EXPLORING THE LANDSCAPE OF CLUSTERINGS

by

Parasaran Raman

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

December 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Parasaran Raman**

has been approved by the following supervisory committee members:

Suresh Venkatasubramanian	, Chair	October 24, 2013
		Date Approved
Thomas Fletcher	, Member	October 24, 2013
		Date Approved
Feifei Li	, Member	October 24, 2013
		Date Approved
Jeff Phillips	, Member	October 24, 2013
		Date Approved
Thomas Seidl	, Member	November 7, 2013
		Date Approved

and by **Alan L. Davis** , Chair of the Department of **School of Computing**

and by **David B. Kieda** , Dean of the Graduate School.

ABSTRACT

With the tremendous growth of data produced in the recent years, it is impossible to identify patterns or test hypotheses without reducing data size. Data mining is an area of science that extracts useful information from the data by discovering patterns and structures present in the data. In this dissertation, we will largely focus on clustering which is often the first step in any exploratory data mining task, where items that are similar to each other are grouped together, making downstream data analysis robust.

Different clustering techniques have different strengths, and the resulting groupings provide different perspectives on the data. Due to the unsupervised nature i.e., the lack of domain experts who can label the data, validation of results is very difficult. While there are measures that compute "goodness" scores for clustering solutions as a whole, there are few methods that validate the assignment of individual data items to their clusters. To address these challenges we focus on developing a framework that can generate, compare, combine, and evaluate different solutions to make more robust and significant statements about the data.

In the first part of this dissertation, we present fast and efficient techniques to generate and combine different clustering solutions. We build on some recent ideas on efficient representations of clusters of partitions to develop a well founded metric that is spatially aware to compare clusterings. With the ability to compare clusterings, we describe a heuristic to combine different solutions to produce a single high quality clustering. We also introduce a Markov chain Monte Carlo approach to sample different clusterings from the entire landscape to provide the users with a variety of choices. In the second part of this dissertation, we build certificates for individual data items and study their influence on effective data reduction. We present a geometric approach by defining regions of influence for data items and clusters and use this to develop adaptive sampling techniques to speedup machine learning algorithms. This dissertation is therefore a systematic approach to study the landscape of clusterings in an attempt to provide a better understanding of the data.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Graduate school has been a wonderful journey and I would like to thank many people who have helped me along the way. My advisor, Suresh Venkatasubramanian, takes major credits for keeping me motivated for five years. It has been an absolute privilege to work with him and I could not have asked for a better advisor. He is a constant source of encouragement and his deep insights have been extremely helpful in tackling all my research problems. I am grateful to Suresh for giving me a free hand in choosing the problems to work on and being a very friendly advisor in and out of academic life.

Jeff Phillips has been a wonderful mentor and I would like to thank him for having a big impact on my ability to ask better questions in research as well as improving my presentation skills. I would like to express my sincere gratitude to my committee members, Tom Fletcher, Feifei Li, and Thomas Seidl for their active support during my proposal and dissertation defense.

My colleagues in the theory lab, Avishek, John, and Samira have been wonderful company for research and otherwise. My friends in Salt Lake, including Dhanashree, Rahul, Rohit, Raj, Sanket, Shomit, and Sriram have been a wonderful relief from the stress that graduate school offered. I am also grateful to my dear friend, Franziska for her love and affection and primarily for bringing me food to the lab, late in the night. I also extend many thanks to my first circle of friends, Afroz, Deepan, Dhileeban, Hari, Jagan, Nithya, Priya, Senthil, Sowmya, and Vimal for shaping me socially.

A few people made a deep impact on me and I would like to thank those who particularly inspired me to go to graduate school. They include, Vijay Triplicane and Aravindan Raghuveer (my cousins), Selvarajan (my uncle), my best friend, Janani and her parents, Indira and K.P.N. Murthy, and my high school teachers, Sherlin, Saravanan, and Raviselvan. Lastly, I owe everything to Raman and Radha (my parents), Sreeranjani (my sister), Parthasarathy (my brother-in-law), and my grandparents and I would like to thank them for all their love, support and motivation through the years.

CHAPTER 1

INTRODUCTION

The past decade has seen a tremendous growth in the internet and thus, data creation and consumption. A few quintillion bytes of data gets generated every day [IBM et al., 2011]. Storage and processing of this data therefore becomes a big challenge. Data size explosion has become a huge challenge in data analysis. Even with advances in distributed and parallel processing and significant improvement in hardware, the "big data" clearly emerges the winner. The deluge of data, in terms of volume, velocity, and variety [Laney, 2001], makes it very difficult to process and make the data useful. With the rapid increase in the number of researchers and businesses generating huge volumes of data, we need to be able to develop techniques that can manage and analyze them efficiently in order to get the maximum juice out of the data. From recommendations on Netflix to understanding complex brain functionalities, researchers and businesses need to understand and reason about big data. Unless we develop efficient methods to perform data squashing that will preserve statistical information, we might end up in a state of "data paralysis."

Data reduction can be very useful in this context. Various data mining methods have been introduced to reduce the quantum of data to manageable sizes. In particular, clustering is a very common method of data analysis, where the goal is to group similar items together. It is unsupervised and so can be run automatically on large datasets. Due to the unsupervised nature, we do not require training data and therefore no labeling effort by a domain expert is necessary. It is therefore the most common choice for large scale exploratory data analysis [Aggarwal, 2009; Aggarwal et al., 2007; Asharaf and Murty, 2003; Bandyopadhyay et al., 2006; Barbar and Chen, 2003; Bhaduri, 2008; Chang, 2003; Costa and Venturini, 2007; Datta et al., 2006; Dean and Ghemawat, 2008; Dhillon and Modha, 2000; Forman and Zhang, 2000; Goil et al., 1999; Guha et al., 2001; Huang, 1998; Januzaj et al., 2004; Johnson and Kargupta, 2000; Kettenring, 2009; Lv et al., 2010;

Murtagh, 1999; Nagesh et al., 2001, 2000; Olman et al., 2009; Otoo et al., 2001; Qian et al., 2003; Sander et al., 1998; Sassi and Grissa, 2009; Wang et al., 2002; Zhang et al., 1996; Zhao et al., 2009; Zhou et al., 1999, 2010]. Many of these algorithms are quite adept at handling massive datasets. Since clustering creates homogeneous clusters of data, we can use more sophisticated learning tools on each of them for downstream data analysis.

The last few years have seen a number of clustering algorithms being developed and each method has its own advantages and disadvantages and provide different perspectives on the data [Arthur and Vassilvitskii, 2006, 2007; Berkhin, 2006; Bonner, 1964; Byun et al., 2007; Das et al., 2009; Dubes and Jain, 1976; Fraley and Raftery, 1998; Har-Peled and Sadri, 2005; Hartigan, 1975; Hartigan and Wong, 1979; Jain, 2010; Jain and Dubes, 1988; Jain et al., 1999; Keim and Hinneburg, 1999; Ostrovsky et al., 2006; Tan et al., 2005; Tsangaris and Naughton, 1992; Xu and Wunsch, 2009]. Although each clustering method can be successful on data containing specific structures, no single clustering method has been good enough for all kinds of data.

Clustering is an area of research that is often considered to be "a mile wide and an inch deep." Although many common clustering methods have been developed in the last few decades, no single clustering method works on all kinds of data. Data is often in high dimensions and it becomes very difficult to visualize the data and study the structures present in the data. It is impossible to pick the "best" clustering method unless we know the right patterns present in the data. Therefore, rather than trying to find a single good answer that explains a dataset, the goal of this dissertation is to explore the landscape of "good" partitions[1], reconcile between a few good quality solutions and validate both the clustering solution and assignment of individual data points to clusters. The study of such problems is called metaclustering[2] [Caruana et al., 2006].

This raises the following interesting questions:

---

[1] In what follows, we use the term partition instead of clustering (for both hard and soft cases) to represent a set of clusters decomposing a dataset. This avoids confusion between the terms "cluster," "clustering," and the procedure used to compute a partition, and will help us avoid phrases like, "We compute consensus clusterings by clustering clusters in clusterings!" We retain the use of the term "clustering" when we refer to the clustering method.

[2] Metaclustering is also referred to in the literature as generation of multiple clusterings.

1. Can we obtain a better result by integrating a collection of solutions from various clustering methods?
2. Can we compare and contrast different partitions of data to gain a perspective about the performance of various clustering methods and the clustering landscape of the data in general?
3. Can we identify more than one "good" partition, if present, of the data?
4. Can we evaluate the quality of the clustering solutions in order to decide if we have partitioned the data in the right way?
5. Can we validate the cluster assignments of individual data points?

These are some of the many metaquestions that we can ask about clustering to ensure generation of robust partitions of data. We believe that the questions we ask and the methods we develop will strengthen clustering as the first choice for the first step in exploratory data analysis.

In this dissertation, we conduct a systematic study of various metaclustering problems and address issues in validating partitions with an eye towards accountability in data mining. We make use of a geometric representation [Berlinet and Thomas-Agnan, 2004; Gretton et al., 2008; Jegelka et al., 2009; Smola et al., 2007; Sriperumbudur et al., 2010] for partitions that lets us discuss many metaclustering questions in a common framework. We introduce efficient algorithms to solve various metaclustering problems. Our goal is to make the metaclustering solutions work on massive datasets while being easy to implement. Using this geometric framework, we study the landscape of all possible partitions of data. We also develop methods to evaluate partitions by validating data to cluster assignments in the absence of an oracle that can provide and verify the labeling. We hope that this will help us to obtain robust information about the data along with increased user interaction during the data mining process. We also discuss in detail a few applications of the methods we develop in validating partitions and cluster assignments in semisupervised learning where there is an abundance of unlabeled data like in the case of unsupervised learning.

1.1 Thesis Statement

This dissertation aims to explore a few metaaspects of clustering and stability of partitions and its applications listed below.

1. Comparing and reconciling between partitions.
2. Generating a diverse set of partitions.
3. Evaluating the stability of partitions and the assignment of data to clusters.
4. Adaptive sampling to speedup semisupervised learning techniques using cluster affinity scores.

Towards this end, we build sound distance metrics to compare partitions and heuristics that compute a 1-median solution in the space of partitions, sample a diverse set of good quality partitions to explore the clustering landscape, construct measures to evaluate partitions and data assignment to individual clusters and build adaptive sampling strategies based on cluster affinity scores that reduce the amount of unlabeled data for speedup of semisupervised learning.

1.2 Organization of this Thesis

We discuss related work relevant to this dissertation in the next section. The rest of the dissertation is organized into two parts. In Part I, we present methods to compare partitions, generate a variety of partitions and reconcile between them. In Part II, we develop measures to evaluate data assignments to clusters and the quality of the partitions themselves and we study their applications. These two parts comprise a total of four chapters. We conclude the dissertation with a chapter that summarizes the contributions. In the following, we give an overview of each of these chapters.

- **Chapter 2**. This chapter surveys existing literature in various meta-aspects of clustering. We begin by reviewing measures that compare partitions. We contrast the different comparison measures and study the importance of the right measure. Next, we discuss different existing strategies that produce a single partition by comparing and reconciling between a set of input partitions. We follow this up with a description of methods in the literature that generate more than a single interesting partition. We also discuss various existing measures to evaluate partitions both externally and internally. Lastly, we discuss some model selection methods and a few semisupervised techniques as applications to some of the algorithms we develop.
- **Chapter 3**. This chapter introduces a new spatially aware metric to compare partitions that is both well founded and fast to compute. Using this metric and prior work

on representing partitions, we develop a simple heuristic to compute an ensemble solution given a variety of input partitions. This consensus procedure is very simple to implement. Since we reduce partitions to points in high dimensions, we can leverage all the advancements in clustering to compute the ensemble solution.

- **Chapter 4**. This chapter discusses a new quality measure to evaluate partitions. Along with the distance metric that we discuss in the previous chapter, we use a Markov chain Monte Carlo approach to sample for good partitions that are non-redundant from the space of all partitions. We show that we discover many partitions with this technique that are not found by traditional clustering methods. This method lets us explore the landscape of all partitions of the data allowing us to see how clusterable the data is.
- **Chapter 5**. We introduce local certificates for individual data points to make clustering decisions accountable beginning at the data level. Due to the unsupervised nature of clustering, it is difficult to validate the assignment of data to clusters. We introduce a geometric approach by defining regions of influence of data points and clusters and looking at how they overlap as a way of doing this validation. This approach to validate partitions is different from the existing approaches where the validation is only done at the partition level giving a single goodness score for each partition.
- **Chapter 6**. Many semisupervised learning methods have been developed in the last decade where the unlabeled examples are often available along with the labeled examples at the time of learning. Typically the number of unlabeled examples is much larger than the labeled examples and this makes the semisupervised learning methods very slow because of the polynomial complexity in the number of unlabeled data points. In this chapter, we look at transductive support vector machines (T-SVM) which is a very common method to compute a classifier by taking into account the distribution of the unlabeled data. We use adaptively sampling strategies to reduce the large unlabeled set by using the fact that T-SVM looks for sparse regions in the entire data to draw the classifier.
- **Chapter 7**. Finally, this chapter provides a summary of the all the contributions and future directions that spring from our foundational work. We discuss different

approaches to speed up machine learning and data mining methods. We also discuss approaches to make data mining accountable and towards this end we look at two problems of recent interest, ε-differential privacy and k-anonymity.

CHAPTER 2

RELEVANT WORK

Different clustering methods (partitional, hierarchical, density-based, grid-based and so on) are used as a preliminary data mining technique in a variety of applications in business, biology, city planning, geology, libraries, medicine, climate science, and any other large data generating entity including the ones found on the internet. However, we limit this chapter to survey the relevant work concerning the methodologies developed as a part of this dissertation and we do not cover the different types of clustering methods or its applications which are well documented at this time [Hartigan, 1975; Jain and Dubes, 1988; Xu and Wunsch, 2009].

In this chapter, we present background material that serves as a short compendium of related prior work for the subsequent chapters of this dissertation. We start with a detailed description of the theory of reproducing kernel Hilbert spaces (RKHS) that is used to represent partitions succinctly. The understanding of this is critical since we use this method to represent partitions in our work (see in Part I of the dissertation). We then discuss various methods to compare partitions including combinatorial and spatial distances. We also discuss various state of the art methods in computing consensus of multiple partitions as well as generating multiple partitions. This covers a survey of existing methodologies that help us understanding the research in metaclustering. Next, we discuss different clustering evaluation strategies. Most methods perform a global evaluation of any given partition while few methods look at how good a partition is beginning at the data point level. Finally, we describe a few semisupervised learning (SSL) methods that we will improve in Part II of the dissertation.

2.1 Reproducing Kernel Hilbert Spaces

The notion of reproducing kernel Hilbert spaces [Aronszajn, 1950; Wahba, 1990] has been well studied and is a crucial part of the theory of kernel machines.

2.1.1 Hilbert Space

A Hilbert space $\mathcal{H}$ is a complete, finite (or infinite) dimensional linear space endowed with an inner product. The inner product of two elements $u, v \in \mathcal{H}$ is denoted by $\langle u, v \rangle_{\mathcal{H}}$. For all $u, v, w \in \mathcal{H}$ and for all $\alpha \in \mathbb{R}$, the inner product satisfies the following nice properties.

1. Associativity ($\langle \alpha u, v \rangle_{\mathcal{H}} = \alpha \langle u, v \rangle_{\mathcal{H}}$).
2. Commutativity ($\langle u, v \rangle_{\mathcal{H}} = \langle v, u \rangle_{\mathcal{H}}$).
3. Distributivity ($\langle u + v, w \rangle_{\mathcal{H}} = \langle u, w \rangle_{\mathcal{H}} + \langle v, w \rangle_{\mathcal{H}}$).
4. Positive definiteness ($\langle u, u \rangle_{\mathcal{H}} \geq 0$ and equal if and only if $u = 0$).

The norm of the element u is given by $\|u\| = \sqrt{\langle u, u \rangle_{\mathcal{H}}}$. An example of an infinite-dimensional Hilbert space is L_2 with the set of functions $f : \mathbb{R}^d \to \mathbb{R}$ such that f is square integrable, i.e., $\int_{-\infty}^{\infty} |f(x)|^2 dx$ is finite. The inner product is given by $\langle f, g \rangle_{\mathcal{H}} = \int_{-\infty}^{\infty} f(x)g(x)dx$ and the norm is given by $\| f \| = \int_{-\infty}^{\infty} f^2(x)dx$. The Hilbert space is a complete metric space with respect to the distance function induced by the inner product and this is one of the useful properties of the Hilbert space that makes it attractive.

To be able to use a function predictively, it is not just enough for the function to be square integrable. We need to be able to evaluate the function at any particular data value. This property will differentiate a reproducing kernel Hilbert space from ordinary Hilbert spaces.

2.1.2 RKHS

A Hilbert space $\mathcal{H}$ is the $L_2(\mathcal{X})$ space of functions from $\mathcal{X}$ to $\mathbb{R}$ where $\mathcal{X}$ is any measurable space. An evaluation functional over the Hilbert space of functions $\mathcal{H}$ is defined as a linear functional δ_x: $\mathcal{H} \to \mathbb{R}$ that evaluates each function $f \in \mathcal{H}$ at point $x \in \mathcal{X}$, i.e., $\delta_x[f] = f(x)$.

A Hilbert space $\mathcal{H}$ is a reproducing kernel Hilbert space if the evaluation functionals are bounded, i.e., $\forall f \in \mathcal{H}, \exists m > 0$ such that $|\delta_x[f]| = |f(x)| \leq m \| f \|_{\mathcal{H}}$.

The evaluation functionals δ lie in a dual space of $\mathcal{H}$. Lets call this $\mathcal{H}^*$. In fact $\mathcal{H}^*$ is isomorphic to $\mathcal{H}$ and each function in $\mathcal{H}$ has its dual in $\mathcal{H}$ itself. It is important to note that a Hilbert space $\mathcal{H}$ admits a reproducing kernel if and only if point evaluation is a continuous and bounded functional. $\mathcal{H}$ is a function space and can be infinite-dimensional. However, minimizing over the RKHS is equivalent to minimizing over a finite-dimensional space $\mathbb{R}^n$.

The notion of boundedness allows us to evaluate each function in the space $\mathcal{H}$ at every point in the domain $\mathcal{X}$. To discuss the notion of a reproducing kernel, we first discuss the reproducing property that makes the RKHS interesting. We look at a particular theorem called the Riesz representation theorem that will help us define the reproducing property.

2.1.2.1 Riesz representation theorem. If $f : \mathcal{H} \to \mathbb{R}$ is a continuous linear functional mapping from the Hilbert space $\mathcal{H}$ to a scalar, then for all $x \in \mathcal{H}$, there exists a unique $g \in \mathcal{H}$ such that $f(x) = \langle x, g \rangle$.

2.1.3 Reproducing Property

The reproducing property makes the RKHS different from an ordinary Hilbert space. If $\mathcal{H}$ is a RKHS, then for each $x \in \mathcal{X}$, there exists a function $K_x \in \mathcal{H}$ s.t., $\delta_x[f] = \langle K_x, f \rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}$. This is called the reproducing property and for each point x there exists a K_x, called the representer of x. Intuitively, the dot product of a function with a representer for a particular x retrieves that corresponding component of f. Thus the dot product evaluated at all possible locations $x \in \mathcal{X}$ will give us components $f(x)$ that will let us set up the basis of f. f can now be viewed as a linear combination of K_x.

The reproducing property helps us to represent the evaluation functional by computing an inner product with a function in $\mathcal{H}$. Since K_x is a function in $\mathcal{H}$ (in particular the representer of x), for another point $y \in \mathcal{X}$ the reproducing property gives us, $f(y) = \langle K_y, f \rangle_{\mathcal{H}}$. In particular, $K_x(y) = \langle K_y, K_x \rangle_{\mathcal{H}}$. Let us define $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to be a kernel if K is symmetric, i.e., for $x, y \in \mathcal{X}$, $K(x,y) = K(y,x)$ and K is positive semidefinite, i.e., $\forall x_1, x_2, \cdots, x_n \in \mathcal{X}$, the gram matrix $\mathcal{K}$ defined by $\mathcal{K}_{ij} = k(x_i, x_j)$ is positive semidefinite.

We can now define a reproducing kernel. The reproducing kernel of $\mathcal{H}$ is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined by $K(x,y) = K_x(y)$. Note that any positive semidefinite kernel $K(x,t)$ can be used to construct a RKHS $\mathcal{H}_k$ associated with it. A RKHS defines a corresponding reproducing kernel and vice versa.

2.1.4 Importance of RKHS

The input domain $\mathcal{X}$ might not be linear and not have any nice properties, but the RKHS gives us a geometric understanding along with properties like linearity to play with. The RKHS induced by a kernel K on a set $\mathcal{X}$ is unique.

Reproducing kernel Hilbert spaces offer a unified framework for model building. We

can use appropriate kernels for modeling and learning different kinds of data. For example, for the metaclustering problems one of the primary goals is to create a robust representative for a cluster C. If we believe that the data was drawn from a Gaussian distribution, we can use a Gaussian kernel as a reproducing kernel to take the points in the cluster to a RKHS. Since the Gaussian centered at each point $x \in C$ takes into account the location of all neighboring points, giving less importance to the points that are faraway (possibly in other clusters), the lifted point $\Phi(x)$ in $\mathcal{H}$ corresponding to x contains more information than just about itself. This along with the property of linearity of the space allows us to define robust representatives by taking a linear sum of feature maps of each point in the cluster.

2.1.4.1 Feature map Φ. We can define a feature map Φ between the input domain $\mathcal{X}$ and the feature space $\mathcal{H}$. This takes a point $x \in \mathcal{X}$ and maps it to an element in $\mathcal{H}$. For a RKHS, the feature map is given by $\Phi(x) = K(x,\cdot)$ which satisfies $\langle \Phi(x), \Phi(y) \rangle = K(x,y)$. Computing the feature map is usually very expensive. It is therefore not tractable to compute the inner products in the feature space and therefore norms and distances between elements in the feature space. But we have the value of the kernel functions equal to that of the inner products in the feature space and we can make use of this kernel trick to avoid expensive computation in the feature space. We illustrate the feature mapping in Figure 2.1.

2.1.4.2 Mercer kernel map. If K is a kernel satisfying a certain theorem due to Mercer, we can construct the mapping Φ into a space where K acts as a dot product, i.e., $\langle \Phi(x), \Phi(y) \rangle = K(x,y)$. Furthermore, for any $\varepsilon > 0$, there exists a map Φ_n into a n-dimensional dot product space such that $|K(x,y) - \langle \Phi_n(x), \Phi_n(y) \rangle| \leq \varepsilon$. This tells us that we can work in a finite-dimensional space without any concerns of tractability while only incurring a small amount of error in the dot product evaluations of the elements of the RKHS. Thus, in the metaclustering problems, that we learn in Part I embed into a finite-dimensional Euclidean space, we can make use of existing clustering methods in this space to solve metaclustering problems.

2.2 Comparing Partitions

There are two aspects to the comparison of partitions: the combinatorial element asks, "are these two elements grouped together by both partitions, or do they disagree?" and

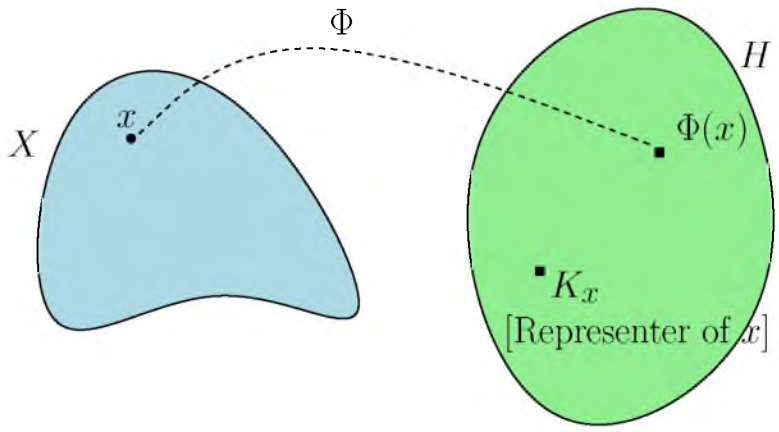


Figure 2.1. Feature map from input domain $\mathcal{X}$ to a RKHS $\mathcal{H}$. Also shown is a representer for a point x which defines the reproducing property of the feature space.

the geometric element asks, "how compact are the clusters in one partition, in comparison with the other." Although numerous metrics (and similarity measures) have been proposed to compare partitions, for the most part they are based on comparing the combinatorial structure of the partitions. This is done either by examining pairs of points that are grouped together in one partition and separated in another [Ben-Hur et al., 2002; Fowlkes and Mallows, 1983; Mirkin and Cherny, 1970; Rand, 1971], or by information theoretic considerations stemming from building a histogram of cluster sizes and normalizing it to form a distribution [Meilă, 2007; Strehl and Ghosh, 2003]. These methods ignore the actual spatial description of the data, merely treating the data as atoms in a set and using set information to compare the partitions. As has been observed by many researchers [Bae et al., 2010; Coen et al., 2010; Zhou et al., 2005], ignoring the spatial relationships in the data can be problematic.

2.2.1 Clusters as Distributions

The core idea in doing a spatially aware comparison of partitions is to treat a cluster as a distribution over the data, for example, as a sum of δ-functions at each point of the cluster [Coen et al., 2010] or as a spatial density over the data [Bae et al., 2010]. The distance between two clusters can then be defined as a distance between two distributions over a metric space (the underlying spatial domain). We will review a few spatially aware methods to compare partitions in what follows.

Zhou et al. [2005] define a distance metric CC by replacing each cluster by its centroid (this of course assumes the data does not lie in an abstract metric space) and computing a weighted transportation distance between the sets of cluster centroids. Technically, their method yields a pseudo metric, since two different clusters can have the same centroid. See Figure 2.2 where we illustrate this case with concentric ring clusters. It is also oblivious to the distribution of points within a cluster.

Coen et al. [2010] avoid the problem of selecting a cluster center by defining the distance between clusters as the transportation distance between the full sets of points comprising each cluster. This yields a metric on the set of all clusters in both partitions. In a second stage, they define the similarity distance CDISTANCE between two partitions as the ratio between the transportation distance between the two partitions (using the metric just constructed as the base metric) and a "noninformative" transportation distance in which each cluster center distributes its mass equally to all cluster centers in the other partition. While this measure is symmetric, it does not satisfy triangle inequality and is therefore not a metric.

2.2.1.1 Spatially aware comparison of partitions. Bae et al. [2010] take a slightly different approach. They build a spatial histogram over the points in each cluster and use the counts as a vector signature for the cluster. Cluster similarity is then computed via a dot product and the similarity between two partitions is then defined as the sum of cluster similarities in an optimal matching between the clusters of the two partitions, normalized by the self similarity of the two partitions.

In general, such a spatial partitioning would require a number of histogram bins exponential in the dimension; they get around this problem by only retaining information about the marginal distributions along each dimension. One weakness of this approach is that only points that fall into the same bin contribute to the overall similarity. This can lead dissimilar clusters to be viewed as similar; in Figure 2.3, the two $\triangle$ (red) clusters will be considered as similar as the two $\bigcirc$ (blue) clusters.

Their approach yields a similarity, but not a distance metric. In order to construct a metric, they have to do the usual transformation (distance $= 1 -$ similarity) and then add one to each distance between nonidentical items, which yields the somewhat unnatural (and discontinuous) metric D_{ADCO}. Their method also implicitly assumes (like Zhou

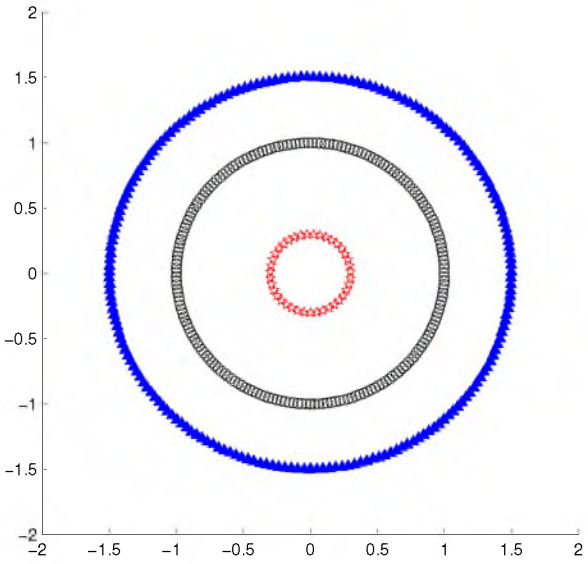


Figure 2.2. Dataset with three concentric circles, each representing a cluster partitioning the data. The CC distance [Zhou et al., 2005] can not distinguish between these clusters.

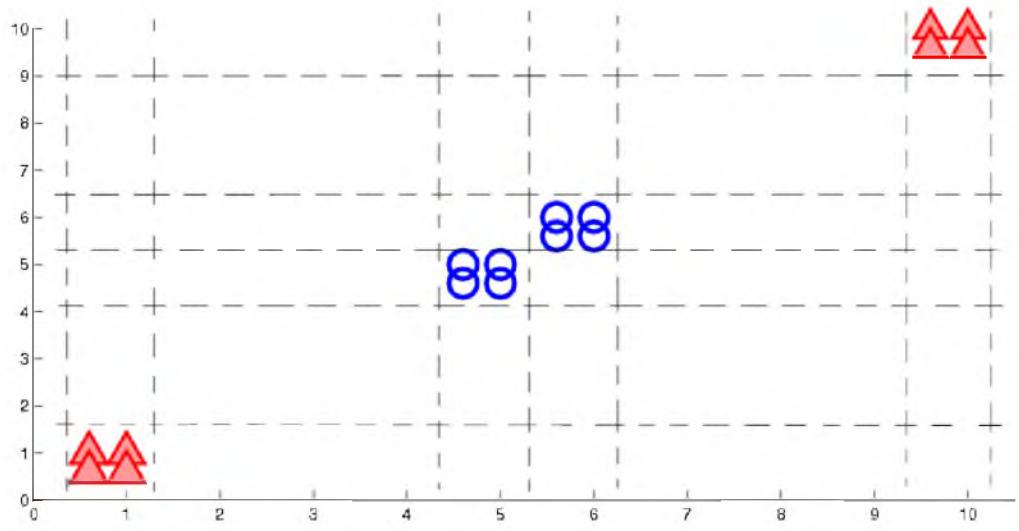


Figure 2.3. Dataset with four clusters, each a set grouped in a single grid cell. The two clusters with blue open circles are as close as the two clusters with filled red triangles under the D_{ADCO} distance.

et al. [2005]) that the data lies in Euclidean space.

2.2.2 Metrizing Distributions

There are standard constructions for defining such a distance; the most well known metrics are the transportation distance [Givens and Shortt, 1984] (also known as the Wasserstein distance, the Kantorovich distance, the Mallows distance, or the Earth mover's distance), and the metric introduced by Prokhorov [1956]. Another interesting approach was initiated by Müller [1997]. Müller developed a metric between general measures based on integrating test functions over the measure. When the test functions are chosen from a reproducing kernel Hilbert space [Aronszajn, 1950] (RKHS), the resulting metric on distributions has many nice properties [Berlinet and Thomas-Agnan, 2004; Gretton et al., 2008; Smola et al., 2007; Sriperumbudur et al., 2010]. Most importantly, it can be isometrically embedded into the Hilbert space, yielding a convenient (but infinite-dimensional) representation of a measure.

This measure has been applied to the problem of computing a single partition by Jegelka et al. [2009]. In their work, each cluster is treated as a distribution and the partition is found by maximizing the intercluster distance of the cluster representatives in the RKHS. They use the RKHS-based representation of clusters to formulate a new cost function for computing a single partition. In particular, they considered the optimization problem of finding the partition $\mathcal{P} = \{C_1, C_2\}$ of two clusters to maximize

$$\begin{aligned} C(\mathcal{P}) = &|C_1| \cdot |C_2| \cdot \|\Phi(C_1) - \Phi(C_2)\|^2_{\mathcal{H}_K} \\ &+ \lambda_1 \|\Phi(C_1)\|^2_{\mathcal{H}_\kappa} + \lambda_2 \|\Phi(C_2)\|^2_{\mathcal{H}_\kappa}, \end{aligned}$$

for various choices of kernel κ and regularization terms λ_1 and λ_2. They mention that this could then be generalized to find an arbitrary k-partition by introducing more regularizing terms. Their paper focuses primarily on the generality of this approach and how it connects to other clustering frameworks. We modify this distance and its construction in our work.

A parallel line of development generalized this idea independently to measures over higher dimensional objects (lines, surfaces and so on). The resulting metric (the current distance) is exactly the above metric when applied to 0-dimensional objects (scalar measures) and has been used extensively [Durrleman et al., 2008; Glaunès and Joshi, 2006;

Joshi et al., 2011; Vaillant and Glaunès, 2005] to compare shapes. In fact, thinking of a cluster of points as a "shape" was a motivating factor in this work.

2.3 Consensus Clustering

Consensus clustering (also referred to in the literature as ensemble clustering) studies how to combine partitions has been studied extensively [Ansari et al., 2010; Ayad and Kamel, 2010; Bonizzoni et al., 2008; Boulis and Ostendorf, 2004; Coleman and Wirth, 2010; Day, 1986; Fern and Brodley, 2003; Fred and Jain, 2005; Ghaemi et al., 2009; Gionis et al., 2007; Goder and Filkov, 2008; Murino et al., 2009; Neumann and Norton, 1986; Nguyen and Caruana, 2007; Strehl and Ghosh, 2003,?; Topchy et al., 2003; Wang et al., 2010]. Consensus clustering is a common tool of choice in the biological realm, being especially common in the area of microarray analysis [Allison et al., 2006; Benedict et al., 2006; Filkov and Skiena, 2003; Giancarlo et al., 2008; Grotkjaer et al., 2006; Laderas and McWeeney, 2007; Monti et al., 2003; Vinh and Epps, 2009]. In general, consensus methods attempt to find a partition whose sum of distances from all input partitions is minimized akin to a 1-median solution. Some of the common distances used are the Rand distance [Rand, 1971], the Jaccard distance [Ben-Hur et al., 2002], the variation of information [Meilă, 2007], and normalized mutual information [Strehl and Ghosh, 2003]. Most consensus methods are typically limited to using only combinatorial information about the partitions.

One of the most common methods for computing consensus between a collection of partitions is the majority rule: for each pair of points, each partition "votes" on whether the pair of points is in the same cluster or not, and the majority vote wins. While this method is simple, it is expensive and is spatially oblivious; two points might lie in separate clusters that are close to each other.

Alternatively, consensus can be defined via a 1-median formulation: given a distance between partitions, the consensus partition is the partition that minimizes the sum of distances to all partitions. If the distance function is a metric, then the best partition from among the input partitions is guaranteed to be within twice the cost of the optimal solution (via triangle inequality). In general, it is challenging to find an arbitrary partition that minimizes this function. For example, the above majority-based method can be viewed as

a heuristic for computing the 1-median under the Rand distance. Algorithms with formal approximations exist for this problem [Gionis et al., 2007].

Recently Ansari et al. [2010] extended these above schemes to be spatially aware by inserting CDISTANCE in place of Rand distance above. This method is successful in grouping similar clusters from an ensemble of partitions, but it is quite slow on large datasets P since it requires computing an expensive transportation distance on the full dataset. Alternatively, using representations of each cluster in the ambient space (such as its mean, as in CC [Zhou et al., 2005]) would produce another spatially aware ensemble clustering variant, but would be less effective because its representation causes unwanted simplification of the clusters (see Figure 2.2).

2.4 Alternative Clustering

The problem of generating alternative clustering [Bae and Bailey, 2006; Bae et al., 2006, 2010; Dang and Bailey, 2010a,b; Davidson and Qi, 2008; Gondek and Hofmann, 2004; Jain et al., 2008; Nguyen and Caruana, 2007; Nguyen and Epps, 2010; Qi and Davidson, 2009; Wurst et al., 2006] has received much less attention. Most of the existing literature on alternative clustering focuses on generating one additional partition of high quality that should be far from a given set (typically of size one) of existing partitions.

Most algorithms for generating alternative partitions operate as follows. Generate a single partition using a clustering algorithm of choice. Next, find another partition that is both far from the first partition and of high quality. Most methods stop here, but a few methods try to discover more alternative partitions; they repeatedly find new, still high quality, partitions that are far from all existing partitions. The alternative clustering methods usually account for diverse partitions by integrating the "diversity" criteria in the objective function. This ranges from constructing cannot-link and must-link constraints to learning a new distance function and transforming data into different space. This effectively produces a variety of partitions, but the quality of each successive partition degrades quickly.

Although there are a few other methods that try to discover alternative partitions simultaneously [Caruana et al., 2006; Jain et al., 2008; Niu et al., 2010], they are usually limited to discovering two partitions of the data. Other methods that generate more than just two partitions either randomly weigh the features or project the data onto different subspaces,

but use the same clustering technique to get the alternative partitions in each round. Using the same clustering technique tends to generate partitions with clusters of similar shapes and might not be able to exploit all the structure in the data.

Another related problem is that of picking a diverse set of partitions from a large input set. This area of work does not focus on generating partitions but assumes a collection of input partitions is already available. Given as input a set of $m \gg k$ partitions of a single dataset, the goal here is to produce k distinct partitions. To obtain the input for this approach, either the output of several distinct clustering algorithms, or the output of multiple runs of the same randomized algorithm with different initial seeds are considered [Zhang and Li, 2011]. This problem can then be viewed as a clustering problem; that is, finding k clusters of partitions from the set of input partitions. Therefore, there are many possible optimization criteria or algorithms that could be explored for this problem as there are for clustering in general. Most formal optimization problems are intractable to solve exactly, making heuristics the only option. Furthermore, no matter the technique, the solution is only as good as the input set of partitions, independent of the optimization objective.

