Bayesian Decision Theoretical Framework for Clustering

CHEN, Mo

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
July 2011

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Abstract of thesis entitled:

Bayesian Decision Theoretical Framework for Clustering

Submitted by CHEN, Mo

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in July 2011

In this thesis, we establish a novel probabilistic framework for the data clustering problem from the perspective of Bayesian decision theory. The Bayesian decision theory view justifies the important questions: what is a cluster and what a clustering algorithm should optimize.

We prove that the spectral clustering (to be specific, the normalized cut) algorithm can be derived from this framework. Especially, it can be shown that the normalized cut is a non-parametric clustering method which adopts a kernel density estimator as its density model and tries to minimize the expected classification error or Bayes risk.

By the Bayesian decision theoretical view, we propose several extensions of current popular graph based methods. Several data-dependent graph construction approaches are proposed by adopting more flexible density estimators. The advantage of these approaches is that the parameters for constructing the graph can be estimated from the data. The constructed graph explores the intrinsic distribution of the data. As a result, the algorithm is more robust. It can obtain good performance constantly across different data sets. Using the flexible density models can result in directed graphs which cannot be handled by traditional graph partitioning algorithms. To tackle this problem,

we propose general algorithms for graph partitioning, which can deal with both undirected and directed graphs in a unified way.

在本论文中，我们从贝叶斯决策论的角度出发，对数据聚类问题建立了一个概率框架。从贝叶斯决策论的视角我们对以下重要问题作出了解释：什么是类簇，一个聚类算法应该对什么进行优化。

我们证明了谱聚类算法（具体地说正规化切算法）可以由所提出框架导出。特别的，可以证明，正规化切算法是一种使用核密度估计作为概率密度模型的，试图最小化期望分类错误率的非参数聚类方法。

从贝叶斯决策论出发，我们给出多种对现有基于图的聚类算法的扩展。多种使用了更加灵活的概率密度估计的数据驱动的建图方法被提出。这些方法的优势在于在建图过程中的所有参数由算法自动由数据中估计出来。所建的图考察了数据的内在分布，从而使算法更加鲁棒，对不同类型数据都能得到很好的性能。使用更加灵活的概率密度估计方法的结果是所建的图有可能为有向图。传统的谱聚类方法不能处理有向图。我们进一步提出了更为一般的图切分算法，使得我们可以统一的对有向图和无向图进行切分。

Acknowledgement

This thesis would not have been possible without the support of many individuals, to whom I would like to express my gratitude. First and foremost, I would like to thank my supervisor, Prof. Xiaoou Tang, for the opportunity of working with him, and for his continuous guidance, motivations, and encouragement in these years. I would also like to thank my co-supervisor, Prof. Jianzhuang Liu. He impressed me with the great patience and support to detail in every possible way. Finally, I would like to thank all members of mmlab for helpful discussion, collaboration and all kinds of assistance they provided.

This work is dedicated to my mother Lisa Feng.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Clustering is one of the most widely used techniques for exploratory data analysis. Across all disciplines, from social sciences over biology to computer science, people try to get a first intuition about their data by identifying meaningful groups among the data points. Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. In general, cluster analysis is the organization of a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into clusters based on certain goodness measurement. Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. The variety of techniques for representing data, measuring proximity (similarity) between data elements, and grouping data elements has produced a rich and often confusing assortment of clustering methods.

Clustering is useful in several exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification. However, in many such problems,

there is little prior information available about the data, and the decision-maker must make as few assumptions about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment (perhaps preliminary) of their structure.

The term "clustering" is used in several research communities to describe methods for grouping of unlabeled data. These communities have different terminologies and assumptions. It is important to understand the difference between clustering (unsupervised classification) and discriminant analysis (supervised classification). In supervised classification, we are provided with a collection of labeled (preclassified) patterns; the problem is to label a newly encountered, yet unlabeled, pattern. Typically, the given labeled (training) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern. Contrary to data analysis methods such as regression or classification for clustering there exists no ground truth. We wish to consider the common situation in which clustering takes place without having any significant prior knowledge about the subject data set. In the case of clustering, the problem is to group a given collection of unlabeled patterns into meaningful clusters. In a sense, labels are associated with clusters also, but these category labels are data driven: that is, they are obtained solely from the data. The very reason for performing clustering is that we want to discover a structure in the data which we did not know about before. If a clustering algorithm does not achieve good results we do not know whether the reason is that the algorithm performs poorly or whether there is just no group structure in our data.

In this thesis, we focus on graph based methods for data clustering (e.g. spectral clustering). These methods first construct a similarity graph from the given data, then partition

the graph into disjoint subgraphs to obtain the clustering result. Compared to the traditional algorithms such as k-means or single linkage, graph based methods has many fundamental advantages. Results obtained by these methods often outperform the traditional approaches, spectral clustering is very simple to implement and can be solved efficiently by standard linear algebra methods.

Spectral clustering goes back to [22], in which the authors first suggested to construct graph partitions based on eigenvectors of the adjacency matrix. In the same year, Fiedler [26] discovered that bi-partitions of a graph are closely connected with the second eigenvector of the graph Laplacian. and he suggested to use this eigenvector to partition a graph. Since then, spectral clustering has been discovered, re-discovered, and extended many times in different communities, see for example [64, 69, 9. 34, 40, 74, 3, 70, 33]. A nice overview over the history of spectral clustering can be found in [51]. In the machine learning community, spectral clustering has been made popular by the works of [67. 61, 54]. Subsequently, spectral clustering has been extended to many non-standard settings, for example spectral clustering applied to the co-clustering problem [19], spectral clustering with additional side information [42] connections between spectral clustering and the weighted kernel-k-means algorithm [20], learning similarity functions based on spectral clustering [2], or spectral clustering in a distributed environment [45]. Relations between spectral clustering and (kernel) principal component analysis rely on the fact that the smallest eigenvectors of graph Laplacians can also be interpreted as the largest eigenvectors of kernel matrices (Gram matrices). Two different flavors of this interpretation exist: The authors of [7] interpret the affinity matrix as a kernel matrix, the authors in [65] interpret the Moore-Penrose inverses of Laplacian matrix as a kernel matrix. Both interpretations can be used to construct

(different) out-of-sample extensions for spectral clustering. Concerning application cases of spectral clustering, in the last few years such a huge number of papers has been published in various scientific areas that it is impossible to cite all of them. We encourage the reader to query his favorite literature data base with the phrase "spectral clustering" to get an impression no the variety of applications.

The success of spectral clustering is mainly based on the fact that it does not make strong assumptions on the form of the clusters. As opposed to k-means, where the resulting clusters form convex sets (or, to be precise, lie in disjoint convex sets of the underlying space), spectral clustering can solve very general problems like intertwined spirals. Moreover, spectral clustering can be implemented efficiently even for large data sets, as long as we make sure that the similarity graph is sparse. Once the similarity graph is chosen, we just have to solve a linear problem, and there are no issues of getting stuck in local minima or restarting the algorithm for several times with different initializations. However, we have already mentioned that choosing a good similarity graph is not trivial, and spectral clustering can be quite unstable under different choices of the parameters for the neighborhood graphs. So spectral clustering cannot serve as a "black box algorithm" which automatically detects the correct clusters in any given data set. But it can be considered as a powerful tool which can produce good results if applied with care.

In the field of machine learning, graph Laplacians are not only used for clustering, but also emerge for many other tasks such as semi-supervised learning (e.g., [13] for an overview) or manifold reconstruction (e.g., [4]). In most applications, graph Laplacians are used to encode the assumption that data points which are close should have a similar label. One other way to interpret the use of graph Laplacian is by the smoothness as-

sumptions they encode. A function f which has a low value of $f^T Lf$ has the property that it varies only a little bit in regions where the data points lie dense (i.e., the graph is tightly connected), whereas it is allowed to vary more (e.g., to change the sign) in regions of low data density. In this sense, a small value of $f^T Lf$ encodes the so called cluster assumption in semi-supervised learning, which requests that the decision boundary of a classifier should lie in a region of low density. With this intuition one can use the quadratic form $f^T Lf$ as a regularizer in a transductive classification problem. An intuition often used is that graph Laplacians formally look like a continuous Laplace operator (and this is also where the name "graph Laplacian" comes from). This intuition has been made precise in the works [5, 48, 37, 38, 6, 36, 30]. In general, it is proved that graph Laplacians are discrete versions of certain continuous Laplace operators, and that if the graph Laplacian is constructed on a similarity graph of randomly sampled data points, then it converges to some continuous Laplace operator (or Laplace-Beltrami operator) on the underlying space. Belkin [5] studied the first important step of the convergence proof, which deals with the convergence of a continuous operator related to discrete graph Laplacians to the Laplace-Beltrami operator. His results were generalized from uniform distributions to general distributions by [48]. Then in [6], the authors prove pointwise convergence results for the unnormalized graph Laplacian using the Gaussian similarity function on manifolds with uniform distribution. At the same time, Hein et al. [37] prove more general results, taking into account all different graph Laplacians, more general similarity functions, and manifolds with arbitrary distributions. In [30], distributional and uniform convergence results are proved on manifolds with uniform distribution. The paper [36] studies the convergence of the smoothness functional induced by the graph Laplacians and shows uniform convergence

results.

Apart from applications of graph Laplacians to partitioning problems in the widest sense, graph Laplacians can also be used for completely different purposes, for example for graph drawing [47, 17]. In fact, there are many more tight connections between the topology and properties of graphs and the graph Laplacian matrices than we have mentioned in this tutorial. Now equipped with an understanding for the most basic properties, the interested reader is invited to further explore and enjoy the huge literature in this field on his own.

Despite this popularity of clustering, clustering is a difficult problem. Little is known about theoretical properties of clustering. One of the main reasons is that it is very difficult to evaluate the quality of a partition of some given data set. There is no general agreement for what criterion should be optimize in order to clustering general data. Differences assumptions and criteria are proposed in different communities to address the problem of clustering on the domain specific data sets. However these assumptions and criteria are mostly ad-hoc. There is no theoretical ground to justify what is the right way to do clustering. The lacking of a theoretical framework for clustering makes the transfer of useful generic concepts and methodologies slow to occur.

In this thesis, we adopt the probabilistic approach to model the clustering problem. Clustering can be viewed as assigning labels to samples in a given data set. This label assigning action is a decision making procedure under uncertainty. Probabilistic approach is a nature choice for modeling uncertainty. Bayesian decision theory which is built upon the probability theory is a theory that provides a theoretical ground for decision making problem. Bayesian decision theory deals with situations where decisions have to be made under a state of uncertainty, and its goal is to provide a rational framework for dealing with such

situations.

The decision theory for supervised learning problems is well established. Starting from this point, we first establish the decision theory for clustering problem. Then, we propose a novel probabilistic view of those graph based algorithms. In our framework, constructing the graph can be seen as a implicit way to model the density of the data using nonparametric density estimator. And various partition criteria (e.g. normalized cut) are somehow related to the Bayes risk, which the expected classification error is, given a partition. From this new perspective, we generalize the graph based clustering methods to use more general graph construction approaches using more flexible density estimators, which better explores the intrinsic distribution of the data. As a result, better clustering result can be obtained. . Using the flexible density models can result in directed graphs which cannot be handled by traditional graph partitioning algorithm. We propose general algorithms for graph partitioning which can deal with both undirected and directed graphs in a unified way.

The rest of the thesis is organized as follows:

- In Chapter 2, we review the probabilistic density modeling methods used in this thesis including: mixture models, nonparametric density model and Bayesian nonparametric models. We derive the Bayesian decision theory for clustering problems from the Bayesian decision theory for classification problems.

- In Chapter 3, we develop the Bayesian decision theoretical spectral clustering algorithms.

- In Chapter 4, we develop the isoperimetric cut algorithm for graph bipartitioning problem.

- In Chapter 5, we develop the random walk hitting time

based algorithm for multiway graph partitioning problem.

- Finally, in Chapter 6, we summarize the ideas of the thesis and point to directions of future work.

□ **End of chapter.**

Chapter 2

Bayesian Decision Theory for Clustering

Summary

In this chapter, we review the Bayesian decision theory for classification problem and the nonparametric technique for density modeling. We then derive the Bayesian decision theory for clustering problem which is the central theory the rest of the thesis is based on.

Bayesian decision theory is the fundamental statistical theory to the problem of decision making under uncertainty. This approach is based on quantifying the tradeoff between various decisions using probability and the costs that accompany such decisions. Here we briefly review the well established Bayesian decision theory for the classification problem [24, 8, 35, 28, 52]. Then we introduce the nonparametric density models for density estimation. Based on these models, we derive the Bayesian decision theory for the clustering problem.

2.1 Bayesian Decision Theory

Suppose we have an input sample vector $\mathbf{x}$, and our goal is to predict t given a new value for $\mathbf{x}$. For the classification problem, t represents class label which takes value from the set $\{\mathcal{C}_k\}_k$. The joint probability distribution $p(\mathbf{x}, t)$ provides a complete summary of the uncertainty associated with these variables. Determination of $p(\mathbf{x}, t)$ from a set of training data is an example of inference. In a practice, we must often make a specific prediction for the value of t, or more generally take a specific action based on our understanding of the values t is likely to take, and this aspect is the subject of decision theory.

We are interested in the probabilities of the classes given the sample, which are given by $p(t|\mathbf{x})$ Using Bayes' theorem, these probabilities can be expressed in the form

$$p(t|\mathbf{x}) = \frac{p(\mathbf{x}|t)p(t)}{p(\mathbf{x})},$$

where the evidence $p(\mathbf{x})$ is given by

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k).$$

Note that any of the quantities appearing in Bayes' theorem can be obtained from the joint distribution $p(\mathbf{x}, t)$ by either marginalizing or conditioning with respect to the appropriate variables. We can now interpret $p(t)$ as the prior probability for the class k, and $p(t|\mathbf{x})$ as the corresponding posterior probability. If our goal is to minimize the chance of assigning $\mathbf{x}$ to the wrong class, then we should choose the class having the higher posterior probability. We now justify this approach, and we also discuss more general criteria for making decisions.

2.1.1 Bayes Decision Rule

Suppose that our goal is simply to make as few misclassifications as possible. The probability of misclassification rate is given by

$$p(\lambda) = \int p(\lambda, \mathbf{x}) d\mathbf{x} = \int p(\lambda|\mathbf{x}) p(\mathbf{x}) d\mathbf{x},$$

where $p(\lambda)$ is the probability that a error occurs and $p(\lambda|\mathbf{x})$ is the probability that a sample $\mathbf{x}$ is misclassified. If we ensure that $p(\lambda|\mathbf{x})$ is small as possible for every $\mathbf{x}$, the integral must be as small as possible.

We need a rule that assigns each value of $\mathbf{x}$ to one of the available classes. Such a rule will divide the input space into regions R_k called decision regions, one for each class, such that all points in R_k are assigned to class $\mathcal{C}_k$. The boundaries between decision regions are called decision boundaries or decision surfaces. Note that each decision region need not be contiguous but could comprise some number of disjoint regions. A mistake occurs when an input vector belonging to class $\mathcal{C}_k$ is assigned to classes other than $\mathcal{C}_k$ which are denoted by $\overline{\mathcal{C}}_k$. The probability of this occurring is given by

$$\begin{aligned} p(\lambda) &= \sum_k P(\mathbf{x} \in \mathcal{R}_k, \overline{\mathcal{C}}_k) = \sum_k \int_{\mathcal{R}_k} p(\mathbf{x}, \overline{\mathcal{C}}_k) d\mathbf{x} \\ &= \sum_k \int_{\mathcal{R}_k} p(\overline{\mathcal{C}}_k|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \sum_k \int_{\mathcal{R}_k} \left(1 - p(\mathcal{C}_k|\mathbf{x})\right) p(\mathbf{x}) d\mathbf{x}, \end{aligned} \tag{2.1}$$

which is called Bayes error.

We are free to choose the decision rule that assigns each point $\mathbf{x}$ to one of the classes. Clearly to minimize $p(\lambda)$ we should arrange that each $\mathbf{x}$ is assigned to whichever class has the smallest value of the integrand in (2.1). Thus, for a given value of $\mathbf{x}$ we should assign it to class $\mathcal{C}_k$ if $p(\mathbf{x}, \mathcal{C}_k)$ has the largest value. From

the product rule of probability we have $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$. Because the factor $p(\mathbf{x})$ is common to all terms, we can restate this result as saying that the minimum probability of making a mistake is obtained if each value of $\mathbf{x}$ is assigned to the class for which the posterior probability $p(\mathcal{C}_k|\mathbf{x})$ is largest.

Therefore, the rule that minimize the probability of misclassification rate is

$$\mathcal{C}^* = \text{argmax}[p(\mathcal{C}_k|\mathbf{x})], \tag{2.2}$$

which is called Bayes decision rule. Under this decision rule, the expected error is

$$p(\lambda|\mathbf{x}) = \min[p(\mathcal{C}_k|\mathbf{x})].$$

It is easy to see that minimizing $p(\lambda)$ is equivalent to maximizing the probability of being correct which is written as

$$\begin{aligned} p(\gamma) &= \sum_k P(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) - \sum_k \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x} \\ &= \sum_k \int_{\mathcal{R}_k} p(\mathcal{C}_k|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

Since the equation $p(\gamma) + p(\lambda) = 1$ always holds, the the decision rule (2.2) also maximizes the probability of being correct.

2.1.2 Bayes Risk

For a general treatment, we introduce a loss function, also called a cost function $\lambda(t, f(\mathbf{x}))$, which is a single, overall measure of loss incurred in taking any of the available decisions or actions $f(\cdot)$ when the state of nature is t. Our goal is then to minimize the total loss incurred

$$E(\lambda) = \int \lambda(t, f(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt.$$

In the classification problem, both t and $f(\mathbf{x})$ are taking discrete values. Therefore, the value of λ is taken from a matrix, which is called loss matrix.

Suppose that, for a new value of $\mathbf{x}$, the true class is $\mathcal{C}_k$ and that we assign $\mathbf{x}$ to class $\mathcal{C}_j$ (where j may or may not be equal to k). By doing so, we incur some level of loss that we denote by λ_{kj}, which we can view as the k, j element of a loss matrix.

The optimal solution is the one which minimizes the loss function. However, the loss function depends on the true class, which is unknown. For a given input vector $\mathbf{x}$, our uncertainty in the true class is expressed through the joint probability distribution $p(\mathbf{x}, \mathcal{C}_k)$ and so we seek instead to minimize the average loss, where the average is computed with respect to this distribution, which is given by

$$E(\lambda) = \sum_k \sum_j \int_{\mathcal{R}_j} \lambda_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}. \tag{2.3}$$

Each $\mathbf{x}$ can be assigned independently to one of the decision regions R_j . Our goal is to choose the regions R_j in order to minimize the expected loss (2.3), which implies that for each $\mathbf{x}$ we should minimize $\sum_k \lambda_{kj} p(\mathbf{x}, \mathcal{C}_k)$. As before, we can use the product rule $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$ to eliminate the common factor of $p(\mathbf{x})$. Thus the decision rule that minimizes the expected loss is the one that assigns each new $\mathbf{x}$ to the class j for which the quantity

$$\sum_k \lambda_{kj} p(\mathcal{C}_k|\mathbf{x})$$

is a minimum. This is clearly trivial to do, once we know the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$.

In the classification problem, on special loss function which is of particular interest is the zero-one loss function. It has

elements $\lambda_{kj} = 1 - \gamma_{kj}$ where

$$\gamma_{kj} = \begin{cases} 1 & k = j \\ 0 & k \neq j. \end{cases}$$

The risk corresponding to this loss function is, again, the expected misclassification rate.

