

3.4.1 Generalization of Gaussian Density

The heat diffusion kernel $K_t(\mathbf{x}, \mathbf{y})$ is a special solution to Eq. (2.3) with a special initial condition called the delta function $\delta(\mathbf{x} - \mathbf{y})$, which describes a unit heat source at position **y** with no heat in other positions, in other words, $\delta(\mathbf{x} - \mathbf{y}) = 0$ for $\mathbf{x} \neq \mathbf{y}$ and $\int_{-\infty}^{+\infty} \delta(\mathbf{x} - \mathbf{y}) d\mathbf{x} = 1$. Based on this, the heat kernel $K_t(\mathbf{x}, \mathbf{y})$ describes the heat distribution at time t diffusing from the initial unit heat source at position **y**. Since arbitrary initial conditions can be considered as a combination of heat sources with different intensities at different positions and the heat equation is linear, the heat kernel can be used to generate the solution to Eq. (2.3) as $f(\mathbf{x}, t) = \int_M K_t(\mathbf{x}, \mathbf{y}) f_0(\mathbf{y}) d\mathbf{y}$. It is interesting to investigate the special case when M is the familiar $m-$dimensional Euclidean Space. In such a case, Δ is the Laplacian, Δf is simplified as $\Delta f = \sum\limits_i \frac{\partial^2 f}{\partial x_i^2}$, and the heat kernel has an explicit form

$$K_t(\mathbf{x}, \mathbf{y}) = (4\pi t)^{-\frac{m}{2}} e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{4t}}, \tag{3.7}$$

which is the same as the Gaussian density. From this point of view, the heat kernel $K_t(\mathbf{x}, \mathbf{y})$ can be considered as a generalization of Gaussian density–when the geometric manifold varies, the corresponding heat kernel varies and can be considered as the generalization of Gaussian density from a flat Euclidean space to a general manifold.

3.4.2 Hopfield Model

The Hopfield neural network [1] [14] consists of N pairwise connected neurons. The i-th neuron can be either in $f_i = -1$ (off) or $f_i = +1$ (on). The connections between points are undirected and have strengths that are fixed real numbers. Let w_{ij} be the strength of the connection from neuron j to neuron i. The strengths usually satisfy: $w_{ij} = w_{ji}$, and $w_{ii} = 0$. Define the state vector **f** (to be a binary vector (+1) whose i-th component corresponds to the state of the i-th neuron.

Each neuron examines its inputs and decides whether to turn itself on or off according to the effect of its neighbors on it and the action threshold. More specifically, it is described as follows:

Let T_i be the threshold voltage of the i-th neuron. If the weighted sum over all of its inputs is greater than or equal to T_i, the $i-$th neuron turns on and its state becomes +1. If the sum is less than T_i, the neuron turns off

[1]Thank the anonymous reviewer for pointing out the relation between the Hopfield Model and the Heat Diffusion Model.

and its state becomes -1. The action of each neuron at time $t+1$ simulates a general threshold function of $N-1$ variables (the states of all the other neurons) at time t. If all the neurons update their values simultaneously, the Hopfield network can be described:

$$f_i(t+1) = sgn(\sum_{j=1}^{N} w_{ij} f_j(t) - T_j) \tag{3.8}$$

Let **W** be an $N \times N$ real-valued, zero-diagonal symmetric matrix. The entries of **W** are the w_{ij} defined above; Let the threshold vector **t** be a real-valued vector whose i-th component is the threshold voltage of the i-th neuron. Then Eq. (3.8) can be written as a matrix form:

$$\mathbf{f}(t+1) = sgn(\mathbf{W}\mathbf{f}(t) - \mathbf{T}) \tag{3.9}$$

From the viewpoint of referring to neighbors, *G-HDC* is similar to Hopfield model, which is the original model which determines class by looking at immediate neighbors.

It is interesting to point out that Eq. (3.9) and Eq. (4.5) has similar appearance. However, it is difficult for us to compare these two models in practice because we need to determine the matrices **W** and **T** if Eq. (3.9) is employed as a classifier in the similar way as Eq. (4.5) although **W** and **T** can be calculated as a content-addressable memory [14].

3.5 Experiments

The KNN graph, SKNN graph, and MST are considered as input of the *G-HDC*, and results in three algorithms *KNN-HDC*, *KN-HDC*, *MST-HDC*. Moreover, *KNN*, *PWA* (Parzen Window Approach when the window function is a multivariate normal kernel), and *SVM* are employed to be reference algorithms. All algorithms are applied to three synthetic datasets and six datasets from the UCI Repository [13]. Since discrete attributes and the problem of missing values are out of the scope of this paper, we simply remove all the discrete attributes and remove all the cases that contain missing values. Table 3.1 describes the resulting datasets we use. Syn-1, Syn-2 and Syn-3 are synthetic datasets. Syn-2 and Syn-3 are from the same spiral data as shown in Figure (4.2)(a), but with different numbers of data. Syn-1 is obtained from Syn-2 by ignoring the third attribute. In the spiral data set, the data in one class are distributed on a spiral rotated clockwise while the data in another class are distributed on a spiral rotated anti-clockwise.

We obtain the free parameters in *KNN-HDC*, *MST-HDC*, and *SKNN-HDC* via five cross-validations on the training data and unlabelled data (a transductive learning setting), and those in *PWA* and *KNN* only on the training

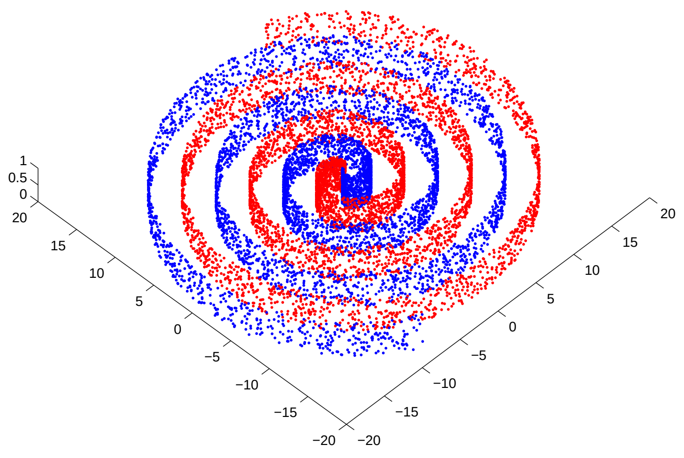
Table 3.1. Description of the Datasets

Dataset	Cases	Classes	Attributes
Syn-1	100	2	2
Syn-2	100	2	3
Syn-3	200	2	3
Breast-w	683	2	9
Glass	214	6	9
Iono	351	2	34
Iris	150	3	4
Sonar	208	2	60
Vehicle	846	4	18