2.5 Evaluating Clusterings

A number of indirect approaches have been developed to validate a partition at a global level [Halkidi et al., 2001; Xu and Wunsch, 2009]. These include internal, external, and relative validation techniques, and methods based on clustering stability that assume a clustering (algorithm) is good if small perturbations in the input do not affect the output clustering significantly. There are supervised variants of clustering. However, these typically require domain knowledge and the immense popularity of clustering comes precisely from the fact that it can be applied as a first filter to acquire a deeper understanding of the data.

Clusterings can be validated globally in three different ways [Xu and Wunsch, 2009]. Internal validation mechanisms look at the structure of a clustering and attempt to determine its quality [Liu et al., 2010]. For example, the ratio of the minimum intercluster distance to the maximum intracluster distance is a measure of how well-separated clusters are, and thus how good the partition is. External validation measures can be employed when a reference partition exists. In this case, an appropriate distance between partitions must

be defined. The given partition can then be compared to the reference partition [Halkidi et al., 2001]. Relative validation measures look at different runs of a clustering algorithm and compare the resulting partitions produced [Halkidi et al., 2001].

Cluster stability [Ben-David et al., 2006; Ben-Hur et al., 2002; Bezdek and Pal, 1998] is another way to validate partitions. The goal here is to determine how robust a clustering solution is to small perturbations in algorithm parameters. This idea was used to do model selection; for example, the "right" number of clusters is the one that exhibits the most stable partitions. Stability in general has been studied extensively in the statistics and machine learning communities, as a way to understand generalization properties of algorithms. The paper by Elisseeff et al. [2006] provides a good overview of this literature and the monograph by Luxburg [2009] focuses on clustering.

Where admissible (for example, when effective models of the data can be built), probabilistic modeling yields posterior likelihoods for a cluster assignment in the form of conditional probabilities $p(C \mid \mathbf{x})$ for point $\mathbf{x}$ and cluster C. We view our approach as complementary to (and more general than) model-based validation. Our approach is purely data driven with no further assumptions, which is appropriate when initially exploring a dataset. We also show that the affinity scores produced by our method closely match the likelihoods produced by a standard clustering approach like Gaussian mixture models. Note that probabilistic modeling can be used to choose a particular way of clustering the data, but in the setting we consider, a clustering is already given to us (possibly even by consensus clustering or some other method), and the goal is to validate it.

Local validation bears a superficial resemblance to outlier detection: in both cases the goal is to evaluate individual points based on how well they "fit" into a clustering. There are important differences though. An outlier affects the cost of a partition by being faraway from any cluster, but it will usually be clear what cluster it might be assigned to. In contrast, a point whose labeling might be invalid is usually in the midst of the data. Assigning it to one cluster or another might not actually change the clustering cost, even though the label itself is now unreliable.

2.6 Semisupervised Learning

We discuss applications of attaching cluster affinity scores, which are discussed in Chapter 3, to semisupervised learning. In particular, transductive support vector machines (T-SVM) in Chapter 6. We review a few existing works here on fast semisupervised learning (SSL).

As dataset sizes grow, there is an increasing interest in scaling up semisupervised learning algorithms. We briefly review prior work on large scale SSL and in particular, recent approaches to scale up T-SVM. Especially in web applications, such as link prediction and link propagation, unlabeled data exists in abundance and several specialized algorithms were proposed to incorporate these large datasets into training [Kashima et al., 2009; Settles, 2011; Suzuki and Isozaki, 2008]. These publications introduce new algorithms that are highly specialized for specific learning tasks and typically do not generalize to common SSL classification.

Mann and McCallum [2007] achieve large scale SSL by constraining classifiers to match expectation constraints, made by domain experts. The authors successfully incorporate unlabeled data with the help of side information. Chen et al. [2011] adapt cotraining [Blum and Mitchell, 1998] to single-view data and subsample unlabeled data to obtain a subset on which the theoretical conditions for cotraining [Balcan et al., 2004] can be satisfied. In contrast to our work, they do not focus on speedup during training but instead assume that unlabeled data is noisy and aim to remove such noisy samples.

We also describe some work on fast T-SVM as our work explicitly focuses on T-SVM. We make no assumptions on the data or task and require no additional side information unlike e.g., Vural et al. [2008]. There is a significant amount of prior work on speeding up the T-SVM algorithm [Collobert et al., 2006a,b; Liao et al., 2007; Sindhwani and Keerthi, 2007; Vural et al., 2008]. Other researchers achieve their speedups through novel optimization algorithms, explicitly designed for the T-SVM algorithm [Collobert et al., 2006a,b; Liao et al., 2007; Sindhwani and Keerthi, 2007]. For example, Sindhwani and Keerthi [2007] use Newton's method to take advantage of second order information for accelerated convergence. Collobert et al. [2006b] relax the problem into a concave convex procedure (CCCP) [Yuille and Rangarajan, 2001] to speed up the optimization, which results in impressive speedups.

PART I

METACLUSTERING

CHAPTER 3

CONSENSUS CLUSTERING[1]

We study the problem of consensus or ensemble clustering [Ansari et al., 2010; Ayad and Kamel, 2010; Bonizzoni et al., 2008; Boulis and Ostendorf, 2004; Coleman and Wirth, 2010; Day, 1986; Fern and Brodley, 2003; Fred and Jain, 2005; Ghaemi et al., 2009; Gionis et al., 2007; Goder and Filkov, 2008; Murino et al., 2009; Neumann and Norton, 1986; Nguyen and Caruana, 2007; Strehl and Ghosh, 2003; Topchy et al., 2003; Wang et al., 2010], where the goal is to determine a high quality partition that is "close" to a given set of input partitions.

The problem of metaclustering has become important in recent years as researchers have tried to combine the strengths and weaknesses of different clustering algorithms to find patterns in data. A common metaclustering problem is that of finding a consensus (or ensemble) partition from among a set of candidate partitions. Ensemble-based clustering has been found to be very powerful when different clusters are connected in different ways, each detectable by different classes of clustering algorithms [Strehl and Ghosh, 2003]. For instance, no single clustering algorithm can detect clusters of symmetric Gaussian-like distributions of different density and clusters of long thinly connected paths; but these clusters can be correctly identified by combining multiple techniques (i.e., k-means and single-link) [Ghaemi et al., 2009]. Moreover, given an abstract dataset, a practitioner may not know what types of data patterns exist, and which specific clustering algorithm to use. Hence, consensus clustering is becoming a more robust and general way to approach clustering.

Other related and important metaclustering problems include finding a different and yet informative partition to a given one, or finding a set of partitions that are mutually diverse

[1]Reprinted with permission of SIAM, 2011, Parasaran Raman, Jeff M. Phillips, and Suresh Venkatasubramanian, Spatially-Aware Comparison and Consensus for Clusterings. Eleventh SIAM International Conference on Data Mining, Pages 307-318.

(and therefore informative). In all these problems, the key underlying step is comparing two partitions and quantifying the difference between them. Numerous metrics (and similarity measures) have been proposed to compare partitions, and for the most part they are based on comparing the combinatorial structure of the partitions. This is done either by examining pairs of points that are grouped together in one partition and separated in another [Ben-Hur et al., 2002; Fowlkes and Mallows, 1983; Mirkin and Cherny, 1970; Rand, 1971], or by information theoretic considerations stemming from building a histogram of cluster sizes and normalizing it to form a distribution [Meilă, 2007; Strehl and Ghosh, 2003].

These methods ignore the actual spatial description of the data, merely treating the data as atoms in a set and using set information to compare the partitions. Ignoring the spatial relationships in the data can be problematic. Consider the three partitions in Figure 3.1. The first partition (FP) is obtained by a projection onto the y-axis, and the second (SP) is obtained via a projection onto the x-axis. Partitions (FP) and (SP) are both equidistant from partition (RP) under any of the above mentioned distances, and yet it is clear that (FP) is more similar to the reference partition, based on the spatial distribution of the data.

Some researchers have proposed spatially aware distances between partitions, but they all suffer from various deficiencies. They compromise the spatial information captured by the clusters [Bae et al., 2010; Zhou et al., 2005], they lack metric properties [Coen et al., 2010; Zhou et al., 2005] (or have discontinuous ranges of distances to obtain metric properties [Bae et al., 2010]), or they are expensive to compute, making them ineffective for large datasets [Coen et al., 2010].

3.1 Overview of Our Work

We exploit a concise, linear reproducing kernel Hilbert space (RKHS) representation of clusters. We use this representation to construct an efficient spatially aware metric between partitions and an efficient spatially aware consensus clustering algorithm.

We use ideas from some recent work. We leverage the fact that a cluster can be viewed as a sample of data points from a distribution [Jegelka et al., 2009], and through a similarity kernel K, a distribution can be losslessly lifted to a single vector in a RKHS [Müller, 1997].

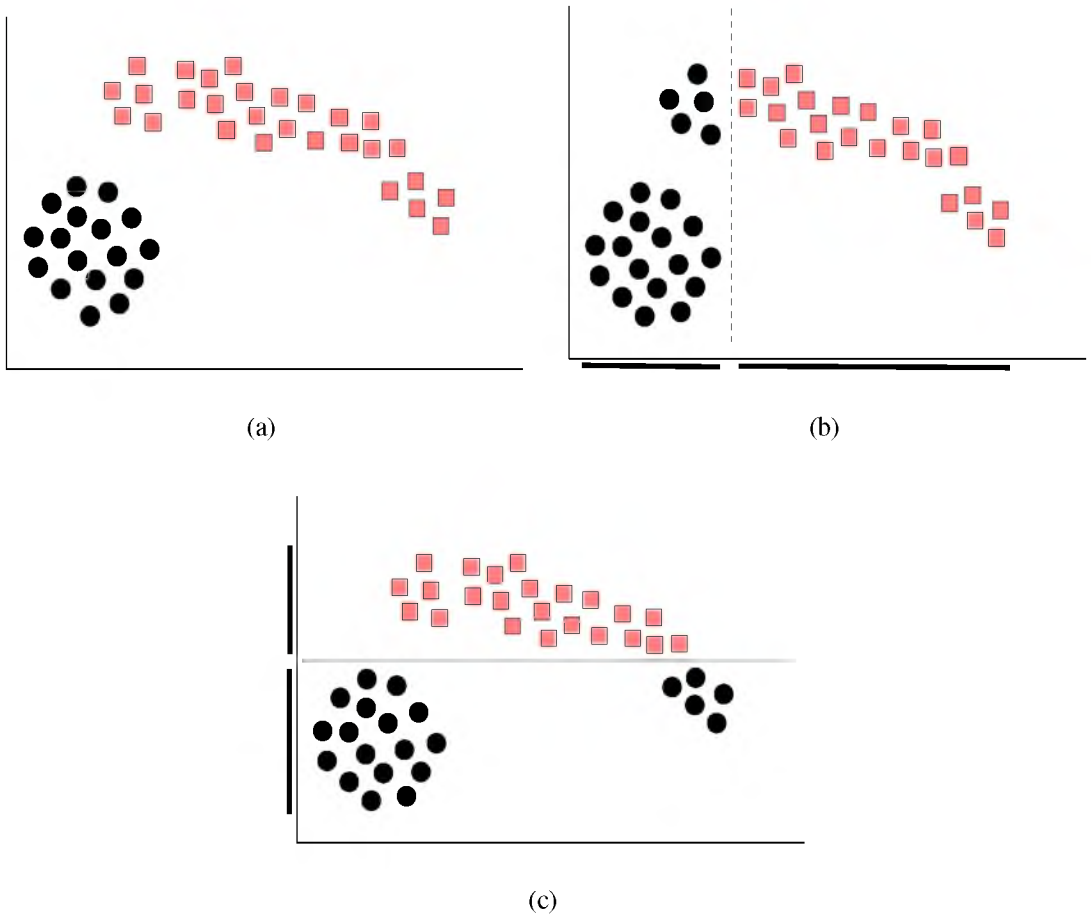


Figure 3.1. Spatially aware distances are important: (b) first parition (FP) and (c) second partition (SP) are equidistant from (a) the reference partition (RP) under a set-based distance. However, FP is clearly more similar to RP than SP.

3.1.1 Representations

We adapt the representation of the clusters in the RKHS (which we discussed in Chapter 2) in two ways: approximation and normalization. Typically, vectors in a RKHS are infinite-dimensional, but they can be approximated arbitrarily well in a finite-dimensional ℓ_2 space that retains the linear structure of the RKHS [Joshi et al., 2011; Rahimi and Recht, 2007]. This provides concise and easily manipulated representations for entire clusters. The resulting distance between the representative vectors of two distributions in the RKHS can be used as a metric on the distributions [Müller, 1997; Smola et al., 2007; Sriperumbudur et al., 2010]. The distance metric between the approximate normalized representation of clusters is fast to compute. Additionally, we normalize these vectors to

focus on the spatial information of the clusters. This turns out to be important in consensus clustering, as explained in Section 3.3.2.

3.1.2 Distance Computation

Using this convenient representation (an approximate normalized RKHS vector), we develop a metric between partitions. Since the clusters can now be viewed as points in (scaled) Euclidean space we can apply standard measures for comparing point sets in such spaces. In particular, we define a spatially aware metric LIFTEMD between partitions as the transportation distance [Givens and Shortt, 1984] between the representatives, weighted by the number of points they represent. While the transportation distance is a standard distance metric on probability distributions, it is expensive to compute (requiring $O(n^3)$ time for n points) [Kuhn, 1955]. However, since the points here are clusters, and the number of clusters (k) is typically significantly less than the data size (n), this is not a significant bottleneck as we will see in Section 3.6.

3.1.3 Consensus

We exploit the linearity of the RKHS representations of the clusters to design an efficient consensus clustering algorithm. Given several partitions, each represented as a set of vectors in a RKHS, we can find a partition of this data using standard Euclidean clustering algorithms. In particular, we can compute a consensus partition by simply running k-means (or hierarchical agglomerative clustering) on the lifted representations of each cluster from all input partitions. This reduction from consensus to Euclidean clustering is a key contribution: it allows us to utilize the extensive research and fast algorithms for Euclidean clustering, rather than designing complex hypergraph partitioning methods [Strehl and Ghosh, 2003].

3.1.3.1 Evaluation. All of these aspects of our technical contributions are carefully evaluated on real world and synthetic data. As a result of the convenient isometric representation, the well founded metric, and reduction to many existing techniques, our methods perform well compared to previous approaches and are much more efficient.

3.1.4 Comparison to Prior Work

In Chapter 2, we discussed in detail various methods and measures to compare partitions both combinatorially and in a spatially aware manner. We discussed the need for a spatially aware measure as it captures the properties that lead to the formation of the clusters in the first place. We also highlighted the shortcomings of the existing spatially aware measures to compare partitions (see Section 2.2). Our method, centered around the RKHS-based metric between distributions, addresses all of the above problems. It yields a true metric, incorporates the actual distribution of the data correctly, and avoids exponential dependency on the dimension. The price we pay is the requirement that the data lie in a space admitting a positive definite kernel. However, this actually enables us to apply our method to clustering objects like graphs and strings, for which similarity kernels exist [Gärtner, 2002; Lodhi et al., 2002] but no convenient vector space representation is known.

In Section 2.3, we discussed various methods to compute the consensus solution given a set of input partitions. We discussed various common consensus procedures that are based on the majority rule: for each pair of points, each partition "votes" on whether the pair of points is in the same cluster or not, and the majority vote wins and ones that are spatially aware where the consensus is defined via a 1-median formulation. Our method is very elegant in that we reduce the problem of computing consensus to a clustering problem and our method is very fast since we only use the succinct cluster representatives in the consensus procedure. Not only can we leverage and use any of the common clustering techniques of choice to compute the consensus solution, but we also retain succinct representations of the input partitions and the consensus solution that help reduce the storage size.

3.2 Preliminaries

3.2.1 Definitions

Let P be the set of points being clustered, with $|P| = n$. We use the term **cluster** to refer to a subset C of P (i.e., an actual cluster of the data), and the term **partition** to refer to a partitioning of P into clusters (i.e., what one would usually refer to as a partition of P). Clusters will always be denoted by the capital letters $A, B, C, \ldots$, and partitions will be denoted by the symbols $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$. We will also consider **soft partitions** of P, which are

fractional assignments $\{p(C|x)\}$ of points x to clusters C such that for any x, the assignment weights $p(C|x)$ sum to one.

We will assume that P is drawn from a space X endowed with a reproducing kernel $\kappa : X \times X \to \mathbb{R}$ [Aronszajn, 1950]. The kernel κ induces a Hilbert space $\mathscr{H}_\kappa$ via the lifting map $\Phi : X \to \mathscr{H}_\kappa$, with the property that $\kappa(x,y) = \langle \Phi(x), \Phi(y) \rangle_\kappa$, $\langle \cdot, \cdot \rangle_\kappa$ being the inner product that defines $\mathscr{H}_\kappa$.

Let p,q be probability distributions defined over X. Let $\mathcal{F}$ be a set of real valued bounded measurable functions defined over X. Let $\mathcal{F}_\kappa \triangleq \{f \in \mathcal{F} \mid \|f\|_\kappa \leq 1\}$ denote the unit ball in the Hilbert space $\mathscr{H}_\kappa$. The integral probability metric [Müller, 1997] γ_κ on distributions p,q is defined as $\gamma_\kappa(p,q) = \sup_{f \in \mathcal{F}_\kappa} |\int_X f dp - \int_X f dq|$. We will make extensive use of the following explicit formula for $\gamma_\kappa(p,q)$:

$$\begin{aligned} \gamma_\kappa^2(p,q) = &\iint_X \kappa(x,y) dp(x) dp(y) \\ &+ \iint_X \kappa(x,y) dq(x) dq(y) \\ &- 2 \iint_X \kappa(x,y) dp(x) dq(y), \end{aligned} \tag{3.2.1}$$

which can be derived (via the kernel trick) from the following formula for γ_κ [Sriperumbudur et al., 2010]: $\gamma_\kappa(p,q) = \| \int_X \kappa(\cdot,x) dp(x) - \int_X \kappa(\cdot,x) dq(x) \|_{\mathscr{H}_\kappa}$. This formula also gives us the lifting map Φ, since we can write $\Phi(p) = \int_X \kappa(\cdot,x) dp(x)$.

3.2.1.1 The transportation metric. Let $D : X \times X \to \mathbb{R}$ be a metric over X. The transportation distance between p and q is then defined as

$$d_T(p,q) = \inf_{f : X \times X \to [0,1]} \int_X \int_X f(x,y) D(x,y) dx dy, \tag{3.2.2}$$

such that $\int_X f(x,y) dx = q(y)$ and $\int_X f(x,y) dy = p(x)$. Intuitively, $f(x,y)D(x,y)$ measures the work in transporting $f(x,y)$ mass from $p(x)$ to $q(y)$.

3.2.2 An RKHS Distance Between Clusters

We use γ_κ to construct a metric on clusters. Let $C \subset P$ be a cluster. We associate with C the distribution $p(C) = \sum_x p(C|x) w(x) \delta_x(\cdot)$, where $\delta_x(\cdot)$ is the Kronecker δ-function and $w : P \to [0,1]$ is a weight function. Given two clusters $C, C' \subset P$, we define $d(C,C') = \gamma_\kappa(p(C), p(C'))$. It is important to note that we are not introducing a completely new distance. We introduce a wrapper that encapsulates existing distance metric between clusters in order to facilitate comparing partitions.

3.2.2.1 An example. A simple example illustrates how this distance generalizes pure partition-based distances between clusters. Suppose we fix the kernel $\kappa(x,y)$ to be the discrete kernel: $\kappa(x,x) = 1$, $\kappa(x,y) = 0\ \forall x \neq y$. Then it is easy to verify that $d(C,C') = \sqrt{|C\Delta C'|}$ is the square root of the cardinality of the symmetric difference $C\Delta C'$, which is a well known set theoretic measure of dissimilarity between clusters. Since this kernel treats all distinct points as equidistant from each other, the only information remaining is the set theoretic difference between the clusters. As κ acquires more spatial information, $d(C,C')$ incorporates this into the distance calculation.

3.2.2.2 Representations in $\mathscr{H}_\kappa$. There is an elegant way to represent points, clusters and partitions in the RKHS $\mathscr{H}_\kappa$. Define the lifting map $\Phi(x) = \kappa(\cdot,x)$. This takes a point $x \in P$ to a vector $\Phi(x)$ in $\mathscr{H}_\kappa$. A cluster $C \subset P$ can now be expressed as a weighted sum of these vectors: $\Phi(C) = \sum_{x\in C} w(x)\Phi(x)$. Note that for clarity, in what follows we will assume without loss of generality that all partitions are hard; to construct the corresponding soft partition-based expression, we merely replace terms of the form $\{x \in C\} = \mathbf{1}_{x\in C}$ by the probability $p(C|x)$.

$\Phi(C)$ is also a vector in $\mathscr{H}_\kappa$, and we can now rewrite $d(C,C')$ as

$$d(C,C') = \|\Phi(C) - \Phi(C')\|_{\mathscr{H}_\kappa}. \tag{3.2.3}$$

Finally, a partition $\mathcal{P} = \{C_1, C_2, \ldots C_k\}$ of P can be represented by the set of vectors $\Phi(\mathcal{P}) = \{\Phi(C_i)\}$ in $\mathscr{H}_\kappa$. We note that as long as the kernel is chosen correctly [Sriperumbudur et al., 2010], this mapping is isometric, which implies that the representation $\Phi(C)$ is a lossless representation of C.

The linearity of representation is a crucial feature of how clusters are represented. The cluster in $\mathscr{H}_\kappa$ is merely the weighted sum of the corresponding vectors $\Phi(x)$. As a consequence, it is easy to represent soft partitions as well. A cluster C can be represented by the vector

$$\Phi(C) = \sum_x w(x)p(C|x)\Phi(x). \tag{3.2.4}$$

The lifting map [Aronszajn, 1950] Φ associated with κ makes this construction different and more powerful than other approaches for comparing distributions. The major advantages of using this representation are as follows. We have a very defined structure. Euclidean space is very well studied, and we can utilize all the tools available to us for

data analysis in this space. The linearity of the representation helps us to handle both hard partitions (a point assigned to exactly one cluster) and soft partitions (where points assign their weight fractionally to different clusters). This technique is agnostic to the underlying domain the data comes from. For example, it is possible to compare and combine partitions of images based on text attributes and based on image information. We have a Euclidean space representation and it allows us to leverage existing data structures and code bases.

3.3 Approximate Normalized Cluster Representation

We adapt the RKHS-based representation of clusters $\Phi(C)$ in two ways to make it more amenable to our metaclustering goals. First, we approximate $\Phi(C)$ to a finite-dimensional (ρ-dimensional) vector. This provides a finite representation of each cluster in $\mathbb{R}^\rho$ (as opposed to a vector in the infinite-dimensional $\mathcal{H}_\kappa$), it retains linearity properties, and it allows for fast computation of distance between two clusters. Second, we normalize $\Phi(C)$ to remove any information about the size of the cluster; retaining only spatial information. This property becomes critical for consensus clustering.

3.3.1 Approximate Lifting Map $\tilde{\Phi}$

The lifted representation $\Phi(x)$ is the key to the representation of clusters and partitions, and its computation plays a critical role in the overall complexity of the distance computation. For kernels of interest (like the Gaussian kernel), $\Phi(x)$ cannot be computed explicitly, since the induced RKHS is an infinite-dimensional function space.

However, we can take advantage of the shift invariance of commonly occurring kernels[2]. For these kernels a random projection technique in Euclidean space defines an approximate lifting map $\tilde{\Phi} : X \times X \to \mathbb{R}^\rho$ with the property that for any $x, y \in P$,

$$\left| \|\tilde{\Phi}(x) - \tilde{\Phi}(y)\|_2 - \|\Phi(x) - \Phi(y)\|_{\mathcal{H}_\kappa} \right| \leq \varepsilon,$$

where $\varepsilon > 0$ is an arbitrary user defined parameter, and $\rho = \rho(\varepsilon)$. Notice that the approximate lifting map takes points to ℓ_2^ρ with the standard inner product, rather than a general Hilbert space. The specific construction is due to Rahimi and Recht [2007] and analyzed by Joshi et al. [2011], to yield the following result:

[2] A kernel $\kappa(x, y)$ defined on a vector space is shift invariant if it can be written as $\kappa(x, y) = g(x - y)$.

Theorem 3.3.1 (Joshi et al. [2011]) *Given a set of n points $P \subset X$, shift invariant kernel $\kappa : X \times X \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a map $\tilde{\Phi} : X \times X \to \mathbb{R}^{\rho}$, $\rho = O((1/\varepsilon^2)\log n)$, such that for any $x, y \in P$, $\left| \|\tilde{\Phi}(x) - \tilde{\Phi}(y)\|_2 - \|\Phi(x) - \Phi(y)\|_{\mathscr{H}_\kappa} \right| \leq \varepsilon$.*

The actual construction is randomized and yields a $\tilde{\Phi}$ as above with probability $1 - \delta$, where $\rho = O((1/\varepsilon^2)\log(n/\delta))$. For any x, constructing the vector $\tilde{\Phi}(x)$ takes $O(\rho)$ time.

3.3.2 Normalizing $\Phi(C)$

The lifting map Φ is linear with respect to the weights of the data points, while being nonlinear in its location. Since $\Phi(C) = \sum_{x \in C} w(x)\Phi(x)$, this means that any scaling of the vectors $\Phi(x)$ translates directly into an uniform scaling of the weights of the data, and does not affect the spatial positioning of the points. This implies that we are free to normalize the cluster vectors $\Phi(C)$, so as to remove the scale information, retaining only, and exactly, the spatial information. In practice, we will normalize the cluster vectors to have unit length; let

$$\bar{\Phi}(C) = \Phi(C)/\|\Phi(C)\|_{\mathcal{H}_\kappa}.$$

Figure 3.2 shows an example of why it is important to compare RKHS representations of vectors using only their spatial information. In particular, without normalizing, small clusters C will have small norms $\|\Phi(C)\|_{\mathcal{H}_\kappa}$, and the distance between two small vectors $\|\Phi(C_1) - \Phi(C_2)\|_{\mathcal{H}_\kappa}$ is at most $\|\Phi(C_2)\|_{\mathcal{H}_\kappa} + \|\Phi(C_2)\|_{\mathcal{H}_\kappa}$. Thus all small clusters will likely have similar unnormalized RKHS vectors, irrespective of spatial location.

3.3.3 Computing the Distance Between Clusters

For two clusters C, C', we defined the distance between them as

$$d(C, C') = \gamma_\kappa(p(C), p(C')).$$

Since the two distributions $p(C)$ and $p(C')$ are discrete (defined over $|C|$ and $|C'|$ elements, respectively), we can use (3.2.1) to compute $d(C, C')$ in time $O(|C| \cdot |C'|)$. While this may be suitable for small clusters, it rapidly becomes expensive as the cluster sizes increase.

If we are willing to approximate $d(C, C')$, we can use Theorem 3.3.1 combined with the implicit definition of $d(C, C')$ as $\|\Phi(C) - \Phi(C')\|_{\mathscr{H}_k}$. Each cluster C is represented as

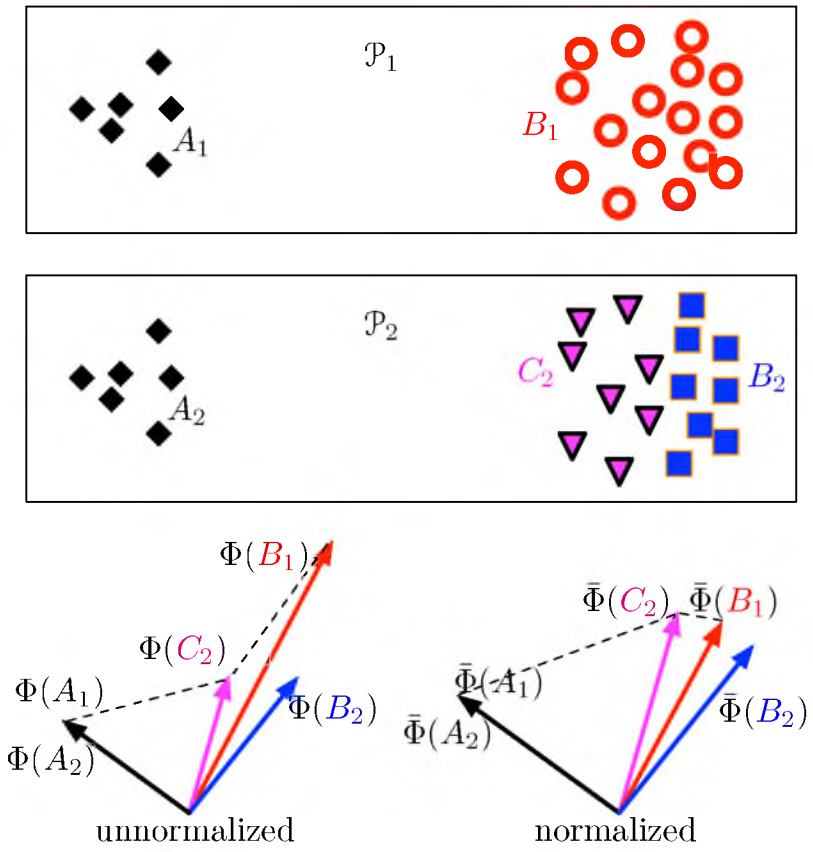


Figure 3.2. Two partitions $\mathcal{P}_1 = \{A_1, B_1\}$ and $\mathcal{P}_2 = \{A_2, B_2, C_2\}$ of the same dataset, and a 2-d visualization of all of their representations in a RKHS. Note that the unnormalized vectors (on the left) have $\Phi(B_1)$ far from $\Phi(B_2)$ and $\Phi(C_2)$ even though the second two are subsets of the first. The normalized vectors (on the right) have $\bar{\Phi}(B_1)$ close to both $\bar{\Phi}(B_2)$ and $\bar{\Phi}(C_2)$. In particular, $\Phi(C_2)$ is closer to $\Phi(A_1)$ than $\Phi(B_1)$, but $\bar{\Phi}(C_2)$ is much closer to $\bar{\Phi}(B_1)$ than $\bar{\Phi}(A_1)$.

a sum of $|C|$ ρ-dimensional vectors and can be computed in $O(|C|\rho)$ time. The ℓ_2 distance between the resulting vectors can be computed in $O(\rho)$ time. The following approximation guarantee on $d(C, C')$ then follows from the triangle inequality and an appropriate choice of ε. For any two clusters C, C' and any $\varepsilon > 0$, $d(C, C')$ can be approximated to within an additive error ε in time $O((|C| + |C'|)\rho)$ time, where $\rho = O((1/\varepsilon^2)\log n)$.

3.4 New Distances between Partitions

Let $\mathcal{P} = \{C_1, C_2, \ldots\}$ and $\mathcal{P}' = \{C'_1, C'_2, \ldots\}$ be two different partitions of P with associated representations $\Phi(\mathcal{P}) = \{\Phi(C_1), \Phi(C_2), \ldots\}$ and $\Phi(\mathcal{P}') = \{\Phi(C'_1), \Phi(C'_2), \ldots\}$. Similarly, $\bar{\Phi}(\mathcal{P}) = \{\bar{\Phi}(C_1), \bar{\Phi}(C_2), \ldots\}$ and $\bar{\Phi}(\mathcal{P}') = \{\bar{\Phi}(C'_1), \bar{\Phi}(C'_2), \ldots\}$. Since the two

representations are sets of points in a Hilbert space, we can draw on a number of techniques for comparing point sets from the world of shape matching and pattern analysis.

We can apply the transportation distance d_T on these vectors to compare the partitions, treating the partitions as distributions. In particular, the partition $\mathcal{P}$ is represented by the distribution

$$\sum_{\bar{\Phi}(C)\in\bar{\Phi}(\mathcal{P})} \frac{|C|}{|P|} \cdot \delta_{\bar{\Phi}(C)}, \tag{3.4.1}$$

where $\delta_{\bar{\Phi}(C)}$ is a Dirac delta function at $\bar{\Phi}(C) \in \mathcal{H}_{\mathcal{K}}$ and $\mathscr{H}_{\mathcal{K}}$ is the underlying metric. We will refer to this metric on partitions as

$$\text{LIFTEMD}(\mathcal{P},\mathcal{P}') = d_T(\bar{\Phi}(\mathcal{P}),\bar{\Phi}(\mathcal{P}')). \tag{3.4.2}$$

3.4.1 An Example, Continued

Once again, we can simulate the loss of spatial information by using the discrete kernel as in Section 3.2.2. The transportation metric is computed (see Section 3.2.1) by minimizing a functional over all partial assignments $f(x,y)$. If we set $f(C,C') = |C \cap C'|/n$ to be the fraction of points overlapping between clusters, then the resulting transportation cost is precisely the Rand distance between the two partitions! This observation has two implications. First, that standard distance measures between partitions appear as special cases of this general framework. Second, $\text{LIFTEMD}(\mathcal{P},\mathcal{P}')$ will always be at most the Rand distance between $\mathcal{P}$ and $\mathcal{P}'$.

We can also use other measures. Let

$$\vec{d_H}(\Phi(\mathcal{P}),\Phi(\mathcal{P}')) = \max_{v\in\Phi(\mathcal{P})} \min_{w\in\Phi(\mathcal{P}')} \|v-w\|_{\mathscr{H}_{\mathcal{K}}}. \tag{3.4.3}$$

Then the Hausdorff distance [Chew et al., 1997] is defined as

$$d_H(\Phi(\mathcal{P}),\Phi(\mathcal{P}')) = \max\left(\vec{d_H}\left(\Phi(\mathcal{P}),\Phi(\mathcal{P}')\right), \vec{d_H}\left(\Phi(\mathcal{P}'),\Phi(\mathcal{P})\right)\right). \tag{3.4.4}$$

We refer to this application of the Hausdorff distance to partitions as

$$\text{LIFTH}(\mathcal{P},\mathcal{P}') = d_H(\bar{\Phi}(\mathcal{P}),\bar{\Phi}(\mathcal{P}')). \tag{3.4.5}$$

We could also use our lifting map again. Since we can view the collection of points $\Phi(\mathcal{P})$ as a spatial distribution in $\mathscr{H}_{\mathcal{K}}$ (see (3.4.1)), we can define $\gamma_{\mathcal{K}'}$ in this space as well,

with κ' again given by any appropriate kernel (for example, $\kappa'(v,w) = \exp(-\|v-w\|^2_{\mathscr{H}_{\kappa'}})$. We refer to this metric as

$$\text{LIFTKD}(\mathcal{P},\mathcal{P}') = \gamma_{\kappa'}(\bar{\Phi}(\mathcal{P}),\bar{\Phi}(\mathcal{P}')). \tag{3.4.6}$$

3.4.2 Computing the Distance Between Partitions

The approximate lifting map $\tilde{\Phi}$ is efficient in two ways. First, it is fast to generate a representation of a cluster C ($O(|C|\rho)$ time), and second, it is easy to estimate the distance between two clusters ($O(\rho)$ time). This implies that after a linear amount of processing, all distance computations between partitions depend only on the number of clusters in each partition, rather than the size of the input data. Since the number of clusters is usually orders of magnitude smaller than the size of the input, this allows us to use asymptotically inefficient algorithms on $\tilde{\Phi}(\mathcal{P})$ and $\tilde{\Phi}(\mathcal{P}')$ that have small overhead, rather than requiring more expensive (but asymptotically cheaper in k) procedures. Assume that we are comparing two partitions $\mathcal{P},\mathcal{P}'$ with k and k' clusters, respectively. LIFTEMD is computed in general using a min cost flow formulation of the problem, which is then solved using the Hungarian algorithm. This algorithm takes time $O((k+k')^3)$. While various approximations of d_T exist [Indyk and Thaper, 2003; Shirdhonkar and Jacobs, 2008], the exact method suffices for our setting for the reasons mentioned above.

It is immediately clear from the definition of LIFTH that it can be computed in time $O(k \cdot k')$ by a brute force calculation. A similar bound holds for exact computation of LIFTKD. While approximations exist for both of these distances, they incur overhead that makes them inefficient for small k.

3.5 Computing Consensus Partitions

As an application of our proposed distance between partitions, we describe how to construct a spatially aware consensus from a collection of partitions. This method reduces the consensus problem to a standard clustering problem, allowing us to leverage the extensive body of work on standard clustering techniques. Furthermore, the representations of clusters as vectors in $\mathbb{R}^\rho$ allows for very concise representation of the data, making our algorithms extremely fast and scalable.

3.5.1 A Reduction from Consensus Finding to Clustering

Our approach exploits the linearity of cluster representations in $\mathscr{H}_\kappa$, and works as follows. Let $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_m$ be the input (hard or soft) partitions of P. Under the lifting map Φ, each partition can be represented by a set of points $\{\Phi(\mathcal{P}_i)\}$ in $\mathscr{H}_\kappa$. Let $Q = \bigcup_i \Phi(\mathcal{P}_i)$ be the collection of these points.

A (soft) consensus k-partition of $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_m$ is a partition $\mathcal{P}_{\text{con}}$ of $\bigcup_i \mathcal{P}_i$ into k clusters $\{C_1^*, \ldots, C_k^*\}$ that minimizes the sum of squared distances from each $\bar{\Phi}(C_{i,j}) \in \bar{\Phi}(\mathcal{P}_i)$ to its associated $\bar{\Phi}(C_l^*) \in \mathcal{P}_{\text{con}}$. Formally, for a set of k vectors $V = \{v_1, \ldots, v_k\} \subset \mathcal{H}_\kappa$ define

$$\textsc{Lift-SSD}(\{\mathcal{P}_i\}, V) = \sum_{C_{i,j} \in \bigcup_i \mathcal{P}_i} \frac{|C_{i,j}|}{n} \min_{v \in V} \left\| \bar{\Phi}(C_{i,j}) - v \right\|_{\mathcal{H}_\kappa}^2$$

and then define $\mathcal{P}_{\text{con}}$ as the minimum such set

$$\mathcal{P}_{\text{con}} = \operatorname*{argmin}_{V^* = \{v_1^*, \ldots, v_k^*\} \in \mathcal{H}_\kappa} \textsc{Lift-SSD}(\{\mathcal{P}_i\}_i, V^*).$$

How do we interpret $\mathcal{P}_{\text{con}}$? Observe that each element in Q is the lifted representation of some cluster $C_{i,j}$ in some partition $\mathcal{P}_i$, and therefore corresponds to some subset of P. Consider now a single cluster in $\mathcal{P}_{\text{con}}$. Since $\mathcal{H}_\kappa$ is linear and $\mathcal{P}_{\text{con}}$ minimizes distance to some set of cluster representatives, it must be in their linear combination. Hence it can be associated with a weighted subset of elements of $\Phi(P)$, and is hence a soft partition. It can be made hard by voting each point $x \in P$ to the representative $C_l^* \in \mathcal{P}_{\text{con}}$ for which it has the largest weight. Figure 3.3 shows an example of three partitions, their normalized RKHS representative vectors, and two clusters of those vectors.