We classify a sample as belonging to class $\mathcal{C}_j$ if

$$\sum_k \lambda_{kj} p(\mathcal{C}_k|\mathbf{x}) - \sum_k (1 - \gamma_{kj}) p(\mathcal{C}_k|\mathbf{x}) - 1 - \sum_k \gamma_{kj} p(\mathcal{C}_k|\mathbf{x})$$

is a minimum. This suggest that we should choose class $\mathcal{C}_j$ for which

$$1 - p(\mathcal{C}_j|\mathbf{x})$$

is the smallest. Therefore, Minimizing the expect loss will minimize the misclassification rate.

2.1.3 Generative VS. Discriminative

We have broken the classification problem down into two separate stages, the inference stage in which we use training data to learn a model for $p(\mathcal{C}_k|\mathbf{x})$, and the subsequent decision stage in which we use these posterior probabilities to make optimal class assignments.

For the classification problem, the key idea is to model the posterior for each class. Then classifying samples is simply to apply the Bayes decision rule. There are two distinct ways to accomplish this goal called discriminative and generative approaches.

In generative approach, we first determine the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ for each class $\mathcal{C}_k$ individually. Also separately infer the prior class probabilities $p(\mathcal{C}_k)$. Then use Bayes' theorem to find the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$. Equivalently, we can model the joint distribution $p(\mathbf{x}, t)$ directly and then

normalize to obtain the posterior probabilities. Generative approach explicitly or implicitly models the distribution of inputs as well as outputs, because by sampling from them it is possible to generate synthetic data points in the input space.

In discriminative approach, we find a function $f(\mathbf{x})$, called a discriminant function, which maps each input $\mathbf{x}$ directly onto a class label. Actually, the decision rule

$$\mathcal{C}^* = \text{argmax}[p(\mathcal{C}_k|\mathbf{x})]$$

is exactly such a function. Therefore, the discriminative approach can be viewed as directly modeling the posterior $p(\mathcal{C}_k|\mathbf{x})$ without modeling the generative density $p(\mathbf{x}|\mathcal{C}_k)$.

The decision function can be obtained by minimizing following risk

$$E(\lambda|\mathcal{C}_k) = \int_{\mathcal{X}} \lambda(\mathcal{C}_k, f(\mathbf{x}))p(\mathbf{x}|\mathcal{C}_k)d\mathbf{x}$$

or for discrete problem

$$E(\lambda|\mathcal{C}_k) = \sum_{\mathbf{x}\subset\mathcal{X}} \lambda(\mathcal{C}_k, f(\mathbf{x}))p(\mathbf{x}|\mathcal{C}_k).$$

2.2 Nonparametric Density Models

As non-parametric methods make fewer assumptions, their applicability is much wider than the corresponding parametric methods [41, 68]. In particular, they may be applied in situations where less is known about the application in question. Here we give brief introduction to the nonparametric models which will be used in this thesis.

Let us suppose that observations are drawn from certain unknown distribution with the probability density function $p(\mathbf{x})$ in certain d-dimensional space, which we shall take to be Euclidean. We wish to estimate $p(\mathbf{x})$. Let us consider some small

region $\mathcal{R}$ containing $\mathbf{x}$. The probability mass associated with this region is given by

$$\int_{\mathcal{R}} p(\mathbf{x})d\mathbf{x}.$$

Suppose that we have collected a data set comprising n observations drawn from $p(\mathbf{x})$. Because each data point has a probability P of falling within $\mathcal{R}$, the total number K of points that lie inside $\mathcal{R}$ will be distributed according to the binomial distribution

$$\text{Bin}(K|n, P) = \frac{n!}{K!(n \quad K)!} P^K (1 - P)^{n \quad K}.$$

The mean of points falling inside the region is $E[K/n] = P$, and the variance around this mean is $\text{var}[K/n] = P(1 - P)/n$. For large n, this distribution will be sharply peaked around the mean and so

$$K \simeq nP. \tag{2.4}$$

If, however, we also assume that the region $\mathcal{R}$ is sufficiently small that the probability density $p(\mathbf{x})$ is roughly constant over the region, then we have

$$P \simeq p(\mathbf{x})V, \tag{2.5}$$

where V is the volume of $\mathcal{R}$. Combining (2.4) and (4.3), we obtain our density estimate in the form

$$p(\mathbf{x}) - \frac{K}{nV}. \tag{2.6}$$

We can exploit the result (2.6) in two different ways. Either we can fix K and determine the value of V from the data, which gives rise to the K-nearest-neighbour technique discussed shortly, or we can fix V and determine K from the data, giving rise to the kernel approach. It can be shown that both the K-nearest-neighbour density estimator and the kernel density estimator

converge to the true probability density in the limit as long as following conditions hold

$$\lim_{n\to\infty} V = 0, \qquad \lim_{n\to\infty} K = \infty. \qquad \lim_{n\to\infty} K/n = 0.$$

2.2.1 Kernel Density Estimator

In statistics, kernel density estimator (KDE) is a nonparametric way of estimating the probability density function of a random variable. We begin by discussing the kernel method in detail, and to start with we take the region $\mathcal{R}$ to be a small hypercube centered on the point $\mathbf{x}$ at which we wish to determine the probability density. In order to count the number K of points falling within this region, it is convenient to define the following function

$$\kappa(\mathbf{u}) = \begin{cases} 1 & |u_i| \leq 1/2, i = 1, \ldots, d, \\ 0 & \text{otherwise}, \end{cases} \tag{2.7}$$

which represents a unit cube centered on the origin. The function $\kappa(\mathbf{u})$ is an example of a kernel function, and in this context is also called a Parzen window. From (2.7), the quantity $\kappa((\mathbf{x} - \mathbf{x}_i)/h)$ will be one if the data point $\mathbf{x}_i$ lies inside a cube of side h centered on $\mathbf{x}$, and zero otherwise. The total number of data points lying inside this cube will therefore be

$$K = \sum_{j=1}^{n} \kappa(\frac{\mathbf{x} - \mathbf{x}_j}{h}).$$

Substituting this expression into (2.6) then gives the following result for the estimated density at $\mathbf{x}$

$$p(\mathbf{x}) - \frac{1}{n} \sum_{j-1}^{n} \frac{1}{h^d} \kappa(\frac{\mathbf{x} - \mathbf{x}_j}{h}), \tag{2.8}$$

where we have used $V = h^d$ for the volume of a hypercube of side h in d dimensions. Using the symmetry of the function $\kappa(\mathbf{u})$, we

can now re-interpret this equation, not as a single cube centered on $\mathbf{x}$ but as the sum over n cubes centered on the n data points $\mathbf{x}_i$. We can choose a smoother kernel function, and a common choice is the Gaussian kernel

$$\kappa(\frac{\mathbf{x}-\mathbf{y}}{h}) = \frac{1}{(\sqrt{2\pi})^d} \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2h^2}),$$

which gives rise to kernel density estimator

$$p(\mathbf{x}) = \frac{1}{n(h\sqrt{2\pi})^d} \sum_{j=1}^{n} \exp(-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{2h^2}), \tag{2.9}$$

where h represents the standard deviation of the Gaussian components. Thus our density model is obtained by placing a Gaussian over each data point and then adding up the contributions over the whole data set, and then dividing by n so that the density is correctly normalized.

The parameter h plays the role of a smoothing parameter, and there is a trade-off between sensitivity to noise at small h and over-smoothing at large h. Again, the optimization of h is a problem in model complexity.

We can choose any other kernel function $\kappa(\mathbf{u})$ in (2.8) subject to the conditions

$$\kappa(\mathbf{u}) \geq 0$$
$$\int \kappa(\mathbf{u}) d\mathbf{u} = 1,$$

which ensure that the resulting probability distribution is non-negative everywhere and integrates to one.

Here, we show how to using kernel density estimator to solve classification problem. For a multi-class classification problem, the estimate of the density associated with each class is

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{n_k(h\sqrt{2\pi})^d} \sum_{j \in V_k} \exp(-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{2h^2}),$$

where d is the dimension of the input samples and n_k is the number of samples in class k. We assume that the prior distribution of t is given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}.$$

We can use the Bayes' theorem to obtain the posterior probability of the class membership

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{\sum_{j\in V_k} \kappa((\mathbf{x}-\mathbf{x}_j)/h)}{\sum_{j=1}^{n} \kappa((\mathbf{x}-\mathbf{x}_j)/h)}.$$

If we wish to minimize the probability of misclassification, this is done by assigning the test point $\mathbf{x}$ to the class having the largest posterior probability.

2.2.2 Nearest Neighbors Density Estimator

We return to our general result (2.6) for local density estimation, and instead of fixing V and determining the value of K from the data, we consider a fixed value of K and use the data to find an appropriate value for V. To do this, we consider a small sphere centered on the point $\mathbf{x}$ at which we wish to estimate the density $p(\mathbf{x})$, and we allow the radius of the sphere to grow until it contains precisely K data points. The estimate of the density $p(\mathbf{x})$ is then given by (2.6) with V set to the volume of the resulting sphere. This technique is known as K nearest neighbors (KNN). The value of K now governs the degree of smoothing and that again there is an optimum choice for K that is neither too large nor too small.

The KNN technique for density estimation can be extended to the problem of classification. To do this, we apply the KNN density estimator to each class separately and then make use of Bayes' theorem. Let us suppose that we have a data set comprising n_k points in class $\mathcal{C}_k$ with n points in total, so that $\sum_k n_k = n$. If we wish to classify a new point $\mathbf{x}$, we draw a

sphere centered on $\mathbf{x}$ containing precisely K points irrespective of their class. Suppose this sphere has volume V and contains K_k points from class $\mathcal{C}_k$. Then (2.6) provides an estimate of the density associated with each class

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{n_k V}. \tag{2.10}$$

Similarly, the unconditional density is given by

$$p(\mathbf{x}) = \frac{K}{nV}, \tag{2.11}$$

while the class priors are given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}.$$

We can use the Bayes' theorem to obtain the posterior probability of the class membership

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}.$$

If we wish to minimize the probability of misclassification, this is done by assigning the test point $\mathbf{x}$ to the class which has the largest value of K_k/K. Thus to classify a new point, we identify the K nearest points from the training data set and then assign the new point to the class having the largest number of representatives amongst this set.

Although the nonparametric density estimator (both KDE and KNN) are suffered from the curse of dimensionality (due to V), using them to do classification is not such a bad idea. The posterior is normalized by the evidence which makes V vanish. Therefore, the convergence rate of the classifier is not exponential any more.

2.2.3 An Alternative Mixture Model View

Assume that the nonparametric density estimator has the general form

$$p(\mathbf{x}) = p(\mathbf{x}|\mathcal{X}) = \frac{1}{n}\sum_{i=1}^{n} p(\mathbf{x}|\mathbf{x}_i),$$

where the data set X is viewed as a set of parameter of the density model. For example the Gaussian kernel density estimator

$$p(x) = \frac{1}{n(\sigma\sqrt{2\pi})^d}\sum_{j-1}^{n} \exp(-\frac{\|x - x_j\|^2}{2\sigma^2})$$

can be seen as a mixture model [53] with n component, each component $p(\mathbf{x}|\mathbf{x}_i)$ is a Gaussian distribution with mean $\mathbf{x}_i$. These n components can be further grouped to K components as

$$p(\mathbf{x}|\mathcal{X}) = \sum_{k} p(\mathbf{x}|\mathcal{X}_k, \mathcal{C}_k)p(\mathcal{C}_k),$$

where the conditional density for each component is given by

$$p(\mathbf{x}|\mathcal{X}_k, \mathcal{C}_k) = \frac{1}{n_k}\sum_{j\in\mathcal{V}_k} p(\mathbf{x}|\mathbf{x}_j)$$

and the prior is given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}.$$

2.3 Bayesian Decision Theory for Clustering

Here, we try to formulate the Bayesian decision theory for the clustering problem from the well established decision theory for the classification problem.

The basic assumption made in this thesis is that the samples in the data set are independent and identically distributed according certain underlying probabilistic distribution. The density of this distribution is $p(\mathbf{x})$. Intuitively, the distribution has

to be a mixture model, i.e., $p(\mathbf{x}) = \sum_k \pi_k p(\mathbf{x}|\theta_k)$. If it is not, there would not be any cluster structure in the data. For example, if the samples of the data set are drawn from a Gaussian distribution, there is no clear boundary to separate the data into disjoint groups. As a result, there is no point to perform clustering on such a data set. We call those distributions which can not be represented as a mixture model as atom distributions.

2.3.1 Bayes Error for Clustering

To formalize the intuition, we assume that the samples and their labels follow the joint distribution $p(\mathbf{x}, t)$, where t is a discrete random variable taking values from set $\{\mathcal{C}_k\}_k$. Assume there are K intrinsic clusters in the data. Then marginalizing the random variable t we have

$$p(\mathbf{x}) = \sum_t p(\mathbf{x}|t)p(t) = \sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k), \tag{2.12}$$

where $p(\mathcal{C}_k) = \pi_k$, which is a mixture model.

If we already have the density model (2.12), as discussed previously, the optimal partitioning that minimize the Bayes error is to assign each sample $\mathbf{x}$ with label

$$\mathcal{C}^* = \text{argmax}[p(\mathcal{C}_k|\mathbf{x})].$$

This is indeed the case for clustering data using Gaussian mixture model. In the Gaussian mixture model, the density function is obtained by fitting the parameters to maximize the likelihood. Then, the clustering result can be obtained by assign each sample $\mathbf{x}$ to the cluster with maximal posterior. The second step minimizes the Bayes error given that the density model is already known.

However, this approach makes very strong assumption. It assumes that the data can be modeled by a mixture of K atom

distributions and the number of intrinsic clusters is the number of atom components. However, the assumption made here is too strong to be practical. Most data sets we are dealing with are not so simple. They usually have more complex structures.

Fortunately, a wonderful result we can utilize is that any data (no matter what intrinsic distribution they are from) can be approximately modeled by mixture models. We denote the mixture model as

$$p(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j p(\mathbf{x}|\theta_j), \tag{2.13}$$

where $m \leq n$ and each $p(\mathbf{x}|\theta_j)$ is an atom distribution. However, the intrinsic number of clusters K in general is not equal to m. We have $K \leq m \leq n$. Then the clustering problem becomes grouping or partitioning the m atom distributions in (2.13) to K supper components in (2.12) where the conditional density of each supper component is

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_{j \in \mathcal{V}_k} \frac{\alpha_j}{\pi_k} p(\mathbf{x}|\theta_j),$$

where $\pi_k = \sum_{j \subset \mathcal{V}_k} \alpha_j$. The joint distribution is

$$p(\mathbf{x}, \mathcal{C}_k) = p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) = \sum_{j \in \mathcal{V}_k} \alpha_j p(\mathbf{x}|\theta_j).$$

We denote the partitioning as $\Omega = \{\mathcal{V}_k\}_k$, where $\mathcal{V}_k$ is the index set for those atom distributions which are grouped into the kth component. This grouping step does not change the density model of the data set. We can see that

$$\begin{aligned} p(\mathbf{x}) &= \sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}|\mathcal{C}_k) \\ &= \sum_{k=1}^{K} \pi_k \sum_{j \in \mathcal{V}_k} \frac{\alpha_j}{\pi_k} p(\mathbf{x}|\theta_j) = \sum_{j=1}^{m} \alpha_j p(\mathbf{x}|\theta_j). \end{aligned}$$

After the grouping the m atom distributions into K components, we can treat the K components as classifier of a K class classification problem. The classification function is simply the decision rule

$$f(\mathbf{x}) = \text{argmax}[p(\mathcal{C}_k|\mathbf{x})].$$

where the posterior is given by

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{k=1}^{K} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)} = \frac{\sum_{j\in\mathcal{V}_k} \alpha_j p(\mathbf{x}|\theta_j)}{\sum_{j=1}^{m} \alpha_j p(\mathbf{x}|\theta_j)}.$$

The expected loss (Bayes risk) with respect to the partitioning Ω is

$$E_\Omega(\lambda) - \sum_t \int \lambda(t, f(\mathbf{x}))p(\mathbf{x}, t)d\mathbf{x}.$$

If zero-one loss is used, the expected loss is equal to Bayes error.

Then we can formulate the clustering problem as finding the optimal partitioning $\Omega = \{\mathcal{V}_k\}_k$ which minimizes the Bayes risk. In other words, we should solve the following optimization problem

$$\Omega^* = \underset{\Omega}{\text{argmin}} \; E_\Omega(\lambda).$$

However, the optimization problem is ill posed. The integral over the whole sample space is intractable to evaluate. Even we are able to evaluate the integral, the optimization over all possible partitioning is a hard combinatorial problem which is NP-complete. In order to have a practical algorithm which utilizes the decision theory, various approximations have to be made to derive a tractable model. For example, we can approximate the integral by the empirical summation over samples. It is also possible to derive upper bounds for the Bayes risk which are easy to be optimized which is similar to the SVM in the classification problem.

In the following chapters, we will utilize the Bayes clustering theory derived above and use different approximation methods

and different density models to develop practical algorithms for data clustering.

□ **End of chapter.**

Chapter 3

Decision Theoretic Spectral Clustering

Summary

In this chapter, we propose a novel probabilistic view of the spectral clustering algorithm. In our framework, the spectral clustering algorithm can be seen as assigning optimal class labels to samples that minimizes the nonparametric kernel density estimation based Bayes error. From this perspective, we obtain an insight of how to construct a graph in order to make spectral clustering performances better. We generalize the spectral clustering to using more general graph construction methods, which may result in directed graphs. In order to cluster the vertices of the directed graph, we propose a directed graph partitioning algorithm. The partitioning result can be efficiently obtained by eigenvalue decomposition.

3.1 Introduction

Spectral clustering [12, 54, 61, 67, 81, 44] is graph based and widely used for general data clustering. Compared to the traditional algorithms such as k-means or single linkage, spectral clustering has many fundamental advantages [78, 77]. It does not make strong assumptions about the global structure of the data. Instead, local similarities between local sample points are considered and a global decision is then made to divide all data points into disjoint sets according to some criterion. Therefore, these methods can potentially deal with data sets whose clusters are of irregular shapes. Results obtained by spectral clustering often outperform the traditional approaches. Spectral clustering is also very simple to implement and can be solved efficiently by standard linear algebra methods.

Given a set of data points $\{\mathbf{x}_i\}_{i=1,\ldots,n}$, an intuitive goal of clustering is to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar to each other. A nice way of representing the data is in form of the similarity graph $G = (\mathcal{V}, \mathcal{E})$. Each vertex v_i in this graph represents a data point $\mathbf{x}_i$. Two vertices are connected by an edge with weight w_{ij}. Here the edge weight represents the similarity between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that the similarity is a symmetric measurement, i.e., $w_{ij} = w_{ji}$. Therefore the similarity graph G is an undirected graph. The problem of clustering can now be reformulated using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other). A meta algorithm of spectral clustering is shown in 1. To be able to formalize this intuition we first introduce some

Algorithm 1 Meta algorithm of spectral clustering

1. Construct a graph G from the data $X = [\mathbf{x}_i]_{i=1,..,n}$
2. Obtain K eigenvectors of certain matrix of the graph G
3. Discretize the eigenvectors to obtain the cluster membership indicators

basic graph notation and briefly discuss the kind of graphs we are going to study.

3.2 Spectral Clustering

3.3 Graph Notation

Spectral clustering is a graph based clustering algorithm which first builds a graph G from a given data set $X = \{\mathbf{x}_i\}_{i=1,...,n}$, and then partitions the graph into disjoint subgraphs to obtain the clustering result.

