Machine Learning Models on Random Graphs

Haixuan YANG

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in
Department of Computer Science & Engineering

Supervised by
Prof. Irwin KING & Prof. Michael R. LYU

Machine Learning Models on Random Graphs

submitted by

Haixuan YANG

for the degree of Doctor of Philosophy

at the Chinese University of Hong Kong

Abstract

Abstract

In this thesis, we establish three machine learning models on random graphs: Heat Diffusion Models on Random Graphs, Predictive Random Graph Ranking, and Random Graph Dependency. The heat diffusion models on random graphs lead to Graph-based Heat Diffusion Classifiers (*G-HDC*) and a novel ranking algorithm on Web pages called *DiffusionRank*. For *G-HDC*, a random graph is constructed on data points. The generated random graph can be 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. Experiments show that *G-HDC* can achieve better performance in accuracy in some benchmark datasets. For *DiffusionRank*, theoretically we show that it is a generalization of *PageRank* when the heat diffusion coefficient tends to infinity, and empirically we show that it achieves the ability of anti-manipulation.

Predictive Random Graph Ranking (*PRGR*) incorporates *DiffusionRank*. *PRGR* aims to solve the problem that the incomplete information about the Web structure causes inaccurate results of various ranking algorithms. The Web structure is predicted as a random graph, on which ranking algorithms are expected to be improved in accuracy. Experimental results show that the *PRGR* framework can improve the accuracy of the ranking algorithms such as PageRank and Common Neighbor.

Three special forms of the novel Random Graph Dependency measure on two random graphs are investigated. The first special form can improve the speed of the C4.5 algorithm, and can achieve better results on attribute selection than γ used in Rough Set Theory. The second special form of the general random graph dependency measure generalizes the conditional entropy because it becomes equivalent to the conditional entropy when the random graphs take their special form–equivalence relations. Experiments demonstrates that the second form is an informative measure, showing its success in decision trees on small sample size problems. The third special form can help to search two parameters in *G-HDC* faster and more accurate than the cross-validation method.

In summary, the viewpoint of random graphs indeed provides us an opportunity of improving some existing machine learning algorithms.

Acknowledgment

There are many persons I would like to thank. First and foremost, I want to thank my supervisors, Prof. Irwin King and Prof. Michael R. Lyu. I gain too much from their guidance in both the attitude in doing research and the detailed technique things during my Ph. D study. I would like to express my sincere gratitude and appreciation to their supervision, encouragement, and support at all levels. I will always be grateful for the outstanding research environment fostered by our department, and also for so related work done by our clerical staffs.

I would like to thank my colleagues and my friends. I acknowledge the help provided by Patrick Lau, Zhenjiang Lin and Zenglin Xu in the early work related to the random graph ranking. I thank Wenye Li and Kun Zhang for the constructive discussions in conducting the research work in this thesis.

I would like to express my thanks to many anonymous reviewers for valuable comments.

I also want to thank my office-mates, Steven Chu-Hong Hoi, Jianke Zhu, Ming Cai, Hon Hei Edward Yau, and Hongbo Deng. Their cooperative spirit creates a good working and discussion environment in the office.

I extend my gratitude to Kaizhu Huang, Xia Cai, Xinyu Chen, Xiaoqi Li, Edith Ngai, Pat Chan, Yi Liu, Yangfan Zhou, Shi Lu, Hackker Wong, Chi-Hang Chan, Hao Ma and Xiang Peng for their help and discussion in many aspects of my research work.

Finally, I want to thank my family. Without their deep love and constant support, this thesis could not have been completed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The objective of this thesis is to provide a random graph perspective in the field of machine learning. To address the motivations of this perspective, we introduce the concepts of random graphs and machine learning, and provide an intersection between random graphs and machine learning. As a summary, we present the objectives of this thesis and outline the contributions. Finally, we provide an overview of the rest of this thesis.

We hope this thesis can provide an exciting direction where graph theory and machine learning go hand in hand to spawn new research results.

1.1 Random Graphs

The definition of a random graph [15] is given below.

Definition 1 A random graph $RG = (U, P = (p_{ij}))$ is defined as a graph with a vertex set U in which the edges are chosen independently, and for $1 \leq i, j \leq |U|$ the probability of (v_i, v_j) being an edge is exactly p_{ij}. Unless stated otherwise, in this thesis we set $p_{ii} = 1$ for $1 \leq i \leq |U|$, meaning that the edge (v_i, v_i) exists with the probability one. We can simply say random graph $RG = P$ if the vertex set U is clear in its context. Or by setting the edges explicitly, we can also denote a random graph $RG = (U, P = (p_{ij}))$ as $RG = (U, E, P = (p_{ij}))$, where $E = \{(i, j) | p_{ij} > 0\}$.

In the real world, there are a lot of data that can be represented by random graphs.

1.1.1 Equivalence Relations

An equivalence relation is a binary relation between two elements of a set.

Definition 2 Let U be a set. Let ρ be a binary relation on U. Let a, b, c be elements of U. The binary relation ρ on U is called an equivalence relation, if ρ satisfies the properties of reflexivity, symmetry, and transitivity. In other words, for all elements a, b, and c of the set U, the following must hold for ρ:

1. Reflexivity: $(a, a) \in \rho$,
2. Symmetry: if $(a, b) \in \rho$, then $(b, a) \in \rho$, and
3. Transitivity: if $(a, b) \in \rho$ and $(b, c) \in \rho$, then $(a, c) \in \rho$.

Furthermore, the equivalence class of a is the subset of U that contains all elements of U that are equivalent to a under ρ. We denote the equivalence class of a by $[a]$, i.e., $[a] = \{b : (a, b) \in \rho, b \in U\}$. The set of all possible equivalence classes of U by ρ, denoted by $U/\rho = \{[a] : a \in U\}$, is the quotient set of U by ρ. As ρ is a subset of $U \times U$ and each element of ρ can be considered as an edge, (U, ρ) can be considered as a random graph, in which the edge (a, b) exists with probability one if $(a, b) \in \rho$, and zero otherwise.

An information system is represented by an attribute-value table in which rows are labeled by objects of the universe and columns by their attributes. Equivalence relations can be induced by a subset of attributes shown as follows. Denote the universe of objects by U, the set of attributes or features by A, and the set of all possible values of attribute a by V_a. Let P be a subset of A, that is, P is a subset of attributes. The P-indiscernibility relation, denoted by $IND(P)$, defined as

$$IND(P) = \{(x, y) \in U \times U \mid (\forall a \in P)\ a(x) = a(y)\},$$

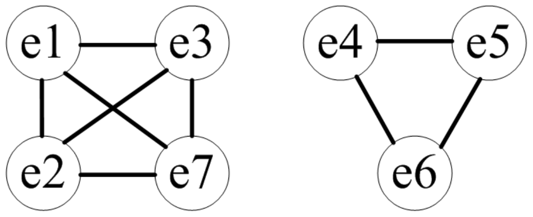


Figure 1.1: An illustration of an equivalence relation as a random graph, in which $\{e1, e2, e3, e7\}$ and $\{e4, e5, e6\}$ are two equivalence classes.

is an equivalence relation. The set of equivalence classes is denoted by $U/IND(P)$ or by U/P, and the equivalence class in U/P is called the P-class. For $x \in X$, let $P(x)$ denote the P-class containing x.

	headache (a)	**pain** (b)	**temperature** (c)	**influenza** (d)
e1	Y	Y	0	N
e2	Y	Y	1	Y
e3	Y	Y	2	Y
e4	N	Y	0	N
e5	N	N	3	N
e6	N	Y	2	Y
e7	Y	N	4	Y

Table 1.1: Influenza data (a)

Example 3 For example, in Table 1.1, a, b, c, and d represent *headache, muscle pain, body temperature* and *influenza*, respectively. Let $P = \{a\}$. Then we have $P(e1) = P(e2) = P(e3) = P(e7) = \{e1, e2, e3, e7\}, P(e4) = P(e5) = P(e6) = \{e4, e5, e6\}$, and

$$IND(P) = \{e1, e2, e3, e7\} \times \{e1, e2, e3, e7\} \cup \{e4, e5, e6\} \times \{e4, e5, e6\}.$$

The corresponding random graph can be seen in Figure 1.1.

1.1.2 Random Graphs Generated by Continuous Attributes

As we have seen in the previous section, discrete attributes can generate equivalence relations, which are special random graphs. In a supervised learning setting, the label information can produce an equivalence relation, which is a random graph. If we want to measure the degree, to which the label information depends on continuous attributes, the viewpoint of understanding a continuous attribute as a random graph makes two obviously different attributes become the same level, and so facilitates to measure the dependency between them. As an example, we show one way to translate a continuous attribute into a random graph.

Example 4 In Table 1.1, if c is understood as a category attribute, then it produces an equivalence relation shown in Figure 1.2 (a), which losses the distance information; if, on the other hand, c is understood as a continuous attribute, and if a random edge is generated between two objects x and y with a probability of $p(x, y)$, where

$$p(x,y) = \begin{cases} e^{-|c_1-c_2|}, & \text{if } e^{-|c_1-c_2|} > 0.2, \\ 0, & \text{otherwise,} \end{cases} \tag{1.1}$$

where $c_1 = c(x)$ and $c_2 = c(y)$, then the generated random graph is shown in Figure 1.2 (b). Note that if c is understood as a continuous attribute, and if a threshold c_{th} (also called cut) is given, then an equivalence relation $\{(x, y) \in U \times U \mid (x \leq c_{th} \wedge x \leq c_{th}) \bigvee (x > c_{th} \wedge x > c_{th})\}$ is generated, and in this way, we consider that all attributes actually work on equivalence relations in C4.5 decision tree [82].

1.1.3 Web Graphs

The Web pages on the Internet are related to one another by hyperlink structure, which form a directed graph. For example, see Figure 1.3. When we consider the reliability of Web sites, the users' behaviors to browse Web pages, and dynamic nature of a Web page, it is better to model the Web graph as a random graph, i.e., links exist in a random way.

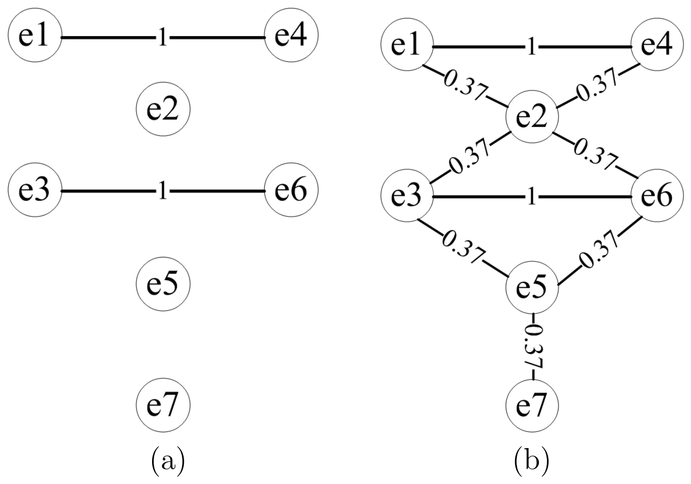


Figure 1.2: The random graph generated by attribute c using Eq. (1.1).

1.2 Basics of Machine Learning

Machine learning is a broad subfield of artificial intelligence. The task of machine learning is to design algorithms and techniques to help computers "learn" useful knowledge from data. Its applications include natural language processing, syntactic pattern recognition, speech and handwriting recognition, search engines, medical diagnosis, finance engineering, bioinformatics, cheminformatics, and so on. For good introductory materials, see [13, 29, 93].

1.2.1 Types of Machine Learning

Different authors use slightly different names for transductive learning and semi-supervised learning. In the following we follow the convention used in [112].

At a general level, there are two types of learning: inductive and transductive. Inductive machine learning methods can extract rules, by which it can handle the unseen data. Transductive learning will be used to contrast inductive learning. A learner is transductive if it only works on the labeled and unlabeled training data, and can label the unlabeled data, but cannot handle

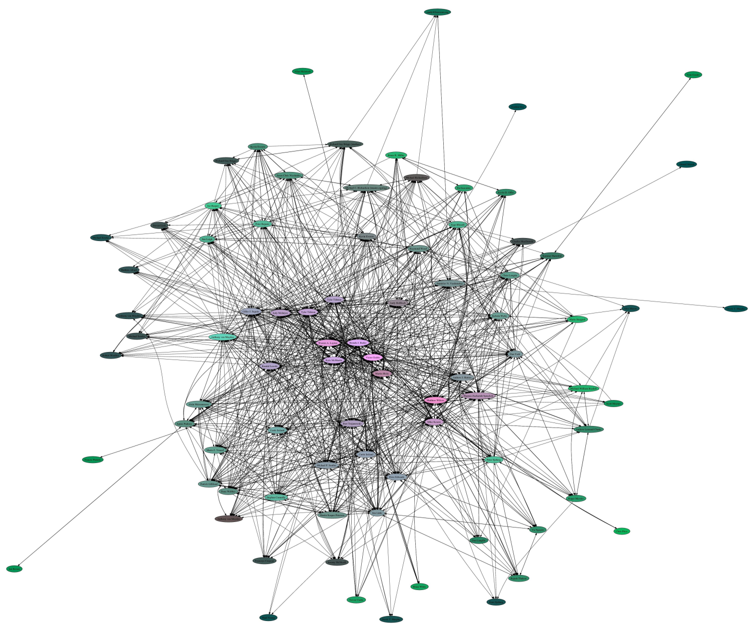


Figure 1.3: The Web graph taken from [2].

unseen data.

Classifying the learning methods by the existence of a teacher to supervise the learning process, there are three types of learning: supervised, semi-supervised, and unsupervised. In supervised learning, a teacher provides a category label or cost for each pattern in a training set, then a learning method uses these labeled data to extract rules for future unlabeled data; in semi-supervised learning, a teacher only labels part of all pattern in the training set, then a learning method uses both these labeled data and unlabeled data in the training set to label the unlabeled data (in a transductive setting) or to extract rules for both unlabeled data and future unlabeled data (in a inductive setting); in unsupervised learning, all the data are unlabeled, i.e., there is no teacher to label the data, then a learning method used all these unlabeled data to learn the intrinsic information hidden in the data.

Decision trees [83, 82], Decision Forest [18] and SVM [93] belong to supervised learning; Transductive SVMs [24, 48] and the early graph-based methods [108, 112] belong to semi-supervised learning; and Principal Component Analysis (PCA) [8], Independent Component Analysis (ICA) [6], clustering [45] belong to unsupervised learning. Ranking methods [76, 111] also belong to unsupervised learning because there is no teacher to label the nodes.

In the next section, we will briefly show the topics closely related to ours.

1.2.2 Graph-based Methods, Ranking and Decision Trees

Graph-based Methods

Graph-based semi-supervised methods define a graph where the nodes are labeled and unlabeled examples in the dataset, and edges (may be weighted) reflect the similarity of examples. These methods usually assume label smoothness over the graph [112]. Graph-based unsupervised methods generate a graph representing the relationship between data points, based on which clustering or dimension reduction can be performed.

Ranking

The importance of a Web page is an inherently subjective matter, which depends on the readers' interests, knowledge and attitudes [76]. 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.

All the mentioned ranking algorithms in this thesis are established on a graph, and will be established on a random graph. For our convenience, we first give some notations. We denote a static graph by $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{(v_i, v_j) \,|\, \text{there is an edge from } v_i \text{ to } v_j\}$ is the set of all edges. Let $I(v_i)$ and $|I(v_i)|$ denote the nodes that link to node v_i and the in-degree of node v_i respectively. d_i denotes the out-degree of node v_i, and also denote the degree of node v_i in an undirected graph. A static graph $G = (V, E)$ is considered as a special random graph $RG = (V, E, P)$, where $P_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise.

Decision Trees

Decision trees are popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent rules, which can be understood by human easily and can be directly used in a database access language like SQL, so that records falling into a particular category may be retrieved.

Decision trees employ a tree structure, in which each node is either a leaf node or a decision node. The leaf node indicates the value of the label, and the decision node determines which subtree will be followed. A decision tree can be used to classify an example by starting at the root of the tree and moving through it until reaching a leaf node, the label of which provides the classification of the instance.

C4.5 has its origins in Hunt's Learning Systems by way of ID3 [81, 82]. The latest version of C4.5 with open source codes is C4.5R8 [83]. The C4.5R8 algorithm uses a divide-and-conquer approach to grow decision trees. To make this thesis self-contained, a brief explanation of the C4.5R8 algorithm is given here. For further details, see [83, 82]. The basic idea of the C4.5R8 decision tree algorithm is similar to that of ID3. It divides the whole training set into smaller subsets until the subsets with all of data corresponding to the same class are created or the number of elements in the subsets is smaller than a threshold. It generates a decision tree from the whole training set. The whole training set corresponds to the root node. Each of the interior nodes including the root node of the tree is labeled by an attribute, while branches that lead from the node are labeled by the value of the attribute. The leaves of the tree correspond to the classes.

The tree construction process is guided by choosing the most informative attribute at each step. In C4.5, some information measures are employed to select the most informative attribute. In the next section, we will introduce the information measures.

1.2.3 Information Measures

There are two information measures will be compared in this thesis. One is the dependency degree $\gamma(C, D)$ [78], which is interesting in its simple suggestive form, and from which our work in measuring the dependency of two random graphs is established. The other is the conditional entropy employed in the attribute selection procedure in C4.5. Note that these two measures can be understood to be defined on equivalence relations, which are special cases of random graphs.

The Dependency Degree $\gamma(C, D)$

In Rough Set Theory [77, 78, 79, 80], an information system is formally set as a four-tuple $S = (U, A, V, f)$, where U represents the universe of objects, A represents the set of attributes or features, V represents the set of possible attribute or feature values, V_a denotes the set of all possible values of attribute a, and f is the information function that maps a given object and a given attribute to a value, i.e.,

$$f : U \times A \to V.$$

By $a(x)$ we denote the value of $f(x, a)$.

For any class X where $X \subseteq U$, and for any subset of attributes P, the P-lower approximation of X, denoted by $\underline{P}(X)$, is defined as

$$\underline{P}(X) = \cup\{Y \in U/IND(P) \,|\, Y \subseteq X\}.$$

Let C and D be two subsets of A. The dependency degree $\gamma(C, D)$ is defined in [78] as

$$\gamma(C, D) = 1/|U| \sum_{X \in U/D} |\underline{C}(X)|, \tag{1.2}$$

where $|U|$ and $|\underline{C}(X)|$ denote the cardinality of the set U and the cardinality of the set $\underline{C}(X)$ respectively. $| \cdot |$ denotes the cardinality of a set without further notice throughout the thesis. From Eq. (1.2), we can see that $\gamma(C, D)$ is actually defined on two equivalence relations $IND(C)$ and $IND(D)$.

The Conditional Entropy

The conditional entropy is well discussed in the literature of Information Theory [25, 105], and is used in the C4.5 decision tree algorithm [82]. The formulation for the conditional entropy is as follows:

$$\begin{aligned} H(D|C) &= -\sum_c \sum_d \Pr(c) \cdot \Pr(d|c) \cdot \log_2(\Pr(d|c)) \\ &= -\sum_c \Pr(c) \cdot \sum_d \Pr(d|c) \cdot \log_2(\Pr(d|c)), \end{aligned} \tag{1.3}$$

where c and d denote the vectors consisting of the values of attributes in C and in D respectively.

Note that an empirical estimation of the conditional entropy can be understood to be defined on two equivalence relations $IND(C)$ and $IND(D)$. This is shown below.

$$D(x) = \{y| \, (\forall a \in D) \, a(x) = a(y)\},$$

$$C(x) = \{y| \, (\forall a \in C) \, a(x) = a(y)\}.$$

$C \cup D$ is also a subset of A, and the equivalence relation IND$(C \cup D)$ partitions U into a disjoint union of some equivalence classes called $(C \cup D)$-classes. Let $x \in U$ such that $C(x) = c, D(x) = d$. Empirically $\Pr(d|c)$ is estimated as $\frac{|(C \cup D)(x)|}{|C(x)|}$, and $\Pr(c, d) = \frac{|(C \cup D)(x)|}{|U|}$. Since $(C \cup D)(x) = C(x) \cap D(x)$, and $C(x)$ is an equivalence class in $IND(C)$, we can say, the empirical estimation of the conditional entropy is defined on two equivalence relations $IND(C)$ and $IND(D)$.

1.3 Motivations and Contributions

A viewpoint of random graphs in the field of machine learning is needed in order to extend currently existing algorithms to a larger extent, since random graphs exist in many situations as we showed in Section 1.1. Moreover, if the data is in essence random, a random graph representation of the underlying data should be more accurate than others, and so is expected to improve the accuracy of some existing algorithms.

With the above considerations, in this thesis, we aim to propose three models in the field of machine learning related to random graphs: Heat Diffusion Models on Random Graphs, Predictive Random Graph Ranking, and Random Graph Dependency. All of these paradigms adopt the viewpoint of random graphs. Heat Diffusion Models on Random Graphs lead to a family of classifiers–Heat Diffusion Classifier on a Graph (*G-HDC*), and a ranking algorithm *DiffusionRank*. Predictive Random Graph Ranking is a framework that incorporates *DiffusionRank*. To provide a basic tool to measure the dependency between two random graphs, we also propose the Random Graph Dependency measure.

The main contributions of this thesis are further described as follows in detail.

- *Proposed the General Heat Diffusion Model on a random graph*
  - ◇ *Heat Diffusion Classifiers* As will be demonstrated, our proposed heat diffusion model can be applied successfully to a classification task.

  - ◇ *DiffusionRank* We will prove that it is a generalization of *PageRank* when the heat diffusion coefficient tends to infinity, and empirically we will show that it achieves the ability of anti-manipulation by setting the heat diffusion coefficient to be finite.

- *Developed a general ranking scheme on a random graph that includes DiffusionRank as a special case*

  - ◇ *We will extend some current ranking algorithms from a static graph to a random graph.*
  - ◇ *We will propose methods to generate a random graph based on the known information about the Web structure.*

- *Provide a tool to measure dependency between two random graphs*

  - ◇ *In the first special case, the proposed measure can speed up C4.5 decision algorithm.*
  - ◇ *In the second special case, the proposed measure can improve the classification accuracy.*
  - ◇ *In the third special case, the proposed measure can help to find two parameters in the heat diffusion classifiers.*

In a summary, the viewpoint of random graphs indeed provides us an opportunity of improving some existing classification algorithms and ranking algorithms.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2

  We review different learning paradigms in this chapter. We include graph-based methods, decision trees, and ranking algorithm in this chapter.

- Chapter 3

  We propose a framework called Graph-based Heat Diffusion Classifiers (*G-HDC*). We will give related background on heat diffusion models, the theoretical framework of our model, detailed analysis of the formulation, and three candidate graph construction methods for *G-HDC*. Note that the contents in this chapter except the materials related to *VHDC* in this chapter are published in [96, 100].

- Chapter 4

  In this chapter, we propose a solution to the incomplete information problem by formulating a new framework called Predictive Random Graph Ranking (*PRGR*), in which we generate a random graph based on the known information about the Web structure. We will extend some current ranking algorithms from a static graph to a random graph. Besides, we will propose a novel ranking algorithm called *DiffusionRank*, motivated by the way that heat flows, which reflects the complex relationship between nodes in a graph (or points on a geometry). Moreover, we will incorporate it in the *PRGR* framework. Note that the contents in this chapter are published in [97, 98, 99]

- Chapter 5

  We propose a general random graph dependency measure. In its first special case, we will show that it can improve the training time of the C4.5R8 decision trees while preserving the classification accuracy of the original C4.5R8. In its second special case, we will demonstrate its success in decision trees on small sample size problems. In its third special case, we will illustrate that it can help *G-HDC* to obtain two free parameters more efficiently. Note that Section 5.2 in this chapter will be published in [101].

- Chapter 6

  We then summarize this thesis and conduct discussions on future work.

We try to make each of these chapters self-contained. Therefore, in several chapters, some critical contents, e.g., model definitions or illustrative figures, having appeared in previous chapters, may be briefly reiterated.

Chapter 2

A Background Review

2.1 Graph-based Methods

Graph-based methods include graph-based semi-supervised methods and graph-based unsupervised methods. We will establish the heat diffusion model on a random graph, based on which we will construct a heat diffusion classifier *G-HDC*. As a graph-based method, *G-HDC* is transductive learning, and is related to manifold learning and heat kernel. In this section, we show a brief literature review on these closely related topics.

2.1.1 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 [92], local linearity used by [85], local covariance matrix used by [95], and local Laplacian approximation used by [11, 74] 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 a Euclidean space, and can be properly mapped by a smooth transformation. While inheriting this idea in our model, we also adopt the concept of thinking globally and fitting locally described by [87]. 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. In our model *DiffusionRank*, heat diffusion models are established on the Web graph, which is considered to lie on a manifold.

2.1.2 Heat Kernel

Heat kernels are a class of kernels. For materials in learning with kernels, see [71, 89]. Some successful applications of heat kernels have been reported recently. In [11], a nonlinear dimensionality reduction algorithm was proposed based on the graph Laplacian whose elements are induced by a local heat kernel approximation. In [51], 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. In [56], 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. In [90], a kernel was constructed by inserting the discrete heat kernel into a continuous kernel, and was successfully applied to *SVM*.

The heat kernel can be explained as a special solution to the heat equation, which is 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 $\mathcal{M}$:

$$\begin{cases} \frac{\partial f}{\partial t} - \mathcal{L}f & = & 0, \\ f(\mathbf{x}, 0) & = & f_0(\mathbf{x}), \end{cases} \tag{2.1}$$

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 $\mathcal{L}f$ is the *Laplace-Beltrami operator* applied to a function f. Eq. (2.1) describes the heat flow throughout a geometric manifold with initial conditions. In local coordinates, $\mathcal{L}f$ is given by

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

[56]. When the underlying manifold is the familiar $m-$dimensional Euclidean Space, $\mathcal{L}f$ is simplified as $\sum\limits_i \frac{\partial^2 f}{\partial x_i^2}$, and the heat kernel takes the Gaussian RBF form

$$K_t(\mathbf{x}, \mathbf{y}) = (4\pi t)^{-\frac{m}{2}} e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{4t}}. \tag{2.2}$$

It is therefore observed that when the underlying manifold is the Euclidean space the Gaussian RBF kernel is a special case of the heat kernel. Previous research work has shown that the heat kernel is a useful tool when a kernel-based algorithm is employed. However, if the underlying manifold is unknown or the explicit expression for Eq. (2.1) is unknown, we cannot find the heat kernel and cannot apply it to a kernel-based algorithm. We consider removing this limitation by not employing a kernel-based algorithm, instead, we consider constructing classifier directly by employing the solution to the heat equation on a graph in a special setting of the initial condition, although this will result in another limitation–the resulting algorithm is transductive.

2.1.3 Transductive Learning

G-HDC is built on a graph and it is actually a transductive algorithm which needs access to the unlabeled data. For a systematic investigation on a semi-supervised learning, refer to [112]. For transductive learning, the kernel matrix is important. For the kernel matrix learning, refer to [57]. Our method is different from the kernel matrix learning in that we try to construct a kernel from data points directly. Along the line of transductive learning, our method is related to [108, 109, 110] . The models in [109, 110] 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 [109, 110] is inappropriate, and is not provided empirically.

Here we give a detailed description about the consistency method proposed in [108], which is a transductive algorithm in the literature most closely related to our proposed *G-HDC*. Let F be a $n \times c$ matrix. Define an $n \times c$ Y with $Y_{ij} = 1$ if $\mathbf{x}_i$ is labeled as j and $Y_{ij} = 0$ otherwise. The consistency method is described as follows.

1. Form the affinity matrix W defined by $W_{ij} = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2}$ if $i \neq j$ and $W_{ii} = 0$.

2. Construct the matrix $S = D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its $(i,i)-$element equal to the sum of the $i-$th row of W.

3. Iterate $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ until converge, where α is a parameter in $(0, 1)$.

4. Let F^* denote the limit of the sequence $\{F(t)\}$. Label each point $\mathbf{x}_i$ as a label $y_i = \arg\max_{j \le c} F^*_{ij}$.

2.2 Ranking

In this section, we show some existing ranking algorithms, which will be extended from a static graph to a random graph. To clearly present the ranking algorithms, we classify ranking techniques into two types: *Absolute Ranking* and *Relative Ranking*. *Absolute Ranking* assigns a real number to each page, and thus gives a total order for all pages. *PageRank* [76] belongs to *Absolute Ranking*. *Relative Ranking* assigns a real number to each pair of pages, and thus for each one given page, determines a total order relative to the given page. *Common Neighbors* [73], *Jaccard's Coefficient* [62], and *SimRank* [47] belong to *Relative Ranking*.

2.2.1 Absolute Ranking

PageRank

As a kind of *Absolute Ranking*, *PageRank* [76] gives the importance rank of Web pages based on the link structure of the Web. 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 [30], the Web is modeled by a directed graph $G = (V, E)$ 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:

$$x_i = \sum_{(j,i) \in E} a_{ij} x_j, \tag{2.3}$$

where a_{ij} is assumed to be $1/d_j$, d_j is the out-degree of page j. Or in matrix terms, $\mathbf{x} = A\mathbf{x}$. When the concept of "random jump" is introduced, the matrix form in Eq. (2.3) is changed to

Model 1:

$$\mathbf{x} = [(1-\alpha)\mathbf{g}\mathbf{e}^T + \alpha A]\mathbf{x}, \tag{2.4}$$

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. $\mathbf{e}^T\mathbf{g} = 1$). Typically, $\alpha = 0.85$ and $\mathbf{e}$ is the vector of all ones.

TrustRank

TrustRank [38] 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. Although there are many variations of *PageRank*, e.g., a family of link-based ranking algorithms in [7], *TrustRank* is especially chosen for comparisons for three reasons: (1) it is designed for combating spamming; (2) its fixed parameters make a comparison easy; and (3) it has a strong theoretical relations with *PageRank* and *DiffusionRank*.

Manifold Ranking

In [111], 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, and the weight matrix is given by the Gaussian Kernel smoothing. While the manifold ranking algorithm achieves an impressive result on ranking images, the biased vector $\mathbf{g}$ and the parameter k in the general personalized *PageRank* in [111] are unknown in the Web graph setting; therefore, we do not include it in the comparisons.

2.2.2 Relative Ranking

In [62], the authors survey an array of methods for *Relative Ranking*, including *Common Neighbors*, *Jaccard's Coefficient*, and *SimRank*. All the methods assign a connection weigh $s(i, j)$ to pairs of nodes v_i and v_j, based on the input graph. The development of similarity search algorithms is motivated by the "related pages" queries of Web search engines and Web document classification [33]. Both applications require a similarity measure, which is computed by either the textual content of pages or the hyperlink structure or both. As in previous work [33, 44, 47], we focus on similarities solely determined by the hyperlink structure of the Web graph.

Common Neighbors

Common neighbor model is based on the idea that two pages are more similar if they have more common neighbors. The common neighbors of v_i and v_j can be defined as $s(i, j) = |I(v_i) \cap I(v_j)|$. It means that if more nodes point to v_i and v_j at the same time, v_i and v_j are more similar. In [73], the author computes the probability of collaboration between scientists in the Los Alamos as a function of the times of their past collaboration. A pair of scientists with more previous collaborators is more likely to collaborate than those with less previous collaborators. In [62], the authors employ common neighbors to predict if any two authors will coauthor papers in the future.

Jaccard's Coefficient

Another commonly used similarity metric is the *Jaccard coefficient*, which is used to measure the probability that both v_i and v_j share a feature. In [62], the authors take features to be neighbors in graph, which corresponds to the measure $s(i,j) = |I(v_i) \cap I(v_j)|/|I(v_i) \cup I(v_j)|$. In this thesis we utilize this approach as well to measure the similarity between two pages in the Web.

SimRank

SimRank is introduced in [47] to formalize the intuition that "two pages are similar if they are referenced by similar pages." Numerically this is specified by defining the *SimRank* score $s(i,j)$ of two pages v_i and v_j as the fixed point of the following recursive definition,

$$s(i,j) = \begin{cases} 1, & i = j, \\ 0, & |I(v_i)||I(v_j)| = 0, i \neq j, \\ K \sum\limits_{u \in I(v_i), v \in I(v_j)} s(u,v), & \text{otherwise}, \end{cases}$$

for some constant decay factor $C \in (0,1)$, where $K = \frac{C}{|I(v_i)||I(v_j)|}$. The *SimRank* iteration starts with $s(i,j) = 1$ for $i = j$ and $s(i,j) = 0$ otherwise.

Heat Diffusion Ranking

Heat diffusion is a physical phenomena. In a medium, heat always flow from position with a high temperature to position with a low temperature. Heat kernel is used to describe the amount of heat that one point receives from another point. Inspired by heat diffusion, we will propose *DiffusionRank*, which belongs to both *Absolute Ranking* and *Relative Ranking*. When we consider the temperature distribution to be a ranking result, *DiffusionRank* belongs to *Absolute Ranking*; when we consider the pair relations by heat kernel, it belongs to *Relative Ranking*.

2.3 Decision Trees

C4.5R8 employs a gain criterion and a gain ratio criterion to select the most informative attribute at each subset of training cases. If the algorithm is run with the gain criterion, then for every condition attribute a and for the set D of class attributes, the information gain $G(D, \{a\})$ is computed as follows. $G(D, \{a\}) = H(D) - H(D|\{a\})$ when a is a discrete attribute, and $G(D, \{a\}) = H(D) - H(D|\{a\}) - \log_2(N-1)/|U|$ when a is a continuous attribute, where N is the number of distinct values of the attribute a, and $\log_2(N-1)/|U|$ is used to reduce the bias towards the continuous attribute. The attribute that has the maximum gain among all the condition attributes is chosen. If, instead, the algorithm is run with the gain ratio criterion, then for every condition attribute a, the information gain ratio is computed by the formula $\frac{G(D,\{a\})}{H(\{a\})}$.

C4.5 is successful in terms of its speed and accuracy [63]. Further extensions of C4.5 are proposed in [106] to address problems of classifying partially specified instances. A different paradigm for the criterion of building trees is proposed in [64], in which decision trees are built by minimizing the sum of the misclassification and test costs. Further run-time improvement is achieved in [86], and significant improvements in classification accuracy can be achieved by growing an ensemble of trees and letting them vote for the most popular class [18]. Bagging [17], boosting [88, 34, 35] and randomization of the internal decisions [28] are three methods that generate a diverse ensemble of classifiers by manipulating the training data to the base algorithm, and an experimental comparison of these three methods can be found in [28]. Further developments along the line of the ensemble of decision trees can be found in [18, 5]. In [18], some theoretical properties of random forests are given, and it is shown that using a random selection of features to split each node yields error rates that compare favorably to Adaboost [35]. In [5], first order random forests with complex aggregates are shown to be an efficient and effective approach towards learning relational classifiers that involve aggregates over complex selections.

Although the classification accuracy of decision trees can be improved considerably by forming an appropriate decision forest, the testing time is proportional to the number of trees in the decision forest, and so is greatly increased. Because of this, it is favorable to improve the classification accuracy of one single decision tree. On the other hand, the computation time of the feature selection in an interior node is proportional to that of the conditional entropy, the number of data and the number of attributes left in the current node. As a result, if the dataset is large and there are many features, the computation time of the feature selection will be large. Under such a consideration, an information measure that can be computed faster is expected in order to reduce the feature selection time.

A few attempts at improving the use of continuous attributes in C4.5 can be found in [32, 83]. In [32], the efficiency of selecting a decision threshold (cut) for continuous-valued attributes is improved, and in [83], a penalty is applied to tests on continuous attributes. However, few papers consider the inaccuracies in handling continuous attributes in C4.5. In the following example, we will analyze these. Discretization is an alternative method that has been discussed in the literature for improving the use of continuous attributes. For a systematic study of discretization methods with the history of their development and their effects on classification including C4.5, see [66]. Although a new discretization method can be obtained by the proposed random graph dependency and further improvements can be expected, we focus on handling the continuous attributes by inheriting the idea employed in C4.5, i.e., choosing the cut such that the information gain (or gain ratio) is maximal, in order to distinguish the single factor (in improving the accuracy of C4.5) of the replacement of the conditional entropy by the random graph dependency from other factors such as the discretization and the boosting. By doing so, we hope to show that how much improvement can be made through this single factor.

We focus on improving the speed of C4.5 by one form of the proposed random dependency measure and improving its accuracy by another form of the novel measure.

2.4 Information Measures

In [102, 103], rules are classified into two types: one-way rule and two-way rule. The dependency degree $\gamma(C, D)$ and the conditional entropy are measures for one-way rule.

2.4.1 The Dependency Degree $\gamma(C, D)$

$\gamma(C, D)$ expresses the percentage of objects that can be correctly classified into the D-class by employing attribute C. γ becomes a traditional measure in Rough Set Theory [36].

Varying the measure, a family of γ-like statistics is introduced in [36], the idea of which is to count the number of errors. The problem of extending γ to incomplete information systems is considered in the literature. The simplest method is to remove examples with unknown values. Replacing every missing value with the set of all possible values is another method [65]. Introducing the similarity relation and completion of an incomplete information system is a more accurate way to handle missing values [52, 53, 60].

In a recent approach, γ is employed to generate rules in a case study, and achieves high accuracy rates and less number of rules [41].

2.4.2 The Conditional Entropy $H(D|C)$

The Shannon entropy function is applied [59, 68, 72] to measure the "information content" of the data in the columns of an attribute set. They extend the idea to develop a measure that, given a finite table T, quantifies the amount of information the columns of C contain about D. This measure is the conditional entropy [37].

The conditional entropy is referred to as an information dependency measure, denoted by $H_{C \to D}$ [27], and a variety of arithmetic inequalities is developed for this measure. The conditional entropy is well discussed in the literature of Information Theory [25, 105], and is used in the C4.5 decision tree algorithm [82], and the latest version C4.5R8 [83].

2.5 A Brief Book Review

In [70], there is a systematic discussion on random graphs for statistical pattern recognition. The topics include various graph construction methods such as Delaunay Triangulation [58], Alpha Hulls [69], KNN Graphs, Relative-Neighbor Graphs [46], Gabriel Graphs [16, 107], etc. This book also incorporates a lot of graph-based materials such as clustering, image segmentation, outlier detection, and etc, but it is seldom related to heat diffusion, ranking and random graph measure, which are our focus in this thesis. Nevertheless, it is possible to feed the heat diffusion classifiers by the above mentioned graphs. The readers are recommended to read this book if they hope to extend the scope of the heat diffusion classifiers.

Chapter 3

Heat Diffusion Model on a Random Graph

The aim of this chapter is to establish a framework called Graph-based Heat Diffusion Classification (*G-HDC*). The framework consists of two stages:

- **Random Graph Generation Stage**–The first stage engages the data cloud to construct a random graph representing the data relationship in a local way. Statistical and geometrical methods can be applied to generate this random graph. Currently we focus on geometrical methods.
- **Heat Diffusion Calculation Stage**–The second stage takes the random graph output and the label information, and then calculates the temperatures of unlabeled data after a fixed time period, based on a heat diffusion model on a random graph. The temperatures are employed to classify the unlabeled data.

This chapter is organized as follows. In Section 3.1, we show the motivations. In Section 3.2, we construct the heat diffusion model on a random directed graph, and in Section 3.5, we propose the Graph-based Heat Diffusion Classifiers (*G-HDC*). To feed *G-HDC*, we propose three candidate random graphs in Section 3.3 and Section 3.4. In Section 3.6, Section 3.7, and Section 3.8, we provide detailed interpretations of the heat diffusion model. Then

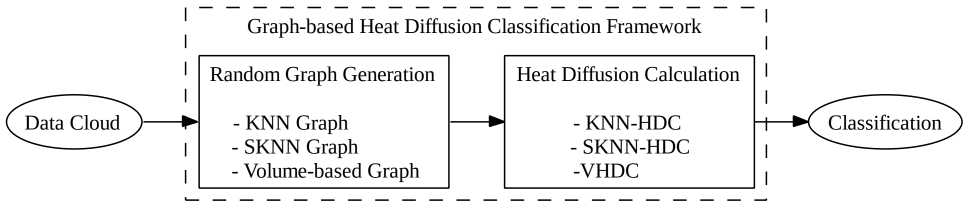


Figure 3.1: The graph-based heat diffusion classification framework.

in Section 3.10, we show the experimental results. Section 3.11 provides a summary.

3.1 Motivations

The successful applications [51, 56, 90] of the heat kernel motivate us to investigate the heat equation Eq. (2.1) and its solutions. Traditional numerical methods for solving differential equations are in fact established on a triangulation mesh or on a grid, and they have been classified into three main approaches: finite element (FE), boundary element (BE), and finite difference (FD) methods [10]. 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 [23, 91]).