3.5.2 Algorithm

We will use the approximate lifting map $\tilde{\Phi}$ in our procedure. This allows us to operate in a ρ-dimensional Euclidean space, in which there are many clustering procedures we can use. For our experiments, we will use both k-means and hierarchical agglomerative clustering (HAC). That is, let LIFT-KM be the algorithm of running k-means on $\bigcup_i \tilde{\Phi}(\mathcal{P}_i)$, and let LIFT-HAC be the algorithm of running HAC on $\bigcup_i \tilde{\Phi}(\mathcal{P}_i)$. For both algorithms, the output is the (soft) clusters represented by the vectors in $\Phi(\mathcal{P}_{\text{con}})$. Our results will show that the particular choice of clustering algorithm (e.g., LIFT-KM or LIFT-HAC) is

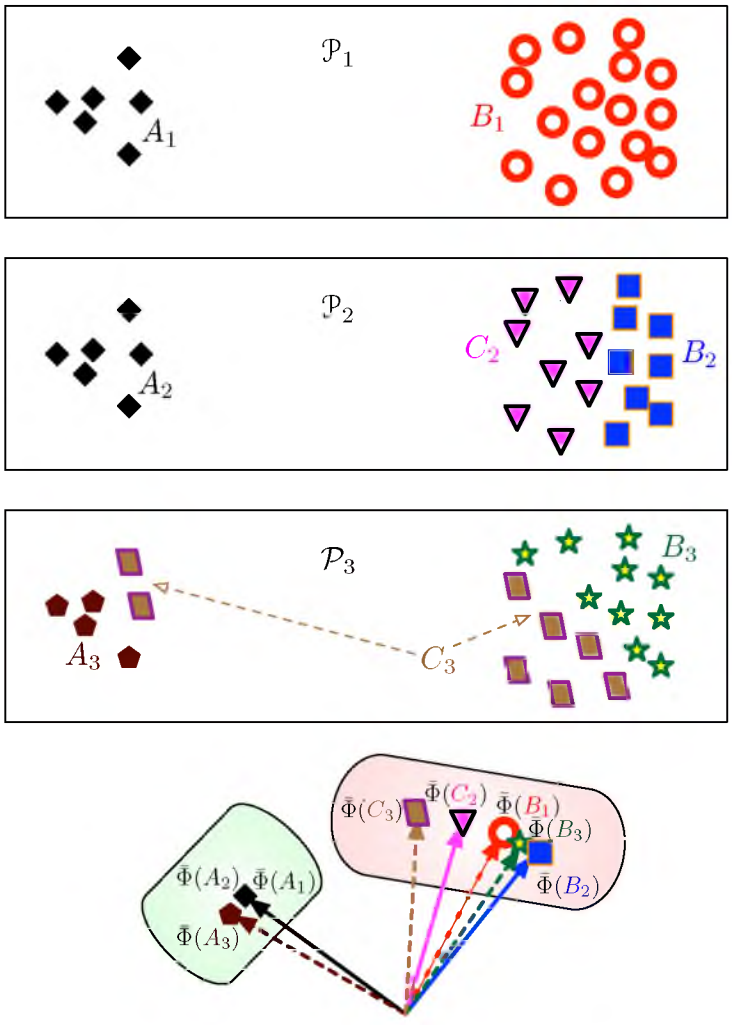


Figure 3.3. Three partitions $\mathcal{P}_1 = \{A_1, B_1\}$, $\mathcal{P}_2 = \{A_2, B_2, C_2\}$, $\mathcal{P}_3 = \{A_3, B_3, C_3\}$, and a 2-d visualization of the RKHS vectors. These vectors are then clustered into $k = 2$ consensus clusters consisting of $\{\bar{\Phi}(A_1), \bar{\Phi}(A_2), \bar{\Phi}(A_3)\}$ and $\{\bar{\Phi}(B_1), \bar{\Phi}(B_2), \bar{\Phi}(C_2), \bar{\Phi}(B_3), \bar{\Phi}(C_3)\}$.

not crucial. There are multiple methods to choose the right number of clusters and we can employ one of them to fix k for our consensus technique. Algorithm 1 summarizes our consensus procedure.

3.5.2.1 Cost analysis. Computing Q takes $O(mn\rho) = O(mn \log n)$ time. Let $|Q| = s$. Computing $\mathcal{P}_{\text{con}}$ is a single call to any standard Euclidean algorithm like k-means that takes time $O(sk\rho)$ per iteration, and computing the final soft partition takes time linear in $n(\rho + k) + s$. Note that in general we expect that $k, s \ll n$. In particular, when $s < n$ and m is assumed constant, then the runtime is $O(n(k + \log n))$.

Algorithm 1 Consensus finding

Input: (soft) partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_m$ of P, kernel function κ
Output: consensus (soft) partition $\mathcal{P}_{\text{con}}$

1: Set $Q = \cup_i \tilde{\Phi}(\mathcal{P}_i)$.
2: Compute $V^* = \{v_1^*, \ldots v_k^*\} \subset \mathcal{H}_\kappa$ to minimizes LIFT-SSD(Q, V^*).
(Via k-means for LIFT-KM or HAC for LIFT-HAC)
3: Assign each $p \in P$ to the cluster $C_i \in \mathcal{P}_{\text{con}}$ associated with the vector $v_i^* \in V^*$
- for which $\langle \tilde{\Phi}(p), v_i \rangle$ is maximized for a hard partition, or
- with weight proportional to $\langle \tilde{\Phi}(p), v_i \rangle$ for a soft partition.

4: Output $\mathcal{P}_{\text{con}}$.

3.6 Experimental Evaluation

In this section we empirically show the effectiveness of our distance between partitions, LIFTEMD and LIFTKD, and the consensus clustering algorithms that conceptually follow, LIFT-KM and LIFT-HAC.

3.6.1 Data

We created two synthetic datasets in $\mathbb{R}^2$, namely 2D2C for which data is drawn from two Gaussians to produce two visibly separate clusters and 2D3C for which the points are arbitrarily chosen to produce three visibly separate clusters. We also use five different datasets from the University of California, Irvine (UCI) machine learning repository [Frank and Asuncion, 2010] (Wine, Ionosphere, Glass, Iris, Soybean) with various numbers of dimensions and labeled data classes. To show the ability of our consensus procedure and the distance metric to handle large data, we use both the training and test data of the mixed national institute of standards and technology (MNIST) database of handwritten digits which has 60,000 and 10,000 examples, respectively, in $\mathbb{R}^{784}$.

3.6.2 Methodology

We will compare our approach with the partition-based measures, namely Rand distance and Jaccard distance and information theoretic measures, namely normalized mutual information and normalized variation of information [Wagner and Wagner, 2007], as well as the spatially aware measures D_{ADCO} [Bae et al., 2010] and CDISTANCE [Coen et al., 2010]. We ran k-means, single-linkage, average-linkage, complete-linkage and Ward's method [Tan et al., 2005] on the datasets to generate input partitions to the consensus

clustering methods. We use accuracy [Ding and Li, 2007] and Rand distance to measure the effectiveness of the consensus clustering algorithms by comparing the returned consensus partitions to the original class labeling. We compare our consensus technique against a few hypergraph partitioning based consensus methods: cluster-based similarity partitioning algorithm (CSPA), hypergraph partitioning algorithm (HGPA) and metaclustering algorithm (MCLA) [Strehl and Ghosh, 2003]. For the MNIST data we also visualize the cluster centroids of each of the input and consensus partitions in a 28x28 grayscale image.

Accuracy studies the one-to-one relationship between clusters and classes; it measures the extent to which each cluster contains data points from the corresponding class. Given a set P of n elements, consider a set of k clusters $\mathcal{P} = \{C_1, \ldots, C_k\}$ and $m \geq k$ classes $\mathcal{L} = \{L_1, \ldots, L_m\}$ denoting the ground truth partition. Accuracy is expressed as

$$\mathsf{A}(\mathcal{P}, \mathcal{L}) = \max_{\mu:[1:k]\to[1:m]} \sum_{i=1}^{k} \frac{|C_i \cap L_{\mu(i)}|}{n},$$

where μ assigns each cluster to a distinct class. The Rand distance counts the fraction of pairs, which are assigned consistently in both partitions, as in the same or in different classes. Let $R_S(\mathcal{P}, \mathcal{L})$ be the number of pairs of points that are in the same cluster in both $\mathcal{P}$ and $\mathcal{L}$, and let $R_D(\mathcal{P}, \mathcal{L})$ be the number of pairs of points that are in different clusters in both $\mathcal{P}$ and $\mathcal{L}$. Now we can define the Rand distance as

$$\mathsf{R}(\mathcal{P}, \mathcal{L}) = 1 - \frac{R_S(\mathcal{P}, \mathcal{L}) + R_D(\mathcal{P}, \mathcal{L})}{\binom{n}{2}}.$$

3.6.3 Code

We implement $\tilde{\Phi}$ as the random projection feature map [Joshi et al., 2011] in C to lift each data point into $\mathbb{R}^\rho$. We set $\rho = 200$ for the two synthetic datasets 2D2C and 2D3C and all the UCI datasets. We set $\rho = 4000$ for the larger datasets, MNIST training and MNIST test. The same lifting is applied to all data points, and thus all clusters.

The LIFTEMD, LIFTKD, and LIFTH distances between two partitions $\mathcal{P}$ and $\mathcal{P}'$ are computed by invoking brute force transportation distance, kernel distance, and Hausdorff distance on $\tilde{\Phi}(\mathcal{P}), \tilde{\Phi}(\mathcal{P}') \subset \mathbb{R}^\rho$ representing the lifted clusters.

To compute the consensus clustering in the lifted space, we apply k-means (for LIFT-KM) or HAC (for LIFT-HAC) (with the appropriate numbers of clusters) on the set $Q \subset \mathbb{R}^\rho$

of all lifted clusters from all partitions. The only parameters required by the procedure are the error term ε (needed in our choice of ρ) associated with $\tilde{\Phi}$, and any clustering-related parameters.

We used the cluster analysis functions in MATLAB with the default settings to generate the input partitions to the consensus methods and the given number of classes as the number of clusters. We implemented the algorithm provided by the authors [Bae et al., 2010] in MATLAB to compute D_{ADCO}. To compute CDISTANCE, we used the code provided by the authors [Coen et al., 2010]. We used the ClusterPack MATLAB toolbox [Strehl, Strehl] to run the hypergraph partitioning based consensus methods CSPA, HGPA and MCLA.

3.6.4 Spatial Sensitivity

We start by evaluating the sensitivity of our method. We consider three partitions: the reference partition (RP), and manually constructed first and second partitions (FP and SP) for the datasets 2D2C (see Figure 3.1) and 2D3C (see Figure 3.4). For both the datasets, the reference partition is, by construction, spatially closer to the first partition than the second partition, but each of the two partitions are equidistant from the reference under any partition-based and information theoretic measures. Table 3.1 shows that in each example, our measures correctly conclude that RP is closer to FP than it is to SP.

3.6.5 Efficiency

We compare our distance computation procedure to CDISTANCE. We do not compare against D_{ADCO} and CC because they are not well founded.

Both LIFTEMD and CDISTANCE compute d_T between clusters after an initial step of either lifting to a feature space or computing d_T between all pairs of clusters. Thus the proper comparison, and runtime bottleneck, is the initial phase of the algorithms; LIFTEMD takes $O(n \log n)$ time, whereas CDISTANCE takes $O(n^3)$ time. Table 3.2 summarizes our results. For instance, on the 2D3C dataset with $n = 24$, our initial phase takes 1.02 milliseconds, and CDISTANCE's initial phase takes 2.03 milliseconds. On the Wine dataset with $n = 178$, our initial phase takes 6.9 milliseconds, while CDISTANCE's initial phase takes 18.8 milliseconds. As the dataset size increases, the advantage of LIFTEMD over CDISTANCE becomes even larger. On the MNIST training data with $n = 60,000$, our initial phase takes a little less than 30 minutes, while CDISTANCE's initial phase takes

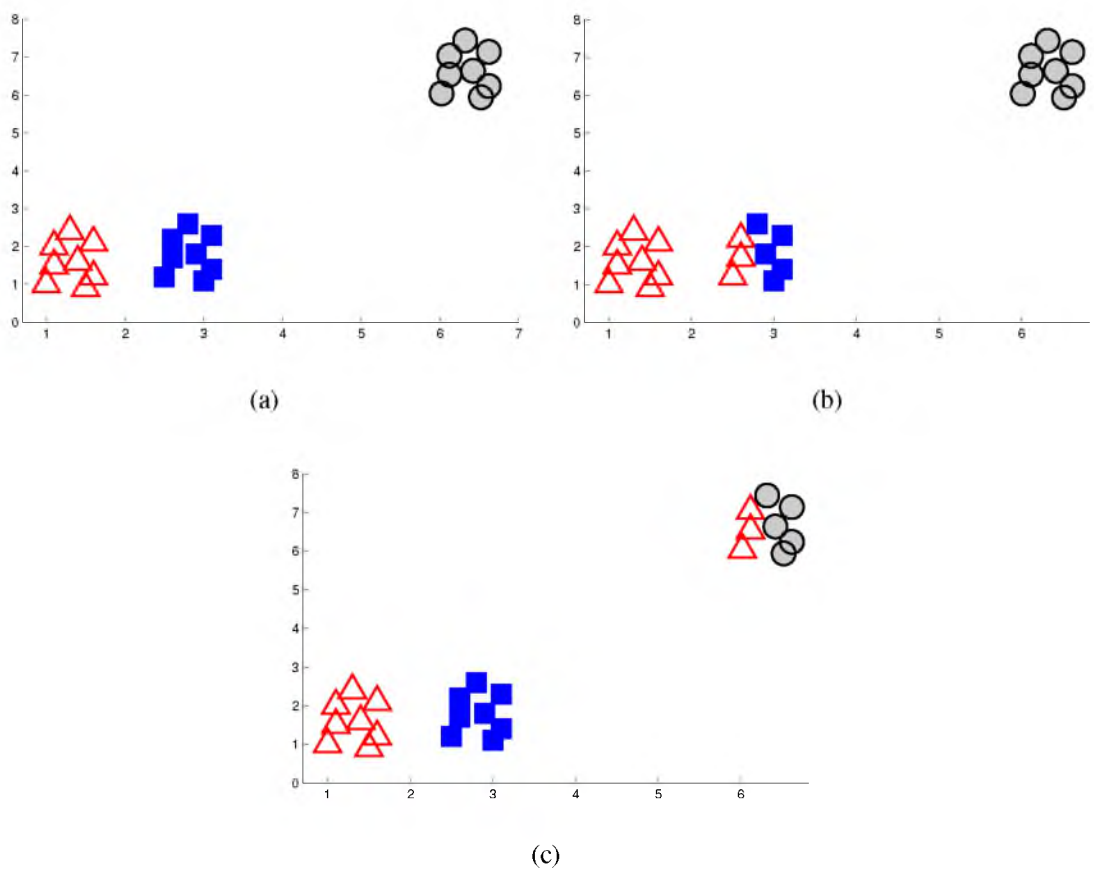


Figure 3.4. Different partitions ((a) RP, (b) FP and (c) SP) of 2D3C dataset.

Table 3.1. Comparing partitions: each cell indicates the distance returned under the methods along the rows for the dataset in the column. Spatially, the left column of each dataset (2D2C or 2D3C) should be smaller than the right column; this holds for all five spatial measures/algorithms tested. In all cases, the two partition-based measures and the two information theoretic measures yield the same values for d(RP, FP) and d(RP, SP), but are not shown.

Technique	Dataset 2D2C		Dataset 2D3C	
	d(RP,FP)	d(RP,SP)	d(RP,FP)	d(RP,SP)
D_{ADCO}	1.710	1.780	1.790	1.820
CDISTANCE	0.240	0.350	0.092	0.407
LIFTEMD	0.430	0.512	0.256	0.310
LIFTKD	0.290	0.325	0.243	0.325
LIFTH	0.410	0.490	1.227	1.291

Table 3.2. Comparison of runtimes: distance between true partition and partition generated by k-means)

Dataset	Number of points	Number of dimensions	CDistance	LIFTEMD
2D3C	24	2	2.03 ms	1.02 ms
2D2C	45	2	4.10 ms	1.95 ms
Wine	178	13	18.80 ms	6.90 ms
MNIST test data	10,000	784	1360.20 s	303.90 s
MNIST training data	60,000	784	202681 s	1774.20 s

more than 56 hours.

3.6.6 Consensus Clustering

We now evaluate our spatially aware consensus clustering method. We do this first by comparing our consensus partition to the reference solution based on using the Rand distance (i.e., a partition-based measure) in Table 3.3. Note that for all datasets, our consensus clustering methods (LIFT-KM and LIFT-HAC) return answers that are almost always as close as the best answer returned by any of the hypergraph partitioning based consensus methods, CSPA, HGPA, or MCLA. We get very similar results using the accuracy [Ding and Li, 2007] measure in place of Rand.

In Table 3.4, we then run the same comparisons, but this time using LIFTEMD (i.e., a spatially aware measure). Here, it is interesting to note that in all cases (with the slight exception of Ionosphere) the distance we get is smaller than the distance reported by the hypergraph partitioning based consensus methods, indicating that our method is returning a consensus partition that is spatially closer to the true answer. The two tables also illustrate the flexibility of our framework, since the results using LIFT-KM and LIFT-HAC are mostly identical (with one exception being IRIS under LIFTEMD).

To summarize, our method provides results that are comparable or better on partition-based measures of consensus, and are superior using spatially aware measures. The running time of our approach is comparable to the best hypergraph partitioning based approaches, so using our consensus procedure yields the best overall result.

We also run consensus experiments on the MNIST test data and compare against CSPA

Table 3.3. Comparison of LIFT-KM and LIFT-HAC with hypergraph partitioning based consensus methods under the Rand distance (with respect to ground truth). The numbers are comparable across each row corresponding to a different dataset, and smaller numbers indicate better accuracy. The top two methods for each dataset are highlighted.

Dataset	CSPA	HGPA	MCLA	LIFT-KM	LIFT-HAC
IRIS	**0.088**	0.270	0.115	**0.114**	0.125
Glass	**0.277**	**0.305**	0.428	0.425	0.430
Ionosphere	0.422	0.502	**0.410**	0.420	**0.410**
Soybean	0.188	**0.150**	0.163	**0.150**	0.154
Wine	**0.296**	0.374	0.330	0.320	**0.310**
MNIST test data	0.149	-	0.163	**0.091**	**0.110**

Table 3.4. Comparison of LIFT-KM and LIFT-HAC with hypergraph partitioning based consensus methods under LIFTEMD (with respect to ground truth). The numbers are comparable across each row corresponding to a different dataset, and smaller numbers indicate better accuracy. The top two methods for each dataset are highlighted.

Dataset	CSPA	HGPA	MCLA	LIFT-KM	LIFT-HAC
IRIS	**0.113**	0.295	0.812	**0.106**	0.210
Glass	0.573	**0.519**	0.731	**0.531**	0.540
Ionosphere	**0.729**	0.767	0.993	0.731	**0.720**
Soybean	0.510	0.495	0.951	**0.277**	**0.290**
Wine	0.873	0.875	0.917	**0.831**	**0.842**
MNIST test data	0.182	-	0.344	**0.106**	**0.112**

and MCLA. We do not compare against HGPA since it runs very slow for the large MNIST datasets ($n = 10,000$); it has quadratic complexity in the input size, and in fact, the authors do not recommend this for large data.

Figure 3.5 provides a visualization of the cluster centroids of input partitions generated using k-means, complete linkage HAC and average linkage HAC and the consensus partitions generated by CSPA and LIFT-KM.

From the k-means input, only five clusters can be easily associated with digits ("0", "3", "6", "8", "9"); from the complete linkage HAC input, only seven clusters can be easily associated with digits ("0", "1", "3", "6", "7", "8", "9"); and from the average linkage HAC output, only six clusters can be easily associated with digits ("0", "1", "2", "3", "7", "9"). The partition that we obtain from running CSPA lets us identify up to six digits ("0", "1",

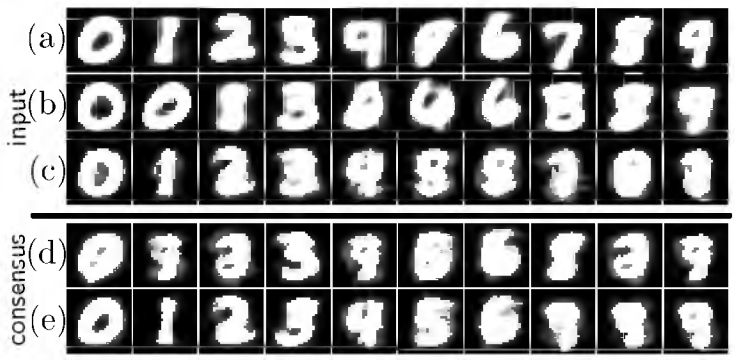


Figure 3.5. 28x28 pixel representation of the cluster centroids for MNIST test input partitions generated using (a) k-means, (b) complete linkage HAC, and (c) average linkage HAC, and the consensus partitions generated by (d) CSPA and (e) LIFT-KM.

"2", "3", "8", "9"). In these cases, there occurs cases where two clusters seem to represent the same digit. In contrast, we can identify nine digits ("0", "1", "2", "3", "4", "5", "7", "8", "9") with only the digit ("6") being noisy from our LIFT-KM output.

3.6.7 Error in $\tilde{\Phi}$

There is a tradeoff between the desired error ε in computing LIFTEMD and the number of dimensions ρ needed for $\tilde{\Phi}$. Our empirical evaluations show that setting ρ between 100 and 1000 suffices to bring the error in LIFTEMD down to $\varepsilon = 0.005$ or less on a variety of datasets. Figure 3.6 shows the error as a function of ρ on the 2D2C dataset ($n = 45$). From the chart, we can see that $\rho = 100$ dimensions suffice to yield a very accurate approximation for the distances. Figure 3.7 shows the error as a function of ρ on the MNIST training dataset that has $n = 60,000$ points. From the chart, we can see that $\rho = 4,000$ dimensions suffice to yield a very accurate approximation for the distances.

3.7 Summary

We provide a well founded spatially aware metric between partitions based on a RKHS representation of clusters that captures the true properties that lead to the formation of clusters in the first place. We also introduce a spatially aware consensus clustering formulation using this representation that reduces to Euclidean clustering, thus allowing us to leverage all the research in clustering to run a consensus. We demonstrate that our algorithms are efficient and are comparable to or better than prior methods.

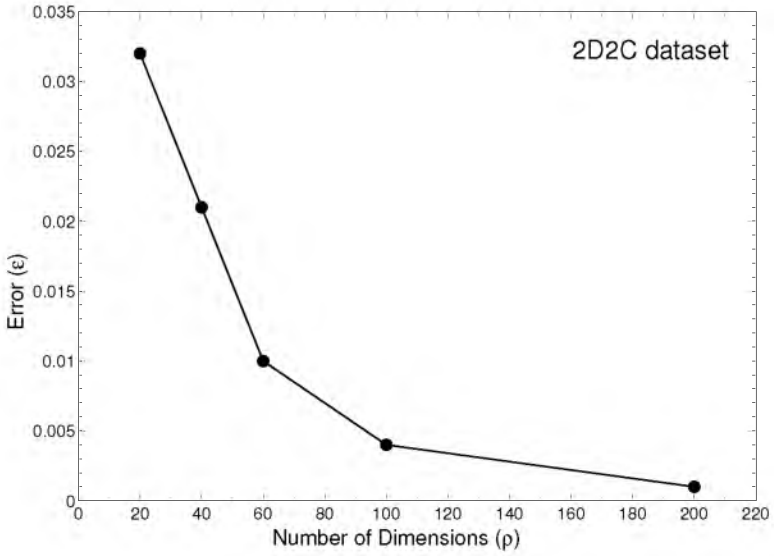


Figure 3.6. Error in LIFTEMD on 2D2C dataset (45 samples) as a function of ρ.

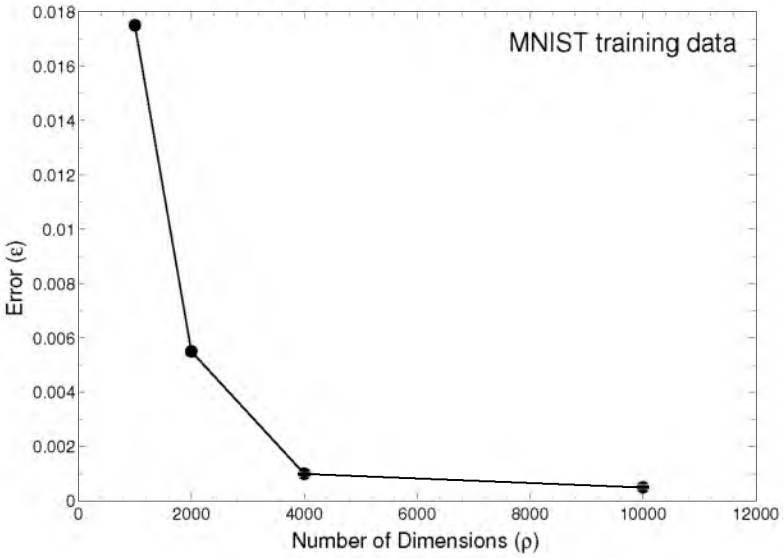


Figure 3.7. Error in LIFTEMD on MNIST training data (60,000 samples) as a function of ρ.

CHAPTER 4

GENERATING THE LANDSCAPE OF PARTITIONS[1]

When one partition is not good enough to understand the structures present in the data, data miners look to uncover more partitions. We illustrate this idea with an illustration (see Figure 4.1) of different partitions on a subset of the multimedia information retrieval flickr (MIRFLICKR-25000) [Huiskes and Lew, 2008] dataset. Each clustering method identifies different kinds of structure in data, reflecting different desires of the end user. Thus, when used as an exploratory tool for data analysis, there is a need to identify a diverse and meaningful collection of partitions of a dataset, in the hope that these distinct partitions will yield different insights about the underlying data. A common metaclustering problem is alternative clustering [Caruana et al., 2006; Dang and Bailey, 2010a,b; Gondek and Hofmann, 2004; Jain et al., 2008; Qi and Davidson, 2009] where the goal is to generate many nonredundant partitions of good quality. In general, the alternative clustering methods employ one of the distances discussed above, as well as external notions of quality. Most algorithms for generating alternative partitions operate as follows. Generate a single partition using a clustering algorithm of choice. Next, find another partition that is both far from the first partition and of high quality. Most methods stop here, but a few methods try to discover more alternative partitions; they repeatedly find new, yet high quality, partitions that are far from all existing partitions.

The input to our problem is a single dataset X. The output is a set of k partitions of X. A **partition** of X is a set of subsets $\mathcal{X}_i = \{X_{i,1}, X_{i,2}, \ldots, X_{i,s}\}$ where $X = \bigcup_{j=1}^{s} X_{i,j}$ and for all j, j' $X_{i,j} \cap X_{i,j'} = \emptyset$. Let $\mathcal{P}_X$ be the space of all partitions of X; since X is fixed throughout the discussions in this chapter, we just refer to this space as $\mathcal{P}$.

---

[1] Reprinted with permission of CEUR-WS, 2011, Jeff M. Phillips, Parasaran Raman, and Suresh Venkatasubramanian, Generating a Diverse Set of High-Quality Clusterings. 2nd MultiClust Workshop: Discovering, Summarizing and Using Multiple Clusterings, Vol. 772, Pages 81-90.

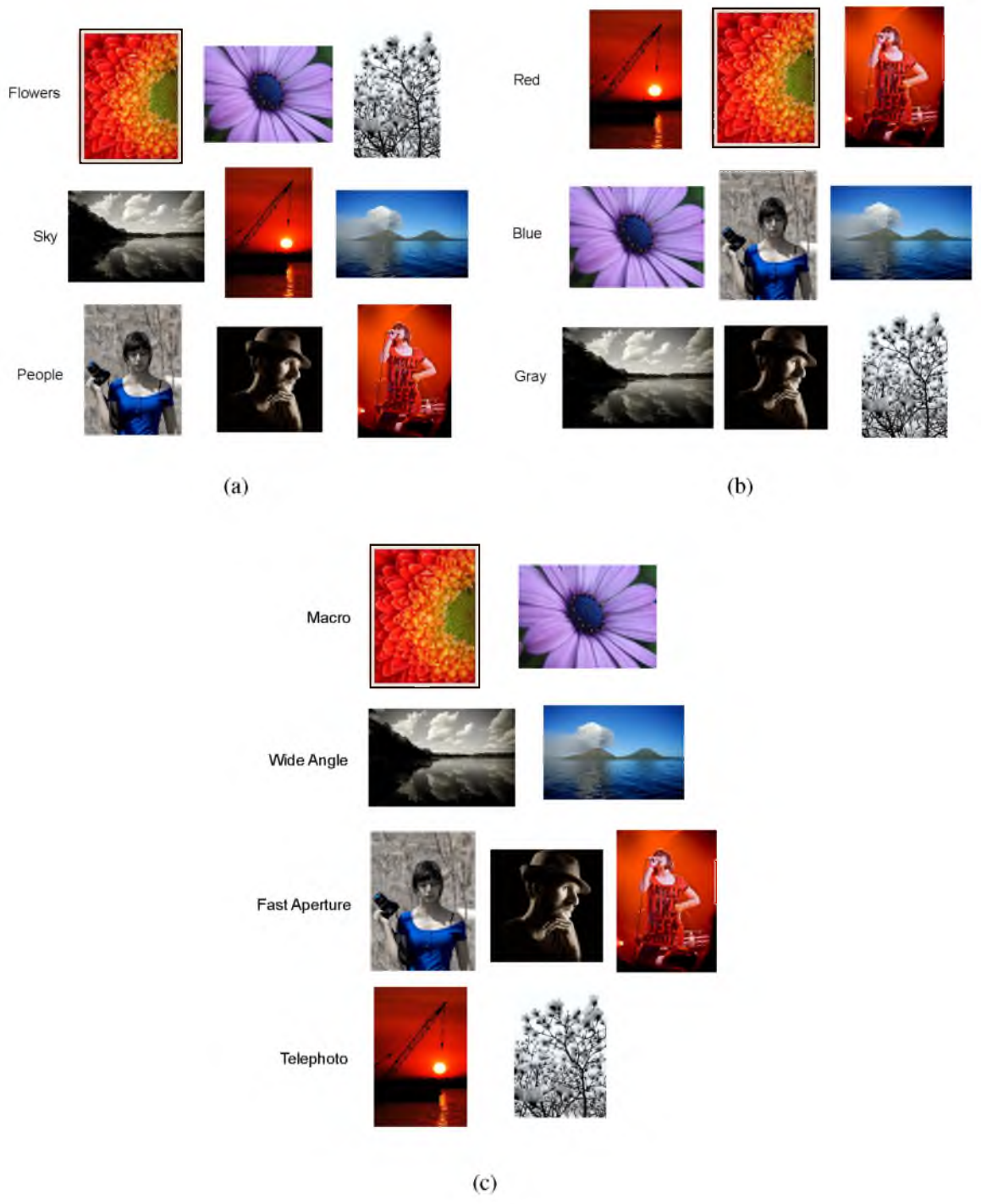


Figure 4.1. Three possible partitions based on (a) object contained, (b) dominant color, and (c) type of lens used on a subset of MIRFLICKR-25000 dataset. All images are offered under creative commons copyright licenses.

There are two quantities that control the nature of the partitions generated. The **quality** of a partition, represented by a function $Q : \mathcal{P} \rightarrow \mathbb{R}^+$, measures the degree to which a particular partition captures intrinsic structure in data; in general, most clustering algorithms that identify a single partition attempt to optimize some notion of quality. The **distance** between partitions, represented by the function $d : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$, is a quantity measuring how dissimilar two partitions are. The partitions $\mathcal{X}_i \in \mathcal{P}$ that do a better job of capturing the structure of the dataset X will have a larger quality value $Q(\mathcal{X}_i)$. The partitions $\mathcal{X}_i, \mathcal{X}_{i'} \in \mathcal{P}$ that are more similar to each other will have a smaller distance value $d(\mathcal{X}_i, \mathcal{X}_{i'})$. A good set of diverse partitions all have large distances from each other and all have high quality scores.

Thus, the goal here is to generate a set of k partitions that best represent all high quality partitions as accurately as possible.

4.1 Overview of Our Work

To generate multiple good partitions, we present a new paradigm which decouples the notion of distance between partitions and the quality of partitions. Prior methods that generate multiple diverse partitions cannot explore the space of partitions entirely since the distance component in their objective functions biases against partitions close to the previously generated ones. These could be interesting partitions that might now be left out. Also, the methods which use both the quality and the distance term in the objective function suffer from the problem of tradingoff between two incomparable terms. To avoid this, we will first look at the space of all partitions more thoroughly and then pick nonredundant partitions from this set. Let k be the number of diverse partitions that we seek. Our approach works in two steps.

1. Generation step: we first sample from the space of all partitions proportional to their quality. Stirling numbers of the second kind, $S(n,s)$ are the number of ways of partitioning a set of n elements into s nonempty subsets. Therefore, this is the size of the space that we sample from. We illustrate the sampling in Figure 4.2. This generates a set of size $m \gg k$ to ensure we get a large sample that represents the space of all partitions well. We generate a conservative number of samples to avoid "accidentally" missing some high quality region of $\mathcal{P}$.

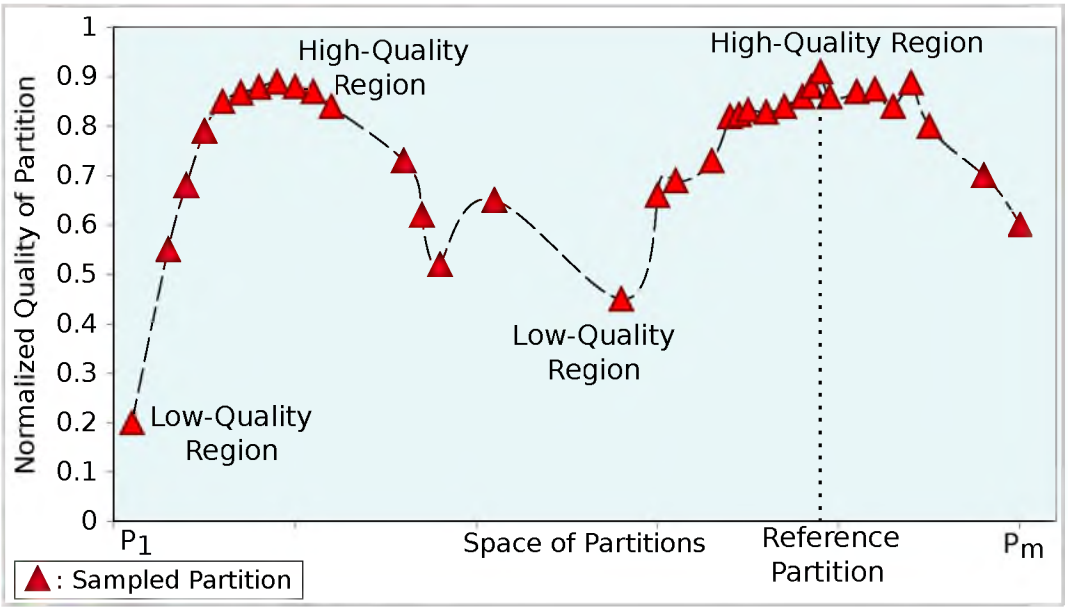


Figure 4.2. Sampling partitions proportional to its quality from the space of all partitions with s clusters.

2. Grouping step: we cluster this set of m partitions into k sets, resulting in k clusters of partitions. We then return one representative from each of these k clusters as our output alternative partitions.

Note that because the generation step is decoupled from the grouping step, we treat all partitions fairly, independent of how far they are from the existing partitions. This allows us to explore the true density of high quality partitions in $\mathcal{P}$ without interference from the choice of initial partition. Thus, if there is a dense set of close interesting partitions our approach will recognize that. Apart from helping us to understand the landscape of the partitions of the given dataset, this can also provide different statistical insights like the number of modes present in the data and the "clusterability" of the data. Since the grouping step is run separate from the generation step, we can abstract this problem to a generic clustering problem and we can choose one of many approaches. This allows us to capture different properties of the diversity of partitions and understand the structure of the landscape of partitions. Also, this allows the user to pick a distance measure of choice either guided just by the spatial distance between partitions, or also by a density-based distance which only takes into account the number of high quality partitions assigned to a cluster.

From our experimental evaluation, we note that decoupling the generation step from the grouping step helps as we are able to generate a lot of very high quality partitions. In fact, the quality of some of the generated partitions is better than the quality of the partition obtained by a consensus clustering technique called LIFTSSD [Raman et al., 2011]. The relative quality with respect to the reference partition of a few generated partitions even reach close to one. To our best knowledge, such partitions have not been uncovered by other previous metaclustering techniques. The grouping step also picks out representative partitions faraway from each other. We observe this by computing the closest-pair distance between representatives and comparing it against the distance values of the partitions to their closest representative.