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertex set $\mathcal{V} = \{v_i\}_{i=1,..,n}$. In the following we assume that the graph G is weighted, that is each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. The weighted adjacency matrix of the graph is the matrix $W = [w_{ij}]_{i,j=1,...,n}$. If $w_{ij} = 0$, it means that the vertices v_i and v_j are not connected by an edge. If G is an undirected, we require $w_{ij} = w_{ji}$. For a directed graph the edge weights w_{ij} and w_{ji} might be different. The degree of a vertex $v_i \in \mathcal{V}$ is defined as

$$d_i = \sum_j w_{ij}.$$

The degree vector is defined as $\mathbf{d} = [d_i]^T_{i=1,...,n}$. The degree matrix D is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal. The graph (unnormalized) Laplacian matrix is defined as

$$L = D - W.$$

It can be proven that for every vector $\mathbf{y} \in \mathcal{R}^n$ we have

$$\mathbf{y}^T L \mathbf{y} = \frac{1}{2} \sum_{i,j} w_{ij} (y_i - y_j)^2.$$

Therefore L is symmetric and positive semi-definite matrix. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbf{1}$. Note that the unnormalized graph Laplacian does not depend on the diagonal elements of the adjacency matrix W. Each adjacency matrix which coincides with W on all off diagonal positions leads to the same unnormalized graph Laplacian L. In particular, self-edges in a graph do not change the corresponding graph Laplacian. The graph Laplacian and its eigenvalues and eigenvectors can be used to describe many properties of graphs, see [56, 57].

Given a subset of vertices $\mathcal{A} \subset \mathcal{V}$, we denote its complement $\mathcal{V} \backslash \mathcal{A}$ by $\overline{\mathcal{A}}$. For convenience we introduce the shorthand notation $i \in \mathcal{A}$ for the set of indices $\{i | v_i \in \mathcal{A}\}$. For two not necessarily disjoint sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$ we define

$$W(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}, j \subset \mathcal{B}} w_{ij}.$$

In this paper, we use $\mathbf{x}_i \in X$ to denote an input sample vector and $i \in \mathcal{V}$ to denote the corresponding vertex on the graph. $X_k \subset X$ denotes a subset of samples of X and $\mathcal{V}_k \subset \mathcal{V}$ denotes the subset of vertices corresponding to X_k. w_{ij} is the weight of the edge from vertices i to j. A multi-class partitioning is to partition the vertex set into K disjoint subsets such that $\sqcup_{k=1}^{K} \mathcal{V}_k = \mathcal{V}$, which means $\mathcal{V}_i \bigcap \mathcal{V}_j = \emptyset$ and $\bigcup_{k=1}^{K} \mathcal{V}_k = \mathcal{V}$. The corresponding samples are then clustered as $\sqcup_{k=1}^{K} X_k = X$. We denote the K-way partitioning as $\Gamma = \{\mathcal{V}_k\}_{k=1, \ldots, K}$. Without ambiguity, sometimes we also denote the partitioning as $\Gamma = \{X_k\}_{k=1, \ldots, K}$.

3.3.1 Graph Construction

The first step of spectral clustering is to construct a graph from the input vector form data. There are several popular constructions to transform a given set $\{\mathbf{x}_i\}_{i=1,\ldots,n}$ of data points into a graph. When constructing similarity graphs the goal is to model the local neighborhood relationships between the data points. Here we consider three widely used methods for construction graph.

The fully connected graph: Here we simply connect all points with positive similarity with each other. As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the Gaussian similarity function $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$, where the parameter controls the width of the neighborhoods. In this method, there is a free parameter σ need to be set by the user.

The neighborhood binary graphs: Here the goal is to connect vertex v_i with vertex v_j if v_j is among the neighbors of v_i. Either the K-nearest-neighbor (KNN) or ϵ-neighborhood can be used to determine the neighborhood. However, the K-nearest-neighbor method can lead to a directed graph, as the neighborhood relationship is not symmetric. There are two ways of making this graph undirected. The first way is to simply ignore the directions of the edges, that is we connect v_i and v_j with an undirected edge if v_i is among the K nearest neighbors of v_j or if v_j is among the K nearest neighbors of v_i. The resulting graph is what is usually called the k-nearest-neighbor graph. The second choice is to connect vertices v_i and v_j if both v_i is among the K nearest neighbors of v_j and v_j is among the K nearest neighbors of v_i. The resulting graph is called the mutual K-nearest-neighbor graph. After connecting the appropriate vertices we simply set the edge weights to be 1. Later we will set

for a more general formulation of spectral clustering, the symmetry condition is not necessary. In this method, there is a free parameter for the neighborhood (K or ϵ) need to be set by the user.

The neighborhood weighted graph: Here the graph structure is constructed in the same way as The neighborhood binary graphs except that the edge weight is set to $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$. This method can be seen as a combination of the first two methods. Some time it leads to better results than others. However, in this method, there are two free parameters need to be set by the user.

All graphs mentioned above are regularly used in spectral clustering. To our knowledge, theoretical results on the question how the choice of the similarity graph influences the spectral clustering result do not exist. In this paper, we will show that how these methods affect the results of spectral clustering.

3.3.2 Multi-class Normalized Cut

After obtaining a graph, the problem of clustering is transformed to a graph partitioning problem. The normalized cut bi-partitioning algorithm is first proposed in [67], which is later generalized to multi-class normalized cut by [81]. The multi-class normalized cut algorithm is to minimize following criterion

$$\mathrm{nc}(\Gamma) = \frac{1}{K}\sum_{k=1}^{K}\frac{W(\mathcal{V}_k, \overline{\mathcal{V}}_k)}{W(\mathcal{V}_k, \mathcal{V})}. \tag{3.1}$$

We also define the normalized association criterion as

$$\mathrm{na}(\Gamma) = \frac{1}{K}\sum_{k=1}^{K}\frac{W(\mathcal{V}_k, \mathcal{V}_k)}{W(\mathcal{V}_k, \mathcal{V})}. \tag{3.2}$$

Apparently, where $\mathrm{nc}(\Gamma) + \mathrm{nc}(\Gamma) = 1$. Therefore minimizing the normalized cut criterion is equivalent to maximizing the normalized association criterion.

3.3.3 Relaxation Optimization

Let $\mathbf{z}_k \in \{0,1\}^n$ be the binary indicator vector for the kth cluster and $Z = [\mathbf{z}_k]_{k=1,\ ,K}$ be the indicator matrix. The normalized cut criterion can be rewritten as

$$\mathrm{nc}(Z) - \frac{1}{K}\sum_{k=1}^{K}\frac{\mathbf{z}_k^T L \mathbf{z}_k}{\mathbf{z}_k^T D \mathbf{z}_k},$$

Therefore normalized cut problem can be formulated as following combinatory optimization problem

$$\begin{aligned} \min_{\mathbf{z}_1,\ ,\mathbf{z}_K} \quad & \frac{1}{K}\sum_{k=1}^{K}\frac{\mathbf{z}_k^T L \mathbf{z}_k}{\mathbf{z}_k^T D \mathbf{z}_k} \\ \text{s.t.} \quad & \mathbf{z}_k \in \{0,1\}^n,\ k = 1,\ldots,K. \end{aligned} \tag{3.3}$$

Unfortunately, solving for the exact solution of this discrete optimization problem is NP-hard [80]. Instead of solving (3.3) directly, we try to solve a relaxed problem of (3.3).

Define the scaled partition matrix [12, 81]

$$F = Z(Z^T D Z)^{-1/2},$$

Since $Z^T D Z$ is a diagonal matrix, the columns of F are the columns of Z scaled by the inverse square root of the degree. Clearly we have

$$F^T D F = (Z^T D Z)^{-1/2} Z^T D Z (Z^T D Z)^{-1/2} = I. \tag{3.4}$$

Given a scaled partition matrix F, we can restore the corresponding Z by

$$Z = \mathrm{Dg}(\mathrm{dg}^{1/2}(FF^T))F, \tag{3.5}$$

where $M = \mathrm{Dg}(\mathbf{v})$ denotes constructing a diagonal matrix M from the vector $\mathbf{v}$ and $\mathbf{v} = \mathrm{dg}(M)$ denotes extracting the diagonal elements of matrix M to form a vector $\mathbf{v}$.

Substituting F to (3.3) and relaxing F to take real values that satisfy the constraints $F^TDF - I$. we have a convex optimization problem as:

$$\begin{aligned} \min_{F} \quad & \frac{1}{K}\mathrm{tr}(F^TLZ) \\ \text{s.t.} \quad & F^TDF = I. \end{aligned} \tag{3.6}$$

By the Rayleigh-Ritz theorem [50] it can be seen immediately that the solution of this problem is given by the vectors F^* achieved by solving the following generalized eigen-decomposition problem

$$LZ^* = DF^*\Lambda, \tag{3.7}$$

where $\Lambda = \mathrm{Dg}([\lambda_k]_k)$ contains the eigenvalues of the above eigen system. It can be easily seen that F^* and Λ that satisfy

$$(I - P)F^* = F^*\Lambda, \tag{3.8}$$

also satisfy (3.7). Denote the normalized Laplacian as

$$L_n = D^{-1/2}LD^{-1/2}$$

Let $U = [u_k]_k$ be the matrix that $u_k, k = 1, \ldots, K$ are the eigenvectors of L_n. It can also be shown that $F^* = D^{-1/2}U$ also satisfy (3.7) and (3.30).

The global optimum of the problem is not unique but a subspace spanned by the columns of F^* through orthonormal matrices. Let R be a $K \times K$ matrix. If F^* is a feasible solution to (3.28), so is the subspace:

$$\{F^*R|\ R^TR = I\}. \tag{3.9}$$

Furthermore, they have the same objective value. Therefore, a feasible solution remains equally good w.r.t. the problem (3.28) with arbitrary rotations and reflections of F^*. The optimal objective value of (3.28) provides an lower bound to the problem in (3.3).

3.3.4 Discretization

After obtaining the matrix F^*, we can recover Z^* by substituting F^* back into (3.27). However, Z^* is still a real valued matrix. In order to obtain the clustering result, we need to find the discrete solution Z. One way to do that is to find a discrete Z that is close to the subspace (3.31). This can be done by solving such a problem.

$$\begin{aligned} \min_{Z\,R} \quad & \phi(Z,R) = \|Z - Z^*R\|_F^2 \\ \text{s.t.} \quad & Z \in \{0,1\}^{N\times K},\ Z\mathbf{1} = \mathbf{1}. \\ & R^TR = I. \end{aligned} \tag{3.10}$$

The details of how to find a minimum of (3.10) can be found in [81]. Other discretization approaches [2, 61, 82, 81] are also possible to perform the discretization.

3.4 Decision Theoretic View of Normalized Cut

In this section, we establish the relationship among nonparametric density estimator, Bayes risk and spectral clustering. Assume that we have the density model for the data set. If partitioning $\Omega = \{\mathcal{X}_k\}_{k=1,\ ,K}$ is given, we can treat the problem as a classifying problem. We then can calculate the probability that a sample in class $\mathcal{C}_k$ is misclassified by the density model which is given by

$$p(t \neq \mathcal{C}_k | \mathbf{x} \in \mathcal{X}_k),$$

which is also the expected misclassification rate (or Bayes risk with zero-one loss).

The clustering problem can then be seen as finding a good partitioning which makes the following average of the expected

misclassification rate as small as possible.

$$\lambda(\Omega) = \frac{1}{K}\sum_{k=1}^{K} p(t \neq \mathcal{C}_k | \mathbf{x} \in \mathcal{X}_k), \tag{3.11}$$

We call (3.11) the Bayes clustering risk. Here, we show that for a given density estimator, the Bayes clustering risk is equivalent to the normalized cut criterion.

3.4.1 KDE Based Bayes Clustering Risk

Assume that the graph is constructed using Gaussian kernel, where the graph weight is set as

$$w_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), \tag{3.12}$$

Also assume that the density model of the data set is given by following kernel density estimator

$$p(\mathbf{x}) = \frac{1}{n(\sigma\sqrt{2\pi})^d}\sum_{j=1}^{n} \exp(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2\sigma^2}). \tag{3.13}$$

which can be reformulated as mixture model

$$p(\mathbf{x}) = \sum_{k} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k),$$

where the conditional density for each cluster is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{n_k(\sigma\sqrt{2\pi})^d}\sum_{j\in\mathcal{V}_k} \exp(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2\sigma^2}), \tag{3.14}$$

and the prior is given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}. \tag{3.15}$$

Based on the density model and the graph construction method given above, we have following lemmas

Lemma 3.4.1.

$$p(\mathbf{x} \in \mathcal{X}_k) = \frac{1}{n(\sigma\sqrt{2\pi})^d} W(\mathcal{V}_k, \mathcal{V}).$$

Proof.

$$\begin{aligned}
p(\mathbf{x} \in \mathcal{X}_k) - \sum_{i \in \mathcal{V}_k} p(\mathbf{x}_i) &= \sum_{i \in \mathcal{V}_k} \sum_{l=1}^{K} p(\mathbf{x}_i | \mathcal{C}_l) p(\mathcal{C}_l) \\
&= \sum_{i \in \mathcal{V}_k} \sum_{l=1}^{K} \frac{n_l}{n} \frac{1}{n_l(\sigma\sqrt{2\pi})^d} \sum_{j \in \mathcal{V}_l} \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \\
&= \frac{1}{n(\sigma\sqrt{2\pi})^d} \sum_{i \in \mathcal{V}_k} \sum_{l=1}^{K} \sum_{j \in \mathcal{V}_l} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \\
&= \frac{1}{n(\sigma\sqrt{2\pi})^d} \sum_{i \in \mathcal{V}_k, j \in \mathcal{V}} w_{ij} = \frac{1}{n(\sigma\sqrt{2\pi})^d} W(\mathcal{V}_k, \mathcal{V}),
\end{aligned}$$

where $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$. □

It is worth noting that here the $W(\mathcal{V}_k, \mathcal{V}) = \sum_{i \in \mathcal{V}_k, j \in \mathcal{V}} w_{ij}$ includes all w_{ii}, which is equivalent to constructing a graph with self loops. For the case of Gaussian kernel, the edge weights for the self loops are $w_{ii} = 1$.

Lemma 3.4.2.

$$p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) = \frac{1}{n(\sigma\sqrt{2\pi})^d} W(\mathcal{V}_k, \overline{\mathcal{V}}_k).$$

Proof.

$$
\begin{aligned}
p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) &= \sum_{i \in \mathcal{V}_k} p(\mathbf{x}_i, t \neq \mathcal{C}_k) \\
&= \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_i, \mathcal{C}_l) = \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_i | \mathcal{C}_l) p(\mathcal{C}_l) \\
&= \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} \frac{n_l}{n} \frac{1}{n_l (\sigma\sqrt{2\pi})^d} \sum_{j \in \mathcal{V}_l} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}) \\
&= \frac{1}{n(\sigma\sqrt{2\pi})^d} \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} \sum_{j \in \mathcal{V}_l} \exp(-\frac{\|\mathbf{x}_i \quad \mathbf{x}_j\|^2}{2\sigma^2}) \\
&= \frac{1}{n(\sigma\sqrt{2\pi})^d} \sum_{i \in \mathcal{V}_k, j \in \overline{\mathcal{V}}_k} w_{ij} = \frac{1}{n(\sigma\sqrt{2\pi})^d} W(\mathcal{V}_k, \overline{\mathcal{V}}_k).
\end{aligned}
$$

□

From the lemmas, we are ready to show that the Bayes clustering risk is equivalent to the normalized cut criterion which is given by following proposition

Proposition 3.4.3.

$$\lambda(\Omega) = \mathrm{nc}(\Omega). \tag{3.16}$$

Proof.

$$
\begin{aligned}
\lambda(\Omega) &= \frac{1}{K} \sum_{k=1}^{K} p(t \neq \mathcal{C}_k | \mathbf{x} \in \mathcal{X}_k) = \frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x} \in \mathcal{X}_k | t \neq \mathcal{C}_k) p(t \neq \mathcal{C}_k)}{p(\mathbf{x} \in \mathcal{X}_k)} \\
&= \frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k)}{p(\mathbf{x} \in \mathcal{X}_k)} = \frac{1}{K} \sum_{k=1}^{K} \frac{W(\mathcal{V}_k, \overline{\mathcal{V}}_k)}{W(\mathcal{V}_k, \mathcal{V})} = \mathrm{nc}(\Omega).
\end{aligned}
$$

□

The proposition concludes that the partitioning, which minimizes the normalized cut criterion on a graph with self loops

constructed by the fixed bandwidth kernel function, is a partitioning that minimizes the average expected misclassification rate where the conditional density of each class is modeled by the kernel density estimator.

3.4.2 KNN Based Bayes Clustering Risk

Assume that the graph is constructed by KNN approach. Denote $\mathcal{N}(\mathbf{x})$ as neighborhood set of $\mathbf{x}$. The graph weight is set as

$$w_{ij} = \begin{cases} 1 & \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ 0 & \text{otherwise.} \end{cases}$$

Also assume that the density model of the data set is given by following KNN density estimator

$$p(\mathbf{x}) - \frac{K}{nV}, \tag{3.17}$$

which can be reformulated as mixture model

$$p(\mathbf{x}) - \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k),$$

where the conditional density for each cluster is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{n_k V} \tag{3.18}$$

and the prior is given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}. \tag{3.19}$$

Based on the density model and the graph construction method given above, we also have similar results

Lemma 3.4.4.

$$p(\mathbf{x} \in \mathcal{X}_k) = \frac{1}{nV} W(\mathcal{V}_k, \mathcal{V}).$$

Proof.

$$p(\mathbf{x} \in \mathcal{X}_k) = \sum_{\imath \in \mathcal{V}_k} p(\mathbf{x}_\imath) - \sum_{\imath \in \mathcal{V}_k} \frac{K}{nV}$$
$$= \frac{1}{nV} \sum_{\imath \in \mathcal{V}_k \, \jmath \in \mathcal{V}} w_{\imath\jmath} = \frac{1}{nV} W(\mathcal{V}_k, \mathcal{V}).$$

□

It is worth noting that here the $W(\mathcal{V}_k, \mathcal{V}) - \sum_{\imath \in \mathcal{V}_k, \jmath \in \mathcal{V}} w_{\imath\jmath}$ includes all $w_{\imath\imath}$, which is equivalent to constructing a graph with self loops which have weights $w_{\imath\imath} - 1$.

Lemma 3.4.5.

$$p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) = \frac{1}{nV} W(\mathcal{V}_k, \overline{\mathcal{V}}_k).$$

Proof.

$$\begin{aligned} p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) &= \sum_{\imath \in \mathcal{V}_k} p(\mathbf{x}_\imath, t \neq \mathcal{C}_k) \\ &= \sum_{\imath \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_\imath, \mathcal{C}_l) = \sum_{\imath \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_\imath | \mathcal{C}_l) p(\mathcal{C}_l) \\ &= \sum_{\imath \in \mathcal{V}_k} \sum_{l \neq k} \frac{n_l}{n} \frac{K_l}{n_l V} = \frac{1}{nV} \sum_{\imath \in \mathcal{V}_k, \jmath \in \overline{\mathcal{V}}_k} w_{\imath\jmath} = \frac{1}{nV} W(\mathcal{V}_k, \overline{\mathcal{V}}_k). \end{aligned} \tag{3.20}$$

□

From the lemmas, we can easily verify that the proposition (3.4.8) still holds. Therefore we can see that that the partitioning, which minimizes the normalized cut criterion on a KNN graph with self loops. is a partitioning that minimizes the average expected misclassification rate where the conditional density of each class is modeled by KNN density estimator.