If a simplicial surface S with vertex set V can be constructed from the data cloud, then by the results in [14], the discrete Laplace-Beltrami operator $\mathcal{L}$ of a simplicial surface S can be established as:

Definition 5 For a function $f : V \to R^m$ on the vertices, the value of $\mathcal{L}f : V \to R^m$ ar $x_i \in V$ is

$$\mathcal{L}f(x_i) = \sum_{x_j \in V:(x_i,x_j)\in E_D} \rho(x_i, x_j)(f(x_i) - f(x_j)), \tag{3.1}$$

where E_D is the edge set of a Delaunay triangulation of S and the weights are given by

$$\rho(x_i, x_j) = \begin{cases} \frac{1}{2}(\cot\alpha_{ij} + \cot\alpha_{ji}) & \text{for interior edges} \\ \frac{1}{2}\cot\alpha_{ij} & \text{for boundary edges} \end{cases} \tag{3.2}$$

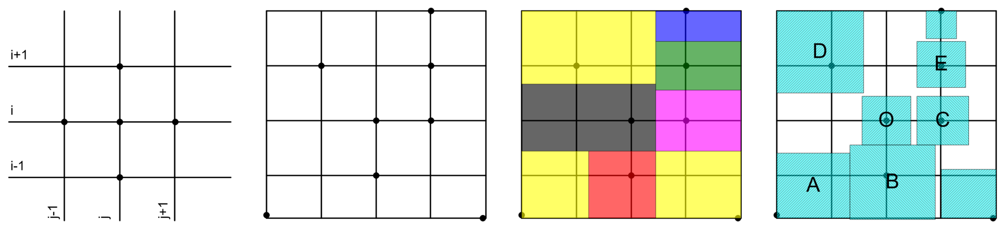
Here α_{ij} (and α_{ji} for interior edges) are the angles opposite the edge (x_i, x_j) in the adjacent triangles of the Delaunay triangulation.

However, there is not a clear picture of constructing the mentioned simplicial surface S when faced a cloud of data points in an unknown geometry. For the same reason, we cannot construct the triangle mesh directly in our model. It is true that meshing algorithms exist and are widely employed in scientific computation, for example, see [9, 26]. They are highly refined for low-dimensional point clouds and generate meshes for FE and BE. However, in situations where the data is quite high-dimensional and sparse, we are unaware of any effective meshing algorithm and therefore we cannot use the FE and BE methods.

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 in Eq. (2.1) becomes

$$\begin{cases} \frac{\partial f}{\partial t} - \frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} & = & 0, \\ f(x, y, 0) & = & f_0(x, y). \end{cases} \tag{3.3}$$

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 temperature at position x_i, y_j at time t_k. The grid on the plane is shown in Fig. 3.2(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 $|i - i'| + |j - j'| = 1$.



(a) (b) (c) (d)

Figure 3.2: (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.

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:

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

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

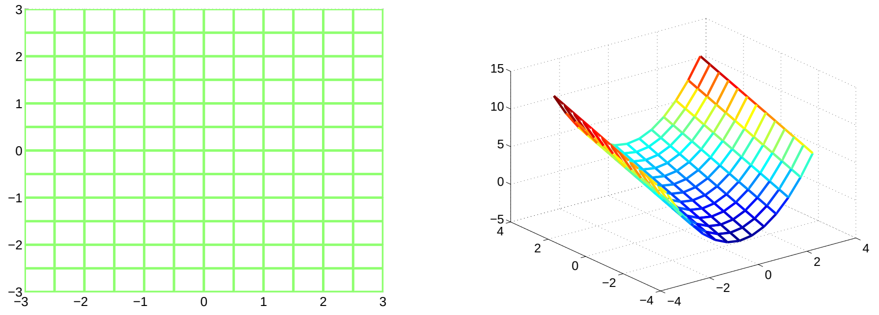
$$\left.\frac{\partial^2 f}{\partial y^2}\right|_{(i,j,k)} \approx \frac{f(i,j-1,k) - 2f(i,j,k) + f(i,j+1,k)}{(\Delta y)^2}.$$

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

$$\begin{aligned} \tfrac{f(i,j,k+1)-f(i,j,k)}{\Delta t} &= \tfrac{f(i-1,j,k)-2f(i,j,k)+f(i+1,j,k)}{(\Delta x)^2} + \tfrac{f(i,j-1,k)-2f(i,j,k)+f(i,j+1,k)}{(\Delta y)^2} \\ &= \tfrac{[(f(i-1,j,k)-f(i,j,k))+(f(i+1,j,k)-f(i,j,k))}{d^2} + \tfrac{(f(i,j-1,k)-f(i,j,k))+(f(i,j+1,k)-f(i,j,k))]}{d^2} \end{aligned} \tag{3.4}$$

The above two discretization methods are successful when the underlying triangulation mesh or the grid can be constructed successfully, however, in the real data analysis, the graph constructed from the data points is irregular, i.e., it is neither a triangulation mesh or the grid. Even worse, we often face the following problems where we cannot employ these two discretization methods.

1. The manifold is unknown;



(a) (b)

Figure 3.3: (a) The grid on the two-dimensional Euclidean space. (b) The grid on the curved Euclidean space.

2. The differential equation expression is unknown even if the manifold is known.

We aim to solve these problems using a Heat Diffusion Model (*HDM*) on a graph by considering the above problems while preserving the common features in Eq. (3.1) and Eq. (3.4) such that the neighbor x_j of x_i affects x_i in proportional to the difference $f(x_j) - f(x_i)$. In fact, we try to establish the heat diffusion model on a graph by going back the Fourier law, on which the original differential heat diffusion equation is established. The novel heat diffusion model on the graph are expected to grasp some nature of the data, and are expected to lead to a novel classifier called *Heat Diffusion Classifier* (*HDC*) are competitive to some state-of-the-art transductive classifiers.

The intuition behind is that the heat equation on a graph is considered as an approximation to Eq. (2.1). For example, heat diffusion behaviors in the grid shown in Fig. 3.3(a) should be the same as those in Fig. 3.3(b) if the weights between two grid nodes in these two grids are the same. Consequently, by the bridge of the grid considered as a special graph, the difficulty of solving the heat diffusion in the curved manifold in Fig. 3.3(b) is reduced. Next we consider establishing a diffusion model on a random directed graph.

3.2 Heat Diffusion Model on a Random Directed Graph

Consider a directed random graph $G = (V, E, P)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $P = (p_{ij})$, where p_{ij} is the probability that edge (v_i, v_j) exists, and $E = \{(v_i, v_j) \,|\text{there is an edge from } v_i \text{ to } v_j \text{ and } p_{ij} > 0\}$ is the set of all edges.

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 $f_j(t) - f_i(t)$.

As a result, the expected 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

$$\frac{f_i(t+\Delta t) - f_i(t)}{\Delta t} = \alpha \sum_{(j,i)\in E} p_{ji}(f_j(t) - f_i(t)) \tag{3.5}$$

To find a closed form solution to Eq. (3.5), we express it as a matrix form:

$$\frac{\mathbf{f}(t+\Delta t) - \mathbf{f}(t)}{\Delta t} = \alpha H \mathbf{f}(t), \tag{3.6}$$

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

$$H_{ij} = \begin{cases} -\sum_{k:(k,i)\in E} p_{ki}, & \text{if } j = i; \\ p_{ji}, & \text{if } (j,i) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

In the limit $\Delta t \to 0$, Eq. (3.6) becomes

$$\frac{d}{dt}\mathbf{f}(t) = \alpha H \mathbf{f}(t), \tag{3.8}$$

Solving Eq. (3.8), we get

$$\mathbf{f}(t) = e^{\alpha t H}\mathbf{f}(0) = e^{\gamma H}\mathbf{f}(0), \tag{3.9}$$

where $\gamma = \alpha t$, and $e^{\gamma H}$ is approximated by

$$e^{\gamma H} = I + \gamma H + \frac{\gamma^2}{2!}H^2 + \frac{\gamma^3}{3!}H^3 + \cdots . \tag{3.10}$$

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.

For the sake of computational considerations, $e^{\gamma H}\mathbf{f}(0)$ can be approximated as $(I + \frac{\gamma}{p}H)^p\mathbf{f}(0)$, where p is a large integer. The latter can be calculated by iteratively applying the operator $(I + \frac{\gamma}{p}H)$ to $\mathbf{f}(0)$.

3.3 Candidate Random 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 [56], 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 [11, 85, 87, 92]. 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 probability p_{ji} equal to $e^{-d^2(i,j)/\beta}$ if $\mathbf{x}_j$ is one of the K nearest neighbors of $\mathbf{x}_i$.

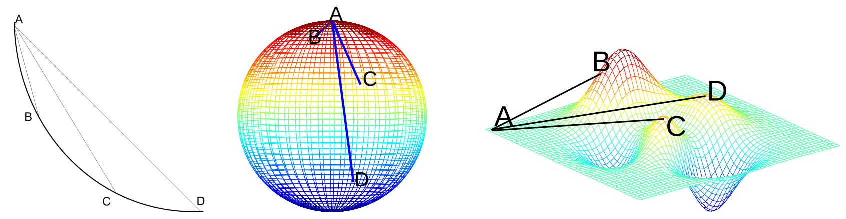


Figure 3.4: Illustrations on a manifold on which the shorter line is more accurate.

Note that there are $K * (M + N)$ directed edges in the resulting graph. 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 3.4. 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 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. Set edge probability p_{ji} equal to $e^{-d^2(i,j)/\beta}$ if $d(i, j)$ is among the smallest $K * (M + N)/2$ undirected edges.

The third candidate will be shown in next section. It is motivated by more accurately modeling the heat diffusion equation by a volume representation.

3.4 Volume-based Heat Diffusion Model on a Graph

We consider the representation ability of each node. In a manifold, there are infinitely many nodes on the manifold, but only a finite number $M + N$ of nodes are known and form the graph. We can assume that there is a small patch $P(j)$ 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. The volume of the small patch $P(j)$ is $V(j)$.

3.4.1 Establishment of VHDM

In this section, we try to establish the heat diffusion model by employing Fourier's law, which states that the rate of heat flow through a homogenous solid is directly proportional to the area of the section at right angles to the direction of heat flow, and to the temperature difference along the path of heat flow.

Suppose, at time t, each unit volume containing i receives an amount $HM(i, j, t, \Delta t)$ of heat from its neighbor j during a period of Δt. Then according to Fourier's law, we assume that

1. The heat $HM(i, j, t, \Delta t)$ should be proportional to the time period Δt and the temperature difference $f_j(t) - f_i(t)$.

2. The amount of heat that patch $P(j)$ diffuses to the unit volume containing i is proportional to the surface area $S(i)$ of the unit volume.

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 one-dimensional Euclidean space, as described in Eq. (2.2). Consequently we further assume that $HM(i, j, t, \Delta t)$ is proportional to $e^{-w_{ij}^2}$, the amount of heat that a unit heat source at node j transferred to node i, which is a fact in one-dimensional Euclidean space. In addition, the temperature in the small patch $P(j)$ at time t is almost equal to $f_j(t)$ because every unseen node in the small patch is near node j, and so the amount of heat in patch $P(j)$ is proportional to $V(j)$. As a result,

$$HM(i,j,t,\Delta t) = \alpha S(i) e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))V(j)\Delta t.$$

The amount of heat in the unit volume containing i is equal to $f_i \cdot 1$. The heat difference in this unit volume should be $f_i(t+\Delta t) - f_i(t)$, which is caused by the sum of the heat that it receives from all its neighbors and the small patches around these neighbors. This is formulated as

$$f_i(t+\Delta t) - f_i(t) = \alpha \sum_{(j,i)\in E} S(i) e^{-w_{ij}^2/\beta}(f_j(t) - f_i(t))V(j)\Delta t \tag{3.11}$$

The solution to Eq. (3.11) is $\mathbf{f}(t) = e^{\gamma H}\mathbf{f}(0)$, where $H = (H_{ij})$, and

$$H_{ij} = \begin{cases} -\sum_{k:(k,i)\in E} S(i) e^{-w_{ik}^2/\beta} V(k), & j = i, \\ S(i) e^{-w_{ij}^2/\beta} V(j), & (j,i) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

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. Currently we define $V(i)$ to be mean of $1/n$ and the normalized volume of the hypercube whose side length is the distance between node i and its nearest neighbor. Formally,

$$V(i) = \eta \min_{j:(j,i)\in E} w_{ij}^{\nu}/2 + 1/2n, \tag{3.13}$$

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

In the above discussions, we established *VHDM* by physical intuitions. Next we will show a mathematical justification for the introduction of volumes, and as a by-product, we find a way to calculate the contact area $S(i)$.

3.4.2 Necessity of Introducing Volumes

In this section, we show that it is necessary to introduce the concept of volumes from three aspects.

Justification by Integral Approximations

In this section, except for volumes, we follow the approximation techniques employed in [11]. Note that when all the volumes are equal, the last approximation in Eq. (3.15) becomes the case in [11].

It turns out that in an appropriate coordinate system $K_t(\mathbf{x},\mathbf{y})$ on a manifold is approximately the Gaussian:

$$K_t(\mathbf{x},\mathbf{y}) = (4\pi t)^{-\frac{m}{2}} e^{-||\mathbf{x}-\mathbf{y}||^2/4t}(\phi(\mathbf{x},\mathbf{y}) + O(t)), \tag{3.14}$$

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 [11, 84].

It is well known that the solution to Eq. (2.1) can be expressed as $f(\mathbf{x},t) = \int_M K_t(\mathbf{x},\mathbf{y}) f_0(\mathbf{y})$. From $\mathcal{L}f(\mathbf{x},t) = \frac{\partial f(\mathbf{x},t)}{\partial t}$, we have

$$\begin{aligned}
\mathcal{L}f(\mathbf{x}_i,t) &\approx (-f(\mathbf{x}_i,t) + f(\mathbf{x}_i,t+\Delta t))/\Delta t \\
&\approx (-f(\mathbf{x}_i,t) + \textstyle\int_M K_{\Delta t}(\mathbf{x}_i,\mathbf{y}) f(\mathbf{y},t))/\Delta t \\
&\approx (-f(\mathbf{x}_i,t) + (4\pi\Delta t)^{-\frac{m}{2}} \textstyle\int_M e^{-||\mathbf{x}_i-\mathbf{y}||^2/4\Delta t} f(\mathbf{y},t))/\Delta t \\
&\approx (-f(\mathbf{x}_i,t) + (4\pi\Delta t)^{-\frac{m}{2}} \sum_{(j,i)\in E} e^{-||\mathbf{x}_i-\mathbf{x}_j||^2/4\Delta t} f(\mathbf{x}_j,t) V(j))/\Delta t,
\end{aligned} \tag{3.15}$$

where volumes are considered the partition of M, and the last approximation is based on the definition of the integral, which will become an equality if $\bigcup_{\{j|(j,i)\in E\}} P(j) = M$, $P(j) \cap P(k) = \emptyset$, and $\max V(j) \to 0$. To satisfy these three conditions, the volume of a patch should occupy the manifold as much as possible while each pair of patches are not intersected. When the number of data is large enough, $\max V(j)$ will be small enough. This motivates us to define the volume in Eq. (3.13).

However, in practice, the above three conditions cannot be satisfied, and so an error arises in the last approximation. To correct this approximation error, $(4\pi\Delta t)^{-\frac{m}{2}}$ is set to be $S(i)$. By the knowledge that a constant temperature distribution at time t will also result in a constant temperature distribution at time $t + \Delta t$, we have $1 \approx S(i) \sum\limits_{(j,i)\in E} e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/4\Delta t} V(j)$, and so

$$S(i) \approx 1/(\sum_{(j,i)\in E} e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/4\Delta t} V(j)). \tag{3.16}$$

This is the definition of the surface area of node i. The intuition is that the larger the volumes of its neighbors the less the surface that is left to i. This intuition comes from the observation that, in Fig. 3.2(d), the larger volumes of squares A,B,C,D, and E force the surface of O to be smaller.

Let G be the KNN graph. By this volume-based heat diffusion model, we can see that the underlying random graph is

$$P = (p_{ij}), p_{ij} = \begin{cases} S(i)e^{-w_{ij}^2/\beta} V(j), & \text{if } j \text{ is one neighbor of } i\text{'s;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

With such a random graph, *G-HDC* in Section 3.5 becomes the Volume-based Heat Diffusion Classifier *VHDC*.

Justification by the Definition of a Manifold

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 [56].

Definition 6 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 from U_i to $\phi_i(U_i)$. 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 joining the small patches together.

Justification by Variation of Density

When 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 Fig. 3.2(c), the whole space is covered by small patches, and in Fig. 3.2(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.

3.4.3 Calculation of the Intrinsic Dimension 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 of 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 [94].

Thus, instead of PCA, we choose the maximum likelihood estimation method proposed in [61]. Apart from avoiding the problems with PCA just mentioned, this method also has the advantage that it fits the proposed heat diffusion classifier naturally because the graph is constructed by K nearest neighbors in Eq. (3.17), where the parameter K is the same as the one employed in dimension estimation by the maximum likelihood estimation. In addition, this method helps to reduce the complexity of searching the parameter K in that we can discard those Ks by which the estimated dimensions are greater than the number of attributes or are less than one.

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 [61], as follows,

$$\hat{m}_K(x) = \left[\frac{1}{K-1}\sum_{j=1}^{K-1}\log\frac{T_K(x)}{T_j(x)}\right]^{-1}. \tag{3.18}$$

To avoid overflowing during calculations when $T_j(x)$ is very small, we slightly change Eq. (3.18) to the following:

$$\hat{m}_K(x) = \left[\frac{1}{K-1}\sum_{j=1}^{K-1}\log\frac{T_K(x)+\epsilon}{T_j(x)+\epsilon}\right]^{-1}. \tag{3.19}$$

ϵ is set to be 0.0000001. Then $\nu = \frac{1}{n}\sum\limits_{i=1}^{n}\hat{m}_K(x_i)$. In fact, we observe that an arbitrary selection of the parameter ϵ in the interval $[0.0000001, 0.001]$ cannot produce much difference on the estimation of the dimension, and so we need not pay much care on the selection of ϵ.

3.5 Graph-based Heat Diffusion Classifiers (*G-HDC*)

Based on the closed form solution in Eq. (3.9), 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 labeled 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 labeled data set contain M_k points in class C_k so that $\sum_{k=1}^{c} M_k = M$. Let an unlabeled data set contains N unlabeled samples, represented by $\mathbf{x}_i$ $(i = M+1, M+2, \ldots, M+N)$.

For a given 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 labeled data in class C_k to be one and all other data to be zero, then calculate the amount of heat that each unlabeled data receives from the labeled data in class C_k. Finally, we assign the unlabeled 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 unlabeled data set by a graph construction algorithm.

[**Step 2: Compute the Heat Kernel**] Using Eq. (3.7) and Eq. (3.10), find the heat kernel $e^{\gamma H}$.

[**Step 3: Compute the Heat Distribution**] Let

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

$k = 1, 2, \ldots, c$, 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, $\mathbf{f}^k(t) = e^{\gamma H}\mathbf{f}^k(0)$, $k = 1, 2, \ldots, c$. $\mathbf{f}^k(0)$ 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 $\mathbf{f}^k(t)$ means that the heat distribution at time t is caused by the initial temperature distribution $\mathbf{f}^k(0)$.

[**Step 4: Classify the data**] For $l = 1, 2, \ldots, N$, compare the p-th ($p = M + l$) components of $\mathbf{f}^1(t), \mathbf{f}^2(t), \ldots, \mathbf{f}^c(t)$, 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 unlabeled data $\mathbf{x}_p$, then classify the unlabeled data $\mathbf{x}_p$ to class C_k.

Note that the initial temperature setting is shown in $\mathbf{f}^k(0)$ in the Step 3, and that the stop time t is hidden in $\gamma = \alpha t$, a super-parameter determined by cross-validation.

As an example of Step 1, in Fig. 3.5(a), we show 2,000 points on a 2-dimensional spiral manifold which is embedded into 3-dimensional space. In Fig. 3.5(b), we show the neighborhood graph approximation of the spiral manifold, which contains 1,000 points drawn from the 2,000 points in Fig. 3.5(a), and in which each node has 3 neighbors. As shown in Fig. 3.5(b), there are two classes. In Step 2, the heat diffuses from the labeled data to the unlabeled data along the graph, and consequently, the heat flows along the spiral manifold. In Step 3, if the unlabeled data point is closer to one class in the sense that it receives more heat in total from this class of data, then the unlabeled data point is classified into this class; otherwise, it is classified into the other class.

In section 3.3, we showed three candidate random graphs: KNN graph, SKNN graph, and Volume-based graph, which result in corresponding classifiers: *KNN-HDC*, *SKNN-HDC*, and *VHDC*.

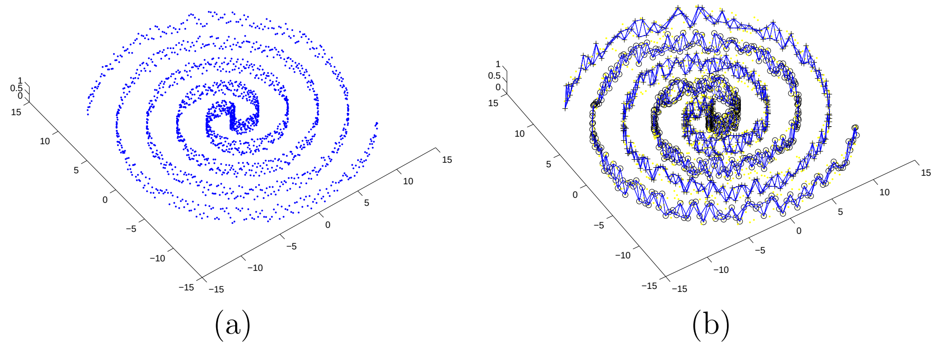


Figure 3.5: An illustration of the spiral manifold and its graph approximation. (a) The 2,000 data points on a spiral manifold. (b) Neighborhood graph of the 1,000 data points on the spiral manifold.

3.6 Correspondences between the Heat Diffusion Model on Graphs and that on Manifolds

In Section 3.1, we have shown that the heat diffusion kernel $K_t(\mathbf{x}, \mathbf{y})$ is a special solution to Eq. (2.1) 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.

In the case that the underlying geometry is unknown or its heat kernel cannot be approximated in the same way as used by [56], 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. (2.1),

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

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 $\mathbf{y}$; 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.

3.7 Roles of the Parameters

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

3.7.1 Local Heat Diffusion Controlled by β

In Section 3.4.1, we assumed that the heat diffuses in the pipe in the same way as it does in the one-dimensional Euclidean space. Next we will justify this assumption. In *VHDM* in Section 3.4.1, 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 approximation in Eq. (3.14) 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, the amount of heat that node i receives from its neighbor j is proportional to $e^{-w_{ij}^2/\beta}$.

3.7.2 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 $e^{\gamma H}\mathbf{f}(0) = I\mathbf{f}(0) = \mathbf{f}(0)$, 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 *G-HDC* 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, $(e^{\gamma H} - I)/\gamma\mathbf{f}(0)$ behaves the same as $e^{\gamma H}\mathbf{f}(0)$ as a classifier. Then we can take the limit over $(e^{\gamma H} - I)/\gamma\mathbf{f}(0)$, and we obtain

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

We consider $H\mathbf{f}(0)$ as the singular behavior of *G-HDC* in the limit $\gamma \to 0$.

3.7.3 Stability of *KNN-HDC* with Respect to Parameters

It is easier to analyze *KNN-HDC* than to analyze *SKNN-HDC* and *VHDC*, because the number of nonzero elements in H in *SKNN-HDC* is not fixed, and the expressions in H in *VHDC* is complicated. In this section, we will analyze the stability of *KNN-HDC* with respect to parameters.

There are three free parameters in *KNN-HDC*. If the parameters in a model are 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 and γ are not sensitive to the classifier *KNN-HDC*.

Since $e^{\gamma H}$ is continuous on β and γ in the sense that small changes in these parameters result in a small change in $e^{\gamma H}$, *KNN-HDC* is not sensitive to these two parameters if they are changed slightly.

The existence of the derivatives of $e^{\gamma H}$ with respect to β and γ can be seen in the following:

$$\frac{de^{\gamma H}}{d\gamma} = e^{\gamma H}H, \tag{3.20}$$

$$\frac{de^{\gamma H}}{d\beta} = \gamma e^{\gamma H}\frac{dH}{d\beta}, \tag{3.21}$$

$$\frac{dH}{d\beta} = (\frac{dH_{ij}}{d\beta}), \tag{3.22}$$

$$\frac{dH_{ij}}{d\beta} = \begin{cases} -\sum_{k:(k,i)\in E} e^{-w_{ik}^2/\beta} w_{ik}^2 \beta^{-2}, & \text{if } j = i; \\ e^{-w_{ij}^2/\beta} w_{ij}^2 \beta^{-2}, & \text{if } (j,i) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{3.23}$$

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 and γ, we can say that $e^{\gamma H}$ is stable with respect to these parameters, and so is $e^{\gamma H}\mathbf{f}(0)$.

If H is symmetric, then we can estimate an upper bound for these derivatives. First of all, we claim that the $i-$row $j-$column element in $e^{\gamma H}$ means the amount of heat that i receives from a unit heat source at j. So physically we claim that $e^{\gamma H}$ is a non-negative matrix. Next, we show that the sum of each row in $e^{\gamma H}$ is equal to one. Let $\mathbf{1}$ and $\mathbf{0}$ be the vector of all ones and the vector of all zeros respectively. Then $H\mathbf{1} = \mathbf{0}$. According to Eq. (3.10), we have

$$e^{\gamma H}\mathbf{1} = I\mathbf{1} + \gamma H\mathbf{1} + \frac{\gamma^2}{2!}H^2\mathbf{1} + \ldots = \mathbf{1},$$

which means that the sum of each row in $e^{\gamma H}$ is equal to one. Consequently we can assume that each row in $e^{\gamma H}$ is a vector $(a_1, a_2, \ldots, a_n)$ satisfying $a_i \geq 0$ and $\sum_i a_i = 1$. Let $(b_1, b_2, \ldots, b_n)^T$ be a column in H. Then each element in $\frac{de^{\gamma H}}{d\gamma}$ is of the form $(a_1, a_2, \ldots, a_n)(b_1, b_2, \ldots, b_n)^T$ by Eq. (3.20). By the Hölder's inequality ($p = 1, q = \infty$), we have

$$(a_1, a_2, \ldots, a_n)(b_1, b_2, \ldots, b_n)^T \leq (\sum_i |a_i|) \max_i |b_i| = \max_i |b_i| \leq K,$$

which means that each element in $\frac{de^{\gamma H}}{d\gamma}$ is not greater than K. Similarly if $\gamma e^{-w_{ij}^2/\beta} w_{ij}^2 \beta^{-2} \leq 1$ for all i and j, then each element in $\frac{de^{\gamma H}}{d\beta}$ is not greater than K.

For the parameter K, it has an unstable effect on the classifier *KNN-HDC*. 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 *KNN-HDC* 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 and β, 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.

3.8 Necessity of Introducing the Heat Diffusion Model in Classification

It is not absolutely necessary to have a physical model behind a learning algorithm. However, the situation is different for the heat diffusion model since many learning algorithms can be interpreted by the heat diffusion model theoretically although not empirically.

Justification by Practice Considerations

In *G-HDC*, if $\beta \rightarrow +\infty$, the graph is of the form as shown in Fig. 3.2(a), which means each node has four neighbors, and if the volume of each node is set to be one, then Eq. (3.11) becomes Eq. (3.4). 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 Fig. 3.2(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.** This is shown in Section 3.4.2.

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.

When both volumes and the contact surfaces are constant, *KNN-HDC* is a special case of *VHDC*. In the next section, we first show that *KNN* can be considered as a special case of *KNN-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 [13] can be considered as a special case of *KNN-HDC* (when $K = n - 1$, and γ is small).

When the parameter γ is small, we can approximate $e^{\gamma H}$ in Eq. (3.10) by its first two items, i.e., $e^{\gamma H} \approx I + \gamma H$, then in *KNN-VHDC*, $\mathbf{f}^k(t) = e^{\gamma H}\mathbf{f}^k(0) \approx \mathbf{f}^k(0) + \gamma H\mathbf{f}^k(0)$. As the constant γ and the first item $\mathbf{f}^k(0)$ have no effect on the classifier, *KNN-HDC* possesses a similar classification ability to that determined by the equation $\mathbf{f}^k(t) = H\mathbf{f}^k(0)$. As a classifier, $H\mathbf{f}^k(0)$ will not be affected by an arbitrary scaling on each row of H, and so the surface factor can be ignored in $H\mathbf{f}^k(0)$. This result will be used in the next two subsections.

3.8.1 KNN-HDC and Parzen Window Approach

First we review the Parzen Windows non-parametric method for density estimation, using Gaussian kernels. When the kernel function $H(u)$ is a multivariate normal kernel, a common choice for the window function, by the estimate of the class-conditional densities for class C_k and Bayes's theorem, we have [13]: the density at the point x is

$$\widetilde{p}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{(2\pi h^2)^{d/2}}e^{-\frac{||\mathbf{X}-\mathbf{X}_i||^2}{2h^2}}. \tag{3.24}$$

When applying it for classification, we need to construct the classifier through the use of Bayes's theorem. This involves modeling the class-conditional densities for each class separately, and then combining them with priors to give models for the posterior probabilities which can then be engaged to make classification decisions [13]. The class-conditional densities for class C_k can be obtained by extending Eq. (3.24):

$$\widetilde{p}(\mathbf{x}|C_k) = \frac{1}{M_k}\sum_{i:C_{k_i}=C_k}\frac{1}{(2\pi h^2)^{d/2}}e^{-\frac{||\mathbf{X}-\mathbf{X}_i||^2}{2h^2}}, \tag{3.25}$$

where the priors can be estimated by $\widetilde{p}(C_k) = \frac{M_k}{M}$. By Bayes's theorem, we get

$$\widetilde{p}(C_k|\mathbf{x}) = \frac{\sum_{i:C_{k_i}=C_k} e^{-||\mathbf{X}-\mathbf{X}_i||^2/2h^2}}{Mp(\mathbf{X})(2\pi h^2)^{d/2}}. \tag{3.26}$$

If we set $K = n-1$, and if γ is small, then the graph constructed in Step 1 will be a complete graph, and the matrix H in Eq. (3.12) becomes

$$H_{ij} = \begin{cases} -\sum_{k\neq i} e^{-w_{ik}^2/\beta}, & j = i; \\ e^{-w_{ij}^2/\beta}, & j \neq i. \end{cases} \tag{3.27}$$

Then, in *KNN-HDC*, the heat $f_p^k(t)$ that unlabeled data $\mathbf{x}_p$ receives from the data points in class C_k will be equal to $\sum_{i:C_{k_i}=C_k} e^{-||\mathbf{X}_p-\mathbf{X}_i||^2/\beta}$, which is the same as Eq. (3.26) if we let $\gamma = 1/Mp(\mathbf{x})(2\pi h^2)^{d/2}$, and $\beta = 2h^2$. This means that, when the window function is a multivariate normal kernel, the Parzen Window Approach can be considered as a special case of *KNN-HDC* (when $K = n-1$, and γ is small in *KNN-HDC*).

3.8.2 KNN-HDC and *KNN*

If β tends to infinity, then $-w_{ij}^2/\beta$ will tend to zero, and the matrix H in Eq. (3.12) becomes

$$H_{ij} = \begin{cases} -O_i, & j = i; \\ 1, & \mathbf{x}_j \text{ is one neighbor of } \mathbf{x}_i; \\ 0, & \text{otherwise.} \end{cases} \tag{3.28}$$

Here O_i is the outdegree of the point $\mathbf{x}_i$ (note that the indegree of the point $\mathbf{x}_i$ is K). Then, in *KNN-HDC* when γ is small, the heat $f_p^q(t)$ that unlabeled data x_p receives from the data points in class C_q will be equal to $f_p^q(t) = \sum_{i:l_i=C_q} 1 \ = K_q$, where K_q is the number of the labeled data points from class C_q, which are the K nearest neighbors of the unlabeled data point $\mathbf{x}_p$. Note that when $N = 1$, i.e., when the number of unlabeled data is equal to one, $\sum_{q=1}^{c} K_q = K$. According to Step 4, we will classify the unlabeled data $\mathbf{x}_p$ to the class C_k such that $f_p^k(t) = K_k$ is the maximal among all $f_p^q(t) = K_q$. This is exactly what *KNN* does, and so *KNN* can be considered as a special case of *KNN-HDC* (when β tends to infinity, $N = 1$ and γ is small in *VHDC*).

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 *KNN-HDC* is less than $2P$ if β is infinity and volumes are constant. It should be tighter if appropriate parameters for *KNN-HDC* are found.

3.8.3 *G-HDC* and Some other Popular Algorithms

As explained in [12, 31], a number of popular algorithms such as SVM, Ridge regression, and splines may be broadly interpreted as regularization algorithms with different empirical cost functions and complexity measures in an appropriately chosen Reproducing Kernel Hilbert Space (RKHS). For a Mercer kernel $K : X \times X \rightarrow R$, there is an associated RKHS $\mathcal{H}_K$ of functions $X \rightarrow R$ with the corresponding norm $|| \, ||_K$. Given a set of labeled examples $(\mathbf{x}_i; \mathbf{y}_i)$, $i = 1, \ldots, l$ the standard framework estimates an unknown function by minimizing

$$f^* = \arg \min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\mathbf{x}_i, y_i, f) + \gamma ||f||_K^2,$$

where V is some loss function, such as squared loss $(y_i - f(\mathbf{x}_i))^2$ for RLS or the hinge loss function $\max[0, 1 - y_i f(\mathbf{x}_i)]$ for SVM. Penalizing the RKHS norm imposes smoothness conditions on possible solutions. The classical Representer Theorem states that the solution to this minimization problem exists in $\mathcal{H}K$ and can be written as

$$f^*(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}, \mathbf{x}_i). \tag{3.29}$$

If K takes the Gaussian RBF $e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{\sigma^2}}$, then $f^*(\mathbf{x})$ in Eq. (3.29) is the solution of the following heat diffusion equation on a $m-$dimensional Euclidean space:

$$\begin{cases} \frac{\partial f}{\partial t} - \mathcal{L}f &= 0, \\ f(\mathbf{x}, 0) &= f_0(\mathbf{x}), \end{cases} \tag{3.30}$$

where $f_0(\mathbf{x}) = (4\pi t)^{\frac{m}{2}} \sum_{i=1}^{l} \alpha_i \delta(\mathbf{x} - \mathbf{x}_i)$ and $4t = \sigma^2$. This amounts to the solution in Eq. (3.29) can be obtained by solving a heat diffusion equation with a special initial temperature setting.

It is interesting to mention that the Representer Theorems for the Laplacian Regularized Least Squares and the Laplacian SVM (manifold regularization) are similar:

$$f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \tag{3.31}$$

where $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$ denotes the u unlabeled examples. This also means that the solution for Gaussian RBF kernel can be obtained by solving a heat diffusion equation with a special initial temperature setting.

Therefore we can say that many learning algorithms can be interpreted by the heat diffusion model theoretically. This shows the necessity of introducing the heat diffusion model. However, the problem is where and how we set the initial conditions in the heat diffusion equation in what kind of space. This chapter shows what we can achieve by a simple setting of the initial conditions—set the temperature to be one at the training data points (before we can find an optimization method to find the best initial setting, we have to adopt this "simple and stupid" setting). Although such an attempt may not achieve much accuracy improvement, it inspires a broad research space for future investigations on the heat diffusion equation since we feel we have still not fully fulfilled its potential for its applications in classification tasks.

3.9 Comparisons with Related Work

The outside appearance of $e^{\gamma H}$ is the same as that in [51, 90]; however, the numerical value of $e^{\gamma H}$ in our thesis is quite different from [51, 90] when we consider the Volume-based KNN graph. The heat kernel in [51, 56, 90] is applied to a large margin classifier; in contrast, our heat kernel is employed directly to construct a classifier. Although our method is limited to the transductive learning setting, it is interesting and challenging to apply the proposed $e^{\gamma H}$ to inductive algorithms such as *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 thesis, and so the empirical comparison on heat kernels is not provided.

The success in [56] 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.

We are interested in comparing the consistency method [108], which is described in Chapter 3. Although our model adopts a different approach, there is an overlap between our solution and that in [108]. The overlap happens when γ is small in our model, while α is small and the normalization is not performed in [108]. 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 [108] is not performed, and when 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 3.7.2.

3.10 Experiments

In this section, we make an experimental comparison between our methods with some closely related methods. The Parzen Window Approach (*PWA*) and *KNN* are special cases of *KNN-HDC*. Consistency Method (*CM*) is closely related with our methods. These related algorithms will be compared with three heat diffusion classifiers *KNN-HDC*, *SKNN-HDC* and *VHDC*. Along the line of *SVM*, transductive *SVM* algorithms (*UniverSVM*[24] and *SVMLight*[48]) are popular. Employed as baselines, the recent one, *UniverSVM*, will also be compared to our method.

We test our models against *PWA*, *KNN*, *USVM*, and *CM*. on one synthetic dataset and ten datasets from the UCI Repository [42]. Since discrete attributes and the problem of missing values are out of the scope of this chapter, we simply remove all the discrete attributes and remove all the cases that contain missing values. Table 3.1 describe the resulting datasets we use. Spiral-1000 is a synthetic dataset, which is illustrated in Fig. 3.5(b). 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.

Table 3.1: Datasets description

Dataset	**Spiral-1000**	**Credit-a**	**Iono**	**Iris**	**Diabetes**	
Cases	1000	666	351	150	768	
Classes	2	2	2	3	2	
Features	3	6	34	4	8	
Dataset	**Breast-w**	**Waveform**	**Wine**	**Anneal**	**Heart-c**	**Glass**
Cases	683	300	178	898	303	214
Classes	2	3	3	5	2	6
Features	9	21	13	6	5	9

We employ the Gaussian RBF kernels for *UniverSVM*. We obtain the free parameters in *PWA*, *KNN*, *UniverSVM CM*, *KNN-HDC*, *SKNN-HDC* and *VHDC* via nine-fold cross-validations on the training dataset including the testing data without labels.

The values shown in Table 3.2 are the mean accuracy of ten runs by dividing the data into 10% for training and 90% for testing and their variances. Note that the results are quite different if we choose the best values in each run in hindsight, i.e., the testing data with label is given when we choose the parameters. The better results of *VHDC* over *KNN-HDC* and *SKNN-HDC* show the necessity of introducing the volume representation of a node in a graph. From the results, we also observe that both *KNN-HDC* and *VHDC* outperforms *PWA* and *KNN* in accuracy, as we expected.

The overall results on the ten benchmark datasets indicate that our approach *VHDC* is competitive with the Consistency Method and *UniverSVM*

Table 3.2: Mean accuracy on the 11 datasets achieved by ten runs by dividing the data into 10% for training and 90% for testing

Dataset	**PWA**	**KNN**	**USVM**	**CM**	**KNN-HDC**	**SKNN-HDC**	**VHDC**
Spiral-1000	81.2	78.2	66.6	80.5	92.7	85.9	**94.1**
Variance	0.56	0.92	1.73	0.69	0.61	0.75	0.65
Credit-a	52.3	**64.4**	54.9	55.1	61.6	54.0	63.8
Variance	0.96	1.00	0.15	0.00	1.53	0.97	0.93
Iono	67.5	79.7	**85.6**	71.4	80.3	68.1	80.2
Variance	1.73	1.38	1.66	2.02	1.67	0.74	1.19
Iris	**94.3**	91.1	93.6	93.5	91.7	89.0	92.4
Variance	0.89	2.18	1.09	1.08	2.18	0.75	2.15
Diabetes	65.1	**67.8**	65.1	65.6	67.1	67.7	67.2
Variance	0.71	0.56	0.09	0.32	0.88	1.23	0.88
Glass	54.3	51.2	49.9	54.7	55.5	47.6	**56.4**
Variance	1.16	1.12	3.79	1.71	1.36	1.63	1.18
Breast-w	95.3	95.7	65.1	**96.3**	95.7	94.9	96.0
Variance	0.26	0.12	0.06	0.15	0.21	0.28	0.15
Waveform	74.7	72.0	69.0	**76.4**	74.4	73.1	73.9
Variance	1.31	1.52	2.21	1.18	1.23	1.14	1.17
Wine	61.6	66.5	36.6	63	63.6	**66.8**	63.4
Variance	2.76	2.68	0.79	2.70	2.05	1.80	2.40
Anneal	**76.2**	75.8	45.8	**76.2**	75.6	72.3	75.3
Variance	0.00	0.54	1.34	0.00	0.50	0.52	0.68
Heart-c	55.0	60.5	54.6	52.1	59.3	57.7	**61.5**
Variance	0.59	0.52	0.32	0.44	1.32	1.14	1.12
Average	70.68	72.99	62.44	71.35	74.32	70.65	**74.93**

on problems without any *a priori* knowledge. The better results on the synthetic dataset show that *VHDC* fits problems with a manifold structure especially well. Despite its success, *VHDC* is still not perfect. Next we discuss three aspects:

1. **Dependency on Distance Measure**. The boolean attributes in the Zoo dataset are considered as continuous attributes. We observe that PWA, *KNN*, *CM*, *HDC* and *VHDC* achieve a 40.6% classification rate, and perform more poorly than *UniverSVM* (with a 97.2% classification rate) 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 encouraging: *VHDC* achieves a 97.1% classification rate, the same as what *UniverSVM* achieves. This example shows that *VHDC* relies heavily on the local distance, and so a suitable feature extraction method may help to increase its accuracy. A possible solution is to employ the semi-supervised metric learning method proposed in [104].