4.1.1 Comparison to Prior Work

The existing literature in alternative clustering focuses on generating one additional partition of high quality that should be far from a given set (typically of size one) of existing partitions.

Most algorithms for generating alternative partitions operate as follows. Generate a single partition using a clustering algorithm of choice. Next, find another partition that is both far from the first partition and of high quality. Most methods stop here, but a few methods try to discover more alternative partitions; they repeatedly find new, still high quality, partitions that are far from all existing partitions. However, because of the way the objective function is formulated these procedures often suffer a loss in the quality of the partitions generated due to the constraints. Although there are a few other methods that try to discover alternative partitions simultaneously [Jain et al., 2008; Niu et al., 2010], they are usually limited to discovering two partitions of the data. In both cases, these subproblems avoid the full objective of constructing a diverse set of partitions that explore and represent the landscape of all high quality partitions.

The existing alternative clustering approaches are often too reliant on the initial partition and only have limited success in generalizing the initial step to generate k partitions. In the second approach to selecting diverse partitions from an input collection of partitions, there is no way to verify that the input represents the space of all high quality partitions, so a representative set of those input partitions is not necessarily a representative set of all high quality partitions.

4.1.2 Outline

In Section 4.2, we discuss a sampling-based approach for generating many partitions proportional to their quality; i.e., the higher the quality of a partition, the more likely it is to be sampled. In Section 4.3, we describe how to choose k representative partitions from the large collection of partitions already generated. We will present the results of our approach in Section 4.4. We have tested our algorithms on a synthetic dataset, a standard clustering dataset from the UCI repository and a subset of images from the extended Yale face database B.

4.2 Generating Many High Quality Partitions

In this section we describe how to generate many high quality partitions. This requires (1) a measure of quality and (2) an algorithm that generates a partition with probability proportional to its quality.

4.2.1 Quality of Partitions

Most work on clustering validity criteria looks at a combination of how compact clusters are and how separated two clusters are. Some of the common measures that follow this theme are SDbw, CDbw, SD validity index, maximum likelihood, and Dunn index [Aldrich, 1997; Dave, 1996; Dunn, 1974; Halkidi and Vazirgiannis, 2001; Halkidi et al., 2000; MacKay, 2002; Milligan and Cooper, 1985; Theodoridis and Koutroumbas, 2006]. Ben-David and Ackerman [2008] also discuss similar notions of quality, namely VR (variance ratio) and WPR (worst pair ratio) in their study of clusterability. We briefly describe a few specific notions of quality below.

4.2.1.1 K-means quality. If the elements $x \in X$ belong to a metric space with an underlying distance $\delta : X \times X \to \mathbb{R}$ and each cluster $X_{i,j}$ in a partition $\mathcal{X}_i$ is represented by a single element x_j, then we can measure the inverse quality of a cluster by $q(X_{i,j}) = \sum_{x \in X_{i,j}} \delta(x, x_j)^2$. The quality of the entire partition is then the inverse of the sum of the inverse qualities of the individual clusters,

$$Q_{km}(\mathcal{X}_i) = 1/(\sum_{j=1}^{s} q(X_{i,j})). \tag{4.2.1}$$

This corresponds to the quality optimized by s-means clustering[2], which is quite common, but is susceptible to outliers. If all but one element of X fit neatly in s clusters, but the one remaining point is faraway, then this one point dominates the cost of the clustering, even if it is effectively noise. Specifically, the quality score of this measure is dominated by the points which fit least well in the clusters, as opposed to the points which are best representative of the true data. Hence, this quality measure may not paint an accurate picture about the partition.

4.2.1.2 Kernel distance quality. We introduce a method to compute quality of a partition, based on the kernel distance [Joshi et al., 2011]. Here we start with a similarity function between two elements of X, typically in the form of a (positive definite) kernel: $K : X \times X \to \mathbb{R}^+$. $K(x_u, x_v)$ is smaller when $x_u, x_v \in X$ are less similar, and takes a maximum value of 1 when $x_u = x_v$. Then the overall (normalized) similarity score between two clusters $X_{i,j}, X_{i,j'} \in \mathcal{X}_i$ is defined by

$$\kappa(X_{i,j}, X_{i,j'}) = \frac{1}{|X_{i,j}| \cdot |X_{i,j'}|} \sum_{x \in X_{i,j}} \sum_{x' \in X_{i,j'}} K(x, x'), \tag{4.2.2}$$

and the self similarity of a single cluster $X_{i,j} \in \mathcal{X}_i$ is defined by $\kappa(X_{i,j}, X_{i,j})$. Finally, the overall quality of a partition is defined by the equation,

$$Q_W(\mathcal{X}_i) = \sum_{j=1}^{s} \kappa(X_{i,j}, X_{i,j}). \tag{4.2.3}$$

This captures the compactness (also called the width) of each cluster. We also define another quality measure that captures both the compactness and separation (also called the split) between the clusters. This is defined as the difference between the square of the self similarity term and the cross similarity term,

$$Q_{W-S}(\mathcal{X}_i) = [\sum_{j=1}^{s} \kappa(X_{i,j}, X_{i,j})]^2 - \sum_{\ell=1}^{s} \sum_{j=1}^{s} \kappa(X_{i,\ell}, X_{i,j}). \tag{4.2.4}$$

We use the square of the self similarity term to account for the fact that there are only s normalized similarity scores in the self similarity term while there are s^2 of them in the

[2]It is commonplace to use k in place of s, but we reserve k for other notions in this chapter

cross similarity term. Here the cross similarity terms account for the similarity between clusters and we subtract this to achieve a good separation between the clusters. We define these two notions of quality to understand the effect of the "right" quality function on the clustering landscape generation procedure.

If X is a metric space, the highest quality partitions divide X into s Voronoi cells around s points – similar to s-means clustering. However, its score is dominated by the points which are a good fit to a cluster, rather than outlier points which do not fit well in any cluster. This is a consequence of how kernels like the Gaussian kernel taper off with distance and is the reason we recommend this measure of cluster quality in our experiments. We illustrate this property in Figure 4.3. It is important to note that the choice of the measure to compute the quality of the partition is not tied to the process of generating the partitions. However, we observe that good quality measures generate better partitions.

4.2.2 Generation of Partitions Proportional to Quality

We now discuss how to generate a sample of partitions proportional to their quality. This procedure will be independent of the measure of quality used, so we will generically let $Q(\mathcal{X}_i)$ denote the quality of a partition. Now the problem becomes to generate a set $Z \subset \mathcal{P}$ of partitions where each $\mathcal{X}_i \in Z$ is drawn randomly proportional to $Q(\mathcal{X}_i)$.

The standard tool for this problem framework is a Metropolis-Hastings (M-H) random walk sampling procedure [Hastings, 1970; Hoff, 2009; Metropolis et al., 1953]. Given a domain X to be sampled and an energy function $Q : X \to \mathbb{R}$, we start with a point $x \in X$ and suggest a new point x_1 that is typically "near" x. The point x_1 is accepted unconditionally if $Q(x_1) \geq Q(x)$ and is accepted with probability $Q(x_1)/Q(x)$ if not. Otherwise, we say that x_1 was rejected and instead set $x_1 = x$ as the current state. After some sufficiently large number of such steps t, the expected state of x_t is a random draw from $\mathcal{P}$ with probability proportional to Q. To generate many random samples from $\mathcal{P}$ this procedure is repeated many times.

In general, M-H sampling suffers from high autocorrelation, where consecutive samples are too close to each other. This can happen when faraway samples are rejected with high probability. To counteract this problem, often Gibbs sampling is used [Davidson, 2000; Roberts et al., 1997]. Here, each proposed step is decomposed into several orthogonal suggested steps and each is individually accepted or rejected in order. This effectively

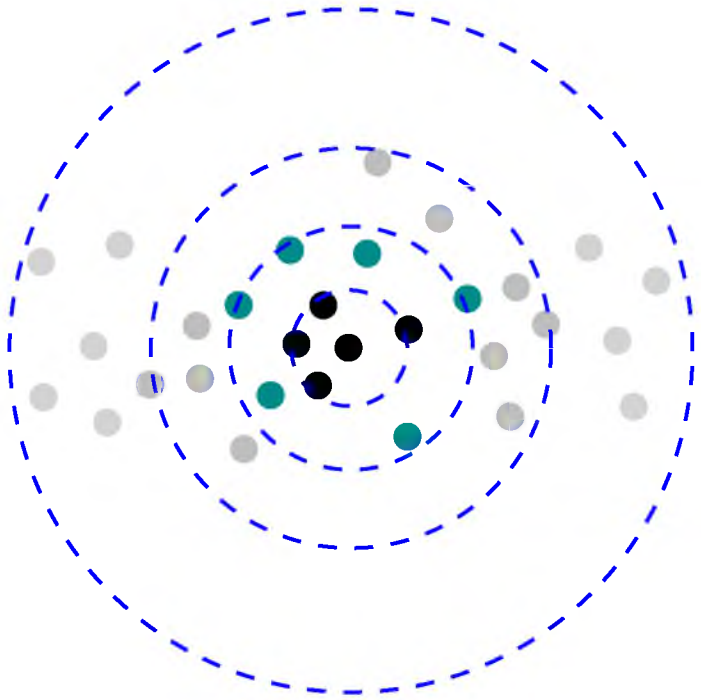


Figure 4.3. The kernel distance quality with a Gaussian kernel penalizes the outliers with the black points given much more importance than the gray ones.

constructs one longer step with a much higher probability of acceptance since each individual step is accepted or rejected independently. Furthermore, if each step is randomly made proportional to Q, then we can always accept the suggested step, which reduces the rejection rate.

4.2.2.1 Metropolis-Hastings-Gibbs sampling for partitions. The M-H procedure for partitions works as follows. Given a partition $\mathcal{X}_i$, we wish to select a random subset $W \subset X$ and randomly reassign the elements of W to different clusters. If the size of W is large, this will have a high probability of rejection, but if W is small, then the consecutive clusters will be very similar. Thus, we use a special case of the M-H procedure called Gibbs sampling. Gibbs sampling is a common method when the joint distribution is either not known or is difficult to sample from, but the conditional distribution of each variable is known. The samples form a Markov chain and approximate the joint distribution of the variables. At each step we choose a random ordering σ of the elements of X. Instead of probabilistically picking the next state all at once in the random walk, Gibbs sampling makes a separate probabilistic choice for each variable. Our separate probabilistic choice

at each step is the reassignment of each data point.

We start with the current partition $\mathcal{X}_i$ and choose the first element $x_{\sigma(1)} \in X$. We assign $x_{\sigma(1)}$ to each of the s clusters generating s suggested partitions $\mathcal{X}_i^j$ and calculate s quality scores $q_j = Q(\mathcal{X}_i^j)$. Note that we can divide each q_j by $\sum_j Q(\mathcal{X}_i^j)$ thereby making the ratio of acceptance probabilities into actual probabilities. Finally, we select index j with probability q_j and assign $x_{\sigma(1)}$ to cluster j. Rename the new partition as $\mathcal{X}_i$. We illustrate this procedure in Figure 4.4 with a partition consisting of three clusters. We repeat this for all points in order. Finally, after all elements have been reassigned, we set $\mathcal{X}_{i+1}$ to be the resulting partition.

Note that autocorrelation effects may still occur since we tend to have partitions with high quality, but this effect will be much reduced. We run this entire procedure each time we need a new random sample. It is common in practice to run this procedure for some number t_0 (typically $t_0 = 1000$) of burn in steps, and then use the next m steps as m random samples from $\mathcal{P}$. The rationale is that after the burn in period, the induced Markov chain is expected to have mixed. Therefore, each new step would yield a random sample from the stationary distribution and the correlation of the consecutive samples will dissipate over the m steps.

4.3 Grouping the Partitions

Having generated a large collection Z of size $m \gg k$ high quality partitions from $\mathcal{P}$ by random sampling, we now describe a grouping procedure that returns k representative partitions from this collection. We will start by placing a metric structure on $\mathcal{P}$. This allows us to view the problem of grouping as a metric clustering problem where each "point" is a partition. Our approach is independent of any particular choice of metric; obviously, the specific choice of distance metric and clustering algorithm will affect the properties of the output set we generate. There are many different approaches to comparing partitions. Since our approach is independent of the particular choice of distance measure used, we review the main classes to give the user an idea of the various kinds of choices available to compare partitions.

1. **Membership-based distances**. The most commonly used class of distances used to compare partitions is membership-based. These distances compute statistics about

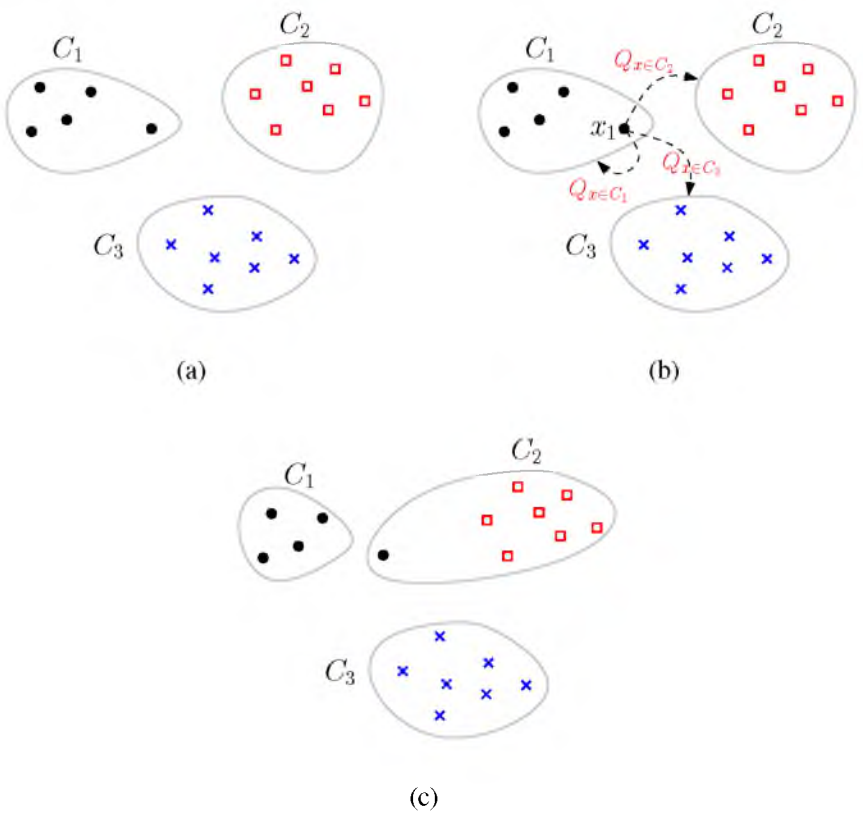


Figure 4.4. Gibbs sampling by reconfiguring a partition proportional to quality. The intial configuration is shown in (a), the three possible moves are explained in (b), and the final configuration is shown in (c).

the number of pairs of points which are placed in the same or different cluster in both partitions and return a distance based on these statistics. Common examples include the Rand distance, the variation of information, and the normalized mutual information [Ben-Hur et al., 2002; Meilă, 2007; Rand, 1971; Strehl and Ghosh, 2003]. While these distances are quite common, they ignore information about the spatial distribution of points within clusters, and so are unable to differentiate between partitions that might be significantly different.

2. **Spatially sensitive distances**. In order to rectify this problem, a number of spatially aware measures have been proposed. In general, they work by computing a concise representation of each cluster and then use the earthmover's distance (EMD) [Givens and Shortt, 1984] to compare these sets of representatives in a spatially aware manner. These include CDistance [Coen et al., 2010], d_{ADCO} [Bae et al., 2010], CC

distance [Zhou et al., 2005], and LIFTEMD [Raman et al., 2011]. LIFTEMD has the benefit of being both efficient as well as a well founded metric and is the method used here.

3. **Density-based distances**. The partitions we consider are generated via a sampling process that samples more densely in high quality regions of the space of partitions. In order to take into account dense samples in a small region, we use a density sensitive distance that intuitively spreads out regions of high density. Consider two partitions $\mathcal{X}_i$ and $\mathcal{X}_{i'}$. Let $d : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$ be any of the above natural distances on $\mathcal{P}$. Then let $d_Z : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$ be a density-based distance defined as $d_Z(\mathcal{X}_i, \mathcal{X}_{i'}) = |\{\mathcal{X}_l \in Z \mid d(\mathcal{X}_i, \mathcal{X}_l) < d(\mathcal{X}_i, \mathcal{X}_{i'})\}|$.

4.3.1 Clusters of Partitions

Once we have specified a distance measure to compare partitions, we can cluster them. See Figure 4.5 for a 2-dimensional visualization of picking nonredundant partitions from the landscape generated. We will use the notation $\phi(\mathcal{X}_i)$ to denote the representative partition $\mathcal{X}_i$ is assigned to. The goal is to pick k representative partitions which are "diverse" from Z. We use two common clustering methods described below after computing the LIFTEMD distance between all pairs of partitions.

1. We run average-linkage hierarchical agglomerative clustering on the distance matrix. Hierarchical clustering methods are suited for our problem since we have the distances matrix rather than the points themselves (since the "points" are the partitions). The algorithm starts by putting two clusters (of partitions) together whose average distance between the partitions contained in them is minimum. After the merging is complete, we make a cut in the dendogram to obtain the clusters at level k to obtain k clusters of partitions.
2. A simple algorithm by Gonzalez [1985] provides a 2-approximation to the best partition that minimizes the maximum distance between a point and its assigned center. The algorithm maintains a set of centers $k' < k$ in C. The algorithm chooses $\mathcal{X}_i \in Z$ with maximum value $d(\mathcal{X}_i, \phi(\mathcal{X}_i))$. It adds this partition $\mathcal{X}_i$ to C and repeats until C contains k partitions. We then assign each of the remaining partitions to the closest of the $k \in C$ partitions.

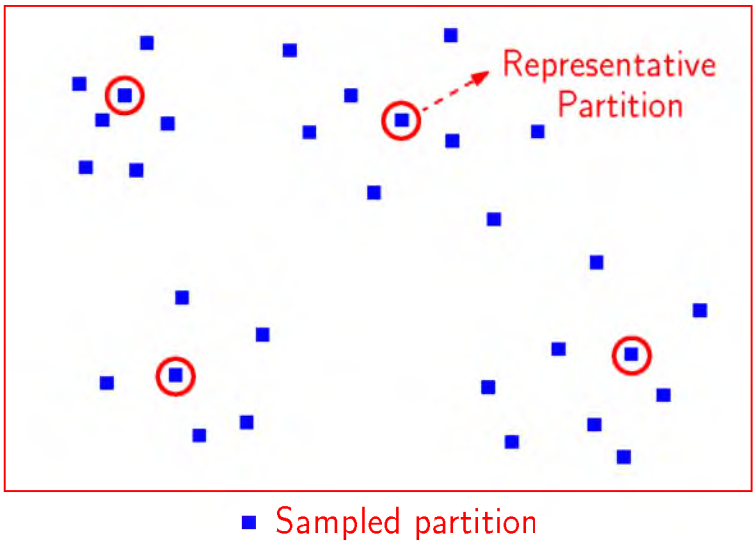


Figure 4.5. A 2-d illustration of the space of partitions. Each square is a sampled partition and there are four groups of partitions from which we pick one representative each.

Once we have the clusters of partitions, we pick the best quality partition from each cluster as the representative partition to ensure better overall quality. We run both the hierarchical clustering and the Gonzalez method using LIFTEMD as the distance between partitions. Since these methods are independent of the choice of the distance measure used to compare the partitions, we can substitute LIFTEMD with any other choice of distance between partitions. Also, the clustering methods that we use are picked because they are easy to run on the distance matrix. Since the only goal is to pick k faraway partitions, we can replace these choices with any clustering method that can take as input an all pairs distance matrix.

4.4 Experimental Evaluation

In this section, we show the effectiveness of our technique in generating partitions of good divergence and its power to find partitions with very high quality, well beyond usual consensus techniques.

4.4.1 Data

We created a synthetic dataset 2D5C, shown in Figure 4.6 with 100 points in 2-dimensions, for which the data is drawn from five Gaussians to produce five visibly separate clusters. We also test our methods on the Iris dataset containing 150 points in 4-dimensions and the

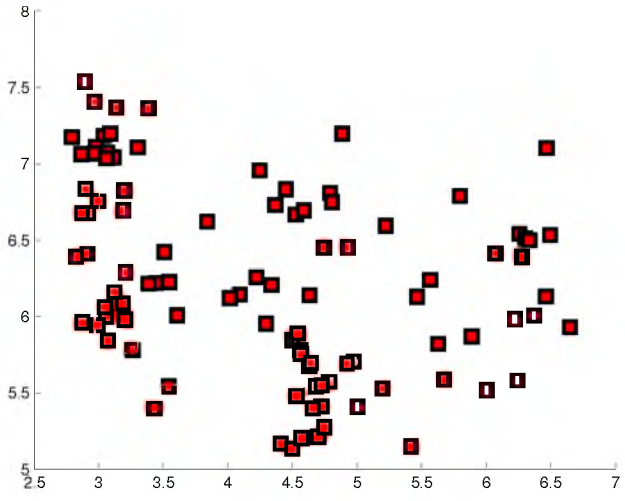


Figure 4.6. 2D5C dataset with 100 points in 2-dimensions.

Adult data containing 48842 points in 14-dimensions, both from the UCI machine learning repository [Frank and Asuncion, 2010]. We also use the extended Yale face database B [Georghiades et al., 2001], which has 38 individuals with approximately 64 frontal poses under different illumination scenarios. The images are resized to 32x32 pixels. Thus, the face dataset contains 2414 points in 1024-dimensions.

4.4.2 Methodology

For each dataset, we first run k-means to get the first partition with the same number of clusters specified by the reference partition. Using this as a seed, we generate $m = 5000$ partitions, after throwing away the first 1000 of them. We report the results of the experiments using the Q_{W-S} quality function as this performed much better compared to the Q_W quality function. We then select a few representative partitions by one of the following two methods.

1. Run average-linkage hierarchical agglomerative clustering on the distance matrix of the partitions generated and pick a partition from each group after making a cut on the dendogram to obtain $k << m$ clusters.
2. Run the Gonzalez k-center method to find $k \ll m$ representative partitions.

We associate each of the $(m-k)$ remaining partitions with the closest representative partition.

Typically, we pick $k = 10$ partitions. We compute and report the quality of each of these representative partitions. We use Q_{W-S} (the quality measure used to generate the partitions) and Rand index as an independent measure to evaluate the partitions generated. We measure the LIFTEMD distance to each of these partitions from the reference partition as well as the average pair wise distance between the partitions generated. We compare these numbers to the representative partitions picked to evaluate the diversity in the context of the given landscape. We also plot the quality of consensus partitions generated by LIFTSSD [Raman et al., 2011] using inputs from k-means, single-linkage, average-linkage, complete-linkage, and Ward's method. Since consensus clustering tries to compute the best quality partition from the inputs, this gives us a good reference point on how clusterable the data is by the individual clustering methods.

4.4.3 Performance Evaluation

4.4.3.1 Evaluating partition diversity. The first aspect that we focus on is how "diverse" the generated partitions are. We can evaluate partition diversity by determining the LIFTEMD and Rand distance between all pairs of representative partitions. We can look at this in the context of the all pairs distances between all the generated partitions. Low LIFTEMD and Rand distance values between representatives will indicate redundancy and that the partition in consideration is similar to other representatives. If two representatives have high LIFTEMD and Rand distance values, it indicates good diversity among partition representatives.

The resulting distribution of distances of (a) the partitions generated and (b) the representative partitions is presented in Figure 4.7. We plot the LIFTEMD distance in Figures 4.7(a), 4.7(c), 4.7(e) and 4.7(g) for Iris, Adult, Yale face B and the 2D5C datasets respectively. We plot the Rand distance in Figures 4.7(b), 4.7(d), 4.7(f) and 4.7(h) for Iris, Adult, Yale face B and the 2D5C datasets, respectively.The blue shaded region corresponds to the area plot of the distribution of the all pairs distances of all the partitions generated, while the orange region plots the all pairs distance distribution for only the ten representative partitions picked. We expect that the representative partitions will be far from each other. Since distance measures suffer from calibration issues, the blue region provides a baseline. For all datasets, a majority of the representative partitions are comparably

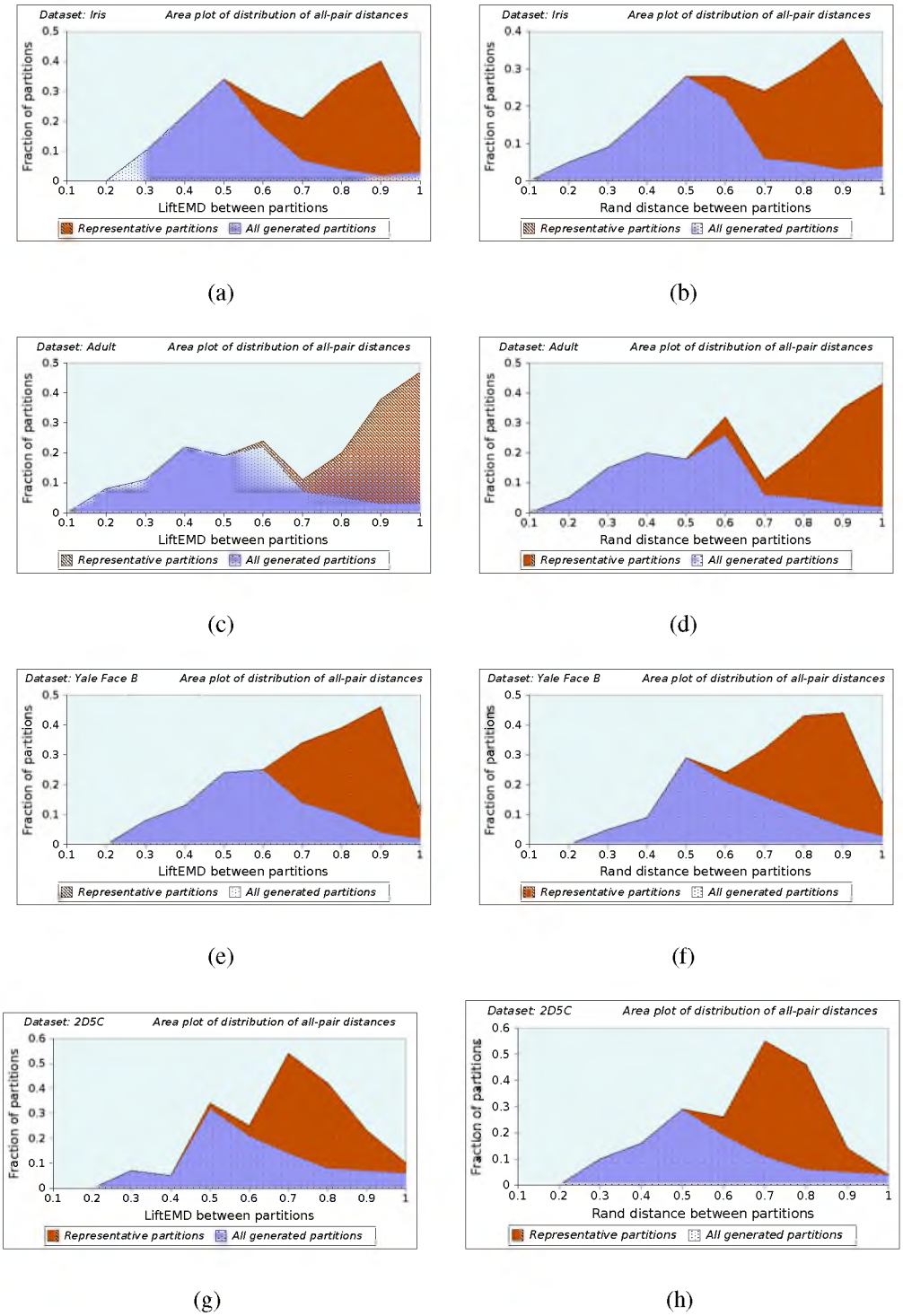


Figure 4.7. Stacked area plot comparing the distributions of all pairs distances of all generated partitions and the all pairs distances of the representative partitions. (a), (c), (e), and (g) show the LIFTEMD distance and (b), (d), (f), and (h) show the Rand distance.

faraway from each other (this is indicated by the orange region lying well to the right of the blue region). This indicates nice clusters of partitions in the landscape that we generated. In Figure 4.8, we plot the heat map of the all pairs LIFTEMD distance matrix to show the separation between the representative partitions. We notice that the heat map is predominantly hot showing that the representative partitions are diverse.

4.4.3.2 Evaluating partition quality. Secondly, we would like to inspect the quality of the partitions generated. Since we intend the generation process to sample from the space of all partitions proportional to the quality, we hope for a majority of the partitions to be of high quality. The ratio between the kernel distance quality Q_{W-S} of a partition to that of the reference partition gives us a fair idea of the relative quality of that partition, with values closer to one indicating partitions of higher quality. We also compute the relative Rand index between each partition and the ground truth as an independent measure. The distribution of quality is plotted in Figure 4.9. We plot the kernel distance quality (Q_{W-S}) in Figures 4.9(a), 4.9(c), 4.9(e) and 4.9(g) for Iris, Adult, Yale face B and the 2D5C datasets, respectively. We plot the Rand index in Figures 4.9(b), 4.9(d), 4.9(f) and 4.9(h) for Iris, Adult, Yale face B and the 2D5C datasets, respectively.

We observe that for all the datasets, we get a normally distributed quality distribution. In addition, we compare the quality of our generated partitions against the partition generated by the consensus technique LIFTSSD. We mark the quality of the representative partitions with red squares and that of the consensus partition with a green circle. For instance, on the Iris dataset (Figure 4.9(a)), we can see that the relative quality with respect to the ground truth partition of over half of the representative partitions is better than that of the consensus partition. This demonstrates the value of our alternative clustering method to explore the landscape since the consensus partition would be the best partition that is available to us without looking at the entire landscape. For the Yale face B data, note that we have two reference partitions, namely "by person" and "by illumination" and we chose the partition "by person" as the reference partition due to its superior quality.

4.4.3.3 Visual inspection of partitions. First, we do a visual inspection of two partitions picked at random from the representative partitions generated using each of Q_W and Q_{W-S} quality functions. In Figure 4.10, we can see that the Q_{W-S} quality function that captures both the compactness and separation of clusters produces visually

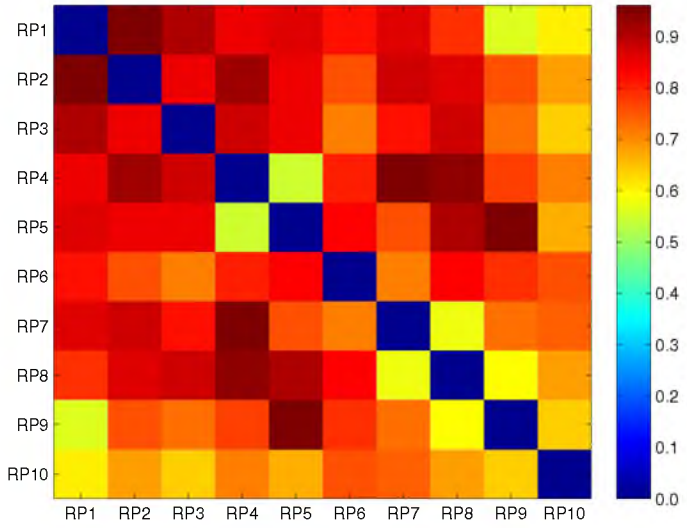


Figure 4.8. Yale face B: heat map of the all pairs LIFTEMD distance matrix of the representative partitions. Red regions correspond to faraway partitions and blue regions are similar partitions.

good clusters, whereas the partition that was generated using the Q_W quality function has overlapping clusters and is thus probably inferior. The average Rand index of the representative partitions generated using Q_{W-S} as the quality function is 0.92, whereas it drops to 0.83 when Q_W is used. We use this to reiterate the importance of choosing the "right" quality measure. Since our sampling procedure is agnostic to the choice of quality measure chosen, the user can take advantage of the prior work on evaluating partitions to pick the right quality measure.

We also visualize the partitions obtained on Yale face B to demonstrate the ability of our method to be used as an exploratory data analysis tool. Figure 4.11 shows two interesting representative partitions. We visualize the mean of the data points of ten clusters that are picked at random from each representative partition. The first representative partition shown in Figure 4.11(a) resembles a partition where each cluster corresponds to the face of a different person and each cluster is of a different illumination level in the second representative partition (Figure 4.11(b)). Both of these partitions are interesting and good quality partitions and are very difficult to generate without looking at the landscape of all interesting partitions.

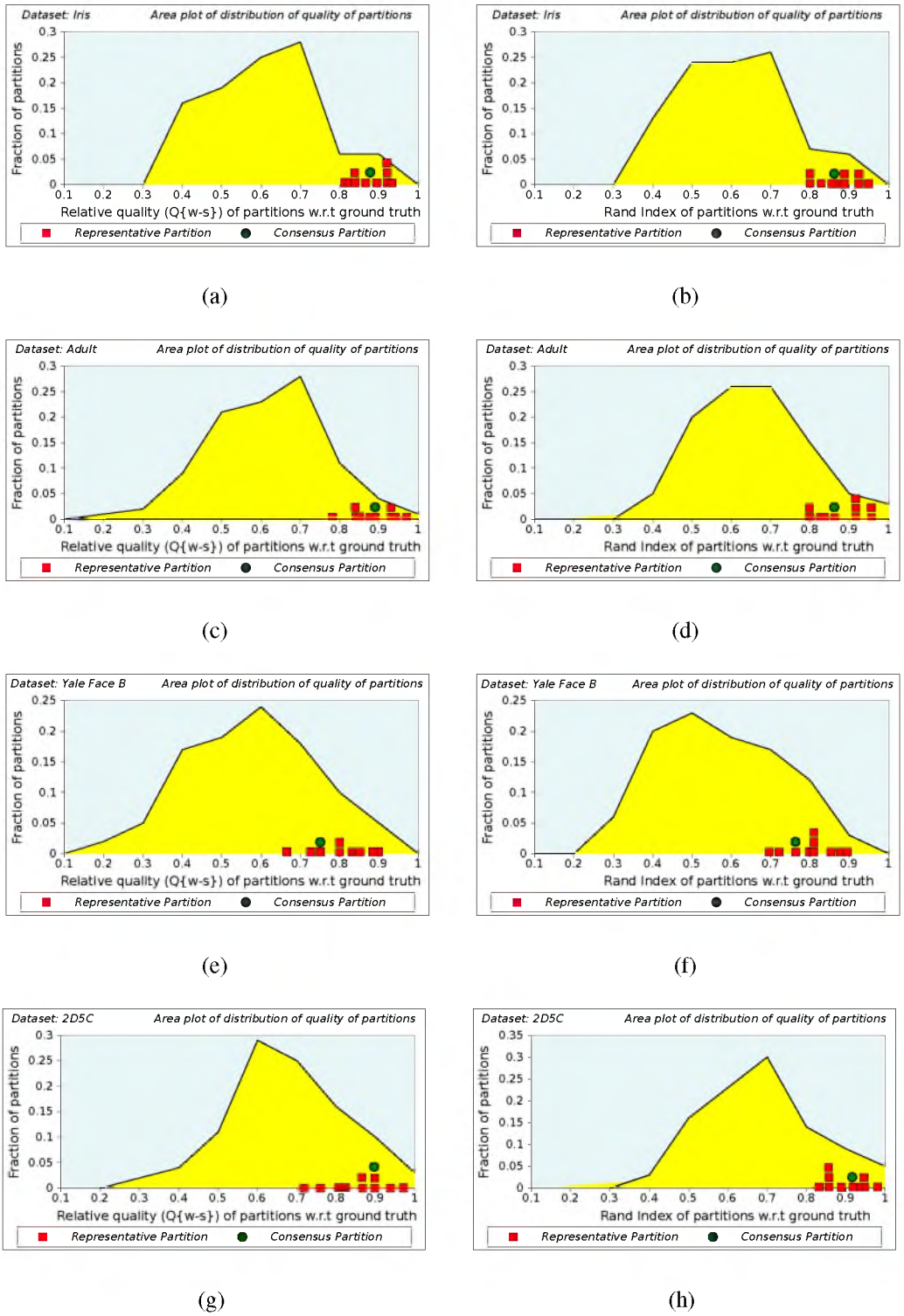
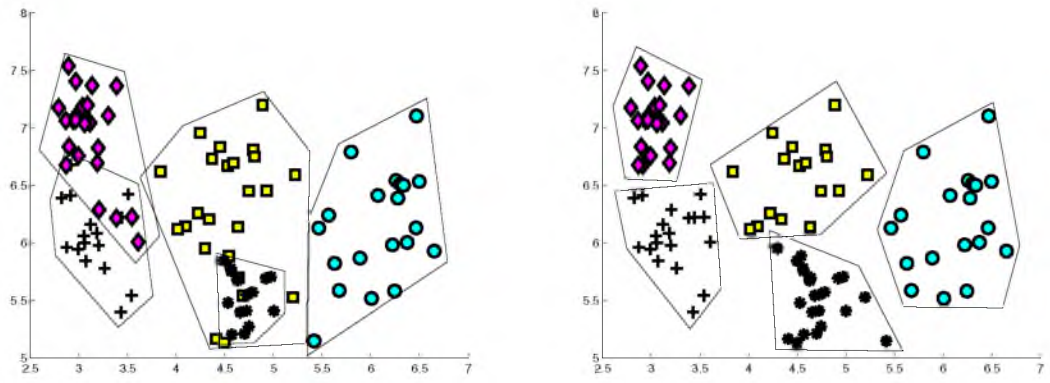
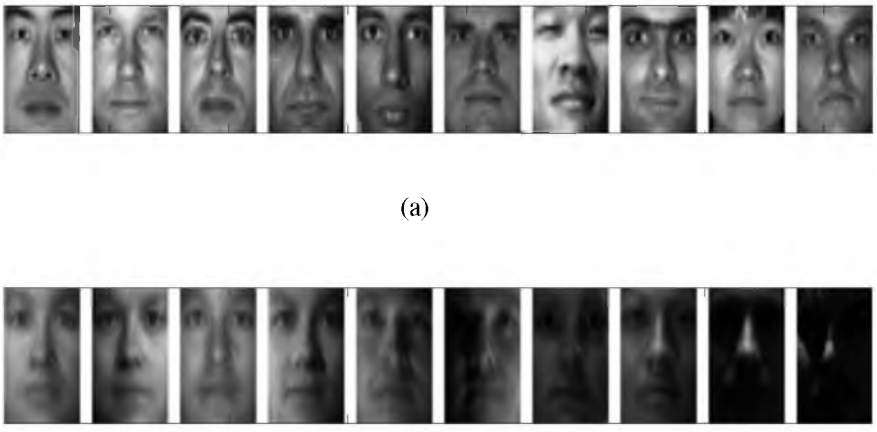


Figure 4.9. Stacked area plot of the distribution of quality of all generated partitions. (a), (c), (e), and (g) show Q_{W-S} and (b), (d), (f), and (h) show the Rand index.