3.4.3 General Nonparametric Density Estimator Based Bayes Clustering

Assume the nonparametric density estimator has the general form

$$p(\mathbf{x}) = p(\mathbf{x}|X) = \frac{1}{n}\sum_{j=1}^{n} p(\mathbf{x}|\mathbf{x}_j),$$

where the data set X is viewed as a set of parameter of the density model. For example the Gaussian kernel density estimator

$$p(x) = \frac{1}{n(\sigma\sqrt{2\pi})^d}\sum_{j=1}^{n}\exp(-\frac{\|x - x_j\|^2}{2\sigma^2}) \tag{3.21}$$

can be seen as a mixture model with n atom, each atom $p(\mathbf{x}|\mathbf{x}_j)$ is a Gaussian distribution with mean $\mathbf{x}_j$. These n components can be further grouped to K components as

$$p(\mathbf{x}|X) = \sum_{k} p(\mathbf{x}|\mathcal{X}_k, \mathcal{C}_k)p(\mathcal{C}_k),$$

where the conditional density for each component is given by

$$p(\mathbf{x}|\mathcal{X}_k, \mathcal{C}_k) = \frac{1}{n_k}\sum_{j\in\mathcal{V}_k} p(\mathbf{x}|\mathbf{x}_j), \tag{3.22}$$

and the prior is given by

$$p(\mathcal{C}_k) = \frac{n_k}{n}. \tag{3.23}$$

Assume each atom can be represented in the general form of

$$p(\mathbf{x}|\mathbf{x}_j) = \kappa_j(\mathbf{x}, \mathbf{x}_j),$$

where $\kappa_j(\mathbf{x}, \mathbf{x}_j)$ is a multivariate kernel. Note that, in general the kernel might not be symmetric, i.e. $\kappa_j(\mathbf{x}_i, \mathbf{x}_j) = \kappa_i(\mathbf{x}_j, \mathbf{x}_i)$.

Therefore the kernel can be data dependent. However, we do require

$$\kappa_j(\mathbf{x}, \mathbf{x}_j) \geq 0$$
$$\int \kappa_j(\mathbf{x}, \mathbf{x}_j) d\mathbf{x} = 1$$

so that $p(\mathbf{x}|\mathbf{x}_j)$ is valid density function. Assume that we construct a graph by setting the graph weight as

$$w_{ij} = \kappa_j(\mathbf{x}_i, \mathbf{x}_j).$$

Based on the density model and the graph construction method given above, we have following lemmas

Lemma 3.4.6.

$$p(\mathbf{x} \in \mathcal{X}_k) = \frac{1}{n} W(\mathcal{V}_k, \mathcal{V}).$$

Proof.

$$p(\mathbf{x} \in \mathcal{X}_k) = \sum_{i \in \mathcal{V}_k} p(\mathbf{x}_i) = \frac{1}{n} \sum_{i \in \mathcal{V}_k} \sum_{j \in \mathcal{V}} \kappa_j(\mathbf{x}_i, \mathbf{x}_j)$$
$$= \frac{1}{n} \sum_{i \in \mathcal{V}_k, j \in \mathcal{V}} w_{ij} = \frac{1}{n} W(\mathcal{V}_k, \mathcal{V}).$$

□

Again, here the $W(\mathcal{V}_k, \mathcal{V}) = \sum_{i \in \mathcal{V}_k, j \subset \mathcal{V}} w_{ij}$ includes all w_{ii}, which is equivalent to constructing a graph with self loops.

Lemma 3.4.7.

$$p(\mathbf{x} \in \mathcal{X}_k, t \neq C_k) = \frac{1}{n} W(\mathcal{V}_k, \overline{\mathcal{V}}_k).$$

Proof.

$$\begin{aligned} p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) &= \sum_{i \in \mathcal{V}_k} p(\mathbf{x}_i, t \neq \mathcal{C}_k) \\ &= \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_i, \mathcal{C}_l) = \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} p(\mathbf{x}_i | \mathcal{C}_l) p(\mathcal{C}_l) \\ &= \sum_{i \in \mathcal{V}_k} \sum_{l \neq k} \frac{n_l}{n} \frac{1}{n_l} \sum_{j \in \mathcal{V}_l} \kappa_j(\mathbf{x}_i, \mathbf{x}_j) \\ &= \frac{1}{n} \sum_{i \in \mathcal{V}_k} \sum_{j \in \bar{\mathcal{V}}_k} w_{ij} = \frac{1}{n} W(\mathcal{V}_k, \overline{\mathcal{V}}_k). \end{aligned}$$

□

Therefore, we establish the equivalency between the Bayes clustering risk and normalize cut

Proposition 3.4.8.

$$\lambda(\Omega) = \mathrm{nc}(\Omega).$$

Proof.

$$\lambda(\Omega) - \frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k)}{p(\mathbf{x} \in \mathcal{X}_k)} = \frac{1}{K} \sum_{k=1}^{K} \frac{W(\mathcal{V}_k, \overline{\mathcal{V}}_k)}{W(\mathcal{V}_k, \mathcal{V})} = \mathrm{nc}(\Omega).$$

□

Note that if uniform kernel is used for the atom (the kernel function is the same for all atoms), we have

$$p(\mathbf{x}|\mathbf{x}_j) - \frac{1}{V} \kappa(\mathbf{x}, \mathbf{x}_j),$$

where V is a normalized constant satisfying

$$V = \int \kappa(\mathbf{x}, \mathbf{x}_j) d\mathbf{x}.$$

Then it can be easily seen that the previous lemmas can rewritten as

Lemma 3.4.9.

$$p(\mathbf{x} \in \mathcal{X}_k) = \frac{1}{nV} W(\mathcal{V}_k, \mathcal{V}).$$

Lemma 3.4.10.

$$p(\mathbf{x} \in \mathcal{X}_k, t \neq \mathcal{C}_k) = \frac{1}{nV} W(\mathcal{V}_k, \overline{\mathcal{V}}_k).$$

Then after normalization, the normalization constant is canceled. Note the normalization grows exponential w.r.t. d. In general the nonparametric density estimator might suffer from the dimensionality curse. In order to make the estimator consistent, we require that n increase exponentially w.r.t d ($V \propto h^d$). However, the consistency of spectral clustering does not has this requirement since V is canceled by the normalization. Therefore, spectral clustering does not suffer from the dimensionality curse.

It is also easy to see that, the normalized cut on a graph without self-loops can be viewed as minimizing the leave-one-out cross validation Bayes clustering risk.

3.4.4 Discussion

The analysis given above does not depend on the choice of the kernel density estimator. However, the analysis does suggest that in order to obtain a good clustering result, one should construct the underlying graph in a way that it reflects the underlying true data distribution.

Therefore, one can use a more general form of kernel functions and prior to construct the graph in order to model the true density of the data. Location dependent kernels, such as the adaptive bandwidth kernel, can also be used to improve the graph construction, which allows the bandwidth to vary from one observation to another. It gives the flexibility of using a smaller bandwidth (hence reduces the bias of the estimate) in

regions where there are many observations, and a larger bandwidth (hence reduces the variance of the estimate) in regions where there are relatively few observations. However if one uses it in a consistent way, one should properly normalize the kernel function when computing the edge weights.

We can also use domain specific kernel function for different data types. For example, for data of binary code, one can use Bernoulli density function. For data of positive integer features, Poisson or multinomial density function can be used.

In general, partitioning a graph by minimizing the normalized cut criterion can be seen as clustering the data in the feature space by minimizing the Bayes clustering risk if proper connection between the graph construction method and the nonparametric density estimator is established.

Given the Bayes risk view, we bridge the originally unrelated two steps of spectral clustering: graph construction and graph partitioning. We can treat the spectral clustering as a unified problem. The normalized cut algorithm is to minimize the Bayes clustering risk while the data distribution is modeled by certain nonparametric density model. However, in order to make the normalized cut meaningful, the graph has to be constructed properly reflecting the underlying data distribution.

It is worth noticing that constructing a graph to minimize the normalized cut criterion dose not necessarily lead to good clustering result. It is similar to the classification problem in which choosing a classification function by minimizing the training error is not a good idea. It often leads to overfitting. We can always construct a graph which has zero normalized cut ratio. One extreme example is that we can construct a graph with only self loop, and then any partitioning will minimize the Bayes clustering risk. Minimizing the leave-one-out version of error still has the same problem. We can construct a graph that each sample only links to one other sample in the data set. On this

graph, one still can obtain zero normalized cut ratio with an arbitrary bad partitioning.

Therefore, it does not make sense to simultaneously optimize normalized cut criterion, the graph construction and partitioning. One can first fit a good density model before the partitioning. Or one can optimize the normalized cut with respect the graph construction by restricting the complexity of the function that is used. A regularized version of normalized cut criterion might serve this purpose.

3.4.5 Normalized Cut on General Graphs

From the derivation above, the Bayes clustering risk does require the graph to be an undirected one. For example, in the KNN graph construction method, a sample point $\mathbf{x}_i$, of which $\mathbf{x}_j$ is among the K nearest neighbors, is not necessarily among K nearest neighbors of $\mathbf{x}_j$. Previously this problem is solved by perform a post process step to force the mutual neighborhood. However, this post process step is not necessary. Here, we derive a general spectral clustering algorithm which can work on both directed graphs and undirected graphs.

3.4.6 Spectral Relaxation

Let $G = (\mathcal{V}, \mathcal{E})$ be a directed graph with vertex set $\mathcal{V} = \{v_i\}_{i=1, \ldots, n}$. Denote $w_{ij} \geq 0$ as the weight of a directed edge from vertices v_i to v_j. The weighted adjacency matrix of the graph is the matrix $W = [w_{ij}]_{i,j=1, \ldots, n}$. In general, we do not require $w_{ij} = w_{ji}$. The (out) degree of a vertex $v_i \in \mathcal{V}$ is defined as

$$d_i = \sum_j w_{ij}.$$

The degree vector is defined as $\mathbf{d} = [d_i]^T_{i=1, \ldots, n}$. The degree matrix D is defined as the diagonal matrix with the degrees

$d_1, \ldots, d_n$ on the diagonal. For two not necessarily disjoint sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$ we define

$$W(\mathcal{A}, \mathcal{B}) = \sum_{i \subset \mathcal{A}, j \subset \mathcal{B}} w_{ij}.$$

Let $\mathbf{z}_k \in \{0, 1\}^n$ be the binary indicator vector for the kth cluster and $Z = [\mathbf{z}_k]_{k=1, \ldots, K}$ be the indicator matrix. We have

$$W(\mathcal{V}_k, \mathcal{V}) = \mathbf{z}_k^T D \mathbf{z}_k \tag{3.24}$$

and

$$W(\mathcal{V}_k, \mathcal{V}_k) = \mathbf{z}_k^T W \mathbf{z}_k. \tag{3.25}$$

From (3.24) and (3.25) we have

$$\begin{aligned} W(\mathcal{V}_k, \overline{\mathcal{V}}_k) &= W(\mathcal{V}_k, \mathcal{V}) - W(\mathcal{V}_k, \mathcal{V}_k) \\ &= \mathbf{z}_k^T D(I - P)\mathbf{z}_k, \end{aligned}$$

where $P = D^{-1}W$. Then the normalized cut criterion can be rewritten as

$$\text{ncut}(\Omega) \quad \frac{1}{K} \sum_{k=1}^{K} \frac{\mathbf{z}_k^T D(I - P)\mathbf{z}_k}{\mathbf{z}_k^T D \mathbf{z}_k}.$$

Define the scaled partition matrix

$$F = Z(Z^T D Z)^{-1/2},$$

where $Z = [\mathbf{z}_k]_k$, $k = 1, \cdots, K$. Since $Z^T D Z$ is a diagonal matrix, the columns of F are the columns of Z scaled by the inverse square root of the out degree. Clearly we have

$$F^T D F = (Z^T D Z)^{-1/2} Z^T D Z (Z^T D Z)^{-1/2} = I. \tag{3.26}$$

Given a scaled partition matrix F, we can restore the corresponding Z by

$$Z = \text{Dg}(\text{dg}^{-1/2}(F F^T))F, \tag{3.27}$$

where $M = \mathrm{Dg}(\mathbf{v})$ denotes constructing a diagonal matrix M from the vector $\mathbf{v}$ and $\mathbf{v} = \mathrm{dg}(M)$ denotes extracting the diagonal elements of matrix M to form a vector $\mathbf{v}$.

Substituting F to (3.3) and relaxing F to take real values that satisfy the constraints $F^T DF = I$, we have a convex optimization problem as:

$$\begin{aligned} \min_{F} \quad & \mathrm{nc}(F) = \frac{1}{K}\mathrm{tr}(F^T D(I-P)F) \\ s.t. \quad & F^T DF - I. \end{aligned} \tag{3.28}$$

The global optimal solution F^* of this problem is achieved by solving the following generalized eigendecomposition problem

$$D(I-P)F^* = DF^*\Lambda, \tag{3.29}$$

where $\Lambda = \mathrm{Dg}([\lambda_k]_k),\ k = 1, \cdots, K$, contains the eigenvalues of the above eigensystem. It can be easily seen that F^* and Λ that satisfy

$$(I-P)F^* = F^*\Lambda, \tag{3.30}$$

also satisfy (3.7), . The smallest eigenvectors corresponding to the smallest eigenvalues of matrix $I-P$ are also the eigenvectors corresponding to largest eigenvalues of matrix P. Since P is stochastic matrix, it has real value eigenvectors according to Perron-Frobenius theorem.

The global optimum of the problem is not unique but a subspace spanned by the columns of F^* through orthonormal matrices. Let R be a $K \times K$ matrix. If F^* is a feasible solution to (3.28), so is the subspace:

$$\{F^*R|\ R^T R = I\}. \tag{3.31}$$

Furthermore, they have the same objective value, i.e., $\mathrm{nc}(F^*) = \mathrm{nc}(F^*R)$. Therefore, a feasible solution remains equally good w.r.t. the normalized cut objective with arbitrary rotations and

reflections of F^*. The optimal objective value of (3.28) provides an lower bound to the problem in (3.3).

After obtaining the relaxed solution of normalized cut, we have to discretize the real value solution to obtain the class indicators. Many discretization approaches [81, 2, 20, 61, 82] are proposed to perform the discretization. In this paper, we adopt the one in [81] to generate the clustering results.

3.4.7 Local Scaling Directed Graph Construction

In this chapter, we use a variable bandwidth KDE to construct the graph from a data set. The variable bandwidth KDE is given by

$$f_b(x) = \sum_{j=1}^{n} \frac{1}{nh_j} K(\frac{x - x_j}{h_j}), \tag{3.32}$$

where the bandwidth h_j depends on the context information of x_j. It is well known that with a fixed bandwidth, the kernel estimate tends to oversmooth at the main part and undersmooth at the tail part of the distribution. This is the basic motivation for considering a variable bandwidth KDE, which allows the bandwidth to vary from one observation to another. It gives the flexibility of using a smaller bandwidth (hence reduces the bias of the estimate) in regions where there are many observations, and a larger bandwidth (hence reduces the variance of the estimate) in regions where there are relatively few observations.

The local bandwidth h_j is set to be the distance between x_j and its kth nearest neighbor. The parameter k is selected by cross validation. The k with which the variable bandwidth KDE (4.5) has the largest leave-one-out likelihood on the given data set is used in our algorithm. According to the previous analysis, by using the variable bandwidth KDE, the edge weight of the constructed graph is $w_{ij} = \frac{1}{h_i} K(\frac{x_i - x_j}{h_i})$. Here, the Gaussian kernel is used. The edge weight of the constructed graph between

x_i and x_j is

$$w_{ij} = \frac{1}{h_i} \exp(-\frac{\|x_i - x_j\|^2}{2h_i^2}).$$

Notice that, in general, w_{ij} is not necessarily equal to w_{ji}. Therefore, the constructed graph is a directed graph.

3.5 Experiments

In this section, we present the clustering results obtained by the proposed local Gaussian based digraph spectral clustering (DSC) algorithm on a number of synthetic and the real data sets. We also compare our DSC algorithm with K-means, the NJW [61], and the self-tuning spectral clustering (STSC) [82] algorithms on the real data sets. We conduct experiments on the following real data sets:

- USPS-All: This data set consists of images of 10 handwritten digits. Each category contains 500 samples selected randomly from the USPS database.
- USPS-5: All the samples are from digits 2, 3, 5, 6, and 8 in USPS-All.
- UMist-All: This data set consists of face images of 20 different persons.
- UMist-10: The data are from UMist data set belonging to classes 1 to 10.
- IRIS: The data are from the UCI repository comprising 3 classes of 50 instances each, where each class refers to a type of iris plant.

More details of the data sets are summarized in Table 5.1.

To evaluate the performances of the clustering algorithms, we compute the following two performance measures from the

Table 3.1 Descriptions of the data sets used in the experiments

Dataset	K	d	n
USPS-All	10	256	5000
USPS-5	5	256	2500
UMist-All	20	10304	575
UMist-10	10	10304	265
IRIS	3	4	150

Table 3.2 Clustering results by the four algorithms

Method	Measure	USPS-All	USPS-5	UMist-All	UMist-10	IRIS
K-means	Error	0.5448	0.3600	0.5339	0.5208	0.1067
	NMI	0.4569	0.4491	0.6726	0.6131	0.7582
NJW	Error	0.4912	0.4180	0.4035	0.3019	0.1000
	NMI	**0.6091**	0.6075	0.8139	0.8528	0.7908
STSC	Error	0.7084	0.4704	0.5426	0.5321	0.0600
	NMI	0.2843	0.3064	0.6291	0.5919	0.8334
DSC	Error	**0.4582**	**0.1832**	**0.3652**	**0.2151**	**0.0333**
	NMI	0.5610	**0.6343**	**0.8411**	**0.8954**	**0.8551**

clustering results: normalized mutual information (NMI) and minimal clustering error (Error). The NMI is defined as

$$\mathrm{NMI}(x, y) - \frac{I(x, y)}{\sqrt{H(x)H(y)}}, \tag{3.33}$$

where $I(x, y)$ is the mutual information between x and y, and $H(x)$ and $H(y)$ are the entropies of x and y respectively. Note that $0 \leq \mathrm{NMI}(x, y) \leq 1$ and $\mathrm{NMI}(x, y) = 1$ when $x = y$. The larger is the value of NMI, the better is a clustering result.

The clustering error is defined as the minimal classification error among all possible permutation mappings defined as:

$$\mathrm{Error} - \min(1 - \frac{1}{n}\sum_{i=1}^{n} \delta(y_i, \mathrm{perm}(c_i))), \tag{3.34}$$

where y_i and c_i are the true class label and the obtained clustering result of x_i, respectively, $\delta(x, y)$ is the delta function that equals 1 if $x = y$ and 0 otherwise.

The clustering results by the four algorithms, K-means, N-JW, STSC, and DSC, are summarized in Table 5.2. The DSC algorithm obtains the smallest errors in all the cases, and produces the largest NMI values on all the data sets except one. These results demonstrate that the DSC can achieve good performances consistently on real world data sets.

3.6 Conclusion

We have established the relationship among Bayesian decision theory, nonparametric density models and spectral clustering. We are able to bridge the original unrelated two steps of spectral clustering algorithm to one unified approach. From this new perspective, we gain the insight on how to construct a graph for clustering purpose. Other extensions for spectral clustering are also possible. For example, we can also use mixture models or Bayesian nonparametric models to model the data density. This preprocess step will dramatically decrease the size of the graph. Even better, the parameters of the graph weights can be automatically determined. We will investigate this approach in our future work.

□ **End of chapter.**

Chapter 4

Isoperimetric Cut on Density Graphs

Summary

In this chapter, we propose the probabilistic view of isoperimetric cut on a graph constructed by using a certain kernel function. From this probabilistic perspective, the algorithm can be seen as assigning optimal class labels to samples that minimizes the nonparametric kernel density estimation based Bayes error rate of a two class problem. We then propose to construct graphs using variable bandwidth kernel density estimators, which naturally results in directed graphs. A directed graph encodes the local density information of the data. In order to cluster the vertices of the directed graph, we propose an algorithm which performs isoperimetric cut on a directed graph. The cut solution can be obtained efficiently by solving a system of linear equations.