2. **Local Minimum**. Note that $VHDC$ generalizes both PWA and *KNN*. But it is observed that, on dataset Iris, $VHDC$ performs worse than PWA; on dataset Wine, $VHDC$ performs worse than *KNN*. We think that there exits local minimum problems hidden in the cross-validation search for best parameters in *VHDC*. A possible way to this kind of problem is to understand the initial temperature distribution as a random field, to estimate the covariance of the random field at time t, and then to minimize an appropriately defined error measure including both the fitting error and the variance.

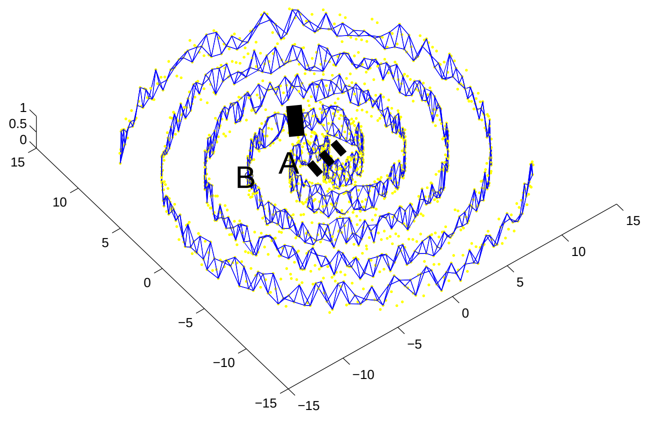


Figure 3.6: An illustration showing that the equal setting of initial temperatures is not perfect. Only two data points A and B are labeled, the equal initial temperature setting on these two points will result in classification errors. The decision boundary will be the bar while the dashed line should be the ideal decision boundary.

3. **Initial Temperature Setting**. According to the discussions in Section 3.8.3, by appropriately setting the initial temperatures, the heat diffusion model can interpret many learning algorithms. Currently the initial temperature is set to be one, and this simple setting will result in some errors. For example, in Fig. 3.6, a higher initial temperature in point B than that in A is expected in order to achieve the best decision boundary, as indicated by the dashed line. This problem is quite open now.

3.11 Summary

The proposed *G-HDM* has the following advantages: 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. For *VHDC*, it has an extra advantage that it can model the effect of unseen points by introducing the volume of a node. While *VHDC* is a generalization of *KNN-HDC*, which 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.

Chapter 4

Predictive Random Graph Ranking on the Web

The incomplete information about the Web structure causes inaccurate results of various ranking algorithms. In this chapter, we propose a solution to this problem by formulating a new framework, called Predictive Random Graph Ranking (*PRGR*), in which we generate a random graph based on the known information about the Web structure. The random graph can be considered as the predicted Web structure, on which ranking algorithms are expected to be improved in accuracy. For this purpose, we extend some current ranking algorithms from a static graph to a random graph. Besides, we propose a novel ranking algorithm called *DiffusionRank*, motivated by the way that heat flows, which reflects the complex relationship between nodes in a graph (or points on a geometry). Moreover, we incorporate it in the *PRGR* framework.

The rest materials are organized as follows. In the next section, we provide our motivations for this work. In Section 4.2, we describe our predictive strategy. In Section 4.2.4, we propose HDM and incorporate it with the predictive strategy. In Section 4.3, we propose *DiffusionRank*. In Section 4.4, we describe the data sets that we worked on and the experimental results for the *PRGR* framework. In Section 4.5, we demonstrate experimental results for *DiffusionRank*. In Section 4.6, we provide a summary and present possible future work.

4.1 Motivations

While the *PageRank* algorithm [76] has proven to be very effective for ranking Web pages, inaccurate *PageRank* results are induced because of the incomplete information about the Web structure and because of Web page manipulations by people for commercial interests.

The incomplete information problem is caused by the following phenomena:

1. *The Web is dynamic (temporal dimension)*–The link structure evolves temporally. Some links are created and modified, while others are destroyed.

2. *The observer is partial (spatial dimension)*–For different observers (or crawlers), the Web structure may be different.

3. *Links are different (local dimension)*–Not all out-links are created equal. Some out-links are more significant than others. For example, some people may tend to put the most important link on the top of their pages.

The manipulation problem is also called the Web spam, which refers to hyperlinked pages on the World Wide Web, created with the intention of misleading search engines [38]. It is reported that approximately 70% of all pages in the .biz domain and about 35% of the pages in the .us domain belong to the spam category [75]. The reason for the increasing amount of Web spam is explained in [75]: 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, Keyword Stuffing and

Link Stuffing are being used widely [38, 75]. 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 problem. Machine learning methods are employed to handle the keyword stuffing. To successfully apply machine learning methods, we need to dig out some useful textual features for Web pages, to mark part of the Web pages as either spam or non-spam, then to apply supervised learning techniques in marking other pages. For example, see [20, 75]. Link analysis methods are also employed to handle the link stuffing problem. One example is the *TrustRank* [38], a link-based method, in which the link structure is utilized so that human labeled trusted pages can propagate their trust scores through their links. We focus on the link-based method for combating the Web spam problem in this thesis.

For the problem of the incompleteness and impreciseness of the Web structure, we have three contributions in this chapter. First, we provide a random graph perspective for the above phenomena. In temporal dimension, unknown links are modeled by random links; in spatial dimension, in order to generate a more accurate structure, different perspectives can be combined by means of a random graph; and in local dimension, the different orders of links are seen to have random importance.

Secondly, by the random graph perspective, we establish *PRGR* framework. As illustrated in Figure 4.1, the framework consists of two stages:

- **Random Graph Generation Stage**–The first stage engages the temporal, spatial and local link information to construct a random graph that can better model the Web. Statistical and other methods can be applied to generate this random graph that can better approximate the incomplete Web.

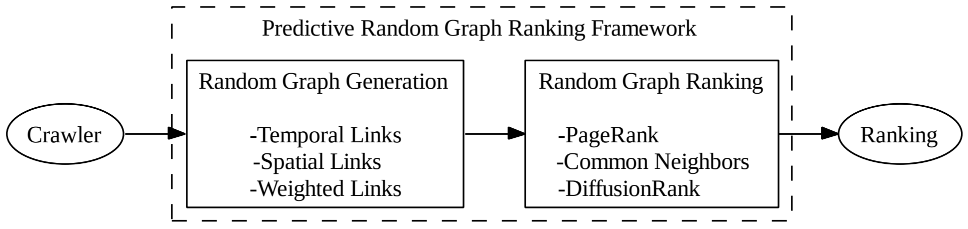


Figure 4.1: The predictive random graph ranking framework.

- **Random Graph Ranking Stage**–The second stage takes the random graph output and then calculates the ranking result based on a candidate ranking algorithm, such as, *PageRank*, *Common Neighbors, Jaccard's Coefficient, SimRank,* etc.

The intuition in the *PRGR* framework is that: The more accurately we know the structure of the Web, the more accurately we can infer about the Web.

Thirdly, we propose the novel *DiffusionRank* as another candidate in the *PRGR* framework. *DiffusionRank* proves to have anti-manipulation effects. 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 or even negative values for spam pages) to avoid Web spam.

The input-independent feature of *PageRank* can be further explained as follows. $P = [(1-\alpha)\mathbf{g}\mathbf{1}^T + \alpha A]$ 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 the largest eigenvalue is 1 and no other eigenvalue whose absolute value is equal to 1, which is guaranteed by the Perron Theorem [67]. Let $\mathbf{y}$ be the eigenvector corresponding to 1, then we have $P\mathbf{y} = \mathbf{y}$. Let $\{\mathbf{x}_k\}$ be the sequence generated from the iterations $\mathbf{x}_{k+1} = P\mathbf{x}_k$, and $\mathbf{x}_0$ is the initial input. If $\{\mathbf{x}_k\}$ converges to $\mathbf{x}$, then $\mathbf{x}_{k+1} = P\mathbf{x}_k$ implies that $\mathbf{x}$ must satisfy $P\mathbf{x} = \mathbf{x}$. Since the only maximum eigenvalue is 1, 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 $\mathbf{x}_0$.

In our opinion, $\mathbf{g}$ and α are objective parameters determined by the users' behaviors and preferences. A, α and $\mathbf{g}$ are the "true" Web structure. While A is obtained by a crawler and the setting $\alpha = 0.85$ is accepted by the people, we think that $\mathbf{g}$ should be determined by a user behavior investigation, something like [3]. Without any prior knowledge, $\mathbf{g}$ has to be set as $\mathbf{g} = \frac{1}{n}\mathbf{1}$.

TrustRank model does not follow the "true" Web structure by setting a biased $\mathbf{g}$, but the effects of combating spamming are achieved in [38]; *PageRank* is on the contrary in some ways. We expect a ranking algorithm that has an effect of anti-manipulation as *TrustRank* while respecting the "true" Web structure as *PageRank*.

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 the 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 well connected, and thus x and y have a high similarity in the sense of a high mutual connection.

4.2 Predictive Strategy

In this section, we first show the origin of the idea of the predictive strategy, then we show that the concept of a random graph is necessary, and next we show how a random graph can be generated in various situations. This forms the first stage of the framework *PRGR*, and can be found in Section 4.2.2 and Section 4.2.3. In Section 4.2.4 and Section 4.3, we extend several ranking models from static graphs to random graphs. These are the second stage of the *PRGR* framework.

4.2.1 Origin of Predictive Strategy

In [97], we propose a predictive ranking technique to improve the accuracy of *PageRank* through the estimation of the incomplete information caused by partial crawling on the Web. The more accurate the estimated Web structure is, the better results the *PageRank* will achieve. We extend the basic idea in [97] from *PageRank* to a collection of ranking algorithms, from temporal incomplete information to spatial uncertainty and weighted links.

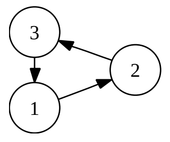


Figure 4.2: A static graph.

4.2.2 From Static Graphs to Random Graphs

The concept of a random graph is necessary for *PageRank*. For example, the graph in Figure 4.2 may be encountered by a crawler in the early stage if all the unvisited nodes are ignored. If we employ Eq. (2.3) and use the power iterative method to solve the page rank problem, then we will suffer the problem of divergence unless the entire initial values of x_i $(i = 1, 2, 3)$ take the value of 1/3, which usually cannot be found in practice. However, if we employ Eq. (2.4), the power iterative method will converge. This is because the modified matrix in Eq. (2.4) is a positive stochastic matrix, and so 1 is its largest absolute eigenvalue and no other eigenvalue whose absolute value is equal to 1, which is guaranteed by the Perron Theorem [67]. Behind Eq. (2.4), we can see that the Web graph has been modeled as a random graph, in which the original link exists with a probability of α, and there is a link that connects each pair of pages with a probability of $1 - \alpha$.

Furthermore, in the following, we discuss three situations: (1) *temporal links*, (2) *spatial links* and (3) *weighted links*, in which the concept of a random graph is also necessary.

Random Graph Generated Temporal Links

If we want to model the estimation about the temporal links, the concept of a random graph is necessary. In Figure 4.2, when the time continues, the crawler will visit more nodes, but at the current time, the links (called temporal links) from the currently unvisited node are unknown. In general, it is difficult to estimate the temporal link structure accurately; however, some elementary estimation is possible. Currently we only estimate the in-degree of each node in the set of nodes that have been found, and thus some information about the link structure can be inferred statistically. For more discussions, see Section 4.2.3.

Random Graph Generated by Several Graphs

Several crawlers may visit some pages at different times and from different starting sites, and a link may exist for one crawler, but disappear for another. This causes the partial observer problem–the Web graph is viewed differently from different points. Suppose that different Web graphs $G_i = (V_i, E_i), (i = 1, 2, \ldots, N)$ are obtained by N different observers (or crawlers). We can combine these different graphs and generate a random graph $RG = (V, P)$, where $V = \cup_{i=1}^{N} V_i, P = (p_{ij}), p_{ij} = n(i,j)/N, n(i,j)$ is the number of the graphs where the link (i,j) appears. The intuition behind is that the more a link is reliable, the more times different observers will find it.

Random Graph Generated by Weighted Links

We have observed that some out-links are more significant than others. As an example, we may model the out-link significance by the exponential decay rule: e^{1-k} where k is the out-link order number from a particular page. Then a random graph generated by this rule will be $P = (p_{ij})$ where $p_{ij} = 0$ if there is no link from i to j, and $p_{ij} = e^{1-k(i,j)}$ if j is the $k(i,j)$-th out-link from i. By doing so, the significance of different out-links from a particular page is distinguished. The original static graph is changed to a random graph.

In the next subsection, we emphasis on the problem of dangling nodes, which is caused by the nature of the dynamic Web. This problem is handled by predicting the link structure as a random graph. To sum up, it is necessary to extend the current ranking algorithms from a static graph to a random graph.

4.2.3 From Visited Nodes to Dangling Nodes

Why We Consider Dangling Nodes

Pages that either have no out-link or have no known out-link are called dangling nodes [30]. In [76], the authors suggested simply removing the pages without out-link and the links pointing to them. After doing so, it is suggested that they can be "added back in" without significantly affecting the results. However the situation is changed now and the dangling nodes problem has to be handled more accurately and directly.

On the one hand, we can see that the *PageRank* algorithm depends on part of the Web structure, and that the visited fraction of the whole Web page by a crawler becomes smaller and smaller as the Web continues to grow. More and more dangling nodes appear because of the difficulty of sampling the entire Web. In [76], the authors reported that they have 51 million URLs not yet downloaded when they have 24 million pages downloaded. In [39], dynamic pages are estimated to be 100 times more than static pages, and in [30], the authors point out in their experiment that the number of uncrawled pages still far exceeds the number of crawled pages, and that there are an essentially infinite number of URLs which is estimated to be at least 64^{2000}. These experimental results and theoretical analyses mean that in reality, the huge number of unvisited pages tends to exceed the ability of a crawler.

On the other hand, some dangling pages are worthy of ranking because they contain important information. In such a situation, ranking those pages that only have been found may enrich the content of a search engine. As an example, a search engine may return the users the URLs of unvisited pages with high ranking scores. Moreover, including dangling nodes in the overall ranking may have significant effect not only on the rank value of non-dangling pages but also on the rank order. This will be shown in the Experiment section.

How to Classify Dangling Nodes

In the following, we follow the ideas in [30] in analyzing the reasons that cause the dangling nodes, and we classify dangling nodes into three classes according to these reasons.

Dangling nodes of class 1 ($DNC1$) are defined as nodes that have been found but have not been visited. One reason to produce such kind of dangling nodes is that the Web is so large that we cannot visit all the pages; another reason is that new Web pages are always being created.

Dangling nodes of class 2 ($DNC2$) are defined as nodes that have been tried but not visited successfully. The reason to produce dangling nodes of class 2 is that some pages may exist before, but are now damaged or are in maintenance, or they are protected, or they are wrongly created.

Dangling nodes of class 3 ($DNC3$) are defined as nodes that have been visited successfully but from which no out-link is found. Dangling nodes of class 3 exist because there are many files on the Web with no hyperlink structure.

How to Handle Dangling Nodes

We first partition all the nodes V of the graph G ($|V| = n$) into three subsets: D^0, D^1, and D^2, where D^0 ($|D^0| = m$) denotes the subset of all nodes that have been crawled successfully and have at least one out-link; D^1 ($|D^1| = m_1$) denotes the set of nodes of $DNC3$; D^2 ($|D^2| = n - m - m_1$) denotes the set of nodes of $DNC1$. Nodes of $DNC2$ are ignored here. The main idea of handling dangling nodes is to handle different nodes in different ways. In the following, we describe our method in detail.

1. We predict the real in-degree $d^-(v_i)$ by the number of found links $fd^-(v_i)$ from visited nodes to node v_i. With the breadth-first crawling method, we assume that the real number of links from all nodes in V to node v_i is proportional to the number of found links $fd^-(v_i)$ from visited nodes to node v_i, and further

we assume that

$$d^-(v_i) \approx \frac{n}{(m+m_1)} \cdot fd^-(v_i)(i = 1, 2, \ldots, n).$$

This assumption is based on the intuition that a crawler's ability of finding new links to a given node v_i depends on the density of these links. The density of these links to the node v_i is equal to $d^-(v_i)/n$. The crawler has found $fd^-(v_i)$ such kind of links when it has crawled m nodes, and we consider $\frac{fd^-(v_i)}{(m+m_1)}$ as an approximate estimate of the density of these links. Following this, the above approximate equality holds.

2. With the approximate in-degree $d^-(v_i)$, we can re-arrange the matrix. All the found links $fd^-(v_i)$ are from the nodes in D^0, and the remaining links $d^-(v_i) - fd^-(v_i)$ are from the nodes in D^2 (it is impossible that some of these links are from the nodes in D^1). Since we infer the number of the remaining links only out of $m + m_1$ visited nodes and the total number nodes is n, there is a risk of over-prediction. To prevent the over-prediction, we adopt a confidence index (or certainty) $(m + m_1)/n$ about this estimation, and so we expect $(d^-(v_i) - fd^-(v_i))(m + m_1)/n$ remaining links. Without any prior information about the distribution of these remaining links, we have to assume that they are distributed uniformly from the nodes in D^2 to the node v_i, i.e., these remaining links are shared by all the nodes in D^2. So matrix A^T representing the random graph can be divided into six blocks as shown below

$$A^T = \begin{pmatrix} C & X & M \\ D & Y & N \end{pmatrix},$$

where $(C, D)^T$ is used to model the known link structure from D^0 to V. Let $C = (c_{ij}), D = (d_{ij})$, then

$$c_{ij}, d_{i,j} = \begin{cases} 1, & \text{there is a link from } j \text{ to } i, \\ 0, & \text{otherwise.} \end{cases}$$

In A^T, $(X, Y)^T$ will be defined later, $(M, N)^T$ is used to model the link structure from D^2 to V, and is defined as follows:

$$\begin{pmatrix} M \\ N \end{pmatrix} = \begin{pmatrix} l_1 & 0 & 0 & 0 \\ 0 & l_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_n \end{pmatrix} \mathbf{1}_{n\times(n-m-m_1)},$$

where $l_i = \frac{(d^-(v_i)-fd^-(v_i))(m+m_1)}{n(n-m-m_1)}$, $(i = 1, 2, \ldots, n)$, $n - m - m_1$ means that the expected remaining in-links $(d^-(v_i)-fd^-(v_i))(m+m_1)/n$ are shared uniformly by all nodes in D^2.

3. When we want to model the users' teleportation, we assume that the users will jump to node v_i with a probability of g_i when they get bored in following the actual links. So the matrix modeling the teleportation is $\mathbf{g}\mathbf{e}^T$. We denote here $(g_1\, g_2\, \ldots\, g_n)^T$ by $\mathbf{g}$.

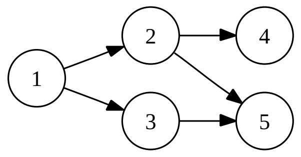
4. When the user encounters a node of $DNC3$, there is no out-link that the user can follow. In this case, we assume that the same kind of teleportation as in step 3 will happen, and so the matrix $(X, Y)^T$ in step 2 is used to model the link structure from D^1 to V and it is assumed to be

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} g_1 & 0 & 0 & 0 \\ 0 & g_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g_n \end{pmatrix} \mathbf{1}_{n\times m_1}.$$

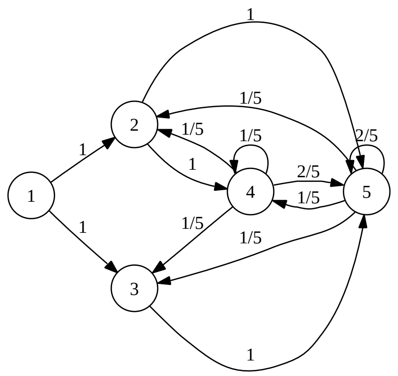
5. We further assume that α is the probability of following an actual out-link from a page, $1-\alpha$ is the probability of taking a "random jump" rather than following a link. Then the random matrix P is modeled as

$$P^T = (1-\alpha)\mathbf{g}\mathbf{e}^T + \alpha A^T. \tag{4.1}$$

The matrix P corresponds to a random graph, which models the temporal Web to predict a future Web graph by an early Web graph. This is called *Temporal Web Prediction Model.*



(a) Original Static Graph



(b) Random Graph produced

Figure 4.3: Illustration on the random graph

From the static graph in Figure 4.3(a), where node 4 and node 5 are assumed to be nodes of $DNC1$, a random graph in Figure 4.3(b) is generated by the above model ($\alpha = 1$).

Compare with the Previous Work

In [4], an absorbing model was suggested. This model can handle dangling nodes by modifying the original graph. Specifically speaking, it adds additional nodes (called *clones*), adds links from all the original nodes to their clones on the Web, and adds links from all the clones to themselves. As a result, the modified graph has no dangling node and so it is robust against dangling nodes. However, since the structure of the modified graph is different from the original Web structure, there will be a great difference between the final ranking results based on the modified graph and the ones based on the original graph.

In [49], pages whose out-degree is zero are handled by adding jump to a randomly selected page with probability 1 from every dangling node, and then by adding teleportation. More formally, the model 1 is modified as

Model 2:

$$\mathbf{x} = [(1-\alpha)E + \alpha P']\mathbf{x}, \tag{4.2}$$

where $E = \mathbf{f}\mathbf{e}^T,\ P' = A + \mathbf{f}\mathbf{v}^T,\ \mathbf{f} = \mathbf{e}/n$, and $\mathbf{v}$ denotes the $n-$dimensional column vector identifying the dangling nodes:

$$v_i = \begin{cases} 1 & \text{if } i \text{ is a dangling node,} \\ 0 & \text{otherwise.} \end{cases}$$

$\mathbf{f}$ is referred as the personalization vector, which models the behavior of users when they get bored in following the link and decide to jump randomly.

Further, Kamavar et. al. [49] speed up the *PageRank* algorithm by exploiting the block structure of the Web.

In [30], dangling pages are handled in a similar way, but achieve more computational efficiency though sacrificing some kind of accuracy. We reinterpret the model formally as follows.

Model 3:

$$\begin{aligned} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} &= \begin{pmatrix} \alpha C + (1-\alpha)/m \cdot \mathbf{1} & 1/m\mathbf{1} \\ \alpha D & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \\ &= (\alpha A + (1-\alpha)B) \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \end{aligned} \tag{4.3}$$

where $A = \begin{pmatrix} C & 1/m\mathbf{1} \\ D & 0 \end{pmatrix}, B = \begin{pmatrix} 1/m \cdot \mathbf{1} & 1/m\mathbf{1} \\ 0 & 0 \end{pmatrix}$, m is the number of nodes that have been crawled successfully, n is the number of nodes that have been found by the crawler, $C = (c_{ij})$, $D = (d_{ij})$ and if d_j is the out-degree of node j,

$$c_{ij} = \begin{cases} d_j^{-1} & \text{if there is a link node } i \text{ node } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$d_{ij} = \begin{cases} d_j^{-1} & \text{if there is a link node } i \text{ node } j, \\ 0 & \text{otherwise.} \end{cases}$$

Respectively by C and D we also denote the set of all nodes that have been crawled successfully and the set of remaining nodes.

In this model, the matrix A models the users' behavior in case of following the actual links and the unknown links from dangling nodes to visited nodes. The matrix B models the users' teleportation. Then the linear convex combination of the matrix A and the matrix B models the total behaviors of the users.

By adding a virtual node $n+1$, the Eq. (4.3) is equivalent to the following

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ z \end{pmatrix} = \begin{pmatrix} \alpha C & O & \mathbf{e}/m \\ \alpha D & O & 0 \\ (1-\alpha)\mathbf{e}^T & \mathbf{e}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ z \end{pmatrix} \tag{4.4}$$

which can be found in [30]. Exploiting this structure, the authors developed the following reduced eigen-system:

$$\begin{pmatrix} \mathbf{x} \\ z \end{pmatrix} = \begin{pmatrix} \alpha C & e/m \\ (1-\alpha)e^T + \alpha e^T D & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ z \end{pmatrix}$$

After solving the reduced eigen-system iteratively, the vector $\mathbf{y}$ can be calculated in one step:

$$\mathbf{y} = \alpha D \mathbf{x}.$$

While this form of linear equation can be calculated efficiently by exploiting the special structure of the above matrix and the computation complexity is a very important factor (if not the most important), in the case of extremely large Web, it is not as accurate as the model 2.

From Eq. (4.3), we can see what the problem is. For our convenience, we denote the matrix $\begin{pmatrix} C & 1/m\mathbf{1} \\ D & 0 \end{pmatrix}$ as $\begin{pmatrix} X & M \\ Y & N \end{pmatrix}$, the link information about

X and Y is already known because the crawler has visited all the nodes in C and therefore all the link from the nodes in C to nodes in C and D have been known by the crawler.

But the information about links from the nodes in D to nodes in C and D is unknown by the crawler because the nodes in D have not been visited yet or have not been visited successfully. Hidden in the matrix $\begin{pmatrix} C & 1/m\mathbf{1} \\ D & 0 \end{pmatrix}$, there is an assumption that the users will jump randomly and uniformly from every node in D only to nodes in C, and therefore $M = 1/m\mathbf{1}$. This assumption can be improved to be more accurate. In reality, the users may jump from nodes in D to other nodes in D, and thus the assumption that all the elements in the bottom-right part of the matrix are zero is problematic, and the assumption about the top-right part of the matrix needs to be adjusted accordingly. In our model, we assume that the users will jump randomly but not uniformly from every node in D to both nodes in C and nodes in D.

Our model is different from the absorbing model in that our model try to predict the unknown link information and therefore handle the dangling node robustly and accurately while the absorbing model is new paradigm for ranking. So the ranking values derived from these two models are not comparable.

Our model is different from model 2 in that we get the information about the unknown part of the matrix by prediction while the model 2 assumes the uniform distribution on the unknown part of the matrix. The authors in [49] also suggest re-defining the vector $\mathbf{f}$ as a non-uniform distribution; however, they only consider the vector $\mathbf{f}$ as the personalization factor, which is a subjective factor, and from which the *PageRank* vector can be biased toward certain kinds of pages.

Our model is different from model 3 in two folds:

1. The users will not jump uniformly from every node in D to other nodes.
2. The users will jump from every node in D not only to nodes in C but

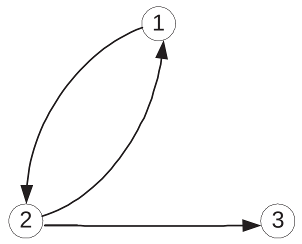


Figure 4.4: A case in which considering dangling node will have significant effect on the ranks of non-dangling nodes

also to nodes in D.

The authors in [30] further discuss the "link rot" problem and suggest new methods of ranking motivated by the hierarchical structure of the Web. Although we can combine our model with the technique used in solving the "link rot" problem, we do not include it herein.

Other related work is the Page Popularity Evolution Model in [21, 22], in which the popularity of a page evolves with the time; it is also a kind of prediction which looks outside the Web structure, while our model looks inside the Web structure at the current time.

A Simple Example

We consider a case in which dangling nodes are so significant that including them in the overall ranking may not only change the rank value of non-dangling nodes but also change the order of the non-dangling nodes.

In the example of Figure 4.2.3, there are three pages, with one of them being a dangling node with a link from page 2. If we compute *PageRank* by the model 2, and let $\alpha = 0.85$, the matrix in the model 2 is

$$\begin{pmatrix} 0.05 & 0.475 & 1/3 \\ 0.9 & 0.05 & 1/3 \\ 0.05 & 0.475 & 1/3 \end{pmatrix}$$

By power iteration, the *RageRank* scores are $(x_1, x_2, x_3) = (0.3032, 0.3936, 0.3032)$. So in model 2, rank for node 2 is much higher than that for node 1. If we simply remove the dangling node 3, then by Eq. (2.3), the *PageRank* scores for nodes 1 and 2 are $(x_1, x_2) = (0.5, 0.5)$, in which the rank for node 1 is same as that for node 2. From this example, we can see that whether we handle dangling nodes will not only change the rank value of the non-dangling nodes but also their order.

4.2.4 Random Graph Ranking

We need to extend the standard ranking technique to random graphs in order to handle the random graph outputs produced by the predictive strategy in three situations: (1) temporal links, (2) spatial links and (3) weighted links.

PageRank on a Random Graph

We extend the *PageRank* from the setting of a static graph to the setting of a random graph. Similar to *PageRank*, the page rank vector x on a random graph can be defined recursively in terms of random graphs:

$$x_i = \sum_j q_{ij} x_j,$$

where $q_{ij} = p_{ji} / \sum_k p_{jk}$. Or in matrix form, $\mathbf{x} = Q\mathbf{x}$, where $Q = (q_{ij})$. In a static graph, if there is a link from v_j to v_i, then the probability of a random surfer will follow the link is $1/d_j$, where d_j is the out-degree of v_j. In a random graph, since the sum $\sum_k p_{jk}$ is the expected out-degree of v_j and the link from v_j to v_i exists with a probability of p_{ji}, the expected probability of a random surfer will follow the link (v_j, v_i) is $p_{ji} / \sum_k p_{jk}$. Consequently, the above equation is established.

Common Neighbor on a Random Graph

We extend the *Common Neighbor* approach from the setting of a static graph to the setting of a random graph.

First, the random neighbor set $RI(v_i)$ of v_i is defined as

$$RI(v_i) = \{(v_k, p_{ki})|v_k \in V\},$$

where p_{ki} is the probability of v_k as a neighbor of v_i. In the setting of a random graph, each node v_k is linked to node v_i with a probability p_{ki}, so v_k is the neighbor of v_i with a probability p_{ki}. This extends the definition of the set of neighbors of node v_i in the setting of a static graph.

Second, the set of the common random neighbors of v_i and v_j is defined as

$$RI(v_i) \cap RI(v_i) = \{(v_k, p_{ki}p_{kj})|v_k \in V\}.$$

The sets of random neighbors of v_i and v_j are $RI(v_i)$ and $RI(v_j)$ respectively. If v_k is the neighbor of v_i with a probability p_{ki}, and v_k is the neighbor of v_j with a probability p_{kj}, then we can say v_k is the common neighbor of v_i and v_j with a probability $p_{ki}p_{kj}$ since the random edges are drawn independently. This extends the meaning of the common neighbor.

Third, the expected number of nodes in $RI(v_i) \cap RI(v_i)$ is considered as the similarity measure $s(i, j)$, and is defined as

$$s(i, j) = \sum_k p_{ki}p_{kj}.$$

This extends the definition of the number of common neighbors of v_i and v_j in the setting of a static graph.

Jaccard's Coefficient on a Random Graph

The *Jaccard's Coefficient* in the setting of a random graph is defined as

$$\begin{aligned} s(i, j) &= |RI(v_i) \cap RI(v_j)|/|RI(v_i) \cup RI(v_j)| \\ &= \textstyle\sum_k p_{ki}p_{kj}/\sum_k(p_{ki} + p_{kj} - p_{ki}p_{kj}), \end{aligned}$$

where $RI(v_i) \cup RI(v_j) = \{(v_k, p_{ki}+p_{kj}-p_{ki}p_{kj})|v_k \in V\}$. The expected number elements in $RI(v_i) \cup RI(v_j)$ is equal to $\sum_k(p_{ki} + p_{kj} - p_{ki}p_{kj})$. Since v_k is not the neighbor of v_i with a probability $1-p_{ki}$, and is not the neighbor of v_j with a probability $1-p_{kj}$, we assume that v_k is not the neighbor of either v_i or v_j with a probability $(1-p_{kj})(1-p_{kj})$, and we have that v_k is the neighbor of either v_i or v_j with a probability $1-(1-p_{kj})(1-p_{kj}) = p_{ki}+p_{kj}-p_{ki}p_{kj}$. Therefore, $RI(v_i) \cup RI(v_j) = \{(v_k, p_{ki}+p_{kj}-p_{ki}p_{kj}) | v_k \in V\}$. The expected number of elements is thus equal to $\sum_k (p_{ki}+p_{kj}-p_{ki}p_{kj})$.

SimRank on a Random Graph

In the setting of a random graph, the *SimRank* score $s(i,j)$ of two pages v_i and v_j can be naturally redefined as the fixed point of the following recursive definition,

$$s(i,j) = \begin{cases} 1, & i=j, \\ \frac{C}{|RI(v_i)||RI(v_j)|}\sum_{u,v} p_{ui}p_{vj}s(u,v), & otherwise, \end{cases}$$

for some constant decay factor $C \in (0,1)$, where $|RI(v_i)| = \sum_k p_{ki}$, $|RI(v_j)| = \sum_k p_{kj}$.

Note that one can easily conclude that when the random graph becomes a static graph, the algorithms described in the above subsections degrade into the original algorithms. This means ranking algorithms on a random graph generalize the original ones.

Heat Diffusion Model on a Random Directed Graph

Recall that, in Section 3.2, we have established the heat diffusion model on a random directed graph (V,E,P) as follows.

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

where $\gamma = \alpha t$

$$H_{ij} = \begin{cases} -\sum_{k:(k,i)\in E} p_{ki}, & j=i, \\ p_{ji}, & (j,i)\in E, \\ 0, & \text{otherwise.} \end{cases}$$

We consider defining $\mathbf{f}(1) = e^{\gamma H}\mathbf{f}(0)$ as a ranking result, and in such case $\gamma = \alpha$. In order to fit various applications, H needs to be specified according to the circumstance.

For example, on an undirected graph, if each edge is equally trusted, then

$$\mathbf{f}(1) = e^{\gamma H}\mathbf{f}(0), H_{ij} = \begin{cases} -d_j, & j = i, \\ 1, & (v_j, v_i) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

But on the Web, the links between Web pages are directed. On one Web page a, when the pagemaker creates a link (a, b) to another page b, he actually forces the energy flow, for example, users' 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, but not from b to a. Moreover, we assume that a Web surfer will jump to the next page uniformly with a probability $1/d_j$, where d_j is the out-degree of node j. With extra energy, the diffusion does not follow the Fourier law, i.e., even the temperatures on the two ends of an edge are equal, there is still heat flow on that edge. Based on such considerations, we modify the heat diffusion model on an undirected graph as follows.

Suppose, at time t, each node v_i receives $RH = RH(i, j, t, \Delta t)$ amount of heat from v_j during a period of Δt. We have three assumptions: (1) RH should be proportional to the time period Δt; (2) RH should be proportional to the heat at node v_j; and (3) RH 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 neighbors that point to it.

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 heat at node v_i. (3) Each node has the same ability of diffusing heat. This fits the intuition that a Web surfer has only 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 make 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 nodes 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 $f_i(t+\Delta t) - f_i(t) = -\gamma f_i(t)\Delta t + \sum_{j:(v_j,v_i)\in E} \gamma/d_j f_j(t)\Delta t$. Similarly, we obtain

$$\mathbf{f}(1) = e^{\gamma H}\mathbf{f}(0), H_{ij} = \begin{cases} -1, & j = i, \\ 1/d_j, & (v_j, v_i) \in E, \\ 0, & \text{otherwise}. \end{cases} \tag{4.6}$$

For real world applications, we have to consider random edges. This can be seen in two viewpoints. The first one is that in Eq. (2.4), the Web graph is actually modeled as a random graph, where there is an edge from node v_i to node v_j with a probability of $(1-\alpha)g_j$ (see the item $(1-\alpha)\mathbf{g}\mathbf{1}^T$), and that the Web graph is predicted by another random graph [97]. The second one is that the Web structure is in essence a random graph if we consider the content similarity between two pages, though this is not done currently. For these reasons, the model would become more flexible if we extend it to random graphs.

Since the probability of the link (v_j, v_i) is p_{ji}, the expected out-degree of node j is $RD^+(v_j) = \sum_k p_{jk}$, and a Web surfer will jump to the next page with a probability $p_{ji}/RD^+(v_j)$. Therefore, we have

$$\mathbf{f}(1) = e^{\gamma R}\mathbf{f}(0), R_{ij} = \begin{cases} -1, & j = i; \\ p_{ji}/RD^+(v_j), & j \neq i. \end{cases} \tag{4.7}$$

When the graph is large, a direct computation of $e^{\gamma R}$ is time-consuming, and we adopt its discrete approximation:

$$\mathbf{f}(1) = (I + \frac{\gamma}{N}R)^N \mathbf{f}(0). \tag{4.8}$$

The matrix $(I + \frac{\gamma}{N}R)^N$ in Eq. (4.8) and matrix $e^{\gamma R}$ in Eq. (4.7) are called *Discrete Diffusion Kernel* and *Continuous Diffusion Kernel*, respectively. Based on the Heat Diffusion Models and their solutions, *DiffusionRank* can be established on undirected graphs, directed graphs, and random graphs. In the next section, we mainly focus on *DiffusionRank* in the random graph setting.

4.3 DiffusionRank

For a random graph, the matrix $(I + \frac{\gamma}{N}R)^N$ or $e^{\gamma R}$ can measure the similarity relationship between nodes. Let $f_i(0)= 1, f_j(0) = 0$ if $j \neq i$, then the vector $\mathbf{f}(0)$ represents the unit heat at node v_i while all other nodes have zero heat. For such $\mathbf{f}(0)$ in a random graph, we can find the heat distribution at time 1 by using Eq. (4.7) or Eq. (4.8). The heat distribution is exactly the $i-$th row of the matrix of $(I + \frac{\gamma}{N}R)^N$ or $e^{\gamma R}$. So the i-th row j-th column element h_{ij} in the matrix $(I + \gamma \Delta t R)^N$ or $e^{\gamma R}$ means the amount of heat that v_i can receive from v_j between time 0 and 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 $(I + \frac{\gamma}{N}H)^N$ or $e^{\gamma H}$ 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; and 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 the width of the pipes.

Algorithm 1 DiffusionRank Function

Input: The transition matrix A; the inverse transition matrix U; the decay factor α_I for the inverse *PageRank*; the decay factor α_B for *PageRank*; the number of iterations M_I for the inverse *PageRank*; the number of trusted pages L; the thermal conductivity coefficient γ.
Output: *DiffusionRank* score vector $\mathbf{h}$.

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

4.3.1 Algorithm

For the ranking task, we adopt the heat kernel on a random graph. Formally the *DiffusionRank* is described in Algorithm 1, in which 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 0 otherwise. This trusted-page selection procedure by inverse *PageRank* is completely borrowed from *TrustRank* [38] except for a fixed 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, which in turn point to many other pages and so on. In the algorithm, the underlying random graph is set as $P = \alpha_B \cdot A + (1-\alpha_B) \cdot \frac{1}{n} \cdot \mathbf{1}_{n\times n}$, which is induced by the Web graph. As a result, $R = -I + P$.

In fact, the more general setting for *DiffusionRank* is $P = \alpha_B \cdot A + (1-\alpha_B) \cdot \frac{1}{n} \cdot \mathbf{g} \cdot \mathbf{1}^T$. By such a setting, *DiffusionRank* is a generalization of *TrustRank* when γ tends to infinity and when g is set in the same way as that in *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, and 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 cannot find it without any priori knowledge.

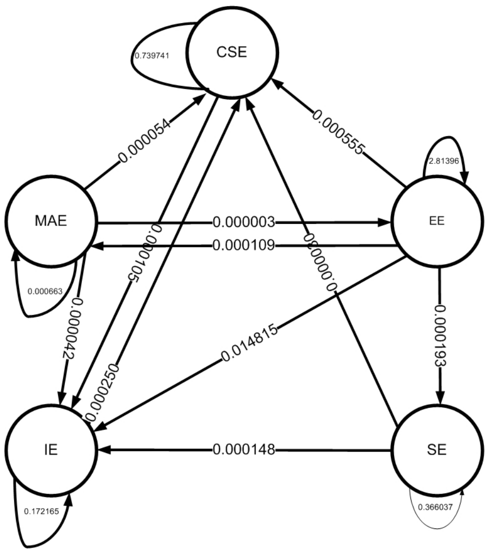
Remark. In a social network interpretation, *DiffusionRank* first recognizes a group of trusted people, who may not be highly ranked, but who know many other people. The initially trusted people are endowed with the power to decide who can be further trusted, but cannot decide the final voting results, and so they are not dictators.

4.3.2 Advantages

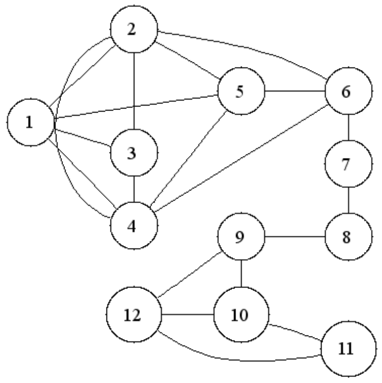
Next we show the four advantages for *DiffusionRank*.

Two closed forms

First, the solutions to *DiffusionRank* have two forms, both of which are closed forms. 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 will be shown in the proof of theorem in the next section.