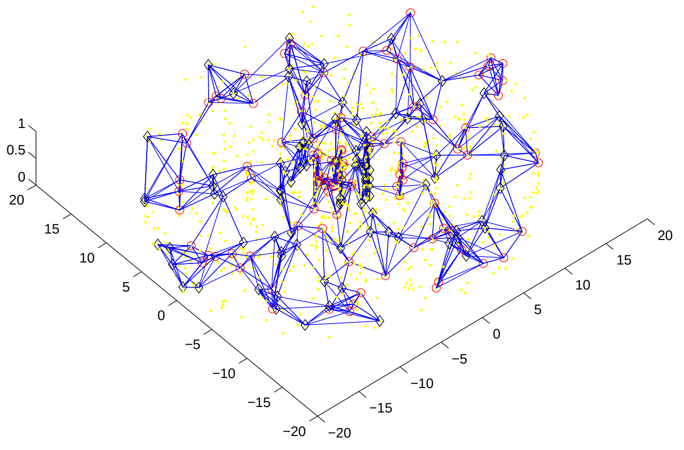
Table 3.2. Mean Accuracy of SVM, KNN,PWA, KNN-HDC (KNN-H), MST-HDC (MST-H) and SKNN-HDC (SKNN-H) on the 6 datasets

Dataset	SVM	KNN	PWA	KNN-H	MST-H	SKNN-H
Syn-1	66.0	67.0	80.0	93.0	**95.0**	**95.0**
Syn-2	34.0	67.0	83.0	93.0	**94.0**	89.0
Syn-3	54.0	79.5	**92.0**	91.0	90.0	**92.0**
Breast-w	96.8	94.1	96.6	96.9	95.9	**99.4**
Glass	68.1	61.2	63.5	68.1	68.7	**70.5**
Iono	93.7	83.2	89.2	**96.3**	**96.3**	**96.3**
Iris	96	97.3	95.3	**98**	92.0	94.7
Sonar	88.5	80.3	53.9	90.9	91.8	**94.7**
Vehicle	**84.8**	63.0	66.0	65.5	83.5	66.6

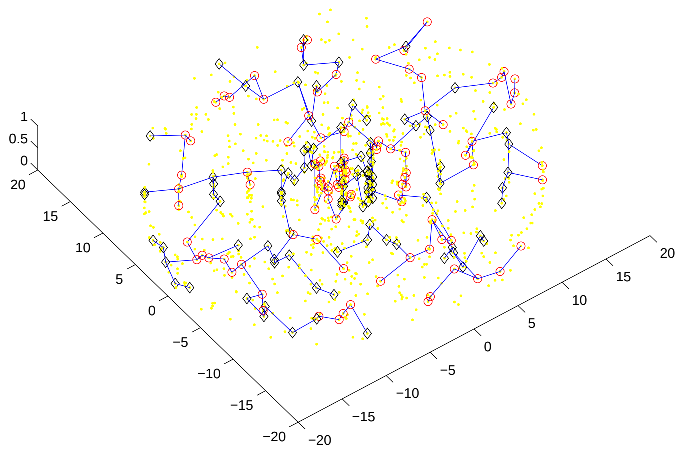
data. We employ the Gaussian kernels for the *SVM*; the experimental results are obtained by using the LIBSVM software [6], in which the width parameter is obtained via five cross-validations on the training data. The figure shown in Table 4.1 is the mean accuracy of the ten-fold cross-validations. Note that the results on Glass, Iris, Sonar, and Vehicle are slightly different from those in [34], in which parameters are tuned outside the cross-validation. In each row, the maximum results are shown in bold. From the results, we observe that both *SKNN-HDC* and *MST-HDC* perform better than *PWA* and *KNN* in accuracy, which shows the necessity of introducing the new graph constructing method for *G-HDC*. It is interesting to mention that *KNN-HDC*, *MST-HDC*, and *SKNN-HDC* can employ unlabelled data to construct graphs so that better accuracy is achieved.



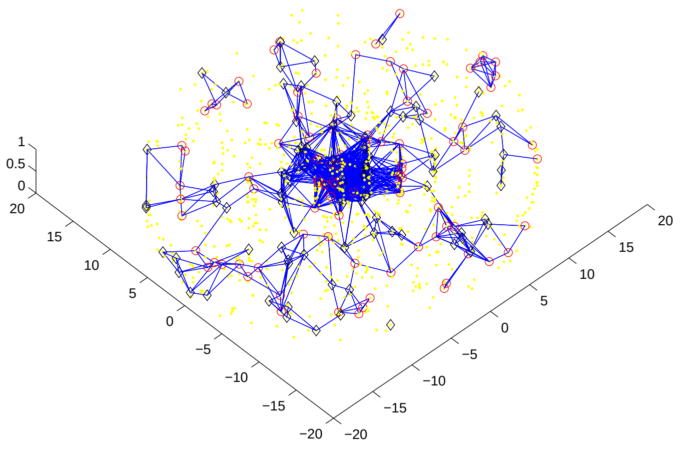
(a)



(b)



(c)



(d)

Figure 3.2. An illustration on the spiral manifold and its graph approximation. (a) The 2000 points (b) KNN Neighborhood Graph (c) Minimum Spanning Tree (d)KN shortest edges

Chapter 4

Novel Volume-based Heat Diffusion Model and Its Resulting Classifier

4.1 Motivations

In order to develop the Heat Diffusion Model further, we investigate various methods of solving Eq. (2.3). Numerical methods for differential equation in continuum mechanics have traditionally been classified into three main approaches: finite element (FE), boundary element (BE), and finite difference (FD) methods [1]. For the heat diffusion equation, the situation is similar. The FE method for the heat diffusion equation is used in surface smoothing (for example, see [7, 30]). Because we do not know the true geometry in which the data points lie, we cannot construct the triangle mesh in our model and therefore we cannot use the FE and BE methods.

Fortunately we find that we can generalize the FD method. In the following, we illustrate the FD method for the heat diffusion equation by considering the special case when the manifold is a two-dimensional Euclidean space. In such a case, the heat diffusion equation (2.3) becomes

$$\begin{cases} \frac{\partial f}{\partial t} - \frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} & = & 0, \\ f(x, y, 0) & = & f_0(x, y). \end{cases} \tag{4.1}$$

The FD method begins with the discretization of space and time. For simplicity, we assume equal spacing of the points x_i in one dimension with intervals of size $\Delta x = x_{i+1} - x_i$, equal spacing of the points y_j in another dimension with intervals of size $\Delta y = y_{j+1} - y_j$ (assume $\Delta y = \Delta x = d$ for simplicity), and equal spacing of the time steps t_k at intervals of $\Delta t = t_{k+1} - t_k$. $f(i, j, k)$ is the heat at position x_i, y_j at time t_k. The grid on the plane is shown in Figure (4.1)(a). The grid creates a natural graph: the set of nodes is $\{(i, j)\}$, and node (i, j) is connected to node (i', j') if and only if $|i - i'| + |j - j'| = 1$. Note that each node (i, j) has four neighbors:

$(i-1,j), (i+1,j), (i,j-1)$, and $(i,j+1)$.