4.1 Introduction

Due to the success of spectral clustering methods [54, 61, 67, 81, 44], graph based clustering algorithms are of great interests recently. These methods first compute the pairwise similarities of the data to construct an undirected graph. Then the global clustering result is obtained by partitioning the vertexes of the graph into disjoint sets according to some criterions. One advantage of these methods is that they do not make strong assumptions about the global distribution of the data. Therefore, they can potentially deal with data of irregular shapes.

Despite the success of the spectral clustering, there are still some problems. First, from a theoretical perspective, it is still unknown what a good graph for clustering task is. Therefore how to construct a graph for better performance is still a open problem. Second, from practical perspective, it is still a hard problem to perform eigen-decomposition on a very large affinity matrix. Therefore how to perform the graph based clustering algorithm on very large data set is another problem which we cannot ignore.

In this chapter, we first present a novel probabilistic view of the graph based algorithm. We show that the isoperimetric ration of a graph constructed by a kernel function is equivalent to the Bayes risk of a where the distribution of the data is modeled by kernel density estimator. From this viewpoint, we can see that in order to obtain a good clustering result, one should construct a graph reflecting the underlying density of the data. Therefore, we propose to construct a graph by using variable bandwidth kernel density estimators which naturally results in a directed graph. The digraph effectively explores the local density of the data.

The isoperimetric cut (IsoCut) digraph bipartitioning algorithm [16] is proposed to cut the constructed directed graph

into two disjoint parts by minimizing a the isoperimetric ratio. We also provide a random walk view of the IsoCut algorithm. By adopting the random walk view, we can handle both the directed and undirected graph partitioning problems in a unified framework. Finding the exact solution of this combinatorial problem is NP-hard. However, an approximate solution can be efficiently achieved by solving a sparse linear system of equations. Given a data set, the clustering result is then obtained by iteratively cutting the constructed digraph into disconnected subgraphs.

4.2 Probabilistic View of Isoperimetric Cut

In this section, we revisit the isoperimetric constant, the spectral clustering algorithm, and kernel density estimation. Then we establish the relationships among them.

4.2.1 The Isoperimetric Constant on Manifolds

The isoperimetric constant is originally defined by Cheeger [14] in Riemannian geometry [43]. Let $\mathcal{M}$ be a d-dimensional closed Riemannian manifold. $\mathrm{Vol}(S)$ be the volume of a d-dimensional submanifold S, and $\mathrm{Vol}(\partial S)$ be the volume of the boundary ∂S, which is a $(d-1)$-dimensional submanifold. The Cheeger isoperimetric constant of $\mathcal{M}$ is defined as

$$h = \inf_{S} \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)}.$$

Intuitively, the Cheeger isoperimetric constant defines the small bottleneck boundary of the manifold which separates part of the manifold from rest of it.

4.2.2 The Isoperimetric Constant on Graphs

In the context of an undirected graph, S is a subset of the vertices in the graph. The boundary of S is defined as $\partial S = \{e_{ij} | i \in S, j \in \overline{S}\}$. Then the isoperimetric constant h_G is [55, 18]

$$h_G = \min_S \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)}, \tag{4.1}$$

where $\mathrm{Vol}(\partial S) = \sum_{i \in S, j \in \overline{S}} w_{ij}$, $\mathrm{Vol}(S) = \sum_{i \subset S, j \subset V} w_{ij}$ and $\mathrm{Vol}(S) \leq \mathrm{Vol}(V)/2$. w_{ij} is the weight of edge e_{ij} computed from the sample pair $\mathbf{x}_i$ and $\mathbf{x}_j$ by $w_{ij} = \exp(-\|\mathbf{x}_i \quad \mathbf{x}_j\|^2/(2\sigma^2))$.

The isoperimetric constant of an undirected graph satisfies $h_G \in [0, 1]$, and is strictly positive iff the graph is connected. In [31, 32], the authors propose an algorithm that minimizes the graph isoperimetric constant to solve an image segmentation problem.

The spectral clustering algorithm proposed in [67, 44] is a graph bi-partitioning algorithm which minimizes the normalized cut criterion

$$\min_S \frac{1}{2} \left(\frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)} + \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(\overline{S})} \right). \tag{4.2}$$

As shown in [44, 18], the algorithm also minimizes the upper bound of the graph isoperimetric constant.

4.2.3 A Bayesian Decision Theory on Manifolds

Here we analyze the isoperimetric cut algorithm from the Bayesian decision theoretic perspective.

In the application of data clustering, we assume that the data $X = \{\mathbf{x}_i\}_{i=1,\ldots,n}$ are indecently sampled from a underlying distribution with the probability measure $P(\mathbf{x})$ from $\mathcal{M}$. Assume that S is a subset of indices of the samples in X and S is the submanifolds enclosing the samples $\{x_i | i \in S\}$. We denote the

labels of the sample set $\{x_i | i \in S\}$ as $\mathcal{C}_S$ and others as $\mathcal{C}_{\overline{S}}$. The volume of S is

$$\mathrm{Vol}(S) - \int_S dP(\mathbf{x}) = \int_S p(\mathbf{x}) d\mathbf{x},$$

where $P(\mathbf{x})$ is a probability measure on $\mathcal{M}$ satisfying $\int_{\mathcal{M}} dP(\mathbf{x}) = 1$. $p(\mathbf{x})$ is the corresponding probability density function.

The probability that samples in region S are misclassified is $P(\mathcal{C}_{\overline{S}} | \mathbf{x} \in S)$ which can be rewritten as

$$\begin{aligned} P(\mathcal{C}_{\overline{S}} | \mathbf{x} \in S) - & \frac{P(\mathbf{x} \in S. \mathcal{C}_{\overline{S}})}{P(\mathbf{x} \in S)} = 1 - \frac{P(\mathbf{x} \in S, \mathcal{C}_S)}{P(\mathbf{x} \in S)} \\ & -1 \quad P(\mathcal{C}_S | \mathbf{x} \in S), \end{aligned}$$

where

$$P(\mathbf{x} \in S) = \int_S p(\mathbf{x}) d\mathbf{x}$$

and

$$P(\mathbf{x} \in S, \mathcal{C}_S) = \int_S p(\mathbf{x} | \mathcal{C}_S) P(\mathcal{C}_S) d\mathbf{x}.$$

Evaluation of these integral are intractable. However we can empirically approximate $P(\mathcal{C}_S | \mathbf{x} \in S)$ by summation over samples. We adopt following approximations

$$\int_S p(\mathbf{x}) d\mathbf{x} \approx \frac{\sum_{i \in S} p(\mathbf{x}_i)}{\sum_{i \in V} p(\mathbf{x}_i)}$$

and

$$\int_S p(\mathbf{x} | \mathcal{C}_S) d\mathbf{x} \approx \frac{\sum_{i \in S} p(\mathbf{x}_i | \mathcal{C}_S)}{\sum_{i \in V} p(\mathbf{x}_i | \mathcal{C}_S)}.$$

Then we following approximation

$$\begin{aligned} P(\mathcal{C}_S | \mathbf{x} \in S) = & \frac{\int_S p(\mathbf{x} | \mathcal{C}_S) P(\mathcal{C}_S) d\mathbf{x}}{\int_S p(\mathbf{x}) d\mathbf{x}} \\ \approx & \frac{\sum_{i \in S} p(\mathbf{x}_i | \mathcal{C}_S) P(\mathcal{C}_S)}{\sum_{i \in S} p(\mathbf{x}_i)}, \end{aligned}$$

Here, we utilize the the same approximation used in the non-parametric density models

$$P(\mathbf{x} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x})d\mathbf{x} \approx Vp(\mathbf{x}), \tag{4.3}$$

where $\mathcal{R}$ is a small region on the manifold.

4.2.4 Bayesian Decision Theoretic View for Isoperimetric Cut

We assume the density function is modeled by a kernel density estimator given by

$$p(\mathbf{x}) = \frac{1}{n(h\sqrt{2\pi})^d} \sum_{j=1}^{n} \exp(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2h^2}). \tag{4.4}$$

The density function can be written as a mixture model of two components as

$$p(\mathbf{x}) = p(\mathcal{C}_S)p(\mathbf{x}|\mathcal{C}_S) + p(\mathcal{C}_S)p(\mathbf{x}|\mathcal{C}_{\bar{S}}),$$

where the conditional density functions are

$$p(\mathbf{x}|\mathcal{C}_S) = \frac{1}{|S|h\sqrt{2\pi}} \sum_{j \in S} \exp(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2h^2})$$

and

$$p(\mathbf{x}|\mathcal{C}_{\overline{S}}) = \frac{1}{|\overline{S}|h\sqrt{2\pi}} \sum_{j \in \overline{S}} \exp(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{2h^2}).$$

The priors are

$$P(\mathcal{C}_S) = |S|/n, \; P(\mathcal{C}_{\bar{S}}) = |\overline{S}|/n,$$

Substituting the conditional density functions and priors into

$$\begin{aligned} P(\mathcal{C}_{\overline{S}}|\mathbf{x} \in S) =& 1 - P(\mathcal{C}_S|\mathbf{x} \in S) \\ \approx& 1 - \frac{\sum_{i \in S} p(\mathbf{x}_i|\mathcal{C}_S)P(\mathcal{C}_S)}{\sum_{i \in S} p(\mathbf{x}_i)} \end{aligned}$$

and define

$$w_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2h^2}),$$

we have

$$\begin{aligned} P(\mathcal{C}_{\overline{S}}|\mathbf{x} \in S) \approx & 1 - \frac{W(S,S)}{W(S,V)} = \frac{W(S,\overline{S})}{W(S,V)} \\ = & \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)} = h(G), \end{aligned}$$

where $W(A,B) = \sum_{i\in A, j\in B} w_{ij}$. Then we can see that the Cheeger isoperimetric constant which can be equivalently written as

$$h = \min_S P(\mathcal{C}_{\overline{S}}|\mathbf{x} \in S)$$

is the minimal misclassification rate.

Therefore, the cut what minimizes the isoperimetric ratio on the graph constructed by Gaussian kernel is the partitioning that minimizing the misclassification rate. As shown in Fig.4.1, the optimal boundary which minimizes the isoperimetric constant cuts through the low density area and separates the manifold into two disjoint parts with large volumes.

It is worth noting that the above analysis does not depend on the choice of the kernel density estimator. However, the analysis does suggest that in order to obtain a good clustering result, one should construct the underlying graph, i.e., the kernel density estimator, to reflect the data distribution. In general, the constructed graph is not necessarily an undirected graph. For example a data dependent kernel may result in a directed graph. Therefore, we have to develope a general isoperimetric cut algorithm which can deal with both directed and undirected graphs.

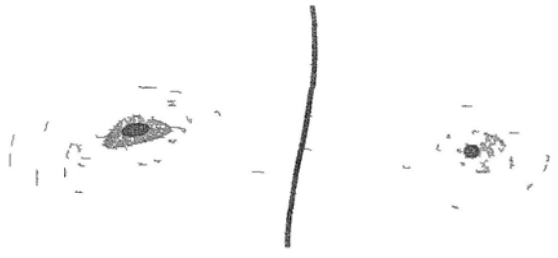


Figure 4.1 A boundary that minimizes the isoperimetric constant

4.3 Isoperimetric Cut on General Graphs

In this section, we first introduce variable bandwidth kernel density estimator based directed graph construction methods. Then we propose the random walk isoperimetric cut algorithm to partition the vertexes of the graph into disjoint subsets. We also analyze the IsoCut algorithm from different viewpoints.

4.3.1 Local Scaling Directed Graph Construction

The first step of spectral graph clustering methods is to construct a graph from a vector data set. The edge weights are usually computed by the Gaussian kernel $\exp(-\|x_i - x_j\|^2/(2\sigma^2))$[1]. However, as indicated in [59], for certain data sets, say, multi-scale data, the Gaussian kernel with a single uniform scaling parameter σ is not informative enough for modeling the pairwise relations. The constructed graph is not able to capture the underlying data distribution. As a result, the intrinsic clusters of the data may not be obtained by partitioning the graph.

[1]Without ambiguity, in this section, x or x_i is used to denote a sample vector

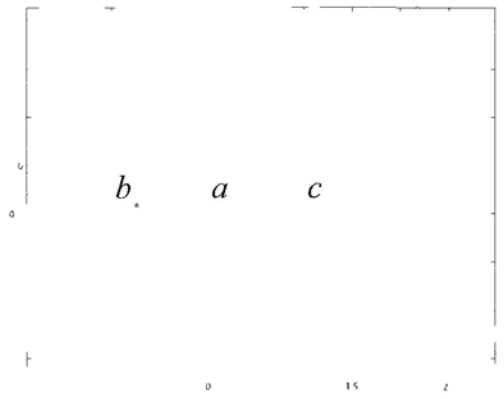


Figure 4.2 The local density affects the relationships between sample pairs

Instead of selecting a single scaling parameter, several papers suggest to compute the edge weights by incorporating local information in various ways. The authors of [15] propose to estimate local Gaussian distributions to construct a directed probabilistic graph. The authors of [83] construct the graph using coding length. However, these methods are very time consuming which limits their practical use. The authors of [82] suggest to replace the uniform σ^2 of the Gaussian kernel with a location dependent scale $\sigma(x_i)\sigma(x_j)$, but they do not provide a principal justification why the edges should be constructed this way.

Consider the data shown in Figure 4.2. The Euclidean distances $d(a, b)$ and $d(a, c)$ are equal. Then the similarities computed by the fixed bandwidth Gaussian kernel are the same. However with the context data points around sample a, apparently a is more likely to belong to the same cluster as b than as c. Here the local density of the data distribution is important for modeling the relationships between the sample pairs. This intuitive example motivates us to use a location dependent method to construct the graph.

In this paper, we incorporate the nonparametric density estimation view to use a variable bandwidth KDE to construct the graph from a data set. The variable bandwidth KDE is given

by

$$f_b(x) = \sum_{j=1}^{n} \frac{1}{nh_j} K(\frac{x - x_j}{h_j}), \tag{4.5}$$

where the bandwidth h_j depends on the context information of x_j. It is well known that with a fixed bandwidth, the kernel estimate tends to oversmooth at the main part and undersmooth at the tail part of the distribution. This is the basic motivation for considering a variable bandwidth KDE, which allows the bandwidth to vary from one observation to another. It gives the flexibility of using a smaller bandwidth (hence reduces the bias of the estimate) in regions where there are many observations, and a larger bandwidth (hence reduces the variance of the estimate) in regions where there are relatively few observations.

In this chapter, the local bandwidth h_j is set to be the distance between x_j and its kth nearest neighbor. The parameter k is selected by cross validation. The k with which the variable bandwidth KDE (4.5) has the largest leave-one-out likelihood on the given data set is used in the IsoCut algorithm. According to the previous analysis, by using the variable bandwidth KDE, the edge weight of the constructed graph is $w_{ij} = \frac{1}{h_i} K(\frac{x_j - x_i}{h_i})$. Here, the Gaussian kernel is used. The edge weight of the constructed graph between x_i and x_j is

$$w_{ij} = \frac{1}{h_i} \exp(-\frac{\|x_i - x_j\|^2}{2h_i^2}).$$

Notice that, in general, w_{ij} is not necessarily equal to w_{ji}. Therefore, the constructed graph is a directed graph.

4.3.2 Isoperimetric Cut on Directed Graphs

After constructing a graph from the data by using KDE, we have to partition the graph into disjoint subgraphs in order to obtain a clustering result of the original data. However, in general

the graph can be either directed or undirected. Uniform treatment of both graphs is needed. In this section, we generalize the isoperimetric so that we are able to treat the isoperimetric problem of undirected and directed graphs in a unified way.

A directed graph $G = (V, E)$ consists of a finite set of vertices $v \in V$ together with a subset of edges $e \in E \subseteq V \times V$. An edge e_{ij} of the directed graph is an ordered pair from vertex v_i to vertex v_j associated with the edge weight w_{ij}. The degree of vertex v_i is $d_i = \sum_j w_{ij}$.

For a given weighted directed graph, there is a natural random walk on the graph with the one step transition probability from v_i to its adjacent v_j defined as $p_{ij} = w_{ij}/d_i$. Also define $\pi_i = d_i / \sum_i d_i$ as a prior of each vertex. For all sample pairs, we have the stochastic matrix $P = [p_{ij}]_{ij}$, $i, j = 1, \cdots, |V|$ satisfying $P\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ is a vector with all entries being 1. For background reading on random walks in general we refer to [62, 11], and for random walks on graphs we recommend [1, 49].

For a finite state irreducible Markov chain on a graph with the transition probability matrix P, define the volume of the boundary of the vertex (state) set S as the sum of the weighted transition probabilities: $\mathrm{Vol}(\partial S) = \sum_{i \in S, j \in \overline{S}} \pi_i p_{ij}$. $\mathrm{Vol}(\partial S)$ is also the probability with which a random walker jumps from S to its complement set $\overline{S}$ Also define the volume of S as $\mathrm{Vol}(S) = \sum_{i \in S} \pi_i$. $\mathrm{Vol}(S)$ is the probability with which the random walker occupies a vertex in S. Then the isoperimetric constant of the random walk is defined as

$$h_R = \inf_S \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)} = \min_S \frac{\sum_{i \in S, j \in S} \pi_i p_{ij}}{\sum_{i \subset S} \pi_i}. \tag{4.6}$$

The constant h_R is the minimal probability of the random walker jumping from the vertex set S to its complement set $\overline{S}$ in one step if the current state is in S, i.e., $\min_S P(S \to \overline{S}|S)$. It represents the probability bottleneck on the state space of the random walk process.

Define an indicator vector $\mathbf{z} \in \{0,1\}^n$, where

$$z_i = \begin{cases} 1 & \mathbf{x}_i \in \overline{S} \\ 0 & \mathbf{x}_i \in S. \end{cases}$$

Then the volume of the boundary can be rewritten as

$$\mathrm{Vol}(\partial S) = \sum_{i,j} \pi_i p_{ij} (z_i - z_j)^2 - \mathbf{z}^T \Pi (I \quad P)\mathbf{z},$$

and the volume of the vertex set S becomes

$$\mathrm{Vol}(S) = \mathbf{z}^T \pi = \mathbf{z}^T \Pi \mathbf{1},$$

where Π is a diagonal matrix of the elements of the stationary distribution vector, i.e., $\Pi = \mathrm{diag}(\pi)$.

Finally the isoperimetric constant of the random walk can be rewritten as

$$h_R = \inf_S \frac{\mathrm{Vol}(\partial S)}{\mathrm{Vol}(S)} = \min_{\mathbf{z}} \frac{\mathbf{z}^T \Pi (I - P)\mathbf{z}}{\mathbf{z}^T \Pi \mathbf{1}}. \tag{4.7}$$

This definition of the isoperimetric constant in terms of the random walk is consistent with the definition (4.1) for an undirected graph. Given an undirected graph with the adjacent matrix W, we have the natural random walk with the transition probability $P = D^{-1}W$, where D is a diagonal matrix with each entry on the diagonal being the degree of each vertex, i.e., $D = \mathrm{diag}(W\mathbf{1})$. This Markov chain is reversible. The stationary probability of the random walk is proportional to the degree of each vertex. Substituting $P = D^{-1}W$ and $\Pi = D/\mathrm{trace}(D)$ to (4.7), we have (4.1).