(a) Group to Group Relations



(b) An undirected graph

Figure 4.5: Two graphs

Group-group relations

Second, it can be naturally employed to detect the group-group relation. For example, let $G2$ and $G1$ denote two groups, containing pages $(j_1, j_2, \ldots, j_s)$ and $(i_1, i_2, \ldots, i_t)$, respectively. Then $\sum_{u,v} h_{i_u, j_v}$ is the total amount of heat that $G1$ receives from $G2$, where h_{i_u, j_v} is the i_u-th row j_v-th column element of the heat kernel. More specifically, we need to first 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. Next we employ Eq. (4.7) to calculate

$\mathbf{f}(1) = (f_1(1), f_2(1), \ldots, f_n(1))^T$, and finally we sum those $f_j(1)$ where $j \in \{i_1, i_2, \ldots, i_t\}$. Fig. 4.5 (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, and the numbers in Fig. 4.5 (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 engage other ranking techniques to such an application because they lack similar 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 Fig. 4.5 (b) is an undirected graph, and so we employ Eq. (4.5). If we know that node 1 belongs to one community and that node 12 belongs to another, 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 the corresponding heat solution.

Anti-manipulation

Fourth, it can be used to combat manipulation. Let $G2$ contain 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 $G2$, and can be computed by the discrete approximation of Eq. (4.6) in the case of a static graph or Eq. (4.8) 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 one unit heat while all the others have zero heat. In doing so, a 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 4.1. Based on the fact that the connection from a trusted page to a "bad" page should be weak, i.e., less uncross paths, longer distance and narrower pipe, etc., we can say *DiffusionRank* can resist Web spam if we can select trusted pages. It is fortunate that the trusted pages selection method in [38], which is the first part of *TrustRank*, can help us to fulfill this task. 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. (4.8), by which we need N iterations and for each iteration we need a multiplication operation between a matrix and a vector, while in Eq. (2.4) we also need a multiplication operation between a matrix and a vector for each iteration.

4.3.3 The Physical Meaning of γ

γ plays an important role in the anti-manipulation effect of *DiffusionRank*. γ is the thermal conductivity, i.e., the heat diffusion coefficient. If it has a high value, heat will diffuse very quickly. Otherwise, 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 7 When γ tends to infinity and $\mathbf{f}(0)$ is not the zero vector, $e^{\gamma R}\mathbf{f}(0)$ is proportional to the stable distribution produced by *PageRank*.

Let $\mathbf{g} = \frac{1}{n}\mathbf{1}$. By the Perron Theorem [67], we have shown that 1 is the largest eigenvalue of $P = [(1-\alpha)\mathbf{g}\mathbf{1}^T + \alpha A]$, and that there is no other eigenvalue whose absolute value is equal to 1. Let $\mathbf{x}$ be a stable distribution, and so $P\mathbf{x} = \mathbf{x}$. $\mathbf{x}$ is the eigenvector corresponding to the eigenvalue 1. Assuming the other $n-1$ eigenvalues of P are $|\lambda_2| < 1, \ldots, |\lambda_n| < 1$, we can find an invertible matrix $S = (\ \mathbf{x} \quad S_1\ )$ such that

$$S^{-1}PS = \begin{pmatrix} 1 & * & * & * \\ 0 & \lambda_2 & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & \lambda_n \end{pmatrix}. \tag{4.9}$$

Since $e^{\gamma R} = e^{\gamma(-I+P)} =$

$$S^{-1}\begin{pmatrix} 1 & * & * & * \\ 0 & e^{\gamma(\lambda_2-1)} & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & e^{\gamma(\lambda_n-1)} \end{pmatrix}S, \tag{4.10}$$

all eigenvalues of the matrix $e^{\gamma R}$ are $1, e^{\gamma(\lambda_2-1)}, \ldots, e^{\gamma(\lambda_n-1)}$. When $\gamma \to \infty$, they become $1, 0, \ldots, 0$, respectively, which means that 1 is the only nonzero eigenvalue of $e^{\gamma R}$ when $\gamma \to \infty$. We can see that when $\gamma \to \infty$, $e^{\gamma R}e^{\gamma R}\mathbf{f}(0) = e^{\gamma R}\mathbf{f}(0)$, and so $e^{\gamma R}\mathbf{f}(0)$ is an eigenvector of $e^{\gamma R}$ when $\gamma \to \infty$. On the other hand, $e^{\gamma R}x = (I + \gamma R + \frac{\gamma^2}{2!}R^2 + \frac{\gamma^3}{3!}R^3 + \ldots)\mathbf{x} = I\mathbf{x} + \gamma R\mathbf{x} + \frac{\gamma^2}{2!}R^2\mathbf{x} + \frac{\gamma^3}{3!}R^3\mathbf{x} + \ldots = \mathbf{x}$ since $R\mathbf{x} = (-I+P)\mathbf{x} = -\mathbf{x} + \mathbf{x} = \mathbf{0}$, and hence $\mathbf{x}$ is the eigenvector of $e^{\gamma R}$ for any γ. Therefore both $\mathbf{x}$ and $e^{\gamma R}\mathbf{f}(0)$ are the eigenvectors corresponding to the unique eigenvalue 1 of $e^{\gamma R}$ when $\gamma \to \infty$, and consequently $\mathbf{x} = ce^{\gamma R}\mathbf{f}(0)$.

By this theorem, we see that *DiffusionRank* is a generalization of *PageRank*. When $\gamma = 0$, the ranking value is the most robust against manipulation since no heat is diffused and the system is unchangeable, but the Web structure is completely ignored since $e^{\gamma R}\mathbf{f}(0) = e^{0R}\mathbf{f}(0) = I\mathbf{f}(0) = \mathbf{f}(0)$; when $\gamma = \infty$, *DiffusionRank* becomes *PageRank*, which can be manipulated easily. We expect an appropriate setting of γ that can reach a balance. There is no theoretical result in achieving an optimal point, but in practice we find that $\gamma = 1$ works well in Section 4.4. Next we discuss how to determine the number of iterations if we employ the discrete heat kernel.

4.3.4 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. (4.8) in an iterative way. Then the problem about determining N, the number of iterations, arises:

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

Since it is difficult to solve this problem, we propose a heuristic motivated by the following observations. When $R = -I + P$, by Eq. (4.9), we have $(I + \frac{\gamma}{N}R)^N = (I + \frac{\gamma}{N}(-I + P))^N =$

$$\mathbf{S}^{-1}\begin{pmatrix} 1 & * & * & * \\ 0 & (1+\frac{\gamma(\lambda_2-1)}{N})^N & * & * \\ 0 & 0 & \ddots & * \\ 0 & 0 & 0 & (1+\frac{\gamma(\lambda_n-1)}{N})^N \end{pmatrix}\mathbf{S}. \tag{4.11}$$

Comparing Eq. (4.10) and Eq. (4.11), we observe that the eigenvalues of $(I + \frac{\gamma}{N}R)^N - e^{\gamma R}$ 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 only for positive λ's.

We also observe that if $\gamma = 1$ and $\lambda < 1$, 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. Currently we use the relatively accurate setting $N = 100$ to make the real eigenvalues in $(I + \frac{\gamma}{N}R)^N - e^{\gamma R}$ less than 0.005.

4.4 Experiments for *PRGR* framework

The temporal dimension of the *PRGR* framework can actually be designed to be tested in experiments, although it is difficult to measure whether a link analysis algorithm is better than another because of the different intuitions for different ranking algorithms. For this, we design a comparison method by calculating the ranking difference and order difference between the early results (less accurate) and the final results (relatively accurate, and considered as a ground truth). For more details, see Section 4.5.2.

4.4.1 Data Description

Our input data consist of a synthetic data set and a real-world data set. A detailed description follows.

Synthetic Web Graph

The degree sequences of the World Wide Web are shown to be well approximated by a power law distribution [50, 54, 55]. That is, the probability that a Web page has k outgoing or incoming links follows a power law over many orders of magnitude, that is, $P_{out}(k) \sim k^{-\gamma_{out}}$ and $P_{in}(k) \sim k^{-\gamma_{in}}$.

The power law distribution of the degree sequence appears to be a very robust property of the Web despite its dynamic nature; therefore, we can generate synthetic Web-like random graphs to test the performance of our

t	1	2	3	4	5	6
V[t]	1000	1100	1200	1300	1400	1500
T[t]	1764	1778	1837	1920	1927	1936
t	7	8	9	10	11	
V[t]	1600	1700	1800	1900	2000	
T[t]	1952	1954	1964	1994	2000	

Table 4.1: Description of the synthetic graph series

t	1	2	3	4	5	6
V[t]	7712	78662	109383	160019	252522	301707
T[t]	18542	120970	157196	234701	355720	404728
t	7	8	9	10	11	
V[t]	373579	411724	444974	471684	502610	
T[t]	476961	515534	549162	576139	607170	

Table 4.2: Description of real data sets within domain cuhk.edu.hk

algorithms.

Several approaches to modeling power law graphs [50, 54] have been proposed. In our numerical experiment, we use the (α, β) model [54] to generate random graphs. By setting $\alpha = 0.52$ and $\beta = 0.58$, the model generates a random power law graph with $\gamma_{out} = 2.1$ and $\gamma_{in} = 2.38$, both of whose values match the Web.

By simulating the procedure of crawling, we can obtain a series of growing incomplete graphs containing pages of $DNC1$. The number $V[t]$ of pages visited and the total number $T[t]$ of pages found at time t are shown in Table 4.1.

Real Web Graph

The data of a real Web graph were obtained from the domain *cuhk.edu.hk*. The graph series are snapshots during the process of crawling pages restricted within this domain. The number $V[t]$ of pages visited and the total number $T[t]$ of pages found at time t are shown in Table 4.2.

4.4.2 Methodology

The algorithms we run include *PageRank* and *DiffusionRank*. For each algorithm A, we have two versions, denoted by A and $PreA$. A is the original version without using the *Temporal Web Prediction Model*, and $PreA$ is the version using the *Temporal Web Prediction Model*. Both $PreA$ and A are run on two data series, including the synthetic data series and the real data series. Each data series contains 11 data sets which are obtained by taking snapshots during the process of a crawler or a simulated crawler. Finally, for each data series and for each algorithm A, we obtained 22 ranking results, namely,

$$\begin{array}{llll} A_1, & A_2, & \ldots, & A_{11}, \\ PreA_1, & PreA_2, & \ldots, & PreA_{11}. \end{array}$$

The results on the first 10 data sets are not accurate because these data are incomplete, and the Web is dynamically changing. The result A_{11} on the synthetic data should be the same as $PreA_{11}$ because *Temporal Web Prediction Model* will not have effect on complete information, but the result A_{11} on the real data is not the same as $PreA_{11}$ because of the existence of dangling nodes of $DNC1$ in time 11.

If the difference between the results on time t and the results on time 11 is smaller, we think it is more accurate. We calculate the value difference and order difference described below.

Value Difference. The value difference between A_t ($PreA_t$) and A_{11} is measured as

$$||A_t/Max_t - Cut(t, A_{11})/CutMax_t||_2(||PreA_t/Max_t - Cut(t, A_{11})/CutMax_t||_2),$$

where $cut(t, A_{11})$ is the results cut from A_{11} such that it has the same dimension as A_t, and $CutMax_t$ (Max_t) means the maximal value among results in $cut(t, A_{11})$ (A_t).

Order Difference. The order difference between A_t ($PreA_t$) and A_{11} is measured as the significant order difference between A_t and $Cut(t, A_{11})$ ($PreA_t$

and $Cut(t, A_{11})$). The significant order difference between two similarity matrices M and N is calculated by the sum of the significant order difference for each row of M and N. For each row $M(i), N(i)$ of M and N, the pair $(M(i,j), M(i,k))$ and $(N(i,j), N(i,k))$ is considered as a significant order difference if it satisfies the following criteria:

Both $M(i,j) > M(i,k) + 0.005Max_M$ and $N(i,k) > N(i,j) + 0.005Max_N$, or both $M(i,k) > M(i,j) + 0.005Max_M$ and $N(i,j) > N(i,k) + 0.005Max_N$, where Max_M (Max_N) is the maximum value of M (N).

4.4.3 Experimental Set-up

We conduct experiments on the workstation whose hardware model is Nix Dual Intel Xeon 2.2GHz with 1GB RAM and a Linux Kernel 2.4.18-27smp (RedHat7.3). We set $\alpha = 0.85$ and set g to be the uniform distribution in both *PageRank* and *PrePageRank*. Note that we use the modified *PageRank* algorithm [49], in which dangling nodes of $DNC1$ are considered to have random links uniformly distributed to each node. For *DiffusionRank* and *PreDiffusionRank*, we use the *Discrete Diffuse Kernel* for computing, and we set $\sigma = 1$, $N = 20$ and β to be the inverse of the maximal out-degree in both. Note that *DiffusionRank* uses the *Discrete Diffuse Kernel* on a static graph and *PreDiffusionRank* uses *Discrete Diffuse Kernel* on a random graph.

4.4.4 Experimental Results

Figure 4.6 demonstrates the *PageRank* results on the synthetic data and the real data. On the synthetic data, in 60% early stages, *PreRageRank* is closer to the final result in value difference; while in 100% early stages, *PrePageRank* is closer to the final result in significant order difference. Since the graph in time 11 is complete, there is no difference between the *PrePageRank* and *PageRank*, and the curves meet at time 11. On the real data, since the data set at time 11 contains unvisited pages, *PrePageRank* and *PageRank* are different on this data set, and the employment of *PageRank* results on this data set as a reference will cause a bias against *PrePageRank*. Even so, in 60% early stages, *PreRageRank* is closer to the final *PageRank* result in value difference; while in 70% early stages, *PrePageRank* is closer to the final *PageRank* result in significant order difference.

Figure 4.7 demonstrates the *DiffusionRank* results. On the synthetic data, in 100% early stages, *PreDiffusionRank* is closer to the final result in value difference, and in 100% early stages, *PreDiffusionRank* is closer to the final result in significant order difference. On the real data, since the data at time 11 contains unvisited pages, *PreDiffusionRank* and *DiffusionRank* are different on this data set, and the employment of *DiffusionRank* results on this data set as a reference will cause a bias against *PreDiffusionRank*. Even so, in 70% early stages, *PreDiffusionRank* is closer to the final *PageDiffusionRank* result in value difference; while in 70% early stages, *PreDiffusionRank* is closer to the final *DiffusionRank* result in significant order difference.

Figure 4.8 demonstrates the *Jaccard's Coefficient* results. On the synthetic data, in 100% early stages, *PreJaccard's Coefficient* is closer to the final result both in value difference and in significant order difference. On the real data, since the data at time 11 contains unvisited pages, *PreJaccard's Coefficient* and *Jaccard's Coefficient* are different on this data set, and the employment of *Jaccard's Coefficient* results on this data set as a reference will cause a bias against *PreJaccard's Coefficient*. Even so, in 70% early stages, *PreJaccard's Coefficient* is closer to the final *Jaccard's Coefficient* result in value difference; while in 70% early stages, *PreJaccard's Coefficient* is closer to the final *accard's Coefficient* result in significant order difference. For *Common Neighbor*, similar results are obtained.

The improvement of prediction on *Common Neighbor* is relatively small, and in Figure 4.9 we only show the differences of results of Common Neighbor and those of PreCommon Neighbor.

4.4.5 Discussion

For *SimRank*, similar experiments are conducted on small datasets. On *Common Neighbor*, slight improvement is achieved, but on *SimRank* we do not obtain expected results for the *Temporal Web Prediction Model*. These abnormal results may be caused by the ignorance of the power law distribution. Before *Temporal Web Prediction Model*, the data set satisfies the power law distribution, but after the model, the in-degrees of all nodes are increased with the same proportions. This in fact breaks the power law distribution, for example, nodes whose in-degree is 1 do not exist after prediction while nodes whose in-degree is 1 should have the highest density according to the power law distribution. *SimRank* seems to be sensitive to the distribution of in-degrees and out-degrees. It is interesting and challenging to preserve the power law distribution in the *Temporal Web Prediction Model* so that better accuracy can be achieved on all these algorithms.

4.5 Experiments for *DiffusionRank*

4.5.1 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 Fig. 4.10 (a). 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 graphs were obtained from the domain in our institute in October, 2004. The total number of pages found is 18,542 in the middle-size graph, and 607,170 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.

4.5.2 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 listed as follows.

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

Pairwise Order Difference. The order difference between A and B is measured by the number of significant order differences between A and 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 happens: (1) both $A[i] > [<]A[j] + 0.1$ and $B[i] \leq [\geq]A[j]$, and (2) both $A[i] \leq [\geq]A[j]$ and $B[i] > [<]A[j] + 0.1$.

4.5.3 Experimental Set-up

The experiments on the middle-size graph and the large-size graph are conducted on the workstation, whose hardware model is Nix Dual Intel Xeon 2.2GHz with 1GB RAM and a Linux Kernel 2.4.18-27smp (RedHat7.3). In calculating *DiffusionRank*, we employ Eq. (4.8) and the discrete approximation of Eq. (4.6) for such graphs. The related tasks are implemented using C language. While in the toy graph, we employ the continuous diffusion kernel in Eq. (4.6) and Eq. (4.7), and implement related tasks using Matlab.

For nodes that have zero out-degree (dangling nodes), we employ the method in the modified *PageRank* algorithm [49], in which dangling nodes of are considered to have random links uniformly distributed to each node. We set $\alpha = \alpha_I = \alpha_B = 0.85$ in 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 4.3.3 and Section 4.3.4, we set the iteration number to be $M_B = 100$ in *DiffusionRank*, and for accuracy consideration, the iteration number in all the algorithms is set to be 100.

4.5.4 Approximation of *PageRank*

We show that when γ tends to infinity, the value differences between *DiffusionRank* and *PageRank* tend to zero. Fig. 4.10 (b) shows the approximation property of *DiffusionRank*, as proved in Theorem 7, on the toy graph. The horizontal axis of Fig. 4.10 (b) marks the γ value, and the vertical axis corresponds to the value difference between *DiffusionRank* and *PageRank*. All the possible trusted sets with $L = 1$ are considered. 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.

4.5.5 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 newly added nodes that point to the manipulated node. The horizontal axes of Fig. 4.11 stand for the numbers of newly added nodes, and the vertical axes show the corresponding rank values of the manipulated nodes. To be clear, we consider all six situations. Every node in Fig. 4.10 (a) is manipulated respectively, and its corresponding values for *PageRank*, *TrustRank* (TR), *DiffusionRank* (DR) are shown in the one of the six sub-figures in Fig. 4.11. 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 nodes for *DiffusionRank* grow slowest as the number of the newly added nodes increases. On the middle-size graph and the large-size graph, this conclusion is also true, as seen in Fig. 4.12. Note that, in Fig. 4.12 (a), we choose four trusted sets ($L = 1$), on which we test *DiffusionRank* and *TrustRank*, whose results are denoted by DiffusionRank$i$ and TrustRank$i$ ($i = 0, 1, 2, 3$ denotes the four trusted set). In Fig. 4.12 (b), we choose one trusted set ($L = 1$). Moreover, in both Fig. 4.12 (a) and Fig. 4.12 (b), 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, so we only compare the common part of A and PA. This experiment is used to test the stability of all the algorithms. The less the order difference, the more stable the algorithm, in the sense that only a smaller part of the order relations is affected by the manipulation. Figure 4.13 (a) shows that the order difference values change when we add new nodes that point to the manipulated node. We give several γ settings. We find that when $\gamma = 1$, the least order difference is achieved by *DiffusionRank*. It is interesting to note that as γ increases, the order difference will increase first; however after reaching a maximum value, it will decrease, and finally it tends to the *PageRank* results. We show this tendency in Fig. 4.13 (b), in which we choose three different settings, where the number of manipulated nodes are 2,000, 5,000, and 10,000 respectively. From these figures, we can see that when $\gamma < 2$, the values are less than those for *PageRank*, and that when $\gamma > 20$, the difference between *PageRank* and *DiffusionRank* is very small. After these investigations, we find that in all the graphs we tested, *DiffusionRank* (when $\gamma = 1$) is the most robust against manipulation both in value difference and in order difference. The trust set selection algorithm proposed in [38] is effective for both *TrustRank* and *DiffusionRank*.

4.5.6 Manipulation Detection

The difference between the *PageRank* value and the *DiffusionRank* (*TrustRank*) value can help us detect pages being manipulated. The larger the difference is, the more probably the page is manipulated. The next subsection is devoted to this finding, by which 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 a task in the same way.

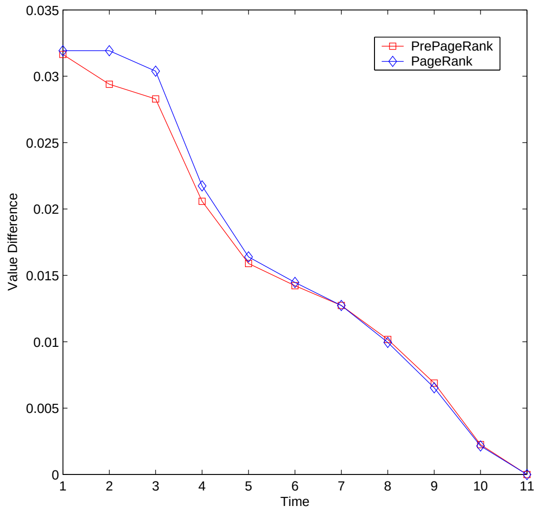
To test this idea, we randomly choose 100 pages, and manipulate them all in different extents. Then we draw the Recall-Precision curves as follows: For a given recall rate ν, find the threshold τ such that the recall rate is exactly ν, and then calculate the ratio of the number of nodes being manipulated and the number of nodes whose difference between *PageRank* and *DiffusionRank* (or between *PageRank* and *TrustRank*) is greater than τ. The higher precision rate under the same recall rate means that the less "good" nodes are mixed with "bad" nodes. The upper panel in Fig. 4.14 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 has 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 excellently 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 the size of the areas below the curves, we find *DiffusionRank* performs slightly better than *TrustRank*.

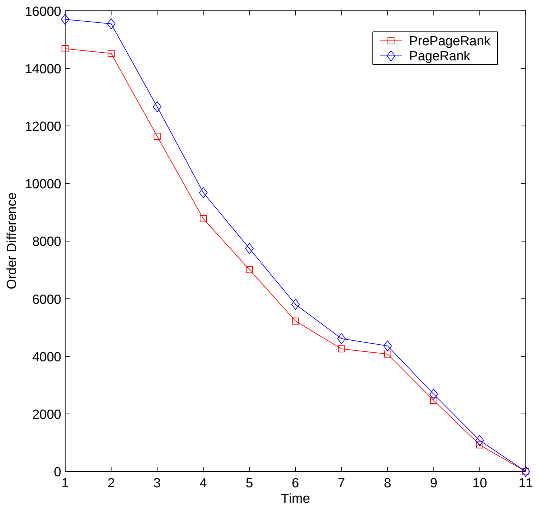
4.6 Summary

We have shown that the *Temporal Web Prediction Model* is effective in *PageRank* and *DiffusionRank*. Because our model mines more information about the Web structure, the results of predictive strategy on these two algorithms are more accurate than those without it, even our model breaks the power law distribution. We conclude that the random graph input indeed extends the scope of some original ranking techniques, and significantly improve some of them.

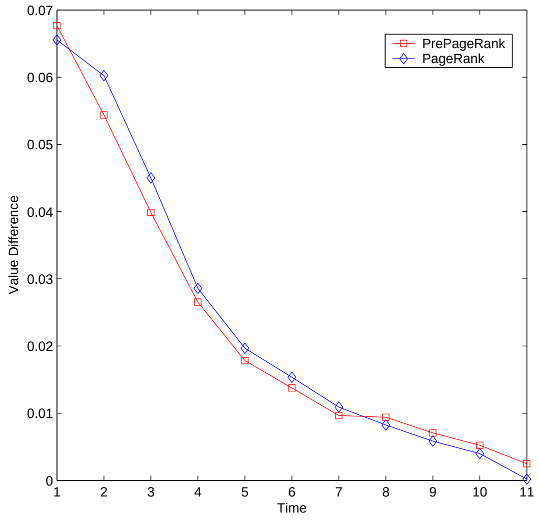
DiffusionRank is a generalization of *PageRank*, which is interesting in that the heat diffusion coefficient γ can balance the extent that we want to model the original Web graph and the extent that we want to reduce the effect of link manipulations. The experimental results show that we can actually achieve such a balance by empirically setting $\gamma = 1$, although the best setting including varying γ_i 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 the best both in reduction effects and in its stability among all the three algorithms.



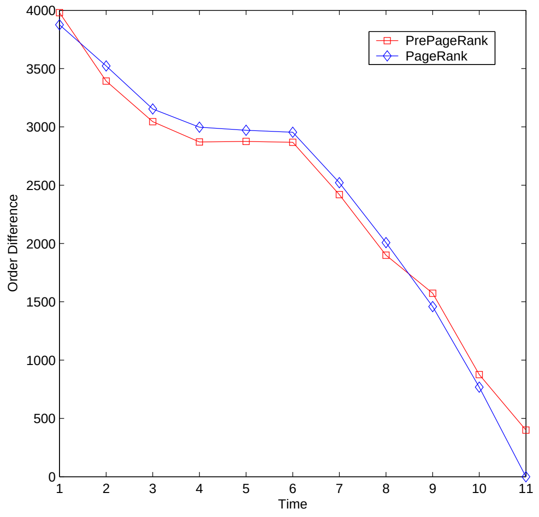
(a)-VD in synthetic data



(b)-OD in synthetic data

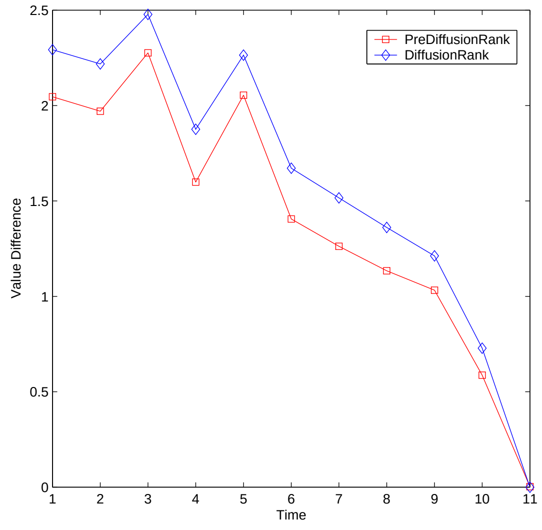


(c)-VD in real data

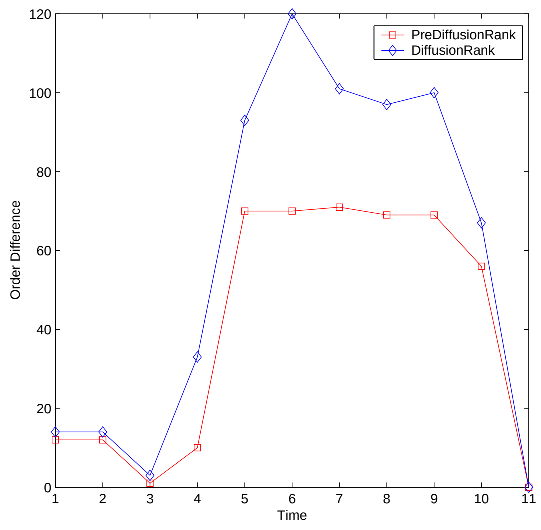


(b)-OD in real data

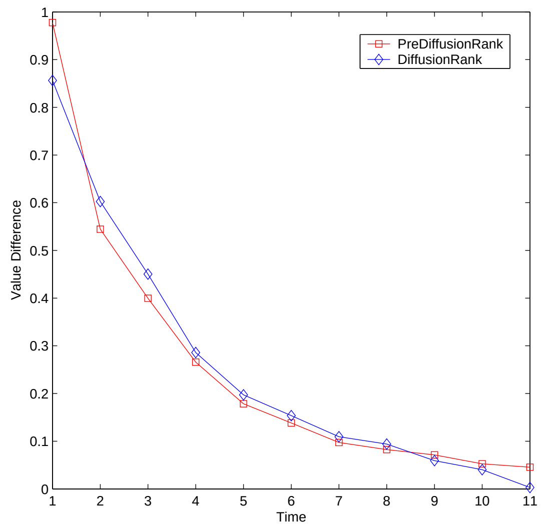
Figure 4.6: PageRank comparison results



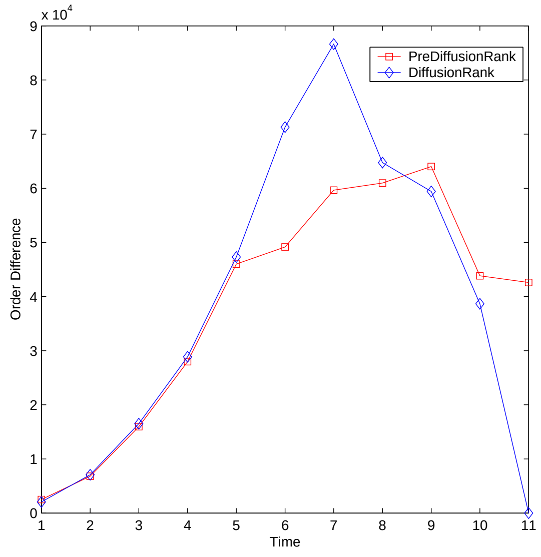
(a)-VD in synthetic data



(b)-OD in synthetic data

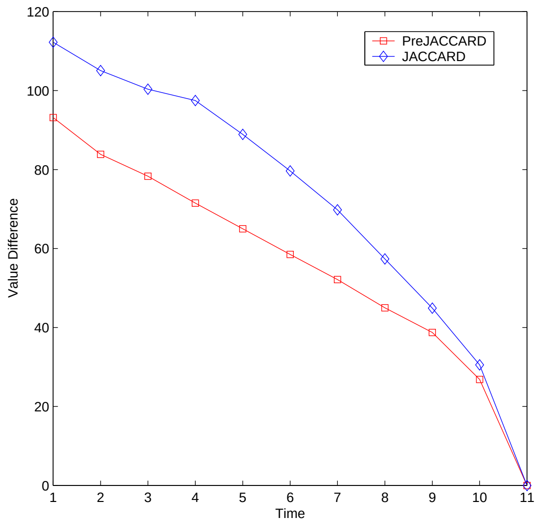


(c)-VD in real data

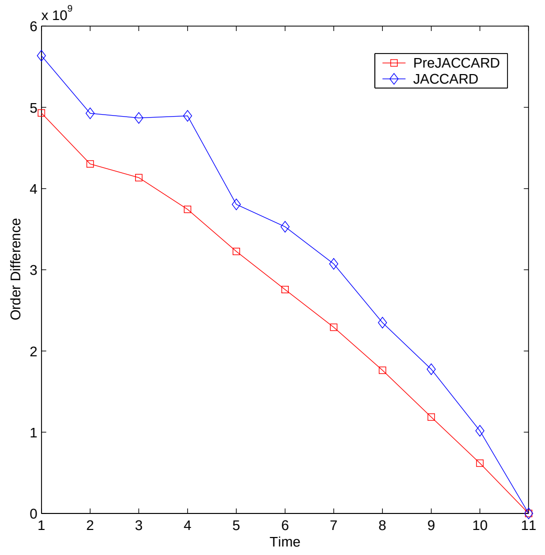


(b)-OD in real data

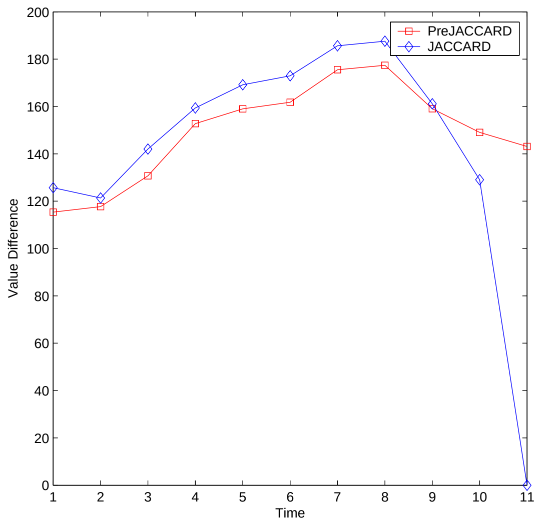
Figure 4.7: DiffusionRank comparison results



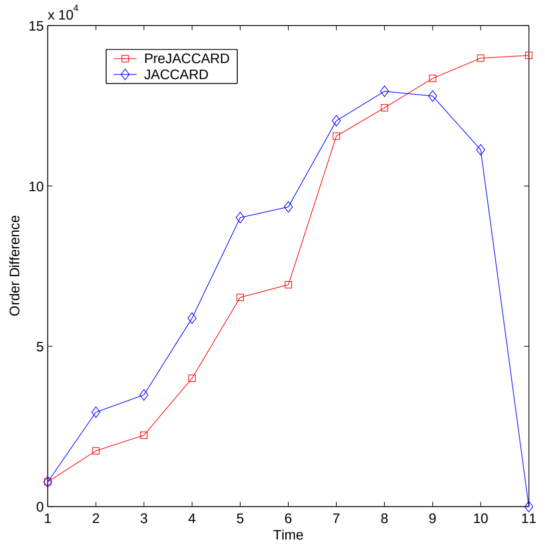
(a)-VD in synthetic data



(b)-OD in synthetic data

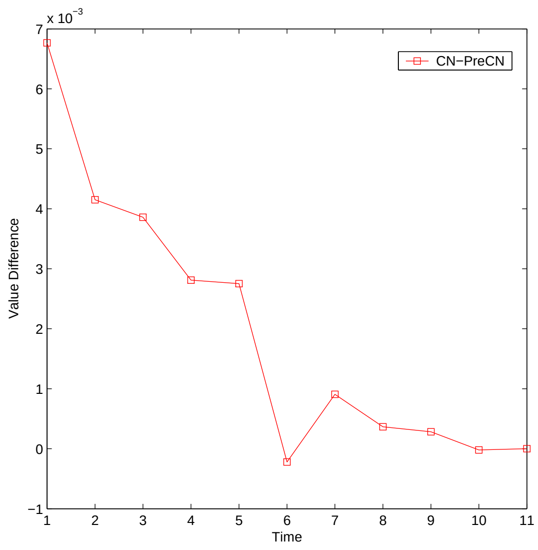


(c)-VD in real data

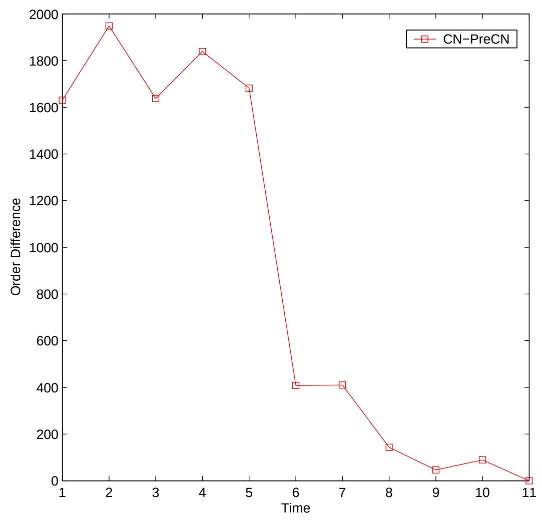


(d)-OD in real data

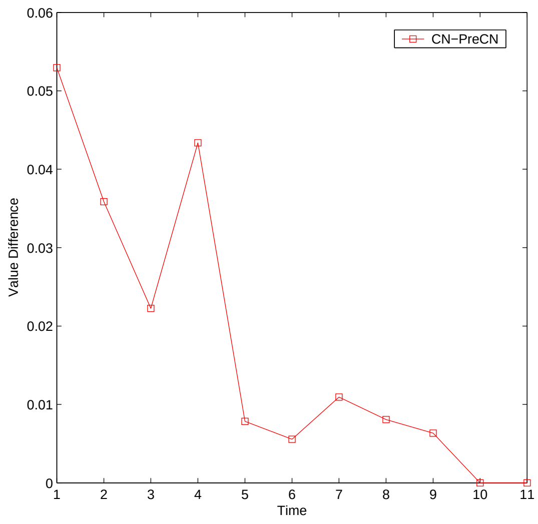
Figure 4.8: Jaccard's Coefficient comparison results



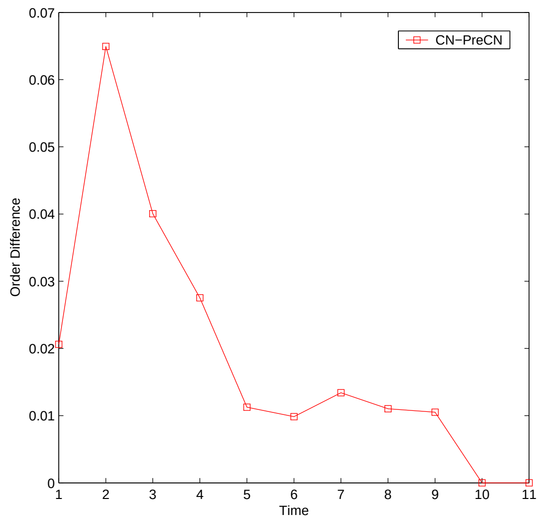
(a)-VD in synthetic data



(b)-OD in synthetic data



(c)-VD in real data



(b)-OD in real data

Figure 4.9: CN comparison results

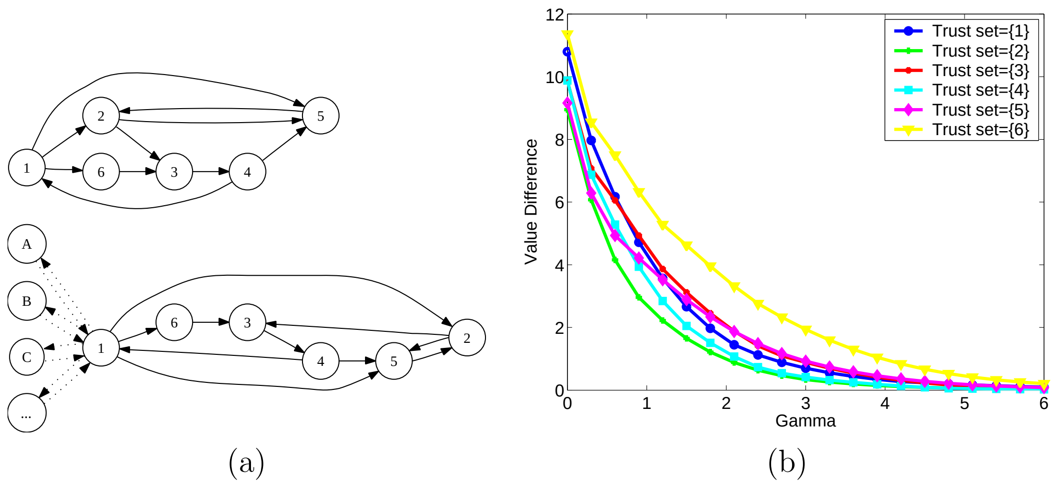


Figure 4.10: (a) The toy graph consisting of six nodes, and node 1 is being manipulated by adding new nodes $A, B, C, \ldots$ (b) The approximation tendency to *PageRank* by *DiffusionRank*

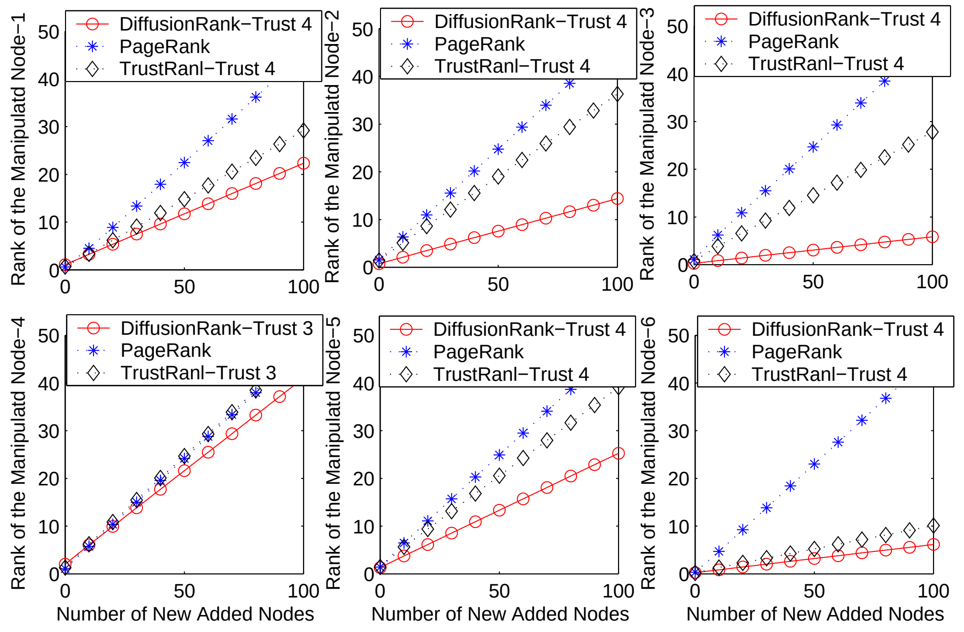
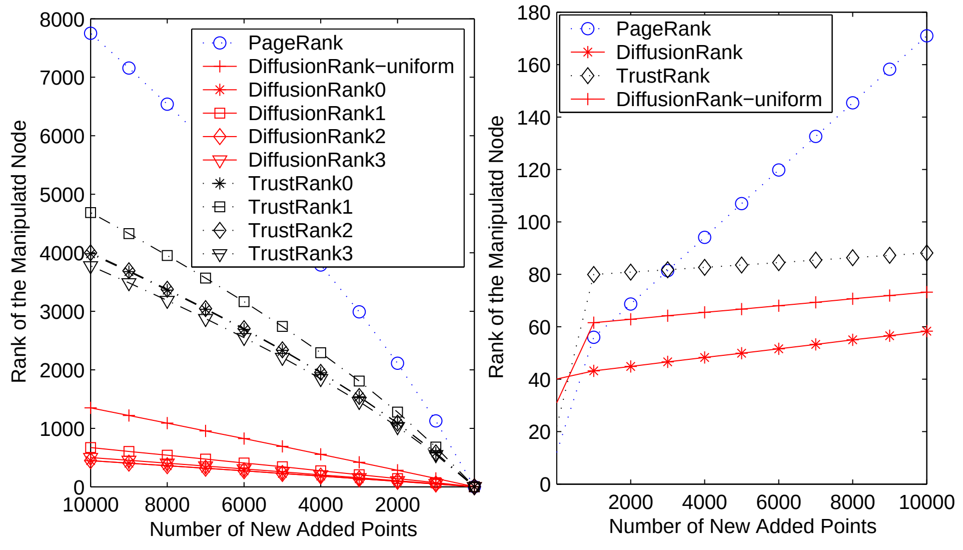
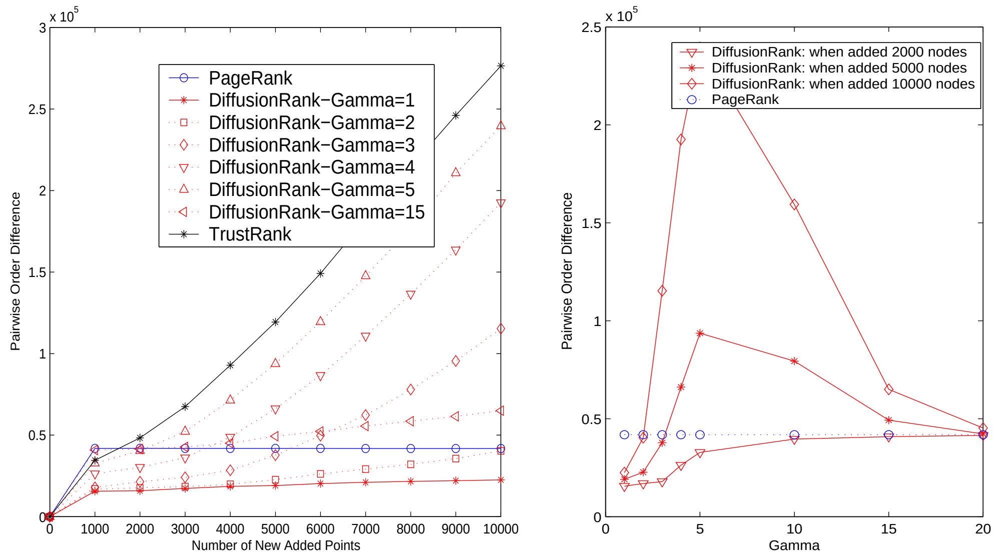


Figure 4.11: The rank values of the manipulated nodes on the toy graph



(a) (b)

Figure 4.12: (a) The rank values of the manipulated nodes on the middle-size graph; (b) The rank values of the manipulated nodes on the large-size graph



(a) (b)

Figure 4.13: (a) Pairwise order difference on the middle-size graph, the least it is, the more stable the algorithm; (b) The tendency of varying γ

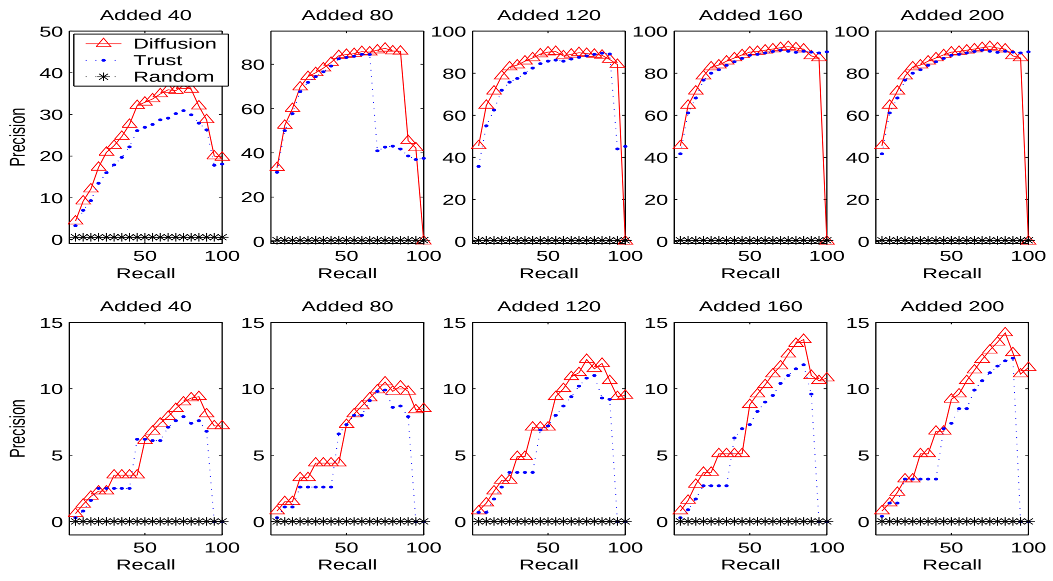


Figure 4.14: Precision vs Recall when $L = 50$: the larger the area below the curve, the better.