Based on this discretization and approximation of the function, we then write the following approximations of its derivatives in space and time:

$$\left.\frac{\partial f}{\partial t}\right|_{(i,j,k)} \approx \frac{f(i,j,k+1)-f(i,j,k)}{\Delta t},$$
$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{(i,j,k)} \approx \frac{f(i-1,j,k)-2f(i,j,k)+f(i+1,j,k)}{(\Delta x)^2},$$
$$\left.\frac{\partial^2 f}{\partial y^2}\right|_{(i,j,k)} \approx \frac{f(i,j-1,k)-2f(i,j,k)+f(i,j+1,k)}{(\Delta y)^2}.$$

This leads to the difference form of the heat equation as follows:

$$\begin{aligned}\frac{f(i,j,k+1)-f(i,j,k)}{\Delta t} &= \frac{f(i-1,j,k)-2f(i,j,k)+f(i+1,j,k)}{(\Delta x)^2} + \frac{f(i,j-1,k)-2f(i,j,k)+f(i,j+1,k)}{(\Delta y)^2} \\ &= \frac{[(f(i-1,j,k)-f(i,j,k))+(f(i+1,j,k)-f(i,j,k))}{d^2} + \frac{(f(i,j-1,k)-f(i,j,k))+(f(i,j+1,k)-f(i,j,k))]}{d^2}\end{aligned} \tag{4.2}$$

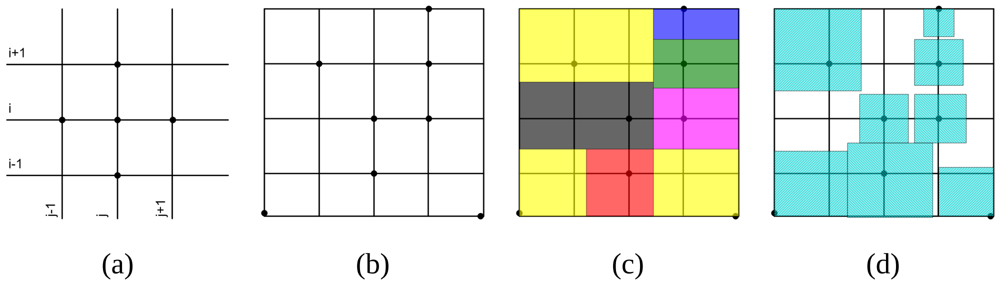


Figure 4.1. (a) The grid on the two dimensional space. (b) The eight irregularly positioned points. (c) The small patches around the irregular points. (d) The square approximations of the small patches.

In the real data analysis, we often face problems where we cannot employ the FD method directly:

1. The graph constructed from the data points is irregular;
2. The density of data varies; this also results in an irregular graph;
3. The manifold is unknown;
4. The differential equation expression is unknown even if the manifold is known.

We aim to solve these problems using a Volume-based Heat Diffusion Model (*VHDM*) by generalizing the FD method. The novel heat diffusion model on the graph leads to a novel classifier called *Volume-based Heat Diffusion Classifier* (*VHDC*).

The remainder of this chapter is organized as follows. In Section 4.2, we establish *VHDM*. In Section 4.3, we establish *VHDC*. In Section 4.4, we interpret the parameters in more detail. Then in Section 4.5, we show the related work. In Section 4.6, we show the experimental results.

4.2 Volume-based Heat Diffusion Model on a Graph

First, we give our notation for the heat diffusion model. Consider a directed weighted graph $G = (V, E, W)$, where $V = \{v_1, v_2, \ldots, v_n\}, E = \{(v_i, v_j)$ |there is an edge from v_i to $v_j\}$ is the set of all edges, and $W = (w_{ij})$ is the weight matrix. In contrast to the normal undirected weighed graph, the edge (v_i, v_j) is considered as a pipe that connects to nodes i and j, and the weight w_{ij} is considered as the length of the pipe (v_i, v_j). The value $f_i(t)$ describes the temperature at node i at time t, beginning from an initial distribution of temperature given by $f_i(0)$ at time zero.

Note that the heat and the temperature at points with unit mass are equivalent. As a result, the terms of temperature and heat at such a point are interchangeable. However, we employ these two term carefully to fit the traditional usage.

We establish our model as follows. Suppose, at time t, each node i receives an amount $HM(i, j, t, \Delta t)$ of heat from its neighbor j during a period of Δt. The heat $HM(i, j, t, \Delta t)$ should be proportional to the time period Δt and the temperature difference $f_j(t) - f_i(t)$. Moreover, the heat flows from node j to node i through the pipe that connects nodes i and j, and therefore the heat diffuses in the pipe in the same way as it does in the m-dimensional Euclidean space, as described in Eq. (3.7). Based on the above consideration, we assume that $HM(i, j, t, \Delta t) = \alpha \cdot e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))\Delta t$. Next we consider the representation ability of each node. There are only a finite number of nodes in the graph that are transparent to a certain observer. But in a manifold, there are infinitely many nodes, most of which are unreachable to the observer. We can assume that

1. There is a small patch $SP[j]$ of space containing node j and many nodes around node j; node j is seen by the observer, but the small patch is unseen to the observer.

2. The volume of the small patch $SP[j]$ is $V(j)$, and the heat diffusion ability of the small patch is proportional to its volume. This assumption is reasonable because not only do the nodes take part in the heat diffusion process, but also the unseen small patch has an effect on the heat diffusion.

3. The temperature in the small patch $SP[j]$ at time t is almost equal to $f_j(t)$ because every unseen node in the small patch is near node j.

4. The small patch $SP[j]$ diffuses an amount $HM(i, j, t, \Delta t)V(j)$ of heat to node i.

As a result, the heat difference at node i between time $t + \Delta t$ and time t will be equal to the sum of the heat that it receives from all its neighbors and the small patches around these neighbhors. This is formulated as

$$\frac{f_i(t+\Delta t) - f_i(t)}{\Delta t} = \alpha \sum_{(j,i)\in E} e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))V(j) \tag{4.3}$$

To find a closed form solution to Eq. (4.3), we express it as a matrix form: $\frac{f(t+\Delta t)-f(t)}{\Delta t} = \alpha H f(t)$, where $H = (H_{ij})$, and

$$H_{ij} = \begin{cases} -\sum_{k:(k,i)\in E} e^{-w_{ik}^2/\beta} V(k) & j = i, \\ e^{-w_{ij}^2/\beta} V(j), & (j,i) \in E, \\ 0, & \text{otherwise}. \end{cases} \tag{4.4}$$

In the limit $\Delta t \to 0$, we have $\frac{d}{dt} f(t) = \alpha H f(t)$. Solving it, we get a close form expression:

$$f(t) = e^{\alpha t H} f(0) = e^{\gamma H} f(0), \tag{4.5}$$

where $\gamma = \alpha t$, and $e^{\gamma H}$ is defined as

$$e^{\gamma H} = I + \gamma H + \frac{\gamma^2}{2!}H^2 + \frac{\gamma^3}{3!}H^3 + \cdots \approx (I + \frac{\gamma}{p}H)^p. \tag{4.6}$$

The matrix $e^{\gamma H}$ is called the *diffusion kernel* in the sense that the heat diffusion between nodes from time 0 to t is completely described by the elements in the matrix. For the sake of computational considerations, $e^{\gamma H} f(0)$ can be approximated as $(I + \frac{\gamma}{p}H)^p f(0)$, where p is a large integer. The latter can be calculated by iteratively applying the operator $(I + \frac{\gamma}{p}H)$ to $f(0)$.