The matrix $L = I \quad P$ in (4.7) is also referred to as the random walk Laplacain [39], which has a good asymptotic convergence property. It has been shown that for a non-uniform measure on the submanifold the operator L is guaranteed to converge to the weighted Laplace-Beltrami operator. However the graph

Lapacian and normalized graph Laplacian [18] do not have this property.

Our goal is to design a graph partitioning algorithm to minimize the isoperimetric constant. In fact, directly minimizing the isoperimetric constant is infeasible. The exact solution to this discrete optimization problem is NP-hard [21, 32].

In order to solve the partitioning problem, we relax the binary definition of $\mathbf{z}$ so that it can take nonnegative real values. Then the problem is transformed to

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{z}^T \Pi (I - P) \mathbf{z} \\ \text{s.t.} \quad & \mathbf{z}^T \Pi \mathbf{1} = I. \end{aligned} \tag{4.8}$$

By introducing a Lagrange multiplier λ, we turn (4.8) into a constraint free optimization problem

$$Q(\mathbf{z}) = \mathbf{z}^T \Pi (I - P) \mathbf{z} - \lambda \mathbf{z}^T \Pi \mathbf{1}.$$

Taking the derivative of $Q(\mathbf{z})$ w.r.t. $\mathbf{z}$, and setting it equal to 0, we have

$$2\Pi (I - P) \mathbf{z} = \Pi \mathbf{1}.$$

Therefore, the problem of finding the solution $\mathbf{z}$ that minimizes $Q(\mathbf{z})$ reduces to solving a linear system

$$(I - P) \mathbf{z} = \mathbf{1}, \tag{4.9}$$

where the scalar parts are dropped since only relative values are useful.

The matrix $L = I - P$ is singular since $L\mathbf{1} = 0$. Therefore, the linear system (4.9) is ill posed. To achieve a unique solution of (4.9), we need extra constraints.

As we assume the transition matrix P is irreducible, the directed graph associated with P is strongly connected. We can designate an arbitrary vertex v_g to be included in S, i.e., $z_g = 0$ (v_g is called the ground vertex in the rest of this chapter), which

is equivalent to removing the gth row and column of L (the remaining matrix is denoted by L_0). and the gth row of $\mathbf{z}$ (the remaining vector is denoted by $\mathbf{z}_0$) in (4.9). Then the linear system

$$L_0\mathbf{z}_0 = \mathbf{1} \tag{4.10}$$

is well posed, which can be efficiently solved by the conjugate gradient method. The solution $\mathbf{z}_0$ is a nonnegative real-valued vector. The bi-partitioning result can be obtained by thresholding $\mathbf{z}_0$. Vertexes with an z_i below the threshold are placed in S. We use $\mathbf{z}$ to collectively refer to $\mathbf{z}_0$ and the designated value of $z_g = 0$. Several thresholding strategies can be applied. For example, the jump cut which chooses a threshold that separates vertices on either side of the largest jump in a sorted $\mathbf{z}$, and the criterion cut which chooses the threshold that gives the lowest value of the isoperimetric ratio.

To achieve a multi-class clustering result, the algorithm is recursively applied to the subgraphs with the smallest isoperimetric constants, until the number of subgraphs reaches a predefined value.

There are several ways to choose the vertex v_g, such as randomly picking a vertex. In this chapter, we choose the vertex with the maximal stationary probability. This strategy is based on the heuristic that a vertex with a high stationary distribution is the one with high probability that a random walker jumps to it. Such a vertex is likely in the interior of a cluster but not on the boundary. Empirically, we have found that, as long as v_g is not along the ideal boundary, a reasonable partitioning with a small isoperimetric ratio can be produced.

4.3.3 A Random Walk Hitting Time View

The expected hitting time $h(j|i)$ is defined as the expected number of steps that a random walker, starting from the vertex (s-

tate) $v_i \neq v_j$, will take to reach the vertex (state) v_j for the first time [1]. It can be easily verified that the expected hitting time satisfies the following recurrence relations

$$\begin{cases} h(i|i) = 0, \\ h(j|i) = 1 + \sum_{k=1}^{n} p_{ik} h(j|k), \quad i \neq j. \end{cases} \tag{4.11}$$

Let h_0 be the vector with each entry being the expected hitting time $h(g|i)$ from any vertex $v_i, i \neq g$, to the ground vertex v_g and P_0 be the matrix obtained from the transition matrix P by removing the gth row and column. Then we can write (4.11) in a matrix form as $h_0 = \mathbf{1} + P_0 h_0$, which is equivalent to (4.10). We can see that the approximate solution of the isoperimetric cut problem given by the linear system (4.10) is the expected hitting times $h(g|i)$ from vertices $v_i \in V$ to the ground vertex v_g. From this expected hitting time viewpoint, we have some insights into the isoperimetric cut algorithm.

First, we can easily see that if the ground vertex v_g is selected such that for any other vertex $v_i, i \neq g$, there exists a path from v_i to v_g, then the linear system (4.10) is well posed, even if the graph is not strongly connected.

Second, we can examine the connectivity properties of the partitions obtained by thresholding $\mathbf{z}_0$ obtained from solving (4.10). We will prove that the partition containing the ground vertex (i.e., the set S) must be connected, regardless of how a threshold (i.e., cut) is chosen. The strategy for establishing this is that every vertex has a path to the ground vertex with a monotonically decreasing expected hitting time. Note that the partition not containing the ground vertex may or may not be connected.

Lemma 4.3.1. *For every vertex, v_i, there exists a path to the ground vertex $(v_i, v^1, v^2, \cdots, v_g)$, such that $z_i \geq z^1 \geq z^2 \cdots \geq 0$, when $L_0 \mathbf{z}_0 - \mathbf{1}$.*

Proof. By the definition of the expected hitting time, each non-grounded vertex has a value

$$z_i = 1 + \sum_{e_{ij} \in E} p_{ij} z_j. \tag{4.12}$$

For a vertex set $S \subseteq V$, denote the boundary vertex set of S as $S_b \subset V$, such that $S_b = \{v_j | e_{ij} \in E, \exists v_i \in S. v_j \notin S\}$. Then for any vertex v_i, we can explicitly explicitly construct a path to the ground vertex v_g with nonincreasing expected hitting time by the following procedure:
1) Start with $S = \{v_i\}$.
2) Repeat adding $v_j \in S_b$ to S such that $z_j < \min z_k$, $\forall v_k \in S$ by (4.12), until $v_g \in S_b$.
Step 2) is feasible, because for every vertex $v_k \in S$, there exists a path from v_k to v_g. □

Proposition 4.3.2. *If the set of vertices, V, is strongly connected, for any α, the subgraph with vertex set $M \subseteq V$ defined by $M = \{v_i \in V | z_i < \alpha\}$ is connected when z_0 satisfies $L_0 \mathbf{z}_0 = \mathbf{1}$.*

Proof. Since V is strongly connected, for any $v_g \in V$, $\forall v_i \in V, i \neq g$ there exists a path from v_i to v_g. Then from Lemma 4.3.1, all $v_j \in M$ are connected to v_g. Therefore the subgraph M is connected. □

The relationship between the expected hitting time and the isoperimetric problem also explains why the expected hitting time, as a proximity measure, performs very well in the ranking and retrieval tasks [66]. The small expected hitting time between the query and a sample in the database implies that they are likely of the same class in the clustering sense.

4.4 Experiments

In this section, we conduct experiments on a number of benchmark data sets to evaluate the proposed random walk isoperi-

metric cut (IsoCut). Five recent and related algorithms are compared to show the effectiveness of the IsoCut algorithm, including Kmeans, iterative normalized cut (NCut) [67], NJW [61], self-tuning graph construction based normalized cut (StN-Cut) [82], and the self-tuning graph construction based NJW (StNJW). The parameters in these algorithms are all tuned to ensure the best results in terms of the normalized mutual information evaluation. Furthermore, we analyze the computational efficiency of IsoCut algorithm compared with eigen-decomposition based approaches such as NCut, where the execution times of the algorithms with different numbers of samples and different numbers of neighborhoods are examined.

To evaluate the performances of the clustering algorithms, we compute the following two performance measures from the clustering results: normalized mutual information (NMI) and minimal clustering error (Error). NMI is defined as

$$\mathrm{NMI}(\mathbf{x}, \mathbf{y}) = \frac{I(\mathbf{x}, \mathbf{y})}{\sqrt{H(\mathbf{x})H(\mathbf{y})}},$$

where $I(\mathbf{x}, \mathbf{y})$ is the mutual information between $\mathbf{x}$ and $\mathbf{y}$, and $H(\mathbf{x})$ and $H(\mathbf{y})$ are the entropies of $\mathbf{x}$ and $\mathbf{y}$ respectively. Note that $0 \leq \mathrm{NMI}(\mathbf{x}, \mathbf{y}) \leq 1$ and $\mathrm{NMI}(\mathbf{x}, \mathbf{y}) = 1$ when $\mathbf{x} = \mathbf{y}$. The larger the value of NMI, the better a clustering results.

The clustering error is defined as the minimal classification error among all possible permutation mappings defined as:

$$\mathrm{Error} = \min(1 - \frac{1}{n}\sum_{i=1}^{n} \delta(\mathbf{y}_i, c_i)),$$

where $\mathbf{y}_i$ and c_i are the true class label and the obtained clustering result of $\mathbf{x}_i$, respectively, $\delta(\mathbf{x}, \mathbf{y})$ is the delta function that equals 1 if $\mathbf{x} = \mathbf{y}$ and 0 otherwise.

To validate the IsoCut algorithm on real image data set and demonstrate the superiority of the proposed algorithm compared

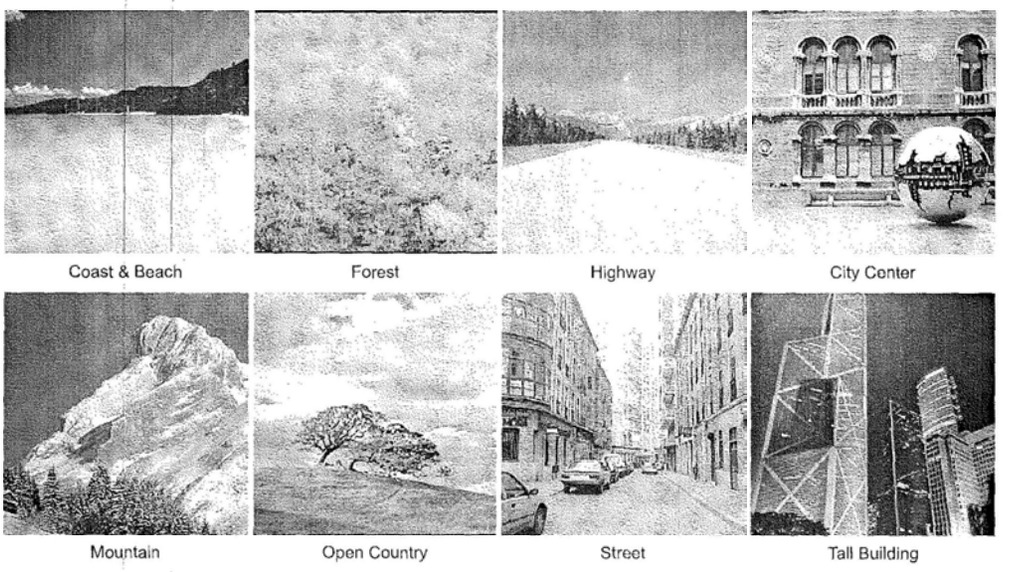


Figure 4.3: Example images in the Scene data set [63].

Table 4.1: Descriptions of the image data sets used in the experiments.

Dataset	clusters	dimensions	objects
Scene	8	512	2688
UMist-all	20	10304	575
UMist-10	10	10304	265
UMist-5	5	91	140
USPS-all	10	256	5000
USPS-5	5	256	2500

with the state-of-the-art related ones, we carry out experiments on three data sets. UMist, USPS. and a scene category data set (Scene). UMist consists of 575 multi-view face images of 20 different persons with varied poses from profiles to frontal views. USPS consists of 5000 images of 10 handwritten digits (0-9). To further exploit the databases, we randomly select 10 and 5 classes from UMist to construct two data sets UMist-10 and UMist-5, and use 5 digital numbers $(2, 3, 5, 6, 8)$ from USPS as another data set USPS-5 for the experiments. For UMist-5, the dimensions of the images are reduced by PCA while maintaining 99% of the total energy. The Scene data set was collected by Oliva and Torralba [63], containing 8 categories of natural scenes as shown in Fig. 4.3. We use the feature called Spatial Envelope [63] to represent each scene image, although other choices can be used. The feature is a 512-dimensional vector, capturing the dominant spatial structure of the scene. The description of the data sets used in our experiments are summarized in Table 4.1.

The clustering results by the six algorithms, Kmeans, NCut, NJW, StNCut. StNJW, and IsoCut, are shown in Table 4.3 and Table 4.4, from which we can see that IsoCut performs best in all the data consistently.

Five data sets (Iris, Wine, WDBC, Satimage, and Segment) from UCI Machine Learning Repository are used in this experiment, which are widely used to evaluate clustering algorithms. The five data sets that origin from the problems in different domains. More details of them are summarized in Table 4.2.

The comparison results are also listed in Table 4.3 and Table 4.4. Among all the 22 comparisons, the IsoCut$_1$ algorithm obtains the best results in 20 cases, and the second best results in another 2 cases. These comparisons demonstrate that IsoCut can achieve excellent performances consistently on real world applications with various numbers of clusters, samples, and dimensionalities.

Table 4.2 Descriptions of the UCI data sets used in the experiments.

Dataset	clusters	dimensions	objects
Iris	3	4	150
Wine	3	13	178
WDBC	2	30	569
Satimage	6	36	6435
Segment	7	19	2310

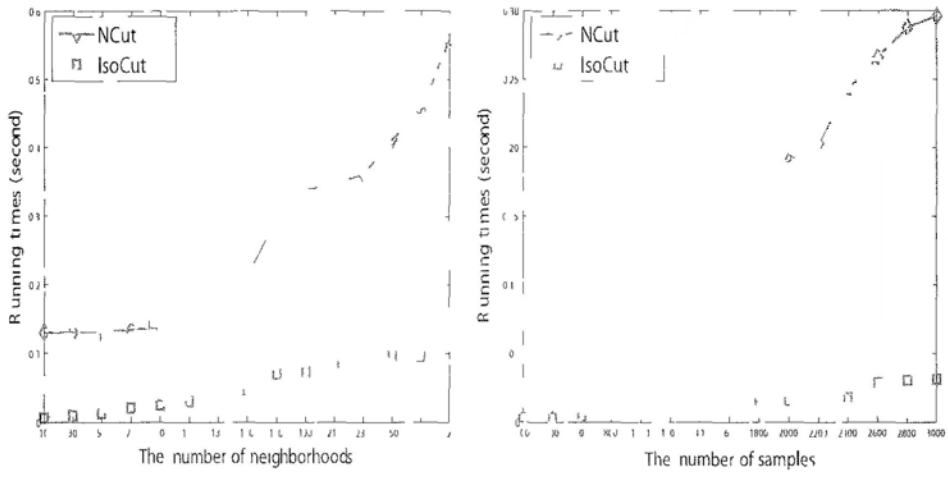


Figure 4.4· Running time comparisons between IsoCut and NCut

4.4.1 Computational Efficiency Analysis

In addition to the excellent performance of the IsoCut algorithm in accuracy, its computational efficiency is also an advantage over the related ones except Kmeans. Fig. 4.4 shows the running times of IsoCut and NCut, with respect to different numbers of samples and neighborhoods. NCut is a representative approach for eigen-decomposition based algorithms, whose computational complexity is similar to NJW, StNCut, and StNJW. From these results, we can see that the IsoCut algorithm is much faster than the other four algorithms, with much smaller time increasing than that with NCut as the numbers of samples and neighborhoods grow. The algorithms are implemented in Matlab, running on a 2.8 GHz Pentium IV PC with 4GB RAM.

Table 4.3: NMI comparison results on the ten real data sets. The best values are bold

Dataset	Kmeans	NCut	NJW	StNCut	StNJW	Ours
Iris	0.7582	0.7571	0.7661	0.6524	0.7857	**0.8449**
Wine	0.4288	**0.4624**	0.4351	0.3665	0.4199	0.4496
WDBC	0.4672	0.5754	0.5358	0.4679	0.4845	**0.5868**
Satimage	0.6138	0.6749	0.6373	0.6336	0.6307	**0.6932**
Segment	0.6124	0.6465	0.6629	0.5852	0.6801	**0.7440**
UMist-all	0.6726	0.6157	0.8009	0.5364	0.6512	**0.8785**
UMist-10	0.6161	0.5769	0.8214	0.4918	0.5850	**0.8634**
UMist-5	0.7065	0.8903	0.8655	0.6384	0.6371	**1**
USPS-all	0.4038	0.4517	0.5180	0.1894	0.3606	**0.6880**
USPS-5	0.4469	0.5789	0.4247	0.2536	0.3197	**0.6910**
Scene	0.3951	0.4100	0.4471	0.3605	0.4204	**0.4695**

4.5 Conclusions

In this chapter, we propose a kernel density estimation based directed graph clustering algorithm. A local kernel density estimation method with automatic bandwidth selection is proposed to construct the directed graph. This method effectively utilizes the local distribution information of the data. An efficient directed graph partitioning algorithm is also developed which optimizes the random walk isoperimetric ratio by solving a linear system. Experimental results show that the proposed method is superior to several popular methods on many benchmark data sets.

Viewing graph cut problem from the kernel density estimation prospective opens a door to solving the graph construction problem. Many nonparametric techniques can be utilized to boost the performance of graph based clustering algorithms.

□ **End of chapter.**

Table 4.4: Error comparison results on the ten real data sets. The best values are bold.

Dataset	Kmeans	NCut	NJW	StNCut	StNJW	Ours
Iris	0.1067	0.0933	0.1000	0.4867	0.0933	**0.0533**
Wine	0.2978	0.2697	0.2921	0 2921	0.2865	**0.2472**
WDBC	0.1459	0.0879	0.109	0.1388	0.1248	**0.0796**
Satimage	0.3310	0.2544	0.2457	0.2810	0.2737	**0.2197**
Segment	0.3342	0.4004	**0.2740**	0.5165	0.3407	0.2922
UMist-all	0.5339	0.5791	0.3948	0.6348	0.5739	**0.2661**
UMist-10	0.5509	0 5208	0.3057	0.5849	0.5547	**0.2604**
UMist-5	0.2214	0.0857	0.1214	0.3786	0.3071	**0**
USPS-all	0.6008	0.6404	0.4882	0 8396	0.6388	**0.3398**
USPS-5	0.3468	0 4140	0.4256	0.6224	0.4572	**0.2232**
Scene	0.5056	0.4835	0.4014	0.5443	0.4725	**0.3857**

Chapter 5

Digraph Multiway Cut via Hitting Time

Summary

In this chapter, we present a clustering algorithm which is based on random walk hitting time on directed graphs. Unlike traditional graph based clustering methods, we do not explicitly calculate the pairwise similarities between points. Instead, we form a transition matrix of random walk on a directed graph directly from the data. Our algorithm constructs the probabilistic dependence relations between sample pairs by studying the local distributions of the data. Based on the random walk model, we compute the expected hitting time for all sample pairs, which explores the global structure information of the underlying graph. A directed graph clustering algorithm based on expected hitting time is proposed. By utilizing the local distribution information of the data and the global structure information of the graph, our method is able to conquer some limitations of traditional pairwise similarity based methods.