Chapter 5

Random Graph Dependency

In this chapter, we are interested in an information measure that can calculate the dependency between two random graphs. The proposed random graph measure is a general form whose applications are not restricted in the field of decision trees, however, in order to show strong motivations, the materials in this chapter are presented in an application-driven way. For the sake of easy understanding, this chapter progresses from a lower level to a higher level: we first show its one special case when random graphs degrade to equivalence relations, then we show its another special case appeared as a generalization of the conditional entropy, and then we generalize these two special cases to arrive at a general form. To show its power and to provide a support to the Graph-based Heat Diffusion Classifiers (*G-HDC*) in Chapter 3, we show an application of the random graph dependency measure in finding the free parameter in *G-HDC*. Although it is possible to find ways to support the model Predictive Random Graph Ranking (*PRGR*) in Chapter 4, e.g., maximizing the dependency between the Web graph and a unknown total order graph considered as a ranking, we leave them in the future work.

5.1 Motivations

The C4.5R8 algorithm is the fastest algorithm in terms of training time among a group of thirty-three tree-based, rule-based and statistics-based classification algorithms, while the prediction accuracy of the C4.5R8 algorithm is not statistically significantly different from POL, whose prediction accuracy is the best among these thirty-three classification algorithms [63]. In machine learning, there are endless interests to improve the accuracy and the speed of a well-known algorithm. In this chapter, we aim to develop a general information measure that can improve one single decision tree in speed by one special case or in accuracy by another special case. These two points are explained further in the next two sections.

5.1.1 Improve the Speed

We observe that, to compute a measure $\gamma(C, D)$ in Eq. (1.2) used in Rough Set Theory, we only need to carry out arithmetic operations, while the computation of the commonly used conditional entropy needs to compute the logarithm of the frequency, a time-consuming operation. In order to improve the speed of C4.5, we want to inherit this merit of γ, and avoid the drawback of γ in the meantime. The drawback of γ is its inaccuracy in measuring the dependency, and will be analyzed in the next example.

Example 8 In Table 5.1 where a, b, c, and d represent *headache, muscle pain, body temperature* and *influenza*, respectively, it is easy to calculate that $\gamma(C, D)=0$ when $C = \{a\}, D = \{d\}$. This happens because none of $C-$classes is completely contained in a $D-$class. But we observe that there is some dependency between $\{a\}$ and $\{d\}$. To measure the dependency more accurately, we propose the generalized dependency degree $\Gamma(C, D)$ so that the dependency between $\{a\}$ and $\{d\}$ is not measured as zero.

	headache (a)	**pain** (b)	**temperature** (c)	**influenza** (d)
e1	Y	Y	0	N
e2	Y	Y	1	Y
e3	Y	Y	2	Y
e4	N	Y	0	N
e5	N	N	1	N
e6	N	Y	2	Y
e7	Y	N	1	Y

Table 5.1: Influenza data (b)

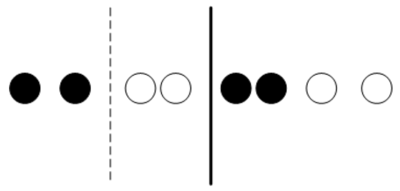


Figure 5.1: An illustration of eight points on the axis x_1, in which the black points belong to one class A while the white points belong to another class B.

In Section 5.2, we will continue to develop a deeper understanding of the generalized dependency degree, and to justify it both theoretically and empirically. On the theoretical side, we give its various forms and describe its properties; on the empirical side, we show its effectiveness in decision trees and in attribute selection.

5.1.2 Improve the Classification Accuracy

There are two observations that motivate us to generalize the traditional conditional entropy. The first observation is explained in the following example.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
x_1	1.0	2.0	3.9	4.0	5.0	5.1	7.0	8.0
x_2	Y	X	Y	Y	X	X	Y	X
y	A	A	B	B	A	A	B	B

Table 5.2: Eight points with three attributes

Example 9 In Figure 5.1, the eight points $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ in Table 5.2 are shown on the axis x_1. The black points belong to one class A

while the white points belong to another class B. When conditional entropy is applied to the cut shown by the dashed line, the information gain is 0.31. If the cut is chosen to be the solid line, then the information gain is 0. Similarly all the information gains for the seven possible cuts from the left to right are 0.14, 0.31, 0.05, 0.0, 0.05, 0.31, and 0.14. The maximal one is 0.31, and so the second cut is chosen as the best cut. On the other hand, for the attribute x_2, the information gain is 0.19, which is smaller than the best information gain 0.31 produced by attribute x_1. Then x_1 is chosen as the attribute for the root of the decision tree, on which there is a decision $x_1 \leq 2$. As the tree-building procedure continues, the whole data set is divided into two parts: $\{v_1, v_2\}$ determined by $x_1 \leq 2$, and $\{v_3, v_4, v_5, v_6, v_7, v_8\}$ determined by $x_1 > 2$. For attribute x_1, the information gains for the five possible cuts from left to right are $0.11, 0.25, 0.00, 0.25$, and 0.11. The maximal one is 0.25, and so the second cut is chosen as the best cut. For attribute x_2, the information gain is 0.46, which is greater than the best information gain of 0.25 produced by attribute x_2. Then x_2 is chosen as the attribute for the node following the root. The data set is further divided into two parts: $\{v_5, v_6, v_8\}$ determined by $x_2 = X$, $\{v_3, v_4, v_7\}$ determined by $x_2 = Y$. By the criteria in C4.5, when the cases in a node have the same class, or when the number of cases is less than 4, the building tree procedure stops. Therefore a decision tree is produced as shown in Figure 5.2, in which an error happens when $x_1 > 2, x_2 = X$. If the gain ratio is applied, the same decision tree is obtained. However, the best decision tree should be the one shown in Figure 5.3, which has no error, but cannot be produced by the traditional conditional entropy.

In the above example, the cut shown by the solid line does not produce any information gain. But it is observed that this middle cut makes the pattern at least equal to or better than what is produced by the cut marked by the dash line, in the sense that the number of further cuts needed is reduced in order

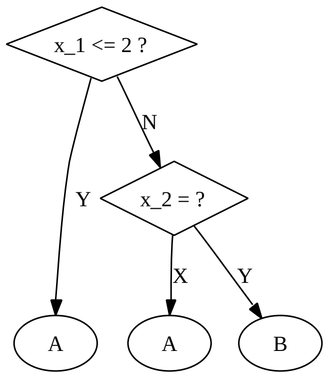


Figure 5.2: An illustration of a decision tree generated by conditional entropy, in which v_8 in Table 5.2 will be misclassified.

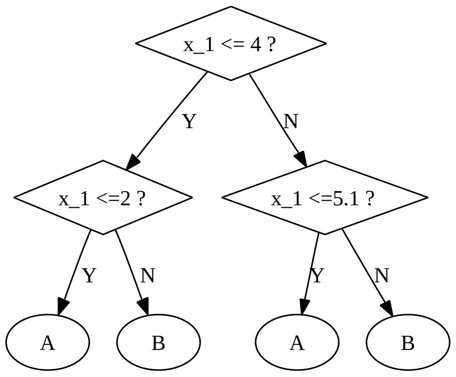
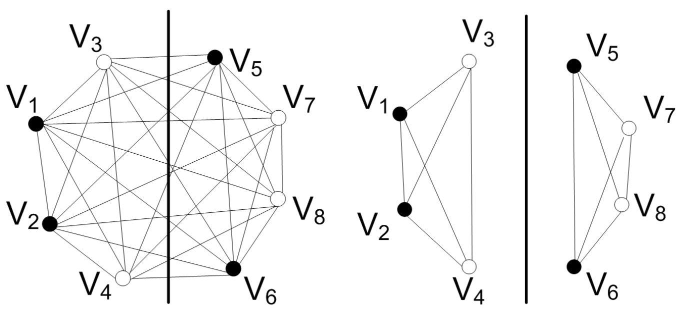


Figure 5.3: An ideal decision tree, which will be generated by the new measure, and in which no point is misclassified.

to distinguish the black points and the white points on both sides. This is one observation that motivates us to develop a new measure, by which a nonzero information gain can match the intuition that the pattern becomes equal or better after this middle cut, and by which the ideal decision tree shown in Figure 5.3 can be produced.

Another observation that encourages us to develop a new measure is shown in Figure 5.4, in which two points are linked by an edge when they are treated equally. Before the middle cut, the eight points in Figure 5.4 (a) are treated equally. After the middle cut, the four points on the left side of the cut are treated equally since they satisfy the same decision $x_1 \leq 4$, so are the four points on the right side of the cut. It is reasonable to consider the eight



(a)
Before the middle cut

(b)
After the middle cut

Figure 5.4: An illustration on how the eight points are treated in C4.5. Before the middle cut, the eight points are treated equally. After the middle cut, the four points on the left side of the cut are treated equally since they satisfy the same decision $x_1 \leq 4$, so are the four points on the right side of the cut.

points in the same way because “before the cut” can be understood as “no attribute information is employed to describe the eight points and so there is no difference between these eight points”. But after the cut, the continuous attribute has been employed to describe these eight points, and therefore the distance information should be considered. But in Figure 5.4 (b), we can see that the distance between the data points less (or greater) than the threshold (produced by the cut) is completely ignored. From the viewpoint of random graphs, we will explain the intuitions in this paragraph further in Section 5.3.2 after we define the new measure.

If the number of samples is enough, the error in Figure 5.2 cannot be produced, because there will be more cases left following the branch of $x_1 > 2$ and $x_2 = X$. This makes it possible to choose x_1 again in such a way that the error cannot be produced. So the inaccurate information gain becomes serious in small sample size problems, in which less informative attributes wrongly chosen by the inaccurate information measure will consume the samples quickly and thus leave little chance for employing the best attributes. Based on this consideration and the above two observations, we aim to define a measure in Section 5.3 that

- can include more distance information;
- can generalize the traditional entropy measure in the sense that when distance information is ignored or the attribute is discrete it degrades to the traditional measure;
- can improve the accuracy of C4.5 significantly on some benchmark datasets with small sample size.

5.1.3 Help to Search the Free Parameters in Heat Diffusion Classifiers

In Chapter 3, *G-HDC* achieves promising results in accuracy. However, there is not a guide to help to select the free parameters in *G-HDC*. The naive cross-validation method is time-consuming, as explained next. If the free parameters in *G-HDC* are sought by cross-validations, then at each fold of the cross-validation for each fixed K, each fixed β and each fixed γ, we need to do multiplications p times between a matrix $I+\frac{\gamma}{p}H$ and a vector $f(0)$, and $K*n*p$ multiplications are needed, where n is the number of data including both labeled data and unlabeled data. This observe motivates us to find time-saving methods for searching these free parameters, and the candidate method is proposed based on the random graph dependency measure so that the number $K*n*p$ is expected to be reduced to $K*n$.

5.2 The Generalized Dependency Degree $\Gamma(R_1, R_2)$ Between Two Equivalence Relations

Inspired by the dependency degree γ, a traditional measure in Rough Set Theory, we propose a generalized dependency degree, Γ, between two given sets of attributes. We first give its definition in terms of equivalence relations, then interpret it in terms of minimal rules, and further describe the algorithm for its computation. To understand Γ better, we investigate its various properties. We further extend Γ to information systems with missing values, called incomplete information systems. To show its advantage, we make a comparative study with the conditional entropy and γ in a number of experiments. This section is organized as follows. In Section 5.2.1, we give the first two forms of the generalized dependency degree. In Section 5.2.2, we discuss the properties of this measure and give its third form. In Section 5.2.3, we extend it to incomplete information systems. In Section 5.2.4, we compare it with conditional entropy. In Section 5.2.5, we conduct experiments to support the generalized dependency degree concept. In Section 5.2.6, we draw a conclusion about the generalized dependency degree.

5.2.1 Definition of the Generalized Dependency Degree

In this section, we first cite the formal language in complete information systems, which is used to describe the decision rules. Then we give the definition of the generalized dependency degree. Finally we connect the minimal decision rules and the generalized dependency degree.

A Formal Language to Describe the Decision Rule

The decision language is defined in [79, 80]. Let $S = (U, A, V, f)$ be an information system. With every $B \subseteq A$ we associate a formal language, i.e., a set of formulae $For(B)$. Formulae of $For(B)$ are built up from attribute-value pairs $a = v$ where $a \in B$ and $v \in V_a$ by means of logical connectives $\wedge$ (and), $\vee$ (or), $\sim$(not) in the standard way. For any $\Phi \in For(B)$, we denote the set of all objects satisfying Φ by $supp(\Phi)$; this is called the support of Φ.

A decision rule in S is an expression $\Phi \to \Psi$, where $\Phi \in For(C)$, $\Psi \in For(D)$, C, D are condition and decision attributes respectively, and Φ and Ψ

are referred to as the condition and decision of the rule respectively. Related to a decision rule, we will need the following three definitions.

Definition 10 A decision rule $\Phi \to \Psi$ is called a *deterministic rule* in S if $supp(\Phi) \subseteq supp(\Psi)$, and an *indeterministic rule* otherwise.

Definition 11 With every decision rule $\Phi \to \Psi$ we associate a conditional probability called the *confidence* (the *certainty factor* of the rule $\Phi \to \Psi$), and denote it by $Con(\Phi \to \Psi)$, which can be written as

$$Con(\Phi \to \Psi) = \frac{|supp(\Phi \wedge \Psi)|}{|supp(\Phi)|}.$$

Definition 12 We denote the *strength* of decision rule $\Phi \to \Psi$ by $Str(\Phi \to \Psi)$, which is defined as:

$$Str(\Phi \to \Psi) = \frac{|supp(\Phi \wedge \Psi)|}{|U|}.$$

The denominator $|supp(\Phi)|$ in $Con(\Phi \to \Psi)$ only counts the objects that satisfy the formula Φ while the denominator $|U|$ in $Str(\Phi \to \Psi)$ counts all the objects.

We show the definitions of the confidence and strength of a rule by the following example.

Example 13 In Table 5.1, $A = \{a, b, c, d\}$, $U = \{e1, e2, e3, e4, e5, e6, e7\}$. Let $C = \{a, b\}$, and $D = \{d\}$. We consider the rule $a = Y \wedge b = Y \to d = Y$. The condition part Φ is equal to $a = Y \wedge b = Y$ while decision part Ψ is equal to $d = Y$, so $\Phi \wedge \Psi$ is equal to $a = Y \wedge b = Y \wedge d = Y$. Since $supp(\Phi) = \{e1, e2, e3\}$ and $supp(\Phi \wedge \Psi) = \{e2, e3\}$, we have $|supp(\Phi)| = 3$ and $|supp(\Phi \wedge \Psi)| = 2$, and so $Con(\Phi \to \Psi) = 2/3$, and $Str(\Phi \to \Psi) = 2/7$.

Generalized Dependency Degree

We give our first form of the generalized dependency degree $\Gamma(C, D)$ in terms of equivalence relations as follows.

Definition 14 The generalized dependency degree $\Gamma(C, D)$ is defined as

$$\Gamma(C, D) = \frac{1}{|U|} \sum_{x \in U} \frac{|D(x) \cap C(x)|}{|C(x)|}, \tag{5.1}$$

where $D(x)$ and $C(x)$ denote the D-class containing x and C-class containing x respectively (recall that, in the introduction section, we defined P-class for any attribute set P).

Note that, the dependency degree $\gamma(C, D)$ can be rewritten as

$$\gamma(C, D) = \frac{1}{|U|} \sum_{x \in U \wedge C(x) \subseteq D(x)} \frac{|D(x) \cap C(x)|}{|C(x)|}. \tag{5.2}$$

From this form of $\gamma(C, D)$, one can discern the difference between $\Gamma(C, D)$ and $\gamma(C, D)$ easily. In $\gamma(C, D)$, if $|D(x) \cap C(x)|/|C(x)| < 1$, then x is not counted, while in $\Gamma(C, D)$ every object is counted by a fraction $|D(x) \cap C(x)|/|C(x)|$ that may not be equal to 1.

We show the definition of $\Gamma(C, D)$ by the following two examples.

Example 15 In Table 5.1, $A = \{a, b, c, d\}$, $U = \{e1, e2, e3, e4, e5, e6, e7\}$. We use Eq. (5.1) to calculate $\Gamma(C, D)$ when $C = \{a, b, c\}$, $D = \{d\}$. Since $C(e1) = \{e1\}$, $C(e2) =\{e2\}$, $C(e3) =\{e3\}$, $C(e4) =\{e4\}$, $C(e5) =\{e5\}$, $C(e6) =\{e6\}$, $C(e7) =\{e7\}$, $D(e1) =D(e4) = D(e5) =\{e1, e4, e5\}$, $D(e2) = D(e3) = D(e6) = D(e7) = \{e2, e3, e6, e7\}$, we have $\Gamma(C, D) = \frac{1}{7}(\frac{|D(e1) \cap C(e1)|}{|C(e1)|} + \frac{|D(e2) \cap C(e2)|}{|C(e2)|} + \frac{|D(e3) \cap C(e3)|}{|C(e3)|} + \frac{|D(e4) \cap C(e4)|}{|C(e4)|} + \frac{|D(e5) \cap C(e5)|}{|C(e5)|} + \frac{|D(e6) \cap C(e6)|}{|C(e6)|} + \frac{|D(e7) \cap C(e7)|}{|C(e7)|})$
$= (1 + 1 + 1 + 1 + 1 + 1 + 1)/7 = 1$.

Example 16 Also in Table 5.1, we calculate $\Gamma(C, D)$ and $\gamma(C, D)$ when $C = \{a\}$, $D = \{d\}$. Since $C(e1) = C(e2) = C(e3) = C(e7) = \{e1, e2, e3, e7\}, C(e4) =$

$C(e5) = C(e6) = \{e4, e5, e6\}$, $D(e1) = D(e4) = D(e5) = \{e1, e4, e5\}$, $D(e2) = D(e3) = D(e6) = D(e7) = \{e2, e3, e6, e7\}$, we have $\Gamma(C, D) = (1/4+3/4+3/4+2/3+2/3+1/3+3/4)/7 = 25/42$, while we have $\gamma(C, D) = 0$ according to Eq. (5.2).

Next we will interpret $\Gamma(C, D)$ from another point of view, i.e., we will change our view from the equivalence classes to minimal decision rules.

Connection between the Generalized Dependency Degree and Minimal Rule

We first give the definition of the minimal formula.

Definition 17 (Minimal Formula) A formula $\Phi \in For(B)$ is called a minimal formula in $For(B)$ if $supp(\Phi) \neq \emptyset$, and for any $\Psi \in For(B)$, $supp(\Psi) \subset supp(\Phi)$ implies $supp(\Psi) = \emptyset$.

The minimal formula has the meaning that the support of a formula $\Psi \in For(B)$ cannot be smaller than the support of the minimal formula unless Ψ has an empty support. For example, in Table 5.1, let $B = \{a, b\}$, then $a = Y \wedge b = Y$ is a minimal formula in $For(B)$. The support of $a = Y \wedge b = Y$ is $\{e1, e2, e3\}$.

We call a rule a minimal rule if both its condition part and decision part are minimal formulae, and we define the minimal rule formally in the following.

Definition 18 (Minimal Rule) Let $C = \{c_1, c_2, c_3, ..., c_n\}$, $D = \{d_1, d_2, d_3, ..., d_m\}$. Then we call the rule

$$c_1 = u_1 \ \wedge \ c_2 = u_2 \ \wedge ... \wedge c_n = u_n \ \rightarrow d_1 = v_1 \ \wedge \ d_2 = v_2 \wedge ... \wedge d_m = v_m$$

a minimal rule, where $u_1 \in V_{c_1}, u_2 \in V_{c_2}, u_3 \in V_{c_3}, \cdots, u_n \in V_{c_n}, v_1 \in V_{d_1}, v_2 \in V_{d_2}, \cdots, v_m \in V_{d_m}$.

If $x \in U$, by $C(x) \to D(x)$ we denote the rule $c_1 = c_1(x) \ \wedge \ c_2 = c_2(x) \ \wedge \ c_3 = c_3(x) \wedge ... \wedge c_n = c_n(x) \ \to d_1 = d_1(x) \ \wedge \ d_2 = d_2(x) \wedge ... \wedge d_m = d_m(x)$, where $C(x)$ is the C-Class containing x, $D(x)$ is the D-class containing x, $c_i(x)$ is the value of x at the attribute c_i, and $d_j(x)$ is the value of x at the attribute d_j.

Note that the rule $C(x) \to D(x)$ is a minimal rule, and that any minimal rule, whose confidence and strength are not equal to zero, can be written as $C(x) \to D(x)$.

Let $MinR(C, D)$ be the set of all the minimal rules, r be any rule in $MinR(C, D)$, $Con(r)$ be the confidence of the rule r, and $Str(r)$ be the strength of the rule r. Then

$$\sum_{r \in MinR(C,D)} Str(r) \cdot Con(r), \tag{5.3}$$

the weighted average of the confidence $Con(r)$ of minimal rule r weighted by the strength $Str(r)$, is exactly the generalized dependency degree $\Gamma(C, D)$. This is our second form of the generalized dependency degree $\Gamma(C, D)$, which is defined in terms of minimal rules. We explain this by the following.

Let X be a $(C \cup D)$-class. Then for any $y, x \in X$, y has the same values as x at the attributes in $(C \cup D)$, and so y has the same values as x at the attributes in both C and D, i.e., y and x are both in the same C-class and in the same D-class, i.e., $C(y) = C(x)$, $D(y) = D(x)$, and therefore for any $x \in X$ we can denote $C(x)$ by $C(X)$, $D(x)$ by $D(X)$, and $|D(x) \cap C(x)|/|C(x)|$ by $|D(X) \cap C(X)|/|C(X)|$. Since $|X| = |D(X) \cap C(X)|$, we have

$$\begin{aligned}
\Gamma(C, D) &= \frac{1}{|U|} \sum_{x \in U} \frac{|D(x) \cap C(x)|}{|C(x)|} \\
&= \frac{1}{|U|} \sum_{X \in U/(C \cup D)} \sum_{x \in X} \frac{|D(x) \cap C(x)|}{|C(x)|} \\
&= \frac{1}{|U|} \sum_{X \in U/(C \cup D)} \sum_{x \in X} \frac{|D(X) \cap C(X)|}{|C(X)|}
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{|U|}\sum_{X\in U/(C\cup D)} |X|\frac{|D(X)\cap C(X)|}{|C(X)|} \\
&= \frac{1}{|U|}\sum_{X\in U/(C\cup D)} \frac{|D(X)\cap C(X)|^2}{|C(X)|} \\
&= \sum_{X\in U/(C\cup D)} \frac{1}{|U|}\frac{|D(X)\cap C(X)|^2}{|C(X)|} \\
&= \sum_{X\in U/(C\cup D)} \frac{|D(X)\cap C(X)|}{|U|}\cdot\frac{|D(X)\cap C(X)|}{|C(X)|} \\
&= \sum_{X\in U/(C\cup D)} Str(C(X)\to D(X))\cdot Con(C(X)\to D(X)) \\
&= \sum_{r\in MinR(C,D)} Str(r)\cdot Con(r)
\end{aligned}$$

The dependency degree $\gamma(C, D)$ can be rewritten correspondingly as

$$\gamma(C,D) = \sum_{r\in MinR(C,D)\,\wedge\, Con(r)=1} Str(r)\cdot Con(r),$$

which means that in $\gamma(C, D)$, only those minimal rules whose confidences are equal to 1 are counted while in $\Gamma(C, D)$, every minimal rule whose confidence is not equal to zero is counted. In other words, $\gamma(C, D)$ only counts deterministic minimal rules while $\Gamma(C, D)$ counts both deterministic minimal rules and indeterministic minimal rules.

In fact, we can include $\gamma(C, D)$ and $\Gamma(C, D)$ in a general form $\gamma^{\varepsilon}(C, D)$, which is defined as

$$\gamma^{\varepsilon}(C,D) = \sum_{r\in MinR(C,D)\,\wedge\, Con(r)\geq\varepsilon} Str(r)\cdot Con(r).$$

When $\varepsilon = 0$, $\gamma^{\varepsilon}(C, D)$ becomes $\Gamma(C, D)$, while when $\varepsilon = 1$, $\gamma^{\varepsilon}(C, D)$ becomes $\gamma(C, D)$. In this section, we only focus on $\Gamma(C, D)$.

5.2.2 Properties of the Generalized Dependency Degree

Recall that in the introduction section, we define the P-indiscernibility relation for a subset P of attributes, denoted by $IND(P)$, which is an equivalence relation on U, the universe of objects. $\Gamma(C,D)$ is actually defined on two equivalence relations induced by subsets C and D of attributes. The definition of $\Gamma(C,D)$ can be easily generalized to the definition of $\Gamma(R_1,R_2)$ for any two equivalence relations R_1 and R_2 on the universe U as follows:

$$\Gamma(R_1,R_2) = \frac{1}{|U|}\sum_{x\in U}\frac{|R_2(x)\cap R_1(x)|}{|R_1(x)|}, \tag{5.4}$$

$$\Gamma(R_1,R_2) = \sum_{r\in MinR(R_1,R_2)} Str(r)\cdot Con(r). \tag{5.5}$$

Here the set $MinR(R_1,R_2)$ is the set of all the minimal rules, r is any rule in $MinR(R_1,R_2)$, and by $Con(r)$ and $Str(r)$ we denote the confidence and strength of the rule r, respectively.

Definition 19 The minimal rule in $MinR(R_1,R_2)$ is defined as

$$x\in\ G \quad\rightarrow\quad x\in H, \tag{5.6}$$

where G and H are any R_1-class and R_2-class, respectively.

Note that $\Gamma(C,D)=\Gamma(IND(C),IND(D))$. In fact, the rough set model is extended to any binary relation [65]. Eq. (5.4) is a general form, in which the equivalence relations can be understood as any binary relations. This explains why we can say that the first form of the generalized dependency degree is a flexible form. In this section, we only focus on the case of equivalence relations.

The definition of $\gamma(C,D)$ can also be generalized to $\gamma(R_1,R_2)$ for any equivalence relations R_1,R_2 on the universe U. We rewrite $\gamma(R_1,R_2)$ as follows:

$$\gamma(R_1,R_2) = \frac{1}{|U|}\sum_{x\in U\,\wedge\,R_1(x)\subseteq R_2(x)}\frac{|R_2(x)\cap R_1(x)|}{|R_1(x)|}, \tag{5.7}$$

$$\gamma(R_1,R_2) = \frac{1}{|U|}\sum_{x\in U\,\wedge\,R_1(x)\subseteq R_2(x)} Str(r)\cdot Con(r). \tag{5.8}$$

Throughout the rest of this section, all the relations we use are all on the finite universe U, and the set of all equivalence relations on U is denoted by $\mathcal{ER}(U)$. By the definition of $\gamma(R_1, R_2)$ and $\Gamma(R_1, R_2)$ we have the following theorem:

Theorem 20 For any equivalence relations R_1 and R_2, the inequality $0 \leq \gamma(R_1, R_2) \leq \Gamma(R_1, R_2) \leq 1$ holds.

This Theorem shows that $\Gamma(R_1, R_2)$ can serve as an index because it is less than or equal to one and larger than or equal to zero. Moreover, it reveals the relation between $\Gamma(R_1, R_2)$ and $\gamma(R_1, R_2)$. By the next theorem, we will show their relation further in the extreme condition that one of them is equal to one.

Theorem 21 For any equivalence relations R_1 and R_2,

$$\gamma(R_1, R_2) = 1 \Leftrightarrow \Gamma(R_1, R_2) = 1 \Leftrightarrow \gamma(R_1, R_2) = \Gamma(R_1, R_2).$$

Proof: According to Eqs (5.4) and (5.7), the conclusion follows immediately. ■

By the following theorem, we will reveal how $\Gamma(R_1, R_2)$ changes when the second equivalence relation R_2 is changed to be larger.

Theorem 22 (Partial Order Preserving Property) For any equivalence relations R_1, R_2 and R. If $R_2 \subseteq R$, then $\Gamma(R_1, R_2) \leq \Gamma(R_1, R)$.

Proof: According to Eq. (5.4), the conclusion follows immediately. ■

This means that the finer the equivalence relation R_2 is, the less the equivalence relation R_2 depends on the equivalence relation R_1. From the viewpoint of classification, the more the decision attribute values group together, i.e., the larger the equivalence class induced by the decision attribute is, the easier we can classify the objects into the new D-class by employing attribute C.

Example 23 In Table 5.1, Let $C = \{a\}$, $D = \{d\}$, $V_d = \{Y, N\}$; if we group Y and N together such that both Y and N become a new value Z, then $D' = \{d\}$, $V_d = \{Z\}$, and Table 5.1 becomes Table 5.3. D induces the equivalence relation $IND(D)$, and the set of the equivalence classes is calculated as $U/D = \{\{e1, e4, e5\}, \{e2, e3, e6, e7\}\}$; on the other hand, D' induces the equivalence relation $IND(D')$, and $U/D' = \{\{e1, e2, e3, e4, e5, e6, e7\}\}$. Let $R_1 = IND(C)$, $R_2 = IND(D)$, $R = IND(D') = U \times U$, then $\Gamma(R_1, R_2) = 25/42$ as shown in Example 16. For each $x \in U$, $R(x) = U$, so we have $R(x) \cap R_1(x) = R_1(x)$, and therefore $\Gamma(R_1, R) = 1/|U| \sum_{x \in U} |R(x) \cap R_1(x)|/|R_1(x)| = 1/|U| \sum_{x \in U} |R_1(x)|/|R_1(x)| = 1$. The inequality $\Gamma(R_1, R_2) \leq \Gamma(R_1, R)$ means that we can classify objects into U/D' more easily than into U/D.

	a	b	c	d
e1	Y	Y	0	Z
e2	Y	Y	1	Z
e3	Y	Y	2	Z
e4	N	Y	0	Z
e5	N	N	1	Z
e6	N	Y	2	Z
e7	Y	N	1	Z

Table 5.3: Influenza data (c)

Theorem 22 leads to the following theorem, which shows the properties of $\Gamma(R_1, R_2)$ when R_2 becomes the smallest equivalence relation (the identity relation) or the largest equivalence relation (the universal relation).

Theorem 24 For any given equivalence relation R_1.

$$\min_{R_2 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = \Gamma(R_1, I_U), \quad \max_{R_2 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = \Gamma(R_1, U \times U) = 1,$$

where I_U is the identity relation on U, and $U \times U$ is the universal relation on U.

Proof: Since $I_U \subseteq R_2 \subseteq U \times U$, the conclusion follows immediately by Theorem 22. ■

In order to obtain more information about properties of the generalized dependency degree $\Gamma(R_1, R_2)$, we need the following lemma.

Lemma 25 The inequality

$$\frac{a_1^2}{b_1} + \frac{a_2^2}{b_2} + \cdots + \frac{a_n^2}{b_n} \geq \frac{(a_1 + a_2 + \cdots a_n)^2}{b_1 + b_2 + \cdots + b_n} \tag{5.9}$$

holds for any real number a_i, and real number $b_i > 0$, $i = 1, 2, \ldots, n$.

Proof: It is well known that for any function $f(x)$ in which $f''(x) > 0$, the inequality

$$f(\mu_1 x_1 + \mu_2 x_2 + \cdots + \mu_n x_n) \leq \mu_1 f(x_1) + \mu_2 f(x_2) + \cdots + \mu_n f(x_n)$$

holds if $\mu_1, \mu_2, \ldots, \mu_n \geq 0, \quad \mu_1 + \mu_2 + \cdots + \mu_n = 1$. In this inequality, let

$$f(x) = x^2, x_i = (b_1 + b_2 + \cdots + b_n)a_i/b_i, \mu_i = b_i/(b_1 + b_2 + \cdots + b_n),$$

$i = 1, 2, \ldots, n$. Then the desired inequality follows. ■

In Theorem 22, we have shown the partial order preserving property of $\Gamma(R_1, R_2)$ on the second item. By the next theorem, we continue to show the anti-partial order preserving property of $\Gamma(R_1, R_2)$ on the first item. We first show the third form of the generalized dependency degree.

Suppose that there are m R_2-classes, denoted by $X_1, X_2, X_3, \ldots, X_m$. Then we analyze $\Gamma(R_1, R_2)$ for any given R_1. In order to achieve this goal, we need to examine the set X_i. We assume there are k_i different nonempty subsets of X_i of the form $R_1(x) \cap X_i$, for $i = 1, 2, ..., m$. Note that for any $x, y \in U$, either $R_1(x) = R_1(y)$ or $R_1(x) \cap R_1(y) = \phi$, and $\cup_{x \in U} R_1(x) = U$. So we can assume that these k_i different nonempty subsets of X_i take the forms

$$R_1(x_{i1}) \cap X_i,\ R_1(x_{i2}) \cap X_i,\ \ldots,\ R_1(x_{ik_i}) \cap X_i,$$

and they satisfy

$$(R_1(x_{ip}) \cap X_i)\ \cap\ (R_1(x_{iq}) \cap X_i) = \phi,$$

for $p \neq q$, $p, q = 1, 2, \ldots, k_i$; and

$$\bigcup_{p=1}^{k_i} (R_1(x_{ip}) \cap X_i) = X_i.$$

Note that for $y \in R_1(x_{ij}) \cap X_i$,

$$R_1(y) \cap X_i = R_1(x_{ij}) \cap X_i.$$

By Eq. (5.4), we have

$$\begin{aligned}
\Gamma(R_1, R_2) &= \frac{1}{|U|} \sum_{i=1}^{m} \sum_{x \in X_i} \frac{|X_i \cap R_1(x)|}{|R_1(x)|} \\
&= \frac{1}{|U|} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \sum_{x \in R_1(x_{ij}) \cap X_i} \frac{|X_i \cap R_1(x_{ij})|}{|R_1(x_{ij})|} \\
&= \frac{1}{|U|} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \frac{|X_i \cap R_1(x_{ij})|^2}{|R_1(x_{ij})|} \\
&= \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \frac{|U|}{|R_1(x_{ij})|} |X_i \cap R_1(x_{ij})|^2. \qquad (5.10)
\end{aligned}$$

Eq. (5.10) is our third form of the generalized dependency degree.

Among our three different forms of the generalized dependency degree, the first form of the measure (in terms of equivalence relations) is the most important. Besides its simplicity, the first form is flexible, and it can therefore be extended not only to an equivalence relation but also to an arbitrary relation and a random graph. The first form (in terms of equivalence relations) and the second form (in terms of minimal rules) share the advantage of being easily understood, while the third form of the measure (in terms of arithmetic operations) is computationally efficient. So these three forms of the measure are suited to different situations. When we want to extend the measure to a more complicated data structure (such as partial order relation, totally order relation or random graphs) than an equivalence relation, or when we want to find some properties of this measure, we can employ the first two forms of the measure. When we use it in a computing situation, the third form of the measure may be the best choice. In fact, in this section, we determine its properties using the first two forms, and then in the experiments we use the third form.

In Algorithm 2, we give the complete description of the computation of $\Gamma(C, D)$ according to Eq. (5.10).

Algorithm 2 Input: $S = (U, A, V, f)$: an information table; C, D: two attribute sets. Output $\Gamma(C, D)$ PROCEDURE $\Gamma(C, D)$

```
Find all the $D-$ classes $X(1), X(2), \ldots, X(m)$ and all the $C-$ classes $Y(1), Y(2), \ldots, Y(n)$.
$Total \leftarrow$ the number of cases
for $j = 1$ TO $n$ do
    $b(j) \leftarrow$ the number of cases in $Y(j)$
end for
$\Gamma \leftarrow 0$
for $i = 1$ TO $m$ do
    for $j = 1$ TO $n$ do
        $a(i, j) \leftarrow$ the number of cases in $Y(j) \cap X(i)$
        $\Gamma \leftarrow \Gamma + a(i, j) * a(i, j)/b(j)$
    end for
end for
$\Gamma \leftarrow \Gamma/Total$
RETURN
```

By Theorem 22, we show the partial order preserving property of the generalized dependency degree on the second item, in the following, we continue to show the anti-partial order preserving property of the generalized dependency degree on the first item.