(a) (b)

Figure 4.10. Visual illustration of two representative partitions generated with different quality functions (a) Q_W and (b) Q_{W-S} on 2D5C data.



(a)

(b)

Figure 4.11. Visual illustration of average points of ten clusters in two interesting representative partitions on Yale face B. In (a), each cluster is a different person and in (b), each cluster represents a different illumination level.

4.5 Summary

In this chapter we introduced a new framework to generate multiple nonredundant partitions of good quality. Our approach is a two stage process: in the generation step, we focus on sampling a large number of partitions from the space of all partitions proportional to the quality and in the grouping step, we identify k representative partitions that best summarizes the space of all partitions.

PART II

STABILITY

CHAPTER 5

VALIDATING DATA MEMBERSHIPS[1]

Clustering generates predictions in the form of implicit labels for points. These predictions are used for exploration, data compression, and other forms of downstream data analysis, and so it is important to verify the accuracy of these labels. However, because of the unsupervised nature of clustering, there is no direct way to validate the data assignments. As a consequence, a number of indirect approaches have been developed to validate a clustering at a global level [Halkidi et al., 2001; Xu and Wunsch, 2009]. These include internal, external and relative validation techniques, and methods based on clustering stability that assume a clustering (algorithm) is good if small perturbations in the input do not affect the output partition significantly. There are supervised variants of clustering. However, these typically require domain knowledge, and the immense popularity of clustering comes precisely from the fact that it can be applied as a first filter to acquire a deeper understanding of the data.

All these approaches are global. They assign a single number to a partition and cannot capture the potentially wide variation in label quality within a partition. Consider, for example, a partition of the MNIST digits database (a few example images are displayed in Figure 5.1). By global measures of clusterability, the partition would be considered "good." However, as we can see in the picture in the top row, there are a number of images for which the correct cluster is not as obvious. What we would like in this case is a way to quantify this lack of confidence for each image separately. Such a measure would give a lower confidence rating to the labels for images in the top row, and a downstream analysis task could incorporate this uncertainty into its reasoning. Note that a single number describing the quality of the partition would not suffice in this case, because the downstream analysis

[1] Reprinted with permission of IEEE, 2013, Parasaran Raman and Suresh Venkatasubramanian, Power to the Points: Validating Data Memberships in Clusterings. Thirteenth IEEE International Conference on Data Mining.

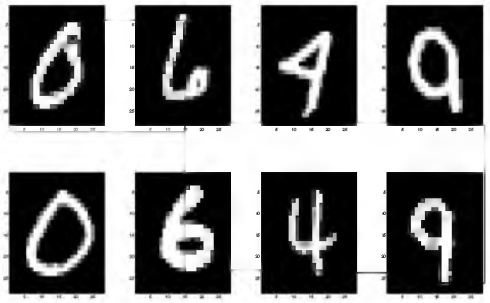


Figure 5.1. MNIST handwritten digits. L-R are numbers {"0", "6", "4", "9"}. The numbers on the top row are very hard to identify even for a human. The bottom row is unambiguous.

might only select a few points (cluster centers, or a representative sample) for further processing.

5.1 Our Work

In this chapter we present a scheme to assign local affinity scores to points that indicate the "strength" of their assignment to a cluster. Our approach has a number of attractive features.

1. It is very general. It takes a partition generated by any method and returns the local affinity scores without relying on probabilistic or other modeling assumptions. It does this by using the ideas of proximity and shared volume: intuitively, a point has strong affinity for a cluster if (when treated as a singleton cluster) its region of influence overlaps significantly with the region of influence of the cluster.
2. It is very efficient to compute. Computing the local affinity of a point depends solely on the number of clusters in the data and an error parameter: there is no dependence on the data size or dimensionality. We show that this can be improved further by progressive refinement, allowing us to avoid computing affinities for points that we are very confident about.
3. It lends itself to easy visualization, which is very useful for diagnostic purposes.
4. The local affinities we compute can also be used to validate the number of clusters in the data as well speeding up clustering computations by focusing attention on points that can affect decision boundaries (as with active learning techniques).

5.1.1 Overview of Our Work

Clustering is about proximity: points are expected to have similar labels if they are close to each other and not to others. In other words, the regions of influence of points belonging to the same cluster must overlap [Houle et al., 2010]. Therefore, a point should be associated with a cluster if its region of influence significantly overlaps the region of influence of the cluster, and does not have such an overlap with other clusters. We can quantify the confidence of this association by measuring the degree of overlap.

The method we propose elaborates on this idea to incorporate a variety of more general notions of regions of influence that can incorporate cluster importance, density and even different cluster shapes. The key idea is to define regions of influence as elements of an appropriate weighted power diagram (a generalization of a Voronoi diagram) and use shared volume to quantify how different regions overlap.

At first glance, this idea is doomed to fail: computing Voronoi regions (and their volumes) is extremely difficult in high dimensions. We show how the volumes of these regions can be estimated (a) without actually computing them and (b) with provable guarantees on the estimates via the use of ε-net sampling and techniques for sampling from convex bodies in high dimensions efficiently. The resulting scheme is accurate and yields the affinity score of a point in time independent of the data size and dimensionality. It runs extremely fast in practice, taking only milliseconds to compute the scores. These scores can also be computed progressively using iterative refinement, so we can focus on the problem cases (points of low affinity) directly.

5.1.2 Applications

The local affinity scores we compute can be viewed as a general diagnostic tool for evaluating partitions and even computing partitions faster. We demonstrate this with a set of key applications.

5.1.2.1 Evaluating the clusterability of data. We have already explained how we expect local affinity scores to certify whether data labels are accurate or not. In addition, combining local affinity scores provides another measure for the global quality of a partition. We will show that this measure matches prior notions [Halkidi et al., 2001; Liu et al., 2010] of global quality of a partition and thus is a more general tool for clustering quality. We will also show that this global measure can be used to solve the vexing problem of

identifying the right number of clusters in a partition [Rousseeuw, 1987; Sugar and James, 2003; Tibshirani et al., 2001], and has certain advantages over other approaches like the common "elbow method" [Tibshirani et al., 2001].

5.1.2.2 Active clustering. Clustering algorithms usually have a nonlinear time dependence on the input size, and so as data sizes grow, the time to cluster grows even faster. This motivates "bootstrapping" strategies where the algorithm first clusters a small sample of the data, and uses this partial clustering to find points that lie on cluster boundaries (and would have greater influence on the resulting partition). The most important step in this "active" approach to clustering [Eriksson et al., 2011; Hofmann and Buhmann, 1998; Settles, 2012] is selecting the points to add to the process. We show that if we use points of low affinity as the active points used to seed the next round of clustering, we can obtain accuracy equal to that obtained from the entire dataset but with orders of magnitude faster running time.

5.2 Preliminaries

Let P be a set of n points in $\mathbb{R}^d$. We assume a distance measure D on $\mathbb{R}^d$, which for now we will take to be the Euclidean distance. A clustering is a partition of P into clusters $C = \{C_1, C_2, \ldots, C_k\}$. We will assume that we can associate a representative c_i with a cluster C_i. For example, the representative could be the cluster centroid, or the median.

A **Voronoi diagram** [De Berg et al., 2008] on a set of sites $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\} \subset \mathbb{R}^d$ is a partition of $\mathbb{R}^d$ into regions $V_1, \ldots V_k$ such that for all points in V_i, the site s_i is the closest neighbor. Formally, $V_i = \{\mathbf{p} \in \mathbb{R}^d \mid D(\mathbf{p}, \mathbf{s}_i) \leq D(\mathbf{p}, \mathbf{s}_j), j \neq i\}$. When D is the Euclidean distance, the boundary between two regions is always a hyperplane, and therefore each cell V_i is a convex polyhedron with at most $(k-1)$ faces.

We will also make use of a generalization of the Voronoi diagram called the **power diagram** [Aurenhammer, 1987]. Suppose that we associate an importance score w_i with each site $\mathbf{s}_i$. Then the power diagram on S (see Figure 5.2) is also a partition of $\mathbb{R}_d$ into k regions V_i, such that $V_i = \{\mathbf{p} \in \mathbb{R}^d \mid D^2(p, \mathbf{s}_i) - w_i \leq D^2(\mathbf{p}, \mathbf{s}_j) - w_j, j \neq i\}$. Power diagrams allow different sites to have different influence, but retain the property that all boundaries between regions are hyperplanes and all regions are polyhedra in Euclidean

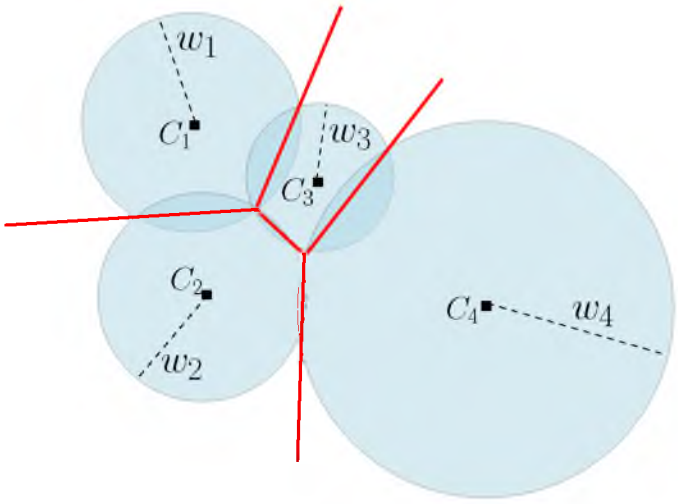


Figure 5.2. The power diagram of a set of points $C_1 \ldots C_4$. The sphere radius is proportional to the weights $w_1 \ldots w_4$.

space[2].

Finally, we will frequently refer to the **volume** $\text{Vol}(S)$ of a region $S \subset \mathbb{R}^d$. In general, this denotes the d-dimensional volume of S with respect to the standard Lebesgue measure on $\mathbb{R}^d$. If S is not full-dimensional, this should be understood as referring to the lower-dimensional volume, or the volume of the relative interior of S; for example, the "volume" of a triangle in 3-dimensions is its area, and the volume of a line segment is its length.

5.3 Defining Affinity Scores

As we discussed in the beginning of this chapter, the **region of influence** of a point is how we define its affinity to clusters. Each cluster has a region of influence. If we now consider a particular point in the data and treat it as a singleton cluster, its region of influence will overlap neighboring clusters. We measure the affinity of a point to a cluster to be the proportion of influence it overlaps from that cluster. We now define these ideas formally.

Let $\mathscr{C} = C_1, C_2, \ldots C_k$ be a partition of n points. A **region of influence function** is a function $R : \mathscr{C} \to 2^{\mathbb{R}^d}$ on $\mathscr{C}$ such that all $R(C_i)$ (which are subsets of $\mathbb{R}^d$) are disjoint.

The simplest region of influence function is a Voronoi cell. Specifically, consider a partition with k clusters, each cluster C_i having representative c_i. Let C be the set of these

[2]The squared distance is crucial to making this happen; without it, arcs could be elliptical or hyperbolic.

representatives. Consider any point $x \in CH(C)$ (the convex hull of C). Let $V_1, V_2, \ldots, V_k$ be the Voronoi partition of C, and let $U_1, U_2, \ldots, U_k, U_x$ be the Voronoi partition of $C \cup \{x\}$, with U_x being the Voronoi cell of x. Then we define the region of influence, $R(C_i) = V_i$, and $R_x(C_i) = U_i$.

Let R be a region of influence function. Let $\mathscr{C} = C_1, C_2, \ldots C_k$ be a partition. For any point x, let $\mathscr{C}_x$ denote the partition $C_1 \setminus \{x\}, C_2 \setminus \{x\}, \ldots, C_k \setminus \{x\}, \{x\}$, and let $R_x(C)$ denote the region of influence of a cluster $C \in \mathscr{C}_x$. Then the affinity score of x is the vector $(\alpha_1, \alpha_2, \ldots, \alpha_k)$, where

$$\alpha_i = \frac{\text{Vol}(R(C_i) \cap R_x(\{x\}))}{\text{Vol}(R_x(\{x\}))}.$$

In the above definition, $R_x(\{x\})$ is the region of influence x has carved out for itself, and α_i merely captures the proportion of $R_x(\{x\})$ that comes from the (original) cluster C_i.

Continuing our example of Voronoi regions of influence, the Voronoi cell U_x of x "steals" volume from Voronoi cells around it (Figure 5.3 illustrates this concept). We can compute the fraction of U_x that comes from any other cell. For any point $p_i \in P$, let $\alpha_i = \frac{\text{Vol}(V_i \cap U_x)}{\text{Vol}(U_x)}$. Then α_i represents the (relative) amount of volume that x "stole" from p_i. Note that $\sum \alpha_i = 1$, and if $x = p_i$, then $\alpha_i = 1$.

The affinity score captures the entire set of interactions of a point with the clusters. It is often convenient to reduce this to a single score value. For example, since at most, one α_i can be strictly greater than 0.5, we can define a point as stable if such an α_i exists, and say that it is assigned to cluster i. In general, we will define the stability of a point to be $\sigma(\mathbf{p}) = \max \alpha_i$. The stability of a point lies between zero and one and a larger value indicates greater stability.

It is important to note that the idea of area stealing was first defined in the context of natural neighbor interpolation (NNI) [Sibson, 1980, 1981], where the α_i values were then used to compute an interpolation of function values at the p_i. Widely used by researchers in the geosciences community, NNI is a powerful technique that has not yet appealed to the data mining community at large. NNI is a spatial interpolation method that works on Voronoi tessellation of a collection of sites. It is especially attractive since it provides a smooth approximation of the function that operates on the Voronoi sites, at any arbitrary point. The NNI function is also continuous everywhere within the convex hull of the data. NNI is a weighted average technique that uses the neighboring Voronoi cells of a selected

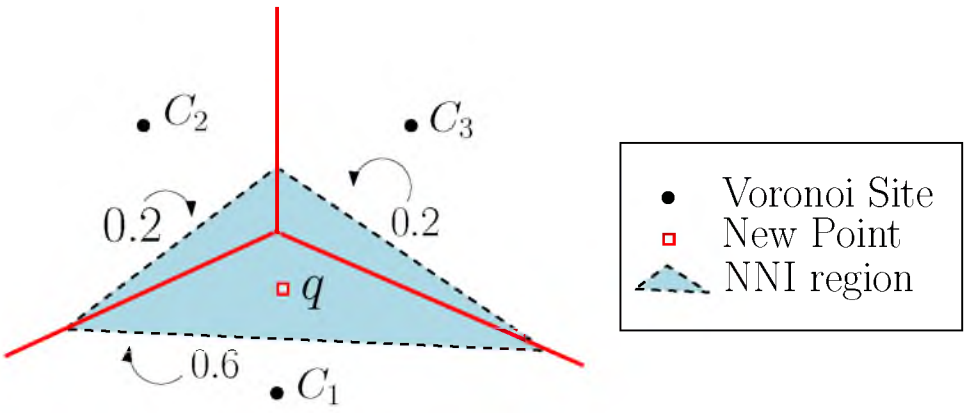


Figure 5.3. In this example, the red point is "stealing" the shaded area from the Voronoi cells of C_1, C_2, C_3.

arbitrary point x. Upon picking the point x for which the NNI is to be calculated, the weight of each neighbor is proportional to the area that the Voronoi cell of x (from the new Voronoi tessellation with all the original sites and x) "steals" from the Voronoi cell of the neighbors in the absence of x. Therefore intuitively, if x is well within a particular Voronoi region, that cell will have a bigger weight. In this chapter we will use the α_i directly without computing any interpolants.

5.3.1 A Rationale for Affinity

The simplest way to define influence is by distance. For example, we could define the affinity of a point to a cluster as the (normalized) distance between the point and the cluster representative. Our definition of affinity generalizes distance ratios: in 1-dimension, affinity calculations yield the same result as distance ratios, since the "area" stolen from a cell is merely half the distance to that cell. But, affinity can capture stronger spatial effects, as our next example shows.

Consider the configuration shown in Figure 5.4. The point $\mathbf{q}_1$ is equidistant from the cluster centers c_2 and c_3 and so would have the same distance-based influence with respect to these clusters. But, when we examine the configuration more closely, we see that the presence of c_4 is reducing the influence of c_3 on $\mathbf{q}_1$, and this effect appears only when we look at a planar region of influence. We validate by using 100 runs of k-means with random seeds. We observe that $\mathbf{q}_1$ was assigned to c_2 in 15 runs and to c_3 in only two runs. A distance-based affinity would have suggested an equal "affinity" for the two clusters,

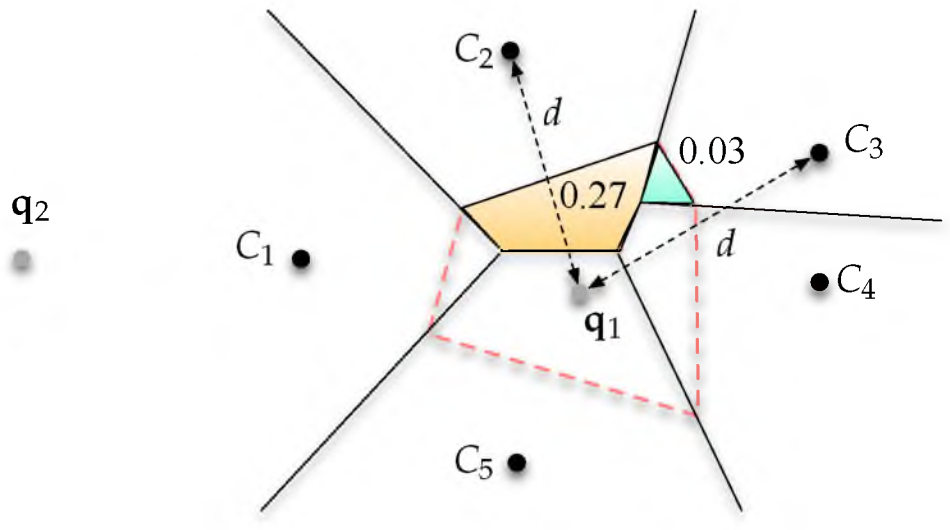


Figure 5.4. Illustration of the difference between distance-based and area-based influence measures.

whereas a volume-based affinity incorporates the effects of other clusters.

Similarly, consider $\mathbf{q}_2$. It is twice as close to c_1 compared to c_2 or c_5, which would result in the distance-based influence of c_1 being equal to the influence of c_2 and c_5 combined. When we validate this using k-means, we find that $\mathbf{q}_2$ is exclusively assigned to cluster center c_1. Here, C_1 has a "shielding" effect on q_2 that prevents it from ever being assigned to those clusters: this shielding can only be detected with a truly spatial affinity measure.

5.3.2 Visualization

The affinity scores define a vector field over the space the data is drawn from. The stability $\sigma(\mathbf{p})$ defines a scalar field and can be visualized (in low-dimensions). Consider the partition depicted in Figure 5.5(a). We can draw a contour map (see Figure 5.5(b)) where each level connects points with the same stability score (unlike in a topographical map, more deeply nested contours correspond to lower stability scores). We can also render this as a greyscale heat map (see Figure 5.5(c), where the lower the affinity, the brighter the color). These visualizations, while simple, provide a visual rendering of affinity scores that is useful as part of an exploratory analysis pipeline.

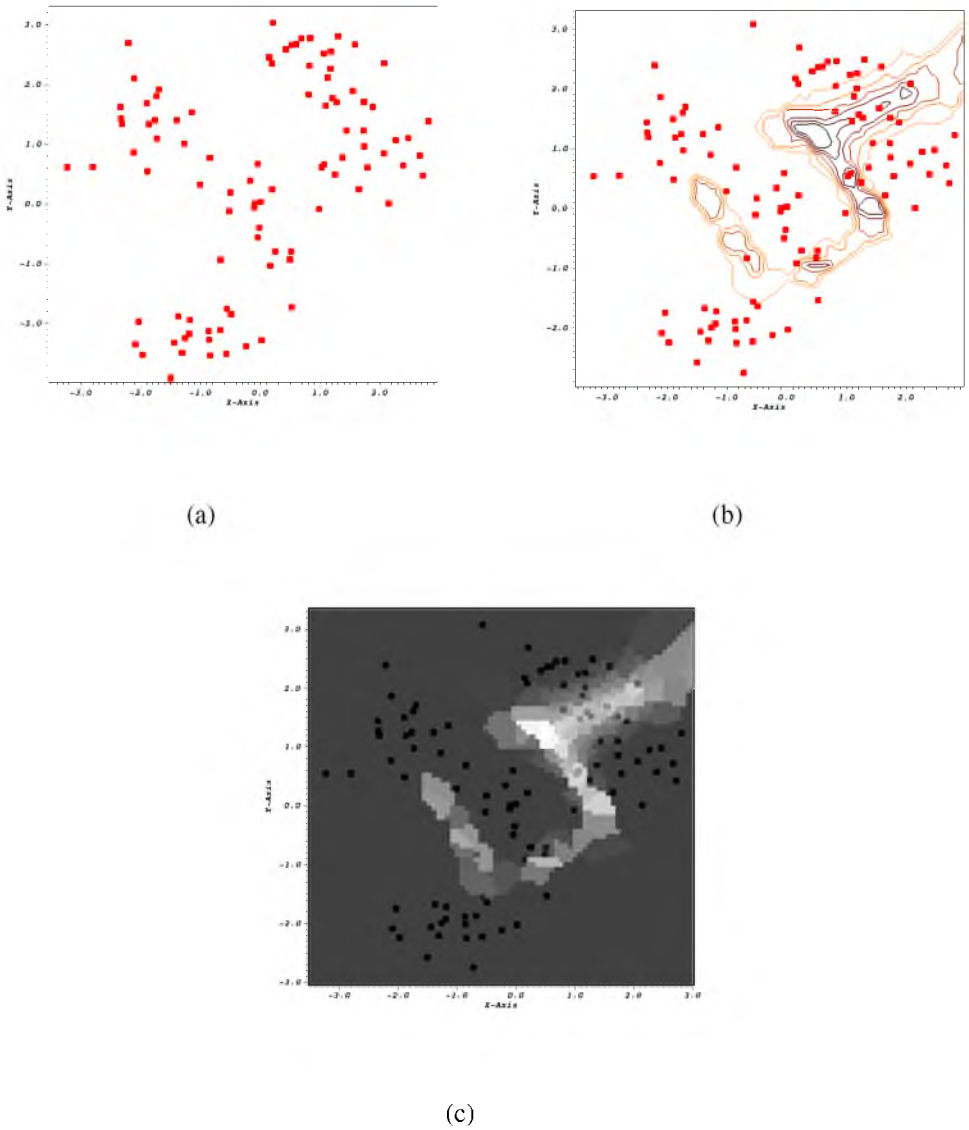


Figure 5.5. Visualizing the affinity scores. We plot (a) the data with 5 clusters, (b) the contour plot, and (c) the heat map.

5.3.3 Extensions

Our definition of affinity is not limited to Euclidean spaces. It can be generalized to a variety of spaces merely by modifying the way in which we construct the Voronoi diagrams. In all cases, the resulting affinity scores will result from a volume computation over polyhedra.

5.3.3.1 Giving clusters varying importance: density-based methods. Consider a generalized clustering instance where each cluster C_i has an associated weight w_i, with a

larger w_i indicating greater importance. Instead of constructing the Voronoi diagram, we will construct the power diagram defined in Section 5.2. Specifically, the region of influence R_i for cluster C_i will be defined as the set $R(C_i) = \{x | d^2(p_i, x) - w_i \leq d^2(p_j, x) - w_j\}$. We compute the affinity vector as before, with the weight of a singleton $\mathbf{x}$ set appropriately depending on the weight function used. For example, if $w(C_i) = |C_i|/n$, then $w(\mathbf{x}) = 1/n$.

Consider the examples depicted in Figure 5.6. The left hand figure has 100 points in each of the five clusters, and the right hand figure has 500 points in each of the four outer clusters and 100 points in the center cluster. Notice that there is a lot more instability (as seen by the contours) in the sparser example, much of which is due to the presence of the central cluster. However, once the density of the outer clusters increases, the effect of the inner cluster is much weaker, and there are fewer unstable regions.

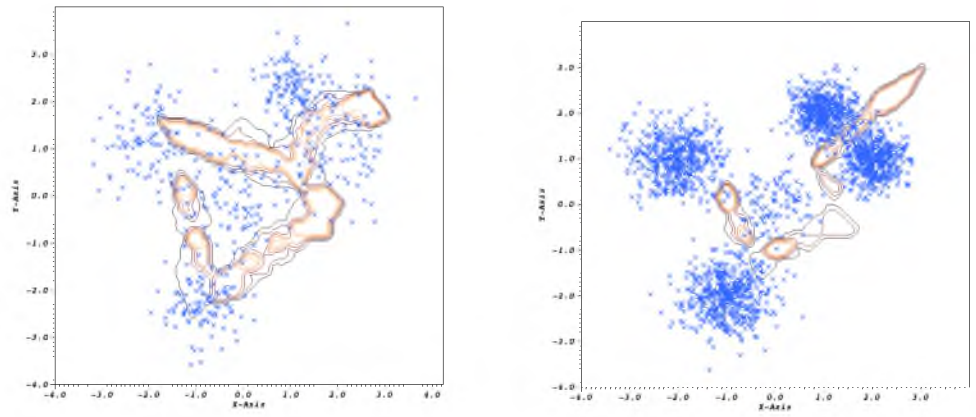
We can also extend our Voronoi-based definition of affinity to partitions in Bregman spaces [Bregman, 1967] and kernel spaces [Schölkopf and Smola, 2002]. In each case, the resulting affinity score reduces to volume computation on polyhedra, just as in the Euclidean space.

5.4 Estimating Affinity

The many different ways of defining affinity scores via regions of influence all reduce to the following: given a set of representatives $C = \{c_1, \ldots, c_k\}$ and a query point x, estimate the volume of a single cell in the Voronoi diagram of C or $C \cup \{x\}$, and estimate the volume of the intersection of two such cells.

In 2-dimensions, the Voronoi (or weighted Voronoi) diagram of k points can be computed in time $O(k \log k)$ [De Berg et al., 2008], and the intersection of two convex polygons can be computed in $O(k)$ time [Toussaint, 1985]. Any polygon with k vertices can be triangulated in $O(k)$ time using $O(k)$ triangles, and then the area can be computed exactly in $O(k)$ time ($O(1)$ time per triangle). In 3-dimensions, computing the Voronoi diagram takes $O(k^2)$ time, and computing the intersection of two convex polyhedra can be done in linear time [Chazelle, 1992]. Tetrahedralizing the convex polyhedron can also be done in linear time [Lennes, 1911].

This direct approach to volume computation does not scale. In general, a single cell in the Voronoi diagram of k points in $\mathbb{R}^d$ can have complexity $O(k^{\lceil d/2 \rceil})$. We now propose



(a) (b)

Figure 5.6. Visualizing the affinity scores for datasets with different densities. There are 100 points in each cluster in (a) and 500 points in the clusters on the boundary in (b).

an alternate strategy that provably approximates the affinity scores to any desired degree of accuracy in polynomial time using random sampling.

Let $U_{\mathbf{x}}$ be the Voronoi cell of $\mathbf{x}$ in the Voronoi diagram of $C \cup \{\mathbf{x}\}$. We say that the point $\mathbf{y}$ is stolen from $s(\mathbf{y}) \triangleq c_i$ if (i) $\mathbf{y} \in U_{\mathbf{x}}$ and (ii) $\mathbf{y}$'s second nearest neighbor is $\mathbf{c}_i$. We can then write $\alpha_i = \text{Vol}(\{\mathbf{x} \mid s(\mathbf{x}) = \mathbf{c}_i\})/\text{Vol}(U_{\mathbf{x}})$. Note that given a point $\mathbf{x}$ and any point $\mathbf{y}$, we can verify in $O(k)$ time whether $\mathbf{y} \in U_{\mathbf{x}}$ and also compute $s(\mathbf{y})$ by direct calculation of the appropriate distance measure.

Let $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ be the affinity scores for x. Suppose we now sample a point $\mathbf{y}$ uniformly at random from $U_{\mathbf{x}}$. We can find $s(\mathbf{y})$ in $O(k)$ time and this provides one update to α_i. The number of such samples needed to get an accurate estimate of each α_i is given by the theory of ε-samples. Let μ be a measure defined over X and let $\mathscr{R}$ be a collection of subsets of X. An **ε-sample** with respect to $(X, \mathscr{R})$ and μ is a subset $S \subset X$ such that for any subset $R \in \mathscr{R}$,

$$\left|\frac{\mu(S \cap R)}{\mu(S)} - \frac{\mu(R)}{\mu(X)}\right| \leq \varepsilon.$$

By standard results in VC-dimension theory [Har-Peled, 2011; Li et al., 2001; Talagrand, 1994], a random subset of size $O(\frac{1}{\varepsilon^2}(\eta + \log\frac{1}{\delta})$ is an ε-sample for a range space $(X, \mathscr{R})$ of VC-dimension η, with probability at least 1-δ.

If we now consider the discrete space $[1 \ldots k]$ with the measure $\mu(i) = \alpha_i$, then the set of ranges $\mathscr{R}$ is the set of singleton queries $\{1 \ldots k\}$, and the VC-dimension of $([1 \ldots k], \mathscr{R})$ is a constant. This means that if we sample a set S of $O(\frac{d}{\varepsilon^2} \log \frac{1}{\varepsilon})$ points from $U_{\mathbf{x}}$, and set $\tilde{\alpha}_i = |x \in S \mid s(x) = i|/|S|$, then $|\tilde{\alpha}_i - \alpha_i| \leq \varepsilon$ for all i.

5.4.1 Sampling from $U_{\mathbf{x}}$

We now have a strategy to estimate the affinity scores of $\mathbf{x}$. Sample the number of points from $U_{\mathbf{x}}$ as prescribed above and then estimate $\tilde{\alpha}_i$ by computing the owners of samples. Standard rejection sampling (sample from a ball enclosing $U_{\mathbf{x}}$ and reject points outside it) does not work in high dimensions as the number of rejected points grows exponentially with the dimension. For example, in 20-dimensions, over 1000 points are rejected for each good sample in experiments.

To solve this problem, we make use of the extensive literature on sampling from a convex polyhedron in time polynomial in d, following the groundbreaking randomized polynomial time algorithm of Dyer, Frieze and Kannan. At a high level, these are all Markov chain Monte Carlo (MCMC) methods: they use different random walks to extract a single uniform sample from the polyhedron efficiently. We describe the sampling procedure in Algorithm 2. One of the most effective strategies in practice for doing this is known as hit and run [Smith, 1984]. It works as follows. Starting with some point $\mathbf{x}$ in the desired polytope K, we pick a direction at random, and then pick a point uniformly on the line segment emanating from $\mathbf{x}$ in that direction and ending in the boundary of K. We refer to this step as hit and run. It has been shown [Lovász, 1999] that this random walk mixes very well, making $O(d^3)$ calls to a membership oracle to produce a single sample (under some technical assumptions). Figure 5.7 illustrates the distribution of samples using hit and run for the Voronoi cell of the point q. Algorithm 3 (AFFINITY) summarizes the process for computing the affinity score of a single point.

5.4.1.1 Reducing dimensionality. The above sampling procedure runs in time $O(d^3)$ per point. However, d can be quite large. We make one final observation that replaces terms involving d by terms involving $k \ll d$ for Euclidean distance measures (or Euclidean distances derived from a kernel).

The Voronoi diagram of k points in d-dimensions, where $k < d$, has a special structure. The k points together define a $(k-1)$-dimensional subspace $\mathscr{H}$ of $\mathbb{R}^d$. This means that

Algorithm 2 SamplePolytope

Input: Collection of halfplanes $\mathscr{H}$ defining convex region $K = \cap_{h \in \mathscr{H}} h$, number of samples m. **Output:** m points uniformly sampled from K.

Construct affine transform T such that TK is centered and isotropic.
Fix burn in parameter b
Run hit and run for d steps on TK, ending in $z = z_0$
for $i = 1 \ldots m$ **do**
  Set z_i to be result of one hit and run move from z_{i-1}
end for
Return $(T^{-1}z_1, \ldots, T^{-1}z_m)$.

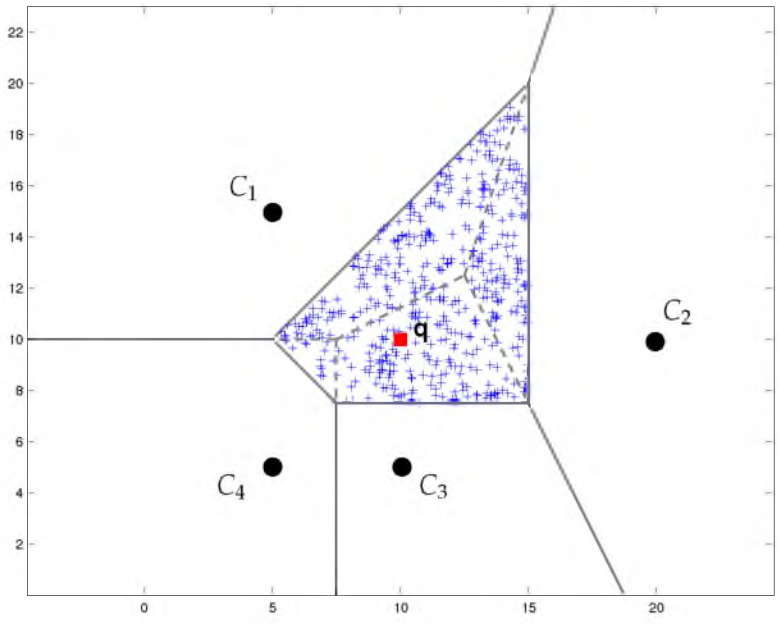


Figure 5.7. Illustration of hit and run for sampling from a Voronoi cell. Samples are shown in blue.

any vector $p \in \mathbb{R}^d$ can be written as $p = u + w$ where $u \in \mathscr{H}$ and $w \perp u$. The Euclidean distance $\|p - p'\|^2$ can be written as $\|u - u'\|^2 + \|w - w'\|^2$. In particular, this means that in any subspace of the form $\mathscr{H} + w$ for a fixed $w \perp \mathscr{H}$, the distance between two points is merely their distance in $\mathscr{H}$.

Therefore, each Voronoi cell V can be written as $V' + \mathscr{H}^{\perp}$, where $V' \subset \mathscr{H}$ and $\mathscr{H}^{\perp}$ is the orthogonal complement of $\mathscr{H}$ consisting of all vectors orthogonal to $\mathscr{H}$. Thus, we can project all points onto $\mathscr{H}$ while retaining the same volume ratios as in the original space. This effectively reduces the problem to a k-dimensional space. The actual projection is performed by doing a singular value decomposition on the $k \times d$ matrix of the cluster

Algorithm 3 AFFINITY: computing the affinity score for a point

Input: A partition $\mathscr{C} = C_1, C_2, \ldots, C_k$ with representatives $c_1, \ldots, c_k$ and a point x.
Output: Affinity vector $(\alpha_1, \ldots, \alpha_k)$ for x

$m \leftarrow \frac{c}{\varepsilon^2} \log \frac{1}{\varepsilon}$
Set all $\alpha_i \leftarrow 0$
for $j = 1 \ldots k$ **do**
  Set $\mathscr{H}_j$ as the halfplane supporting U_x with respect to c_j in the Voronoi diagram.
end for
Call **SamplePolytope**$(\{\mathscr{H}_1, \ldots, \mathscr{H}_k\}, m)$ to generate m samples
$z_1, z_2, \ldots z_m \in U_x = \cap \mathscr{H}_j$.
for $i = 1 \ldots m$ **do**
  Compute $s = \arg\min_{j=1 \ldots k} d(z_i, c_j)$.
  $\alpha_s = \alpha_s + 1/m$
end for
Return $(\alpha_1, \ldots, \alpha_k)$.

representatives. Once this transformation is done, we call AFFINITY as before. The resulting algorithm computes the affinity scores for a point in time $O(k^3 \log(1/\varepsilon)/\varepsilon^2)$.