In the model, $V(i)$ is used to estimate the volume of the small patch around node i. Intuitively, if the data density is high around node i, the nodes around node i will have a high probability of being selected, and thus there are fewer unseen nodes around node i. In this paper, we define $V(i)$ to be the volume of the hypercube whose side length is the average distance between node i and its neighbors. Formally,

$$V(i) = \eta\Big(\frac{1}{K_i} \sum_{j:(j,i)\in E} w_{ij}\Big)^{\nu},$$

where ν is the dimension of the space in which graph G lies, K_i is the number of neighbors of node i, and η is a normalized parameter such that $\sum_{i\in V} V(i) = 1$.

One may notice that, in the current definition, if one point is far from all the other points, the volume of the hypercube will be very high and so this point will dominate after normalization with η, which contradicts the basic idea of local volumes. The truth is that if one point A is far from all the other points, its volume will certainly be large, but the amount of heat that A diffuses to other points is small because, in the term $e^{-w_{ij}^2/\beta}V(j)$ in H,

$e^{-w_{ij}^2/\beta}$ drops very quickly when w_{ij} increases. This is further explained by the fact that a term of the form $e^{-d^2} * d$ tends to zero quickly when d tends to infinity. So there is no contradiction here with the idea of local volumes.

Volumes are theoretically important because heat diffuses throughout the whole of any given volume in a physical system, and the concept of the volume is crucial in its ability to represent the whole space, including both known points and other points between them. Moreover, the idea of volume can be explained further by the definition of local charts in a differential manifold as shown in [20].

Definition 2: An $m-$dimensional differential manifold $\mathcal{M}$ is a set of points that is locally equivalent to the $m-$dimensional Euclidean space $\mathcal{R}^m$ by smooth transformations, supporting operations such as differentiation. Formally, a differentiable manifold is a set $\mathcal{M}$ together with a collection of local charts $\{(U_i, \phi_i)\}$, where $U_i \subset \mathcal{M}$ with $\cup_i U_i = \mathcal{M}$, and $\phi_i : U_i \subset \mathcal{M} \to \mathcal{R}^m$ is a bijection. For each pair of local charts (U_i, ϕ_i) and (U_j, ϕ_j), it is required that $\phi_j(U_i \cap U_j)$ is open and $\phi_{ij} = \phi_i \circ \phi_j^{-1}$ is a diffeomorphism.
The small patch around each point i can be considered as a local charts U_i, and the volume of i is the volume of U_i. Consequently the whole manifold $\mathcal{M}$ is formed by sticking the small patches together.

4.2.1 Calculation of ν

In the definition of volumes, we introduce the parameter ν describing the dimension of the space in which graph G lies. From the definition of a differential manifold, ν corresponds to the unknown dimension m of the local Euclidean space. In the following, we consider how to determine the value off this parameter.

Why PCA is unsuitable

PCA is a traditional method for dimension estimation. In this method, the intrinsic dimension is determined by the number of eigenvalues greater than a given threshold. Both global PCA and local PCA have the disadvantage of introducing another parameter–the threshold. Moreover, global PCA methods fail on nonlinear manifolds, on which our model is established; local methods depend heavily on the precise choice of local regions [32].

Thus, instead of PCA, we choose the maximum likelihood estimation method proposed in [21]. Apart from avoiding the problems with PCA just mentioned, this method also has the advantage that the graph is constructed by K nearest neighbors, as described in Section 4.3, where the parameter K is the same as that in [21].

Maximum Likelihood Estimation of Intrinsic Dimension

If $T_j(x)$ is the Euclidean distance from a fixed point x to its j-th nearest neighbor in the sample, then the local dimension $\hat{m}_K(x)$ at point x can be estimated by a maximum likelihood estimation, as described in [21], as follows.

$$\hat{m}_K(x) = \left[\frac{1}{K-1} \sum_{j=1}^{K-1} \log \frac{T_K(x)}{T_j(x)} \right]^{-1}. \tag{4.7}$$

To avoid overflow of the float calculation when $T_j(x)$ is very small, we slightly change formula (4.7) to the following:

$$\hat{m}_K(x) = \left[\frac{1}{K-1} \sum_{j=1}^{K-1} \log \frac{T_K(x)+\epsilon}{T_j(x)+\epsilon} \right]^{-1}. \tag{4.8}$$

ϵ is set to be 0.0000001 in this paper. Then $\nu = \frac{1}{n} \sum\limits_{i=1}^{n} \hat{m}_K(x_i)$.

4.2.2 Generalization of FD Method

In *VHDM*, if $\beta \to +\infty$, the graph is of the form as shown in Figure (4.1)(a), which means each node has four neighbors, and if the volume of each node is set to be one, then Eq. (4.3) becomes Eq. (4.2). Therefore we can say that *VHDM* generalizes the FD method from Euclidean space to unknown space. The generalization is interesting for its ability to solve the following problems.

1. **Irregularity of the graph.** By setting β to be finite, we actually soften the neighborhood relation between the data points, and thus we avoid the difficulty in handling the irregularity of the graph constructed by the data points. For example, in Figure (4.1)(b), the central data point has four neighbors, which are not positioned on nodes in the grid. The FD method has difficulty in handling such a case. Even worse, in real data sets, each data point has many neighbors, which are positioned in a space with an unknown dimension.

2. **Variation of density.** The data points are not drawn uniformly, and we use the volume of the hypercube around a node to perform the local density estimation around the node. In Figure (4.1)(c), the whole space is covered by small patches, and in Figure (4.1)(d) each small patch is approximated by a small square. In this way, we actually consider the unseen points so that the concept of heat diffusion on a graph can be treated as an approximation of heat diffusion in a space. There is no such consideration in the FD method.

3. **Unknown manifold and unknown differential equation expression.** In most cases, we do not know the true manifold that the data points lie in, or we cannot find the exact expression for the *Laplace-Beltrami operator*; therefore we cannot employ the FD method. In contrast, our model has the advantage of not depending on the manifold expression and the differential equation expression. Moreover, volumes serve as patches that are connected together to form the underlying unknown manifold, while each volume is a local Euclidean space. The idea of volume fits the definition of local charts in differential manifold.

Based on the closed form solution in Eq. (4.5), we establish a classifier as described in the next section.

4.3 Volume-based Heat Diffusion Classifier

Assume that there are c classes, namely, $1, 2, \ldots, c$. Let the labelled data set contain M samples, $(\mathbf{x}_i, k_i)$ ($i = 1, 2, \ldots, M$), which means that the data point $\mathbf{x}_i$ belongs to class k_i. Suppose the labelled data set contain M_k points in class k so that $\sum_k M_k = M$. Let an unlabelled data set contain N unlabelled samples, represented by $\mathbf{x}_i$ ($i = M+1, M+2, \ldots, M+N$).