Theorem 26 (Anti-Partial Order Preserving Property) For any equivalence relations R_1, R_2, and R. If $R_1 \subseteq R$, then $\Gamma(R_1, R_2) \geq \Gamma(R, R_2)$.

Proof: Since $R_1 \subseteq R$, each R-class is the union of some R_1-classes, and each set $R(y_j) \cap X_i$ is the union of some sets of the form $R(x_j) \cap X_i$. We assume that, in X_i, there are l_i different nonempty subsets of the form $R(y_{ij}) \cap X_i$. We assume without loss of generality that

$$\begin{aligned}
R(y_{i1}) \cap X_i &= (R_1(x_{i1}) \cap X_i) \cup (R_1(x_{i2}) \cap X_i) \cup \cdots \cup (R_1(x_{ip_1}) \cap X_i), \\
R(y_{i1}) &\supseteq R_1(x_{i1}) \cup R_1(x_{i2}) \cup \cdots \cup R_1(x_{ip_1}), \\
R(y_{i2}) \cap X_i &= (R_1(x_{ip_1+1}) \cap X_i) \cup (R_1(x_{ip_1+2}) \cap X_i) \cup \cdots \cup (R_1(x_{ip_2}) \cap X_i), \\
R(y_{i2}) &\supseteq R_1(x_{ip_1+1}) \cup R_1(x_{ip_1+2}) \cup \cdots \cup R_1(x_{ip_2}), \\
&\vdots \\
R(y_{il_i}) \cap X_i &= (R_1(x_{ip_{l_i-1}+1}) \cap X_i) \cup (R_1(x_{ip_{l_i-1}+2}) \cap X_i) \cup \cdots \cup (R_1(x_{ip_{l_i}}) \cap X_i), \\
R(y_{il_i}) &\supseteq R_1(x_{ip_{l_i-1}+1}) \cup R_1(x_{ip_{l_i-1}+2}) \cup \cdots \cup R_1(x_{ip_{l_i}}).
\end{aligned}$$

Using Eq. (5.10), we have

$$\begin{aligned}
\Gamma(R_1, R_2) &= \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \frac{|U|}{|R_1(x_{ij})|} |X_i \cap R_1(x_{ij})|^2 \\
&= \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \frac{a_{ij}^2}{b_{ij}},
\end{aligned}$$

where $a_{ij} = |X_i \cap R_1(x_{ij})|$, $b_{ij} = |R_1(x_{ij})|/|U|$, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, k_i$;

$$\begin{aligned}
\Gamma(R, R_2) &= \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{l_i} \frac{|U|}{|R(y_{ij})|} |X_i \cap R(y_{ij})|^2 \\
&= \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{l_i} \frac{{a'_{ij}}^2}{b'_{ij}},
\end{aligned}$$

where

$$\begin{aligned}
a'_{i1} &= |X_i \cap R(y_{i1})| &&= a_{i1} + a_{i2} + \cdots + a_{ip_1}, \\
a'_{i2} &= |X_i \cap R(y_{i2})| &&= a_{ip_1+1} + a_{ip_1+2} + \cdots + a_{ip_2}, \\
&\vdots \\
a'_{il_i} &= |X_i \cap R(y_{il_i})| &&= a_{ip_{l_i-1}+1} + a_{ip_{l_i-1}+2} + \cdots + a_{ip_{l_i}}, \\
b'_{i1} &= |R(y_{i1})|/|U| &&\geq b_{i1} + b_{i2} + \cdots + b_{ip_1}, \\
b'_{i2} &= |R(y_{i2})|/|U| &&\geq b_{ip_1+1} + b_{ip_1+2} + \cdots + b_{ip_2}, \\
&\vdots \\
b'_{il_i} &= |R(y_{il_i})|/|U| &&\geq b_{ip_{l_i-1}+1} + b_{ip_{l_i-1}+2} + \cdots + b_{ip_{l_i}},
\end{aligned}$$

$i = 1, 2, \ldots, m$. By Lemma 25, we have

$$\begin{aligned}
\sum_{j=1}^{k_i} \frac{a_{ij}^2}{b_{ij}} &= \frac{a_{i1}^2}{b_{i1}} + \frac{a_{i2}^2}{b_{i2}} + \cdots + \frac{a_{ip_1}^2}{b_{ip_1}} \\
&+ \frac{a_{ip_1+1}^2}{b_{ip_1+1}} + \frac{a_{ip_1+2}^2}{b_{ip_1+2}} + \cdots + \frac{a_{ip_2}^2}{b_{ip_2}} \\
&+ \cdots \\
&+ \frac{a_{ip_{l_i-1}+1}^2}{b_{ip_{l_i-1}+1}} + \frac{a_{ip_{l_i-1}+2}^2}{b_{ip_{l_i-1}+2}} + \cdots + \frac{a_{ip_{l_i}}^2}{b_{ip_{l_i}}} \\
&\geq \frac{(\sum_{j=1}^{p_1} a_{ij})^2}{\sum_{j=1}^{p_1} b_{ij}} + \frac{(\sum_{j=p_1+1}^{p_2} a_{ij})^2}{\sum_{j=p_1+1}^{p_1} b_{ij}} + \cdots + \frac{(\sum_{j=p_{l_i-1}+1}^{p_{l_i}} a_{ij})^2}{\sum_{j=p_{l_i-1}+1}^{p_{l_i}} b_{ij}} \\
&\geq \frac{{a'_{i1}}^2}{b'_{i1}} + \frac{{a'_{i2}}^2}{b'_{i2}} + \cdots + \frac{{a'_{il_i}}^2}{b'_{il_i}}.
\end{aligned}$$

Therefore $\Gamma(R_1, R_2) = \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \frac{a_{ij}^2}{b_{ij}} \geq \frac{1}{|U|^2} \sum_{i=1}^{m} \sum_{j=1}^{l_i} \frac{{a'_{ij}}^2}{b'_{ij}} = \Gamma(R, R_2)$. ■

This means that the finer the equivalence relation R_1 is, the more R_2 depends on R_1. From the viewpoint of classification, the more the condition attribute values group together, i.e., the larger the equivalence class induced by the decision attribute is, the more difficult it is to classify the objects into the new D-class by employing attribute C.

Example 27 In Table 5.1, let $C = \{c\}$, $V_c = \{0, 1, 2\}$, $D = \{d\}$ if we group 0, 1 and 2 together such that 0, 1, 2 become a new value 3, then $C' = \{c\}$,$V_c = \{3\}$, and Table 5.1 becomes Table 5.4.

In both Table 5.1 and Table 5.4, D induces the equivalence relation $IND(D)$, and the set of the equivalence classes is U/D=$\{\{e1, e4, e5\}$,$\{e2, e3, e6, e7\}\}$.

In Table 5.1, C induces the equivalence relation $IND(C)$, and the set of the corresponding equivalence classes is $U/C = \{\{e1, e4\}, \{e2, e5, e7\}, \{e3, e6\}\}$.

In Table 5.4, C' induces the equivalence relation $IND(C')$, and the set of the corresponding equivalence classes is $U/C' = \{\{e1, e2, e3, e4, e5, e5, e6, e7\}\}$.

Let $R_1 = IND(C)$, $R_2 = IND(D)$, $R = IND(C') = U \times U$.

$\Gamma(R_1, R_2) = 1/|U| \sum_{x\in U} |R_2(x) \cap R_1(x)|/|R_1(x)| = 1/7(2/2 + 2/3 + 2/2 + 2/2 + 1/3 + 2/2 + 2/3) = 17/21$.

For each $x \in U$, $R(x) = U$, so we have $R(x) \cap R_2(x) = R_2(x)$, and therefore

$\Gamma(R, R_2) = 1/|U| \sum_{x\in U} |R_2(x) \cap R(x)|/|R(x)| = 1/|U| \sum_{x\in U} |R_2(x)|/|R(x)| = 1/7(3/7 + 4/7 + 4/7 + 3/7 + 3/7 + 4/7 + 4/7) = 25/49$.

The inequality $\Gamma(R_1, R_2) > \Gamma(R, R_2)$ means that it is harder for us to classify objects into D-class by employing the attribute C' than employing the attribute C.

Because $IND(C) = \cap_{c\in C} IND(\{c\})$, when we drop some attributes from C such that a new attribute set C' is formed, we have $IND(C') \supseteq IND(C)$. So by Theorem 26, $\Gamma(C', D) \leq \Gamma(C, D)$. This means that generally, the less the condition attribute set contains attributes, the harder we can classify the objects into D-class by employing the condition attribute set.

	a	b	c	d
e1	Y	Y	3	N
e2	Y	Y	3	Y
e3	Y	Y	3	Y
e4	N	Y	3	N
e5	N	N	3	N
e6	N	Y	3	Y
e7	Y	N	3	Y

Table 5.4: Influenza data (d)

The next theorem shows the extreme cases when R_1 becomes the smallest equivalence relation (the identity relation) or the largest equivalence relation (the universal relation).

Theorem 28 For any given equivalence relation R_2, we have

$$\max_{R_1 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = \Gamma(I_U, R_2) = 1, \quad \min_{R_1 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = \Gamma(U \times U, R_2).$$

Proof: This follows immediately from Theorem 26. ■

In the following theorem, we show the extreme cases when both R_1 and R_2 vary.

Theorem 29

$$\min_{R_1, R_2 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = \frac{1}{|U|}, \quad \max_{R_1, R_2 \in \mathcal{ER}(U)} \Gamma(R_1, R_2) = 1.$$

Proof: By Theorem 22 and Theorem 26, we only need to verify that $\Gamma(U \times U, I_U) = 1/|U|$. Let $R_1 = U \times U, R_2 = I_U$. According to Eq. (5.4), we have

$$\begin{aligned}
\Gamma(U \times U, I_U) &= \frac{1}{|U|} \sum_{x \in U} \frac{|R_2(x) \cap R_1(x)|}{|R_1(x)|} \\
&= \frac{1}{|U|} \sum_{x \in U} \frac{|\{x\} \cap U|}{|U|} \\
&= \frac{1}{|U|} \sum_{x \in U} \frac{1}{|U|} = \frac{1}{|U|}.
\end{aligned}$$

Then the desired conclusion follows. ■

This means that for any two equivalence relations R_1 and R_2, R_2 depends on R_1 to a degree of at least $1/|U|$, and that we can infer some information about R_2 even when R_1 contains no useful information about R_2. This arises from the fact that R_2 contains useful information about itself. However, in the extreme case when R_1 is the universal relation, R_2 is the identity relation (the identity relation contains little information about itself), and the number of objects tends to infinity, the degree that R_2 depends on R_1 tends to zero.

5.2.3 Extension of the Generalized Dependency Degree Γ to Incomplete Information Systems

In this section, we expand the definition of the generalized dependency degree to incomplete information systems by reinterpreting the meaning of the support of a formula and the cardinality of the support in incomplete information systems.

If an information system has some missing values, we call this information system an incomplete information system. For example, there are three missing values in Table 5.5, indicated by "$*$".

	a	b	c	d
e_1	Y	Y	Normal(0)	N
e_2	Y	$*$	High(1)	Y
e_3	Y	Y	$*$	Y
e_4	N	$*$	Normal(0)	N
e_5	N	N	High(1)	N
e_6	N	Y	Very High(2)	Y
e_7	Y	N	High(1)	Y

Table 5.5: Influenza data (e)

To extend the definition of generalized dependency degree to the case of incomplete information systems, we handle missing values by replacing them with their probabilistic distribution at first, then extending the definition of confidence and strength of a rule to incomplete information systems.

How to Handle Missing Values in Incomplete Information Systems

Here we introduce an approximate approach by replacing each missing value by its possible distributions as shown in Table 5.6:

	a	b	c	d
e_1	Y	Y	Normal(0)	N
e_2	Y	$\{P_1/Y, P_2/N\}$	High(1)	Y
e_3	Y	Y	$\{S_1/0, S_2/1, S_3/2\}$	Y
e_4	N	$\{Q_1/Y, Q_2/N\}$	Normal(0)	N
e_5	N	N	High(1)	N
e_6	N	Y	Very High(2)	Y
e_7	Y	N	High(1)	Y

Table 5.6: Influenza data (f)

In the e_2-row, by $\{P_1/Y, P_2/N\}$ we mean that e_2 takes the value Y with a probability P_1, and N with a probability P_2. In the e_4-row, the expression $\{Q_1/Y, Q_2/N\}$ has a similar meaning. In e_3-row, $\{S_1/0, S_2/1, S_3/2\}$ means that e_3 takes the value 0, 1, and 2 with probability S_1, S_2, and S_3 respectively.

In order to reduce the complexity of computing, we introduce an approximate method for determining the values of all the unknown parameters $P_1, P_2, Q_1, Q_2, S_1, S_2, S_3$. We let P_1, P_2, Q_1, Q_2 take the values of the distribution of Y and N in column b, i.e., $P_1 = Q_1 = 3/5, P_2 = Q_2 = 2/5$; and we let S_1, S_2, S_3 take the values of the distribution of 0, 1 and 2 in column c, i.e., $S_1 = 2/6, S_2 = 3/6, S_3 = 1/6$.

Definition of Γ in Incomplete Information Systems

Although we can also define some kinds of equivalence relations induced by the attributes in an incomplete information table, here we introduce a direct way to calculate the generalized dependency degree Γ in an incomplete information table. That is, we choose Eq. (5.3) as our definition of the generalized dependency degree Γ in an incomplete information table. To carry out this idea, we have to define the confidence and the strength of a rule in an incomplete information table. We show our definition using the example of the Influenza Data in Table 5.6.

Before going forward, we need to re-interpret the meaning of $supp(\Phi)$ and the meaning of $|supp(\Phi)|$ where the set $supp(\Phi)$ may be a "fractional" set in an incomplete information table. Here, we interpret $supp(\Phi)$ as a fuzzy set.

If $x \in U$ satisfies Φ with a probability of p, then we consider that the object x belongs to the set $supp(\Phi)$ with a membership of p, and we write the element x in $supp(\Phi)$ as p/x. For example, in Table 5.6, let Φ be the formula $b = Y$. e_1 satisfies the formula $b = Y$ with a probability of 1, the probability of Y in e_1-row, b-column, while e_2 satisfies the formula $b = Y$ with a probability of $P_1 = 3/5$, the probability of Y in e_2-row, b-column. We have

$$supp(\Phi) = \{1/e_1, 0.6/e_2, 1/e_3, 0.6/e_4, 0/e_5, 1/e_6, 0/e_7\}.$$

We can delete all the elements whose probability are equal to zero, i.e., we can write $supp(\Phi)$ as $supp(\Phi) = \{1/e_1, 0.6/e_2, 1/e_3, 0.6/e_4, 1/e_6\}$.

Then we define the fuzzy set $supp(\Phi)$ inductively as follows: If x belongs to $supp(\Phi)$ with a membership of $\mu_{supp(\Phi)}(x) = p$, and x belongs to $supp(\Psi)$ with a membership of $\mu_{supp(\Psi)}(x) = q$, then x belongs to $supp(\Phi \wedge \Psi)$ with a membership of $\mu_{supp(\Phi\wedge\Psi)}(x) = pq$, x belongs to $supp(\sim \Phi)$ with a membership of $\mu_{supp(\sim\Phi)}(x) = 1 - p$, and x belongs to $supp(\Phi \vee \Psi)$ with a membership of $\mu_{supp(\Phi\vee\Phi)}(x) = 1 - (1 - p)(1 - q)$. Formally $supp(\Phi)$ is defined inductively in terms of algebraic operations of a fuzzy set as follows:

$$\begin{aligned}
F1 &: supp(a = v) = \{\mu(x)/x | x \in U, P(a(x) = v) = \mu(x)\} \\
&\quad \text{for } a \in B \text{ and } v \in V_a \\
F2 &: supp(\Phi \vee \Psi) = supp(\Phi) + supp(\Psi) \\
F3 &: supp(\Phi \wedge \Psi) = supp(\Phi) \cdot supp(\Psi) \\
F4 &: supp(\sim \Phi) = \sim supp(\Phi)
\end{aligned}$$

where $supp(\Phi) + supp(\Psi)$ is the algebraic sum of the fuzzy sets $supp(\Phi)$ and $supp(\Psi)$, $supp(\Phi) \cdot supp(\Psi)$ is the algebraic product of the fuzzy sets $supp(\Phi)$, and $supp(\Psi)$, and $\sim supp(\Phi)$ is the complement of the fuzzy sets $supp(\Phi)$ [113].

The cardinality $|supp(\Phi)|$ can be defined in term of the fuzzy set, i.e.,

$$|supp(\Phi)| = \sum_{x \in U} \mu_{supp(\Phi)}(x). \tag{5.11}$$

Next, as an example, we will calculate the generalized dependency degree between $C = \{a, b, c\}$ and $D = \{d\}$ in Table 5.6 by Eq. (5.3). First we need to calculate the confidence and strength of each minimal decision rule using the following definitions:

$$Con(\Phi \to \Psi) = |supp(\Phi \wedge \Psi)|/|supp(\Phi)| \tag{5.12}$$

$$Str(\Phi \to \Psi) = |supp(\Phi \wedge \Psi)|/|U| \tag{5.13}$$

Example 30 We show in the following calculation process the confidence and strength of one minimal rule; the results of all the other minimal rules are listed in Table 5.7. Since $supp(a = Y \wedge b = Y \wedge c = 0 \wedge d = Y) = \{S_1/e_3\}$, $|\{S_1/e_3\}| = S_1 = 2/6$, $supp(a = Y \wedge b = Y \wedge c = 0) = \{1/e_1, S_1/e_3\}$, $|\{1/e_1, S_1/e_3\}| = 1 + S_1 = 1 + 2/6 = 4/3$, we have the minimal rule $a = Y \wedge b = Y \wedge c = 0 \to d = Y$ with confidence=1/4, strength=1/21.

So we have $\Gamma(C, D) = \sum_{r \in MinR(C,D)} Str(r) \cdot Con(r) = 1/21 \cdot 1/4 + 1/7 \cdot 3/4 + 11/70 \cdot 1 + 1/42 \cdot 1 + 1/5 \cdot 1 + 3/35 \cdot 1 + 1/7 \cdot 1 + 2/35 \cdot 1 + 1/7 \cdot 1 = 13/14$.

By the next theorem, we show one more property of the generalized dependency.

Theorem 31 In an incomplete information system, we have $0 \leq \Gamma(C, D) \leq 1$.

a	b	c	d	Con	Str	a	b	c	d	Con	Str
Y	Y	0	Y	1/4	1/21	N	Y	0	Y	0	0
Y	Y	0	N	3/4	1/7	N	Y	0	N	1	3/35
Y	Y	1	Y	1	11/70	N	Y	1	Y	0	0
Y	Y	1	N	0	0	N	Y	1	N	0	0
Y	Y	2	Y	1	1/42	N	Y	2	Y	1	1/7
Y	Y	2	N	0	0	N	Y	2	N	0	0
Y	N	0	Y	0	0	N	N	0	Y	0	0
Y	N	0	N	0	0	N	N	0	N	1	2/35
Y	N	1	Y	1	1/5	N	N	1	Y	0	0
Y	N	1	N	0	0	N	N	1	N	1	1/7
Y	N	2	Y	0	0	N	N	2	Y	0	0
Y	N	2	N	0	0	N	N	2	N	0	0

Table 5.7: Results of all minimal rules

Proof: Because every object contributes $1/|U|$ to $\sum_{r\in MinR(C,D)} Str(r)$ and there are $|U|$ objects in total, we have $\sum_{r\in MinR(C,D)} Str(r) = 1$. It is obvious that $Con(r) \leq 1$ for any rule r, so we have $\sum_{r\in MinR(C,D)} Str(r) \cdot Con(r) \leq \sum_{r\in MinR(C,D)} Str(r) = 1$ ■

This property means that $\Gamma(C, D)$ still serves as an index in an incomplete information system.

Note that our method enables us to handle an information table whose values are probabilistic distributions. Moreover, an information table without missing values can be understood as a special case of an incomplete information table.

5.2.4 Discussion: Comparison with the Conditional Entropy

The generalized dependency degree is in fact a measure for one-way rule, which is different from the eight information measures for one-way rule summarized in [102]. Among these eight information measures, the conditional entropy is a well-known measure, and we will make a comparison between the generalized dependency degree and the conditional entropy in this section.

The generalized dependency degree and the conditional entropy are similar in two different aspects:

1. Both the generalized dependency degree and the conditional entropy measure the degree to which D depends on C.

2. The generalized dependency degree is computed as a type of weighted average of the confidence of decision rules, and conditional entropy averages over the logarithm of the confidence of decision rules. The same weights are employed in both the generalized dependency degree and the conditional entropy.

However, the generalized dependency degree and the conditional entropy are different in three aspects:

1. The value of the conditional entropy is between zero and infinity while the value of the generalized dependency degree is between zero and one, and from this point of view the generalized dependency degree can serve directly as an index.

2. The first form of the conditional entropy is defined in terms of equivalence relations, and so it can be extended to binary relations.

3. To compute the generalized dependency degree by the third form, we only need to carry out simple arithmetic operations, while to compute the conditional entropy, we have to compute the logarithm of the frequency, a time-consuming operation.

The idea for Γ is based on the idea of rough set, we have compared Γ with γ in Eq. (5.1), Eq. (5.2), Eq. (5.4), Eq. (5.7), Eq. (5.5)), and Eq. (5.8). In the next section, on one hand, we will compare Γ with the conditional entropy on their applications in the decision tree classifier, in which the attribute is selected one by one according to its conditional entropy or Γ value on the current node. On the other hand, we will conduct some experiments to make an empirical comparison between Γ and γ on their applications in attribute selection, in which attributes are selected as a subset according to its Γ value or its γ value.

5.2.5 Experiments

In the previous sections, we have given a detailed explanation of the generalized dependency degree by presenting its various forms and developing its various properties. In this section, we will show its significance in decision trees and attribute selection.

There are several reasons to choose C4.5R8 decision tree classifier for our comparison. First and the most important, C4.5R8 uses the conditional entropy that we want to compare with Γ while neural networks do not use the conditional entropy. Second, C4.5R8 can handle continuous attributes and missing values, which makes it easy to compare Γ with the conditional entropy in various cases–handling discrete attribute, handling continuous attributes and handling missing values. Third, compared to other classifiers, a decision tree can be understood easily. Fourth, it often takes large amounts of time to train a neural network, while C4.5R8 decision tree classifier is efficient in training time [63] and thus suitable for large training sets. Lastly, as comparable with neural networks, decision trees already display good classification accuracy [41].

Comparison with the Conditional Entropy in Decision Trees

We have replaced the conditional entropy used in the C4.5 algorithm with the generalized dependency degree such that a new C4.5 algorithm is formed. C4.5R8 employs gain criterion and gain ratio criterion to select the most informative attribute at each subset of training cases. In the case of missing values, the information gain for attribute a is computed by the formula $G(D,\{a\}) =$

$$\begin{cases} P(a) * (H(D) - H(D|\{a\})), & \text{if } a \text{ is a discrete attribute,} \\ P(a) * (H(D) - H(D|\{a\}) - \log_2(N-1)/|U|), & \text{if } a \text{ is continuous,} \end{cases}$$

where $p(a)$ is the probability that a is known.

We replace the information gain in the original C4.5R8 algorithm with

$$G(D, \{a\}) = \Gamma(\{a\}, D) - \Gamma(D)$$

in our new C4.5 algorithm, where $\Gamma(D) = \Gamma(U \times U, IND(D))$. Note that by Theorem 28, we have $G(D, \{a\}) \geq 0$. Similar to the conditional entropy, if a is a continuous attribute, $\Gamma(\{a\}, D)$ is the maximum value of $\Gamma(\{a^t\}, D)$ among all possible tests such as $a \leq t$ for a potential threshold t, and the new attribute a^t is defined as a^t=true if $a \leq t$, and a^t=false, otherwise. In the case of missing values, we use our definition of $\Gamma(C, D)$ in incomplete information systems introduced in Section 5.2.3.

One further change we make from the original C4.5R8 is that we stop the procedure of building the tree earlier by applying a new criteria: in the current node, if for every attribute, the number of the gain cases is less than a given value 0.75, then the splitting procedure stops. The number of the gain cases is calculated by multiplying the number of cases in the current node by the dependency gain.

Both the original C4.5R8 and the new C4.5 are applied to all of the same twenty datasets [1] from the UCI machine learning repository as Quinlan uses in [83]. Table 5.8 is a description of the datasets we use. The first column shows the names of the datasets, the second column gives the numbers of cases in each dataset, the third column gives the number of classes, the fourth column gives the number of continuous attributes, the fifth column gives the number of discrete attributes, and the final column describes whether there are missing values in each dataset.

The experiments are conducted on a workstation whose hardware model is Nix Dual Intel Xeon 2.2GHz, with 1GB of RAM, and whose OS is Linux Kernel 2.4.18-27smp (RedHat7.3).

[1] Note that the datasets we use may have slight differences from those Quinlan uses (Quinlan, private correspondence). For example, the glass dataset we use has a different order of the cases from Quinlan's.

Dataset	Cases	Classes	Cont	Discr	Missing
Anneal	898	6	6	32	Y
Auto	205	6	15	10	Y
Breast-w	699	2	9	0	Y
Colic	368	2	7	15	Y
Credit-a	690	2	6	9	Y
Credit-g	1000	2	7	13	N
Diabetes	768	2	8	0	N
Glass	214	6	9	0	N
Heart-c	303	2	6	7	Y
Heart-h	294	2	8	5	Y
Hepatitis	155	2	6	13	Y
Allhyper	3772	5	7	22	Y
Iris	150	3	4	0	N
Labor	57	2	8	8	Y
Letter	20000	26	16	0	N
Segment	2310	7	19	0	N
Sick	3772	2	7	22	Y
Sonar	208	2	60	0	N
Vehicle	846	4	18	0	N
Wave	300	3	21	0	N

Table 5.8: Description of the datasets

Both algorithms use ten-fold cross-validations with each task. The figures shown in Table 5.9 are the mean error rate of the ten-fold cross-validations of both the original C4.5R8 and the new C4.5.

The second and fourth columns in Table 5.9 are the mean error rates (error rate = 100% - classification rate) before and after pruning respectively obtained by running with the option that uses the gain criteria (not the gain ratio) and the grouping method in the original C4.5R8 system. The third and fifth columns are the results before and after pruning respectively obtained by running with the same option in the new C4.5 system. This means that when the new gain criteria based on the generalized dependency degree is used, the grouping method is also used. In each row of the second and third columns, the smaller result is shown in bold, and so are the fourth and fifth columns. The final row shows the sum of results of the experiments on the twenty datasets.

The figures shown in Table 5.10 describe the average run time of the ten-fold cross-validations. The time unit in Table 5.10 is 0.01 second. The second

	Unpruned		Pruned	
Dataset	O (%)	N (%)	O (%)	N (%)
Anneal	**3.9**	6.1	**4.6**	7.9
Auto	**20.5**	22.0	**22.0**	22.5
Breast-w	5.7	**4.2**	**4.3**	4.5
Colic	19.8	**16.3**	16.0	**15.4**
Credit-a	19.7	**15.2**	17.1	**15.6**
Credit-g	30.5	**27.2**	28.0	**27.0**
Diabetes	**24.7**	26.0	**24.5**	25.6
Glass	**31.2**	31.7	**30.3**	**30.3**
Heart-c	**22.4**	23.4	**21.4**	23.1
Heart-h	24.2	**20.7**	22.8	**21.1**
Hepatitis	20.0	**19.3**	19.9	**19.3**
Allhyper	1.4	**1.1**	1.4	**1.2**
Iris	6.0	**4.0**	6.0	**4.0**
Labor	24.7	**15.7**	26.3	**19.3**
Letter	**11.9**	12.5	**11.9**	12.4
Segment	**3.2**	3.5	**3.2**	3.7
Sick	1.2	**1.0**	1.1	**1.0**
Sonar	**20.7**	27.9	**20.7**	27.9
Vehicle	**27.8**	30.1	**28.0**	30.4
Wave	28.4	**26.0**	28.4	**26.3**
Average	17.40	**16.70**	**16.90**	16.93

Table 5.9: Mean error rates of the original C4.5 and the new C4.5

column and the fifth column in Table 5.10 show the average run time of the original procedure C4.5R8 before pruning and after pruning in the ten-fold cross-validations. The third and the fourth columns show the average run time of the new C4.5 without the pruning procedure and the reduced time rate relative to the second column, while the sixth and seventh columns give the corresponding results of the new C4.5 with the pruning procedure. Note that the run time does not include the run time for data preparation for the cross-validation, or the run time for final result reporting, in both C4.5 systems.

The figures shown in Table 5.11 describe the average number of leaves of the decision trees of the ten-fold cross-validations. The second and the fourth columns in Table 5.11 show the results of the original procedure C4.5R8 before pruning and after pruning. The third and the fifth columns show the results of the new C4.5 before pruning and after pruning. In each row of the second and third columns, the smaller result is shown in bold, and so do the fourth and fifth columns. The experiments show that the generalized dependency

	Unpruned			Pruned		
Dataset	O	N	Reduced	O	N	Reduced
Anneal	6.2	4.6	25.8	6.8	5.1	25.0
Auto	9.6	2.6	72.9	9.7	2.6	73.2
Breast-w	1.5	1.0	33.3	1.6	1.0	37.5
Colic	3.9	1.5	61.5	4.1	1.5	63.4
Credit-a	7	2.4	65.7	8.3	2.5	69.9
Credit-g	9.5	4.7	50.5	11.5	5.2	54.8
Diabetes	4.2	2.4	42.9	4.6	2.6	43.5
Glass	1.4	0.9	35.7	2.3	1.5	34.8
Heart-c	1.7	0.7	58.8	2.0	0.9	55.0
Heart-h	1.6	0.7	56.3	1.9	0.7	63.2
Hepatitis	0.8	0.6	25	0.9	0.7	22.2
Allhyper	40	18.5	53.8	45.0	18.5	58.9
Iris	0.25	0.2	20.0	0.5	0.4	20.0
Labor	0.2	0.1	50.0	0.4	0.4	0.0
Letter	7.4	5.5	25.1	8.15	5.9	27.1
Segment	41.1	24.8	39.7	46.7	25.5	45.4
Sick	35.6	17.1	52	38.1	20.8	45.4
Sonar	11.2	5.0	55.4	12.9	5.1	60.5
Vehicle	8.4	5.8	31	10.8	6.3	41.7
Wave	5.7	2.0	64.9	6.8	2.1	69.1
Average	205.57	190.64	46.02	169.27	170.60	45.53

Table 5.10: Average run time of the original C4.5R8 and the new C4.5

degree $\Gamma(C, D)$ is a useful measure. We compare three aspects of the new C4.5 algorithm using the generalized dependency degree with the original C4.5R8 algorithm using the conditional entropy:

1. Speed: To compute $\Gamma(C, D)$, we only need to carry out arithmetic operations, while the computation of the commonly used conditional entropy needs to compute the logarithm of the frequency, a time-consuming operation. Furthermore, the building tree procedure in the new C4.5 algorithm stops earlier in most cases. This explains why the new C4.5 procedure with (or without) the pruning procedure runs much faster than the original C4.5R8 procedure with (or without) the pruning procedure. In fact, the new C4.5 procedure runs in about half of the time required by the original C4.5R8 procedure, and the new C4.5 procedure without pruning procedure can run a little faster still.

2. Prediction accuracy: Before pruning, the new C4.5 outperforms the original C4.5R8 in prediction accuracy (the new C4.5 wins 11 cases while C4.5R8 wins 9 cases). After pruning, the new C4.5 is comparable with the original C4.5R8 (the new C4.5 wins 10 cases while C4.5R8 wins 9 cases). The

	Unpruned		Pruned	
Dataset	O (%)	N (%)	O (%)	N (%)
Anneal	**139.8**	144	93.4	**83.0**
Auto	**55.1**	58.1	**45.6**	47.9
Breast-w	41.2	**17.4**	22.2	**15.8**
Colic	80.5	**30.5**	**15.8**	19.1
Credit-a	137.4	**56.2**	59.7	**51.5**
Credit-g	333.6	**151.1**	190.4	**139.4**
Diabetes	**49.4**	90.2	**43.4**	80.8
Glass	**49.0**	55.8	**46.2**	48
Heart-c	69.6	**33.0**	36.0	**26.5**
Heart-h	78.2	**25.8**	**15.7**	19.0
Hepatitis	29.4	**16.8**	**13.8**	15.6
Allhyper	63.7	**46.8**	34.0	**28.2**
Iris	**8.6**	8.8	**8.0**	8.4
Labor	14.1	**7.8**	7.8	**5.3**
Letter	**2581.8**	2694	**2412.4**	2458.4
Segment	**86.4**	97.2	**81.8**	94.8
Sick	66.1	**37.0**	48.8	**37.0**
Sonar	27.2	**25.0**	27.2	**25.0**
Vehicle	**151.0**	171.0	**134.8**	163.2
Wave	49.2	**46.2**	48.4	**45.0**
Average	205.57	**190.64**	**169.27**	170.60

Table 5.11: Average number of leaves of the original C4.5R8 and the new C4.5

new C4.5 algorithm seems more successful in the dataset labor, in which the algorithm achieves a 15.7% prediction error rate, while the original algorithm has a 24.7% error rate.

The original C4.5R8 algorithm performs best using the pruning procedure, while the new C4.5 algorithm performs best without using the pruning procedure.

The difference between the average 16.90% of the results of experiments on 20 datasets in the third column and the average 16.93% in the fourth column is 0.03%. This means that when we compare their best, the new C4.5 algorithm is comparable with the original C4.5R8 algorithm in prediction accuracy. Note that the prediction accuracy of the original C4.5R8 algorithm is not statistically significantly different from POL, whose prediction accuracy is the best among a group of thirty-three classification algorithms [63].

3. Size of tree: Before pruning, in 12 datasets, there are less leaves in trees created by the new C4.5 than those created by the original C4.5R8. After pruning, in 10 datasets, these are less leaves in trees created by the new

C4.5 than those created by the original C4.5R8. This means the new C4.5 is better than the original C4.5R8 in size of tree when we do not use the pruning procedure, while it is comparable with the original C4.5R8 algorithm in terms of the tree size when we use the pruning procedure.

Comparison with γ In Attribute Selection

We compare Γ with γ in attribute section on the zoo dataset, which is obtained from the UCI machine learning repository. The zoo dataset has 101 cases, 16 conditional attributes, and one decision attributes. All the attributes are discrete.

The comparison strategy is described as follows. Let D be the set of the decision attribute, for a given number k, we select set C of conditional attributes such $\Gamma(C, D)$ ($\gamma(C, D)$) is maximal among all possible subsets with k conditional attributes. Then we apply the selected subset by $\Gamma(C, D)$ and that by $\gamma(C, D)$ to the C4.5R8 algorithm. The better the subset of conditional attributes is selected, the better the accuracy should be.

The results are shown in the Table 5.12. The first column is the number of selected attributes. The second column and the sixth column are the set of conditional attribute selected by γ and Γ respectively. The third column and the seventh column are the γ value and the Γ value respectively corresponding the selected attributes.

The fourth column and the eighth column are the mean error rates of the ten-fold cross-validations on the selected attributes by γ and Γ respectively, and the results are obtained by the C4.5R8 with default option; the fifth column and the ninth column are obtained by the C4.5R8 with the option that both the gain criteria and the grouping method are used.

Note that when the number of selected attributes is greater than five, there will be no difference for attribute selection between Γ and γ, and we omit the cases when $5 < k < 16$. This can be explained by Theorem 21 and

Theorem 26. By Theorem 26, when $C \subseteq C'$, $\Gamma(C,D) \leq \Gamma(C',D)$ because of $IND(C) \supseteq IND(C')$, so the maximum $\Gamma(C,D)$ is equal to one when C ranges over all the subsets with k ($k > 5$) conditional attributes since the maximum $\Gamma(C,D)$ is equal to one when C ranges over all the subsets with $k = 5$ conditional attributes; by Theorem 21, both γ and Γ is equal to one if one of them is equal to one, therefore, the maximal value of Γ and the maximal value of γ is equal when $k > 5$. In the seventh row, T denotes the whole conditional attributes, i.e., $T = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$.

The experimental results show that both Γ and γ are efficient in attribute selection on dataset zoo. By Γ and γ, we can find the same best attribute set $\{4,6,8,12,13\}$, on which the C4.5R8 performs better in accuracy than employing all the conditional attributes. However, when the number of selected attributes is less than five, the C4.5R8 performs better in accuracy on the attributes selected by Γ than on those selected by γ.

k	C_1 by γ	$\gamma(C_1,D)$	**O**	**O-g-s**	C_2 by Γ	$\Gamma(C_2,D)$	**O**	**O-g-s**
1	{4}	0.41	39.5	39.5	{13}	0.60	**26.4**	**26.4**
2	{1,13}	0.65	15.7	14.7	{4,13}	0.83	**12.7**	**12.7**
3	{3,12,13}	0.76	13.7	11.9	{4,6,13}	0.92	**12.7**	**10.8**
4	{3,10,13,14}	0.96	17.7	10.9	{4,6,8,13}	0.98	**9.8**	**7.9**
5	{4,6,8,12,13}	1.0	5.9	5.9	{4,6,8,12,13}	1.0	5.9	5.9
16	T	1.0	6.6	6.9	T	1.0	6.6	6.9

Table 5.12: Attribute selection by γ and Γ on the dataset 'zoo'

5.2.6 Summary

We give three different forms of the generalized dependency degree in terms of equivalence relations, minimal rule, and arithmetic operation, respectively.

The generalized dependency degree Γ has some properties, such as the Partial Order Preserving Property and the Anti-Partial Order Preserving Property. Besides, its value is between zero and one. Therefore, it can serve as an index to measure how much decision attributes depend on conditional attributes. The experimental study shows that the generalized dependency degree is an informative measure in decision trees and attribute selection.

5.3 A Novel Random Graph Dependency Measure $H(RG_2|RG_1)$

In the C4.5 decision tree algorithm, the conditional entropy is employed to determine the best attribute according to the information gain criteria or the gain ratio criteria. From the random graph perspective, the traditional method actually models the data points as a complete graph with a unit similarity of all ones; as a result, the distance information among these points is lost. To model the data points more accurately, we propose a random graph dependency measure $H(RG_2|RG_1)$ based on distance information between two random graphs RG_1 and RG_2, which degrades to the traditional measure when RG_1 and RG_2 degrade to equivalence relations.

The rest of this section is organized as follows. In Section 5.3.1, we give the definition of the random graph dependency measure, and investigate its basic properties. In Section 5.3.2, we discuss continuous attributes and the random graph generation method, and provide the algorithm to find the best cut and the corresponding information gain. In Section 5.3.3, we conduct experiments to support the random graph dependency measure. In Section 5.3.4, we draw a conclusion and suggest future work on the random graph dependency measure and its application.

5.3.1 Random Graph Dependency Measure

The definition of the random graph has been given in Eq. (1). In this section, we will first show some examples, give some other definitions related to the random graph dependency measure, and investigate on its basic properties.

Example 32 In Table 5.2, $U = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, and there are four random graphs on U: one (P^1) is generated before any feature is employed, two (P^2, P^3) of them are generated by two attributes x_1 and x_2, and one (P^4) is generated by the class y feature.

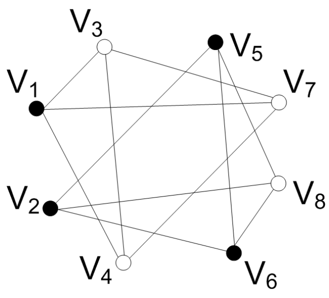
Before any feature is employed, we assume that there is no difference between any pair of points, and thus the we define $p^1_{ij} = 1$ for any pair of v_i and v_j. The corresponding random graph is

$$P^1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

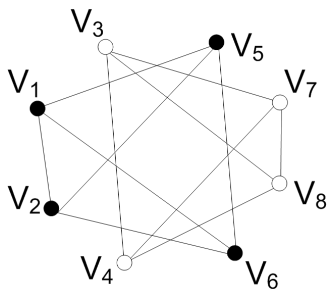
which is called the universal relation on U, and is denoted by $U \times U$. This random graph is shown in Figure 5.4 (a).

For x_1, we can generate a random graph P^2 shown in Eq. (5.22), which will be discussed in Section 5.3.2.