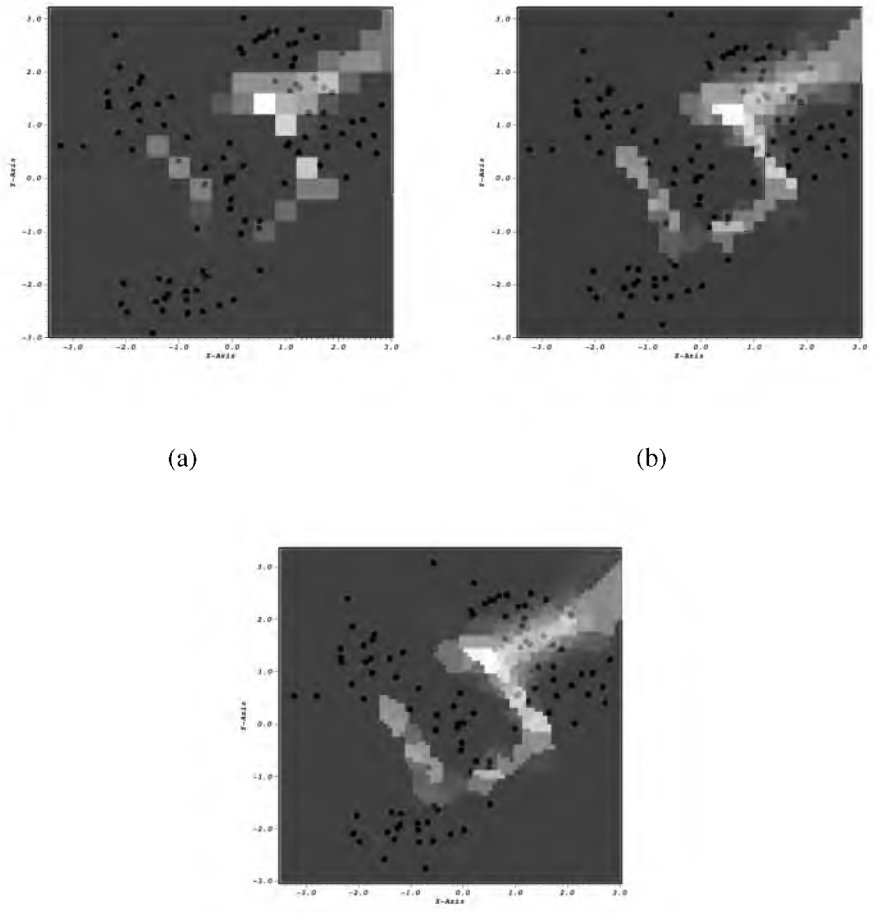
5.4.1.2 Progressive refinement of affinity scores. In many applications, we care only about points with low stability since they define decision boundaries. But, most points are likely to have high stability scores, and computing the scores of all points is wasteful. We describe a progressive refinement strategy that "zooms in" on the unstable points quickly. We begin with a very coarse grid on the data. For each cell, we first compute the stability score of points at the corners of the cell. If the corners are highly stable, we skip this cell, else we subdivide it further and repeat. We seed the process with a grid that has n cells (and therefore is subdivided into $n^{1/d}$ segments in each dimension.

We show the effect of this progressive refinement method for 2-dimensional data in Figure 5.8. The heat map on the left only contains $\sqrt{n}$ cells and the one in the middle contains 10 $\sqrt{n}$ cells. Note that the middle heat map is very similar to the heat map on the right that uses no refinement strategies at all, and uses far fewer stability evaluations.

5.5 Experiments

We demonstrate the following benefits of affinity scores in this section.

1. Affinity scores identify points on the true cluster boundary, which is useful in determining how a particular point affects the clustering of data.



(a) (b)

(c)

Figure 5.8. Reducing computation through progressive refinement: (a) very coarse gridding, (b) moderate gridding, and (c) gridding with all points.

2. Affinity scores can be used to speed up clustering by actively selecting points that matter.
3. Aggregated stability scores help with determining clusterability and model selection.
4. Our method is practical and scales well with dimensionality and data size.

5.5.1 Data and Experimental Setup

In $d = 2$ and $d = 3$ dimensions, affinity scores can be calculated via direct volume computations. We use built in routines provided by the computational geometry and algorithms library (CGAL) (http://www.cgal.org) to compute the scores exactly and validate

our sampling-based algorithm. For higher dimensional data, we perform the initial data transformation (if needed) in `C` and use a native routine for hit and run in MATLAB. All experiments are run on a Intel Quad Core CPU 2.66GHz machine with 4GB RAM. Reported times represent the results of averaging over ten runs.

We created a synthetic dataset in $\mathbb{R}^2$, namely 2D5C for which data is drawn from five Gaussians to produce five visibly separate clusters with 100 points each. We also use a variety of datasets from the UCI repository. See Table 5.1 for details.

5.5.2 Using Affinity Scores to Identify Poorly Clustered Points

We start by evaluating how well affinity scores in general (and stability specifically) pick out points that are "well assigned" or "poorly assigned." The MNIST digits dataset is a good test case because it contains ground truth (the actual labeling) and we can visually inspect the results to see how the method performed.

We run a k-means algorithm on the MNIST test data and compute affinity scores of the points. We sort each digit cluster by the stability score and then pick one element at random from the top ten and one from the bottom ten. Figure 5.9 shows the results for four digits The first row shows points that had high stability in the clustering (close to one in each case). We can see that the digits are unambiguous. The second row shows digits from the unstable region (the top affinity scores are 0.38, 0.46, 0.34 and 0.42, respectively). Notice that in this case the digits are far more blurred. In fact, the "4" and "9" look similar, as do the "0" and "6". The second highest affinity scores for the ones in the bottom row are 0.21, 0.19, 0.24 and 0.28 and they correspond to clusters {"4", "0", "9" and "7"}.

We also validate the affinity scores against the results produced by probabilistic modeling. We run an expectation maximization (EM) algorithm to estimate the data parameters for a Gaussian mixture model and use the final cluster centers obtained to run our volume stealing stability method. To get a holistic view of the label affinities, we compute the entropy of the affinity score for each point (note that the affinity scores sum to one for each point), and we also compute the entropy of the conditional probabilities obtained from the EM algorithm for each point. We now have two vectors of entropies, and we measure their correlation using Pearson's linear correlation coefficient. For 2D5C, Soybean and the Iris datasets, we obtain a correlation of 0.922, 0.893 and 0.935, respectively.

This further shows that affinity scores capture the strength of assignment of a point to

Table 5.1. Datasets.

Dataset	#Points	#Dimensions	#Clusters
Soybean	47	35	4
Iris	150	4	3
Wine	178	13	3
MNIST (Training)	10000	784	10
Protein	17766	357	3
Adult	32561	123	2
MNIST (Test)	60000	784	10
CodRNA	488565	8	2
Covtype	581012	54	7

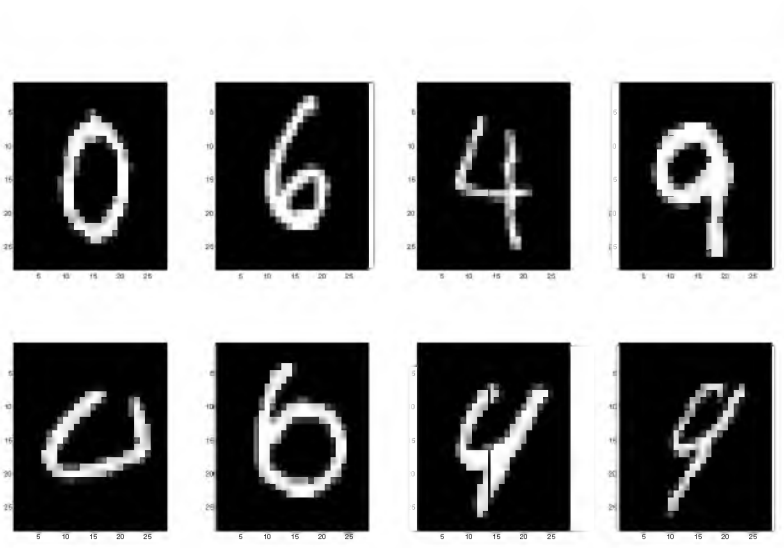


Figure 5.9. Results of running k-means on MNIST training data. First row: high affinity. (L-R) 0.96, 1.0, 1.0, 0.92. Second row: low affinity: (L-R) 0.38, 0.46, 0.34, 0.42.

a cluster. We reiterate that our approach merely requires the user to present a partition obtained by any algorithm.

5.5.3 Using Affinity Scores to Accelerate Clustering

Most clustering algorithms take time that is nonlinear in the number of points. Intuitively, points at the core of a cluster are less useful in determining the cluster boundaries, but there are more of them. Ideally, we would like to subsample points in the core, and supersample points on the boundary to get a subset of points that can effectively recover the true partition. Since many clustering algorithms run in time quadratic in the number of

points, a good heuristic to obtain a fast algorithm is to try and sample $O(\sqrt{n})$ such "good points."

We will use stability scores to identify these points in a two stage iterative approach. Firstly, we run a k-means++ [Arthur and Vassilvitskii, 2007] seeding step to initialize k cluster centers. We then compute stability scores for all points and set the stability threshold at $\sigma(\mathbf{x}) = 0.5$. We fix a fraction $0 < \alpha < 1$ (set by cross validation) and then select a sample of points of size $5\alpha\sqrt{n}$ from the pool of stable points, selecting the remaining $5(1-\alpha)\sqrt{n}$ points at random from the unstable pool. In order to remove anomalies arising from any specific clustering method, we then run a spatially aware consensus procedure [Raman et al., 2011] on this small set using k-means, hierarchical agglomerative clustering (single-linkage, average-linkage and complete-linkage variants) and density-based spatial clustering of applications with noise (DBSCAN) [Ester et al., 1996] as the seed partitions. We then assign all remaining points to their nearest cluster center. We compare this to running the same consensus procedure with all the points.

Table 5.2 summarizes the datasets used, and the sample sizes we used in each case. Figure 5.10 summarizes the results. In each case, the speedup over a full clustering approach is tremendous – typically a 25x speedup. Moreover, the accuracy remains unimpaired: above each bar is the Rand index comparing the partition produced (active or full) to ground truth. In all datasets, the numbers are essentially the same, showing that our method produces as good a partition as one that uses all the data.

As a baseline to evaluate our method, we also compared our approach with a random baseline, where we merely picked a random sample of the same size. We also measured the Rand index of the resulting partitions, and the corresponding numbers were 0.49 for CovType, 0.55 for CodRNA, 0.81 for MNIST, and 0.48 for Protein. In all cases, our method improved over the random baseline, thus demonstrating its effectiveness at finding good partitions.

5.5.4 Using Affinity Scores for Model Selection and Clusterability

While affinity scores are local, we can compute an aggregate score for a partition by averaging the stability scores for each point. We now show that this aggregated score acts as a measure of clusterability and has useful properties that make it more effective in model selection.

Table 5.2. Data setup for active clustering.

Dataset	Points	Samples	# Stable	# Unstable
Protein	17766	665	499	166
MNIST (all)	70000	1323	992	331
CodRNA	488565	3495	2621	874
Covtype	581012	3810	2858	952

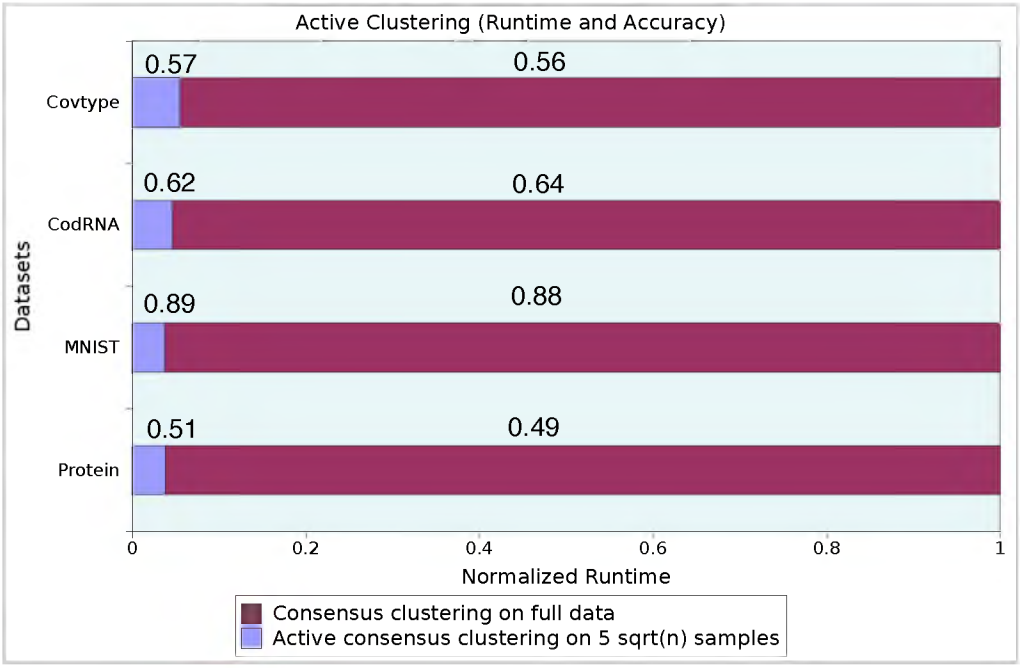


Figure 5.10. Performance of active sampling for consensus clustering. Rand index is displayed above the bar for each method and each dataset.

5.5.4.1 Choosing *k*. Determining the correct number of clusters for a given data is a difficult problem in clustering, especially in an unsupervised setting. The standard approach is to use some variant of the "elbow method" to analyze the trade off curve between number of clusters and clustering cost. Since splitting a cluster typically improves the clustering cost, these methods attempt to find locations where the gradient changes dramatically, or where a point of "diminishing returns" is reached in further splitting.

Aggregate stability is more sensitive to splits of "good clusters." When we split a good cluster we actually decrease the average stability of the partition, because all points along

the boundary of the new cluster used to be very stable and now will no longer be so. We demonstrate this behavior by plotting the cluster cost and average stability score for a variety of datasets from Table 5.1. To demostrate model selection capabilities of global and local methods, we plot the k-means algorithm cost in Figure 5.11(a) and the average stability in Figure 5.11(b). We see that for each dataset, the maximum stability is achieved at precisely the number of clusters prescribed by ground truth. In contrast, the k-means cost function strictly decreases, and it is more difficult to identify clear "elbows" at the right number of clusters.

We also compare aggregate stability to standard measures of global stability like the silhouette method, the Rand index, and the Davies-Bouldin index [Petrovic, 2006]. As we can see in Figure 5.12, all measures behave consistently on the datasets (note that the Davies-Bouldin index is smaller when the partition is better). This shows that aggregate stability acts like a global quality measure while still retaining local structure.

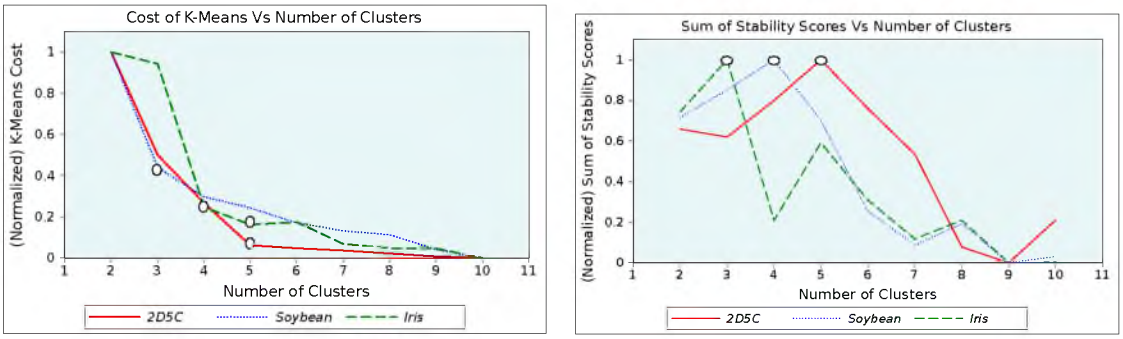
5.5.4.2 Data clusterability. Another use for aggregate stability is as measure of clusterability. We illustrate this by computing the aggregate stability for a clustering of five Gaussians with varying (but isotropic) covariance for each cluster. As we can see, the data becomes progressively less clustered as the variance increases, and therefore becomes less "clusterable."

Figure 5.13 illustrates the aggregate stability scores for these partitions: as we can see, the scores drop similarly, and by the time we reach the fifth instance (which is essentially unclusterable), the stability numbers have dropped to nearly zero. We also annotate the graphs with the number of unstable points (with threshold $\sigma(\mathbf{x}) = 0.5$) to illustrate that the average stability is reducing consistently.

As another illustration of this, we plot in Figure 5.14 the aggregate stability of two different pairs of numbers in the MNIST dataset ("2" vs "6") and ("4" vs "9"). As we have seen earlier, the ("2" vs "6") set is easier to distinguish than the ("4" vs "9") set, and this is reflected in the different stability scores for the clustering on these two pairs. We show what the different partitions look like in Figure 5.15.

5.5.5 Evaluating Performance

Finally, we present an evaluation of the performance of our method in terms of accuracy and running time. To validate the quality of the results, we can compare our sampling-based



(a) (b)

Figure 5.11. Choosing *k*: (a) global using k-means cost vs (b) local using average stability cost.

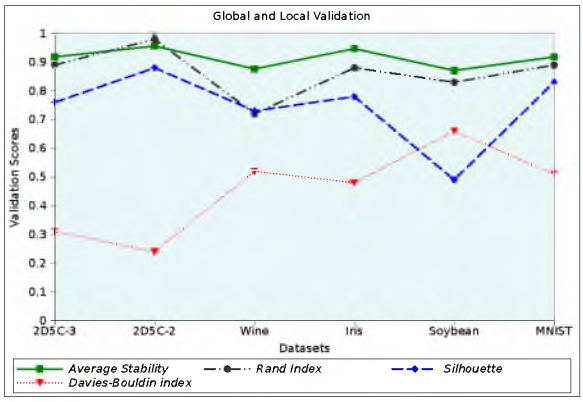


Figure 5.12. Aggregate stability vs global stability.

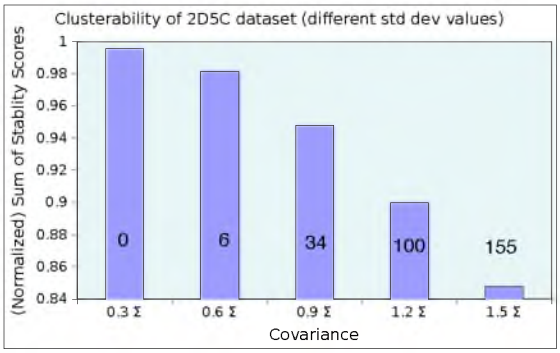


Figure 5.13. Clusterability of 2D5C data: average stability scores dip as variance increases.

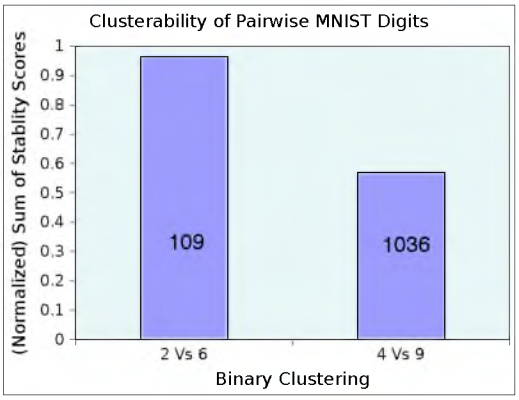


Figure 5.14. Clusterability of two different pairs of digits in the MNIST data.

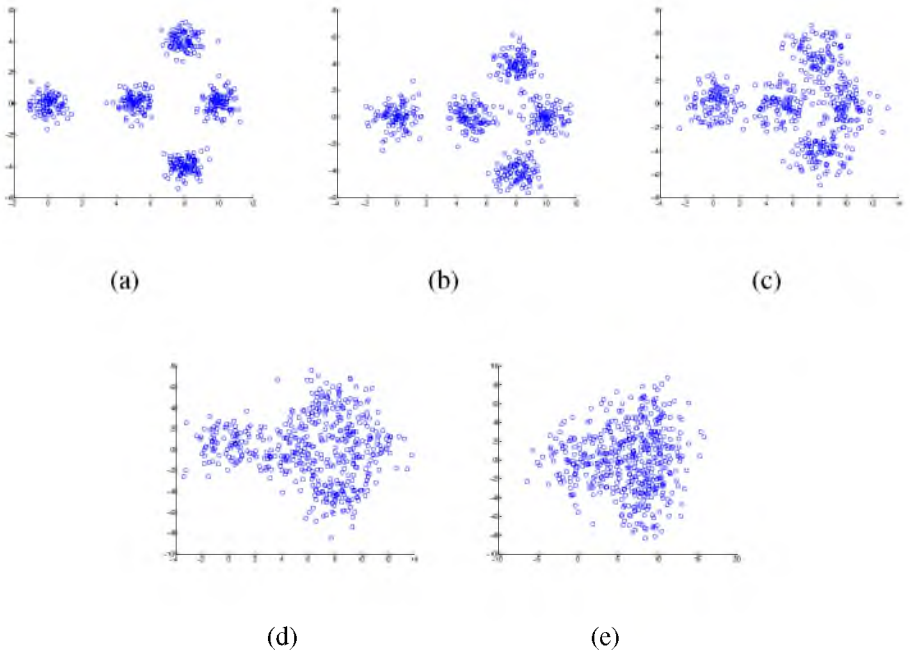


Figure 5.15. Five Gaussians with varying variance: (a) very low, (b) low, (c) moderate, (d) high, and (e) very high.

method to the exact scores we can obtain in $d = 2$ and $d = 3$ dimensions as described earlier. Table 5.3 illustrates this for the 2D5C and 3D5C datasets. We note that these error reports come from choosing 1000 samples after a burn in of 1000 samples (this corresponds to an error $\varepsilon = 0.04$). As we can see, the reported error is well within the predicted range.

Table 5.3 also presents running times for the affinity score computation. We note that

Table 5.3. Runtimes and empirical approximation to exact affinity.

Dataset	n	d	k	Time (sec)	Error
2D5C	500	2	5	0.11 ± 0.005	± 0.02
3D5C	500	3	5	0.19 ± 0.008	± 0.035
IRIS	150	4	3	0.24 ± 0.012	-
Soybean	47	35	4	0.31 ± 0.08	-
MNIST (test)	10000	784	10	0.58 ± 0.5	-

the running times reported are the total for computing the affinity scores for all points. We only report the time taken by the sampler; the preprocessing affine transformation is dominated by the sampling time. In all cases, we used 1000 samples to generate the estimates. Note that the procedure is extremely fast, even for the very high dimensional MNIST data.

5.6 Summary

We view this work as part of a larger effort to personalize validation mechanisms in data mining. In future work we plan on incorporating ideas from topological data mining to add more dimensions to the validation. We hope to develop better visualizations to accompany this method. More generally, we plan on studying other unsupervised learning tasks where local validation is important.

CHAPTER 6

LARGE SCALE TRANSDUCTIVE SVM

Recent years have witnessed an unprecedented explosion of automatically generated data, as collecting data becomes simpler and cheaper. Even as collecting data becomes simpler and cheaper, most of this data is unlabeled. The amount of data that can be labeled by experts is not increasing by the same rate—and the gap between available labeled and unlabeled data is widening rapidly.

In the presence of unlabeled data, semisupervised learning (SSL) [Zhu, 2005] can improve classification accuracy by incorporating additional information from the underlying data distribution. The transductive support vector machine (T-SVM) [Joachims, 1999] is arguably amongst the most successful SSL algorithms. T-SVM extends the large margin principle of support vector machines (SVM) towards the unlabeled data. The separating hyperplane is repeatedly readjusted to stay clear of dense regions of the input space. This approach uses the unlabeled data to uncover cluster structure and naturally incorporates it into the decision boundary. However, incorporating such additional knowledge comes at the price of extra computational complexity. For example, T-SVM scales cubically with the size of the unlabeled dataset [Collobert et al., 2006a]. This means that as unlabeled data sizes increase, not all unlabeled data can be incorporated into training.

In this work we present a subsampling algorithm to make use of unlabeled data more effectively in classification. To understand our approach, it is helpful to understand how T-SVM incorporates unlabeled data. T-SVM uses unlabeled data to guide the decision boundary to have a balanced amount of positive and negative samples on either side. Further, by enforcing a large margin even for the unlabeled data, it avoids scenarios where the hyperplane cuts through dense regions (clusters) within the input space.

We claim that for most datasets only a small amount of unlabeled data can be sufficient to guide the hyperplane in the T-SVM fashion. In other words, our goal is to speed up training by subsampling the unlabeled data as aggressively as possible, while only

minimally affecting T-SVM's decision boundary. Our approach is based on the insight that we only require high sample density in regions near the hyperplane—which are inherently sparse. All other regions of the input only need to be sampled sufficiently to be "avoided" by the classifier. Put another way, if we can identify the regions that are likely to contain support vectors we should oversample them and under sample everything else.

By definition, support vectors lie close to the decision boundary and they tend to be within sparse regions of the unlabeled data, due to the influence of T-SVM. We define efficient heuristics to identify both conditions. Firstly, we can predict which inputs are close to the decision boundary with the current estimate of the T-SVM separating hyperplane, which is refined in each iteration. Secondly, we can identify inputs between dense regions by clustering the data and measuring the instability of each input's cluster assignment. Before each iteration of T-SVM, we subsample a new batch of unlabeled inputs—thus refining our sample based on our current belief about the decision boundary and the data's cluster structure. Figure 6.1 illustrates this intuition on a 2-dimensional classification task with labeled and unlabeled data. The graph also indicates which unlabeled inputs were selected by our subsampling algorithm.

We confirm empirically that our sampling strategy rapidly "narrows in" on the region of actual support vectors—thus effectively managing to distill the important aspects of the unlabeled data. We further demonstrate that we can effectively sample data from these regions and guide T-SVM with only a tiny fraction of the original unlabeled data corpus. The resulting algorithm achieves orders of magnitude speedup during training, without significant impact on the classification accuracy.

Many fast T-SVM algorithms have been introduced in the last decade that are based on various novel optimization algorithms. We consider our work complementary to those of previous approaches. As our method is based on subsampling of the unlabeled data, it can effectively be combined with any one of these algorithms. In fact, we build upon the work by Collobert et al. [2006b], who published one of the fastest T-SVM solvers to date. We use their fast implementation throughout and further improve upon their result by providing several orders of magnitudes in additional speedup.

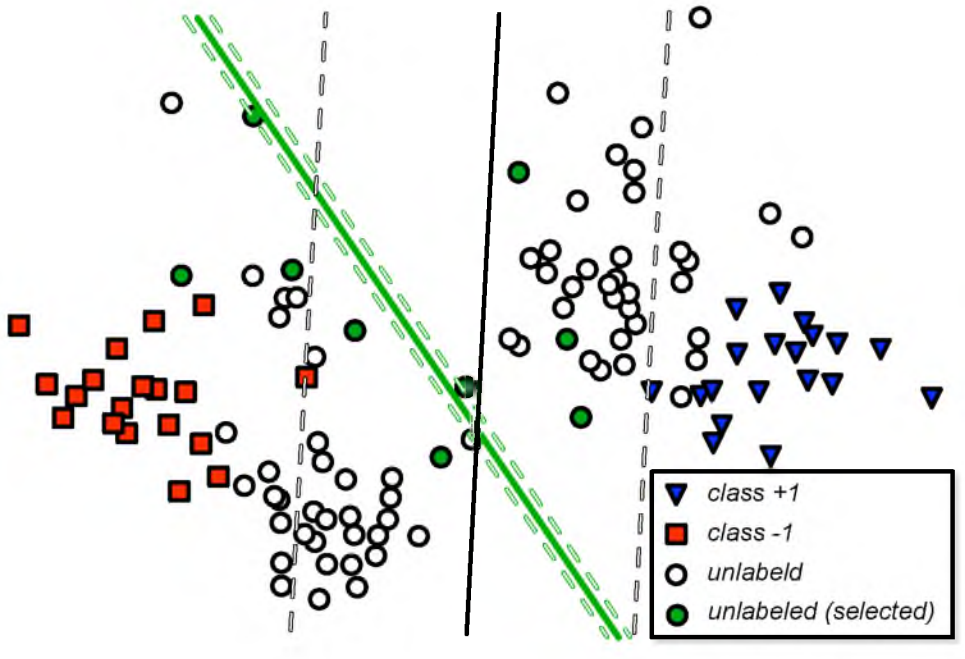


Figure 6.1. Decision boundaries and margins of SVM (black) and T-SVM (green) on a binary classification problem in 2-dimensions. T-SVM uses unlabeled data and finds a better decision boundary through the sparse region. Unlabeled inputs selected by our subsampling algorithm are highlighted in green.

6.1 Preliminaries

Throughout this chapter we type vectors in bold ($\mathbf{x}_i$), scalars in regular (k or C), sets in cursive ($\mathscr{S}$) and matrices in capital bold ($\mathbf{K}$) font. Specific entries in vectors or matrices are scalars and follow the corresponding convention.

We assume that we are provided with a labeled dataset $\mathscr{D}_L = \{\mathbf{x}_1, \dots \mathbf{x}_\ell\} \subset \mathscr{R}^d$ with corresponding binary labels $\{y_1, \dots, y_\ell\} \in \{-1, 1\}$ and unlabeled data $\mathscr{D}_U = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+u}\} \subset \mathscr{R}^d$. Both labeled and unlabeled inputs are sampled i.i.d. from the same (unknown) data distribution. For convenience, let $n = \ell + u$ denote the total number of labeled and unlabeled inputs.

In the following, we provide a brief overview of SVM [Schölkopf and Smola, 2002], T-SVM [Joachims, 1999] and soft c-means clustering [Bezdek, 1981].

6.1.1 SVM

The objective of the original SVM classifier is to learn a function $h_\theta : \mathscr{R}^d \to \{-1, 1\}$, with $\theta = (\mathbf{w}, b)$, such that $h_\theta(\mathbf{x}_i) > 0$ if $y_i = 1$ and $h_\theta(\mathbf{x}_i) < 0$ if $y_i = -1$. The function h_θ defines a separating hyperplane $h_\theta(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. This hyperplane is learned with the

following optimization

$$\min_{\theta} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{\ell} H(y_i h_\theta(\mathbf{x}_i)). \tag{6.1.1}$$

Here, $H(y_i h_\theta(\mathbf{x}_i)) = \max[1 - y_i h_\theta(\mathbf{x}_i), 0]$ denotes the hinge loss, which penalizes all predictions $h_\theta(\mathbf{x}_i)$ whose sign does not agree with the corresponding label y_i and all predictions of low magnitude, i.e., $|h(\mathbf{x}_i)| < 1$. In other words, the hinge loss penalizes inputs that are misclassified or that are too close to the decision boundary; the latter enforces a margin of empty space around the separating hyperplane, which has been shown to have particularly strong generalization properties [Schölkopf and Smola, 2002]. The hyperparameter C regulates how much violations are penalized.

6.1.2 T-SVM

T-SVM [Joachims, 1999] extends this formulation and incorporates the unlabeled inputs $\mathscr{D}_U$. Although the labels for $\mathscr{D}_U$ are unknown, we do know some things about them: first, as D_L and D_U are both sampled i.i.d. from the same distribution, their class ratios should approximately agree. Second, a SVM trained on the labeled data $\mathscr{D}_L$ should generalize to some degree to $\mathscr{D}_U$.

Joachims combines these two insights: a SVM is trained on the labeled data $\mathscr{D}_L$ to obtain a classifier h_θ^0. Each input $\mathbf{x}_j \in \mathscr{D}_U$ is assigned a label $\hat{y}_j = 1$ if $h_\theta^0(\mathbf{x}_j) > t$ and $\hat{y}_j = -1$ otherwise. The threshold t is chosen such that the class proportions in $\mathscr{D}_U$ match those in $\mathscr{D}_L$, i.e., $\frac{1}{\ell}\sum_{i=1}^{\ell} y_i \approx \frac{1}{u}\sum_{j=\ell+1}^{n} \hat{y}_j$.

Once the "labels" $\hat{y}_j$ are assigned, T-SVM incorporates them into the hyperplane optimization in a similar fashion as (6.1.1):

$$\min_{\theta} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{\ell} H(y_i h_\theta(\mathbf{x}_i)) + C^* \sum_{j=\ell+1}^{n} H(\hat{y}_j h_\theta(\mathbf{x}_j)).$$

In other words, T-SVM enforces a margin around unlabeled inputs—thus guiding the hyperplane to avoid densely sampled regions of the space. The constant C^* regulates to what degree the estimated "labels" for the unlabeled data $\hat{y}_j$ are trusted. Initially it is set to a very low value (e.g., $C^* = 10^{-5}$). T-SVM iterates between solving the optimization problem and reassigning the labels for the unlabeled set, while the constant C^* is increased by a multiplicative factor in each iteration.

An important fact to note is that by enforcing a margin on the unlabeled inputs, T-SVM guides the decision boundary through low density regions in the data space. It is this insight, that we make use of in subsequent sections.

Similar to the canonical SVM, there is a natural extension of T-SVM to nonlinear decision boundaries with the kernel trick, as described in Schölkopf and Smola [2002].

6.1.3 Ramp Loss T-SVM

Although the worst case complexity of T-SVM is $O(n^3)$, in practice its computation is dominated by operations with quadratic complexity for most datasets. Collobert et al. [2006b] introduce a faster variation of T-SVM, by substituting the hinge loss H with the ramp loss $R(z) = \min(H(z), s)$. The ramp loss caps the loss suffered by any particular input $\mathbf{x}_j$ to at most $s > 0$. With this slight relaxation, the loss function can be decomposed into a convex concave function, which can be optimized more efficiently with the concave convex procedure (CCCP) [Yuille and Rangarajan, 2001]. Although this variation does not affect the asymptotic complexity, it does tend to speed up the training time significantly in practice. Throughout this chapter we use their implementation, which is the best scaling implementation of T-SVM that we are aware of. We describe the T-SVM algorithm in Algorithm 4.

6.1.4 Soft Clustering

To identify relevant unlabeled points for our sampling strategy, we will make use of a soft clustering subroutine that we describe here. A clustering algorithm takes a collection of inputs $\mathscr{D}$ and partitions them into a fixed number of groups $\mathscr{C}_1, \ldots, \mathscr{C}_c$ of "similar" objects by minimizing an appropriate cost measure. Please note that for purposes of soft partition, we only look at the unlabeled inputs $\mathscr{D}_U$. Therefore, the resulting partition can be represented by an assignment function $f : \mathscr{D}_U \to [1 \ldots c]$ that maps each input to exactly one of c clusters. In a soft partition, inputs may be assigned to multiple clusters with different weights, as long as these weights sum to one. Specifically, a soft partition is represented as a membership probability function $f : \mathscr{D}_U \to \Delta^{c-1}$, where $\Delta^{c-1} = \{(p_1, \ldots, p_c) \mid p_i \geq 0,\ \sum_i p_i = 1\}$ is the standard simplex in c-dimensions. The i^{th} component $f_i(\mathbf{x})$ is the probability that $\mathbf{x}$ is assigned to cluster $\mathscr{C}_i$, and by definition $\sum_i f_i(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathscr{D}_U$.

There are many algorithms that can produce a soft partition minimizing an appropriate

Algorithm 4 T-SVM

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_u, y_1, \ldots, y_\ell$
  Solve (6.1.1) to obtain h_θ^0
  $C^* = 10^{-5}$
  $T = \sum_{i=1}^{\ell} \delta(y_i = 1)$
  while algorithm has not converged **do**
    Let $\ell < j \leq u$ be the T^{th} largest prediction h_θ
    Solve (6.1.2)
    $C^* \leftarrow 2C^*$
  end while
  Return $f_i(j) = w_{ij} / \sum_j w_{ij}$

cost measure [Xu et al., 2005]. For our purposes, we require an algorithm that scales well with the number of unlabeled inputs. Therefore, we use the fuzzy c-means algorithm developed by Bezdek [1981] and described in Algorithm 5. The parameter m controls the "hardness" of the clustering: setting $m = 1$ yields the standard "hard" k-means algorithm [Trevor et al., 2001]. In practice, $m = 2$ is a good choice. Each iteration of the algorithm runs in time $O(u \cdot c)$ and the algorithm typically converges in only a few iterations [1]. Since we will typically use a constant ($c \leq 5$) number of clusters for our algorithm, the algorithm runs effectively in linear time.

We use soft clustering as a fast way of computing clustering affinities. The sampling methods we discuss are agnostic to the choice of the method that computes these affinity scores. In particular, one could use the NNI-based stability scores that we introduced in Chapter 5. Although we discuss a few shortcomings of the affinity score generated by a soft clustering algorithm, we observe very similar affinity vectors to the NNI-based stability score for the case of two clusters, which is the case with binary classification. Hence, we prefer a quick run of fuzzy c-means to compute the affinity scores.

6.2 Adaptive Subsampling for T-SVM

As previously mentioned, T-SVM makes two critical assumptions: (a) the true decision boundary does not pass through dense regions of the input space and (b) the class ratio of the predictions should match the class balance within the labeled data. It utilizes these

[1] For clarification, despite its name, the fuzzy c-means algorithm does not use fuzzy logic in any way.

Algorithm 5 Fuzzy c-means [Bezdek, 1981]

Input: $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n$, #clusters c, hyperparameter m
  Initialize $\mathbf{c}_1, \ldots, \mathbf{c}_C$ randomly
  while algorithm has not converged **do**
    for $i = \ell+1 \ldots n$; $k = 1 \ldots c$ **do**

$$w_{ik} = \left[\sum_{k'=1}^{n} \left(\frac{\|\mathbf{c}_k - \mathbf{x}_i\|}{\|\mathbf{c}_{k'} - \mathbf{x}_i\|} \right)^{2/(m-1)} \right]^{-1}$$

    end for
    for $k = 1 \ldots c$ **do**

$$\mathbf{c}_k = \frac{\sum_i w_{ik} \mathbf{x}_i}{\sum_i w_{ik}}$$

    end for
  end while
  Return $f_i(k) = \dfrac{w_{ik}}{\sum_{k'} w_{ik'}}$

two assumptions and guides the classifier along corridors in the input space of low sample density between dense regions (clusters). The surrounding dense regions on either side of the hyperplane ensure the correct class ratio of the predictions—if there are more inputs sampled on one side of the hyperplane than on the other, the predictions are lopsided and T-SVM corrects the hyperplane accordingly.