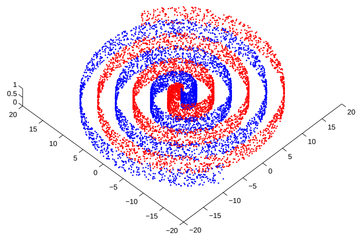
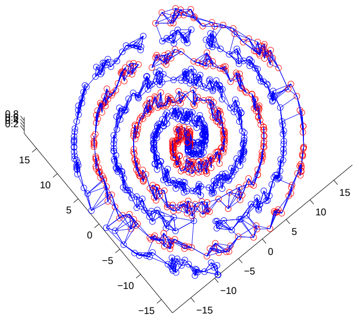
We first employ the neighborhood construction algorithm commonly used in the literature, for example in [2, 27, 28, 31], to form a graph for all the data. Then we apply the heat diffusion kernel to the graphs. For the purpose of classification, for each class k in turn, we set the initial heat at the labelled data point in class k to be one and all other data points to be zero, then calculate the amount of heat that each unlabelled data point receives from the labelled data points in class k. Finally, we assign the unlabelled data point to the class from which it receives most heat. More specifically, we describe the resulting *VHDC* as follows.

Step 1: Construct neighborhood graph Define graph G over all data points both in the training data set and in the unlabelled data set by connecting points $\mathbf{x}_j$ and $\mathbf{x}_i$ from $\mathbf{x}_j$ to $\mathbf{x}_i$ if $\mathbf{x}_j$ is one of the K nearest neighbors of $\mathbf{x}_i$, measured by the Euclidean distance. Let $d(i, j)$ be the Euclidean distance between point $\mathbf{x}_i$ and point $\mathbf{x}_j$. Set edge weight w_{ij} equal to $d(i, j)$ if $\mathbf{x}_i$ is one of the K nearest neighbors of $\mathbf{x}_j$, and set $n = M + N$.

Step 2: Compute the Heat Distribution Using Eq. (4.6), we obtain c results for $f(t)$, namely, $f^k(t) = e^{\gamma H} f^k(0)$, $k = 1, 2, \ldots, c$. Where $f^k(0) = (x_1^k, x_2^k, \ldots, x_M^k, 0, 0, \ldots, 0)^T$, $k = 1, 2, \ldots, c$, $x_i^k = 1$ if $k_i = k$, and $x_i^k = 0$ otherwise. Here $f^k(0)$ means that all the data points in class k have unit heat at the initial time, while other data points have no heat, and the corresponding result $f^k(t)$ means that the heat distribution at time t is caused by the initial temperature distribution $f^k(0)$.

Step 3: Classify the data For $l = 1, 2, \ldots, N$, compare the p-th ($p = M+l$) component of $f^1(t), f^2(t), \ldots, f^c(t)$, and choose class k such that $f_p^k(t) = \max_{q=1}^{c} f_p^q(t)$, i.e., choose the class that distributes the most heat to the unlabelled point $\mathbf{x}_p$, then classify the unlabelled point $\mathbf{x}_p$ to class k.

As an example of Step 1, in Figure 4.2(a), we show 2,000 points on a 2-dimensional spiral manifold which is embedded into 3-dimensional space. In Figure 4.2(b), we show the neighborhood graph approximation of the spiral manifold, which contains 1000 points drawn from the 2000 points in Figure 4.2(a), and in which each node has 3 neighbors. In Step 2, the heat diffuses from the labelled data to the unlabelled data along the graph, and

(a) (b)

Figure 4.2. An illustration of the spiral manifold and its graph approximation. (a) The 2000 data points on a spiral manifold. (b) Neighborhood Graph of the 1000 data points on the spiral manifold.

consequently, the heat flows along the spiral manifold. In Step 3, if the unlabelled data point is closer to one class in the sense that it receives more heat in total from this class of data, then the unlabelled data point is classified into this class; otherwise, it is classified into the other class.

Before discussing the roles of the parameters, we show one advantage of the generalization of *KNN*. It is well known that expected error rate of *KNN* is between P and $2P$ when N tends to infinity, where P is the Bayes error rate. Therefore the upper bound of the expected error rate of *VHDC* is less than $2P$ if β is infinity and volume is constant. It should be tighter if appropriate parameters for *VHDC* are found.

4.4 Roles of the Parameters

It is easy to find that K is employed to control the manifold approximation, and that ν is used to model the true dimensionality of the manifold that the data lie in.

4.4.1 Local Heat Diffusion Controlled by β

In Section 4.2, we assumed that the heat diffuses in the pipe in the same way as it does in the m-dimensional Euclidean space. It turns out [2] that in an appropriate coordinate system $K_t(\mathbf{x}, \mathbf{y})$ on a manifold is approximately the Gaussian:

$$K_t(\mathbf{x}, \mathbf{y}) = (4\pi t)^{-\frac{m}{2}} e^{-||\mathbf{x}-\mathbf{y}||^2/4t}(\phi(\mathbf{x}, \mathbf{y}) + O(t)),$$

where $\phi(\mathbf{x}, \mathbf{y})$ is a smooth function with $\phi(\mathbf{x}, \mathbf{x}) = 1$, and when t is small, $O(t)$ can be neglected. Therefore when $\mathbf{x}$ and $\mathbf{y}$ are close and t is small, we have $K_t(\mathbf{x}, \mathbf{y}) \approx (4\pi t)^{-\frac{m}{2}} e^{-||\mathbf{x}-\mathbf{y}||^2/4t}$. For more details, see [2, 26]. In *VHDM* in Section 4.2, heat flows in a small time period Δt, and the pipe length between node i and node j is small (recall that we create an edge from j to i only when j is one of the K nearest neighbors). So the above approximation can be used in our model, and we rewrite it as $K_{\Delta t}(i,j) \approx (4\pi\Delta t)^{-\frac{m}{2}} e^{-w_{ij}^2/4\Delta t}$. According to the Mean-Value Theorem and the fact that $K_0(i,j) = 0$, we have

$$K_{\Delta t}(i,j) = K_{\Delta t}(i,j) - K_0(i,j) = \left.\frac{dK_{\Delta t}(i,j)}{d\Delta t}\right|_{\Delta t=\beta} \Delta t \approx \alpha \cdot e^{-w_{ij}^2/4\beta}\Delta t,$$

where β is a parameter that depends on Δt, and $\alpha = \frac{1}{4}w_{ij}^2\beta^{-2-m/2} - \frac{1}{2}m\beta^{-1-m/2}$. To make our model concise, α and β simply serve as free parameters because the relation between Δt and β is unknown. This explains the statement that β controls the local heat diffusion from time t to $t + \Delta t$, and the reason why we assume that at time t, each node i receives an amount $HM(i,j,t,\Delta t) = \alpha \cdot e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))\Delta t$ of heat from its neighbor j.