5.1 Introduction

Recently, pairwise relation based clustering algorithms attract great attention. A successful example is the spectral clustering [54, 61, 67, 81]. These methods have the advantage that they do not make strong assumptions about the distribution of the data. Instead, similarities between sample pairs are first computed to construct an undirected graph of the data and then a global decision is made to partition all data points into disjoint sets according to some criterions. Therefore, these methods can potentially deal with data sets whose clusters are of irregular shapes.

Despite the great success of the graph based methods, there are still open problems: (1) How to construct the pairwise similarities between sample points to reflect the underlying distribution of the data; (2) How to deal with multi-scale data; (3) How to handle the data whose clusters are defined by geometry. Moreover, Nadler and Galun recently pointed out that there are fundamental limitations of these graph based approaches [60, 79]. According to their analysis, even with carefully tuned parameters, the spectral clustering algorithms still cannot successfully cluster the multi-scale data sets. They showed examples that the clusters, which can be easily captured by human, cannot be properly identified by the spectral clustering methods.

In this chapter, we show that the widely used parametric Gaussian kernel based similarities are not informative enough for modeling pairwise relations. As a result, the undirected graph constructed based on the similarities does not necessarily capture the intrinsic structure of the underlying data distribution. Therefore, the natural clusters of the data cannot be obtained by partitioning the graph.

From our analysis, we propose a data clustering algorithm based on a directed graph model. The edge weights of the graph

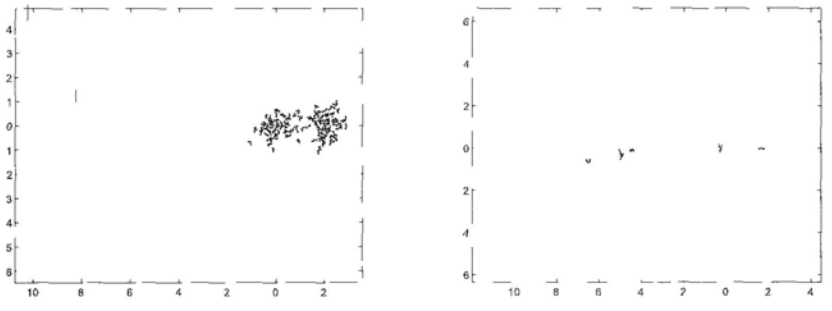
are the probabilistic dependence relations between local sample points, which are constructed by exploring the local distributions of the data. Such relations are asymmetric and more general than the similarities used in traditional undirected graph based methods since they consider both the local density and geometry of the data.

The probabilistic relations between all sample pairs naturally result in a stochastic matrix, which can be considered as the transition matrix of the Markov random walk process on a directed graph. Our new clustering algorithm works on this directed graph, which is based on the random walk model, more specifically the expected hitting time of random walk model [15, 17].

The random walk hitting time has a nice property: it decreases when the number of paths connecting two nodes increases and the length of any path decreases. Informally speaking, the shorter the paths connect two nodes are, the more related the two nodes are; strongly connected nodes are more related than weakly connected nodes.

There are some other applications that consider various measures based on random walk models. For example, the paper of [27] proposes an embedding method based on the commute time distance on undirected graphs for collaborative recommendation systems. Another paper of [10] proposes to use an angular based quantity for semi-supervised learning problems, which can be seen as a normalized version of the commute time on undirected graphs.

All these approaches are based on undirected graph models, with symmetric measures (similarity) between sample pairs. We will see later that symmetric measures cannot fully capture the relations between sample points. Sometimes it may even hinder the performance of the algorithms.



(a) Three Gaussians (1 1 1) (b) Three Gaussians (8 1 1)

Figure 5.1 Clustering results by the NJW algorithm on two multi-scale data sets. Different clusters are denoted by different colors.

5.2 Limitations of Pairwise Similarity Based Methods

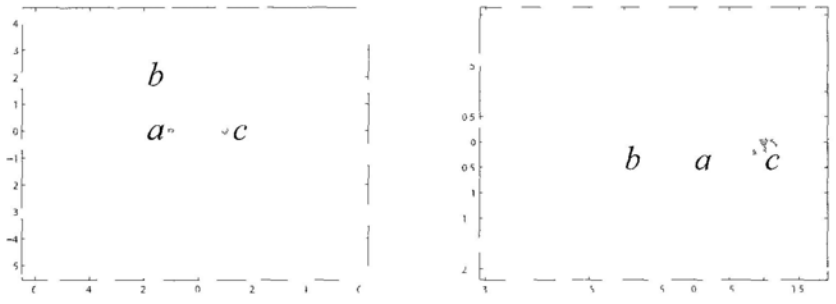
The first step of pairwise relation based algorithms is to construct an undirected graph for the vector data. Sample points are connected by undirected edges. The edge weights reflect the similarities between sample pairs. Usually a Gaussian kernel $\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$ with a manually adjusted parameter σ is used for setting the weights. A problem of this step is how to choose the parameter σ. When it is not properly set, the clustering results can be poor. A more severe problem is that a single σ for all sample pairs implies that if the Euclidean distances between two pairs are the same, the two similarities are the same too. When the input data are with different density and geometry, there may not exist a single value of σ that works well for the whole data set. For certain data set, the intrinsic cluster structure essentially may not be explored by the spectral clustering algorithm, no matter what value of σ is chosen.

Figures 5.1a and 5.1b are two multi-scale data sets from [59], where 1000 sample points are generated by three Gaussians with variances $\sigma_1 = 2$ and $\sigma_2 = \sigma_3 = 0.5$. The point numbers of the

Gaussians are 1:1:1 for Figure 5.1a and 8:1:1 for Figure 5.1b. The best results that can be achieved by the spectral clustering are shown with different colors denoting the clusters. As indicated in [59], these multi-scale problems essentially cannot be solved by spectral clustering, mainly because it does not explore the density information inherent in the data. Even though the parameter σ has been carefully tuned, from the figure we can see that the spectral clustering algorithm cannot obtain satisfactory results on these data sets.

When dealing with these kinds of data, exploring the local data distribution is very important. Consider the data shown in Figure 5.2a. The Euclidean distances $d(a, b)$ and $d(a, c)$ between the sample pairs (a, b) and (a, c) are the same. Then the similarities computed by the Gaussian kernel are the same. However with the context data points around sample a, apparently a is more likely to belong to the same cluster as b than as c. Here the geometric shape of the local data distribution is important for modeling the relations between sample pairs. Another example in Figure 5.2b shows the importance of the density of the local data distribution that affects the relations between sample pairs. Although sample a lies in the middle of b and c, a is more likely to have the same class label as c than as b when considering the density of the context data. Here the local density of the data distribution is important for modeling the relations between the sample pairs.

These two intuitive examples suggest that we should analyze the local data distribution when modeling the pairwise relations between sample points.



(a) Shape (b) Density

Figure 5.2 The local density and shape affect the relations between sample pairs

5.3 Random Walk Hitting Time Based Digraph Clustering

Based on the analysis above, we propose to study the local context of the data to setup the relations between local sample points. The relations between sample pairs are not necessarily symmetric. We adopt a probabilistic framework to model the local pairwise relations to form a directed graph, and then compute the random walk hitting time between all sample pairs which explores the global information of the structure of the underlying directed graph. An iterative algorithm called K-destinations is proposed to cluster the data based on the hitting time measure.

5.3.1 Local Gaussian based Bayesian inference

Let $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, be points that we wish to assign to K clusters $\Omega = \{V_l\}_{l=1, \ldots, K}$. The good performance of a clustering method indicates that the label of a data point can be well estimated based on its neighbors.

The data of a local neighborhood can be deemed as a s-

ingle Gaussian. Denote $N(i)$ as the index set of $\mathbf{x}_i$'s neighbors. In this paper, the k nearest neighbors of $\mathbf{x}_i$ are used to compose the neighborhood $N(i)$. Each sample $\mathbf{x}_j$, $j \in N(i)$, can be thought to lie in a local Gaussian centered at $\mathbf{x}_i$, i.e., $\mathbf{x}_j \sim \mathcal{G}(\mathbf{x}_i, C_i)$, $j \in N(i)$, where $\mathbf{x}_i$ and C_i are the mean and covariance of this Gaussian. The covariance C_i of the Gaussian can be estimated using the data of the neighborhood $N(i)$ by the maximal likelihood estimation (MLE). Let $X_i = [\mathbf{x}_j]_j$, $j \in N(i)$, be a matrix with each column being a neighbor of $\mathbf{x}_i$. A regularized covariance matrix C_i of the local Gaussian distribution $\mathcal{G}(\mathbf{x}_i, C_i)$ is

$$C_i = \frac{1}{|N(i)|}(X_i - \mathbf{x}_i\mathbf{1}^T)(X_i - \mathbf{x}_i\mathbf{1}^T)^T + \alpha I,$$

where $|N(i)|$ is the cardinality of the set $N(i)$, α is the regularization factor, $\mathbf{1}$ is a vector with all entries being 1, and I is the identity matrix [35].

Let $\widehat{C}_i = (X_i - \mathbf{x}_i\mathbf{1}^T)(X_i - \mathbf{x}_i\mathbf{1}^T)^T/|N(i)|$. In this paper, we use a modified version of the regularized covariance matrix proposed in [71]:

$$C_i = \widehat{C}_i + \frac{tr(\widehat{C}_i)}{d} I.$$

We write $\mathcal{G}_i$ as the abbreviation of $\mathcal{G}(\mathbf{x}_i, C_i)$. Then for a sample point $\mathbf{x}_j$, the multivariate Gaussian density, with which $\mathbf{x}_j$ is generated by the Gaussian $\mathcal{G}(\mathbf{x}_i, C_i)$, is given by

$$p(\mathbf{x}_j|\mathcal{G}_i) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^T C_i^{-1}(\mathbf{x}_j - \mathbf{x}_i)\right)}{\sqrt{(2\pi)^d |C_i|}}. \tag{5.1}$$

As shown in Figure 5.3a, given the neighbor Gaussians, the probability that $\mathbf{x}_j$ is generated by the Gaussian $\mathcal{G}(\mathbf{x}_i, C_i)$ can be computed by the Bayesian rule:

$$P(\mathcal{G}_i|\mathbf{x}_j) = \frac{p(\mathbf{x}_j|\mathcal{G}_i)P(\mathcal{G}_i)}{\sum_{i \in N(j)} p(\mathbf{x}_j|\mathcal{G}_i)P(\mathcal{G}_i)}.$$

For simplicity, the prior probabilities are set equal.

$P(\mathcal{G}_i|\mathbf{x}_j)$ can be thought as the local dependence of $\mathbf{x}_j$ on $\mathbf{x}_i$ given the context of local data distributions when determining the cluster membership of each sample, as shown in Figure 5.3a. It represents the dependence of $\mathbf{x}_j$ on $\mathbf{x}_i$. Also denote $p_{ji} = P(\mathcal{G}_i|\mathbf{x}_j)$, then $\sum_i p_{ji} = 1$. Notice that the probabilistic relations between points i and j are not asymmetric, i.e., in general, p_{ji} is not necessarily equal to p_{ij}. Then all samples and the asymmetric relations between sample pairs naturally result in a directed graph.

The advantage of using the local Gaussian based Bayesian inference to model the dependence relations between sample pairs can be seen from Figure 5.3. When solely using the Euclidean distances to model the pairwise relations, the clustering boundary may not be reasonable as shown in Figure 5.3b. By considering the local distribution, we can obtain a satisfactory boundary as shown in Figure 5.3c. Using this approach, we avoid setting the bandwidth parameter σ of the Gaussian kernel in the spectral clustering methods. Moreover, this estimation of local relations considers both the local density and geometry of the data, thus the constructed graph reflects the underlying distribution of the data set.

5.3.2 Random Walk Hitting Time

For all sample pairs, we have the matrix $P = [p_{ij}]_{ij}$, which has the property that $P\mathbf{1} = \mathbf{1}$, i.e., P is stochastic. After obtaining the stochastic matrix P, we can naturally define the Markov random walk on the directed graph associated with P. The random walk is defined with the single-step transition probability p_{ij} of jumping from any node (state) i to one of its adjacent nodes j where $p_{ij} = P[i \to j]$ is the probability of one step discrete time random walk transition from i to j. Then P can be regarded as

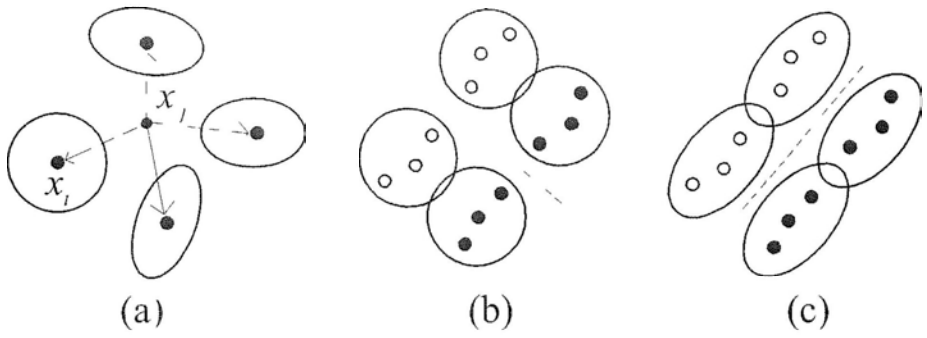


Figure 5 3. Advantage of using local Gaussian based Bayesian inference to construct neighbor relations (a) Local Gaussian based Bayesian inference. (b) Boundary found by modeling pairwise relations with the isotropic Gaussian kernel (c) Boundary found by modeling pairwise relations with local Gaussian estimation

the transition probability matrix of the random walk process.

The transition probabilities depend only on the current states (first-order Markov chain). If the directed graph associated with the matrix P is strongly connected, the Markov chain is irreducible, that is, every state can be reached from any other states. If this is not the case, the Markov chain can be decomposed into closed subsets of states which are independent (there is no communication between them), each closed subset is irreducible. and the procedure can be applied independently to these closed subsets.

The expected hitting time $h(j|i)$ of the random walk is the expected number of steps before node j is visited starting from node i. It can be easily verified that the hitting time satisfies the following recurrence relations

$$\begin{cases} h(i|i) = 0 \\ h(j|i) = 1 + \sum_{k=1}^{n} p_{ik} h(j|k) \quad i \neq j. \end{cases}$$

The recurrence relations can be used in order to iteratively compute the expected hitting time. The meaning of these formulae is quite obvious: in order to jump from node i to node j, one

has to go to any adjacent state k of node i and proceeds from there.

The closed form of the hitting time in terms of transition matrix exists [1]. By introducing a matrix

$$Z = (I - (P - \mathbf{1}\pi^T))^{-1},$$

the matrix H with its entry $H_{ij} = h(j|i)$ can be computed by

$$H_{ij} = (Z_{jj} - Z_{ij})/\pi_j,$$

where Z_{ij} is the entry of Z.

As mentioned before, the hitting time from node i to node j has the property of decreasing when the number of paths from i to j increases and the lengths of the paths decrease [23]. This is a desirable property for representing the dependence of the label of one point on another for the clustering task when the global distribution of the data is taken into account.

A closely related quantity, the commute time $c(i, j)$, is defined as the expected number of steps that a random walker, starting from node $i \neq j$, takes to meet node j for the first time and goes back to i. That is, $c(i, j) = h(j|i) + h(i|j)$. The commute time distance is also known as the resistance distance in the electrical literature [46, 23]. The commute time distance on undirected graphs is widely used in many applications [10, 27, 72]. However we argue that it is not suitable for our case. In the case of a directed graph, if the hitting time from node i to node j is small, which means node i and node j are tightly related, but the hitting time from node j to node i is not necessarily small. Such cases often happen on the points that lie on the boundaries of clusters, which have short hitting times to the central points in the same clusters, but often have very large hitting times from the central points to points close to the boundaries. So in this paper, we use the hitting time instead of commute time as the measure of pairwise relations.

Algorithm 2 Random walk hitting time based digraph clustering algorithm

Require: The data matrix X the number of nearest neighbors k, and the number of clusters K

for all i, i $1, 2,$ $, n$ **do**

Form X_i $[\mathbf{x}_j]_j$. $j \in N(i)$, by the k nearest neighbors of $\mathbf{x}_i$

Compute the m left singular vectors $\widehat{U}_i = [u_{i_1},$ $. u_{i_m}]$ corresponding to the non-zero singular values of $\widehat{X}_i = X_i$ $\mathbf{x}_i \mathbf{1}^T$

For those j with $i \in N(j)$, compute $p(\mathbf{x}_j|\mathcal{G}_i)$ as in (5 2)

end for

Compute $P = [p_{ij}]_{ij}$ with each entry $p_{ij} = p(\mathcal{G}_j|\mathbf{x}_i)/\sum_{j \in N(i)} p(\mathbf{x}_i|\mathcal{G}_j)$.

Compute $Z = (I$ $P - \epsilon \pi^T)^{-1}$ and compute H where $H_{ij} = (Z_{jj} - Z_{ij})/\pi_j$.

Perform the hitting time based K-destinations algorithm.

5.3.3 K-destinations Algorithm

After obtaining the asymmetric hitting times between all sample pairs, we are ready to apply a clustering algorithm to categorize the data into disjoint classes. Since traditional pairwise relation based clustering algorithms often require the pairwise relations be symmetric and the similarity functions be semi-definite, such as the spectral clustering [61, 67, 81], they are not suitable for clustering using the hitting time measure.

In this work, we develop an iterative clustering algorithm based on the asymmetric hitting time measure, which is similar to the K-means algorithm, called the K-destinations that directly works on the pairwise hitting time matrix H.

Each cluster V_l is represented by an exemplar v_l, which is called a destination node in our algorithm. The destination node is selected from the samples. Intuitively, we want to choose the destinations that save the walkers' (hitting) time. Therefore, we propose to cluster the data by minimizing the sum of the hitting times from the samples to the destination node in each cluster:

$$J - \sum_{l=1}^{K} \sum_{i \subset V_l} h(v_l|i).$$

Finding the global optimum of this criterion is a hard problem. Instead, we optimize the function in a greedy manner. Similar to the K-means algorithm, we iteratively minimize J by two steps:

- First, we fix the destination nodes and assign each sample to the cluster that has minimal hitting time from it to the destination node corresponding to the cluster.
- Then, in each cluster, we update the destination node from the samples that minimize the sum of the hitting times from all samples in the cluster to the destination node.

The clustering algorithm repeats the two steps until the cluster membership of each sample does not change. It can be seen that the algorithm monotonously decreases the value of J in each iteration, so the convergence of the algorithm is guaranteed.