For x_2, if we define $p_{ij} = 1$ if v_i and v_j have same values of x_1, and $p_{ij} = 0$



(a)
P^3 by attribute x_2



(b)
P^4 by class attribute y

Figure 5.5: Two equivalence relations generated by x_2 and y respectively, which can be understood as special random graphs.

otherwise. Then we have P^3. For y, similarly we have P^4.

$$P^3 = \begin{pmatrix} 1&0&1&1&0&0&1&0 \\ 0&1&0&0&1&1&0&1 \\ 1&0&1&1&0&0&1&0 \\ 1&0&1&1&0&0&1&0 \\ 0&1&0&0&1&1&0&1 \\ 0&1&0&0&1&1&0&1 \\ 1&0&1&1&0&0&1&0 \\ 0&1&0&0&1&1&0&1 \end{pmatrix}, P^4 = \begin{pmatrix} 1&1&0&0&1&1&0&0 \\ 1&1&0&0&1&1&0&0 \\ 0&0&1&1&0&0&1&1 \\ 0&0&1&1&0&0&1&1 \\ 1&1&0&0&1&1&0&0 \\ 1&1&0&0&1&1&0&0 \\ 0&0&1&1&0&0&1&1 \\ 0&0&1&1&0&0&1&1 \end{pmatrix}.$$

Random graphs P^3 and P^4 are shown in Figure 5.5 (a) and (b).

We also need the concept of the random neighbor set $RG(v_i)$ of a node v_i and that of the intersection $RG_1(v_i) \cap RG_2(v_i)$ of two random neighbor sets $RG_1(v_i)$ and $RG_2(v_i)$. The random neighbor set $RG(v_i)$ of v_i is defined as

$$RG(v_i) = \{(v_k, p_{ik}) | v_k \in U\},$$

where p_{ik} is the probability of v_k being a neighbor of v_i. In the setting of a random graph, each node v_k is linked to node v_i with a probability p_{ik}, so v_k is the neighbor of v_i with a probability p_{ik}. If all the edges in the graph exist in a deterministic way, then the value of p_{ij} is equal to zero or one; in such a case,

$RG(v_i)$ is the set of the all neighbors of node v_i. We denote the cardinality of $RG(v_i)$ by

$$|RG(v_i)| = \sum_k p_{ik}.$$

Example 33 For random graphs P^1, P^2, P^3 and P^4, we have
$P^1(v_1) = \{(v_1, 1), (v_2, 1), (v_3, 1), (v_4, 1), (v_5, 1), (v_6, 1), (v_7, 1), (v_8, 1)\}$,
$|P^1(v_1)| = 8$,
$P^2(v_1) = \{(v_1, 1), (v_2, 0.65), (v_3, 0.29), (v_4, 0.28), (v_5, 0.18), (v_6, 0.17), (v_7, 0.08), (v_8, 0.05)\}$,
$|P^2(v_1)| = 1 + 0.65 + 0.29 + 0.28 + 0.18 + 0.17 + 0.08 + 0.05 = 2.7$,
$P^3(v_1) = \{(v_1, 1), (v_2, 0), (v_3, 1), (v_4, 1), (v_5, 0), (v_6, 0), (v_7, 1), (v_8, 0)\}$,
$|P^3(v_1)| = 4$,
$P^4(v_1) = \{(v_1, 1), (v_2, 1), (v_3, 0), (v_4, 0), (v_5, 1), (v_6, 1), (v_7, 0), (v_8, 0)\}$,
$|P^4(v_1)| = 4$.

The intersection of two neighbor sets $RG_1(v_i)$ and $RG_2(v_i)$ is defined as

$$RG_1(v_i) \cap RG_2(v_i) = \{(v_k, a_{ik}b_{ik}) | v_k \in U\},$$

where $RG_1(v_i) = \{(v_k, a_{ik}) | v_k \in U\}$ is induced by a random graph $RG_1 = (a_{ij})$, and $RG_2(v_i) = \{(v_k, b_{ik}) | v_k \in U\}$ is induced by another random graph $RG_2 = (b_{ij})$. As a result, we have

$$|RG_1(v_i) \cap RG_2(v_i)| = \sum_k a_{ik}b_{ik}.$$

Hidden in the definition, there is a local independent assumption: probability a_{ik} is independent to b_{ik}. With such a local independent assumption, we deduce that v_k is the common neighbor of v_i in both RG_1 and RG_2 with a probability $a_{ik}b_{ik}$ since v_k is the neighbor of v_i in RG_1 with a probability a_{ik}, and v_k is the neighbor of v_i in RG_2 with a probability b_{ik}.

Example 34 For random graphs P^2 and P^4, we have $P^2(v_1) \cap P^4(v_1) = \{(v_1, 1),(v_2, 0.65), (v_3, 0), (v_4, 0), (v_5, 0.18), (v_6, 0.17), (v_7, 0), (v_8, 0)\}$, and thus $|P^2(v_1) \cap P^4(v_1)| = 1 + 0.65 + 0.18 + 0.17 = 2$.

We are ready to define the random graph dependency measure $H(RG_2|RG_1)$.

Definition 35 Let $U = \{v_1, v_2, \ldots, v_n\}$. $RG_1 = (a_{ij})$ and $RG_2 = (b_{ij})$ are two random graphs defined on U. The random graph dependency measure $H(RG_2|RG_1)$ is defined as

$$\begin{aligned} H(RG_2|RG_1) &= \tfrac{1}{|U|} \sum_{x \in U} \log_2 \tfrac{|RG_2(x) \cap RG_1(x)|}{|RG_1(x)|} \\ &= \tfrac{1}{n} \sum_{i=1}^{n} \log_2 \tfrac{|RG_2(v_i) \cap RG_1(v_i)|}{|RG_1(v_i)|} \\ &= \tfrac{1}{n} \sum_{i=1}^{n} \log_2 \tfrac{\sum_k a_{ik} b_{ik}}{\sum_k a_{ik}}, \end{aligned} \qquad (5.14)$$

where $RG_2(x)$ and $RG_1(x)$ denote the random neighbors of x in RG_2 and RG_1 respectively.

In Eq. (5.14), $\frac{|RG_2(v_i) \cap RG_1(v_i)|}{|RG_1(v_i)|} = 1$, if and only if $RG_2(v_i) \cap RG_1(v_i) = RG_1(v_i)$, if and only if $\sum_k a_{ik} b_{ik} = \sum_k a_{ik}$, and if and only if $b_{ik} = 1$ if $a_{ik} \neq 0$. This means that we can predict exactly the neighbor v_k of v_i in RG_2 from the nonzero neighbors of v_i in RG_1. On the other hand, $\sum_k a_{ik} b_{ik} = 0$ if and only $b_{ik} = 0$ if $a_{ik} \neq 0$. This means that we cannot predict any thing about the nonzero neighbors of v_i in RG_2 from the nonzero neighbors of v_i in RG_1. In most cases, $0 < \frac{|RG_2(v_i) \cap RG_1(v_i)|}{|RG_1(v_i)|} < 1$; this term can be intuitively understood as the extent to which we can predict the neighbors of v_i in RG_2 from the neighbors of v_i in RG_1. Based on such an intuition, we average $\frac{|RG_2(v_i) \cap RG_1(v_i)|}{|RG_1(v_i)|}$ to obtain the dependency measure.

The traditional method of measuring the "information content" of the data in the columns of an attribute set is the conditional entropy [37]. The conditional entropy is well discussed in the literature of Information Theory [25, 105], and is used in the C4.5 decision tree algorithm [82]. Its definition is as follows.

Definition 36 The formulation for the conditional entropy is as follows:

$$\begin{aligned} H(D|C) &= -\sum_c \sum_d \Pr(c) \cdot \Pr(d|c) \cdot \log_2(\Pr(d|c)) \\ &= -\sum_c \Pr(c) \cdot \sum_d \Pr(d|c) \cdot \log_2(\Pr(d|c)), \end{aligned} \qquad (5.15)$$

where C and D are two subsets of A containing only discrete attributes, and c and d denote the vectors consisting of the values of attributes in C and in D respectively.

No literature has yet measured the "information content" between two random graphs. Since the random graph dependency measure is being proposed here for the first time, some preliminary investigations of its properties may be helpful for its being understood, its being further investigated by others, and its finding further applications in other fields besides the decision trees. For readers who are interested only in decision trees, please skip the next section.

Properties of the Generalized Dependency Degree

Although we focus on decision tree improvement in this section, the random graph dependency has potential applications other than decision trees. For the sake of its future potential usages and for theoretical completeness, we investigate its preliminary properties in this section. We first show that when the underlying attribute is discrete, $H(RG_2|RG_1)$ becomes the conditional entropy. For this purpose, we consider the equivalence relations induced by an information system. We also show some other basic properties, which have counterparts in conditional entropy.

Recall that the condition entropy is defined in Eq. (1.3) as

$$\begin{aligned} H(D|C) &= -\sum_c \sum_d \Pr(c) \cdot \Pr(d|c) \cdot \log_2(\Pr(d|c)) \\ &= -\sum_c \Pr(c) \cdot \sum_d \Pr(d|c) \cdot \log_2(\Pr(d|c)), \end{aligned}$$

where c and d denote the vectors consisting of the values of attributes in C and in D respectively. The formulation of the entropy is

$$H(D) = -\sum_d \Pr(d) \cdot \log_2(\Pr(d)).$$

Recall that, in an information system $S = (U, A, V, f)$. Let P be a subset of A, the P-indiscernibility relation $\mathrm{IND}(P)$ is defined as

$$\mathrm{IND}(P) = \{(x, y) \in U \times U \,|\, (\forall a \in P)\ a(x) = a(y)\}.$$

The equivalence relation $\mathrm{IND}(P)$ partitions the whole dataset into a disjoint union of some equivalence classes, which can be considered as a special random graph on U. More specifically, in such a random graph, there is an edge with a probability of one between each pair of points in the same equivalence class, and there is no edge (or an edge with a probability of zero) between any pair of points that belong to two different classes. Based on such an interpretation, the concepts of random neighbor set and its cardinality can be applied to equivalence relations, and are rewritten as follows. For simplicity, denote $\mathrm{IND}(P)(x) = P(x)$. According to the definition of $\mathrm{IND}(P)$, we have

$$P(x) = \{(y, p_{yx}) | p_{yx} = 1 \text{ if } (\forall a \in P)\, a(x) = a(y) \text{ and } 0 \text{ otherwise}\}. \tag{5.16}$$

Since $p_{yx} = 1$ or $p_{yx} = 0$ in an equivalence relation, we can write $P(x)$ without ambiguity as follows.

$$P(x) = \{y | (\forall a \in P)\, a(x) = a(y)\}.$$

In the following, we show an example about $\mathrm{IND}(P)$ and $P(x)$ to aid familiarity with these concepts.

Example 37 Table 5.2 is an information system, $U = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$,

$A = \{x_1, x_2, y\}$. Let $P = \{x_2\}$, then

$$\begin{aligned} \mathrm{IND}(P) = \quad & \{(v_1,v_1),(v_1,v_3),(v_1,v_4),(v_1,v_7),\\ & (v_3,v_1),(v_3,v_3),(v_3,v_4),(v_3,v_7),\\ & (v_4,v_1),(v_4,v_3),(v_4,v_4),(v_4,v_7),\\ & (v_7,v_1),(v_7,v_3),(v_7,v_4),(v_7,v_7),\\ & (v_2,v_2),(v_2,v_5),(v_2,v_6),(v_2,v_8),\\ & (v_5,v_2),(v_5,v_5),(v_5,v_6),(v_5,v_8),\\ & (v_6,v_2),(v_6,v_5),(v_6,v_6),(v_6,v_8),\\ & (v_8,v_2),(v_8,v_5),(v_8,v_6),(v_8,v_8)\}. \end{aligned}$$

$\mathrm{IND}(P)(v_1) = P(v_1) = \{v_1, v_3, v_4, v_7\}$.

Let C and D be two subsets of A. When the two equivalence relations $\mathrm{IND}(C)$ and $\mathrm{IND}(D)$ are applied to the random graph dependency measure $H(RG_2|RG_1)$, we obtain the following theorem, which shows that $H(RG_2|RG_1)$ is a generalization of the conditional entropy.

Theorem 38 (Generalization of the Conditional Entropy) Let C and D be two subsets of A. Then $H(\mathrm{IND}(D)|\mathrm{IND}(C)) = -H(D|C)$.

Proof: Note that

$$D(x) = \{y|\,(\forall a \in D)\, a(x) = a(y)\},$$

$$C(x) = \{y|\,(\forall a \in C)\, a(x) = a(y)\}.$$

$C \cup D$ is also a subset of A, and the equivalence relation $\mathrm{IND}(C \cup D)$ partitions U into a disjoint union of some equivalence classes called $(C \cup D)$-classes. The set of all the $(C \cup D)$-classes is denoted by $U/(C \cup D)$. Let X be a $(C \cup D)$-class. Then for any $y, x \in X$, y has the same values as x at the attributes in $(C \cup D)$, and so y has the same values as x at the attributes in both C and D, i.e., y and x are both in the same C-class and in the same D-class, i.e., $C(y) = C(x)$,

$D(y) = D(x)$, and therefore for any $x \in X$ we can denote $C(x)$ by $C(X)$, $D(x)$ by $D(X)$, and $|D(x) \cap C(x)|/|C(x)|$ by $|D(X) \cap C(X)|/|C(X)|$. Given the fact that $|X| = |D(X) \cap C(X)|$ and the definition of the random graph dependency, we have

$$\begin{aligned}
& H(\mathrm{IND}(D)|\mathrm{IND}(C)) \\
&= \tfrac{1}{|U|} \textstyle\sum_{x \in U} \log_2 \tfrac{|D(x) \cap C(x)|}{|C(x)|} \\
&= \tfrac{1}{|U|} \textstyle\sum_{X \in U/(C \cup D)} \sum_{x \in X} \log_2 \tfrac{|D(x) \cap C(x)|}{|C(x)|} \\
&= \tfrac{1}{|U|} \textstyle\sum_{X \in U/(C \cup D)} \sum_{x \in X} \log_2 \tfrac{|D(X) \cap C(X)|}{|C(X)|} \\
&= \tfrac{1}{|U|} \textstyle\sum_{X \in U/(C \cup D)} |X| \log_2 \tfrac{|D(X) \cap C(X)|}{|C(X)|} \\
&= \textstyle\sum_{c,d} \tfrac{|X|}{|U|} \log_2 \tfrac{|D(X) \cap C(X)|}{|C(X)|} \\
&= \textstyle\sum_c \sum_d Pr(c,d) \log_2(Pr(d|c)) \\
&= \textstyle\sum_c Pr(c) \sum_d Pr(d|c) \log_2(Pr(d|c)) \\
&= -H(D|C).
\end{aligned} \tag{5.17}$$

where c, d denote the vectors consisting of the values of attributes in C and in D for any element in X respectively, and $\frac{|X|}{|U|}$ is frequency of elements of X in U, i.e., the frequency of c, d. ■

Remark. The random graph dependency is also a generalization of the entropy when $RG_1 = U \times U$ and $RG_2 = IND(D)$. This is formally written as $H(D) = -H(\mathrm{IND}(D)|U \times U)$.

Theorem 39 (Bounds of the Random Graph Dependency Measure) $-\log_2 |U| \leq H(RG_2|RG_1) \leq 0$.

Proof: Since both $RG_2(x)$ and $RG_1(x)$ contain at least one element x, and $RG_1(x)$ contains at most $|U|$ elements,

$$\frac{|RG_2(x) \cap RG_1(x)|}{|RG_1(x)|} \geq \frac{1}{|U|},$$

and so

$$\begin{aligned} H(RG_2|RG_1) &= \tfrac{1}{|U|}\textstyle\sum_{x\in U}\log_2 \frac{|RG_2(x)\cap RG_1(x)|}{|RG_1(x)|} \\ &\geq \tfrac{1}{|U|}\textstyle\sum_{x\in U}\log_2 \frac{1}{|U|} \\ &= -\log_2 |U|. \end{aligned} \tag{5.18}$$

The desired conclusion follows.

■

Remark. It is possible to design a new building tree stop criterion by employing this property, for example, $H(RG_2|RG_1) < -c\log_2 |U|$, although this is not being considered in the empirical part of this section.

In the following theorems, we discuss the relation among these random graphs $RG_1 = (a_{ij})$, $RG_2 = (b_{ij})$, and $RG_3 = (c_{ij})$. We first reveal how $H(RG_2|RG_1)$ changes when RG_2 is changed to be a larger RG_3, meaning that the probability of an edge in RG_2 is less than or equal to the probability of the corresponding edge in RG_3. Let $U = \{v_i|\, i = 1, 2, \ldots, n\}$.

Theorem 40 (Partial Order Preserving Property) For any random graphs RG_1, RG_2 and RG_3 on U, if RG_3 is larger than RG_2, then
$H(RG_2|RG_1) \leq H(RG_3|RG_1)$.

Proof:

$$\begin{aligned} H(RG_2|RG_1) &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\log_2 \frac{\sum_k b_{ik}a_{ik}}{\sum_k a_{ik}} \\ &\leq \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\log_2 \frac{\sum_k c_{ik}a_{ik}}{\sum_k a_{ik}} \\ &= H(RG_3|RG_1). \end{aligned} \tag{5.19}$$

The desired conclusion follows. ■

This means that the weaker relations in RG_2 are, the less RG_2 depends on RG_1. Thus in Theorem 40, we have shown the partial order preserving property of $H(RG_2|RG_1)$ on the item RG_2. In the next theorem, we continue to show the anti-partial order preserving property of $H(RG_2|RG_1)$ on the item RG_1. Next we discuss the situation when RG_1 varies.

Theorem 41 (Anti-Partial Order Preserving Property) For any random graphs RG_1, RG_2 and RG_3 on U, if RG_3 is larger than RG_2 on the edge (v_i, v_j) where b_{ij} is minimal in the set $\{b_{ik}|\, k = 1, 2, 3, \ldots, n\}$, then $H(RG_2|RG_1) \geq H(RG_2|RG_3)$.

Proof: $H(RG_2|RG_1) = \frac{1}{n}\sum_{i=1}^{n}\log_2 \frac{\sum_k b_{ik}a_{ik}}{\sum_k a_{ik}}$. The partial derivative of $H(RG_2|RG_1)$ with respect to a_{ij} can be calculated as

$$\begin{aligned}\frac{\partial H(RG_2|RG_1)}{\partial a_{ij}} &= \frac{\log_2 e}{n}\sum_{i=1}^{n}\left(\frac{b_{ij}}{\sum_k b_{ik}a_{ik}} - \frac{1}{\sum_k a_{ik}}\right)\\ &= \frac{\log_2 e}{n}\sum_{i=1}^{n}\left(\frac{1}{\sum_k b_{ik}a_{ik}/b_{ij}} - \frac{1}{\sum_k a_{ik}}\right),\end{aligned}$$

which is less than or equal to zero if b_{ij} is minimal in the set $\{b_{ik}|\, k = 1, 2, 3, \ldots, n\}$. The desired conclusion follows. ■

It is easy to get the opposite conclusion if b_{ij} is maximal in the set $\{b_{ik}|\, k = 1, 2, 3, \ldots, n\}$. This property shows that, on a weak edge such as the minimal b_{ij} in RG_2, an increasing probability on the corresponding edge in the conditional random graph RG_1 will decrease the dependency, while on an edge with a maximal b_{ij} an increase of a_{ij} will result in a larger dependency.

5.3.2 Discussion on Continuous Attributes

As we have shown in the previous section, the random graph dependency measure is the same as the conditional entropy on discrete attributes except for the sign. In this section, we will focus on the continuous attribute.

We showed in Section 5.1 that before the cut all points are treated equally and after the cut the points less (greater) than the threshold produced by the cut are treated equally. From the perspective of random graphs, the calculation of information gain by the traditional approach undoubtedly ignores some distance information. To explain this statement, we investigate the measure when RG_2 is an equivalence relation induced by the class attribute $D = \{y\}$ and RG_1 is a complete graph, in which each pair of nodes has an edge with a probability of one. For such a special random graph setting, we have

$$\begin{aligned} H(RG_2|RG_1) &= \tfrac{1}{n}\sum_{i=1}^{n}\log_2 \frac{\sum_k b_{ik}a_{ik}}{\sum_k a_{ik}} \\ &= \tfrac{1}{n}\sum_{i=1}^{n}\log_2 \frac{\sum_k b_{ik}}{n} \\ &= -H(D). \end{aligned}$$

If a is a continuous attribute, t is a potential threshold, and the new attribute a^t is defined as a^t=true if $a \leq t$, and a^t=false otherwise. Let $LU = \{x_i|x_i < t\}$, $RU = \{x_i|x_i > t\}$ The traditional information gain $H(D) - H(D|\{a^t\})$ is thus calculated by

$$H(D) - Pr(true)H(D|true) - Pr(false)H(D|false).$$

By this analysis, we see that on the left (right) side of the cut, all points are treated equally as true (false), and thus the distance information between points in LU (RU) is ignored.

On the other hand, the loss of distance information can be seen by rewriting this traditional calculation of information gain in terms of random graphs as

$$\begin{aligned} &\tfrac{|LU|}{|U|}H(\text{IND}(D)|LU \times LU) + \tfrac{|RU|}{|U|}H(\text{IND}(D)|RU \times RU) \\ &-H(\text{IND}(D)|U \times U), \end{aligned} \tag{5.20}$$

where $H(D) = -H(\text{IND}(D)|U\times U)$, $Pr(true) = \frac{|LU|}{|U|}$, $Pr(false) = \frac{|RU|}{|U|}$. In the term $H(\text{IND}(D)|LU \times LU)$ ($H(\text{IND}(D)|RU \times RU)$), the equivalence relation $\text{IND}(D)$ is induced on LU (RU).

The loss of distance information can be seen in the three universal relations on LU, RU, and U. $U\times U$ means that all points are considered as the same in the calculation of $H(\text{IND}(D)|U \times U)$; this is reasonable because " before the cut" means no attribute is selected. $LU\times LU$ ($RU\times RU$) means that all points less (greater) than t are treated as equal in the calculation of $H(\text{IND}(D)|LU \times LU)$ ($H(\text{IND}(D)|RU\times RU)$). $LU\times LU$ ($RU\times RU$) indeed ignores the distance information, in which the potential information gains from further cuts is not considered. This might be the reason that traditional entropy produces a decision tree with an error in Figure 5.2, and fails to produce the ideal decision tree in Figure 5.3 in Example 9.

We now consider the inclusion of the distance information by establishing a random graph $RG = (U, P)$ on U. For a cut t, the random graph RG will induce two random graphs $LRG = (LU, LP)$ on the set LU and $RRG = (RU, RP)$ on the set RU. Note that the matrix $P = (p_{ij})$ can be partitioned as $\begin{pmatrix} LP & * \\ * & RP \end{pmatrix}$. Based on such a partition, the information gain $Gain(t)$ from the random graph dependency measure is calculated as

$$\tfrac{|LU|}{|U|} * H(\mathrm{IND}(D)|LRG) + \tfrac{|RU|}{|U|} * H(\mathrm{IND}(D)|RRG) - H(\mathrm{IND}(D)|U \times U), \tag{5.21}$$

where $U \times U$ is the universal relation, and is the representation of a complete graph. Before the cut, there is no attribute information affecting $\mathrm{IND}(D)$, and so we employ $U \times U$ in the third term $H(\mathrm{IND}(D)|U \times U)$; however, after the cut, the information of the attribute a has an effect on $\mathrm{IND}(D)$, and the random graphs LRG and RRG, containing distance information, are employed in the first term $H(\mathrm{IND}(D)|LRG)$ and the second term $H(\mathrm{IND}(D)|RRG)$) respectively. Note that Eq. (5.21) becomes the traditional information gain when both the left random graph LRG defined on the LU cut and right random graph RRG defined on RU are complete graphs.

Random Graph Generation on Data Points

Next, we aim to employ the random graph dependency measure to improve the C4.5 decision tree. In this section, we only discuss the data points on one real axis x because only the most informative attribute is chosen in each step of the tree-building procedure in C4.5.

As in Figure 5.1, we begin with some sorted data points $U = \{v_1, v_2, \dots, v_n\}$ whose values on the axis x satisfy $a_1 \le a_2 \le \dots \le a_n$. We hope to define a random graph $RG = (U, (p_{ij}))$ on these data points with the following properties:

1. $0 \leq p_{ij} \leq 1$ (Nonnegativity),
2. If $|a_i - a_j| > |a_k - a_l|$, then $p_{ij} \leq p_{kl}$ (Monotony on Distance), and
3. If $a_i \leq a_j \leq a_k$, then $p_{ij}p_{jk} = p_{ik}$ (Transitivity).

A natural choice is $p_{ij} = e^{-\sigma * |a_i - a_j|}$. In fact, it is our only choice if all these properties are satisfied on any set of data points. It is interesting to note that the random graph ($p_{ij} = e^{-\sigma * |a_i - a_j|}$) becomes a compete graph if $\sigma = 0$. In practice, we need to consider the scaling effect and adjust it. For this purpose, we set $p_{ij} = e^{-\sigma * |a_i - a_j| / |a_n - a_1|}$.

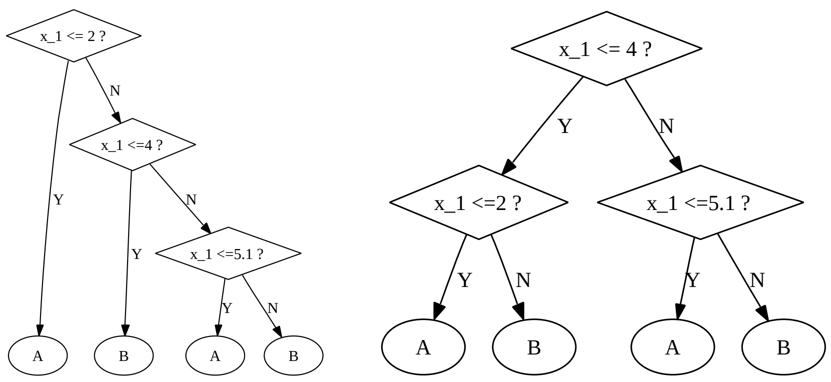
Remark. The above random graph generation method is only one possible way, although it is the only form that satisfies the three properties: nonnegativity, monotony on distance, and transitivity. Other methods of generating the random graph may better model the data points, although such a new method must break one of these properties. This needs further investigation; however, we prefer this natural choice because of its elegant form.

Before we introduce the algorithm for finding the best cut, we show some examples to help the reader to come familiar with the basic concepts in this section.

Examples

In this section, we show how to generate a random graph, how to calculate the random graph dependency measure according to the definition, how to calculate the information gains, and how the ideal decision tree in Figure 5.3 is produced by the new measure.

Example 42 In Table 5.2, for attribute x_1, $a_1 = 1.0 \leq a_2 = 2.0 \leq a_3 = 3.9 \leq a_4 = 4.0 \leq a_5 = 5.0 \leq a_6 = 5.1 \leq a_7 = 7.0 \leq a_8 = 8.0$. We show



(a) By the conditional entropy (b) By new measure ($\sigma = 3$)

Figure 5.6: An illustration on decision trees generated by two measures when x_2 is ignored.

the random graph P^2 generated by the method shown in last section. Before any cut, if we employ the random graph generation method ($\sigma = 3$), then $p^2_{12} = e^{-3*|a_1-a_2|/|a_8-a_1|} = e^{-3*|1.0-2.0|/|8.0-1.0|} = 0.65$.

Similarly we obtain all the elements in the random graph $P^2 =$

$$\begin{pmatrix} 1.00 & 0.65 & 0.29 & 0.28 & 0.18 & 0.17 & 0.08 & 0.05 \\ 0.65 & 1.00 & 0.44 & 0.42 & 0.28 & 0.26 & 0.12 & 0.08 \\ 0.29 & 0.44 & 1.00 & 0.96 & 0.62 & 0.60 & 0.26 & 0.17 \\ 0.28 & 0.42 & 0.96 & 1.00 & 0.65 & 0.62 & 0.28 & 0.18 \\ 0.18 & 0.28 & 0.62 & 0.65 & 1.00 & 0.96 & 0.42 & 0.28 \\ 0.17 & 0.26 & 0.60 & 0.62 & 0.96 & 1.00 & 0.44 & 0.29 \\ 0.08 & 0.12 & 0.26 & 0.28 & 0.42 & 0.44 & 1.00 & 0.65 \\ 0.05 & 0.08 & 0.17 & 0.18 & 0.28 & 0.29 & 0.65 & 1.00 \end{pmatrix}, \tag{5.22}$$

which is defined on $U = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

Example 43 Let $PG_1 = P^2 = (a_{ij})$, $RG_2 = P^4 = (b_{ij})$, then $H(RG_2|RG_1) = \frac{1}{n}\sum_{i=1}^{n} \log_2 \frac{\sum_k b_{ik}a_{ik}}{\sum_k a_{ik}} = -0.68$.

Example 44 In this example, we will show the information gain produced by the cut $x_1 = 4.5$ is 0.52. The details are shown below. Similarly the information gain for the seven possible cuts from the left to right are 0.37, 0.49, 0.41, 0.52, 0.41, 0.49, 0.37.

After the cut $x_1 = 4.5$, $RG_1 = P^2$ becomes

$$\begin{pmatrix}
1.00 & 0.65 & 0.29 & 0.28 & 0 & 0 & 0 & 0 \\
0.65 & 1.00 & 0.44 & 0.42 & 0 & 0 & 0 & 0 \\
0.29 & 0.44 & 1.00 & 0.96 & 0 & 0 & 0 & 0 \\
0.28 & 0.42 & 0.96 & 1.00 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.00 & 0.96 & 0.42 & 0.28 \\
0 & 0 & 0 & 0 & 0.96 & 1.00 & 0.44 & 0.29 \\
0 & 0 & 0 & 0 & 0.42 & 0.44 & 1.00 & 0.65 \\
0 & 0 & 0 & 0 & 0.28 & 0.29 & 0.65 & 1.00
\end{pmatrix}.$$

It induces two random graphs LRG_1 and RRG_1, which are defined on $LU = \{v_1, v_2, v_3, v_4\}$ and $RU = \{v_5, v_6, v_7, v_8\}$ respectively.

$$LRG_1 = \begin{pmatrix}
1.00 & 0.65 & 0.29 & 0.28 \\
0.65 & 1.00 & 0.44 & 0.42 \\
0.29 & 0.44 & 1.00 & 0.96 \\
0.28 & 0.42 & 0.96 & 1.00
\end{pmatrix}, RRG_1 = \begin{pmatrix}
1.00 & 0.96 & 0.42 & 0.28 \\
0.96 & 1.00 & 0.44 & 0.29 \\
0.42 & 0.44 & 1.00 & 0.65 \\
0.28 & 0.29 & 0.65 & 1.00
\end{pmatrix}.$$

$RG_2 = P^4$ is defined on $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. $RG_2|LU$ is the random graph induced by RG_2 on $LU = \{v_1, v_2, v_3, v_4\}$, and $RG_2|RU$ that induced by RG_2 on $RU = \{v_5, v_6, v_7, v_8\}$.

$$RG_2|LU = \begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1
\end{pmatrix}, RG_2|RU = \begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1
\end{pmatrix}.$$

On LU, let $LRG = (a_{ij})$, $RG_2|LU = (b_{ij})$, $n = |LU|$; we have $H(RG_2|LRG) = \frac{1}{n}\sum_{i=1}^{n} \log_2 \frac{\sum_k b_{ik}a_{ik}}{\sum_k a_{ik}} = -0.483$. On RU, let $RRG = (a_{ij})$, $RG_2|RU = (b_{ij})$, $n = |RU|$; we have $H(RG_2|RRG) = -0.483$. On U, let $U \times U = (a_{ij})$, $RG_2 = (b_{ij})$,

$n = |U|$, we have $H(RG_2|U \times U) = -1$. Note that $\text{IND}(D) = RG_2 = P^4$ in this example. By Eq. (5.21), the information gain from the cut $x_1 = 4.5$ is

$$\begin{aligned}&\tfrac{|LU|}{|U|} * H(\text{IND}(D)|LRG) + \tfrac{|RU|}{|U|} * H(\text{IND}(D)|RRG) - H(\text{IND}(D)|U \times U)\\ &= \tfrac{4}{8} * (-0.483) + \tfrac{4}{8} * (-0.483) - (-1)\\ &= 0.52.\end{aligned}$$

As we have shown in Example 9, the maximal information gain achieved in the conditional entropy approach is 0.31, which is obtained by the second cut. If we ignore the attribute x_2, then we choose the second cut as the best cut, which results in a decision tree shown in Figure 5.6 (a). In Example 44, we observe that when $\sigma = 3$, the largest information gain produced by the random graph dependency measure is achieved by the middle cut. Then we choose the fourth cut, which results in the decision tree shown in Figure 5.6 (b). This decision tree is more balanced than the previous one. However, in terms of accuracy, there is no difference between the decision tree produced by the traditional conditional entropy and the one produced by the random graph dependency measure, as shown in Figure 5.6 (a) and (b). The situation is different between the conditional entropy and the random graph dependency measure if we consider the attribute x_2. The details are shown in the coming example.

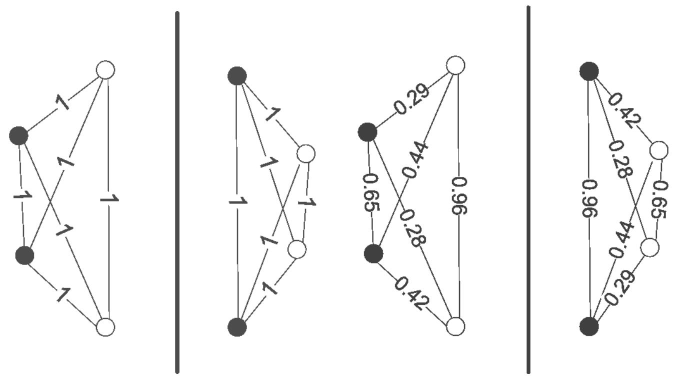
Example 45 If we do not ignore the attribute x_2, then the information gain achieved by this attribute x_2 is 0.19. If we employ the conditional entropy, we obtain the decision tree shown in Figure 5.2. If we employ the random graph generation method ($\sigma = 3$), then the maximal information gain (0.52) from the fourth cut is greater than 0.19, then x_1 is chosen as the attribute for the root of the decision tree, on which there is a decision $x_1 \leq 4$. As the tree-building procedure continues, the whole data set is divided into two parts: $LU = \{v_1, v_2, v_3, v_4\}$ and $RU = \{v_5, v_6, v_7, v_8\}$. On LU, the information gain induced by x_1 for the three possible cuts from left to right are 0.63, 1.00, and

0.66, and the information gain induced by x_2 is 0.31; On RU, the information gain induced by x_1 for the three possible cuts from left to right are 0.66, 1.00, and 0.63, and the information gain induced by x_2 is 0.31. On both LU and RU, we will continue to choose x_1 as the best attribute, and finally we obtain the decision tree shown in Figure 5.3.

Interpretations. Comparing the two decision trees in Figure 5.2 and Figure 5.3, we find that there is a classification error in Figure 5.2, whereas there is no classification error in Figure 5.3. The reason for this phenomenon is that the traditional conditional entropy underestimates the information contained in the continuous attribute x_1, and consequently the attribute x_2 is chosen, even though it is actually less informative. The choice of the less informative attribute will use up the training set quickly and leave no chance that x_1 will be chosen again. This situation is especially serious when the number of cases is small.

The underestimation of the information contained in the continuous attribute x_1 by the conditional entropy can be seen from the viewpoint of random graphs. After the middle cut $x_1 = 4.5$, the situations resulting from setting $\sigma = 0$ and $\sigma = 3$ are shown in Figure 5.7 (a) and (b). When $\sigma = 0$ (the traditional case), all the edges are the same in each of the two subgraphs $LRG = LU \times LU$ and $RRG = RU \times RU$ in Figure 5.7 (a), and thus the two subgraphs lose the distance information; in contrast, when $\sigma = 3$, the two subgraphs LRG and RRG in Figure 5.7 (b) still contain distance information, which can be seen in the fact that the edges are set nonequal in Figure 5.7(b).

Note that $\sigma = 0$ corresponds to the traditional method of computing information gain because $LRG = LU \times LU$ and $RRG = RU \times RU$ when $\sigma = 0$, and thus the proposed information gain in Eq. (5.21) become the traditional information gain in Eq. (5.20).



(a) $\sigma = 0$ (b) $\sigma = 3$

Figure 5.7: An illustration on the random graphs by setting $\sigma = 0$ and $\sigma = 3$ after the middle cut.

Next we will apply the random graph dependency measure to determine the best cut for a continuous attribute.

Algorithm for Best Cut Based on Random Graph Dependency Measure

According to Eq. (5.21) and Definition 35, for a continuous attribute a, we develop Algorithm 3 to choose the best cut and to calculate the corresponding information gain. In this algorithm, we omit the calculation of $Base = H(\mathrm{IND}(D)|U \times U)$, which is very easy.

5.3.3 Experiments

In the above sections, we have given a detailed explanation of the random graph dependency by presenting its definition and its various properties. In this section, we will show its significance in improving the accuracy of decision trees.

We replace the conditional entropy used in the C4.5R8 algorithm with the random graph dependency measure and replace the gain ratio with the corresponding gain ratio produced by the random graph dependency measure, so that a modified C4.5 algorithm is formed. The only difference between the modified C4.5 and the original C4.5R8 is its calculation of information gain on continuous attributes.

While CART [19] is another successful decision tree algorithm, we will not compare the modified C4.5 and CART in this section for the following two reasons:

1. In [63], the comparison between CART and C4.5 has been done.

2. The current definition of random graph dependency is not a generalization of the Gini diversity index used in the splitting criteria in CART; consequently, CART is not the counterpart of the random graph dependency. A direct comparison between the modified C4.5 and CART may not be helpful in distinguishing how much the random graph dependency improves on the traditional entropy.

The original C4.5R8 and the modified C4.5 are applied to six datasets from the UCI machine learning repository. C5.0 is a commercial product developed from C4.5 [1]. Its technical details and source codes are unknown, and so we cannot replace the information measure employed in C5.0 with the proposed random graph dependency. However, a comparison with C5.0 may help to show the significance of the modified C4.5 created by the proposed measure. The demonstration version C5.0R2 is a scaled-down versions of C5.0, limited to small size datasets (up to 400 cases). So we compare C5.0R2 with the modified C4.5 only on datasets with fewer than 401 cases.

Table 5.13 is a description of the datasets we use. The first column shows the names of the datasets, the second column gives the numbers of cases in each dataset, the third column gives the number of classes, the fourth column gives the number of continuous attributes, the fifth column gives the number of discrete attributes, and the final column describes whether there are missing values in each dataset.

Dataset	**Cases**	**Classes**	**Cont**	**Discr**	**Missing**
Glass	214	6	9	0	N
Labor	57	2	8	8	Y
Sonar	208	2	60	0	N
Lymph	148	4	3	15	N
Iono	351	2	34	0	N
Hepatitis	155	2	6	13	Y

Table 5.13: Description of the datasets

The experiments are conducted on a workstation whose hardware model is Nix Dual Intel Xeon 2.2GHz, with 1GB of RAM, using Linux Kernel 2.4.18-27smp (RedHat7.3).

For all the datasets, the strictest cross-validation (leave-one-out) is applied to evaluate the performance; for example, 57-fold cross-validation is applied to the dataset Labor. There is no random factor for the choices of training data and testing data in the leave-one-out cross-validation, and thus the results produced are repeatable and independent of the order of the cases in the datasets; this is not the case for the 10-fold cross-validation.

We compare the original C4.5R8 with the modified C4.5 using the information gain and information gain ratio respectively. Tables 5.14, 5.15, 5.16, and 5.17 show the results for the option of information gain, and Tables 5.18, 5.19, 5.20, and 5.21 show the results for the option of information gain ratio. The parameter σ in the modified C4.5 is adjusted by cross-validation on training data. As a baseline, we show the results after pruning for C5.0R2 using the default option in both (only the results after pruning are available in C5.0R2).

In Tables 5.14, 5.15, 5.16, 5.18, 5.19, and 5.20, the second and fourth columns are the results of original C4.5R8 before and after pruning respectively. The third and sixth columns are the results of the modified C4.5 before and after pruning respectively, and the fifth column indicates the results of C5.0R2. In all the tables, the numbers in brackets are the standard deviations for the sample means (also called standard errors), which are calculated by $S/\sqrt{n}$, where S^2 is the sample variance, an estimator for the population variance, and n is the size of sample (the number of cross-validations).

If the difference between the error rate for the modified C4.5 before pruning and that for the original C4.5R8 before pruning is statistically significant at a significance level 25%, the best results are shown in bold in Tables 5.14 and 5.18. If the difference between the best and the second best among the error rates for the modified C4.5 after pruning, that for the original C4.5R8 after pruning, and that for C5.0R2 is statistically significant at a significance level 25%, the best results are shown in bold in Tables 5.14 and 5.18.

If the difference between the error rate for the modified C4.5 before pruning (after pruning) and that for the original C4.5R8 before pruning (after pruning) is statistically significant at a significance level 5%, we put a mark $*$ beside the results in the third column (the fifth column) in Tables 5.14 and 5.18. If the difference between the error rate for the modified C4.5 after pruning and that for C5.0R2 is statistically significant at a significance level 5%, we put a mark $\star$ beside the results in the fifth column in Tables 5.14 and 5.18.