4.4.2 Global Heat Diffusion Controlled by γ

From $\gamma = \alpha t$, we can see that γ controls the global heat diffusion from time 0 to t. Another interesting finding is that γ can be explained as a regularization parameter: when $\gamma = 0$, we have $e^{\gamma H} f(0) = If(0) = f(0)$, which results in a classifier that has zero error on the training set. When $\gamma \to +\infty$, the system will stop diffusing heat, and the heat at each node are equal. This means the function on the graph becomes the smoothest in the sense that the variance between values on neighbors is the smallest. The best γ is a tradeoff of the training error and the smoothness, and should not be zero or infinity.

Finally, we investigate the singular behavior of *VHDC* in the limit $\gamma \to 0$. If we simply let $\gamma = 0$ in the equation $e^{\gamma H} f(0)$, then we only get a trivial classifier as shown above. From a different viewpoint, we observe the following interesting phenomena:

Subtracting I from $e^{\gamma H}$ then dividing by γ changes the values of the testing data in the same scale, and so does not change the performance of the classifier, that is, $(e^{\gamma H} - I)/\gamma f(0)$ behaves the same as $e^{\gamma H} f(0)$ as a classifier. Then we can take the limit over $(e^{\gamma H} - I)/\gamma f(0)$, and we obtain

$$\begin{aligned}\lim_{\gamma\to 0}\frac{(e^{\gamma H} - I)}{\gamma} f(0) &= \lim_{\gamma\to 0}\tfrac{I+\gamma H+\frac{\gamma^2}{2!}H^2+\cdots-I}{\gamma} f(0) \\ &= \lim_{\gamma\to 0}(H + \tfrac{\gamma}{2!}H + \cdots)f(0) \\ &= Hf(0).\end{aligned}$$

We consider $Hf(0)$ as the singular behavior of *VHDC* in the limit $\gamma \to 0$.

4.4.3 Stability of *VHDC* with respect to parameters

If the parameters in a model is not stable, then a small deviation from the best value of a parameter may result in a totally different performance. This instability of the parameters is not desirable. In this section, we try to show that the parameters β, γ, and ν are not sensitive to the classifier *VHDC*.

Since $e^{\gamma H}$ is continuous on β, γ, and ν in the sense that small changes in these parameters result in a small change in $e^{\gamma H}$, *VHDC* is not sensitive to these three parameters if they are changed slightly. This can be seen from the existence of the derivatives of $e^{\gamma H}$ with respect to β, γ, and ν:

$$\frac{de^{\gamma H}}{d\gamma} = e^{\gamma H} H$$

$$\frac{de^{\gamma H}}{d\beta} = e^{\gamma H}\frac{dH}{d\beta}, \ \frac{dH}{d\beta} = (\frac{dH_{ij}}{d\beta})$$

$$\frac{dH_{ij}}{d\beta} = \begin{cases} -\sum_{k:(k,i)\in E} e^{-w_{ik}^2/\beta} w_{ik}^2 \beta^{-2} V(k) & j = i, \\ e^{-w_{ij}^2/\beta} w_{ij}^2 \beta^{-2} V(j) & (j,i) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

$$\frac{de^{\gamma H}}{d\nu} = e^{\gamma H}\frac{dH}{d\nu}, \frac{dH}{d\nu} = (\frac{dH_{ij}}{d\nu})$$

$$\frac{dH_{ij}}{d\nu} = \begin{cases} -\sum_{k:(k,i)\in E} e^{-w_{ik}^2/\beta} V(k) \log(\frac{1}{K_k}\sum_{l:(l,k)\in E} w_{kl}) & j = i, \\ e^{-w_{ij}^2/\beta} V(j) \log(\frac{1}{K_j}\sum_{l:(l,j)\in E} w_{jl}) & (j,i) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

It is well-known that $\Delta \mathbf{f} \approx \frac{d\mathbf{f}}{dt}\Delta t$. Since there exist derivatives of $e^{\gamma H}$ with respect to β, γ, and ν, we can say that $e^{\gamma H}$ is stable with respect to these parameters, and so is $e^{\gamma H} f(0)$.

For the parameter K, it has an unstable effect on the classifier *VHDC*. Increasing or decreasing K by one will result in a structural change in the underlying *KNN* graph; as a result, the values in the matrices H and $e^{\gamma H}$ will change dramatically. However, this property of instability has no impact on the performance of *VHDC* because K is a natural number and all possible K can be tested by the cross-validation on the training data, so that the best value can be chosen successfully.

The discrete parameter is quite different from the continuous parameters γ, β and ν, for which we must choose the appropriate values by testing a subset of all their possible values. Under such a circumstance, stability is important for continuous parameters because there may be a small variation between the best value and the nearest one in the subset, and the property of stability can guarantee that there is no big difference on the performance between the true best value and the best-performing value chosen from the subset tested.

4.5 Discussions

In Section 4.1, we have shown that the heat diffusion kernel $K_t(\mathbf{x}, \mathbf{y})$ is a special solution to Eq. (2.3) with a special initial condition called the delta function $\delta(\mathbf{x} - \mathbf{y})$. From this point of view, the heat kernel $K_t(\mathbf{x}, \mathbf{y})$ can be considered as a generalization of Gaussian density–when the geometric manifold varies, the corresponding heat kernel varies and can be considered as the generalization of Gaussian density from a flat Euclidean space to a general manifold. Since we approximate the unknown manifold by a neighborhood graph, it is interesting to show the similarity between heat diffusion on a manifold and heat diffusion on a neighborhood graph.

4.5.1 Neighborhood Graph and Manifold

In the case that the underlying geometry is unknown or its heat kernel cannot be approximated in the same way as used by [20], it is natural to approximate the unseen manifold by the graph created by the K nearest neighbors in our model, and to establish a heat diffusion model on the neighborhood graph rather than on the underlying geometry. The graph embodies the discrete structure of the nonlinear manifold. By doing so, we can imitate the way that heat flows through a nonlinear manifold.

Next, we list some correspondences between the heat diffusion model on graphs and the heat diffusion model on manifolds:

1. The heat diffusion equation on a graph is $\frac{d}{dt}f(t) = \alpha H f(t)$; the heat diffusion equation on a manifold is, from Eq. (2.3),

$$\begin{cases} \frac{\partial f}{\partial t} & = & \Delta f, \\ f(\mathbf{x}, 0) & = & f_0(\mathbf{x}). \end{cases}$$

2. The solution to the heat diffusion equation on a graph is $f(t) = e^{\alpha t H} f(0) = e^{\gamma H} f(0)$; the solution to the heat diffusion equation on a manifold is $f(\mathbf{x}, t) = \int_M K_t(\mathbf{x}, \mathbf{y}) f_0(\mathbf{y}) d\mathbf{y}$.

3. The delta function $\delta(\mathbf{x} - \mathbf{y})$ is used to represent a unit heat source at position **y**; the vector $\mathbf{e}_j$, whose $j-$th element is one while other elements are zero, is used to represent a unit heat source at node j.