5.3.4 Implementation

In typical applications of our algorithm such as image clustering, the dimension d of the data can be very high. Therefore the computation of the Gaussian density in (5.1) is time consuming, where the inverse of $d \times d$ matrices is involved. Usually in the neighborhood, although the covariance matrix $C = \widehat{C} + \alpha I$ is of rank d, the rank of $\widehat{C}$ is very low[1]. Denoting the rank of $\widehat{C}$ as m, we have $m < k \ll d$, where k is the number of neighbors in the neighborhood. Let $U = [u_i]_i$, $i = 1, \ldots, d$, and $\Lambda = \mathrm{dg}([\lambda_i]_i)$, $i = 1, \ldots, d$, where u_i and λ_i are the eigenvectors and eigenvalues of $\widehat{C}$ respectively, then

$$\begin{aligned} C^{-1} &= U(\Lambda + \alpha I)^{-1} U^T \\ &= U(A + \frac{1}{\alpha} I) U^T, \end{aligned}$$

[1]Without ambiguity, here we ignore the subscript i in C_i in this subsection

where A is a diagonal matrix with entries

$$A_{ii} = -\frac{\lambda_i}{\alpha(\lambda_i + \alpha)}.$$

Let $\widehat{U} = [u_i]_i$, $i = 1, \ldots, m$, be a matrix formed by the eigenvectors corresponding to non-zero eigenvalues of $\widehat{C}$, which can be efficiently computed by applying singular value decomposition on a $d \times k$ matrix (see Table 2). Then the Mahalanobis term in (5.1) is

$$\begin{aligned} d_C(\mathbf{x}_j, \mathbf{x}) =& (\mathbf{x}_j - \mathbf{x})^T C^{-1} (\mathbf{x}_j - \mathbf{x}) \\ & - \|\widehat{A}^{1/2} \widehat{U}^T (\mathbf{x}_j - \mathbf{x})\|^2 + \|\mathbf{x}_j - \mathbf{x}\|^2 / \alpha, \end{aligned}$$

where $\widehat{A} = \mathrm{dg}([A_{ii}]_i)$, $i = 1, \ldots, m$, is only a $m \times m$ matrix with $m << d$. Then the density with which $\mathbf{x}_j$ is generated by $\mathcal{G}(\mathbf{x}, C)$ can be computed as

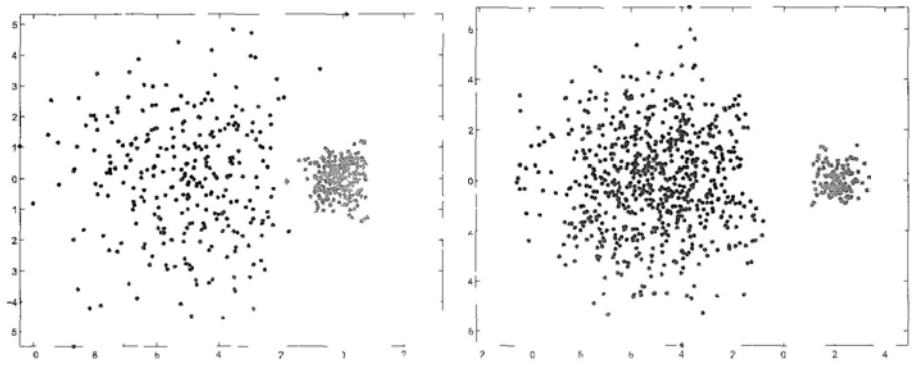
$$p(\mathbf{x}_j | \mathcal{G}) = \exp\left(-\frac{1}{2} \big(d_C(\mathbf{x}_j, \mathbf{x}) + \sum_{i=1}^{m} \ln \lambda_i + d \ln (2\pi) \big) \right). \quad (5.2)$$

Thus we avoid storing the $d \times d$ matrix C and explicitly computing the inverse of it, which brings us time/space efficiency and numerical stability. The complete algorithm is summarized in Table 2.

5.4 Experiments

In this section, we present the clustering results obtained by the proposed random walk hitting time based digraph clustering (HDC) algorithm on a number of synthetic and real data sets. We also compare our HDC algorithm with the K-means and the NJW [61] algorithms.

In order to show the advantage of our HDC algorithm, we apply it to the data sets shown in Figure 5.1, on which the NJW



(a) Three Gaussians (1 1 1) (b) Three Gaussians (8 1 1)

Figure 5.4 Clustering results by HDC on the two data sets. Different clusters are denoted by different colors.

algorithm fails. Our clustering results are given in Figure 5.4. From the figure we can see that, by exploring both the local distribution information of the data and the global structure information of the graph, the HDC algorithm can work well on the multi-scale data sets. The clustering results satisfactorily capture the natural cluster structures of the data.

In this section, we conduct experiments on the following real data sets:

- Iris: The data are from the UCI repository comprising 3 classes of 50 instances each, where each class refers to a type of iris plant.
- Wine: The data are from the UCI repository comprising 3 different wines. This data set is used for chemical analysis to determine the origin of wines.
- Satimage: The data are the 10% sampling of the UCI repository Landsat Satellite, which consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image.
- Ionosphere: The data are from the UCI repository referring

to the radar returns from the ionosphere that is a binary clustering to detect "Good" radar.

- Segmentation: The data are from the UCI repository. The instances are drawn randomly from a database of seven outdoor images.
- WDBC: The data are from the UCI repository that is used for diagnostic Wisconsin Breast Cancer.
- UMist-5: The data are from UMist database containing 5 classes, which are randomly selected from all the face images of 20 different persons in UMist database. The dimension of the data are reduced by principle component analysis (PCA) while maintaining 99% of the total energy.

More details of the data sets are summarized in Table 5.1.

Table 5.1 Descriptions of the data sets used in the experiments

Dataset	K	d	n
Iris	3	4	150
Wine	3	13	178
Satimage	6	36	644
Ionosphere	2	34	351
Segmentation	7	19	2310
WDBC	2	30	569
UMist-5	5	91	140

To evaluate the performances of the clustering algorithms, we compute the following two performance measures from the clustering results: normalized mutual information (NMI) and minimal clustering error (Error). The NMI is defined as

$$\mathrm{NMI}(\mathbf{x}, y) = \frac{I(\mathbf{x}, y)}{\sqrt{H(\mathbf{x})H(y)}}, \tag{5.3}$$

Table 5 2 Error and NMI comparison results on seven data sets The best values are bold

Dataset	Error			NMI		
	K-means	NJW	HDC	K means	NJW	HDC
Iris	0 1067	0 1000	**0.0267**	0 7582	0 7661	**0.8981**
Wine	0 2978	0 2921	**0.2865**	0 4288	0 4351	**0.4544**
Satimage	0 3323	0 2888	**0.2298**	0 6182	0 6693	**0.7039**
Ionosphere	0 2877	0 1510	**0.1266**	0 1349	0 4621	**0.5609**
Segmentation	0 3342	0 2740	**0.2521**	0 6124	0 6629	**0.7039**
WDBC	0 1459	0 1090	**0.1072**	0 4672	**0.5358**	0 5035
UMist-5	0 2214	0 1214	**0.0643**	0 7065	0 8655	**0.8930**

where $I(\mathbf{x}, y)$ is the mutual information between $\mathbf{x}$ and y, and $H(\mathbf{x})$ and $H(y)$ are the entropies of $\mathbf{x}$ and y respectively Note that $0 \leq \mathrm{NMI}(\mathbf{x}. y) \leq 1$ and $\mathrm{NMI}(\mathbf{x}, y) = 1$ when $\mathbf{x} = y$ The larger the value of NMI is, the better a clustering result is

The clustering error is defined as the minimal classification error among all possible permutation mappings defined as

$$\mathrm{Error} = \min(1 - \frac{1}{n}\sum_{i=1}^{n} \delta(y_i, \mathrm{perm}(c_i))), \tag{5 4}$$

where y_i and c_i are the true class label and the obtained clustering result of $\mathbf{x}_i$, respectively, $\delta(\mathbf{x}, y)$ is the delta function that equals 1 if $\mathbf{x} = y$ and 0 otherwise

The clustering results by the three algorithms, K-means, NJW, and HDC, are summarized in Table 5 2 The HDC algorithm obtains the smallest errors in all the cases, and produces the largest NMI values on all the data sets except one These results demonstrate that the HDC can achieve good performances consistently on various real world applications

5.5 Conclusions

We have proposed a random walk hitting time based digraph clustering algorithm for general data clustering. The pairwise relations of probabilistic dependence of the data are obtained by local distribution estimation. A directed graph is constructed based on the asymmetric relations. Then the hitting time measure is computed based the Markov random walk model on the directed graph, which explores the global graph structure. An iterative algorithm is also proposed to work with the asymmetric hitting time measure to cluster the data. Our algorithm is able to conquer some limitations of traditional pairwise similarity based methods. Extensive experiments have shown that convincing results are achieved in both synthetic and real world data by our algorithm.

□ **End of chapter.**

Chapter 6

Conclusions and Contributions

Clustering is one of the most widely used techniques for exploratory data analysis. Despite the popularity of clustering, clustering is a difficult problem. Little is known about theoretical properties of clustering. One of the main reasons is that it is very difficult to evaluate the quality of a partition of some given data set. There is no general agreement for what criterion should be optimized in order to cluster general data. There is no theoretical ground to justify what is the right way to do clustering.

Graph based clustering algorithms are of great interests recently. These methods first compute the pairwise similarities of the data to construct an undirected graph. Then the global clustering result is obtained by partitioning the vertexes of the graph into disjoint sets according to some criterion. One advantage of these methods is that they do not make strong assumptions about the global distribution of the data. Therefore, they can potentially deal with data of irregular shapes. Despite the success of the graph based methods, there are still unsolved theoretical and practical issues: What is the statistical meaning of spectral clustering objective? How to construct the graph in a data-dependent way in order to obtain robust results across data sets?

Bayesian decision theory is "THE" fundamental theory for

decision making under uncertainty. It provides a mathematical tool to guide deriving practical models to analyze data. A model which is consistent with Bayesian decision theory is essentially consistent with probability theory and vice versa. The decision theory for supervised learning problems, such as classification and regression, are well established. With the guidance of the theory, the area of supervised learning is flourishing. Many theory conformal algorithms were developed which are proved working well in practice. However, to the best of our knowledge, there is still lack of such a theoretical foundation for clustering problems. As a result, there is no theoretical criterion to distinguish which algorithms are "better" or even doing the "right" thing. Clustering remains an art rather than science at current stage.

In this thesis we start from the very beginning, the Bayesian decision theory, to establish a theoretical framework for the clustering problems. With the theoretical framework in hand, we interpret the popular graph based clustering methods from the perspective of decision theory. Also motivated by the theory we extend current algorithm to deal with data with more complex structures. We also develop several new algorithms to effectively and efficiently solve the clustering problems arisen from different situation.

The contributions we made in this thesis are as follow

- Based on the Bayesian decision theory, we develop the theoretical foundation of general clustering problem. The clustering problem is treated as a classification problem without training data. Reinterpreted by the probability language, we model the clustering problems as minimizing the expected classification error with respect to the partitioning of the sample space.

- We show that the seemingly unrelated two steps (graph

construction and graph partitioning) of the graph based methods are actually related. Especially, we show that the normalized cut is a nonparametric clustering method which adopts the kernel density estimator as its density model and tries to minimize the expected classification error or Bayes risk with respect to all possible partitioning of the input data.

- By adopting the Bayesian density theoretical view, we propose several extensions of current spectral clustering methods. Several graph construction approaches are proposed to construct the graph in data-dependent ways by using different density estimation methods, such as variable bandwidth kernel density estimators. The advantage of these methods is that the parameters for constructing the graph can be estimated from the data. The constructed graph effectively explores the intrinsic distribution of the data.

- Using the flexible density models can result in directed graphs which cannot be handled by the normalized cut method. We then propose more general algorithms which can deal with both undirected and directed graphs in a unified way. An eigendecomposition based clustering algorithm is proposed which generalizes the traditional spectral clustering to work on both directed and undirected graph. We also develop the isoperimetric cut algorithm to cut directed graphs by solving linear systems which is computationally more efficient. A random walk hitting time based algorithm is also proposed to solve the multiway graph cut problems.

Three clustering algorithms are developed in this thesis. These algorithms are designed for the clustering problems arising from different scenarios. They make trade-off between clustering performance and computational speed. The hitting time algorith-

m is suitable for clustering the data sets with limited samples. In general, it can achieve very good performance across data sets. However the computational demand of this algorithm is large, which prevents applying it on large scale data sets. The isoperimeteric cut algorithm on the other hand is very efficient. It solves a system of linear equations which can scale up to very large data sets. However, the constructed graph in isoperimetric cut algorithm is not so informative and the multiple cluster structure of the graph is not explored. The spectral algorithm which generalizes the traditional spectral clustering algorithm achieves a good balance between the performance and speed. It can be widely used for all kinds of clustering tasks

Although the basic Bayesian decision theoretical framework for clustering problems is established, the work on theoretically modeling clustering problems is still very preliminary. Much more works need to be done along this line. For example, use the same methodologies as the statistical learning theory [75, 76] or PAC learning theory [73] for supervised learning, we can develop corresponding theory for clustering problems. These theories in essence are to develop the lower bound for the Bayes risk with respect to arbitrary underlying distribution from which the data are generated. With the Bayesian decision theory for clustering, we are ready to develop these frequentist lower bounds for clustering problem. The decision theory established in this thesis also can be extended to model the problem of semi-supervised learning problems, which is another popular area lacking of theoretical foundation.

Starting from the Bayesian decision theory for clustering problems, we can also develop other algorithms for specific applications where more strong assumptions can be made. The density view of graph based clustering methods also motivates us to develop more sophistries methods for constructing graphs. For example, we can use Bayesian nonparametric methods [58, 29],

such as Dirichlet process mixture model [25] to build the graphs. In such a method, the model complexity is controlled by optimizing the model evidence, the parameters of bandwidth and number of mixture components can be automatically learned from the data.

□ **End of chapter.**

Bibliography

[1] D. Aldous and J. Fill. *Reversible Markov chains and random walks on graphs*. 2002.

[2] F. Bach and M. Jordan. Learning spectral clustering. *Advances in neural information processing systems*, 2003.

[3] S. Barnard, A. Pothen, and H. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical linear algebra with applications*, 2(4):317–334, 1995.

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

[5] M. Belkin and P. Niyogi. Problems of learning on manifolds. *The University of Chicago*, 2003.

[6] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *Learning theory*, pages 486–500, 2005.

[7] Y. Bengio, O. Delalleau, N. Roux, J. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16(10):2197–2219, 2004.

[8] C. Bishop. *Pattern recognition and machine learning*. Springer New York., 2006.

[9] M. Bolla. *Relations between spectral and classification properties of multigraphs.* DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, 1991.

[10] M. Brand. A random walks perspective on maximizing satisfaction and profit. *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.

[11] P. Bremaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. springer verlag, 1999.

[12] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994.

[13] O. Chapelle, B. Sch
"olkopf, and A. Zien. *Semi-supervised learning*, volume 2. Citeseer, 2006.

[14] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in Analysis*, pages 195–199, 1970.

[15] M. Chen, J. Liu, and X. Tang. Clustering via random walk hitting time on directed graphs. *Proceedings of Twenty-Third Conference on Artificial Intelligence*, pages 616–621, 2008.

[16] M. Chen, J. Liu, and X. Tang. Isoperimetric cut on a directed graph. *CVPR*, 2010.

[17] M. Chen, Q. Yang, and X. Tang. Directed graph embedding. *Proceedings of the Twentieth International Joint Conference on Artificial Intellegence*, pages 2707–2712, 2007.

[18] F. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.

[19] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. pages 269–274, 2001.

[20] I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

[21] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, pages 36-61, 1991.

[22] W. Donath and A. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

[23] P. Doyle and J. Snell. Random walks and electric networks. 1984.

[24] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000.

[25] T. Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230, 1973.

[26] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[27] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355-369, 2007.

[28] A. Gelman. H. Stern, and D. Rubin. *Bayesian Data Analysis*. Boca Raton, Florida. Chapman and Hall/CRC, 2004.

[29] J. Ghosh and R. Ramamoorthi. *Bayesian nonparametrics.* Springer Verlag, 2003.

[30] E. Gine and V. Koltchinskii. Empirical graph laplacian approximation of laplace-beltrami operators: large sample results. *Lecture Notes-Monograph Series*, 51:238–259, 2006.

[31] L. Grady and E. Schwartz. Isoperimetric graph partitioning for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):469–475, 2006.

[32] L. Grady and E. Schwartz. Isoperimetric partitioning: A new algorithm for graph partitioningi. *SIAM Journal on Scientific Computing*, 27:1844, 2006.

[33] S. Guattery and G. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.

[34] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(9):1074 -1085, 1992.

[35] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001.

[36] M. Hein. Uniform convergence of adaptive graph-based regularization. *Learning Theory*, pages 50–64, 2006.

[37] M. Hein, J. Audibert, and U. Von Luxburg. From graphs to manifolds–weak and strong pointwise consistency of graph laplacians. *Learning theory*, pages 470–485, 2005.

[38] M. Hein, J. Audibert, and U. von Luxburg. Graph laplacians and their convergence on random neighborhood

graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

[39] M. Hein, J. Audibert, and U. von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *The Journal of Machine Learning Research*, 8:1325–1370, 2007.

[40] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[41] M. Hollander and D. Wolfe. *Nonparametric statistical methods*. Wiley-Interscience, 1999.

[42] T. Joachims. Transductive learning via spectral graph partitioning. 20(1):290, 2003.

[43] J. Jost. *Riemannian geometry and geometric analysis*. Springer New York, 1995.

[44] R. Kannan and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[45] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. pages 561–568, 2004.

[46] D. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.

[47] Y. Koren. Drawing graphs by eigenvectors: theory and practice*. *Computers & Mathematics with Applications*, 49(11-12):1867 -1888, 2005.

[48] S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, 2004.

[49] L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[50] H. Lutkepohl. *Handbook of matrices.* John Wiley & Sons Inc, 1996.

[51] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[52] D. MacKay. *Information theory, inference, and learning algorithms.* Cambridge Univ Press, 2003.

[53] G. McLachlan and D. Peel. *Finite mixture models.* Wiley-Interscience, 2000.

[54] M. Meila and J. Shi. A random walks view of spectral segmentation. *Eighth International Conference on Artificial Intelligence and Statistics*, 2001.

[55] B. Mohar. Isoperimetric numbers of graphs. *Journal of Combinatorial Theory Series B*, 47(3):274–291, 1989.

[56] B. Mohar. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2:871–898, 1991.

[57] B. Mohar. Some applications of laplace eigenvalues of graphs. *Graph symmetry: algebraic methods and applications*, 497:227–275, 1997.

[58] P. Muller and F. Quintana. Nonparametric bayesian data analysis. *Statistical science*, 19(1):95–110, 2004.

[59] B. Nadler and M. Galun. Fundamental limitations of spectral clustering. *Advances in Neural Information Processing Systems*, 2007.

[60] B. Nadler and M. Galun. Fundamental limitations of spectral clustering. *Advances in Neural Information Processing Systems*, 19:1017, 2007.

[61] A. Ng, M. Jordan, and Y Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2001.

[62] J. Norris. *Markov chains.* Number 2008. Cambridge Univ Pr, 1998.

[63] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *ICCV*, 2001.

[64] A. Pothen, H. Simon, K. Liou, et al. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. MATRIX ANAL. APPLIC.*, 11(3):430–452, 1990.

[65] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. *Machine Learning: ECML 2004*, pages 371–383. 2004.

[66] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. *ICML*, 2008.

[67] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 22(8), 2000.

[68] B. Silverman. *Density estimation for statistics and data analysis.* Chapman & Hall/CRC, 1986.

[69] H. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2-3):135–148, 1991.

[70] D. Spielmat and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. page 96, 1996.

[71] S. Srivastava and M. Gupta. Distribution-based bayesian minimum expected risk for discriminant analysis. *IEEE International Symposium on Information Theory*, pages 2294–2298, 2006.

[72] M. Szummer, T. Jaakkola, and M. Cambridge. Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference*, 2002.

[73] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[74] R. Van Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21(1):29 48, 1995.

[75] V. Vapnik. Statistical learning theory. 1998.

[76] V. Vapnik. *The nature of statistical learning theory.* Springer Verlag, 2000.

[77] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[78] U. Von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Annals of Statistics*, 36(2):555, 2008.

[79] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. *Advances in Neural Information Processing Systems*, 2005.

[80] D. Wagner and F. Wagner. Between min cut and graph bisection. *Mathematical Foundations of Computer Science 1993*, pages 744–750, 1993.

[81] S. Yu and J. Shi. Multiclass spectral clustering. *Proceedings of Ninth IEEE International Conference on Computer Vision*, pages 313–319, 2003.

[82] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 2005.

[83] D. Zhao, Z. Lin, and X. Tang. Contextual distance for data perception. *ICCV*, 2007.