In this section, we first identify a function $\sigma : \mathscr{D}_U \to [0,1]$, which assigns a weight to all unlabeled inputs that captures their likelihood of lying in such regions of interest. We then subsample our unlabeled data proportional to the σ scores and reduce it to a small fraction of its initial size. The function σ consists of two components: one part reflects cluster structure in the data by identifying regions of low density that are more likely to contribute to a decision boundary, and the other part reflects information provided by the current best guess for a separator to identify points that are likely to be informative.

6.2.1 Cluster Entropy

As a first step, we are interested in identifying corridors of low sample density between clusters. Inputs in sparse regions are by definition not near any cluster and therefore will not obtain a sharp cluster assignment from an algorithm like c-means. We can measure this uncertainty by performing c-means and computing the entropy of the resulting clustering distribution $\mathbf{p} = [p_1, \ldots, p_C]^\top$:

$$H_C(\mathbf{p}) = -\sum_{k=1}^{C} p_k \log(p_k). \tag{6.2.1}$$

The cluster entropy is minimized if a single cluster is sharply assigned with probability $p_k = 1$, in which case $H_C(\mathbf{p}) = 0$. In contrast, it is maximized in the setting of maximum uncertainty, i.e., if $p_k = \frac{1}{C}$ for all clusters k. Note that the cluster assignment $\mathbf{p}$ is a function of the input $\mathbf{x}$, and we therefore write $H_C(\mathbf{p}(\mathbf{x}))$ to denote the cluster entropy of input $\mathbf{x}$. We visualize the cluster entropy scores in Figure 6.2.

6.2.2 Label Uncertainty-Based Scores

The cluster entropy identifies regions of low sample density between dense regions (clusters). There may be many such regions, not all of them near the decision boundary of the SVM. Each iteration of T-SVM refines the parameters of the hyperplane. As $C*$ increases, the parameters slowly "freeze" in place and the changes to the decision boundary become smaller and smaller.

We can utilize the fact that with each iteration we obtain a better forecast of the final decision boundary and compute a label uncertainty score of the current T-SVM classifier. Inputs with high label uncertainty are those that lie right around the decision boundary—the region of interest for the T-SVM classifier.

We use Platt's scaling technique [Platt, 1999] to turn the T-SVM predictions into properly scaled probabilities. More explicitly, once we obtain the decision rule $h_\theta(\mathbf{x})$, we define the posterior probability $P(y = 1|\mathbf{x}) = \left[1 + e^{ah_\theta(\mathbf{x})+b}\right]^{-1}$ with the constants a, b obtained via a straight-forward maximum likelihood optimization. This step is extremely fast, because only two free parameters are estimated. With this notation, we can define the label uncertainty score as

$$\sigma_l(\mathbf{x}) = P(\hat{y}_\theta(\mathbf{x})|\mathbf{x}), \tag{6.2.2}$$

where $\hat{y}_\theta(\mathbf{x}) = \text{sign}(h_\theta(\mathbf{x}))$. We visualize the Platt's scaling scores in Figure 6.3.

6.2.3 Adaptive Subsampling

Finally, we combine (6.2.1) and (6.2.2) and define our final weighting function as

$$\sigma(\mathbf{x}) = \sigma_l(\mathbf{x}) H_C(\mathbf{p}(\mathbf{x})). \tag{6.2.3}$$

Given $\sigma(\cdot)$ as defined in (6.2.3), we sample m inputs from $\mathscr{D}_U$, where the input $\mathbf{x}$ is picked without replacement in proportion to the score $\sigma(\mathbf{x})$. To do this, we use a standard

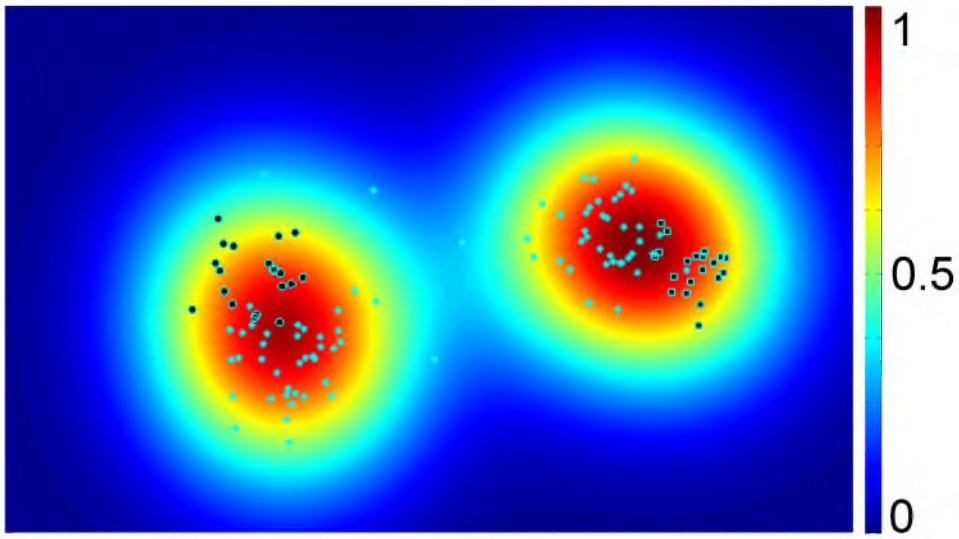


Figure 6.2. Visualizing the cluster entropy scores, running fuzzy c-means.

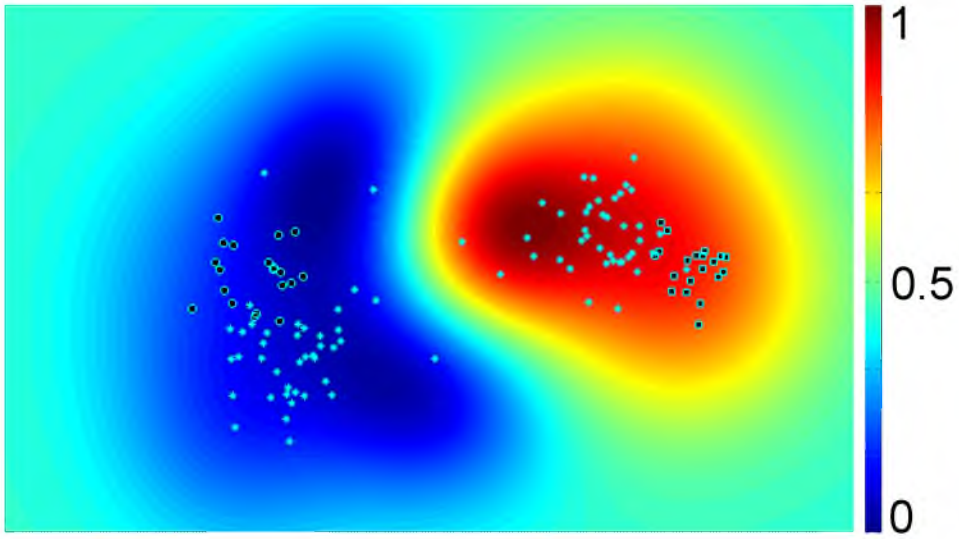


Figure 6.3. Visualizing Platt's scaling scores.

acceptance/rejection sampling algorithm summarized in Algorithm 6. As discussed in the dissertation of Olken [1993], this algorithm is well suited for sampling when the weights are ad hoc and may be updated frequently, as is the case in our setting.

We repeat the subsampling before each iteration to obtain a new set of m unlabeled inputs. We visualize the decision boundaries and margins of running the full T-SVM in Figure 6.4 and the adaptive T-SVM in Figure 6.5. It is important to note that only the value of $\sigma_l(\cdot)$ is updated after each iteration to incorporate the new decision boundary. The value

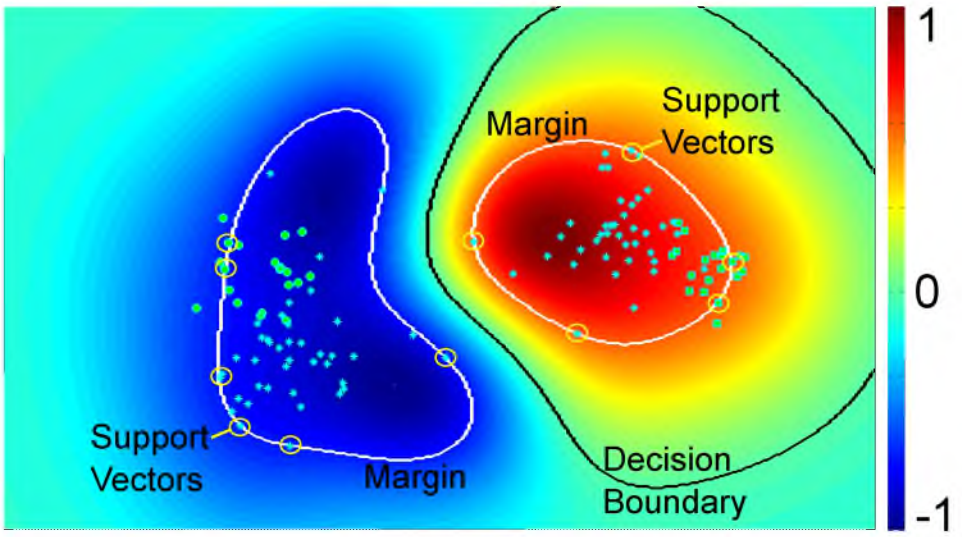


Figure 6.4. Visualizing T-SVM decision boundary and margins on full data.

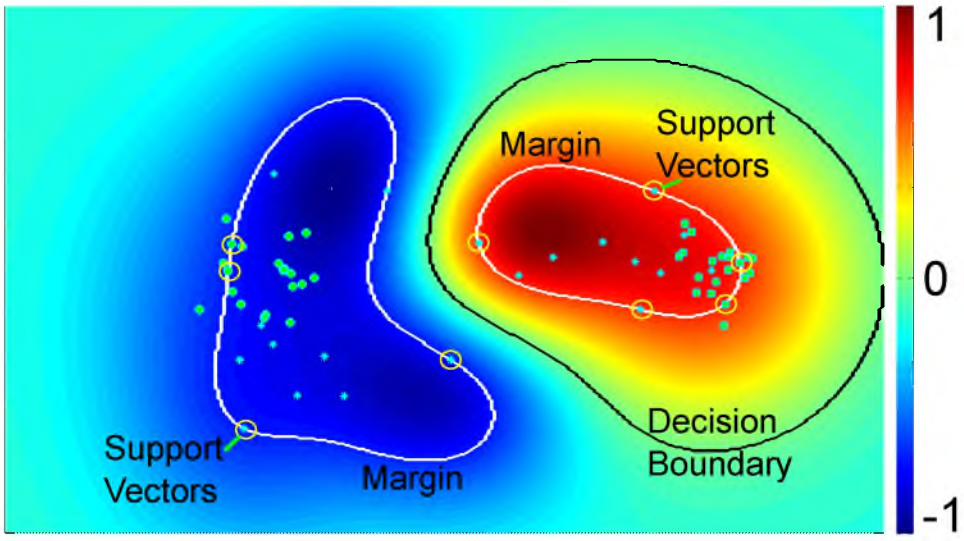


Figure 6.5. Visualizing T-SVM decision boundary and margins on adaptively sampled data.

Algorithm 6 Sampling m inputs without replacement proportional to σ.

$S \leftarrow \emptyset$
while $|S| < m$ **do**
  Pick an index i uniformly in $[\ell+1, n]$.
  Pick a uniform random number $r \in [0,1]$ and add i to the sample S if $r \leq \sigma(\mathbf{x}_i)$
end while

of $H_C(\cdot)$ remains unchanged throughout, as the clustering is performed only once prior to learning. For the first iteration we set $\sigma_l(\mathbf{x}) = 1$ for all inputs, as no classifier has been trained yet.

6.2.4 Complexity Analysis

The motivation behind our adaptive sampling is to reduce T-SVM's input size. This leads to speedups only if the subsampling algorithm itself scales significantly better than T-SVM itself, which scales $O(n^3)$ in the worst case but often behaves like $O(n^2)$ in practice.

Each iteration of the fuzzy c-means algorithm takes linear time and we run it for a fixed number of iterations. Since we run the algorithm only on the unlabeled data $\mathscr{D}_U$, the running time is $O(u)$. Each iteration of the Newton method used to estimate the Platt's scaling scores also takes linear time. In practice, the process converges in a small (constant) number of iterations. Finally, the sampling algorithm will generate a new sample (the inside of the `while` loop in Algorithm 6) in expected time $(E[\sigma])^{-1}$ by standard bounds on geometric distributions. The expression $E[\sigma]$ is the expected value of σ over a uniform sample of the points. It is typically a small constant. Thus sampling m elements takes $O(m)$ time.

Thus, the overall complexity of the sampling procedure is $O(n)$ time per iteration of T-SVM.

6.3 Experiments

We demonstrate the efficacy of our method on a variety of datasets in this section. We compare the adaptive subsampling for T-SVM with uniform subsampling, regular SVM and T-SVM on all unlabeled inputs. We observe that even at roughly 10% sample size, across all datasets, we achieve over 20x speedups with little impact on accuracy.

6.3.1 Experimental Setup

All the experiments were run on an Intel Quad Core CPU 2.66GHz machine with 4GB RAM. Our implementation is a modification of Collobert et al. [2006a]. We describe the data that we use in Table 6.1.

Table 6.1. All five datasets (sorted by fraction of unlabeled data) and their statistics: number of labeled inputs (ℓ), number of unlabeled inputs (u), fraction of unlabeled data ($\frac{u}{\ell+u}$), dimensionality (d).

Dataset	ℓ	u	u/n	d
a8a (Adult)	22696	9865	0.30	123
Gisette	1000	6000	0.56	5000
Svmguide1	3089	4000	0.86	4
CodRNA	59535	429030	0.88	8
a1a (Adult)	1605	30956	0.95	123

6.3.2 Datasets and Methodology

We evaluate our algorithm on five medium and large datasets Svmguide1, Gisette, Adult, and CodRNA downloaded from the LIBSVM data page[2]. All the datasets contain two classes and are feature scaled as a part of preprocessing. Table 6.1 summarizes their statistics in terms of labeled dataset size (ℓ), unlabeled dataset size (u) and dimensionality (d). During the development of our algorithm, none of these datasets were ever used. Instead, we developed adaptive T-SVM with the help of six (smaller) datasets from the UCI machine learning repository [Bache and Lichman, 2013] (Iris, Ionoshpere, Sonar, Heart, Pima, and Mushrooms). We purposely chose datasets that are too small to require subsampling, in order to keep the interesting medium or large scale data "untouched" for evaluation. To enforce a strict separation between development and evaluation data, we omit results on these smaller datasets. (They tend to be comparable or slightly better in terms of accuracy convergence, however with smaller speedups due to their at times tiny sizes.).

We compare our method against a naïve uniform subsampling, where in each iteration, we pick a random sample of the unlabeled inputs for consideration by the T-SVM algorithm. We perform each experiment five times (with identical train/test splits) and report the average accuracies and standard deviations.

---

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

6.3.3 Hyperparameters

Throughout we use T-SVM and SVM with a radial basis function (RBF) kernel and set all hyperparameters, including the bandwidth of the RBF kernel and the regularization constant, by 5-fold cross validation on the labeled training data.

We run fuzzy c-means with $c=2$ clusters to obtain the cluster entropy of the unlabeled inputs. This setting was chosen as it leads to the fastest clustering convergence and resulted in high accuracy on our development datasets. Figure 6.6 shows the sensitivity of adaptive subsampling with respect c on the Svmguide1 data. The graph supports that the algorithm is fairly insensitive to the number of cluster centers and $c=2$ appears to be a good choice.

6.3.4 Performance

Figure 6.7 shows accuracy levels as a function of the size of the subsampled dataset on all five benchmark tasks. It compares T-SVM with adaptive (black line) and uniform subsampling (red line), as well as T-SVM on the full data (blue triangle) and supervised SVM (dashed green line).

6.3.4.1 Accuracy. Concerning the test accuracy, there are three clear trends that can be observed.

1. T-SVM with adaptive subsampling obtains strictly higher accuracies than uniform subsampling on all datasets, at all sampling rates.
2. Adaptive subsampling leads to strictly lower variance than uniform subsampling across all settings.
3. As the sampling size increases, the accuracy of T-SVM with adaptive subsampling rapidly approaches that of T-SVM on the full dataset typically reaching very similar levels of accuracy already well below a sampling rate of 10% (Note that the horizontal axis in Figure 6.7 is in log scale).

Uniform subsampling suffers from high variance especially in the low sample regions—a sign that T-SVM puts too much emphasis on the few unlabeled inputs that are available. If these are positioned far from the hyperplane, large changes can be forced onto the SVM classifier, possibly based on sampling artifacts rather than true structure in the data distribution. Adaptive subsampling mitigates this effect by focusing consistently on inputs near the decision boundary and relevant regions. This observation is also consistent with results on dataset a8a (Adult), which has the lowest fraction of unlabeled data. Here, the

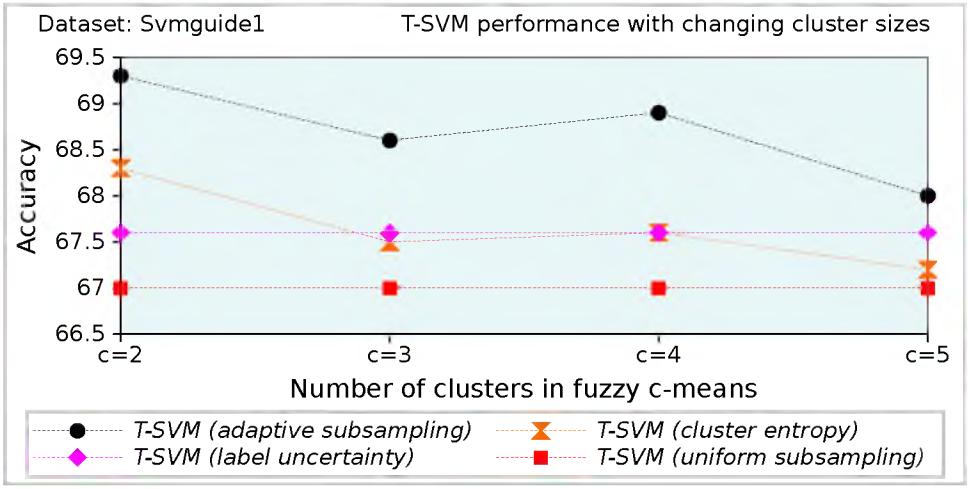


Figure 6.6. Effect of using clustering with different cluster numbers on the different strategies for subsampling data for T-SVM, on the Svmguide1 dataset.

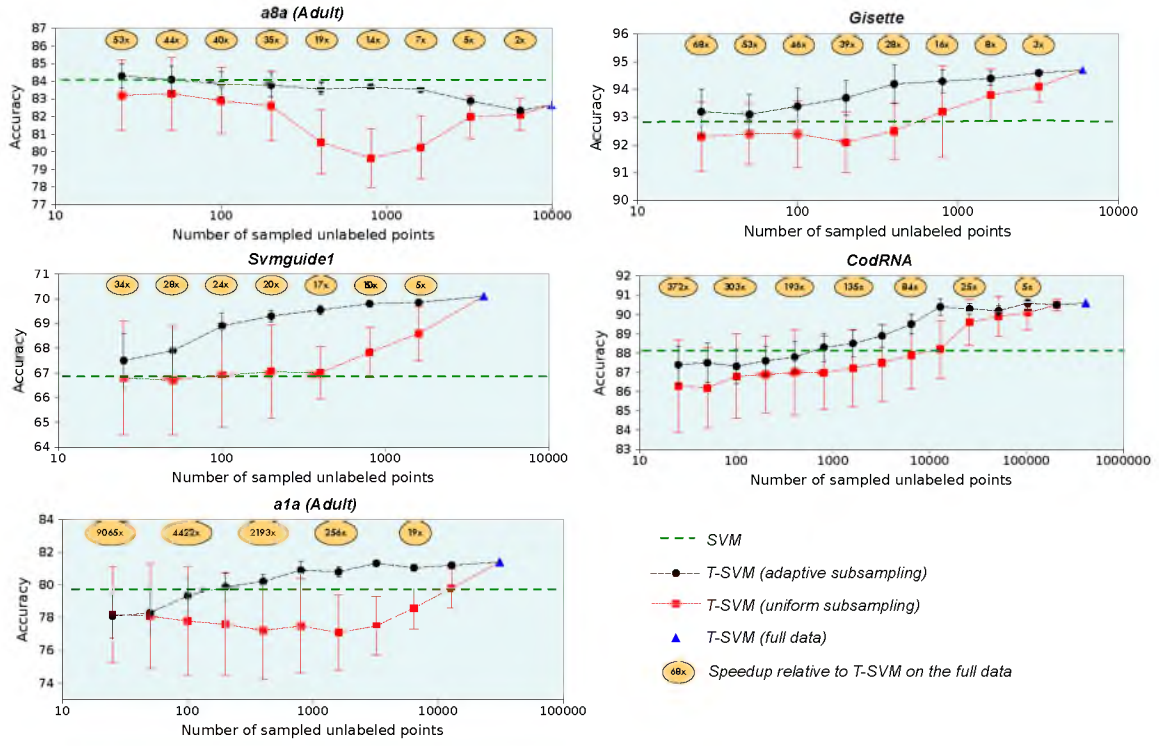


Figure 6.7. Comparing performances of T-SVM with adaptive and uniform subsampling along with T-SVM and SVM on full data.

amount of unlabeled data is insufficient and T-SVM reduces the test accuracy. Uniform subsampling magnifies this negative effect even further, whereas adaptive subsampling dampens it substantially.

6.3.4.2 Training speedup. The additional overhead of the adaptive subsampling (including the clustering) ranges from $1.1s$ for the smallest dataset (Svmguide1) to $66s$ for CodRNA (the largest dataset). Compared with the quadratic/cubic time complexity of the T-SVM training, see Table 6.2, this overhead is negligible for all five datasets.

Consequently speedup can be considered a function of purely the sample size rather than subsampling method. The speedup at each level of subsampling is superimposed on the top of each graph in Figure 6.7.

As expected, datasets with larger fractions of unlabeled inputs (and large number of unlabeled inputs in absolute terms) tend to have higher speedups (a1a, CodRNA). As we observed previously, at a sample size of about 10% the adaptive T-SVM tends to stabilize in terms of the standard deviation and is very close to the T-SVM accuracy without subsampling. Table 6.2 depicts the exact training times and test accuracies after 10% subsampling and on the full dataset. Although adaptive subsampling to 10% reduces the training time to a mere 5% of its original amount, it has surprisingly little impact on the test accuracies. Even higher speedups are possible by sampling fewer inputs and effectively trading off some accuracy for speed (see Figure 6.7).

The high speedups reported in Figure 6.7 and Table 6.2 are particularly impressive as our baseline is already the fastest and best scaling T-SVM implementation we are aware of Collobert et al. [2006a].

6.3.5 Further Analysis

In the following we provide some additional analysis of the individual components of our adaptive subsampling algorithm and its effect on the support vectors.

6.3.5.1 Subsampling. Figure 6.8 compares the various components of adaptive subsampling on the Svmguide1 dataset. The figure shows four graphs that differ in the way unlabeled inputs are subsampled during the T-SVM iterations. The four lines represent adaptive subsampling i.e., subsampling proportional to $H_C(\cdot)c_l(\cdot)$ (black line), sampling proportional to the cluster entropy $H_C(\cdot)$ (orange line), sampling proportional to the label

Table 6.2. Accuracies and training times for the five datasets with a 10% subsampling rate and without subsampling (100%). The table shows that adaptive subsampling reduces the T-SVM training time to a small fraction of the original amount with very little impact on test accuracy.

	training time		test accuracy	
Sampling rate	100%	10%	100%	10%
Svmguide1	1m	6s	70.1	69.3
a1a	1h 30m	33s	82.7	83.6
a8a	1h 12m	6m 17s	80.6	81.0
Gisette	10m	27s	94.7	94.2
CodRNA	2d 1h 37m	2h 29m	90.6	90.3

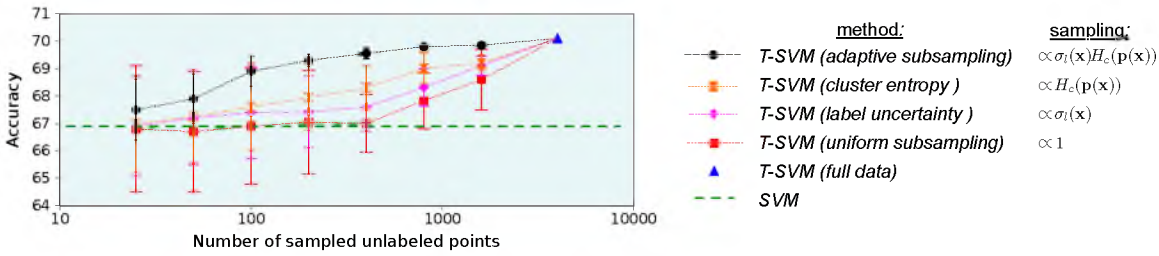


Figure 6.8. A comparison of the different strategies for subsampling data for T-SVM (on the Svmguide1 dataset). Both, the label uncertainty and the cluster entropy, outperform uniform subsampling. Their combination (adaptive subsampling) leads to big additional gains in accuracy throughout, has the least variance and reaches comparable accuracies to T-SVM on the full data much sooner.

uncertainty $\sigma_l(\cdot)$ (pink line), uniform subsampling (red line), T-SVM on the full data (blue dot), and canonical supervised SVM (dashed green horizontal line).

We observe a clear trend that both components of the adaptive subsampling (cluster entropy and label uncertainty) individually improve over uniform subsampling. Their combination, adaptive subsampling, improves even further. In fact, we observe that the gain in accuracy through adaptive subsampling (over uniform) is even higher than the sum of the two gains through cluster entropy and label uncertainty. This indicates that both components are necessary and contribute in complementary ways to the weighting function.

6.3.5.2 Support vectors. Figure 6.7 indicates that T-SVM with adaptive subsampling converges much sooner to the results of the true T-SVM classifier than uniform

subsampling. As the SVM decision boundary can be defined entirely in terms of support vectors [Schölkopf and Smola, 2002], it is interesting to see if the approximate support vectors (aSV) chosen under adaptive subsampling are close to the true support vectors (tSV) chosen without subsampling. To investigate how similar the two sets of support vectors are, we define two metrics in the spirit of precision and recall [Baeza-Yates and Ribeiro-Neto, 1999]. For the two sets of support vectors to define similar decision boundaries it is not only desired to have each aSV close to an existing tSV (precision), but also each tSV close to an aSV (recall). Figure 6.9 shows the average normalized distances from each aSV to its nearest tSV (left) and from each tSV to its nearest aSV (right). The results are averages across five runs at 10% sampling rate with corresponding standard deviations. Note that in contrast to precision and recall, here lower values are better as we consider distances to and from support vectors.

Let the support vectors returned by the adaptive T-SVM be $\mathscr{S}^a$ and the ones returned by T-SVM with random samples be $\mathscr{S}^b$. We first compute the the average value of the normalized nearest neighbor distances between the sets $\mathscr{S}^a$ and $\mathscr{S}^{true}$ and vice versa. We repeat this for the sets $\mathscr{S}^b$ and $\mathscr{S}^{true}$. $\frac{\sum_i d(\mathscr{S}^a_i, NN(\mathscr{S}^a_i, \mathscr{S}^{true}))}{|\mathscr{S}^a|}$ represents the average distance between the sets $\mathscr{S}^a$ and $\mathscr{S}^{true}$ and $\frac{\sum_j d(NN(\mathscr{S}^{true}_j, \mathscr{S}^a), \mathscr{S}^{true}_j)}{|\mathscr{S}^{true}|}$ represents the average distance between the sets $\mathscr{S}^{true}$ and $\mathscr{S}^a$, where *NN* represents the normalized nearest neighbor distance. It is easy to see that these values lie between zero and one.

The graph highlights the drastic difference between uniform and adaptive subsampling across all datasets. Support vectors obtained with adaptive subsampling tend to be consistently at about half the distance to/from original support vectors. Further, we observe that subsampling according to cluster entropy and label uncertainty also guide the decision boundary closer to its accurate location. Similar to Figure 6.8, their combination (adaptive subsampling) leads to substantial additional improvements—indicating that both components act complementary and are necessary to identify regions of likely support vectors.

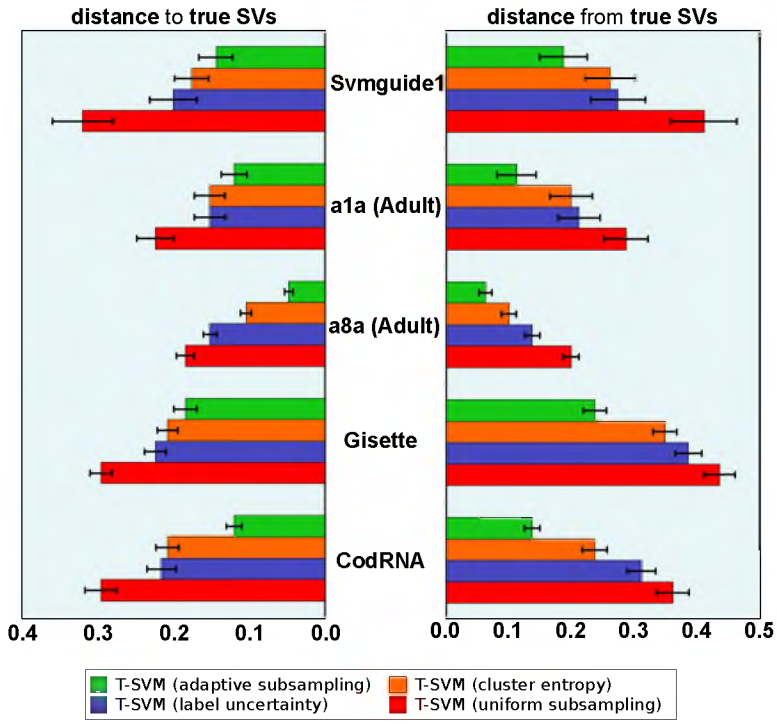


Figure 6.9. Adaptive subsampling samples from regions in the input space that are likely to contain support vectors. This graph shows the average normalized distances between approximate support vectors to the closest true support vectors (obtained with T-SVM without subsampling) and vice versa (right). The graph shows that sampling proportionally to label uncertainty and cluster entropy each reduce the distances to and from true support vectors. The combination of the two (adaptive subsampling) guides the decision boundary most accurately.

6.4 Summary

SSL algorithms can improve classification accuracy because they assume and uncover hidden structure in the input distribution, which can not be extracted from the limited available labeled data. Different algorithms discover different signatures. For example, T-SVM discovers the dense regions in the input space and encourages the decision boundary to circumvent them; Laplacian regularization ([Belkin and Niyogi, 2004]) uncovers the manifold that underlies the data distribution and encourages predictions to change smoothly along this manifold; cotraining (introduced by Blum and Mitchell [1998]) assumes that data can be represented in label-conditionally independent ways and makes classifiers for both views agree as Chen et al. [2011] discovered from the unlabeled data.

Our research is based on the insight that if unlabeled data is only used to discover certain signature properties, it may be possible to still perform this task on a small fraction of its

original size. If this reduction is of smaller complexity than the SSL algorithm, massive speedups are obtainable. As future work we will investigate how to apply this insight on different families of SSL algorithms. For example, manifold data is often oversampled in regions with little curvature, where the manifold is easier to capture, and requires higher sample density in regions of high curvature.

For very large dataset sizes it may also be possible to perform the subsampling already during data collection. This could potentially reduce storage requirements and network traffic drastically. Our current algorithm is integrated into T-SVM, but one could imagine variations that already subsample (less aggressively) prior to learning.

In this chapter we introduced adaptive T-SVM, a novel large-scale SSL algorithm. Our approach subsamples the unlabeled dataset effectively, while preserving regions of interest for T-SVM's decision boundary. This is achieved by incorporating two key components: label uncertainty and cluster entropy. We demonstrated that both components combined can successfully predict regions of interest to the T-SVM algorithm—a fact that can be exploited to drastically reduce the size of the unlabeled data. Adaptive subsampling can obtain orders of magnitude speedups, with negligent or no impact on T-SVM's accuracy.

As datasets keep growing at a much faster rate than data can be labeled, SSL will continue to increase in importance. Our adaptive T-SVM algorithm is amongst the first to make SSL practical on large scale data. Although our approach is focused on T-SVM as a specific algorithm, it follows a paradigm that we hope will spawn interesting followup work across many relevant research areas.

CHAPTER 7

CONCLUSION

Data is ever growing and there is an immediate need for developing robust techniques to analyze data and validate the results. There has been substantial research contributions in developing various clustering algorithms that help to preprocess data and separate data into meaningful groups. In fact, Lloyd's algorithm to k-means, which is one of the top ten clustering methods chosen by IEEE International Conference on Data Mining (ICDM), still remains a good choice for any practitioner. But, there are more fundamental questions that have still not been answered in satisfying manner. These questions include

1. Have I clustered my data correctly?
2. Can I do any better on my data?
3. Do I have the correct number of clusters?
4. Are there different, yet useful partitions that can be mined from the data?
5. Are the data points that have been assigned incorrectly?

We ask these questions in this dissertation and provide solutions that both answer them and open up a few more questions.

While clustering can be immensely useful in exploratory data analysis, there is no oracle to label the data or verify the solutions. This dissertation therefore focused on exploring the landscape of clusterings data, providing the user with a robust variety of solutions and validating the results towards a larger goal of verifiable data mining. The applications that benefit from our methods and measures were introduced throughout the dissertation and support the foundational work done as part of this dissertation. We view our work as one that the practitioners could hugely benefit from, since many of them have the questions that we mentioned above.

Our focus in this dissertation was developing robust algorithms for metaclustering and computing stability of partitions and reduce the computational complexity involved in these methods. We analyze the error tradeoffs arising due to the use of various sampling proce-

dures to make the runtimes faster and substantiate the choice of hyperparameters involved by a collection of experiments. We can now represent partitions succinctly, compare them in a well founded and efficient manner, compute a consensus solution to create an ensemble of the input partitions, generate a variety of partitions, validate data memberships and use them to speed up various data mining and machine learning applications. We now summarize the contributions of this dissertation briefly.

7.1 Summary of Contributions

- **Comparing partitions**. Using prior work in representing point clouds in the powerful reproducing kernel Hilbert space, we proposed a spatially aware metric to compare partitions that goes beyond the traditional combinatorial measures. We discussed efficient algorithms to compute this metric and we view this as an essential hammer for various metaclustering problems.
- **Evaluating partitions**. We also described a new quality measure to evaluate partitions that we observe to be very good in determining the "goodness" of a partition. This measure along with the capability to compare partitions efficiently, allows us to analyze any given collection of partitions.
- **Consensus clustering**. We reduce the usually technically involved consensus clustering problem to simple clustering in reproducing kernel Hilbert space. Armed with the well founded distance metric to compare partitions, we describe simple clustering methods to find the 1-median solution in the space of partitions that results in a consensus solution.
- **Alternative clustering**. We take the problem of generating alternative partitions to the next level. We describe a Markov chain Monte Carlo sampling procedure to explore the space of all possible partitions of the data, thereby allowing the user to both understand the clusterability of the data and provide him/her with multiple choices of partitions to work with.
- **Validating partitions**. We introduce new point level notions of stability by defining regions of influence of clusters and points and how they interact. These affinity scores are very useful especially in the absence of an oracle that can label the data. We also discuss various applications of such an affinity score in speeding up various

data mining methods and modeling for clustering.

- **Speeding up semisupervised learning**. We describe fast and large scale semisupervised techniques by using adaptive sampling strategies based on the insight gained from defining affinity scores. With the abundance of unlabeled data, our method becomes extremely relevant as we demonstrate significant speedups over existing methods.

We round up this dissertation by proposing a few immediate questions that arise from our line of work.

7.2 Future Challenges

Each chapter in this dissertation provide solutions to various metaquestions on clusterings that aid exploratory data analysis. We are particularly interested in exploring the clustering stability work we discuss in Chapter 5. We believe that the methods that we propose could benefit from a good theoretical study to gain a deeper understanding of how and why they work. In the following, we highlight a few specific questions that we think could further the research in clustering.

- **Speeding up different data mining and machine learning methods**. Can we develop adaptive sampling strategies for other machine learning methods like SVM and regression to make them faster [Balcan et al., 2012; Balcan and Feldman, 2013; Dasgupta, 2010; Ho et al., 2011]? Can we develop distributed clustering methods by using our concise clustering representations and attached affinity scores to communicate efficiently? Can we develop sound theory to bound communication in this model?
- **Heterogeneous clustering**. Due to the lack of an appropriate distance measure between the data objects, many existing clustering methods only work on homogeneous data. Can we define the similarity between different feature set types to enable comparison of the data objects, which is essential for clustering? Can we leverage from the techniques and measures we describe in Chapter 3 to compare and cluster the objects that are in different feature spaces [Aerts et al., 2006; De Bie et al., 2007; Filkov and Skiena, 2004; Huang and Zhu, 2007; Liu et al., 2009; Ye et al., 2008; Yu et al., 2008]?