4.5.2 Manifold Learning

When the data points lie on a low-dimensional nonlinear manifold that is embedded into a high-dimensional Euclidean space, the straight-line Euclidean distance may not be accurate because of the nonlinearity of the manifold. For example, on the surface of a sphere, the distance between two points is better measured by the geodesic path. Much recent work has captured the nonlinearity of the curved manifold. One common idea is that the local information such as local distance used by [31], local linearity used by [27], and local covariance matrix used by [33] in a nonlinear manifold is relatively accurate, and can be used to construct the global information. This idea is reasonable because, in a manifold, every small area is equivalent to an Euclidean space, and can be mapped to it by a smooth transformation. While inheriting this idea in our model, we also adopt the concept of thinking globally and fitting locally described by [28]. In practice, we fit the unknown manifold structure locally by the neighborhood graph, and we also fit the heat diffusion locally. Then in the final step we think globally by accumulating the local heat flow.

4.5.3 Heat Kernel

The Heat Kernel is proposed in [19, 20, 29]. The success in [20] is achieved because a closed form approximation to the heat kernel on the multinomial family is found. While this approximation fits the problem of text classification well, for some other geometries, however, there is no closed form solution for the heat kernel. Even worse, in most cases, the underlying geometry structure is unknown. In such cases, it is impossible to construct the heat kernel for the geometry in a closed form. In contrast, there is always a closed form solution – a heat kernel for the graph that approximates the geometry – in our model.

The outside appearance of $e^{\gamma H}$ is the same as that in [19, 29]; however, the numerical value of $e^{\gamma H}$ in our paper is quite different from [19, 29] (refer to Eq. (4.4)). The heat kernel in [19, 20, 29] is applied to a large margin classifier; in contrast, the heat kernel is employed directly to construct a classifier in our model. Admittedly, our method has the limitation of being applied in the inductive learning setting. Nonetheless, it is interesting and challenging to apply the proposed $e^{\gamma H}$ to *SVM* when it is not symmetric (which is usually true when the volume is considered). The heat kernel issues deserve further investigations, but are outside the scope of this paper, and so the empirical comparison on heat kernels is not provided.

4.5.4 Transductive Learning

VHDC is built on a graph and it is actually a semi-supervised algorithm: it needs access to the unlabelled data. Along these lines, our method is related to [38, 39, 40] . The models in [39, 40] are mainly concerned with directed graphs such as the Web link, on which the co-citation is meaningful. This co-citation calculation, however, is not being considered in our model; hence a comparison with [39, 40] is inappropriate, and is not provided empirically.

We are interested in comparing the model proposed in [38], which is the model in the literature most closely related to our proposed *VHDC*. Although our model adopts a different approach, there is an overlap between our solution and that in [38]. The overlap happens when our volume is not being considered and γ is small in our model, while α is small and the normalization is not performed in [38]. This can be seen from the approximation $(I-\alpha S)^{-1} \approx I+\alpha S$ when α is small. As a result, $(I-\alpha S)^{-1}Y$ has similar performance to SY. It is easy to see that, when γ is small, $e^{\gamma H}Y$ has the similar performance as HY. Consequently, when the normalization in [38] is not performed, and when the volume is not considered, S and H are equal except for the diagonal elements, which have no effect on the classifiers SY and HY. Another interesting point is that the classifier $(I-\alpha S)^{-1}Y$ is supported by a regularization framework. It is true that currently we cannot find a similar regularization approach that can output the proposed classifier $e^{\gamma H}Y$, but we can interpret it in another way: γ plays a role like the regularization parameter as shown in Section 4.4.2.

Employed as baselines, two other popular transductive SVM algorithms (*UniverSVM*[8], *SVMLight*[17]) are compared to our method in the experiment section.

4.6 Experiments

The Parzen Window Approach (*PWA*–P) , *KNN*–K, two Transductive SVMs: *UniverSVM*–U [8] and *SVM-Light*–L [17], Consistency Method–C [38] and its two variant, C1 and C2, and *VHDC* and *HDC* (the special version of *VHDC* when the volume is not considered) are applied here to three synthetic datasets and ten datasets from the UCI Repository [13].

Since discrete attributes and the problem of missing values are out of the scope of this paper, we simply remove all the discrete attributes and remove all the cases that contain missing values. The boolean attributes in the Zoo dataset are considered as continuous attributes. The first four columns in Table 4.1 describe the resulting datasets we use. n, c and d are the number of cases, classes and features, respectively. Syn-1, Syn-2 and Syn-3 are synthetic datasets. Syn-2 and Syn-3 are from the same spiral data as shown in Figure (4.2)(b), but with different numbers of data. Syn-1 is obtained from Syn-2 by ignoring the third attribute. In the spiral data set, the data points in one class are distributed on a spiral rotated clockwise while the data points in another class are distributed on a spiral rotated counter-clockwise.

We obtain the free parameters in *PWA*, *KNN*, *C*, *C1*, *C2*, *HDC* and *VHDC* via nine-fold cross-validations on the training dataset including the testing data without labels. We employ the Gaussian kernels for *UniverSVM*, and *SVMLight*; the width parameter σ and penalty parameter C for them are also tuned via nine-fold cross-validation on the training dataset.

Note that the results are quite different if we choose the best values in each cross-validation in hindsight, i.e.,

Table 4.1. Mean Accuracy on the 13 Datasets

Dataset	**n**	**c**	**d**	**K**	**P**	**U**	**L**	**C**	**C1**	**C2**	**HD**	**VHD**	**VHD-ν**
Syn-1	100	2	2	67.0	80.0	88.0	86.0	94.0	93.0	90.0	93.0	95.0	**96.0**
Syn-2	100	2	3	67.0	83.0	89.0	86.0	93.0	93.0	93.0	93.0	**94.0**	**94.0**
Syn-3	200	2	3	79.5	92.0	92.1	**92.5**	92.0	91.5	90.0	91.0	**92.5**	**92.5**
Breast-w	683	2	9	94.1	96.6	64.1	67.8	98.4	97.4	98.1	96.9	97.4	**99.4**
Credit-a	666	2	6	65.5	63.7	76.1	76.9	84.5	84.5	84.5	89.8	89.6	**90.7**
Credit-g	1000	2	7	70.8	70.0	70.0	70.7	**97.0**	**97.0**	**97.0**	96.2	96.2	96.2
Diabetes	768	2	8	73.5	73.4	78.4	78.5	86.8	86.7	84.9	78.9	88.4	**88.8**
Glass	214	6	9	61.2	63.5	67.6	**73.7**	72.3	69.6	70.5	68.1	72.9	72.9
Hepatitis	148	2	3	**79.8**	**79.8**	78.5	79.1	**79.8**	**79.8**	**79.8**	**79.8**	**79.8**	**79.8**
Iono	351	2	34	83.2	89.2	93.7	94.3	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**	**96.3**
Iris	150	3	4	97.3	95.3	96.7	96.7	95.3	95.3	95.3	98.0	**98.7**	**98.7**
Waveform	300	3	21	85.0	85.3	86.7	85.3	86.3	86.0	87.0	**88.3**	**88.3**	**88.3**
Zoo	101	7	16	40.6	40.6	**97.2**	97.0	40.6	40.6	40.6	40.6	40.6	40.6
Zoo+PCA	101	7	8	87.0	90.1	97.1	**98.0**	96.0	94.0	96.0	94.0	97.0	97.0

the testing data with label is given when we choose the parameters.