The values shown in Table 5.14 and Table 5.18 are the mean error rates (error rate = 100% - classification rate) of the leave-one-out cross-validations of the original C4.5R8, the modified C4.5 and C5.0R2. The values shown in Table 5.15 and Table 5.19 describe the average number of nodes of the decision trees of the leave-one-out cross-validations. Since the continuous attributes are used in an improved way by the proposed random graph dependency measure, it would be interesting to know if the continuous attributes are selected more often in the tree when the random graph dependency measure is employed. In Table 5.16 and Table 5.20, we show the average frequency of appearance of continuous attributes in the nodes of the decision trees of the leave-one-out cross-validations. In Table 5.17 and Table 5.21, we show the time required for running the test part of the original C4.5R8 and the modified C4.5 10000 times. Because of the short test time, we have to run the testing part 10000 times in order to record the time. For C5.0R2, because we do not have the source codes and so we cannot insert the time counter in C5.0R2, we cannot show the testing time for C5.0R2 in Table 5.17 and Table 5.21.

	Unpruned		**Pruned**		
Dataset	**C4.5R8 -g**	**N-g**	**C4.5R8 -g**	**C5.0R2**	**N-g**
Glass	32.7 (3.21)	**29.0 (3.10)**	35.0 (3.26)	33.6 (3.23)	**29.0 (3.10)**
Labor	26.3 (5.83)	**12.3 (4.35)** ∗	29.8 (6.06)	19.3 (5.23)	15.8 (4.83) ∗
Sonar	27.5 (3.11)	**17.8 (2.65)** ∗	27.5 (3.11)	26.9 (3.08)	**17.8 (2.65)**∗ ⋆
Lymph	28.4 (3.71)	**23.0 (3.46)**	25.7 (3.59)	22.3 (3.42)	23.0 (3.46)
Iono	12.3 (1.75)	**9.4 (1.56)** ∗	12.0 (1.73)	8.5 (1.49)	9.4 (1.56)∗
Hepatitis	20.6 (3.25)	**11.6 (2.57)** ∗	16.8 (3.00)	21.3 (3.29)	**13.6 (2.75)**⋆

Table 5.14: Mean error rates (percentage) of the original C4.5R8 using information gain (C4.5R8 -g), the modified C4.5 using information gain (N-g), and C5.0R2.

	Unpruned		**Pruned**		
Dataset	**C4.5R8 -g**	**N-g**	**C4.5R8 -g**	**C5.0R2**	**N-g**
Glass	52.8 (0.13)	63.0 (0.28)	52.6 (0.14)	54.0 (0.16)	58.9 (0.28)
Labor	16.4 (0.59)	10.7 (0.48)	8.8 (0.61)	5.4 (0.45)	5.47 (0.36)
Sonar	32.8 (0.11)	34.6 (0.09)	32.8 (0.11)	34.1 (0.22)	34.6 (0.10)
Lymph	57.7 (0.36)	52.3 (0.48)	20.0 (0.12)	26.3 (0.15)	19.4 (0.27)
Iono	36.5 (0.09)	29.1 (0.05)	36.4 (0.10)	30.6 (0.10)	28.4 (0.11)
Hepatitis	36.0 (0.22)	25.6 (0.21)	12.2 (0.26)	19.0 (0.13)	17.8 (0.17)

Table 5.15: Average number of nodes of of the original C4.5R8 using information gain (C4.5R8 -g), the modified C4.5 using information gain (N-g), and C5.0R2.

	Unpruned		**Pruned**		
Dataset	**C4.5R8 -g**	**N-g**	**C4.5R8 -g**	**C5.0R2**	**N-g**
Glass	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Labor	0.47 (0.01)	0.54 (0.03)	0.75 (0.03)	0.88 (0.02)	0.94 (0.02)
Sonar	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Lymph	0.008 (0.002)	0.22 (0.01)	0.009 (0.003)	0.21 (0.003)	0.23 (0.01)
Iono	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Hepatitis	0.44 (0.002)	0.56 (0.01)	0.58 (0.006)	0.44 (0.004)	0.69 (0.00)

Table 5.16: Average frequency of appearance of continuous attributes in the nodes of the original C4.5R8 using information gain (C4.5R8 -g), the modified C4.5 using information gain (N-g), and C5.0R2.

The experiments show that the random graph dependency measure is a useful measure. We compare three aspects of the modified C4.5 algorithm using the random graph dependency measure with the original C4.5R8 algorithm using the conditional entropy:

Prediction accuracy: We observe that, before pruning, the modified C4.5 outperforms the original C4.5R8 in prediction accuracy on these six datasets

Dataset	C4.5R8-g	N-g
Glass	15.607 (0.339)	16.636 (0.323)
Labor	11.228 (1.025)	10.228 (0.611)
Sonar	8.462 (0.277)	9.135 (0.247)
Lymph	9.932 (0.067)	09.865 (0.095)
Iono	8.718 (0.204)	9.459 (0.180)
Hepatitis	10.839 (0.577)	12.258 (0.733)

Table 5.17: Mean time, in milliseconds, for 10,000 test runs of the original C4.5R8 using information gain (C4.5R8 -g) and the modified C4.5 using information gain (N-g).

	Unpruned		Pruned		
Dataset	**C4.5R8**	**N**	**C4.5R8**	**C5.0R2**	**N**
Glass	32.7 (3.21)	**27.1 (3.04)**	35.1 (3.26)	33.6 (3.23)	**26.2 (3.00)** ∗∗
Labor	21.1 (5.40)	**14.0 (4.60)**	22.8 (5.56)	19.3 (5.23)	15.8 (4.83)
Sonar	32.2 (3.24)	**22.1 (2.88)**∗	32.2 (3.24)	26.9 (3.08)	22.1 (2.88) ∗
Lymph	38.3 (0.14)	**21.0 (3.34)**∗	33.5 (0.14)	22.3 (3.42)	22.3 (3.42) ∗
Iono	13.1 (1.80)	**11.1 (1.68)**	13.1 (1.80)	**8.5 (1.49)**	11.1 (1.68)
Hepatitis	18.7 (3.13)	18.7 (3.13)	19.4 (3.17)	21.3 (3.29)	19.4 (3.17)

Table 5.18: Mean error rates (percentage) of the original C4.5R8 using information gain ratio (denoted as C4.5R8), the modified C4.5 using information gain ratio (denoted as N), and C5.0R2.

for both the information gain option and the information gain ratio option. After pruning, the modified C4.5 still outperforms the original C4.5R8. The modified C4.5 algorithm seems more successful in the dataset Labor, in which the algorithm before pruning achieves a 12.3% prediction error rate, while the original algorithm before pruning has a 26.3% error rate; after pruning, the modified C4.5 outperform both the original C4.5R8 and C5.0R2 greatly on dataset Labor.

A reasonable explanation is that the better prediction accuracy shown by the modified C4.5 is achieved by more accurately calculating the information gain hidden in the continuous attributes, which is partly ignored by the traditional conditional entropy. The under-estimated information gain is increased by the new measure, which results in a phenomenon of the continuous attributes being selected more often in the tree, as shown in Table 5.16. But in Table 5.20, there is no such phenomenon because the information gain ratio for the random graph dependency includes the consideration of the efficiency

	Unpruned		**Pruned**		
Dataset	**C4.5R8**	**N**	**C4.5R8**	**C5.0R2**	**N**
Glass	51.2 (0.15)	52.3 (0.24)	45.4 (0.16)	54.0 (0.16)	45.9 (0.42)
Labor	18.8 (0.47)	12.7 (0.37)	5.6 (0.28)	5.4 (0.45)	8.6 (0.33)
Sonar	34.7 (0.17)	32.9 (0.09)	34.6 (0.17)	34.1 (0.22)	32.9 (0.09)
Lymph	57.7 (0.36)	45.2 (0.33)	20.0 (0.12)	26.3 (0.15)	31.0 (0.23)
Iono	34.0 (0.11)	32.9 (0.07)	33.8 (0.12)	30.6 (0.10)	32.3 (0.12)
Hepatitis	31.1 (0.10)	31.1 (0.10)	22.6 (0.12)	19.0 (0.13)	22.6 (0.12)

Table 5.19: Average number of nodes of the original C4.5R8 using information gain ratio (C4.5R8), the modified C4.5 using information gain ratio (N), and C5.0R2.

	Unpruned		**Pruned**		
Dataset	**C4.5R8**	**N**	**C4.5R8**	**C5.0R2**	**N**
Glass	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Labor	0.42 (0.01)	0.77 (0.03)	0.94 (0.02)	0.88 (0.02)	0.89 (0.02)
Sonar	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Lymph	0.27 (0.00)	0.19 (0.00)	0.23 (0.00)	0.21 (0.003)	0.09 (0.00)
Iono	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Hepatitis	0.44 (0.002)	0.44 (0.002)	0.58 (0.006)	0.44 (0.004)	0.58 (0.006)

Table 5.20: Average frequency of appearance of continuous attributes in the nodes of the original C4.5R8 using information gain ratio (C4.5R8), the modified C4.5 using information gain ratio (N), and C5.0R2.

of the partition of the data, which may reduce the frequency of continuous attributes in decision trees. The mixture of the function of improving continuous attributes and that of efficiently partitioning the data in the gain ratio, results in the disappearance of the increasing frequency phenomenon.

Size of tree: From Tables 5.15 and 5.19, we can see that the number of nodes are of the same scale for all the algorithms.

Speed: The calculation of the random graph dependency is more time-consuming than the conditional entropy. This is one weakness of the new measure, and deserves further investigation. As a result, the training of the modified C4.5 is slower than that of the original C4.5R8 and C5.0R2; however, the test time is in the same scale as C4.5R8, as shown in Tables 5.17 and 5.21, and which can be roughly explained by trees having similar numbers of nodes in Tables 5.15 and 5.19. In fact, the time consumption problem is not serious on small sample size problems; moreover, on large size problems, it can be reduced by first employing the conditional entropy when the number of cases

Dataset	**C4.5R8**	**N**
Glass	18.364 (0.253)	17.664 (0.289)
Labor	8.947 (0.641)	10.000 (0.702)
Sonar	9.808 (0.204)	9.519 (0.261)
Lymph	10.000 (0.000)	9.797 (0.278)
Iono	11.339 (0.190)	9.573 (0.198)
Hepatitis	8.710 (0.617)	8.710 (0.617)

Table 5.21: Mean time, in milliseconds, for 10,000 test runs of the original C4.5R8 using information gain ratio (C4.5R8) and the modified C4.5 using information gain ratio (N).

in the current node is greater than a threshold and then employing the random graph dependency when the number of cases is less than the threshold.

We have also tested six other datasets: Auto, Colic, Iris, Heart-c, Heart-h and Wine, all having fewer than 401 cases. In the following, we give a brief description comparing the accuracy between algorithms (before pruning) comparison for the information gain option. We find that on datasets Colic, Wine and Auto, the modified C4.5 slightly outperforms the original C4.5R8. On datasets Iris, Heart-c, and Heart-h, the error rates for the modified C4.5 are exactly the same as those for the original C4.5R8; this happens when σ is found to be zero by the cross-validation on the training data. On datasets Wine and Heart-c, the modified C4.5 slightly outperforms C5.0R2 (again, the error rate decreases by less than 2%); on dataset Iris, there is no difference between the modified C4.5 and C5.0R2; On dataset Colic, the modified C4.5 is slightly worse than C5.0R2; On datasets Auto and Heart-h, C5.0R2 outperforms the modified C4.5 so that the error rate is reduced between 3 and 5 percent, the same margin by which C5.0R2 outperforms C4.5R8. We believe that C5.0R2 can also be improved in accuracy if the information measure used in C5.0R2 is replaced with the random graph dependency.

5.3.4 Summary

The random graph dependency measure $H(RG_2|RG_1)$ has some useful properties, such as the Partial Order Preserving Property and the Anti-Partial Order

Preserving Property. It generalizes the conditional entropy because it becomes equivalent to the conditional entropy when the random graphs takes their special forms–equivalence relations. Our experimental study demonstrates that the random graph dependency measure is an informative measure, showing its success in decision trees on small sample size problems.

5.4 The General Random Graph Dependency Measure $\Gamma_{\alpha}^{\varepsilon}(RG_2|RG_1)$

In sections 5.2 and 5.3, we have proposed two dependency measures, one is restricted to equivalence relations, and is connected to γ used in Rough Set Theory; the other can be applied to general random graphs and is the generalization of the conditional entropy. In this section, we show these two measures and γ are special cases of a general measure. To explain this point clearly, we need to borrow the concept of $\alpha-$mean of positive numbers, for more materials, see [40, 43].

5.4.1 Definitions

Given two positive numbers a and b, their arithmetic mean is $(a+b)/2$. However, there are other types of means: The geometric mean is $\sqrt{ab}$, and the harmonic mean is $2/(a^{-1}+b^{-1})$. The concept of the mean can be generalized in the following way [40]. Let f be a differentiable monotone function.

Definition 46 The $f-$mean of a and b is defined as $m_f(a,b) = f^{-1}(f(a)+f(b))$. For n numbers $a_1, a_2, \ldots, a_n$, the weighted $f-$mean of $a_1, a_2, \ldots, a_n$ is defined as

$$m_f(a_1, a_2, \ldots, a_n) = f^{-1}(w_1 f(a_1) + w_2 f(a_2) + \ldots + w_n f(a_n)) \qquad (5.23)$$

where $w_1, w_2 \ldots, w_n$ are the weights, $w_i > 0$, and $\sum_i w_i = 1$.

The $f-$mean is called linear scale-free, if, for $c > 0$, the $f-$mean of ca and cb is c times their $f-$mean, i.e., $m_f(ca, cb) = cm_f(a, b)$.

Theorem 47 (Hardy, Littlewood and Polya)

The $f-$mean is linear scale-free if and only if f is the following function for some α,

$$f_\alpha(u) = \begin{cases} u^{\frac{1-\alpha}{2}}, & \alpha \neq 1, \\ \log u, & \alpha = 1. \end{cases} \tag{5.24}$$

Now we are ready to define a general random graph dependency measure.

Definition 48 $\Gamma_\alpha^\varepsilon(RG_2|RG_1)$ is defined as

$$\Gamma_\alpha^\varepsilon(RG_2|RG_1) = \frac{1}{|U|} \sum_{x \in U^\varepsilon(RG_1, RG_2)} f_\alpha(\frac{|RG_2(x) \cap RG_1(x)|}{|RG_1(x)|}) \tag{5.25}$$

where $RG_2(x)$ and $RG_1(x)$ denote the random neighbors of x in RG_2 and RG_1 respectively, and

$$U^\varepsilon(RG_1, RG_2) = \{x | x \in U \wedge |RG_2(x) \cap RG_1(x)|/|RG_1(x)| \geq \varepsilon\}.$$

Remark. $\Gamma_\alpha^\varepsilon(RG_2|RG_1)$ can also be defined as $f_\alpha^{-1}(\frac{1}{|U|} \sum_{x \in U^\varepsilon(RG_1, RG_2)} f_\alpha(\frac{|RG_2(x) \cap RG_1(x)|}{|RG_1(x)|}))$ in terms of $f-$mean.

It is easy to check that

Theorem 49 (Generality of $\Gamma_\alpha^\varepsilon(RG_2|RG_1)$)

- When $\varepsilon = 1$, $\alpha = -1$, and RG_2, RG_1 are the equivalence relations induced by two attribute sets D and C, $\Gamma_\alpha^\varepsilon(RG_2|RG_1) = \gamma(C, D)$.
- When $\varepsilon = 0$, $\alpha = -1$, and RG_2, RG_1 are the equivalence relations induced by two attribute sets D and C, $\Gamma_\alpha^\varepsilon(RG_2|RG_1) = \Gamma(C, D)$.
- When $\varepsilon = 0$ and $\alpha = 1$, $\Gamma_\alpha^\varepsilon(RG_2|RG_1) = H(RG_2|RG_1)$.
- When $\varepsilon = 0$, $\alpha = 1$, and RG_2, RG_1 are the equivalence relations induced by two attribute sets D and C,$\Gamma_\alpha^\varepsilon(RG_2|RG_1) = H(D|C)$

Besides these four special cases, there are many intermediate cases. We only apply one of them to find the free parameters in the heat diffusion classifiers, and leave others in the future work.

5.4.2 Find the Free Parameters in Heat Diffusion Classifiers

There are three free parameters in *G-HDC*: K, β, and γ. As an example, in this chapter, we only investigate *KNN-HDC*. Usually we employ the cross-validation to find these parameters. With the tool developed in the previous section, we can find K and β by searching the corresponding dependency values. By such a method, we can reduce the time complexity.

In *KNN-HDC*, at each fold of the cross-validation for each fixed K, each fixed β and each fixed γ, we need to do multiplications p times between a matrix $I + \frac{\gamma}{p}$ and a vector $f(0)$, and $K * n * p$ multiplications are needed, where n is the number of data including both labeled data and unlabeled data; By the random graph dependency, we only need to do $K * n$ multiplications and n divisions, which save us a lot of time. Although we can also find γ by the random graph dependency measure, it is time-consuming because we need to $n * n$ multiplications and n divisions, and so we only search K and β by the measure, and find γ by cross-validation.

Random Graph Generation

The basic idea is that the data points form a random graph $R^1(K, \beta)$ for a given K and β, and the labels form another random graph R^2 as follows.

$$R^1(K,\beta)_{ij} = \begin{cases} 1, & j = i; \\ e^{-w_{ij}^2/\beta}, & \text{if } j \text{ is one of the neighbors of } i; \\ 0, & \text{otherwise.} \end{cases} \tag{5.26}$$

$$R^2_{ij} = \begin{cases} 1, & j = i; \\ 1, & \text{if } j \text{ and } i \text{ are labeled same;} \\ 0, & \text{if } j \text{ and } i \text{ are labeled differently;} \\ p_l, & \text{if one of } i \text{ and } j \text{ is labeled as } l \text{ while the other is not labeled;} \\ r, & \text{if both } j \text{ and } i \text{ are not labeled.} \end{cases} \tag{5.27}$$

Where p_l is the frequency of the data with label l in the labeled data, $r = \sum_{i=1}^{c} p_i^2$, and c is the number of classes.

Interpretation. When one of i and j is labeled as l while the other is not labeled, the unlabeled data has a probability p_l of being labeled as l, and so the probability that i and j have the same label is p_l; when both i and j are not labeled, the probability that i and j have the same label is $\sum_{i=1}^{c} p_i^2$.

Dependency Calculation

We adopt $\Gamma_\alpha^\varepsilon(RG_2|RG_1)$ to measure the dependency between R^1 and R^2. In the same way as $\Gamma(C, D)$ and $H(RG_2|RG_1)$, we set $\varepsilon = 0$. And in this section, we aim to reduce the time complexity, and so we set $\alpha = -1$, by which the time-consuming logarithm operation is avoided. More specifically, when $\varepsilon = 0$ and $\alpha = -1$, $\Gamma_\alpha^\varepsilon(RG_2|RG_1)$ becomes

$$\begin{aligned} \Gamma^0_{-1}(RG_2|RG_1) &= \frac{1}{|U|} \sum_{x \in U} \frac{|RG_2(x) \cap RG_1(x)|}{|RG_1(x)|} \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{|RG_2(v_i) \cap RG_1(v_i)|}{|RG_1(v_i)|} \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_k a_{ik} b_{ik}}{\sum_k a_{ik}}, \end{aligned} \tag{5.28}$$

where $U = \{v_1, v_2, \ldots, v_n\}$, $RG_1 = (a_{ij})$ and $RG_2 = (b_{ij})$.

Calibration of Underestimated Random Graph

It is observed that when $RG_1 = I$, $\Gamma^0_{-1}(RG_2|RG_1)$ reaches its maximal value. For the random graph $R^1(K,\beta)$, when $\beta \to 0$, it becomes I, and $\Gamma^0_{-1}(R^2|R^1(K,\beta))$ becomes maximal. But $\beta = 0$ may not be the best parameter because the true random graph representing the data point relationship may not be I. Such a phenomenon means that we will obtain wrong parameters if we simply employ the maximum dependency principle to find the parameters.

To calibrate $\Gamma^0_{-1}(RG_2|RG_1)$ in the case when RG_1 is underestimated, we multiply a factor $\Gamma^0_{-1}(RG_1|U\times U)$ in $\Gamma^0_{-1}(RG_2|RG_1)$. According to Eq. (5.28), $\Gamma^0_{-1}(RG_1|U \times U) = \frac{1}{n}\sum\limits_{i=1}^{n} \frac{\sum_k a_{ik}}{n} = \frac{1}{n^2}\sum\limits_{i=1}^{n}\sum_k a_{ik}$, which means the total weight of the random graph RG_1. When $RG_1 = I$, $\Gamma^0_{-1}(RG_1|U \times U) = \frac{1}{n}$ is minimal. This fits the intuition that $RG_1 = I$ contains least information. As a result, $\Gamma^0_{-1}(RG_2|RG_1)\Gamma^0_{-1}(RG_1|U \times U)$ can balance the dependency between RG_1 and RG_2 and the information contained in RG_1. we formalize the method of searching K and β as follows.

$$\max_{K,\beta}\Gamma^0_{-1}(R^2|R^1(K,\beta))\Gamma^0_{-1}(R^1(K,\beta)|U \times U)$$

Experiments

Using the same settings as those in Section 3.10, we record the time used for finding K and β, and accuracy by using the cross-validation and the random graph dependency respectively. The results are shown in Table 5.22.

From the results, we observe that the average training time by the random graph dependency measure is reduced greatly (81.27%) while the overall average of accuracy remains the same.

Table 5.22: Mean time in seconds and accuracy on the 11 datasets achieved by ten runs by dividing the data into 10% for training and 90% for testing by the cross-validation and by the random graph dependency measure

Dataset	**Time by CV**	**Time by Γ^0_{-1}**	**Accuracy by CV**	**Accuracy by Γ^0_{-1}**
Spiral-1000	24.8	**1.0**	92.7	89.0
Variance	0.007	0.002	0.61	0.58
Credit-a	252.7	**35.0**	61.6	65.7
Variance	0.098	0.01	1.53	0.72
Iono	39.5	**12.5**	80.3	72.6
Variance	0.004	0.006	1.67	2.27
Iris	34.5	**2.05**	91.7	92.5
Variance	13.566	0.002	2.18	1.08
Diabetes	281.0	**36.6**	67.1	69.3
Variance	0.121	0.016	0.88	0.77
Glass	19.6	**7.9**	55.5	40.2
Variance	0.004	0.006	1.36	0.83
Breast-w	148.8	**37.0**	95.7	95.6
Variance	0.033	0.028	0.21	0.11
Waveform	30.2	**10.3**	74.4	71.9
Variance	0.006	0.005	1.23	1.84
Wine	5.1	**0.3**	63.6	65.7
Variance	0.003	0.002	2.05	2.10
Anneal	62.9	**23.9**	75.6	75.4
Variance	0.032	0.032	0.50	0.46
Heart-c	56.4	**12.4**	59.3	62.1
Variance	0.005	0.006	1.32	0.89
Average	86.86	**16.27**	74.32	74.32

5.4.3 Summary

By using $\Gamma^0_{-1}(RG_2|RG_1)$, the parameters K and β can be found faster than the naive cross-validation method, and the produced accuracy is better in most of these datasets.

Algorithm 3 Input: sorted data points $x_1 \leq x_2 \leq \ldots \leq x_n$ and their class $c_1, c_2, \ldots, c_n$; σ. Output: Best cut t, and Best information gain G

```
$D \leftarrow x_n - x_1$
for $j = 1$ TO $n$ do
  Initialize SumR1[$j$], SumR2[$j$], SumL1[$j$], SumL2[$j$]
  for k=1 TO $n$ do
    $d \leftarrow e^{-\sigma * |x_j - x_k|/D}$
    SumR1[$j$] $\leftarrow$ SumR1[$j$]+ $d$
    if $c_j = c_k$ then
      SumR2[$j$] $\leftarrow$ SumR2[$j$] + $d$
    end if
  end for
end for
for $i = 1$ TO $n - 1$ do
  for $j = 1$ TO $n$ do
    $d \leftarrow e^{-\sigma * |x_j - x_i|/D}$
    SumR1[$j$]-= $d$, SumL1[$j$] +=$d$
    if $c_j = c_i$ then
      SumR2[$j$]-=$d$, SumL2[$j$] += $d$
    end if
  end for
  Initialize $SL1, SL2, SR1, SR2$
  for $j = 1$ TO $n$ do
    if $j < i + 1$ then
      $SL1$ += $\log_2$(SumL1[$j$])
      $SL2$ += $\log_2$(SumL2[$j$])
    else
      $SR1$ += $\log_2$(SumR1[$j$])
      $SR2$ += $\log_2$(SumR2[$j$])
    end if
  end for
  Gain[$i$] $\leftarrow$ ((SL2-SL1 +SR2-SR1) / $n$ - $Base$)
end for
Choose $I$ such that Gain[$I$] is maximal
$t \leftarrow (x_I + x_{I+1})/2$, $G \leftarrow$ Gain[I]
```

Chapter 6

Conclusion and Future Work

In this chapter, an overall summary of this thesis is provided. Following that, we then present future potential extensions of current work both within the proposed models and beyond the developed approaches.

6.1 Conclusion

In this thesis, we provide a random graph perspective over the field of machine learning. This perspective benefits us by two frameworks and one general tool.

Heat diffusion models on random graphs leads to a family of classifiers–Heat Diffusion Classifier on a Graph (*G-HDC*), and a ranking algorithm *Diffusion-Rank*.

G-HDC is one framework, it has the following advantages: 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. As a specific *G-HDC*, *VHDC* has an extra advantage that it can model the effect of unseen points by introducing the volume of a node. While *VHDC* is a generalization of *KNN-HDC*, which 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, which performs better than some recently proposed transductive methods.

Predictive Random Graph Ranking is another framework that incorporates *DiffusionRank*. We have shown that the *Temporal Web Prediction Model* is effective in *PageRank* and *DiffusionRank*. Because our model mines more information about the Web structure, the results of Predictive strategy on some ranking algorithms are more accurate than those without it. We conclude that the random graph input indeed extends the scope of some original ranking techniques, and significantly improve some of them.

Served as a component of the Predictive Random Graph Ranking framework, *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 including varying γ_i 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 best both in reduction effects and in its stability among all the three algorithms.

The random graph dependency measure $\Gamma^{\varepsilon}(RG_2|RG_1))$ is general tool to measure the dependency between two random graphs. In this thesis, we show its three successful specifications.

In the cases that RG_2, RG_1 are the equivalence relations induced by two attribute sets D and C, when $\varepsilon = 1$, $\alpha = -1$, $\Gamma^{\varepsilon}_{\alpha}(RG_2|RG_1)$ becomes the measure $\gamma(C, D)$ used in Rough Set Theory; When $\varepsilon = 0$, $\alpha = 1$, $\Gamma^{\varepsilon}_{\alpha}(RG_2|RG_1)$

becomes $H(D|C)$, the traditional conditional entropy. $\gamma(C, D)$ is not accurate when there are a lot of indeterministic minimal rules, while $H(D|C)$ involve the time-consuming operations of logarithm of the frequency, empirically the middle case when $\varepsilon = 0$, $\alpha = -1$, $\Gamma_{\alpha}^{\varepsilon}(RG_2|RG_1) = \Gamma(C, D)$ can improve the training time of the C4.5R8 decision trees while it preserves the classification accuracy of the original C4.5R8.

Theoretically we give three different forms of $\Gamma(C, D)$ in terms of equivalence relations, minimal rule, and arithmetic operation, respectively. Among our three different forms of the generalized dependency degree, the first form of the measure (in terms of equivalence relations) is the most important. Besides its simplicity, the first form is flexible, and it can therefore be extended not only to an equivalence relation but also to an arbitrary relation. The first form (in terms of equivalence relations) and the second form (in terms of minimal rules) share the advantage of being easily understood, while the third form of the measure (in terms of arithmetic operations) is computationally efficient. So these three forms of the measure are suited to different situations. When we want to extend the measure to a more complicated data structure (such as partial order relation, totally order relation or others) than an equivalence relation, or when we want to find some properties of this measure, we can employ the first two forms of the measure. When we use it in a computing situation, the third form of the measure may be the best choice. In fact, in this thesis, we determine its properties using the first two forms, and then in the experiments we use the third form. The generalized dependency degree Γ has some properties, such as the Partial Order Preserving Property and the Anti-Partial Order Preserving Property. Besides, its value is between zero and one. Therefore, it can serve as an index to measure how much decision attributes depend on conditional attributes. The experimental study shows that the generalized dependency degree is an informative measure in decision trees and attribute selection.

In the case when $\varepsilon = 0$ and $\alpha = 1$, $\Gamma_{\alpha}^{\varepsilon}(RG_2|RG_1) = H(RG_2|RG_1)$. $H(RG_2|RG_1)$ generalizes the conditional entropy because it becomes equivalent to the conditional entropy when the random graphs takes their special forms–equivalence relations. Our experimental study demonstrates that $H(RG_2|RG_1)$ is an informative measure, showing its success in improving accuracy of decision trees on small sample size problems.

In the case when $\varepsilon = 0$ and $\alpha = -1$, $\Gamma_{-1}^{0}(RG_2|RG_1)$ can help to search parameters K and β in *KNN-HDC* faster and more accurate than the cross-validation method.

6.2 Future Work

In order to capitalize on the promising achievements of *G-HDC*, further study is needed on the following problems: How to apply *G-HDC* to inductive learning, how to find a graph that better approximates the manifold in stead of the *KNN* graph, how to construct a better volume representation of the unseen points, how to utilize some feature extraction methods in order to make the local distance more accurate, how to find the parameters with avoiding the local minimum problem, and how to design a criteria for better setting the initial temperatures.

Concerning the random graph ranking, it deserves to put more efforts on *SimRank*. In our experiments, we only test the *Predictive Random Graph Ranking* in the viewpoint of dynamic Web. It deserves further investigation in other two viewpoints of partial observers and weighted links. Such future work involves investigating page-makers' preference on link orders and substantial users-based research.

Concerning the random graph dependency, our experiments only show some possible applications of $\Gamma_{\alpha}^{\varepsilon}(RG_2|RG_1)$ in the field of decision trees, and in the field of searching free parameters in *G-HDC*, and there is much future work left. In the future we will investigate the measure in other fields where the conditional entropy is applicable. For the measure itself, we need to find more properties to better understand it. The time consumption problem in the calculation of the random graph dependency is one problem that we need to solve in the future for other applications such as Web page dependency, in which there are three random graphs—the link graph produced by the hyperlinks, the similarity graph produced by the contents of the pages, and the class graph produced by a supervisor.

All the work depends on the generation of the underlying random graph, and so the more accurate random graph generation methods are expected.

Bibliography

[1] See5: An informal tutorial. *Rulequest Research, http://www.rulequest.com/see5-win.html*, 2006.

[2] http://www.oclug.on.ca/keys/report.php, 2007.

[3] Eugene Agichtein, Eric Brill, and Susan T. Dumais. Improving web search ranking by incorporating user behavior information. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 19–26, 2006.

[4] Gianni Amati, Iadh Ounis, and Vassilis Plachouras. The dynamic absorbing model for the web. Technical Report TR-2003-137, University of Glasgow, April 2003.

[5] Anneleen Van Assche, Celine Vens, Hendrik Blockeel, and Sašo Džeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 64(1):149–182, 2006.

[6] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2003.

[7] Ricardo A. Baeza-Yates, Paolo Boldi, and Carlos Castillo. Generalizing pagerank: damping functions for link-based ranking algorithms. In

Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 308–315, 2006.

[8] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[9] Randolph E. Bank and Michael Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM Review*, 45(2):291–323, 2003.

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

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

[12] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

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

[14] Alexander I. Bobenko and Boris A. Springborn. A discrete laplace-beltrami operator for simplicial surfaces. Preprint http://www.arxiv.org/math/0503219, 2005.

[15] Béla Bollobás. *Random Graphs.* Academic Press Inc. (London), 1985.

[16] Prosenjit Bose, Luc Devroye, William Evans, and David Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeletons. *SIAM J. Discret. Math.*, 20(2):412–427, 2006.

[17] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[19] Leo Breiman, Jerome H. Friedman, Richard Olshen, and Charles Stone. *Classification and regression trees*. Belmont: Wadsworth International Group, 1984.

[20] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 89–96, 2005.

[21] Junghoo Cho and Robert E. Adams. Page quality: In search of an unbiased web ranking. Technical report, UCLA Computer Science Department, November 2003.

[22] Junghoo Cho and Sourashis Roy. Impact of search engines on page popularity. In *Proceeding of the 13th World Wide Web Conference (WWW)*, pages 20–29, 2004.

[23] Moo Chung and Jonathan Taylor. Diffusion smoothing on brain surface via finite element method. In *Proceedings of the 2004 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 432–435, 2004.

[24] Ronan Collobert, Fabian H. Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[25] Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991.

[26] Vittorio Cristini, Jerzy Blawzdziewicz, and Michael Loewenberg. An adaptive mesh algorithm for evolving surfaces: simulation of drop breakup and coalescence. *J. Comput. Phys.*, 168(2):445–463, 2001.

[27] Mehmet M. Dalkilic and Edward L. Robertson. Information dependencies. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principals of Database Systems*, pages 245–253, Dallas, Texas, 2000.

[28] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[29] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, November 2000.

[30] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. Ranking the web frontier. In *Proceeding of the 13th World Wide Web Conference (WWW)*, pages 309–318, 2004.

[31] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[32] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[33] Dániel Fogaras and Balázs Rácz. Scaling link-based similarity search. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web (WWW)*, pages 641–650, 2005.

[34] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[35] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[36] Gunther Gediga and Ivo Duntsch. Rough approximation quality revisited. *Artificial Intelligence*, 132:219–234, 2001.

[37] Chris Giannella and Edward Robertson. On approximation measures for functional dependencies. *Information Systems*, 29(6):483–507, 2004.

[38] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, pages 576–587, 2004.

[39] Siegfried Handschuh, Steffen Staab, and Raphael Volz. On deep annotation. In *Proceeding of the 12th World Wide Web Conference (WWW)*, pages 431–438, 2003.

[40] G.H. Hardy, J.E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1952.

[41] Aboul-Ella Hassanien. Rough set approach for attribute reduction and rule generation: A case of patients with suspected breast cancer. *Journal of the American Society for Information Science and Technology*, 55(11):954–962, 2004.

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

[43] Shun ichi Amari. Integration of stochastic evidences by minimizing $\alpha-$divergence. *Submitted*, 2006.

[44] Hidehiko Ino, Mineichi Kudo, and Atsuyoshi Nakamura. Partitioning of web graphs by community topology. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web (WWW)*, pages 661–669, 2005.

[45] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[46] Jerzy W. Jaromczyk and Godfried T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80:1502–1517, 1992.

[47] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*, pages 538–543, 2002.

[48] Thorsten Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

[49] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[50] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–18, 1999.

[51] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pages 315–322, 2002.

[52] Marzena Kryszkiewicz. Rough set approach to incomplete information systems. *Information Sciences*, 112:39–49, 1998.

[53] Marzena Kryszkiewicz. Rules in incomplete information systems. *Information Sciences*, 113:271–292, 1999.

[54] S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, pages 639–650, 1999.

[55] S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks (Amsterdam, Netherlands)*, 31(11–16):1481–1493, 1999.

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

[57] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[58] D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9(3):219–242, 1980.

[59] Tony T. Lee. An information-theoretic analysis of relational databases part i: data dependencies and information metric. *IEEE Transactions on Software Engineering*, 13(10):1049–1061, 1987.

[60] Yee Leung and Deyu Li. Maximal consistent block technique for rule acquisition in incomplete information systems. *Information Sciences*, 153:85–106, 2003.

[61] Elizaveta Levina and Peter J. Bickel. Maximum likelihood estimation of intrinsic dimension. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS)*, 2004.

[62] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Twelfth International Conference on Information and Knowledge Management*, pages 556–559, 2003.

[63] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000.

[64] Charles X. Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. Decision trees with minimal costs. In Carla E. Brodley, editor, *roceedings of the Twenty-first International Conference on Machine Learning (ICML)*, 2004.

[65] Pawan Lingras and Yiyu Yao. Data mining using extensions of the rough set model. *Journal of the American Society for Information Science*, 49(5):415–422, 1998.

[66] Huan Liu, Farhad Hussain, Chew Lim Tan, and Manoranjan Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.

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

[68] F. Malvestuto. Statistical treatment of the information content of a database. *Information Systems*, 11(3):211–223, 1986.

[69] Deba Prasad Mandal and C. A. Murthy. Selection of alpha for alpha-hull in r^2. *Pattern Recognition*, 30(10):1759–1767, 1997.

[70] David J. Marchette. *Random Graphs for Statistical Pattern Recognition.* Wiley-Interscience, 2004.

[71] Klaus R. Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181 – 201, March 2001.

[72] K.K. Nambiar. Some analytic tools for the design of relational database systems. In *Proceedings of the Sixth International Conference on Very Large Databases*, pages 417–428, Montreal, Quebec, Canada, 2000.

[73] M. E. J. Newman. Scientific collaboration networks. I. Network construction and fundamental results. *Physical Review E*, 64(016131):1–8, 2001.

[74] Jens Nilsson, Fei Sha, and Michael I. Jordan. Regression on manifolds using kernel dimension reduction. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, Accepted, 2007.

[75] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of*

the 15th international conference on World Wide Web (WWW), pages 83–92, 2006.

[76] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry 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.

[77] Zdzislaw Pawlak. *Rough sets-theoretical aspects of reasoning about data.* Dordrecht: Kluwer Academic, 1991.

[78] Zdzislaw Pawlak. Rough classification. *International Journal of Human-Computer Studies*, 51:369–383, 1999.

[79] Zdzislaw Pawlak. Rough sets and intelligent data analysis. *Information Sciences*, 147:1–12, 2002.

[80] Zdzislaw Pawlak. Rough sets, decision algorithms and Bayes' theorem. *European Journal of Operational Research*, 136:181–189, 2002.

[81] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[82] J. Ross Quinlan. *C4.5: Programs for machine learning.* San Mateo: Morgan Kaufmann, 1993.

[83] J. Ross Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

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

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

[86] Salvatore Ruggieri. Efficient c4.5. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):438–444, 2002.

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

[88] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[89] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[90] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 824–831, 2005.

[91] Bei Tang, Guillermo Sapiro, and Vicent Caselles. Direction diffusion. In *International Conference on Computer Vision (ICCV)*, pages 1245–1252, 1999.

[92] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–2323, Dec 2000.

[93] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

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

[95] Pascal Vincent and Yoshua Bengio. Manifold parzen windows. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 825–832, 2003.

[96] Haixuan Yang, Irwin King, and Michael R. Lyu. NHDC and PHDC: Non-propagating and propagating heat diffusion classifiers. In *Proceedings of the 12th International Conference on Neural Information Processing (ICONIP)*, pages 394–399, 2005.

[97] Haixuan Yang, Irwin King, and Michael R. Lyu. Predictive ranking: a novel page ranking approach by estimating the web structure. In *Proceedings of the 14th international conference on World Wide Web (WWW) - Special interest tracks and posters*, pages 944–945, 2005.

[98] Haixuan Yang, Irwin King, and Michael R. Lyu. Predictive random graph ranking on the web. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, pages 3491–3498, 2006.

[99] Haixuan Yang, Irwin King, and Michael R. Lyu. Diffusionrank: A possible penicillin for web spamming. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[100] Haixuan Yang, Irwin King, and Michael R. Lyu. Heat diffusion classifiers on graphs. *Pattern Analysis and Applications*, Accepted, 2006.

[101] Haixuan Yang, Irwin King, and Michael R. Lyu. The generalized dependency degree between attributes. *Journal of the American Society for Information Science and Technology*, Accepted, 2007.

[102] Yiyu Yao. *Entropy Measures, Maximum Entropy and Emerging Applications*, chapter Information-theoretic measures for knowledge discovery and data mining, pages 115–136. Springer: Berlin, 2003.

[103] Yiyu Yao. Probabilistic approaches to rough sets. *Expert Systems*, 20(5):287–297, 2003.

[104] Dit-Yan Yeung and Hong Chang. A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks*, 18(1):141–149, 2007.

[105] Raymond W. Yeung. *A First Course in Information Theory.* Kluwer Academic Publishers, 2002.

[106] Jun Zhang and Vasant Honavar. Learning from attribute value taxonomies and partially specified instances. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 880–887, 2003.

[107] Wan Zhang and Irwin King. Locating support vectors via β-skeleton technique. In Kunihiko Fukushima Soo-Young Lee Lipo Wang, Jagath C. Rajapakse and Xi Yao, editors, *Proceedings to the International Conference on Neural Information Processing (ICONIP2002)*, pages 1423–1427, 2002.

[108] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS)*, 2004.

[109] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In Luc De Raedt and

Stefan Wrobel, editors, *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 1036–1043, 2005.

[110] Dengyong Zhou, Bernhard Schölkopf, and Thomas Hofmann. Semi-supervised learning on directed graphs. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS)*, 2004.

[111] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS)*, 2004.

[112] Xiaojun Zhu. *Semi-supervised learning with graphs.* PhD thesis, Carnegie Mellon University. CMU-LTI-05-192, 2005.

[113] Hans-Jürgen Zimmermann. *Fuzzy Set Theory and its Application.* Kluwer Academic Publishers, 2001.