- **Secondary level distances between partitions**. Distances in high dimensions are unstable since the distribution of pairwise distances is highly skewed towards a single value. From the insights we gained in our work on generating partitions in Chapter 4, we realize that a majority of the partitions have a very narrow range of pairwise distances between them. Can we build a secondary level distance that is induced by LIFTEMD, our primary distance between partitions? Shared nearest neighbors (SNN) [Ertoz et al., 2002; Houle et al., 2010; Houle, 2003; Jarvis and Patrick, 1973] are a common way to counter the effects of curse of dimensionality. Can we use the affinity scores to redefine SNN?
- **Clustering to maximize affinity**. Can we define a new clustering method that maximizes the local stability of the points? Can we define an objective function along the lines of k-means to compute this clustering? Given a partition, how can we find new centers that maximize affinity? Can we still use our Voronoi notions that we discuss in Chapter 5 to determine how the centers navigate? Can we bound the convergence of this method? Can we develop faster algorithms to find high and low stability points directly to make this clustering method faster?
- **Dimensionality reduction and clustering**. Dimensionality reduction and clustering go hand in hand. Due to the "curse of dimensionality" and computational reasons, data are often projected to a lower-dimensional subspace to then cluster in this new space. It will be immensely useful to gain a deeper understanding of how various dimensionality algorithms and clustering methods are related. For instance, can we use the affinity scores that we define in Chapter 5 to build a robust dimensionality reduction methods that can find the best subspace to work with for purposes of clustering?
- **Defining different influence regions of points and clusters**. Can we define other ways of capturing how a cluster influences a point and vice versa to make the notion of affinity more robust? Can we use other interpolation techniques described by Bobach and Umlauf [2006] to understand other ways in which clusters interact? Can we compute this interpolation for kernels? Can we do it on manifolds? Can we work in other non Euclidean spaces?
- **Accountability in data mining**. As data mining results become more personalized,

there is an increasing need for the data miner to explain the results to the user. Can we setup a proof of knowledge system via an interactive proof system that will allow the user acting as the verifier to validate the clustering results? What are the privacy concerns in doing this?

- **Differential privacy and anonymity**. Can we define an ε-differential privacy [Dwork, 2008; Inan et al., 2007; Vaidya and Clifton, 2003; Zhang et al., 2012] setting for clustering using the affinity scores? Can we also describe k-anonymity [Aggarwal et al., 2010; Byun et al., 2007; Lin and Wei, 2008] for clustering by computing column and row leverage scores [Drineas and Mahoney, 2010; Mahoney and Drineas, 2009; Mahoney et al., 2012] using the affinity model described in Chapter 5?

REFERENCES

AERTS, S., LAMBRECHTS, D., MAITY, S., VAN LOO, P., COESSENS, B., DE SMET, F., TRANCHEVENT, L.-C., DE MOOR, B., MARYNEN, P., HASSAN, B., CARMELIET, P., AND MOREAU, Y. 2006. Gene prioritization through genomic data fusion. *Nature Biotechnology 24;* 5, 537–544.

AGGARWAL, C. 2009. A framework for clustering massive-domain data streams. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 102–113.

AGGARWAL, C., HAN, J., WANG, J., AND YU, P. 2007. On clustering massive data streams: a summarization paradigm. In *Data Streams*, A. K. Elmagarmid and C. C. Aggarwal, Eds. Advances in Database Systems Series, vol. 31. Springer US, Boston, MA, Chapter 2, 9–38.

AGGARWAL, G., PANIGRAHY, R., FEDER, T., THOMAS, D., KENTHAPADI, K., KHULLER, S., AND ZHU, A. 2010. Achieving anonymity via clustering. *ACM Trans. Algorithms 6;* 3, 49:1–49:19.

ALDRICH, J. 1997. R. A. Fisher and the making of maximum likelihood 1912–1922. *Statist. Sci. 12;* 3, 162–176.

ALLISON, D. B., CUI, X., PAGE, G. P., AND SABRIPOUR, M. 2006. Microarray data analysis: from disarray to consolidation and consensus. *Nature Reviews Genetics 7*, 55–65.

ANSARI, M. H., FILLMORE, N., AND COEN, M. H. 2010. Incorporating spatial similarity into ensemble clustering. *Proc. 1st Intl. Workshop on Discovering, Summarizing, and Using Multiple Clusterings (KDD Multiclust).*

ARONSZAJN, N. 1950. Theory of reproducing kernels. *Trans. American Math. Society 68*, 337 – 404.

ARTHUR, D. AND VASSILVITSKII, S. 2006. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 153–164.

ARTHUR, D. AND VASSILVITSKII, S. 2007. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1027–1035.

ASHARAF, S. AND MURTY, M. N. 2003. An adaptive rough fuzzy single pass algorithm for clustering large data sets. *Pattern Recognition 36;* 12, 3015–3018.

AURENHAMMER, F. 1987. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing 16;* 1, 78–96.

AYAD, H. G. AND KAMEL, M. S. 2010. On voting-based consensus of cluster ensembles. *Pattern Recogn. 43*, 1943–1953.

BACHE, K. AND LICHMAN, M. 2013. UCI machine learning repository [http://archive.ics.uci.edu/ml]. University of California, Irvine, School of Information and Computer Sciences.

BAE, E. AND BAILEY, J. 2006. COALA: a novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the Sixth International Conference on Data Mining*. ICDM '06. IEEE Computer Society, Washington, DC, USA, 53–62.

BAE, E., BAILEY, J., AND DONG, G. 2006. Clustering similarity comparison using density profiles. In *Australian Conference on Artificial Intelligence*, A. Sattar and B. H. Kang, Eds. Lecture Notes in Computer Science Series, vol. 4304. Springer, 342–351.

BAE, E., BAILEY, J., AND DONG, G. 2010. A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. *Data Min. Knowl. Discov. 21;* 3, 427–471.

BAEZA-YATES, R. A. AND RIBEIRO-NETO, B. 1999. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

BALCAN, M.-F., BLUM, A., FINE, S., AND MANSOUR, Y. 2012. Distributed learning, communication complexity and privacy. *Journal of Machine Learning Research - Proceedings Track 23*, 26.1–26.22.

BALCAN, M.-F., BLUM, A., AND YANG, K. 2004. Co-training and expansion: towards bridging theory and practice. In *Advances in Neural Information Processing Systems 17.* MIT Press, Cambridge, MA, 89–96.

BALCAN, M.-F. AND FELDMAN, V. 2013. Statistical active learning algorithms. *CoRR abs/1307.3102.*

BANDYOPADHYAY, S., GIANNELLA, C., MAULIK, U., KARGUPTA, H., LIU, K., AND DATTA, S. 2006. Clustering distributed data streams in peer-to-peer environments. *Inf. Sci. 176;* 14, 1952–1985.

BARBAR, D. AND CHEN, P. 2003. Using self-similarity to cluster large data sets. *Data Mining and Knowledge Discovery 7*, 123–152. 10.1023/A:1022493416690.

BELKIN, M. AND NIYOGI, P. 2004. Semi-supervised learning on riemannian manifolds. *Machine learning 56;* 1-3, 209–239.

BEN-DAVID, S. AND ACKERMAN, M. 2008. Measures of clustering quality: a working set of axioms for clustering. In *NIPS*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. MIT Press, 121–128.

BEN-DAVID, S., VON LUXBURG, U., AND PÁL, D. 2006. A sober look at clustering stability. In *Proceedings of the 19th annual conference on Learning Theory*. COLT'06. Springer-Verlag, Berlin, Heidelberg, 5–19.

BEN-HUR, A., ELISSEEFF, A., AND GUYON, I. 2002. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*. 6–17.

BENEDICT, C., GEISLER, M., TRYGG, J., HUNER, N., AND HURRY, V. 2006. Consensus by democracy. Using meta-analyses of microarray and genomic data to model the cold acclimation signaling pathway in Arabidopsis. *Plant Physiol 141;* 4, 1219–1232.

BERKHIN, P. 2006. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, J. Kogan, C. Nicholas, and M. Teboulle, Eds. Springer Berlin Heidelberg, 25–71.

BERLINET, A. AND THOMAS-AGNAN, C. 2004. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Netherlands.

BEZDEK, J. AND PAL, N. 1998. Some new indexes of cluster validity. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 28;* 3, 301 –315.

BEZDEK, J. C. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers.

BHADURI, K. 2008. Efficient local algorithms for distributed data mining in large scale peer to peer environments: a deterministic approach. Ph.D. thesis, University of Maryland at Baltimore County, Catonsville, MD, USA. Adviser-Kargupta, Hillol.

BLUM, A. AND MITCHELL, T. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. ACM, 92–100.

BOBACH, T. AND UMLAUF, G. 2006. Natural neighbor interpolation and order of continuity. In *GI Lecture Notes in Informatics, Visualization of Large and Unstructured Data Sets*, H. Hagen, A. Kerren, and P. Dannenmann, Eds. 68–86.

BONIZZONI, P., DELLA VEDOVA, G., DONDI, R., AND JIANG, T. 2008. On the approximation of correlation clustering and consensus clustering. *J. Comput. Syst. Sci. 74*, 671–696.

BONNER, R. E. 1964. On some clustering techniques. *IBM J. Res. Dev. 8*, 22–32.

BOULIS, C. AND OSTENDORF, M. 2004. Combining multiple clustering systems. In *8th European Conference on Principles and Practice of Knowledge Discovery in Database (PKDD), LNAI 3202*. 63–74.

BREGMAN, L. M. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics 7;* 3, 200–217.

BYUN, J.-W., KAMRA, A., BERTINO, E., AND LI, N. 2007. Efficient k-anonymization using clustering techniques. In *Advances in Databases: Concepts, Systems and Applications*, R. Kotagiri, P. Krishna, M. Mohania, and E. Nantajeewarawat, Eds. Lecture Notes in Computer Science Series, vol. 4443. Springer Berlin Heidelberg, 188–200.

CARUANA, R., ELHAWARY, M. F., NGUYEN, N., AND SMITH, C. 2006. Meta clustering. In *ICDM*. IEEE Computer Society, 107–118.

CHANG, J.-W. 2003. An efficient cell-based clustering method for handling large, high-dimensional data. In *Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. PAKDD'03. Springer-Verlag, Berlin, Heidelberg, 295–300.

CHAZELLE, B. 1992. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM Journal on Computing 21;* 4, 671–696.

CHEN, M., CHEN, Y., AND WEINBERGER, K. Q. 2011. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 953–960.

CHEW, L. P., GOODRICH, M. T., HUTTENLOCHER, D. P., KEDEM, K., AND A ND DINA KRAVETS, J. M. K. 1997. Geometric pattern matching under Euclidian motion. *Comp. Geom.: The. and App. 7*, 113–124.

COEN, M., ANSARI, H., AND FILLMORE, N. 2010. Comparing clusterings in space. In *ICML*.

COLEMAN, T. AND WIRTH, A. 2010. A polynomial time approximation scheme for k-consensus clustering. In *SODA*. 729–740.

COLLOBERT, R., SINZ, F., WESTON, J., AND BOTTOU, L. 2006a. Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 201–208.

COLLOBERT, R., SINZ, F. H., WESTON, J., AND BOTTOU, L. 2006b. Large scale transductive svms. *Journal of Machine Learning Research 7*, 1687–1712.

COSTA, D. AND VENTURINI, G. 2007. A visual and interactive data exploration method for large data sets and clustering. In *Proceedings of the 3rd International Conference on Advanced Data Mining and Applications*. ADMA '07. Springer-Verlag, Berlin, Heidelberg, 553–561.

DANG, X. H. AND BAILEY, J. 2010a. Generation of alternative clusterings using the cami approach. In *SDM*. SIAM, 118–129.

DANG, X. H. AND BAILEY, J. 2010b. A hierarchical information theoretic technique for the discovery of non linear alternative clusterings. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. ACM, New York, NY, USA, 573–582.

DAS, S., ABRAHAM, A., AND KONAR, A. 2009. *Metaheuristic Clustering* 1st Ed. Springer Publishing Company, Incorporated.

DASGUPTA, S. 2010. Active learning theory. In *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer, 14–19.

DATTA, S., GIANNELLA, C., AND KARGUPTA, H. 2006. K-means clustering over a large, dynamic network. In *Proceedings of the Sixth SIAM International Conference on Data Mining, Bethesda, MD, USA*.

DAVE, R. N. 1996. Validating fuzzy partitions obtained through c-shells clustering. *Pattern Recogn. Lett. 17*, 613–623.

DAVIDSON, I. 2000. Minimum message length clustering using gibbs sampling. In *Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*. Morgan Kaufmann, San Francisco, CA, 160–167.

DAVIDSON, I. AND QI, Z. 2008. Finding alternative clusterings using constraints. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, 773–778.

DAY, W. 1986. Foreword: comparison and consensus of classifications. *Journal of Classification 3;* 2, 183–185.

DE BERG, M., CHEONG, O., VAN KREVELD, M., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications*. Springer.

DE BIE, T., TRANCHEVENT, L.-C., VAN OEFFELEN, L. M. M., AND MOREAU, Y. 2007. Kernel-based data fusion for gene prioritization. *Bioinformatics 23;* 13, i125–i132.

DEAN, J. AND GHEMAWAT, S. 2008. Mapreduce: simplified data processing on large clusters. *Commun. ACM 51*, 107–113.

DHILLON, I. AND MODHA, D. 2000. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*. 245–260.

DING, C. AND LI, T. 2007. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. ACM, New York, NY, USA, 521–528.

DRINEAS, P. AND MAHONEY, M. W. 2010. Effective resistances, statistical leverage, and applications to linear equation solving. *CoRR abs/1005.3097*.

DUBES, R. C. AND JAIN, A. K. 1976. Clustering techniques: the user's dilemma. *Pattern Recognition 8;* 4, 247–260.

DUNN, J. C. 1974. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics 4*, 95–104.

DURRLEMAN, S., PENNEC, X., TROUVÉ, A., AND AYACHE, N. 2008. Sparse approximation of currents for statistics on curves and surfaces. In *MICCAI*. 390–398.

DWORK, C. 2008. Differential privacy: a survey of results. In *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*. TAMC'08. Springer-Verlag, Berlin, Heidelberg, 1–19.

ELISSEEFF, A., EVGENIOU, T., AND PONTIL, M. 2006. Stability of randomized learning algorithms. *Journal of Machine Learning Research 6;* 1, 55.

ERIKSSON, B., DASARATHY, G., SINGH, A., AND NOWAK, R. 2011. Active clustering: robust and efficient hierarchical clustering using adaptively selected similarities. *arXiv preprint arXiv:1102.3887.*

ERTOZ, L., STEINBACH, M., AND KUMAR, V. 2002. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining.*

ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD.* 226–231.

FERN, X. Z. AND BRODLEY, C. E. 2003. Random projection for high dimensional data clustering: A cluster ensemble approach. In *ICML.* 186–193.

FILKOV, V. AND SKIENA, S. 2003. Integrating microarray data by consensus clustering. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on.* 418 – 426.

FILKOV, V. AND SKIENA, S. 2004. Heterogeneous data integration with the consensus clustering formalism. In *Proceedings of Data Integration in the Life Sciences.* Springer, 110–123.

FORMAN, G. AND ZHANG, B. 2000. Distributed data clustering can be efficient and exact. *SIGKDD Explor. Newsl. 2,* 34–38.

FOWLKES, E. AND MALLOWS, C. 1983. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association 78;* 383, 553–569.

FRALEY, C. AND RAFTERY, A. E. 1998. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal 41;* 8, 578–588.

FRANK, A. AND ASUNCION, A. 2010. UCI machine learning repository [http://archive.ics.uci.edu/ml]. University of California, Irvine, School of Information and Computer Sciences.

FRED, A. AND JAIN, A. 2005. Combining multiple clusterings using evidence accumulation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27;* 6, 835 – 850.

GÄRTNER, T. 2002. Exponential and geometric kernels for graphs. In *NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data.*

GEORGHIADES, A. S., BELHUMEUR, P. N., AND KRIEGMAN, D. J. 2001. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell. 23;* 6, 643–660.

GHAEMI, R., BIN SULAIMAN, M. N., IBRAHIM, H., AND MUSTAPHA, N. World Academy of Science, Engineering and Technology 50, 2009. A survey: clustering ensembles techniques.

GIANCARLO, R., SCATURRO, D., AND UTRO, F. 2008. Computational cluster validation for microarray data analysis: experimental assessment of clest, consensus clustering, figure of merit, gap statistics and model explorer. *BMC Bioinformatics 9;* 1, 462.

GIONIS, A., MANNILA, H., AND TSAPARAS, P. 2007. Clustering aggregation. *ACM Trans. Knowl. Discov. Data 1;* 1.

GIVENS, C. R. AND SHORTT, R. M. 1984. A class of wasserstein metrics for probability distributions. *Michigan Math Journal 31*, 231–240.

GLAUNÈS, J. AND JOSHI, S. 2006. Template estimation form unlabeled point set data and surfaces for computational anatomy. In *Proc. of the International Workshop on the Mathematical Foundations of Computational Anatomy (MFCA-2006)*, X. Pennec and S. Joshi, Eds. 29–39.

GODER, A. AND FILKOV, V. 2008. Consensus clustering algorithms: comparison and refinement. In *ALENEX*. 109–117.

GOIL, S., GOIL, S., NAGESH, H., NAGESH, H., CHOUDHARY, A., AND CHOUDHARY, A. 1999. MAFIA: efficient and scalable subspace clustering for very large data sets. Tech. rep., Center for Parallel and Distributed Computing, Northwestern University.

GONDEK, D. AND HOFMANN, T. 2004. Non-redundant data clustering. In *ICDM*. IEEE Computer Society, 75–82.

GONZALEZ, T. F. 1985. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci. 38*, 293–306.

GRETTON, A., BORGWARDT, K., RASCH, M., SCHÖLKOPF, B., AND SMOLA, A. 2008. A kernel method for the two-sample-problem. *JMLR 1*, 1–10.

GROTKJAER, T., WINTHER, O., REGENBERG, B., NIELSEN, J., AND HANSEN, L. K. K. 2006. Robust multi-scale clustering of large DNA microarray datasets with the consensus algorithm. *Bioinformatics (Oxford, England) 22;* 1, 58–67.

GUHA, S., RASTOGI, R., AND SHIM, K. 2001. Cure: an efficient clustering algorithm for large databases* 1. *Information Systems 26;* 1, 35–58.

HALKIDI, M., BATISTAKIS, Y., AND VAZIRGIANNIS, M. 2001. On clustering validation techniques. *J. Intell. Inf Syst. 17;* 2-3, 107–145.

HALKIDI, M. AND VAZIRGIANNIS, M. 2001. Clustering validity assessment: finding the optimal partitioning of a data set. In *ICDM*, N. Cercone, T. Y. Lin, and X. Wu, Eds. IEEE Computer Society, 187–194.

HALKIDI, M., VAZIRGIANNIS, M., AND BATISTAKIS, Y. 2000. Quality scheme assessment in the clustering process. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*. PKDD '00. Springer-Verlag, London, UK, 265–276.

HAR-PELED, S. 2011. *Geometric Approximation Algorithms*. Vol. 173. American Mathematical Society.

HAR-PELED, S. AND SADRI, B. 2005. How fast is the k-means method? *Algorithmica 41*, 185–202.

HARTIGAN, J. A. 1975. *Clustering Algorithms* 99th Ed. John Wiley & Sons, Inc., New York, NY, USA.

HARTIGAN, J. A. AND WONG, M. A. 1979. Algorithm as136: a k-means clustering algorithm. *Applied Statistics 28*, 100–108.

HASTINGS, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*, 97–109.

HO, C.-H., TSAI, M.-H., AND LIN, C.-J. 2011. Active learning and experimental design with svms. *Journal of Machine Learning Research - Proceedings Track 16*, 71–84.

HOFF, P. D. 2009. *A First Course in Bayesian Statistical Methods*. Springer.

HOFMANN, T. AND BUHMANN, J. M. 1998. Active data clustering. *Advances in Neural Information Processing Systems*, 528–534.

HOULE, M., KRIEGEL, H.-P., KRÖGER, P., SCHUBERT, E., AND ZIMEK, A. 2010. Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management*, M. Gertz and B. Ludäscher, Eds. Lecture Notes in Computer Science Series, vol. 6187. Springer Berlin / Heidelberg, Berlin, Heidelberg, Chapter 34, 482–500.

HOULE, M. E. 2003. Navigating massive data sets via local clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. ACM, New York, NY, USA, 547–552.

HUANG, M. AND ZHU, X. 2007. Combining convolution kernels defined on heterogeneous sub-structures. *Advances in Knowledge Discovery and Data Mining*, 539–546.

HUANG, Z. 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery 2*, 283–304. 10.1023/A:1009769707641.

HUISKES, M. J. AND LEW, M. S. 2008. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. ACM, New York, NY, USA.

IBM, ZIKOPOULOS, P., AND EATON, C. 2011. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.* 1st Ed. McGraw-Hill Osborne Media.

INAN, A., KAYA, S. V., SAYGIN, Y., SAVAS, E., HINTOGLU, A. A., AND LEVI, A. 2007. Privacy preserving clustering on horizontally partitioned data. *Data Knowl. Eng. 63;* 3, 646–666.

INDYK, P. AND THAPER, N. 2003. Fast image retrieval via embeddings. In *Intr. Workshop on Statistical and Computational Theories of Vision (at ICCV)*.

JAIN, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*.

JAIN, A. K. AND DUBES, R. C. 1988. *Algorithms for Clustering Data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. 1999. Data clustering: a review. *ACM Comput. Surv. 31*, 264–323.

JAIN, P., MEKA, R., AND DHILLON, I. S. 2008. Simultaneous unsupervised learning of disparate clusterings. *Stat. Anal. Data Min. 1*, 195–210.

JANUZAJ, E., KRIEGEL, H.-P., AND PFEIFLE, M. 2004. Dbdc: density based distributed clustering. In *EDBT*. 88–105.

JARVIS, R. AND PATRICK, E. 1973. Clustering using a similarity measure based on shared near neighbors. *Computers, IEEE Transactions on C-22;* 11, 1025–1034.

JEGELKA, S., GRETTON, A., SCHÖLKOPF, B., SRIPERUMBUDUR, B. K., AND VON LUXBURG, U. 2009. Generalized clustering via kernel embeddings. In *32nd Annual German Conference on AI (KI 2009).*

JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*. Morgan Kaufmann Publishers, Inc., 200–209.

JOHNSON, E. L. AND KARGUPTA, H. 2000. Collective, hierarchical clustering from distributed, heterogeneous data. In *Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*. Springer-Verlag, London, UK, 221–244.

JOSHI, S., KOMMARAJU, R. V., PHILLIPS, J. M., AND VENKATASUBRAMANIAN, S. 2011. Comparing distributions and shapes using the kernel distance. In *Proceedings of the 27th Annual ACM Symposium on Computational Geometry*. SoCG '11. ACM, New York, NY, USA, 47–56.

KASHIMA, H., KATO, T., YAMANISHI, Y., SUGIYAMA, M., AND TSUDA, K. 2009. Link propagation: a fast semisupervised learning algorithm for link prediction. *SDM09*, 1099–1110.

KEIM, D. A. AND HINNEBURG, A. 1999. Clustering techniques for large data sets from the past to the future. In *Tutorial Notes of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '99. ACM, New York, NY, USA, 141–181.

KETTENRING, J. R. 2009. Massive datasets. *Wiley Interdisciplinary Reviews: Computational Statistics 1;* 1, 25–32.

KUHN, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly 2*, 83–97.

LADERAS, T. AND MCWEENEY, S. 2007. Consensus framework for exploring microarray data using multiple clustering methods. *OMICS: A Journal of Integrative Biology 11;* 1, 116–128.

LANEY, D. 2001. 3D data management: controlling data volume, velocity, and variety. Tech. rep., META Group. February.

LENNES, N. J. 1911. Theorems on the simple finite polygon and polyhedron. *American Journal of Mathematics 33;* 1/4, 37–62.

LI, Y., LONG, P. M., AND SRINIVASAN, A. 2001. Improved bounds on the sample complexity of learning. *J. Comput. Syst. Sci. 62;* 3, 516–527.

LIAO, D.-P., JIANG, B., WEI, X.-Z., LI, X., AND ZHUANG, Z.-W. 2007. Fast learning algorithm with progressive transductive support vector machine. *System Engineering and Electronics 29;* 1, 88–91.

LIN, J.-L. AND WEI, M.-C. 2008. An efficient clustering method for k-anonymization. In *Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society*. PAIS '08. ACM, New York, NY, USA, 46–50.

LIU, X., YU, S., MOREAU, Y., MOOR, B. D., GLÄNZEL, W., AND JANSSENS, F. A. L. 2009. Hybrid clustering of text mining and bibliometrics applied to journal sets. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, Sparks, Nevada, USA*. 49–60.

LIU, Y., LI, Z., XIONG, H., GAO, X., AND WU, J. 2010. Understanding of internal clustering validation measures. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. 911–916.

LODHI, H., SAUNDERS, C., SHAWE-TAYLOR, J., CRISTIANINI, N., AND WATKINS, C. 2002. Text classification using string kernels. *JMLR 2*, 444.

LOVÁSZ, L. 1999. Hit-and-run mixes fast. *Mathematical Programming 86;* 3, 443–461.

LUXBURG, U. V. 2009. Clustering stability: an overview. *Foundations and Trends in Machine Learning 2;* 3, 235–274.

LV, Z., HU, Y., ZHONG, H., WU, J., LI, B., AND ZHAO, H. 2010. Parallel k-means clustering of remote sensing images based on mapreduce. In *Proceedings of the 2010 International Conference on Web Information Systems and Mining*. WISM'10. Springer-Verlag, Berlin, Heidelberg, 162–170.

MACKAY, D. J. C. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

MAHONEY, M. W. AND DRINEAS, P. 2009. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences 106;* 3, 697–702.

MAHONEY, M. W., DRINEAS, P., MAGDON-ISMAIL, M., AND WOODRUFF, D. P. 2012. Fast approximation of matrix coherence and statistical leverage. In *ICML*. icml.cc / Omnipress.

MANN, G. S. AND MCCALLUM, A. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 593–600.

MEILĂ, M. 2007. Comparing clusterings–an information based distance. *J. Multivar. Anal. 98*, 873–895.

METROPOLIS, N., ROSENTBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*.

MILLIGAN, G. AND COOPER, M. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika 50*, 159–179. 10.1007/BF02294245.

MIRKIN, B. G. AND CHERNY, L. B. 1970. Measurement of the distance between distinct partitions of a finite set of objects. *Automation & Remote Control 31*, 786.

MONTI, S., TAMAYO, P., MESIROV, J., AND GOLUB, T. 2003. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning 52*, 91–118. 10.1023/A:1023949509487.

MÜLLER, A. 1997. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability 29;* 2, 429–443.

MURINO, L., ANGELINI, C., BIFULCO, I., FEIS, I. D., RAICONI, G., AND TAGLIAFERRI, R. 2009. Multiple clustering solutions analysis through least-squares consensus algorithms. In *CIBB*. 215–227.

MURTAGH, F. 1999. Clustering in massive data sets. In *Handbook of Massive Data Sets*. Kluwer Academic Publishers, 501–543.

NAGESH, H., GOIL, S., AND CHOUDHARY, A. 2001. Adaptive grids for clustering massive data sets. In *1st SIAM International Conference Proceedings on Data Mining*.

NAGESH, H. S., CHOUDHARY, A., AND GOIL, S. 2000. A scalable parallel subspace clustering algorithm for massive data sets. In *Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing*. ICPP '00. IEEE Computer Society, Washington, DC, USA, 477–.

NEUMANN, D. AND NORTON, V. 1986. Clustering and isolation in the consensus problem for partitions. *Journal of Classification 3;* 2, 281–297.

NGUYEN, N. AND CARUANA, R. 2007. Consensus clusterings. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, 607–612.

NGUYEN, X. V. AND EPPS, J. 2010. Mincentropy: a novel information theoretic approach for the generation of alternative clusterings. In *ICDM*, G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, Eds. IEEE Computer Society, 521–530.

NIU, D., DY, J. G., AND JORDAN, M. I. 2010. Multiple non-redundant spectral clustering views. In *ICML'10*. 831–838.

OLKEN, F. 1993. Random sampling from databases. Ph.D. thesis, University of California, Berkeley.

OLMAN, V., MAO, F., WU, H., AND XU, Y. 2009. Parallel clustering algorithm for large data sets with applications in bioinformatics. *IEEE/ACM Trans. Comput. Biol. Bioinformatics 6*, 344–352.

OSTROVSKY, R., RABANI, Y., SCHULMAN, L. J., AND SWAMY, C. 2006. The effectiveness of Lloyd-type methods for the k-means problem. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 165–176.

OTOO, E. J., SHOSHANI, A., AND HWANG, S.-W. 2001. Clustering high dimensional massive scientific datasets. *J. Intell. Inf. Syst. 17*, 147–168.

PETROVIC, S. 2006. A comparison between the silhouette index and the davies-bouldin index in labeling ids clusters. *Proceedings of the 11th Nordic Workshop on Secure IT-systems, NORDSEC 2006*, 53–64.

PLATT, J. C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 61–74.

PROKHOROV, Y. V. 1956. Convergence of random processes and limit theorems in probability theory. *Theory of Probability and its Applications 1;* 2, 157–214.

QI, Z. AND DAVIDSON, I. 2009. A principled and flexible framework for finding alternative clusterings. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. ACM, New York, NY, USA, 717–726.

QIAN, W., GONG, X., AND ZHOU, A. 2003. Clustering in very large databases based on distance and density. *J. Comput. Sci. Technol. 18*, 67–76.

RAHIMI, A. AND RECHT, B. 2007. Random features for large-scale kernel machines. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*. MIT Press.

RAMAN, P., PHILLIPS, J. M., AND VENKATASUBRAMANIAN, S. 2011. Spatially-aware comparison and consensus for clusterings. In *SDM*. SIAM / Omnipress, 307–318.

RAND, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66;* 336, 846–850.

ROBERTS, G. O., GELMAN, A., AND GILKS, W. R. 1997. Weak convergence and optimal scaling of random walk metropolis algorithms. *Annals of Applied Probability 7*, 110–120.

ROUSSEEUW, P. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math. 20;* 1, 53–65.

SANDER, J., ESTER, M., KRIEGEL, H.-P., AND XU, X. 1998. Density-based clustering in spatial databases: the algorithm gdbscan and its applications. *Data Min. Knowl. Discov. 2*, 169–194.

SASSI, M. AND GRISSA, A. 2009. Clustering large data sets based on data compression technique and weighted quality measures. In *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*. 396 –402.

SCHÖLKOPF, B. AND SMOLA, A. J. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA.

SETTLES, B. 2011. Closing the loop: fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1467–1478.

SETTLES, B. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 6;* 1, 1–114.

SHIRDHONKAR, S. AND JACOBS, D. 2008. Approximate Earth mover's distance in linear time. In *CVPR*.

SIBSON, R. 1980. A vector identity for the dirichlet tessellation. *Mathematical Proceedings of the Cambridge Philosophical Society 87;* 1, 151–155.

SIBSON, R. 1981. A brief description of natural neighbour interpolation. In *Interpolating Multivariate Data*. John Wiley & Sons, New York, USA, Chapter 2, 21–36.

SINDHWANI, V. AND KEERTHI, S. S. 2007. Newton methods for fast solution of semi-supervised linear svms. *Large Scale Kernel Machines*, 155–174.

SMITH, R. L. 1984. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research 32;* 6, 1296–1308.

SMOLA, A., GRETTON, A., SONG, L., AND SCHÖLKOPF, B. 2007. A Hilbert space embedding for distributions. In *Algorithmic Learning Theory*. Springer, 13–31.

SRIPERUMBUDUR, B. K., GRETTON, A., FUKUMIZU, K., SCHÖLKOPF, B., AND LANCKRIET, G. R. 2010. Hilbert space embeddings and metrics on probability measures. *JMLR 11*, 1517–1561.

STREHL, A. Clusterpack matlab toolbox. http://www.ideal.ece.utexas.edu/~strehl/soft.html.

STREHL, A. AND GHOSH, J. 2003. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res. 3*, 583–617.

SUGAR, C. A. AND JAMES, G. M. 2003. Finding the number of clusters in a dataset: an information-theoretic approach. *Journal of the American Statistical Association 98;* 463, pp. 750–763.

SUZUKI, J. AND ISOZAKI, H. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. *Proceedings of ACL-08: HLT*, 665–673.

TALAGRAND, M. 1994. Sharper bounds for gaussian and empirical processes. *The Annals of Probability 22;* 1, pp. 28–76.

TAN, P.-N., STEINBACH, M., AND KUMAR, V. 2005. *Introduction to Data Mining, (First Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

THEODORIDIS, S. AND KOUTROUMBAS, K. 2006. *Pattern Recognition.* Elsevier.

TIBSHIRANI, R., WALTHER, G., AND HASTIE, T. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63;* 2, 411–423.

TOPCHY, A. P., JAIN, A. K., AND PUNCH, W. F. 2003. Combining multiple weak clusterings. In *ICDM*. IEEE Computer Society, 331–338.

TOUSSAINT, G. T. 1985. A simple linear algorithm for intersecting convex polygons. *The Visual Computer 1;* 2, 118–123.

TREVOR, H., ROBERT, T., AND JEROME, F. 2001. The elements of statistical learning: data mining, inference and prediction. *New York: Springer-Verlag 1;* 8, 371–406.

TSANGARIS, M. M. AND NAUGHTON, J. F. 1992. On the performance of object clustering techniques. In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*. SIGMOD '92. ACM, New York, NY, USA, 144–153.

VAIDYA, J. AND CLIFTON, C. 2003. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. ACM, New York, NY, USA, 206–215.

VAILLANT, M. AND GLAUNÈS, J. 2005. Surface matching via currents. In *Proc. Information Processing in Medical Imaging*. Vol. 19. 381–92.

VINH, N. X. AND EPPS, J. 2009. A novel approach for automatic number of clusters detection in microarray data based on consensus clustering. In *Bioinformatics and Bio-Engineering, 2009. BIBE '09. Ninth IEEE International Conference on*. 84 –91.

VURAL, V., FUNG, G., DY, J. G., AND RAO, B. 2008. Fast semi-supervised svm classifiers using a priori metric information. *Optimisation Methods and Software 23;* 4, 521–532.

WAGNER, S. AND WAGNER, D. 2007. Comparing clusterings - an overview. Tech. Rep. 2006-04, ITI Wagner, Informatics, Universität Karlsruhe.

WAHBA, G. 1990. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics.

WANG, F., YANG, C., LIN, Z., LI, Y., AND YUAN, Y. 2010. Hybrid sampling on mutual information entropy-based clustering ensembles for optimizations. *Neurocomputing 73;* 7-9, 1457–1464.

WANG, H., WANG, W., YANG, J., AND YU, P. S. 2002. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. SIGMOD '02. ACM, New York, NY, USA, 394–405.

WURST, M., MORIK, K., AND MIERSWA, I. 2006. Localized alternative cluster ensembles for collaborative structuring. In *ECML*. 485–496.

XU, R. AND WUNSCH, D. 2009. *Clustering*. IEEE Press Series on Computational Intelligence. Wiley-IEEE Press.

XU, R., WUNSCH, D., ET AL. 2005. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on 16;* 3, 645–678.

YE, J., CHEN, K., WU, T., LI, J., ZHAO, Z., PATEL, R., BAE, M., JANARDAN, R., LIU, H., ALEXANDER, G., AND REIMAN, E. 2008. Heterogeneous data fusion for alzheimer's disease study. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. ACM, New York, NY, USA, 1025–1033.

YU, S., DE MOOR, B., AND MOREAU, Y. 2008. Clustering by heterogeneous data fusion: framework and applications. In *Learning from Multiple Sources Workshop (NIPS 2008)*.

YUILLE, A. L. AND RANGARAJAN, A. 2001. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. 1033–1040.

ZHANG, J., ZHANG, Z., XIAO, X., YANG, Y., AND WINSLETT, M. 2012. Functional mechanism: regression analysis under differential privacy. *PVLDB 5;* 11, 1364–1375.

ZHANG, T., RAMAKRISHNAN, R., AND LIVNY, M. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD Record 25;* 2, 103–114.

ZHANG, Y. AND LI, T. 2011. Consensus clustering + meta clustering = multiple consensus clustering. In *FLAIRS Conference*, R. C. Murray and P. M. McCarthy, Eds. AAAI Press, Palm Beach, USA.

ZHAO, W., MA, H., AND HE, Q. 2009. Parallel k-means clustering based on mapreduce. In *Proceedings of the 1st International Conference on Cloud Computing*. CloudCom '09. Springer-Verlag, Berlin, Heidelberg, 674–679.

ZHOU, B., CHEUNG, D. W.-L., AND KAO, B. 1999. A fast algorithm for density-based clustering in large database. In *Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*. PAKDD '99. Springer-Verlag, London, UK, 338–349.

ZHOU, D., LI, J., AND ZHA, H. 2005. A new mallows distance based metric for comparing clusterings. In *Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. ACM, New York, NY, USA, 1028–1035.

ZHOU, D., RAO, W., AND LV, F. 2010. A multi-agent distributed data mining model based on algorithm analysis and task prediction. In *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on*. 1–4.

ZHU, X. 2005. Semi-Supervised Learning Literature Survey. Tech. rep., Computer Sciences, University of Wisconsin-Madison.

DISSEMINATION OF THIS WORK

• **Spatially-Aware Comparison and Consensus for Clusterings**.

Parasaran Raman, Jeff M. Phillips, Suresh Venkatasubramanian.

Proceedings of the Eleventh SIAM International Conference on Data Mining (SDM), April 2011.

• **Generating a Diverse Set of High-Quality Clusterings**.

Jeff M. Phillips, Parasaran Raman, Suresh Venkatasubramanian.

Proceedings of the Second MultiClust Workshop: Discovering, Summarizing and Using Multiple Clusterings (held in conjunction with ECMLPKDD 2011), September 2011.

• **Power to the Points: Validating Data Memberships in Clusterings**.

Parasaran Raman, Suresh Venkatasubramanian.

Proceedings of the Thirteenth IEEE International Conference on Data Mining (ICDM), December 2013 (To appear).

• **Large Scale Transductive SVM via Adaptive Subsampling**.

Parasaran Raman, Suresh Venkatasubramanian, Kilian Q. Weinberger.

Submitted to 31st International Conference on Machine Learning (ICML) 2014. Under review.

• **Generating a Diverse Set of High-Quality Clusterings**.

Jeff M. Phillips, Parasaran Raman, Suresh Venkatasubramanian.

Submitted to ACM Transactions on Knowledge Discovery from Data (TKDD). Under review.