The figure shown in Table 4.1 is the mean accuracy of the ten-fold cross-validations. The last column indicates the results of *VHDC* when we choose the value of ν that corresponds to the best classification accuracy in hindsight. The performance of this choice gives us a benchmark to measure the performance of our calculation of ν by employing the method in [21].

From the results, we observe that both *HDC* and *VHDC* uniformly outperforms *PWA* and *KNN* in accuracy, as we expected. The better results of *VHDC*-ν over *HDC* show the necessity of introducing the volume representation of a node in a graph.

VHDC has the same results as *VHDC*-ν in nine of the thirteen datasets. Among four other datasets, *VHDC* is worse significantly than *VHDC*-ν only on one dataset Breast-w, and *VHDC* achieves slightly worse results than *VHDC*-ν on Syn1, Credit-a and Diabetes. This shows that the local dimension estimation in *VHDC* is successful.

The overall results on the ten Benchmark data indicate that our approach *VHDC* is competitive with the Consistency Method and Transductive SVM on problems without any *a priori* knowledge. The better results on the three synthetic datasets show that *VHDC* fits problems with a manifold structure especially well.

We also observe that *PWA*, *KNN*, *C*, *C1*, *C2*, *HDC* and *VHDC* perform more poorly than Transductive SVM on dataset Zoo; indeed, the difference is as high as 46.6%. This can be explained by the fact that all these methods depend heavily on the distance measure, and as a consequence, if the direct Euclidean distance is not accurate, these methods will perform poorly. We think that the noises in the Zoo dataset causes inaccurate distance measurement between data points. To find the performance of there algorithms on dataset Zoo with less noise, we preprocess it with *PCA* such that the dimensionality is reduced from the original 16 to 8. The results are shown in the last row of Table 4.1. The difference between *VHDC* and *UniverSVM* is thus reduced to 1%.

Chapter 5

Conclusions

We conclude that *DiffusionRank* is a generalization of *PageRank*, which is interesting in that the heat diffusion coefficient γ can balance the extent that we want to model the original Web graph and the extent that we want to reduce the effect of link manipulations. The experimental results show that we can actually achieve such a balance by setting $\gamma = 1$, although the best setting is still under further investigation. This anti-manipulation feature enables *DiffusionRank* to be a candidate as a penicillin for Web spamming. Moreover, *DiffusionRank* can be employed to find group-group relations and to partition Web graph into small communities. All these advantages can be achieved in the same computational complexity as *PageRank*. For the special application of anti-manipulation, *DiffusionRank* performs better both in precision-recall curves and in reduction effects while keeping best stability among all the three algorithms.

All the three classifiers *KNN-HDC*, *SKNN-HDC*, and *MST-HDC* can be employed as candidates for Graph-based Heat Diffusion Classifiers. They share the idea of approximating the manifold by the graph so that we can avoid the difficulty of finding the explicit expression for the unknown geometry in most cases. By establishing the heat diffusion equation on the graph, we avoid the difficulty of finding a closed form heat kernel for some complicated geometries. Our experiments have shown that *SKNN-HDC* and *MST-HDC* are promising, and enrich the family of heat diffusion classifiers.

The proposed *VHDM* has the following advantages: it can model the effect of unseen points by introducing the volume of a node, it avoids the difficulty of finding the explicit expression for the unknown geometry by approximating the manifold by a finite neighborhood graph, and it has a closed form solution that describes the heat diffusion on a manifold. While the proposed *VHDC* is a generalization of both the Parzen Window Approach (when the window function is a multivariate normal kernel) and *KNN*, our experiments have demonstrated that *VHDC* gives accurate results in a classification task. In order to capitalize on these promising achievements, further study is needed on the following problems: How to apply *VHDC* to inductive learning, how to find a graph that better approximates the manifold in stead of the *KNN* graph, and how to construct a better volume representation of the unseen points.

Bibliography

[1] A. Becker. *An Introductory Guide to Finite Element Analysis*. Professional Engineering Publishing: London and Bury St Edmunds,UK, 2004.

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

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] B. Bollobás. *Random Graphs*. Academic Press Inc. (London), 1985.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM Press.

[6] C. Chang and C. Lin. Libsvm: a library for support vector machines., 2001.

[7] M. Chung and J. Taylor. Diffusion smoothing on brain surface via finite element method. In *IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 432–435, 2004.

[8] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. submitted, 2005.

[9] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceeding of the 13th World Wide Web Conference*, pages 309–318, 2004.

[10] A. Ellis and T. Hagino, editors. *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14*. ACM, 2005.

[11] D. Fogaras and B. Rácz. Scaling link-based similarity search. In Ellis and Hagino [10], pages 641–650.

[12] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB*, pages 576–587, 2004.

[13] S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[14] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[15] H. Ino, M. Kudo, and A. Nakamura. Partitioning of web graphs by community topology. In Ellis and Hagino [10], pages 661–669.

[16] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. *Proc. of SIGKDD*, 2002.

[17] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical, pages 41–56. MIT-Press, 1999.

[18] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[19] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, 2002.

[20] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, Jan 2005.

[21] E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *NIPS*, 2004.

[22] C. R. MacCluer. The many proofs and applications of perron's theorem. *SIAM Review*, 42(3):487–498, 2000.

[23] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 83–92, New York, NY, USA, 2006. ACM Press.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report Paper SIDL-WP-1999-0120 (version of 11/11/1999), Stanford Digital Library Technologies Project, 1999.

[25] M. Richardson, A. Prakash, and E. Brill. Beyond pagerank: machine learning for static ranking. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 707–715, New York, NY, USA, 2006. ACM Press.

[26] S. Rosenberg. *The Laplacian on a Riemmannian Manifold*. Cambridge University Press, 1997.

[27] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(22):2323–2326, Dec 2000.

[28] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensinal manifolds. *Journal of Machine Learning Research*, 4:119–155, Jun 2003.

[29] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, 2005.

[30] B. Tang, G. Sapiro, and V. Caselles. Direction diffusion. In *ICCV*, pages 1245–1252, 1999.

[31] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–2323, Dec 2000.

[32] P. J. Verveer and R. P. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):81–86, 1995.

[33] P. Vincent and Y. Bengio. Manifold parzen windows. In *NIPS*, 2002.

[34] H. Yang, I. King, and M. R. Lyu. NHDC and phdc: Non-propagating and propagating heat diffusion classifiers. In *ICONIP2005*, pages 394–399, 2005.

[35] H. Yang, I. King, and M. R. Lyu. Predictive ranking: a novel page ranking approach by estimating the web structure. In *WWW (Special interest tracks and posters)*, pages 944–945, 2005.

[36] H. Yang, I. King, and M. R. Lyu. Predictive random graph ranking on the web. In *2006 International Joint Conference on Neural Networks*, pages 3491–3498, 2006.

[37] H. Yang, I. King, and M. R. Lyu. Heat diffusion classifiers on graphs. submitted, 2006.

[38] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[39] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, 2005.

[40] D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *NIPS*, 2004.

[41] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.